ADOBE® DIRECTOR® 11.5
Skriptwörterbuch

Adobe

Adobe® Director® 11.5 Skriptwörterbuch

Adobe Systems Incorporated, 345 Park Avenue, San Jose, California 95110, USA.

Inhalt

Kapitel 6: Medientypen

Kapitel 7: Skriptobjekte

Kapitel 8: 3D-Objekte

Kapitel 9: Konstanten

Kapitel 10: Ereignisse und Nachrichten

Kapitel 12: Methoden

Kapitel 13: Operatoren

Kapitel 14: Eigenschaften

Kapitel 1: Einführung

Diese Referenz bietet Erläuterungen und Anleitungen zur Skripterstellung in Adobe® Director® 11, sowie Beschreibungen und Beispiele der Skripterstellungs-APIs (Application Programming Interfaces, Anwendungsprogrammierschnittstellen), mit deren Hilfe Sie Skripts schreiben.

Über die Skripterstellungs-APIs greifen Sie skriptgesteuert auf die Funktionalität von Director zu, um Filme mit Interaktivität zu versehen. Mit diesen APIs können Sie für interaktive Funktionalität sorgen, die der der vordefinierten Verhalten im Lieferumfang von Director entspricht, und zusätzlich Funktionalität bereitstellen, die leistungsfähiger und vielgestaltiger als die der vordefinierten Verhalten ist.

Die vordefinierten Verhalten ermöglichen Ihnen das Hinzufügen grundlegender interaktiver Funktionalität zu einem Film, z. B. Bewegen des Abspielkopfes zu einer Bildnummer oder -markierung oder Vergrößern der Ansicht, wenn ein Benutzer auf ein Sprite klickt. Die vordefinierten Verhalten bieten ferner nicht interaktive Funktionalität, z. B. Sprite-Animation, Laden von Medien oder Bildansteuerung. Mithilfe der Skripterstellungs-APIs können Sie diese Funktionstypen erweitern und anpassen.

Zielgruppe

Diese Referenz richtet sich an Benutzer, die folgende Aufgaben ausführen möchten:

- Die aktuelle Funktionalität vordefinierter Verhalten mithilfe von Skript erweitern
- Filme mithilfe von Skript anstatt vordefinierter Verhalten um Funktionalität ergänzen
- Filmen im Vergleich zu vordefinierten Verhalten leistungsstärkere, vielschichtigere und besser angepasste Funktionen hinzufügen

Diese Referenz hat die Ansicht allen Benutzern, ob Neueinsteigern oder Fortgeschrittenen, sämtliche Informationen bereitzustellen, die für das Hinzufügen von Interaktivität zu Filmen mithilfe von Skript benötigt werden. Aus diesem Grund ist keine vorherige Skripterstellungserfahrung erforderlich, um in Director effektive Skripts zu schreiben.

Nehmen Sie sich jedoch unabhängig von Ihrem Erfahrungsgrad mit Director, Lingo, or JavaScript™-Syntax einige Minuten Zeit zum Durchlesen von „Grundlagen der Skripterstellung in Director“ auf Seite 4 und „Schreiben von Skripts in Director“ auf Seite 44, bevor Sie mit dem Erstellen von Skripts beginnen. Wie andere Produkte auch weist Director eine eindeutigen Satz mit Skripterstellungskonventionen, Datentypen usw. auf. Sie müssen mit diesen besonderen Merkmalen von Director vertraut sein, bevor Sie leistungsfähige Skripts schreiben können.

Neues bei der Skripterstellung in Director

Wenn Sie in früheren Versionen von Director Skripts geschrieben haben, werden Ihnen einige Neuerungen und wichtige Änderungen bei der Skripterstellung in dieser Version auffallen.

Reichhaltige Audiomöglichkeiten Dank des 5.1-Surroundklangs, Echtzeit-Mixing, Audioeffekten und DSP-Filtern hören sich Ihre Spiele so spannend an, wie sie aussehen. Weitere Informationen hierzu finden Sie unter Sound, Mixer und Audiofilter.

Reichhaltige Videomöglichkeiten Erweitern Sie das Multimediaerlebnis Ihrer Nutzer mit H.264-Videointegration für hochauflösende Videos im Vollbildmodus in Multimediaanwendungen und Spielen. Weitere Informationen hierzu finden Sie unter „MP4Media/FLV“ auf Seite 130.

Neuer Datentyp Zugriff auf Binärdaten und Manipulieren der Daten mit dem ByteArray-Datentyp. Weitere Informationen hierzu finden Sie unter „ByteArray“ auf Seite 155.

Erweitertes Physikmodul Unterstützung starrer Körper vom Typ „Dynamic Concave“. Weitere Informationen hierzu finden Sie unter „Festkörpermethoden“ auf Seite 1366.

Cross-Domain-Richtlinie Weitere Informationen hierzu finden Sie unter loadPolicyFile().

Erweiteres Textmodul. •Unterstützung für das Einbetten von OTF-Schriftarten (Open Type Font) unter Mac OS.

- Klassenbasierte Unterstützung für Unterschneidungen.
- Schriftarthinweise auf Darstellerebene. Weitere Informationen hierzu finden Sie unter hinting.

Erweiterte Unterstützung für Plattformen und Browser. •Unterstützung für Macintosh OS X 10.5 (Autorenwerkzeug und Runtime).

- Unterstützung für Firefox 3.0.

  Weitere Informationen hierzu finden Sie in der Readme-Datei.

Einschränkungen der Unicode-Unterstützung in Director

Adobe Director unterstützt die Skripterstellung in Unicode.

- Sprachen mit Schreibrichtung von rechts nach links werden nicht unterstützt.
- Xtra-Datei-E/A-Funktionen wie „readchar()“, „getLength()“ und „getPosition()“ funktionieren nur bei Eingabe von 1-Byte-Zeichen. Damit 2- bzw. 3-Byte-Unicode-Zeichen gelesen werden, lesen Sie die gesamte Datei über die „readFile()“-Methode in ein „String“-Objekt ein. Verwenden Sie anschließend die „char...of“-Methode, um das Unicode-Zeichen zu lesen.
- Komponenten von Director Physics unterstützen Unicode nicht.
- Unicode-Namen für HTTP-Pfade werden nicht unterstützt.
- Ein Skript-Xtra kann nicht mithilfe des Registrierungsschlüssels „kMoaMmDictType_MessageTable“ mit einer Unicode-Zeichenfolge benannt werden. Ebenfalls können Lingo-Funktionen nicht mithilfe von Skript-Xtras in Unicode bereitgestellt werden.
- Skript-Xtras stellen Lingo-Funktionen bereit. Diese von den Xtras bereitgestellten Funktionsnamen werden in Unicode nicht unterstützt.
- 3D-Modellnamen in Unicode werden nicht unterstützt.

  Das Skriptfenster enthält neben dem Skript-Editor einen Explorer-Bereich, der standardmäßig links neben dem Skript-Editor angezeigt wird. Sie können den Explorer-Bereich in der Wörterbuch- oder Skript-Browser-Ansicht anzeigen.

Neues in dieser Dokumentation

Wenn Sie in früheren Versionen von Director bereits Skripts geschrieben haben, werden Ihnen einige Änderungen bei der Dokumentation zur Skripterstellung in dieser Version auffallen. Die *Director-Skriptreferenz* ersetzt das *Lingo-Wörterbuch* im Lieferumfang früherer Versionen von Director und zeichnet sich zudem durch einen anderen Aufbau aus.

Im *Lingo-Wörterbuch* orientierten sich die Informationen zum Skripterstellungsmodell an der jeweiligen Funktion. Wenn Sie z. B. lernen wollten, wie mit Sprites in Skript gearbeitet wird, mussten Sie diese Informationen in einem der Abschnitte unter der Überschrift „Sprites“ nachschlagen, z. B. „Ziehen von Sprites“, „Sprite-Dimensionen“ usw. Darüber hinaus waren sämtliche Skripterstellung-APIs in einer einzelnen alphabetischen Liste enthalten, was bedeutete, dass alle Funktionen, Eigenschaften, Ereignisse usw. alphabetisch aufeinander folgten.

In der *Director-Skriptreferenz* orientieren sich die Informationen zum Skripterstellungsmodell am jeweiligen Objekt. Diese Strukturierung gibt die Organisation der tatsächlichen Skripterstellungsobjekte wieder, mit denen Sie in Skripts arbeiten. Wenn Sie z. B. wissen möchten, wie mit Sprites in Skript gearbeitet wird, müssen Sie den Abschnitt „Sprite“ im Kapitel „Core-Objekte in Director“ lesen.

Die Skripterstellungs-APIs sind weiter alphabetisch sortiert, jedoch nach API-Typ kategorisiert. Alle Methoden sind beispielsweise alphabetisch unter der Überschrift „Methoden“, alle Eigenschaften alphabetisch unter der Überschrift „Eigenschaften“ aufgeführt usw.

Suchen von Informationen zur Skripterstellung in Director

Die neu gestaltete *Director-Skriptreferenz* enthält die folgenden Themen:

Grundlagen der Skripterstellung in Director Enthält Informationen zu den grundlegenden Konzepten und Komponenten der Skripterstellung, die Sie beim Schreiben von Skripts in Director verwenden.

Schreiben von Skripts in Director Hier finden Sie Informationen zur Skripterstellungsumgebung von Director sowie anspruchsvollere Skripterstellungskonzepte und -techniken

Debuggen von Skripts in Director In diesem Kapitel finden Sie Informationen zur Problemsuche in Ihren Skripts, wenn diese nicht wie erwartet funktionieren.

Core-Objekte in Director Dieses Kapitel bietet eine Aufstellung der Objekte und APIs für den Zugriff auf die Hauptfunktionen in Director, z. B. die Director-Wiedergabe-Engine, Filmfenster, Sprites, Sounds usw.

Medientypen In diesem Kapitel finden Sie eine Liste der Medientypen und APIs für den Zugriff auf die Funktionalität der verschiedenen Medientypen in Director, z. B. RealMedia, DVD, Animiertes GIF usw., die Filmen als Darsteller hinzugefügt werden.

Skriptobjekte Dieses Kapitel stellt eine Liste der Skriptobjekte (auch als Xtra-Erweiterungen bezeichnet) und APIs bereit, mit denen Sie den Hauptfunktionsumfang von Director erweitern können. Xtra-Erweiterungen stellen Funktionen bereit, mit denen Sie zum Beispiel Filter importieren oder eine Internetverbindung herstellen können.

3D-Objekte Dieses Kapitel enthält eine Liste der Objekte, über die Sie einem Film 3D-Funktionalität hinzufügen.

Konstanten Hier finden Sie eine Liste der Konstanten, die in Director verfügbar sind.

Ereignisse und Nachrichten Hier finden Sie eine Liste der Ereignisse, die in Director verfügbar sind.

Schlüsselwörter Hier finden Sie eine Liste der Schlüsselwörter, die in Director verfügbar sind.

Methoden Hier finden Sie eine Liste der Methoden, die in Director verfügbar sind.

Operatoren Hier finden Sie eine Liste der Operatoren, die in Director verfügbar sind.

Eigenschaften Hier finden Sie eine Liste der Eigenschaften, die in Director verfügbar sind.

Kapitel 2: Grundlagen der Skripterstellung in Director

Wenn Sie noch nicht mit der Skripterstellung in Director® vertraut sind, sollten Sie sich Zeit für das Erlernen grundlegender Skripterstellungskonzepte nehmen, um zu verstehen, wie Skripts in Director geschrieben werden. Zu diesen Grundlagen zählen Definitionen wichtiger Begriffe, Syntaxregeln, verfügbare Datentypen und Informationen zu den Basiselementen der Skripterstellung in Director, z. B. Variablen, Arrays, Operatoren usw.

Skripttypen

Ein Director-Film kann vier Skripttypen enthalten: Verhalten, Filmskripts, Parent-Skripts und Skripts, die mit Darstellern verbunden sind. Verhalten, Filmskripten und Parent-Skripts werden im Besetzungsfenster als unabhängige Darsteller angezeigt. Ein einem Darsteller zugeordnetes Skript ist mit dem Darsteller im Besetzungsfenster verknüpft und wird nicht unabhängig angezeigt.

- **Verhalten** sind Skripts, die Sprites oder Bildern im Drehbuch zugeordnet sind, und werden als Sprite-Verhalten oder Bildverhalten bezeichnet. Die Piktogramme der Verhalten im Besetzungsfenster sind mit einem Verhaltenssymbol in der rechten unteren Ecke gekennzeichnet.

  In der *Director-Skriptreferenz* bezieht sich der Begriff *Verhalten* auf jedes Skript, das Sie einem Sprite oder Bild zuordnen und nicht auf die Verhalten in der Bibliothekspalette von Director. Weitere Informationen zu diesen in Director vordefinierten Verhalten finden Sie im Director-Benutzerhandbuch im Hilfefenster von Director.

  Alle Verhalten, die Sie in eine Besetzungsbibliothek aufgenommen haben, erscheinen im Popupmenü „Verhalten“ des Verhaltensinspektors. Andere Skripttypen werden in diesem Menü nicht angezeigt.

  Sie können dasselbe Verhalten mehreren Positionen im Drehbuch zuordnen. Wenn Sie ein Verhalten bearbeiten, wird die bearbeitete Version an allen Positionen im Drehbuch übernommen, denen das Verhalten zugeordnet ist.

- **Filmskripte** enthalten Prozeduren, die global oder auf Filmebene zur Verfügung stehen. Ereignisprozeduren in einem Filmskript können während der Filmwiedergabe von anderen Skripts im Film aufgerufen werden.

  Ein Filmskript ist am Filmskriptsymbol zu erkennen, das im Besetzungsfenster in der rechten unteren Ecke des Piktogramms erscheint.

  Filmskripte stehen dem gesamten Film zur Verfügung, ganz gleich, in welchem Bild sich der Film befindet oder welche Sprites der Benutzer interaktiv verwendet. Wenn ein Film in einem Fenster oder als verknüpfter Film abgespielt wird, steht das Filmskript nur dem eigenen Film zur Verfügung.

- **Parent-Skripts** sind spezielle Skripts, die Lingo-Code zur Erstellung von Child-Objekten enthalten. Sie können Parent-Skripts verwenden, um Skriptobjekte zu erzeugen, die sich ähnlich verhalten und sich dennoch unabhängig voneinander einsetzen lassen. Ein Parent-Skript ist am Parent-Skriptsymbol zu erkennen, das im Besetzungsfenster in der rechten unteren Ecke des Piktogramms erscheint.

  Weitere Informationen über den Gebrauch von Parent-Skripts und Child-Objekten finden Sie unter „Objektorientierte Programmierung mithilfe von Lingo-Syntax“ auf Seite 50.

  JavaScript arbeitet nicht mit Parent-Skripts oder Child-Objekten, sondern mit herkömmlichen objektorientierten Programmierungstechniken. Weitere Informationen über objektorientierte Programmierung in JavaScript-Syntax finden Sie unter „Objektorientierte Programmierung mithilfe von JavaScript“ auf Seite 60.

- Der Zugriff auf **Flash®-Darstellerkomponenten**, die auf der Bühne abgelegt werden (Flash-Sprites) und unsichtbar sind, kann nur über das „Member"-Objekt erfolgen. Das Verwenden eines Sprite-Objekts für ein Flash-Sprite mit einer unsichtbaren Eigenschaft führt zu einem Fehler.
- **Skripts** werden unabhängig vom Drehbuch Darstellern direkt zugeordnet und sind verfügbar, wenn der betreffende Darsteller einem Sprite zugeordnet ist.

  Im Gegensatz zu Verhalten, Filmskripten und Parent-Skripts sind Darstellerskripten nicht im Besetzungsfenster aufgeführt. Wenn Sie in „Besetzungsfenster-Voreinstellungen" die Option „Darstellerskript-Symbole einblenden" aktivieren, werden Darsteller, denen ein Skript zugeordnet ist, im Besetzungsfenster mit einem kleinen Skriptsymbol in der linken unteren Ecke des Piktogramms gekennzeichnet.

Skriptterminologie

Die Skriptsprachen Lingo und JavaScript arbeiten mit Begriffen, die entweder beide gemeinsam haben oder in beiden unterschiedlich sind.

Die folgende Liste enthält eine Übersicht der wichtigsten Skriptbegriffe in alphabetischer Reihenfolge. Diese Begriffe werden in der gesamten *Director-Skriptreferenz* verwendet, weshalb Sie mit ihnen vertraut sein sollten, bevor Sie fortfahren.

- **Konstanten** sind Elemente, deren Wert sich nicht ändert. Die Konstanten `TAB`, `EMPTY` und `RETURN` haben beispielsweise in Lingo stets denselben Wert und können nicht geändert werden. In JavaScript haben Konstanten wie `Math.PI` und `Number.MAX_VALUE` stets denselben Wert und können nicht geändert werden. Sie können in JavaScript-Syntax mithilfe des Schlüsselwortes `const` auch eigene benutzerdefinierte Konstanten erstellen.

  Weitere Informationen zu Konstanten finden Sie unter „Konstanten" auf Seite 14.

- **Ereignisse** sind Aktionen, die während der Filmwiedergabe erfolgen. Ereignisse erfolgen, wenn ein Film anhält, ein Sprite startet, der Abspielkopf in ein Bild eintritt, der Benutzer eine Taste drückt usw. Alle Ereignisse in Director sind vordefiniert und haben stets dieselbe Bedeutung.

  Weitere Informationen zu Ereignissen finden Sie unter „Ereignisse" auf Seite 28.

- **Ausdrücke** sind Teile einer Anweisung, die einen Wert liefern. Beispiel: `2 + 2`.
- **Funktionen** sind entweder Funktionen oberster Ebene oder bestimmte Typen von JavaScript-Code.

  Eine *Funktion oberster Ebene* weist einen Film an, einen Befehl während der Filmwiedergabe auszuführen, oder gibt einen Wert zurück, wird jedoch nicht von einem bestimmten Objekt aufgerufen. Die Funktion oberster Ebene `list()` wird beispielsweise mithilfe der Syntax `list()` aufgerufen. Wie eine Funktion weist eine *Methode* einen Film an, einen Befehl während der Filmwiedergabe auszuführen, oder gibt einen Wert zurück. Sie wird jedoch stets von einem Objekt aufgerufen.

  Eine Funktion wird in JavaScript-Syntax verwendet, um eine Ereignisprozedur, ein benutzerdefiniertes Objekt, eine benutzerdefinierte Methode usw. abzubilden. Die Verwendung von JavaScript-Funktionen in diesen Fällen wird in den entsprechenden Themen im weiteren Verlauf dieser Referenz beschrieben.

- **Prozeduren** bzw. Ereignisprozeduren sind Anweisungssätze innerhalb eines Skripts, die als Reaktion auf ein bestimmtes Ereignis und eine nachfolgende Nachricht ausgeführt werden. Wenn ein Ereignis eintritt, generiert und sendet Director eine entsprechende Nachricht an Skripts, und eine entsprechende Prozedur wird als Reaktion auf die Nachricht ausgeführt. Die Namen der Prozeduren entsprechend stets den Ereignissen und Nachrichten, auf die sie reagieren.

  Hinweis: *Obgleich in JavaScript-Syntax ein Ereignis tatsächlich von einer Funktion verarbeitet wird, wird der Begriff Prozedur in dieser Referenz allgemein verwendet, um sowohl auf Lingo-Prozeduren als auch JavaScript-Funktionen zu verweisen, die Ereignisse verarbeiten.*

  Weitere Informationen zu Prozeduren finden Sie unter „Prozeduren“ auf Seite 30.

- **Schlüsselwörter** sind reservierte Wörter, die eine besondere Bedeutung haben. Das Schlüsselwort `end` gibt beispielsweise in Lingo das Ende einer Prozedur an. In JavaScript gibt das Schlüsselwort `var` an, dass das ihm folgende Element eine Variable ist.

- **Listen** (Lingo) oder **Arrays** (JavaScript) sind sortierte Wertegruppen zur Verfolgung und Aktualisierung von Datenarrays, z. B. Namenfolgen oder Variablenwerte. Ein einfaches Beispiel hierfür ist die folgende Zahlenliste: `[1, 4, 2].`

  Weitere Informationen über Listen in Lingo- und JavaScript-Syntax finden Sie unter „Lineare Listen und Eigenschaftslisten“ auf Seite 33.

  Informationen zur Verwendung von JavaScript-Arrays finden Sie unter „JavaScript-Arrays“ auf Seite 41.

- **Nachrichten** sind Hinweise, die Director an Skripts sendet, wenn in einem Film bestimmte Ereignisse eintreten. Wenn beispielsweise der Abspielkopf in ein bestimmtes Bild eintritt, findet das Ereignis `enterFrame` statt, und Director sendet eine `enterFrame`-Nachricht. Wenn ein Skript eine `enterFrame`-Prozedur enthält, werden die in dieser Prozedur enthaltenen Anweisungen ausgeführt, da die Prozedur die Nachricht `enterFrame` empfing. Wenn keine Skripts eine Prozedur für eine Nachricht enthalten, wird die Nachricht im Skript ignoriert.

  Weitere Informationen zu Nachrichten finden Sie unter „Nachrichten“ auf Seite 29.

- **Methoden** sind Elemente, die einen Film anweisen, einen Befehl während der Filmwiedergabe auszuführen oder einen Wert zurückgeben. Sie werden von einem Objekt aufgerufen. Sie können z. B. die `insertFrame()`-Methode im Movie-Objekt mithilfe der Syntax `_movie.insertFrame()` aufrufen. Der Unterschied zwischen den hinsichtlich ihrer Funktionalität ähnlichen Funktionen oberster Ebene und Methoden ist, dass Methoden stets aus einem Objekt aufgerufen werden und Funktionen oberster Ebene nicht.

- **Operatoren** sind Elemente, die aus einem oder mehreren Werten einen neuen Wert berechnen. Der Additionsoperator (+) beispielsweise addiert zwei oder mehrere Werte und bildet einen neuen Wert.

  Weitere Informationen zu Operatoren finden Sie unter „Operatoren“ auf Seite 20.

- **Parameter** sind Platzhalter, mit denen Sie Werte an Skripts übergeben können. Parameter eignen sich nur für Methoden und Ereignisprozeduren (nicht für Eigenschaften) und sind bei einigen Methoden obligatorisch, bei anderen optional.

  Die `go()`-Methode des Movie-Objekts lenkt beispielsweise den Abspielkopf zu einem bestimmten Bild und gibt optional den Namen des Films an, in dem sich das Bild befindet. Um diese Aufgabe ausführen zu können, benötigt die `go()`-Methode mindestens einen Parameter und lässt einen zweiten Parameter zu. Der erste, obligatorische Parameter gibt das Bild an, das der Abspielkopf ansteuern soll. Der zweite, optionale Parameter gibt an, in welchen Film sich das Bild befindet. Da der erste Parameter erforderlich ist, erfolgt ein Skriptfehler, wenn dieser Parameter bei Aufruf der `go()`-Methode fehlt. Da der zweite Parameter optional ist, führt die Methode ihre Aufgabe auch dann aus, wenn der Parameter fehlt.

- **Eigenschaften** sind Attribute, die ein Objekt definieren. Ein Sprite in einem Film hat beispielsweise bestimmte Attribute wie Breite, Höhe, Hintergrundfarbe usw. Um auf die Werte dieser drei Attribute zuzugreifen, müssen Sie die Eigenschaften `width`, `height` und `backColor` des Sprite-Objekts verwenden.

  Weitere Informationen über das Zuweisen von Eigenschaften an Variablen finden Sie unter „Speichern und Aktualisieren von Werten in Variablen“ auf Seite 16.

- **Anweisungen** sind gültige Instruktionen, die von Director ausgeführt werden können. Alle Skripts bestehen aus Anweisungssätzen. Der folgende Lingo-Code ist eine einzelne vollständige Anweisung:

  ```
  _movie.go("Author")
  ```

  Weitere Informationen über das Schreiben von Skriptanweisungen finden Sie unter „Skripterstellung im Punktsyntaxformat“ auf Seite 45..

- **Variablen** sind Elemente zum Speichern und Aktualisieren von Werten. Variablen müssen mit einem Buchstaben, einem Unterstrich (_) oder Dollarzeichen ($) beginnen. Nachfolgende Zeichen in einem Variablennamen können auch Ziffern (0-9) sein. Um Variablen Werte zuzuweisen oder die Werte mehrerer Eigenschaften zu ändern, verwenden Sie den Ist-gleich-Operator (=).

  Weitere Informationen zum Verwenden von Variablen finden Sie unter „Variablen“ auf Seite 16.

Skriptsyntax

Die folgenden allgemeinen Syntaxregeln gelten für Lingo- und JavaScript-Syntax.

- Kommentarmarkierungen können in Lingo- und JavaScript-Syntax unterschiedlich sein.

  Lingo-Kommentare sind durch einen vorangestellten doppelten Bindestrich (`--`) gekennzeichnet. Jeder Kommentarzeile, die sich über mehrere Zeilen erstreckt, muss ein doppelter Bindestrich voranstellt werden.

  ```
  -- This is a single-line Lingo comment

  -- This is a
  -- multiple-line Lingo comment
  ```

  JavaScript-Kommentare in einer einzelnen Zeile wird ein doppelter Schrägstrich (`//`) vorangestellt. Kommentaren, die sich über mehrere Zeilen erstrecken, wird `/*` vorangestellt und `*/` angefügt.

```
// This is a single-line JavaScript syntax comment

/* This is a
multiple-line JavaScript syntax comment */
```

Sie können einen Kommentar in eine eigene Zeile setzen oder an eine beliebige Anweisung anhängen. Text, der auf die Kommentarmarkierungen in derselben Zeile folgt, wird ignoriert.

Kommentare können beliebige Informationen enthalten, z. B. Anmerkungen zu einem bestimmten Skript oder einer Prozedur oder Hinweise zu einer Anweisung, deren Zweck nicht offensichtlich ist. Kommentare helfen Ihnen und anderen Benutzern, eine Prozedur zu verstehen, wenn Sie sich schon länger nicht mehr mit ihr befasst haben.

Die Anzahl der Kommentare hat keinerlei Auswirkung auf die Größe Ihrer im komprimierten DCR- oder DXR-Format gespeicherten Filmdatei, da während des Komprimierungsvorgangs sämtliche Kommentare aus der Datei entfernt werden.

Mithilfe von Kommentarmarkierungen können Sie außerdem Codeabschnitte zum Testen oder Debuggen deaktivieren. Durch Hinzufügen von Kommentarmarkierungen lassen sich die Codeabschnitte vorübergehend in Kommentare verwandeln und brauchen nicht gelöscht zu werden. Um Kommentarmarkierungen schnell hinzuzufügen oder zu löschen, wählen Sie den gewünschten Code aus und klicken dann im Skriptfenster auf die Schaltfläche „Kommentar“ oder „Kommentar entfernen“.

- Hinter allen Methoden- und Funktionsnamen sind Klammern erforderlich. Wenn Sie beispielsweise die `beep()`-Methode des Sound-Objekts aufrufen, müssen Sie Klammern hinter das Wort „beep“ setzen. Andernfalls tritt ein Skriptfehler auf.

  ```
  // JavaScript syntax
  _sound.beep(); // this statement will work properly
  _sound.beep; // this statement will result in a script error
  ```

  Wenn Sie eine Methode, Funktion oder Prozedur in einer anderen Methode, Funktion oder Prozedur aufrufen, müssen Sie der aufrufenden Anweisung Klammern hinzufügen. Im folgenden Beispiel enthält die `modifySprite()`-Methode einen Aufruf der `spriteClicked`-Prozedur. Der Aufruf der `spriteClicked`-Prozedur muss Klammern enthalten, da andernfalls ein Skriptfehler auftritt:

  ```
  // JavaScript syntax
  function modifySprite() {
      spriteClicked(); // this call to the handler will work properly
      spriteClicked; // this call to the handler results in a script error
  }
  function spriteClicked() {
      // handler code here
  }
  ```

  Sie können Klammern auch einsetzen, um die vorgegebene Verarbeitungsreihenfolge für mathematische Operationen zu umgehen oder Anweisungen leichter lesbar zu machen. Der erste folgende mathematische Ausdruck ergibt beispielsweise 13, der zweite 5:

  ```
  5 * 3 - 2 -- yields 13
  5 * (3 - 2) -- yields 5
  ```

- Die Ereignisprozedursyntax kann in Lingo und JavaScript unterschiedlich sein. In Lingo verwenden Prozeduren die Syntax `on` *handlerName*. In JavaScript werden Prozeduren als Funktionen implementiert, welche die Syntax `function` *handlerName*`()` verwenden. Die folgenden Anweisungen bilden beispielsweise eine Prozedur, die einen Warnton abspielt, wenn der Benutzer mit der Maus klickt:

```
-- Lingo syntax
on mouseDown
    _sound.beep()
end
// JavaScript syntax
function mouseDown() {
    _sound.beep();
}
```

- Die Parametersyntax von Ereignisprozeduren kann in Lingo und JavaScript unterschiedlich sein. Sowohl Lingo- als auch JavaScript-Syntax unterstützt, dass an eine Prozedur übergebene Parameter in Klammern gesetzt werden. Werden mehrere Parameter übergeben, werden die einzelnen Parameter durch ein Komma getrennt. In Lingo können auch Parameter übergeben werden, die nicht in Klammern gesetzt sind. Die folgende Prozedur namens `addThem` empfängt beispielsweise die beiden Parameter `a` und `b`:

```
-- Lingo syntax
on addThem a, b -- without parentheses
    c = a + b
end

on addThem(a, b) -- with parentheses
    c = a + b
end

// JavaScript syntax
function addThem(a, b) {
    c = a + b;
}
```

- Das Schlüsselwort `const` kann in JavaScript-Syntax verwendet werden, um eine Konstante anzugeben, deren Wert sich nicht ändert. Lingo hat einen eigenen vordefinierten Konstantensatz (`TAB`, `EMPTY` usw.), weshalb das Schlüsselwort `const` nicht für Lingo gilt.

  Die folgende Anweisung gibt beispielsweise die Konstante `intAuthors` an und legt ihren Wert auf 12 fest, der stets 12 bleibt und nicht mithilfe von Skript geändert werden kann:

```
// JavaScript syntax
const intAuthors = 12;
```

- Das Schlüsselwort `var` in JavaScript-Syntax kann vor ein Element gesetzt werden, um anzugeben, dass das Element eine Variable ist. Die folgende Anweisung erstellt die Variable `startValue`:

```
// JavaScript syntax
var startValue = 0;
```

  Hinweis: *Wenngleich das Verwenden von `var` in JavaScript-Syntax optional ist, wird empfohlen, dass Sie lokale JavaScript-Syntaxvariablen (oder Variablen innerhalb einer Funktion) stets mit `var` deklarieren. Weitere Informationen zum Verwenden von Variablen finden Sie unter „Variablen“ auf Seite 16.*

- Das Zeilenfortsetzungssymbol (\) in Lingo gibt an, dass eine lange Zeile mit Beispielcode in zwei oder mehrere Zeilen aufgeteilt wurde. Lingo-Zeilen, die auf diese Weise aufgeteilt werden, sind keine separaten Codezeilen. Der folgende Code kann beispielsweise weiterhin ausgeführt werden:

```
-- Lingo syntax
tTexture = member("3D").model("box") \
    .shader.texture
```

JavaScript-Syntax enthält kein Zeilenfortsetzungssymbol. Um mehrere Zeilen mit JavaScript-Code aufzuteilen, können Sie eine Zeilenschaltung am Ende einer Zeile hinzufügen und anschließend den Code in der folgenden Zeile fortsetzen.

- Zum Angeben des Endes einer Anweisung in JavaScript-Syntaxcode können Sie Semikolons verwenden. Semikolons gelten nicht für Lingo-Syntax.

  Die Verwendung von Semikolons ist optional. Falls verwendet, wird ein Semikolon an das Ende einer vollständigen Anweisung gesetzt. Die beiden folgenden Anweisungen erstellen beispielsweise die Variable `startValue`:

  ```
  // JavaScript syntax
  var startValue = 0
  var startValue = 0;
  ```

  Mit einem Semikolon wird nicht unbedingt das Ende einer Zeile in JavaScript-Syntaxcode angegeben, und mehrere Anweisungen können in einer Zeile platziert werden. Zum Verbessern der Lesbarkeit wird jedoch empfohlen, getrennte Anweisungen in getrennten Zeilen zu platzieren. Die folgenden drei Anweisungen nehmen beispielsweise nur eine Codezeile ein und funktionieren ordnungsgemäß, doch es ist schwierig, den Code zu lesen:

  ```
  // JavaScript syntax
  _movie.go("Author"); var startValue = 0; _sound.beep();
  ```

- Leerzeichen in Ausdrücken und Anweisungen werden in Lingo- und JavaScript-Syntax ignoriert. In Zeichenfolgen, die in Anführungszeichen gesetzt sind, werden Leerzeichen wie Zeichen behandelt. Soll eine Zeichenfolge Leerzeichen enthalten, müssen Sie diese explizit einfügen. Beispiel: Die erste folgende Anweisung ignoriert die Leerzeichen zwischen den Listeneinträgen, die zweite Anweisung enthält dagegen die Leerzeichen.

  ```
  -- Lingo syntax
  myList1 = ["1",      "2",      "3"] -- yields ["1", "2", "3"]
  myList2 = ["  1  ", "  2  ", "  3  "] -- yields ["  1  ", "  2  ", "  3  "]
  ```

- Groß-/Kleinschreibung kann in Lingo- und JavaScript-Syntax unterschiedlich sein.

  Lingo unterscheidet nicht zwischen Groß- und Kleinschreibung, sodass Groß- und Kleinbuchstaben nach Belieben verwendet werden können. Die vier folgenden Anweisungen sind gleichbedeutend:

  ```
  -- Lingo syntax
  member("Cat").hilite = true
  member("cat").hiLite = True
  MEMBER("CAT").HILITE = TRUE
  Member("Cat").Hilite = true
  ```

  Obgleich Lingo nicht zwischen Groß- und Kleinschreibung unterscheidet, wird empfohlen, eine Groß- und Kleinbuchstabensrichtlinie zu wählen und in allen Skripts durchgängig zu befolgen. Dadurch können Namen von Prozeduren, Variablen, Darstellern usw. einfacher bestimmt werden.

  JavaScript unterscheidet Groß- und Kleinbuchstaben bei Verweisen auf Objekte, bei den Eigenschaften und Methoden oberster Ebene und bei Verweisen auf benutzerdefinierte Variablen. Beispiel: Die Methode oberster Ebene `sprite()` gibt einen Verweis auf ein bestimmtes Sprite-Objekt zurück und ist in Director ausschließlich in Kleinbuchstaben implementiert. Die erste nachfolgende Anweisung verweist auf den Namen des ersten Sprites in einem Film, während es bei der zweiten und dritten Anweisung zu einem Skriptfehler kommt.

  ```
  // JavaScript syntax
  sprite(1).name // This statement functions normally
  Sprite(1).name // This statement results in a script error
  SPRITE(1).name // This statement results in a script error
  ```

  Bei Literalzeichenfolgen wird in Lingo- und JavaScript-Syntax stets Groß-/Kleinschreibung unterschieden.

  Weitere Informationen zum Verwenden von Zeichenfolgen finden Sie unter „Zeichenfolgen“ auf Seite 13.

Datentypen

Ein *Datentyp* ist eine Zusammenstellung von Daten mit Werten, die ähnliche, vordefinierte Merkmale haben. Alle Variablen- und Eigenschaftswerte in Director sowie alle von Methoden zurückgegebenen Werte haben einen bestimmten Datentyp.

Betrachten Sie z. B. die beiden folgenden Anweisungen. In der ersten Anweisung wird die Variable `intX` dem Ganzzahlwert 14 zugewiesen. Der Datentyp der Variablen `intX` ist deshalb Integer. In der zweiten Anweisung wird die Variable `stringX` einer Folge von Zeichenwerten zugewiesen. Der Datentyp der Variablen `stringX` ist deshalb String.

```
-- Lingo syntax
intX = 14
stringX = "News Headlines"

// JavaScript syntax
var intX = 14;
var stringX = "News Headlines";
```

Die von Methoden und Funktionen zurückgegeben Werte haben auch einen inhärenten Datentyp. Die `windowPresent()`-Methode des Player-Objekts gibt beispielsweise einen Wert zurück, der angibt, ob ein Fenster vorhanden ist. Der Rückgabewert ist entweder `TRUE` (1) oder `FALSE` (0).

Lingo und JavaScript haben einige Datentypen gemeinsam, während andere für die beiden Skriptsprachen spezifisch sind. Die Zusammenstellung von Datentypen, die Director unterstützt, ist festgelegt und kann nicht geändert werden, was bedeutet, dass keine neuen Datentypen hinzugefügt und vorhandene Datentypen nicht entfernt werden können. Director unterstützt die folgenden Datentypen.

Datentyp	**Beschreibung**
# (symbol)	A self-contained unit that can be used to represent a condition or flag. For example, `#list` oder `#word`.
Array	(Nur JavaScript-Syntax) Wenngleich nicht eigentlich ein Datentyp, kann ein „Array“-Objekt zum Arbeiten mit linearen Wertelisten verwendet werden. Die Funktionalität eines „Array“-Objekts ähnelt der des Datentyps „List“ in Lingo.
Boolesch	Ein Wert, der entweder `TRUE` (1) oder `FALSE` (0) ist. In Lingo sind alle `TRUE`- oder `FALSE`-Werte einfache ganzzahlige Konstanten: 1 steht für `TRUE`, 0 für `FALSE`. In JavaScript entsprechen alle `true`- oder `false`-Werte standardmäßig den tatsächlichen booleschen Werten `true` oder `false`, die jedoch automatisch in einfache ganzzahlige Konstanten umgewandelt werden, wenn dies in Director erforderlich ist. In Lingo können `TRUE` und `FALSE` groß und klein geschrieben werden. In JavaScript müssen `true` und `false` stets klein geschrieben werden.
Farbe	Gibt die Farbe eines Objekts an.
Constant	Ein Datenelement, dessen Wert sich nicht ändert.
Datum	Wenngleich nicht eigentlich ein Datentyp, kann ein „Date“-Objekt in JavaScript-Syntax für das Arbeiten mit Datumsangaben verwendet werden. Wählen Sie in Lingo die „date()“-Methode, um ein „Date“-Objekt zu erstellen und mit Datumsangaben zu arbeiten.
Fließkommazahl (float)	(Nur Lingo) Eine Fließkommazahl, z. B. 2.345 oder 45.43.
Funktion	(Nur JavaScript) Wenngleich nicht eigentlich ein Datentyp, kann ein „Function“-Objekt zum Angeben einer auszuführenden Codezeichenfolge verwendet werden.
Ganzzahl	(Nur Lingo) Eine Ganzzahl, z. B. 5 oder 298.
List	Eine lineare oder Eigenschaftsliste mit Werten bzw. Eigenschaftsnamen und -werten.

Datentyp	Beschreibung
Math	(Nur JavaScript) Wenngleich nicht eigentlich ein Datentyp, kann ein „Math"-Objekt zum Ausführen mathematischer Funktionen verwendet werden.
null	(Nur JavaScript) Gibt eine Variable an, deren Wert sich in numerischen Kontexten als 0 und in booleschen Kontexten als `false` verhält.
Zahl	(Nur JavaScript) Wenngleich nicht eigentlich ein Datentyp, kann ein „Number"-Objekt zum Abbilden numerischer Konstanten, z. B. eines Maximalwertes, von NaN-Werten (Not-a-Number) und Unendlichkeit verwendet werden.
Objekt	Wenngleich nicht eigentlich ein Datentyp, kann ein „Object"-Objekt zum Erstellen eines selbstbenannten Containers mit Daten und Methoden verwendet werden, die auf diese Daten angewendet werden.
Point	Ein Punkt im Koordinatenraum mit einer horizontalen und einer vertikalen Koordinate.
Rect	Ein Rechteck im Koordinatenraum.
RegExp	(Nur JavaScript) Ein reguläres Ausdrucksmuster, das zum Abgleichen von Zeichenkombinationen in Zeichenfolgen verwendet wird.
Zeichenfolge	Eine zusammenhängende Folge von Tastatursymbolen bzw. Zeichenwerten, z. B. "Director" oder "€ 5,00".
undefined	(Nur JavaScript) Kennzeichnet eine Variable ohne Wert.
Vector	Ein Punkt im 3D-Raum.
VOID	(Nur Lingo) Gibt einen leeren Wert an.

Hinweis: *Viele der für JavaScript spezifischen Datentypen und Objekte enthalten einige Methoden- und Eigenschaftssätze, die zur weiteren Bearbeitung dieser Typen verwendet werden können. Einerseits wird in der Director-Skriptreferenz auf einige dieser Methoden und Eigenschaften verwiesen, andererseits werden keine vollständigen weiterführenden Informationen dazu geboten. Detaillierte Informationen zu diesen Datentypen und Objekten sowie ihren Methoden und Eigenschaften finden Sie in Ressourcen anderer Anbieter zum jeweiligen Thema.*

Den vordefinierten Eigenschaften in Director, z. B. der `name`-Eigenschaft des Cast-Objekts, können nur Werte zugewiesen werden, deren Datentyp dem inhärenten Datentyp der Eigenschaft entspricht. Der inhärente Datentyp der Eigenschaft `name` des Cast-Objekts ist beispielsweise eine Zeichenfolge, sodass der Wert eine Zeichenfolge wie `News Headlines` sein muss. Wenn Sie versuchen, dieser Eigenschaft einen Wert mit einem anderen Datentyp zuzuweisen, z. B. die Ganzzahl 20, tritt ein Skriptfehler auf.

Wenn Sie eigene Eigenschaften erstellen, können diese einen beliebigen Datentyp haben, der vom Datentyp des Ausgangswertes unabhängig ist.

Lingo und JavaScript sind dynamisch typisiert. Dies bedeutet, dass Sie beim Deklarieren einer Variablen nicht den Datentyp angeben müssen und dass Datentypen bei der Ausführung eines Skripts dynamisch den Anforderungen entsprechend konvertiert werden.

Die folgende JavaScript-Syntax legt die Variable `myMovie` anfänglich auf eine Ganzzahl und im weiteren Verlauf des Skripts auf eine Zeichenfolge fest. Bei Ausführung des Skripts wird der Datentyp von `myMovie` automatisch konvertiert:

```
-- Lingo syntax
myMovie = 15 -- myMovie is initially set to an integer
...
myMovie = "Animations" -- myMovie is later set to a string

// JavaScript syntax
var myMovie = 15; // myMovie is initially set to an integer
...
myMovie = "Animations"; // myMovie is later set to a string
```

Literalwerte

Ein *Literalwert* ist Teil einer Anweisung oder eines Ausdrucks, der in der vorliegenden Form verwendet und nicht als Variable oder Skriptelement interpretiert wird. Die in Skripts verwendeten Literalwerte sind Zeichenfolgen, Ganzzahlen, Dezimalzahlen, Darstellernamen und -nummern, Bild- und Filmnamen und -nummern, Symbole und Konstanten.

Für die verschiedenen Arten von Literalwerten gelten unterschiedliche Regeln.

Zeichenfolgen

Zeichenfolgen sind Wörter oder Zeichengruppen, die im Skript nicht als Variablen, sondern als reguläre Wörter behandelt werden. Zeichenfolgen müssen stets in doppelte Anführungszeichen gesetzt werden. Mithilfe von Zeichenfolgen können Sie beispielsweise Nachrichten an den Benutzer Ihres Films übermitteln oder Darsteller benennen. In der folgenden Anweisung sind `Hello` und `Greeting` Zeichenfolgen. `Hello` ist der Literaltext, der in den Textdarsteller eingegeben wird, und `Greeting` der Name des Darstellers.

```
-- Lingo syntax
member("Greeting").text = "Hello"
```

Strings müssen auch beim Testen in doppelte Anführungszeichen gesetzt werden. Beispiel:

```
-- Lingo syntax
if "Hello Mr. Jones" contains "Hello" then soundHandler
```

Lingo und JavaScript behandeln Leerzeichen am Anfang oder Ende einer Zeichenfolge als literalen Teil der Zeichenfolge. Der folgende Ausdruck enthält ein Leerzeichen nach dem Wort *to*:

```
// JavaScript syntax
trace("My thoughts amount to ");
```

Während Lingo in Verweisen auf Darsteller, Variablen usw. nicht zwischen Groß- und Kleinbuchstaben unterscheidet, kommt es bei Literalzeichenfolgen sehr wohl auf die richtige Schreibweise an. Die beiden folgenden Anweisungen schreiben unterschiedlichen Text in den angegebenen Darsteller, da es sich bei den Strings `Hello` und `HELLO` um Literalstrings handelt:

```
-- Lingo syntax
member("Greeting").text = "Hello"
member("Greeting").text = "HELLO"
```

In Lingo kann die `string()`-Funktion einen numerischer Wert in eine Zeichenfolge umwandeln. In JavaScript kann die `toString()`-Methode einen numerischer Wert in eine Zeichenfolge umwandeln.

***Hinweis:** Der Versuch, die `toString()`-Methode in JavaScript-Syntax auf einen Wert vom Typ `null` oder `undefined` anzuwenden, führt zu einem Skriptfehler. Dies ist ein Unterschied zu Lingo, da die `string()`-Funktion in dieser Skriptsprache auf alle Werte einschließlich `VOID` angewendet werden kann.*

Zahlen

In Lingo gibt es zwei Zahlentypen: Ganzzahlen und Dezimalzahlen.

Eine *Ganzzahl*(Integer) ist eine ganze Zahl ohne Brüche oder Dezimalstellen im Bereich von -2.147.483.648 bis +2.147.483.647. Geben Sie Ganzzahlen ohne Punkte ein. Negative Zahlen werden mit einem Minuszeichen (-) gekennzeichnet.

Dezimalzahlen werden gelegentlich auch als *Fließkommazahlen bezeichnet und sind Zahlen, die Dezimalstellen aufweisen.* Die Anzahl der angezeigten Dezimalstellen wird in Lingo durch die Eigenschaft `floatPrecision` bestimmt. Director verwendet in Berechnungen stets die vollständige Zahl mit bis zu 15 signifikanten Stellen und rundet Zahlen mit mehr als 15 signifikanten Stellen in Berechnungen auf.

JavaScript unterscheidet nicht zwischen Ganz- und Fließkommazahlen und verwendet lediglich Zahlen. Die folgenden Anweisungen veranschaulichen, dass die Zahl 1 in Lingo- eine Ganzzahl und in JavaScript-Syntax eine Zahl ist und dass die Dezimalzahl 1,05 in Lingo- eine Fließkommazahl und in JavaScript-Syntax eine Zahl ist:

```
-- Lingo syntax
put(ilk(1)) -- #integer
put(ilk(1.05)) -- #float

// JavaScript syntax
trace(typeof(1)) // number
trace(typeof(1.05)) // number
```

Mit der Funktion `integer()` können Sie in Lingo eine Dezimalzahl in eine Ganzzahl umwandeln. Sie können auch eine Ganzzahl in eine Dezimalzahl umwandeln, indem Sie eine mathematische Operation auf die Ganzzahl anwenden, z. B. eine Ganzzahl mit einer Dezimalzahl multiplizieren. In JavaScript-Syntax können Sie mit der `parseInt()`-Funktion eine Zeichenfolge oder Dezimalzahl in eine Ganzzahl umwandeln. Im Gegensatz zur Lingo-Funktion `integer()` rundet `parseInt()` ab. Die folgende Anweisung rundet beispielsweise die Dezimalzahl 3,9 und wandelt sie in die Ganzzahl 4 (Lingo) und die Zahl 3 (JavaScript) um:

```
-- Lingo syntax
theNumber = integer(3.9) -- results in a value of 4

// JavaScript syntax
var theNumber = parseInt(3.9); // results in a value of 3
```

In Lingo kann die `value()`-Funktion eine Zeichenfolge in einen numerischer Wert umwandeln.

Dezimalzahlen lassen sich auch in Exponentialschreibweise angeben, beispielsweise: `-1.1234e-100` oder `123.4e+9`.

Ganzzahlen oder Zeichenfolgen lassen sich in Lingo mit der Funktion `float()` in Dezimalzahlen umwandeln. In JavaScript können Sie mit der `parseFloat()`-Funktion eine Zeichenfolge in eine Dezimalzahl umwandeln. Die folgende Anweisung speichert beispielsweise den Wert 3,0000 (Lingo) und 3 (JavaScript) in der Variablen `theNumber`:

```
-- Lingo syntax
theNumber = float(3) -- results in a value of 3.0000

// JavaScript syntax
var theNumber = parseFloat(3) // results in a value of 3
```

Konstanten

Eine *Konstante* ist ein benannter Wert mit unveränderlichem Inhalt.

Die vordefinierten Elemente `TRUE`, `FALSE`, `VOID` und `EMPTY` sind beispielsweise in Lingo Konstanten, da ihre Werte immer dieselben bleiben. Die vordefinierten Elemente `BACKSPACE`, `ENTER`, `QUOTE`, `RETURN`, `SPACE` und `TAB` sind Konstanten, die auf Tastaturfunktionen verweisen. Mit der folgenden Anweisung können Sie beispielsweise testen, ob der Benutzer als letzte Taste die LEERTASTE gedrückt hat:

```
-- Lingo syntax
if _key.keyPressed() = SPACE then beep()
```

In JavaScript können Sie auf vordefinierte Konstanten mithilfe einiger Datentypen zugreifen, die für JavaScript eindeutig sind. Das Number-Objekt enthält z. B. Konstanten wie `Number.MAX_VALUE` und `Number.NaN`, das Math-Objekt enthält Konstanten wie `Math.PI` und `Math.E` usw.

Hinweis: *Diese Referenz bietet keine allumfassenden Informationen zu den in JavaScript vordefinierten Konstanten. Weitere Informationen zu diesen Konstanten finden Sie in den vielen Ressourcen anderer Anbieter zu diesem Thema.*

Sie können in JavaScript-Syntax mithilfe des Schlüsselwortes `const` auch eigene benutzerdefinierte Konstanten erstellen. Die folgende Anweisung erstellt beispielsweise die Konstante `items` und weist dieser den Wert 20 zu. Nach seiner Festlegung kann dieser Wert nicht geändert werden:

```
// JavaScript syntax
const items = 20;
```

Weitere Informationen zu Konstanten finden Sie unter „Konstanten“ auf Seite 176.

Symbole

Ein *Symbol* ist in Lingo eine Zeichenfolge oder ein anderer Wert, der mit dem Rautenzeichen (#) beginnt.

Symbole sind benutzerdefinierte Konstanten. Mit ihrer Hilfe lassen sich Werte sehr rasch miteinander vergleichen und Skripts mit wesentlich effizienterem Code erstellen.

Hinweis: *In Lingo ist die Großschreibung des Symbols der Kleinschreibung zugeordnet. Wenn Sie versuchen, eine Zeichenfolge (z. B. Sonja) mithilfe der Symbolfunktion in ein Symbol umzuwandeln, lautet die Ausgabe „sonja“.*

Die erste der beiden folgenden Anweisungen wird schneller ausgeführt als die zweite:

```
-- Lingo syntax
userLevel = #novice
userLevel = "novice"
```

Symbole dürfen keine Leer- oder Satzzeichen enthalten.

Mithilfe der `symbol()`-Methode lassen sich in Lingo- und JavaScript-Syntax Zeichenfolgen in ein Symbol umwandeln.

```
-- Lingo syntax
x = symbol("novice") -- results in #novice

// JavaScript syntax
var x = symbol("novice"); // results in #novice
```

Mithilfe der Funktion `string()` (Lingo) oder Methode `toString()` (JavaScript) lassen sich Symbole wieder in Zeichenfolgen umwandeln.

```
-- Lingo syntax
x = string(#novice) -- results in "novice"

// JavaScript syntax
var x = symbol("novice").toString(); // results in "novice"
```

In JavaScript-Syntax können Sie nicht Symbole mit demselben Namen vergleichen, um zu bestimmen, ob sie auf dasselbe Symbol verweisen. Um Symbole mit demselben Namen zu vergleichen, müssen Sie diese zunächst mit der `toString()`-Methode in Zeichenfolgen umwandeln und anschließend vergleichen.

Variablen

Director verwendet zum Speichern und Aktualisieren von Werten *Variablen*. Wie der Name andeutet, enthält eine Variable einen Wert, der beim Abspielen eines Films geändert oder aktualisiert werden kann. Durch Ändern des Wertes einer Variablen während der Filmwiedergabe können Sie zum Beispiel eine URL speichern, das Ende eines Netzwerkvorgangs aufzeichnen oder verfolgen, wie oft ein Benutzer an einem Online-Chat teilnimmt, und viele weitere Aufgaben ausführen.

Beim Deklarieren der Variablen sollten Sie ihr grundsätzlich einen bekannten Anfangswert zuordnen. Dies wird als *Initialisieren* einer Variablen bezeichnet und erleichtert das Verfolgen und Vergleichen ihres Wertes während der Filmwiedergabe.

Variablen können als global oder lokal definiert werden. Eine *lokale Variable* existiert nur so lange wie die Prozedur ausgeführt wird, in der sie definiert ist. *Globale Variablen* existieren und behalten ihren Wert, solange Director ausgeführt wird, auch wenn ein Film zu einem anderen Film verzweigt. Eine Variable kann in einer einzelnen Prozedur, einem bestimmten Skript oder gesamten Film global sein. Der Geltungsbereich hängt davon ab, wie die globale Variable initialisiert wird.

Wenn eine Variable im gesamten Film verfügbar sein soll, sollten Sie sie in einer Prozedur vom Typ `on prepareMovie` (Lingo) oder `function prepareMovie()` (JavaScript) deklarieren. Dadurch wird sichergestellt, dass die Variable vom Anfang des Films an zur Verfügung steht.

Weitere Informationen über den Gebrauch globaler und lokaler Variablen finden Sie unter „Verwenden globaler Variablen“ auf Seite 17 und „Verwenden lokaler Variablen“ auf Seite 19.

Speichern und Aktualisieren von Werten in Variablen

Variablen können Daten für die folgenden Datentypen in Director enthalten: Ganzzahlen, Zeichenfolgen, boolesche Werte vom Typ `TRUE` oder `FALSE`, Symbole, Listen oder Rechenergebnisse. Verwenden Sie den Operator (`=`) zum Speichern der Werte von Eigenschaften und Variablen.

Wie bereits im Abschnitt „Datentypen“ in dieser Referenz erwähnt, sind Variablen in Lingo- und JavaScript-Syntax dynamisch typisiert, was bedeutet, dass sie zu verschiedenen Zeitpunkten unterschiedliche Datentypen enthalten können. (Die Möglichkeit, den Variablentyp zu ändern, ist einer der wesentlichen Vorteile von Lingo gegenüber anderen Programmiersprachen wie z. B. Java™ und C++, bei denen sich der Variablentyp nicht ändern lässt.)

Die Anweisung `x = 1` erstellt beispielsweise die Variable `x` als ganzzahlige Variable, da Sie der Variablen eine Ganzzahl als Wert zugeordnet haben. Wenn Sie zu einem späteren Zeitpunkt die Anweisung `x = "one"` verwenden, wird die Variable `x` zu einer Zeichenfolgenvariablen, da sie nun eine Zeichenfolge enthält.

Sie können eine Zeichenfolge in eine Zahl umwandeln, indem Sie die `value()`-Funktion (Lingo) oder `parseInt()`-Methode (JavaScript) verwenden, bzw. eine Zahl in eine Zeichenfolge umwandeln, indem Sie die `string()`-Funktion (Lingo) oder die `toString()`-Methode (JavaScript) wählen.

Die Werte einiger Eigenschaften können sowohl festgelegt (der Wert wird zugewiesen) als auch zurückgegeben werden (der Wert wird abgerufen). Einige Eigenschaftswerte können nur zurückgegeben werden. Eigenschaften, deren Werte sowohl festgelegt als auch zurückgegeben werden können, haben den Typ *Lesen/Schreiben*. Eigenschaften, deren Wert nur zurückgegeben werden kann, werden *schreibgeschützt* genannt.

In vielen Fällen handelt es sich hierbei um Eigenschaften, die eine Bedingung beschreiben, welche nicht in Director gesteuert werden kann. So ist es beispielsweise nicht möglich, die Darstellereigenschaft `numChannels` festzulegen, welche die Anzahl der Kanäle in einem Film mit Adobe® Shockwave® angibt. Sie können die Anzahl der Kanäle jedoch abfragen, indem Sie auf die Eigenschaft `numChannels` eines Darstellers verweisen.

So weisen Sie einer Variablen einen Wert zu

❖ Verwenden Sie den Ist-gleich-Operator (=).

Die folgende Anweisung weist beispielsweise der Variablen `placesToGo` eine URL zu:

```
// JavaScript syntax
var placesToGo = "http://www.adobe.com";
```

Variablen können auch die Ergebnisse mathematischer Operationen enthalten. Die folgende Anweisung weist der Variablen `mySum` die Summe einer Addition als Wert zu:

```
-- Lingo syntax
mySum = 5 + 5 -- this sets mySum equal to 10
```

Weiteres Beispiel: Die folgende Anweisung ruft die Eigenschaft `member` von Sprite 2 ab und legt diese in der Variablen `textMember` ab:

```
-- Lingo syntax
textMember = sprite(2).member
```

Beim Benennen von Variablen sollten Sie darauf achten, dass der Name den Zweck der Variablen beschreibt. Dadurch werden Ihre Skripts besser lesbar. Der Name der Variablen `mySum` beispielsweise weist darauf hin, dass die Variable eine Summe von Zahlen enthält.

So testen Sie die Werte von Eigenschaften oder Variablen

❖ Verwenden Sie die Funktion `put()` oder `trace()` im Nachrichtenfenster, oder überprüfen Sie die Wert im Watcher-Fenster. (`put()` und `trace()` bieten identische Funktionalität und stehen beide in Lingo- und JavaScript-Syntax zur Verfügung).

Die folgende Anweisung zeigt beispielsweise den Wert der Variablen `myNumber` im Nachrichtenfenster an:

```
-- Lingo syntax
myNumber = 20 * 7
put(myNumber) -- displays 140 in the Message window

// JavaScript syntax
var myNumber = 20 * 7;
trace(myNumber) // displays 140 in the Message window
```

Verwenden globaler Variablen

Globale Variablen können von Prozeduren, Skripts und Filmen gemeinsam genutzt werden. Eine globale Variable existiert und behält ihren Wert, solange Director ausgeführt wird oder Sie die Methode `clearGlobals()` aufrufen.

In Adobe Shockwave Player werden globale Variablen in Filmen beibehalten, die mit der Methode `goToNetMovie()`, nicht aber in Filmen, die mit der Methode `goToNetPage()` angezeigt werden.

Jede Prozedur, die eine Variable als globale Variable deklariert, kann den aktuellen Wert der Variablen verwenden. Wenn die Prozedur den Variablenwert ändert, steht der neue Wert allen anderen Prozeduren zur Verfügung, die die Variable als globale Variable behandeln.

Es empfiehlt sich, beim Benennen globaler Variablen dem Namen stets ein kleines *g* voranzustellen. Auf diese Weise können Sie beim Überprüfen von Code auf einen Blick feststellen, bei welchen Variablen es sich um globale Variablen handelt.

Director bietet eine Möglichkeit, alle aktuellen Variablen und ihre aktuellen Werte anzuzeigen und die Werte aller globalen Variablen zu löschen.

So zeigen Sie alle aktuellen globalen Variablen und deren aktuelle Werte an

❖ Verwenden Sie im Nachrichtenfenster die `showGlobals()`-Methode.

Weitere Informationen über das Nachrichtenfenster finden Sie unter „Debuggen im Nachrichtenfenster“ auf Seite 86.

So löschen Sie alle aktuellen globalen Variablen

❖ Verwenden Sie die `clearGlobals()`-Methode im Nachrichtenfenster, um den Wert aller globalen Variablen auf `VOID` (Lingo) oder `undefined` (JavaScript) zu setzen.

Mithilfe des Objektinspektors können Sie die Werte globaler Variablen während der Filmwiedergabe überwachen. Weitere Informationen über den Objektinspektor finden Sie unter „Debuggen im Objektinspektor“ auf Seite 90.

Globale Variablen in Lingo

In Lingo werden Variablen standardmäßig als lokal behandelt, weshalb Sie vor den Variablennamen kein Schlüsselwort setzen müssen. Um eine globale Variable zu deklarieren, müssen Sie vor die Variable das Schlüsselwort `global` setzen.

Wenn Sie eine globale Variable oben in einem Skript und vor Prozeduren deklarieren, steht die Variable alle Prozeduren in diesem bestimmten Skript zur Verfügung. Wenn Sie eine globale Variable in einer Prozedur deklarieren, steht die Variable nur dieser Prozedur zur Verfügung. Wenn Sie jedoch eine globale Variable mit demselben Namen in zwei getrennten Prozeduren deklarieren, wird eine Änderung des Variablenwertes in einer Prozedur von der Variablen in der anderen Prozedur übernommen.

`gScript`, die allen Prozeduren im Skript zur Verfügung steht, und `gHandler`, die innerhalb der definierenden Prozedur und allen anderen Prozeduren zur Verfügung steht, die sie in der ersten Prozedurzeile deklarieren:

```
-- Lingo syntax
global gScript -- gScript is available to all handlers

on mouseDown
    global gHandler
    gScript = 25
    gHandler = 30
end

on mouseUp
    global gHandler
    trace(gHandler) -- displays 30
end
```

Variablen, die Sie in Lingo mit dem Begriff `global` als globale Variablen definieren, weisen automatisch den Anfangswert `VOID` auf.

Globale Variablen in JavaScript-Syntax

In JavaScript werden Variablen standardmäßig als global behandelt. Der Geltungsbereich einer globalen Variablen hängt von Art und Ort ihrer Deklarierung ab.

- Wenn Sie eine Variable in einer JavaScript-Syntaxfunktion deklarieren, ohne vor den Variablennamen das Schlüsselwort `var` zu setzen, steht die Variable allen Funktionen im Skript zur Verfügung, die diese enthalten.
- Wenn Sie eine Variable außerhalb einer JavaScript-Funktion deklarieren (mit dem oder ohne das Schlüsselwort `var`), steht die Variable allen Funktionen im Skript zur Verfügung, die diese enthalten.
- Wenn Sie eine Variable inner- oder außerhalb einer JavaScript-Syntaxfunktion mithilfe der Syntax `_global.`*varName* deklarieren, steht die Variable allen Skripts in einem Film zur Verfügung.

Das folgende Beispiel verwendet die Syntax `_global.gMovie` in einem Skript, um die Variable `gMovie` als global zu deklarieren. Diese Variable steht allen Skripts in diesem Film zur Verfügung:

```
// JavaScript syntax
_global.gMovie = 1; // Declare gMovie in one script

// Create a function in a separate script that operates on gMovie
function mouseDown() {
    _global.gMovie++;
    return(_global.gMovie);
}
```

Das folgende Beispiel deklariert die globale Variable `gScript` in einem Skript. Diese Variable steht nur Funktionen in diesem Skript zur Verfügung:

```
// JavaScript syntax
var gScript = 1; // Declare gScript in a script

// gScript is available only to functions in the script that defines it
function mouseDown() {
    gScript++;
    return(gScript);
}
```

Wenn Sie in JavaScript-Syntax Variablen vor Prozeduren definieren, haben die Variablen automatisch den Ausgangswert `undefined`.

Verwenden lokaler Variablen

Eine lokale Variable existiert nur so lange, wie die Prozedur, in der sie definiert ist, ausgeführt wird. Solange sich Lingo jedoch in der Prozedur befindet, in der eine lokale Variable definiert wurde, können Sie diese nach Belieben auch in anderen Ausdrücken verwenden oder ihren Wert ändern.

Sie sollten eine Variable als lokale Variable deklarieren, wenn Sie sie nur vorübergehend in einer bestimmten Prozedur verwenden möchten. Auf diese Weise können Sie verhindern, dass ihr Wert durch eine andere Prozedur, die eine gleichnamige Variable enthält, versehentlich geändert wird.

So erstellen Sie eine lokale Variable

- Weisen Sie in Lingo der Variablen mit dem Operator `=` einen Wert zu.
- Setzen Sie in JavaScript innerhalb einer Funktion vor den Variablennamen das Schlüsselwort `var`, und weisen Sie anschließend der Variablen mit dem Gleichheitsoperator einen Wert zu.

  Hinweis: *Da JavaScript-Variablen standardmäßig global sind, ist es möglich, wenn Sie versuchen, eine lokale Variable in einer Funktion ohne das Schlüsselwort* `var` *zu deklarieren, dass Ihr Skript für ein unerwartetes Verhalten sorgt. Deshalb wird, wenngleich das Verwenden von* `var` *optional ist, ausdrücklich empfohlen, alle lokalen JavaScript-Variablen mithilfe von* `var` *zu deklarieren, um unerwartete Verhalten zu vermeiden.*

So zeigen Sie alle aktuellen lokalen Variablen in der Prozedur an

❖ (Nur Lingo) Verwenden Sie die `showLocals()`-Funktion.

Diese Methode kann in Lingo im Nachrichtenfenster oder in Prozeduren zum Debugging eingesetzt werden. Das Ergebnis wird im Nachrichtenfenster angezeigt. Die `showLocals()`-Methode gibt es für JavaScript-Syntax nicht.

Mithilfe des Objektinspektors können Sie die Werte lokaler Variablen während der Filmwiedergabe überwachen. Weitere Informationen über den Objektinspektor finden Sie unter „Debuggen im Objektinspektor“ auf Seite 90.

Operatoren

Operatoren sind Elemente, die Lingo- und JavaScript-Syntax mitteilen, auf welche Weise die Werte eines Ausdrucks miteinander verbunden, verglichen oder modifiziert werden sollen. Lingo und JavaScript unterstützen viele der Operatoren in Director gemeinsam, während andere nur von der einen oder der anderen Sprache unterstützt werden.

Es folgen verschiedene Typen von Operatoren:

- **Arithmetische Operatoren** (z. B. `+`, `-`, `/`, und `*`)
- **Vergleichsoperatoren** (z. B. `<`, `>`, und `>=`), die zwei Argumente miteinander vergleichen
- **Logische Operatoren** (`not`, `and` und `or`), die einfache Bedingungen zu Bedingungssätzen verknüpfen
- **Zeichenfolgenoperatoren** (z. B. `&`, `&&` und `+`), die Zeichenfolgen miteinander verbinden

Hinweis: *JavaScript bietet weitaus mehr Operatortypen als Lingo, die jedoch nicht alle in dieser Referenz behandelt werden. Weitere Informationen zu diesen zusätzlichen Operatoren in JavaScript 1.5 finden Sie in den vielen Ressourcen anderer Anbieter zu diesem Thema.*

Die Elemente, auf die Operatoren angewendet werden, hießen *Operanden*. In Lingo gibt es nur binäre Operatoren. JavaScript unterstützt sowohl binäre als auch unäre Operatoren. Ein *binärer Operator* erfordert zwei Operanden, einen vor und einen hinter dem Operator. Ein *unärer Operator* benötigt nur einen Operanden, entweder vor oder hinter dem Operator.

Im ersten Beispiel veranschaulicht die erste Anweisung einen binären Operator, wobei die Variablen `x` und `y` Operanden sind und das Pluszeichen (`+`) der Operator ist. Die zweite Anweisung veranschaulicht einen unären Operator, wobei die Variable `i` der Operand und `++` der Operator ist.

```
// JavaScript syntax
x + y; // binary operator
i++; // unary operator
```

Weiterführende Informationen zu Operatoren finden Sie unter „Operatoren“ auf Seite 738

Grundlegendes zur Operatorpriorität

Wenn zwei oder mehr Operatoren in derselben Anweisung enthalten sind, bestimmt eine genau festgelegte Hierarchie, in welcher Reihenfolge sie ausgeführt werden. Diese Hierarchie wird als *Prioritätsordnung* von Operatoren bezeichnet. Multiplikationen werden z. B. immer vor Additionen ausgeführt. Allerdings haben in Klammern stehende Elemente Vorrang vor der Multiplikation. Ohne Klammern würde die Multiplikation in der folgenden Anweisung zuerst ausgeführt:

```
-- Lingo syntax
total = 2 + 4 * 3 -- results in a value of 14
```

Steht die Addition jedoch in Klammern, erfolgt diese zuerst:

```
-- Lingo syntax
total = (2 + 4) * 3 -- results in a value of 18
```

Eine Beschreibung der Operatoren und ihrer Prioritätsordnung finden Sie in den folgenden Tabellen. Operatoren mit höherer Priorität werden zuerst ausgeführt. Ein Operator mit Priorität 5 wird beispielsweise vor einem Operator mit Priorität 4 ausgeführt. Operationen derselben Prioritätsstufe werden von links nach rechts durchgeführt.

Arithmetische Operatoren

Arithmetische Operatoren addieren, subtrahieren, multiplizieren und dividieren Werte oder führen andere arithmetische Operationen aus. Hierzu gehören beispielsweise auch Klammern und das Minuszeichen.

Operator	Funktion	Priorität
()	Gruppiert Operationen zur Steuerung der Priorität.	5
-	Kehrt das Vorzeichen einer Zahl um, der er vorangestellt wird.	5
*	Führt eine Multiplikation aus.	4
mod	(Nur Lingo) Führt eine Modulo-Operation aus.	4
/	Führt eine Division aus.	4
%	(Nur JavaScript-Syntax) Gibt den ganzzahligen Rest der Division zweier Operanden zurück.	4
++	(Nur JavaScript) Fügt diesem Operanden 1 hinzu. Gibt bei Verwenden als Präfixoperator (`++x`) den Wert seines Operanden nach Hinzufügen von 1 zurück. Gibt bei Verwenden als Suffixoperator (`x++`) den Wert seines Operanden vor Hinzufügen von 1 zurück.	4
--	(Nur JavaScript) Subtrahiert 1 vom Operanden. Der Rückgabewert entspricht dem des Erhöhungsoperators.	4
+	Führt eine Addition aus, wenn er zwischen zwei Zahlen gestellt wird.	3
-	Führt eine Subtraktion aus, wenn er zwischen zwei Zahlen gestellt wird.	3

Hinweis: *Wenn in Lingo bei einer Operation nur ganzzahlige Werte verwendet werden, ist das Ergebnis ebenfalls eine Ganzzahl. Die Verwendung von Ganz- und Fließkommazahlen in derselben Berechnung ergibt eine Fließkommazahl. In JavaScript resultieren sämtliche Berechnungen im Wesentlichen in Fließkommazahlen.*

Wenn sich beim Teilen eines ganzzahligen Wertes durch einen anderen ganzzahligen Wert keine Ganzzahl ergibt, rundet Lingo das Ergebnis auf die nächstniedrigere Ganzzahl ab. Das Ergebnis von `4/3` ist z. B. `1`. Bei JavaScript wird der tatsächliche Fließkommawert 1,333 zurückgegeben.

Wenn Lingo einen Wert berechnen soll, ohne das Ergebnis zu runden, wenden Sie `float()` auf einen oder mehrere Werte in einem Ausdruck an. Das Ergebnis von `4/float(3)` ist z. B. 1,333.

Vergleichsoperatoren

Vergleichsoperatoren vergleichen zwei Werte und stellen fest, ob der Vergleich wahr oder falsch ist.

Operator	Bedeutung	Priorität
==	(Nur JavaScript) Zwei Operanden sind gleich. Wenn die Operanden nicht denselben Datentyp haben, versucht JavaScript die Operanden in einen für den Vergleich geeigneten Datentyp umzuwandeln.	1
===	(Nur JavaScript) Zwei Operanden sind gleich und haben denselben Datentyp.	1
!=	(Nur JavaScript) Zwei Operanden sind ungleich. Wenn die Operanden nicht denselben Datentyp haben, versucht JavaScript die Operanden in einen für den Vergleich geeigneten Datentyp umzuwandeln.	1
!==	(Nur JavaScript) Zwei Operanden sind ungleich und/oder haben nicht denselben Datentyp.	1
<>	(Nur Lingo) Zwei Operanden sind ungleich.	1
<	Der linke Operand ist kleiner als der rechte.	1

Operator	Bedeutung	Priorität
<=	Der linke Operand ist kleiner gleich dem rechten.	1
>	Der linke Operand ist größer gleich dem rechten.	1
>=	Der linke Operand ist größer gleich dem rechten.	1
=	(Nur Lingo) Zwei Operanden sind gleich.	1

Zuweisungsoperatoren

Ein *Zuweisungsoperator* weist seinem linken Operanden basierend auf dem Wert des rechten Operanden einen Wert zu. Mit Ausnahme des Basiszuweisungsoperators (=) gelten die folgenden Zuweisungsoperatoren nur für JavaScript.

Operator	Bedeutung	Priorität
=	Gleich	1
x += y	(Nur JavaScript) x = x + y	1
x -= y	(Nur JavaScript) x = x - y	1
x *= y	(Nur JavaScript) x = x * y	1
x /= y	(Nur JavaScript) x = x / y	1
x %= y	(Nur JavaScript) x = x % y	1

Logische Operatoren

Logische Operatoren testen, ob zwei logische Ausdrücke wahr oder falsch sind.

Gehen Sie sorgsam vor, wenn Sie in Lingo- oder JavaScript-Syntax logische und Zeichenfolgenoperatoren verwenden. Beispiele: In JavaScript ist `&&` ein logischer Operator, der bestimmt, ob zwei Ausdrücke wahr sind. Doch in Lingo ist `&&` ein Zeichenfolgenoperator, der zwei Zeichenfolgen verbindet und ein Leerzeichen zwischen die beiden Ausdrücke setzt.

Operator	Funktion	Priorität
and	(Nur Lingo) Prüft, ob beide Ausdrücke wahr sind.	4
&&	(Nur JavaScript) Prüft, ob beide Ausdrücke wahr sind.	4
oder	(Nur Lingo) Prüft, ob ein oder beide Ausdrücke wahr sind.	4
||	(Nur JavaScript) Prüft, ob ein oder beide Ausdrücke wahr sind.	4
not	(Nur Lingo) Negiert einen Ausdruck.	5
!	(Nur JavaScript) Negiert einen Ausdruck.	5

Der Operator `not` (Lingo) oder „!“ (JavaScript) wird verwendet, um einen `TRUE`- oder `FALSE`-Wert umzukehren. Die folgende Anweisung beispielsweise schaltet den Sound ein, wenn er ausgeschaltet ist, und schaltet ihn aus, wenn er eingeschaltet ist:

```
-- Lingo syntax
_sound.soundEnabled = not (_sound.soundEnabled)

// JavaScript syntax
_sound.soundEnabled = !(_sound.soundEnabled);
```

Zeichenfolgenoperatoren

Zeichenfolgenoperatoren dienen zum Verketten und Definieren von Zeichenfolgen.

Gehen Sie sorgsam vor, wenn Sie in Lingo- oder JavaScript-Syntax logische und Zeichenfolgenoperatoren verwenden. Beispiele: In JavaScript ist `&&` ein logischer Operator, der bestimmt, ob zwei Ausdrücke wahr sind. Doch in Lingo ist `&&` ein Zeichenfolgenoperator, der zwei Zeichenfolgen verbindet und ein Leerzeichen zwischen die beiden Ausdrücke setzt.

Operator	Funktion	Priorität
&	(Nur Lingo) Verkettet zwei Zeichenfolgen.	2
+	(Nur JavaScript) Verkettet zwei Zeichenfolgenwerte und gibt eine dritte Zeichenfolge zurück, welche die Vereinigung der beiden Operanden darstellt.	2
+=	(Nur JavaScript) Verkettet eine Zeichenfolgenvariable mit einer Zeichenfolgenvariablen und weist den Rückgabewert der Zeichenfolgenvariablen zu.	2
&&	(Nur Lingo) Verkettet zwei Zeichenfolgen und fügt ein Leerzeichen zwischen ihnen ein.	2
"	Markiert den Anfang oder das Ende einer Zeichenfolge.	1

Bedingte Konstrukte

Director führt Skriptanweisungen stets nach demselben Schema aus: Das Programm beginnt mit der ersten Anweisung und arbeitet die Anweisungen der Reihe nach ab, bis es auf die letzte Anweisung oder eine Anweisung trifft, die ein Skript veranlasst, an einer anderen Stelle fortzufahren.

Anweisungen sind in der Reihenfolge anzugeben, in der sie ausgeführt werden sollen. Wenn Sie z. B. eine Anweisung schreiben, die einen berechneten Wert benötigt, müssen Sie zuerst die Anweisung angeben, die den Wert berechnet.

Im folgenden Beispiel werden in der ersten Anweisung zwei Zahlen addiert, deren Summe dann in der zweiten Anweisung als Zeichenfolgendarstellung einem auf der Bühne angezeigten Felddarsteller mit dem Namen `Answer` zugewiesen wird. Die zweite Anweisung konnte nicht vor der ersten platziert werden, da die Variable `x` noch nicht definiert wurde.

```
-- Lingo syntax
x = 2 + 2
member("Answer").text = string(x)

// JavaScript syntax
var x = 2 + 2;
member("Answer").text = x.toString();
```

Sowohl Lingo als auch JavaScript bieten Konventionen für das Ändern der Standardausführungsreihenfolge oder von Skriptanweisungen sowie zum Ausführen von Aktionen basierend auf bestimmten Bedingungen. Sie möchten ggf. beispielsweise Folgendes in Ihren Skripts ausführen:

- Einen Satz mit Anweisungen ausführen, wenn eine logische Bedingung wahr ist, oder alternative Anweisungen ausführen, wenn die logische Bedingung falsch ist.
- Einen Ausdruck auswerten und versuchen, den Wert des Ausdrucks mit einer bestimmten Bedingung abzugleichen.
- Einen Satz mit Anweisungen wiederholt so lange ausführen, bis eine bestimmte Bedingung erfüllt ist.

Testen auf logische Bedingungen

Um eine Anweisung oder einen Satz mit Anweisungen auszuführen, wenn eine angegebene Bedingung wahr oder falsch ist, verwenden Sie die Struktur `if...then...else` (Lingo) oder `if...else` (JavaScript). Sie können z. B. eine `if...then...else`- oder `if...then`-Struktur erstellen, die prüft, ob ein Text vollständig aus dem Internet heruntergeladen wurde, und falls dies der Fall ist, versucht, den Text zu formatieren. Diese Strukturen verwenden das folgende Muster für das Testen auf logische Bedingungen:

- (Lingo- und JavaScript-Syntax) Anweisungen, die prüfen, ob eine Bedingung wahr oder falsch ist, beginnen mit dem Element `if`.
- Wenn in Lingo die Bedingung erfüllt ist, werden die Anweisungen ausgeführt, die auf `then` folgen. In JavaScript-Syntax nehmen geschweifte Klammer (`{ }`) den Platz des Lingo-Elements `then` ein, die jede einzelne `if`-, `else`- und `else if`-Anweisung umgeben müssen.
- Wenn in Lingo- und JavaScript-Syntax die Bedingung nicht vorhanden ist, springen Skripts mithilfe der Elemente `else` oder `else if` zur nächsten Anweisung in der Prozedur.
- In Lingo gibt das Element `end if` das Ende des `if`-Tests an. In JavaScript endet der `if`-Test automatisch, weshalb es kein Element gibt, das den Test explizit beendet.

Sie können die Leistung eines Skripts optimieren, indem Sie zuerst auf die wahrscheinlichsten Bedingungen testen.

Die folgenden Anweisungen testet verschiedene Bedingungen. Das Element `else if` gibt alternative Tests an, die ausgeführt werden sollen, wenn die vorhergehenden Bedingungen nicht erfüllt sind.

```
-- Lingo syntax
if _mouse.mouseMember = member(1) then
    _movie.go("Cairo")
else if _mouse.mouseMember = member(2) then
    _movie.go("Nairobi")
else
    _player.alert("You're lost.")
end if

// JavaScript syntax
if (_mouse.mouseMember = member(1)) {
    _movie.go("Cairo");
}
else if (_mouse.mouseMember = member(2)) {
    _movie.go("Nairobi");
}
else {
    _player.alert("You're lost.");
}
```

Beim Schreiben von `if...then`-Strukturen in Lingo können Sie die auf `then` folgenden Anweisungen entweder in dieselbe Zeile setzen wie `then` oder mit einer Zeilenschaltung nach dem Wort `then` in die folgende Zeile übernehmen. Wenn Sie eine Zeilenschaltung einfügen, müssen Sie die `if...then`-Struktur mit einer `end if`-Anweisung abschließen.

Beim Schreiben von `if`-Strukturen in JavaScript können Sie die auf `if` folgenden Anweisungen entweder in dieselbe Zeile setzen wie `if` oder mit einer Zeilenschaltung nach dem Wort `if` in die folgende Zeile übernehmen.

Die folgenden Anweisungen sind gleichbedeutend:

```
-- Lingo syntax
if _mouse.mouseMember = member(1) then _movie.go("Cairo")

if _mouse.mouseMember = member(1) then
    _movie.go("Cairo")
end if

// JavaScript syntax
if (_mouse.mouseMember = member(1)) { _movie.go("Cairo"); }

if (_mouse.mouseMember = member(1)) {
    _movie.go("Cairo");
}
```

Weiterführende Informationen zum Arbeiten mit `if...then...else`- und `if...else`-Strukturen finden Sie unter if.

Auswerten und Abgleichen von Ausdrücken

Die Strukturen `case` (Lingo) oder `switch...case` (JavaScript) sind verkürzende Alternativen zum Verwenden von `if...then...else`- oder `if...then`-Strukturen, wenn Strukturen mit mehreren Verzweigungen eingerichtet werden sollen. `case`- und `switch...case`-Strukturen sind oftmals effizienter und einfacher zu lesen als eine große Anzahl von `if...then...else`- oder `if...then`-Strukturen.

Die zu prüfende Bedingung folgt in Lingo auf den Begriff `case` in der ersten Zeile der `case`-Struktur. Anschließend werden die einzelnen Zeilen der Reihe nach geprüft, bis Lingo auf einen Ausdruck trifft, der mit der Testbedingung übereinstimmt. Sobald ein passender Ausdruck gefunden ist, führt Director die auf diesen Ausdruck folgenden Lingo-Anweisungen aus.

Die zu prüfende Bedingung folgt in JavaScript auf den Begriff `switch` in der ersten Zeile der Struktur. Jeder Vergleich im Test folgt dem Element `case` für jede Zeile, die einen Test enthält. Jeder `case`-Vergleich kann mithilfe des optionalen Elements `break` beendet werden. Durch Einfügen des Elements `break` wird das Programm aus der `switch`-Struktur entfernt, woraufhin auf die Struktur folgende Anweisungen ausgeführt werden. Wird `break` weggelassen, wird folgender `case`-Vergleich ausgeführt.

In einer `case`- oder `switch...case`-Struktur können Vergleiche als Testbedingung verwendet werden.

Die folgenden `case`- oder `switch...case`-Strukturen ermitteln beispielsweise die vom Benutzer zuletzt gedrückte Taste und reagieren dementsprechend:

- Wenn der Benutzer A gedrückt hat, geht der Film zum Bild mit der Beschriftung „Apple“.
- Hat der Benutzer B oder C gedrückt, führt der Film den angegebenen Übergang aus und geht dann zum Bild mit der Beschriftung „Oranges“.
- Bei Drücken jeder anderen Buchstabentaste gibt der Computer einen Warnton aus.

```
-- Lingo syntax
case (_key.key) of
    "a" : _movie.go("Apple")
    "b", "c":
         _movie.puppetTransition(99)
         _movie.go("Oranges")
    otherwise: _sound.beep()
end case

// JavaScript syntax
switch (_key.key) {
    case "a" :
        _movie.go("Apple");
        break;
    case "b":
    case "c":
         _movie.puppetTransition(99);
        _movie.go("Oranges");
        break;
        default: _sound.beep()
}
```

Hinweis: *In JavaScript-Syntax kann pro* `case`*-Anweisung nur ein Vergleich erfolgen.*

Weitere Informationen zum Verwenden von `case`-Strukturen finden Sie unter case.

Sich wiederholende Aktionen

In Lingo- und JavaScript-Syntax können Sie eine Aktion mit bestimmter Häufigkeit wiederholen oder aber immer wieder ausführen, sofern eine bestimmte Bedingung erfüllt ist.

Um in Lingo eine Aktion mit bestimmter Häufigkeit zu wiederholen, verwenden Sie eine `repeat with`-Struktur und geben im Anschluss an `repeat with` die Anzahl der Wiederholungen in Form eines Bereichs an.

Um in JavaScript eine Aktion mit bestimmter Häufigkeit zu wiederholen, verwenden Sie die `for`-Struktur. Die `for`-Schleife benötigt drei Parameter. Der erste Parameter initialisiert meist eine Zählervariable, der zweite Parameter gibt eine Bedingung an, die bei jedem Durchlaufen der Schleife ausgewertet werden soll, und der dritte Parameter dient in der Regel zum Ändern bzw. Erhöhen der Zählervariablen.

Die `repeat with`- und `for`-Strukturen bieten sich an, wenn Sie denselben Vorgang auf eine Reihe von Objekten anwenden möchten. Die folgende Schleife zum Beispiel weist den Sprites 2 bis 10 den Farbeffekt „Hintergrund transparent“ zu:

```
-- Lingo syntax
repeat with n = 2 to 10
    sprite(n).ink = 36
end repeat

// JavaScript syntax
for (var n=2; n<=10; n++) {
    sprite(n).ink = 36;
}
```

Das folgende Beispiel führt zum selben Ergebnis, ändert die Sprites jedoch in umgekehrter Reihenfolge:

```
-- Lingo syntax
repeat with n = 10 down to 2
    sprite(n).ink = 36
end repeat

// JavaScript syntax
for (var n=10; n>=2; n--) {
    sprite(n).ink = 36;
}
```

Verwenden Sie in Lingo, um eine Reihe von Instruktionen zu wiederholen, so lange eine bestimmte Bedingung erfüllt ist, die `repeat while`-Struktur.

Verwenden Sie in JavaScript, um eine Reihe von Instruktionen zu wiederholen, solange eine bestimmte Bedingung erfüllt ist, die `while`-Struktur.

Die folgenden Anweisungen bewirken beispielsweise, dass der Film einen Warnton ausgibt, solange der Benutzer die Maustaste drückt:

```
-- Lingo syntax
repeat while _mouse.mouseDown
    _sound.beep()
end repeat

// JavaScript syntax
while (_mouse.mouseDown) {
    _sound.beep();
}
```

Lingo- und JavaScript-Skripts durchlaufen die Anweisungen innerhalb der Schleife so lange, bis die Bedingung nicht mehr erfüllt ist oder eine der Anweisungen auf eine Stelle außerhalb der Schleife verweist. Im vorherigen Beispiel beendet das Skript die Schleife, wenn der Benutzer die Maustaste freigibt, da dann die Bedingung `mouseDown` nicht mehr erfüllt ist.

Verwenden Sie in Lingo zum Verlassen einer Schleife die Anweisung `exit repeat`.

Um in JavaScript eine Schleife zu verlassen, geben Sie das Element `break` an. Eine Schleife wird automatisch verlassen, wenn eine Bedingung nicht wahr ist.

Die folgenden Anweisungen bewirken zum Beispiel, dass der Film einen Warnton ausgibt, wenn der Benutzer die Maustaste drückt, es sei denn, der Cursor befindet sich auf Sprite 1. In diesem Fall beendet das Skript die Wiederholungsschleife und die Ausgabe von Warntönen. Die `rollover()`-Methode gibt an, ob sich der Cursor über dem angegebenen Sprite befindet.

```
-- Lingo syntax
repeat while _mouse.stillDown
    _sound.beep()
    if _movie.rollOver(1) then exit repeat
end repeat

// JavaScript syntax
while (_mouse.stillDown) {
    _sound.beep();
    if (_movie.rollOver(1)) {
        break;
    }
}
```

Weiterführende Informationen zu `repeat while`- und `while`-Strukturen finden Sie unter repeat while.

Ereignisse, Nachrichten und Prozeduren

Leistungsstarke, nützliche Skripts können Sie nur erstellen, wenn Sie mit den Konzepten und der Funktionsweise von Ereignissen, Nachrichten und Prozeduren vertraut sind. Wenn Sie die Reihenfolge kennen, in der Ereignisse und Nachrichten gesendet und empfangen werden, können Sie exakt bestimmen, wann bestimmte Skripts oder Teile davon ausgeführt werden sollen. Sie können außerdem Skripts besser debuggen, wenn bestimmte Aktionen nicht zum gewünschten Zeitpunkt erfolgen.

Bei der Wiedergabe eines Films passiert Folgendes:

- Ereignisse erfolgen als Reaktion auf entweder eine vom System oder vom Benutzer angegebenen Aktion.
- Nachrichten, die zu diesen Ereignissen gehören, werden an die Skripts in einem Film gesendet.
- Prozeduren in Skripts enthalten die Instruktionen, die bei Empfang einer bestimmten Nachricht ausgeführt werden.

Der Name eines Ereignisses entspricht dem Namen der Nachricht, die es generiert. Die Prozedur, die das Ereignis verarbeitet, entspricht sowohl dem Ereignis als auch der Nachricht. Wenn beispielsweise das Ereignis `mouseDown` eintritt, generiert und sendet Director an Skripts die Nachricht `mouseDown`, die wiederum von der Prozedur `mouseDown` verarbeitet wird.

Ereignisse

Es gibt zwei Kategorien von Ereignissen, die bei der Wiedergabe eines Films eintreten:

- **Systemereignisse** treten ein, ohne dass ein Benutzer mit dem Film interagiert, und sind in Director vordefiniert und benannt, z. B. wenn der Abspielkopf in ein Bild eintritt, wenn auf ein Sprite geklickt wird usw.
- **Benutzerdefinierte Ereignisse** treten als Reaktion auf Aktionen ein, die Sie festlegen. Sie können beispielsweise ein Ereignis erstellen, das eintritt, wenn sich die Hintergrundfarbe eines Sprites von Rot in Blau ändert, wenn ein Sound fünfmal wiedergegeben wurde usw.

  Viele Systemereignisse wie `prepareFrame`, `beginSprite` usw. treten bei Wiedergabe eines Films automatisch und in vordefinierter Reihenfolge ein. Andere Systemereignisse, insbesondere Mausereignisse wie `mouseDown`, `mouseUp` usw., erfolgen bei Wiedergabe eines Films nicht unbedingt automatisch, sondern wenn ein Benutzer sie auslöst.

  Wenn ein Film erstmals gestartet wird, tritt stets zuerst das Ereignis `prepareMovie`, dann das Ereignis `prepareFrame` usw. ein. Die Ereignisse `mouseDown` und `mouseUp` kommen ggf. nie in einem Film vor, es sei denn, sie werden vom Benutzer durch Klicken auf den Film ausgelöst.

  Die folgenden Listen enthalten die Systemereignisse, die stets während eines Films eintreten, in der entsprechenden Reihenfolge.

Viele Systemereignisse wie `prepareFrame`, `beginSprite` usw. treten bei Wiedergabe eines Films automatisch und in vordefinierter Reihenfolge ein. Andere Systemereignisse, insbesondere Mausereignisse wie `mouseDown`, `mouseUp` usw., erfolgen bei Wiedergabe eines Films nicht unbedingt automatisch, sondern wenn ein Benutzer sie auslöst.

Wenn ein Film erstmals gestartet wird, tritt stets zuerst das Ereignis `prepareMovie`, dann das Ereignis `prepareFrame` usw. ein. Die Ereignisse `mouseDown` und `mouseUp` kommen ggf. nie in einem Film vor, es sei denn, sie werden vom Benutzer durch Klicken auf den Film ausgelöst.

Die folgenden Listen enthalten die Systemereignisse, die stets während eines Films eintreten, in der entsprechenden Reihenfolge.

Beim Starten des Films finden Ereignisse in der folgenden Reihenfolge statt:

1 `prepareMovie`

2 `prepareFrame` Unmittelbar nach dem Ereignis `prepareFrame` spielt Director Sounds ab, zeichnet Sprites und führt Übergänge oder Paletteneffekte aus. Dieses Ereignis findet vor dem Ereignis `enterFrame` statt. Eine `prepareFrame`-Prozedur eignet sich für Skripts, die ausgeführt werden sollen, bevor das Bild gezeichnet wird.

3 `beginSprite` – Dieses Ereignis findet statt, wenn der Abspielkopf in den Bildeinschluss eines Sprites eintritt.

4 `startMovie` – Dieses Ereignis findet im ersten abgespielten Bild statt.

Wenn der Film auf ein Bild trifft, finden Ereignisse in der folgenden Reihenfolge statt:

1 `beginSprite` – Dieses Ereignis findet nur dann statt, wenn neue Sprites im Bild beginnen.

2 `stepFrame`

3 `enterFrame` – Zwischen den Ereignissen `enterFrame` und `exitFrame` verarbeitet Director alle Zeitverzögerungen, die aufgrund von Tempoeinstellungen, Leerlaufereignissen sowie Tasten- oder Mausereignissen erforderlich werden.

4 `exitFrame`

5 `endSprite` – Dieses Ereignis findet nur statt, wenn der Abspielkopf ein Sprite im Bild verlässt.

Beim Anhalten des Films finden Ereignisse in der folgenden Reihenfolge statt:

1 `endSprite` – Dieses Ereignis findet nur statt, wenn der Film zum jeweiligen Zeitpunkt Sprites enthält.

2 `stopMovie`

Nachrichten

Um die ordnungsgemäßen Skriptanweisungen zum richtigen Zeitpunkt auszuführen, muss Director feststellen können, was im Film geschieht und welche Anweisungen als Reaktion auf bestimmte Ereignisse ausgeführt werden sollen.

Director sendet *Nachrichten*, die bestimmte im Film eingetretene Ereignisse angeben, z. B. wenn der Benutzer auf ein Sprite klickt oder eine Taste betätigt, ein Film gestartet wird, der Abspielkopf in ein Bild eintritt oder ein Bild verlässt oder ein Skript ein bestimmtes Ergebnis zurückgibt.

Nachrichten werden in der folgenden allgemeinen Reihenfolge an Objekte gesendet:

1 Zuerst wird eine Nachricht an die Verhalten gesendet, die einem an dem betreffenden Ereignis beteiligten Sprite zugeordnet sind. Weist ein Sprite mehrere Verhalten auf, reagieren diese in derselben Reihenfolge auf die Nachricht, in der sie dem Sprite zugeordnet wurden.

2 Als nächstes wird die Nachricht an ein Skript gesendet, das an dem dem Sprite zugeordneten Darsteller angebracht ist.

3 Dann wird die Nachricht an die dem aktuellen Bild zugeordneten Verhalten gesendet.

4 Zuletzt erhalten die Filmskripten die Nachricht.

Sie können zwar eigene Nachrichtennamen definieren, doch die meisten Ereignisse, die in Filmen häufig auftreten, erzeugen Nachrichten mit vorgegebenen Namen.

Weiterführende Informationen zu den vordefinierten Nachrichten in Director finden Sie unter „Ereignisse und Nachrichten“ auf Seite 184.

Festlegen benutzerdefinierter Nachrichten

Sie können nicht nur die vordefinierten Nachrichtennamen verwenden, sondern auch eigene Nachrichten- und Prozedurnamen definieren. Eine benutzerdefinierte Nachricht kann ein anderes Skript, eine andere Prozedur oder die eigene Prozedur der Anweisung aufrufen. Wenn die Ausführung der aufgerufenen Prozedur beendet ist, wird die Prozedur fortgesetzt, durch die sie aufgerufen wurde.

Ein benutzerdefinierter Nachrichten- und Prozedurname muss folgende Kriterien erfüllen:

- Muss mit einem Buchstaben beginnen.
- Darf nur aus alphanumerischen Zeichen bestehen und keine Sonder- oder Satzzeichen enthalten.
- Er muss entweder aus einem oder aus mehreren durch Unterstriche miteinander verbundenen Wörtern bestehen – Leerzeichen sind nicht zulässig.
- Darf nicht bereits von einem anderen vordefinierten Lingo- oder JavaScript-Syntaxelement belegt sein.

Das Verwenden vordefinierter Lingo- oder JavaScript-Schlüsselwörter für Nachricht- und Prozedurnamen kann für Verwirrung sorgen. Es ist zwar möglich, die Funktionalität eines Lingo-Elements auf diese Weise explizit zu ersetzen oder zu erweitern, doch sollten Sie dies nur in bestimmten Ausnahmesituationen tun.

Wenn Sie über mehrere Prozeduren mit ähnlichen Funktionen verfügen, sollten Sie ihnen Namen mit gleichlautendem Wortanfang zuzuordnen, damit sie in einer alphabetischen Liste (z. B. in der Liste unter „Bearbeiten„ > „Suchen“ > „Prozedur“) nahe beieinander aufgeführt werden.

Prozeduren

Prozeduren sind Anweisungssätze innerhalb eines Skripts, die als Reaktion auf ein bestimmtes Ereignis und eine nachfolgende Nachricht ausgeführt werden. Wenngleich Director vordefinierte Ereignisse und Nachrichten enthält, müssen Sie eigene benutzerdefinierte Prozeduren für jedes Ereignis-/Nachricht-Paar erstellen, das Sie verarbeiten möchten.

Bestimmen, wo Prozeduren platziert werden sollen

Sie können Prozeduren in jeder Art von Skript platzieren, und ein Skript kann mehrere Prozeduren enthalten. Um die Übersicht zu wahren, empfiehlt es sich jedoch, zusammengehörige Prozeduren an einer Stelle zu gruppieren.

Hier einige allgemeine Hinweise:

- Um eine Prozedur mit einem bestimmten Sprite zu verknüpfen oder dafür zu sorgen, dass eine Prozedur als Reaktion auf eine bestimmte auf ein Sprite angewendete Aktion ausgeführt wird, fügen Sie die Prozedur einem Verhalten hinzu, das dem Sprite zugeordnet ist.
- Stellen Sie eine Prozedur, die immer verfügbar sein soll, wenn sich der Film in einem bestimmten Bild befindet, in ein Verhalten, das an diesem Bild angebracht ist.

  Wenn eine Prozedur beispielsweise auf einen Mausklick reagieren soll, während sich der Abspielkopf in einem Bild befindet, ganz gleich, auf welche Stelle der Benutzer klickt, müssen Sie eine `mouseDown`- oder `mouseUp`-Prozedur dem Bildverhalten und nicht einem Sprite-Verhalten hinzufügen.
- Um eine Prozedur einzurichten, die als Reaktion auf Nachrichten zu beliebigen Ereignissen im Film ausgeführt werden soll, fügen Sie die Prozedur einem Filmskript hinzu.
- Um eine Prozedur einzurichten, die als Reaktion auf ein Ereignis ausgeführt wird, das einen Darsteller betrifft, unabhängig davon, welche Sprites den Darsteller verwenden, fügen Sie die Prozedur einem Darstellerskript hinzu.

Festlegen, wann Prozeduren eine Nachricht erhalten

Nach dem Senden einer Nachricht an Skripts sucht Director in einer festgelegten Reihenfolge nach Prozeduren.

1. Director prüft zuerst, ob eine Prozedur im Objekt vorhanden ist, das die Nachricht gesendet hat. Wird eine Prozedur gefunden, wird die Nachricht abgefangen und das Skript in der Prozedur ausgeführt.
2. Wird keine Prozedur gefunden, überprüft Director zuerst die Darsteller in aufsteigender Reihenfolge auf verknüpfte Filmskripten, die ggf. eine Prozedur für die Nachricht enthalten. Wird eine Prozedur gefunden, wird die Nachricht abgefangen und das Skript in der Prozedur ausgeführt.
3. Wird keine Prozedur gefunden, prüft Director anschließend, ob ein Bildskript eine Prozedur für die Nachricht enthält. Wird eine Prozedur gefunden, wird die Nachricht abgefangen und das Skript in der Prozedur ausgeführt.
4. Wird keine Prozedur gefunden, überprüft Director zuerst Sprites in aufsteigender Reihenfolge auf Skripts, die mit den Sprites verknüpft sind und ggf. eine Prozedur für die Nachricht enthalten. Wird eine Prozedur gefunden, wird die Nachricht abgefangen und das Skript in der Prozedur ausgeführt.

Wenn eine Prozedur eine Nachricht abgefangen hat, wird diese Nachricht nicht mehr automatisch an die übrigen Objekte weitergeleitet. In Lingo können Sie allerdings mit der `pass()`-Methode diese Standardeinstellung übergehen und die Nachricht auch an andere Objekte übergeben.

Wird keine passende Prozedur gefunden, nachdem die Nachricht an alle möglichen Objekte übergeben wurde, wird sie von Director ignoriert.

Die genaue Reihenfolge der Objekte, an die Director eine Nachricht sendet, ist von der Nachricht abhängig. Einzelheiten zu der Reihenfolge, in der Director bestimmte Nachrichten an die verschiedenen Objekte sendet, finden Sie unter dem Eintrag zu der jeweiligen Nachricht unter „Ereignisse und Nachrichten“ auf Seite 184.

Übergeben von Werten an eine Prozedur mithilfe von Parametern

Indem Sie Werte mithilfe von Parametern übergeben, können Sie einer Prozedur genau die Werte bereitstellen, die sie zu einem bestimmten Zeitpunkt benötigt, unabhängig davon, wo oder wann Sie die Prozedur im Film aufrufen. Parameter können je nach Situation optional oder erforderlich sein.

- Setzen Sie in Lingo die Parameter hinter den Prozedurnamen.
- Setzen Sie in JavaScript die Parameter in Klammern und anschließend hinter den Prozedurnamen.

Trennen Sie mehrere Parameter durch Kommas.

Beim Aufruf einer Prozedur müssen Sie bestimmte Werte für die Parameter angeben, die die Prozedur verwendet. Hierbei können Sie jede Art von Wert verwenden, z. B. eine Zahl, eine Variable, der ein Wert zugeordnet ist, oder eine Zeichenfolge. Werte in der aufrufenden Anweisung müssen in Klammern stehen und in derselben Reihenfolge aufgeführt sein wie in den Parametern der Prozedur.

Im folgenden Beispiel ruft die Variablenzuweisung `mySum` die `addThem`-Methode auf, an welche die beiden Werte 2 und 4 übergeben werden. Die `addThem`-Prozedur ersetzt die Parameterplatzhalter `a` und `b` durch die beiden übergebenen Werte, speichert das Ergebnis in der lokalen Variablen `c` und verwendet anschließend das Schlüsselwort `return`, um das Ergebnis an die ursprüngliche Methode zurückzugeben, das anschließend `mySum` zugewiesen wird.

Da 2 in der Liste der Parameter an erster Stelle steht, ersetzt dieser Wert `a` in der Prozedur. Da 4 in der Liste der Parameter an zweiter Stelle steht, ersetzt dieser Wert `b` in der Prozedur.

```
-- Lingo syntax
mySum = addThem(2, 4) -- calling statement

on addThem a, b -- handler
    c = a + b
    return c -- returns the result to the calling statement
end

// JavaScript syntax
var mySum = addThem(2, 4); // calling statement

function addThem(a, b) { // handler
    c = a + b;
    return c; // returns the result to the calling statement
}
```

Wenn Sie in Lingo eine benutzerdefinierte Methode für ein Objekt aufrufen, wird ein Verweis auf das Script-Objekt im Speicher stets als implizierter erster Parameter an die Prozedur dieser Methode übergeben. Dies bedeutet, dass Sie den Script-Objektverweis in Ihrer Prozedur berücksichtigen müssen.

Angenommen, Sie haben die benutzerdefinierte Sprite-Methode `jump()` geschrieben, die eine einzelne Ganzzahl als Parameter verwendet, und die Methode einem Verhalten hinzugefügt. Wenn Sie `jump()` in einem Sprite-Objektverweis aufrufen, muss die Prozedur auch einen Parameter enthalten, der den Script-Objektverweis und nicht bloß die einzelne Ganzzahl repräsentiert. In diesem Fall wird der implizierte Parameter vom Schlüsselwort `me` repräsentiert, wobei aber jedes beliebige Element funktioniert.

```
-- Lingo syntax
myHeight = sprite(2).jump(5)

on jump(me,a)
    return a + 15 -- this handler works correctly, and returns 20
end

on jump(a)
    return a + 15 -- this handler does not work correctly, and returns 0
end
```

Sie können auch Ausdrücke als Werte verwenden. Die folgende Anweisung ersetzt beispielsweise `a` durch `3+6` und `b` durch `8>2` (bzw. `1` für `TRUE`) und gibt den Wert `10` zurück:

```
-- Lingo syntax
mySum = addThem(3+6, 8>2)
```

In Lingo beginnt eine Prozedur immer mit dem Wort `on`, gefolgt von der Nachricht, auf die die Prozedur reagieren soll. Die letzte Zeile der Prozedur enthält das Wort `end`. Sie können hinter `end` den Prozedurnamen wiederholen, was jedoch optional ist.

In JavaScript beginnt eine Prozedur immer mit dem Wort `function`, gefolgt von der Nachricht, auf die die Prozedur reagieren soll. Die Anweisungen, welche die Prozedur bilden, sind wie alle JavaScript-Funktionen von öffnenden und schließenden Klammern umgeben.

Zurückgeben von Ergebnissen aus Prozeduren

In vielen Fällen soll eine Prozedur dazu dienen, eine Bedingung oder das Ergebnis einer Aktion zu melden.

- Verwenden Sie das Schlüsselwort `return`, über das eine Prozedur eine Bedingung oder das Ergebnis einer Aktion meldet. Die folgende Prozedur `findColor` gibt z. B. die aktuelle Farbe von Sprite 1 zurück:

```
-- Lingo syntax
on findColor
    return sprite(1).foreColor
end

// JavaScript syntax
function findColor() {
    return(sprite(1).foreColor);
}
```

Sie können das Schlüsselwort `return` auch eigenständig verwenden, um die aktuelle Prozedur zu verlassen und keinen Wert zurückzugeben. Die folgende `jump`-Prozedur gibt nichts zurück, wenn der Parameter `aVal` gleich 5 ist. Andernfalls gibt sie einen Wert zurück:

```
-- Lingo syntax
on jump(aVal)
    if aVal = 5 then return

    aVal = aVal + 10
    return aVal
end
// JavaScript syntax
function jump(aVal) {
    if(aVal == 5) {
        return;
    }
    else {
        aVal = aVal + 10;
        return(aVal);
    }
}
```

Wenn Sie eine Prozedur, die ein Ergebnis zurückgibt, von einer anderen Prozedur aus aufrufen, müssen Sie nach dem Prozedurnamen eine Klammer setzen. Die Anweisung `put(findColor())` ruft beispielsweise die Prozedur `on findColor` auf und zeigt das Ergebnis im Nachrichtenfenster an.

Lineare Listen und Eigenschaftslisten

Sie möchten in Ihren Skripts ggf. Listen mit Daten, z. B. Namen oder einem Variablensatz zugewiesene Werte, nachverfolgen und aktualisieren. Sowohl Lingo als auch JavaScript bieten Zugriff auf lineare und Eigenschaftslisten. In einer linearen Liste ist jedes Element ein einzelner Wert. In einer Eigenschaftsliste besteht jedes Element aus zwei Werten. Der erste Wert ist ein Eigenschaftsname, der zweite ist der mit dieser Eigenschaft verbundene Wert.

Da sowohl Lingo als auch JavaScript Zugriff auf lineare und Eigenschaftslisten bieten, sollten Sie diese Listen nutzen, wenn Werte in Ihrem Code sowohl von Lingo- als auch JavaScript-Skripts gemeinsam verwendet werden.

Wenn Werte in Ihrem Code nur von JavaScript-Skripts verwendet werden, sollten Sie für das Arbeiten mit Datenlisten JavaScript-Objekte vom Typ „Array“ verwenden. Weitere Informationen über den Gebrauch von Arrays finden Sie unter „JavaScript-Arrays“ auf Seite 41.

Erstellen linearer Listen

So erstellen Sie eine lineare Liste

- Verwenden Sie in Lingo entweder die Funktion oberster Ebene `list()` oder den Listenoperator (`[ ]`) sowie Kommas zum Trennen der Listeneinträge.
- Verwenden Sie in JavaScript die Funktion oberster Ebene `list()` sowie Kommas zum Trennen der Listeneinträge.

Der Index einer linearen Liste beginnt stets mit 1.

Wenn Sie die Funktion oberster Ebene `list()` verwenden, geben Sie die Listeneinträge als Parameter der Funktion an. Diese Funktion bietet sich an, wenn auf Ihrer Tastatur keine eckigen Klammern vorhanden sind.

Alle folgenden Anweisungen erstellen eine lineare Liste mit drei Namen und weisen dieser einer Variablen zu:

```
-- Lingo syntax
workerList = ["Bruno", "Heather", "Carlos"] -- using the Lingo list operator
workerList = list("Bruno", "Heather", "Carlos") -- using list()

// JavaScript syntax
var workerList = list("Bruno", "Heather", "Carlos"); // using list()
```

Sie können auch leere lineare Listen erstellen (siehe die folgenden Anweisungen):

```
-- Lingo syntax
workerList = [] -- using the Lingo list operator
workerList = list() -- using list() with no parameters

// JavaScript syntax
var workerList = list(); // using list() with no parameters
```

Erstellen von Eigenschaftslisten

So erstellen Sie eine Eigenschaftsliste

- Verwenden Sie in Lingo entweder die Funktion oberster Ebene `propList()` oder den Listenoperator (`[:]`). Bei Verwenden des Listenoperators zum Erstellen einer Eigenschaftsliste können Sie entweder einen Doppelpunkt zum Kennzeichnen von Name/Wert-Einträgen oder Kommas zum Trennen der Einträge in der Liste oder aber Kommas verwenden, um sowohl Name/Wert-Einträge zu kennzeichnen als auch Einträge in der Liste zu trennen.
- Verwenden Sie in JavaScript die Funktion oberster Ebene `propList()`, und fügen Sie Kommas ein, um sowohl Name/Wert-Einträge zu kennzeichnen als auch Einträge in der Liste zu trennen.

Wenn Sie die Funktion oberster Ebene `propList()` verwenden, geben Sie die Eigenschaftslisteneinträge als Parameter der Funktion an. Diese Funktion bietet sich an, wenn auf Ihrer Tastatur keine eckigen Klammern vorhanden sind.

Eigenschaften können in einer Eigenschaftsliste mehrfach vorhanden sein.

Alle folgenden Anweisungen erstellen eine Eigenschaftsliste mit vier Eigenschaftsnamen (`left`, `top`, `right` und `bottom`) und den dazugehörigen Werten:

```
-- Lingo syntax
sprite1Loc = [#left:100, #top:150, #right:300, #bottom:350]
sprite1Loc = ["left",400, "top",550, "right",500, "bottom",750]
sprite1Loc = propList("left",400, "top",550, "right",500, "bottom",750)

// JavaScript syntax
var sprite1Loc = propList("left",400, "top",550, "right",500, "bottom",750);
```

Sie können auch leere Eigenschaftslisten erstellen (siehe die folgenden Anweisungen):

```
-- Lingo syntax
sprite1Loc = [:] -- using the Lingo property list operator
sprite1Loc = propList() -- using propList() with no parameters

// JavaScript syntax
var sprite1Loc = propList(); // using propList() with no parameters
```

Festlegen und Abrufen von Listeneinträgen

Sie können einzelne Einträge in einer Liste festlegen und abrufen. Hierbei wird für lineare und Eigenschaftslisten eine unterschiedliche Syntax verwendet.

So geben Sie einen Wert in einer linearen Liste an

Führen Sie einen der folgenden Schritte aus:

- Verwenden Sie den Ist-gleich-Operator (=).
- Verwenden Sie die `setAt()`-Methode.

Die folgenden Anweisungen veranschaulichen die Definition der linearen Liste `workerList`, die den einen Wert `Heather` enthält und der als zweiter Wert `Carlos` hinzugefügt wird:

```
-- Lingo syntax
workerList = ["Heather"] -- define a linear list
workerList[2] = "Carlos" -- set the second value using the equal operator
workerList.setAt(2, "Carlos") -- set the second value using setAt()

// JavaScript syntax
var workerList = list("Heather"); // define a linear list
workerList[2] = "Carlos"; // set the second value using the equal operator
workerList.setAt(2, "Carlos"); // set the second value using setAt()
```

So rufen Sie einen Wert aus einer linearen Liste ab

1. Verwenden Sie die Listenvariable, gefolgt von der Nummer der Listenposition des gewünschten Wertes in eckigen Klammern.
2. Verwenden Sie die `getAt()`-Methode.

Die folgenden Anweisungen erstellen die lineare Liste `workerList` und weisen anschließend den zweiten Wert in der Liste der Variable `name2` zu:

```
-- Lingo syntax
workerList = ["Bruno", "Heather", "Carlos"] -- define a linear list
name2 = workerList[2] -- use bracketed access to retrieve "Heather"
name2 = workerList.getAt(2) -- use getAt() to retrieve "Heather"

// JavaScript syntax
var workerList = list("Bruno", "Heather", "Carlos");
var name2 = workerList[2] // use bracketed access to retrieve "Heather"
var name2 = workerList.getAt(2) // use getAt() to retrieve "Heather"
```

So legen Sie einen Wert in einer Eigenschaftsliste fest

Führen Sie einen der folgenden Schritte aus:

- Verwenden Sie den Ist-gleich-Operator (=).
- (Nur Lingo) Verwenden Sie die `setaProp()`-Funktion.

- Verwenden Sie die Punktsyntax.

  Die folgende Lingo-Anweisung verwendet den Ist-gleich-Operator, damit `sushi` der neue Wert der Eigenschaft `Bruno` wird:

  ```
  -- Lingo syntax
  foodList = [:] -- define an empty property list
  foodList[#Bruno] = "sushi" -- associate sushi with Bruno
  ```

  Die folgende Lingo-Anweisung verwendet `setaprop()`, damit `sushi` der neue Wert der Eigenschaft `Bruno` wird:

  ```
  -- Lingo syntax
  foodList = [:] -- define an empty property list
  foodList.setaProp(#Bruno, "sushi") -- use setaProp()

  // JavaScript syntax
  foodList = propList() -- define an empty property list
  foodList.setaProp("Bruno", "sushi") -- use setaProp()
  ```

  Die folgenden Anweisungen verwenden die Punktsyntax, um den `Bruno` zugewiesenen Wert von `sushi` in `teriyaki` zu ändern:

  ```
  -- Lingo syntax
  foodList = [#Bruno:"sushi"] -- define a property list
  trace(foodList) -- displays [#Bruno: "sushi"]
  foodList.Bruno = "teriyaki" -- use dot syntax to set the value of Bruno
  trace(foodList) -- displays [#Bruno: "teriyaki"]

  // JavaScript syntax
  var foodList = propList("Bruno", "sushi"); // define a property list
  trace(foodList); // displays ["Bruno": "sushi"]
  foodList.Bruno = "teriyaki" // use dot syntax to set the value of Bruno
  trace(foodList) -- displays [#Bruno: "teriyaki"]
  ```

So rufen Sie einen Wert aus einer Eigenschaftsliste ab

Führen Sie einen der folgenden Schritte aus:

- Verwenden Sie die Listenvariable, gefolgt von dem Namen der Eigenschaft, die mit dem Wert verbunden ist. Setzen Sie die Eigenschaft in eckige Klammern.
- Verwenden Sie die `getaProp()`- oder `getPropAt()`-Methode.
- Verwenden Sie die Punktsyntax.

Die folgenden Anweisungen verwenden einen Zugriff mit eckigen Klammern, um die Werte der Eigenschaften `breakfast` und `lunch` abzurufen:

```
-- Lingo syntax
-- define a property list
foodList = [#breakfast:"Waffles", #lunch:"Tofu Burger"]
trace(foodList[#breakfast]) -- displays "Waffles"
trace(foodList[#lunch]) -- displays "Tofu Burger"

// JavaScript syntax
// define a property list
var foodList = propList("breakfast", "Waffles", "lunch", "Tofu Burger");
trace(foodList["breakfast"]); // displays Waffles
trace(foodList["lunch"]); // displays Tofu Burger
```

Die folgenden Anweisungen verwenden die `getaProp()`-Methode zum Abrufen des Wertes der Eigenschaft `breakfast` und die `getPropAt()`-Methode zum Abrufen der Eigenschaft an der zweiten Indexposition in der Liste:

```
-- Lingo syntax
-- define a property list
foodList = [#breakfast:"Waffles", #lunch:"Tofu Burger"]
trace(foodList.getaProp(#breakfast)) -- displays "Waffles"
trace(foodList.getPropAt(2)) -- displays #lunch

// JavaScript syntax
// define a property list
var foodList = propList("breakfast", "Waffles", "lunch", "Tofu Burger");
trace(foodList.getaProp("breakfast")) // displays Waffles
trace(foodList.getPropAt(2)) // displays lunch
```

Die folgenden Anweisungen verwenden die Punktsyntax, um auf die Werte der Eigenschaften in einer Eigenschaftsliste zuzugreifen:

```
-- Lingo syntax
-- define a property list
foodList = [#breakfast:"Waffles", #lunch:"Tofu Burger"]
trace(foodList.breakfast) -- displays "Waffles"

// JavaScript syntax
// define a property list
var foodList = propList("breakfast", "Waffles", "lunch", "Tofu Burger");
trace(foodList.lunch); // displays Tofu Burger
```

Überprüfen der Einträge in Listen

Auf folgenden Weisen können Sie die Merkmale einer Liste sowie die Anzahl der Einträge feststellen:

- Wenn Sie den Inhalt einer Liste anzeigen möchten, verwenden Sie die Funktion `put()` oder `trace()`, und übergeben Sie die Variable mit der Liste als Parameter.
- Wenn Sie die Anzahl der Einträge in einer Liste feststellen möchten, verwenden Sie die Methode `count()` (nur Lingo) oder die Eigenschaft `count`.
- Wenn Sie den Typ einer Liste feststellen möchten, verwenden Sie die Methode `ilk()`.
- Wenn Sie den höchsten Wert in einer Liste feststellen möchten, verwenden Sie die Methode `max()`.
- Wenn Sie den niedrigsten Wert in einer Liste feststellen möchten, verwenden Sie die Funktion `min()`.
- Wenn Sie die Position einer bestimmten Eigenschaft feststellen möchten, verwenden Sie den Befehl `findPos`, `findPosNear` oder `getOne`.

Die folgenden Anweisungen verwenden `count()` und `count` zum Anzeigen der Anzahl der Listeneinträge:

```
-- Lingo syntax
workerList = ["Bruno", "Heather", "Carlos"] -- define a linear list
trace(workerList.count()) -- displays 3
trace(workerList.count) -- displays 3

// JavaScript syntax
var workerList = list("Bruno", "Heather", "Carlos"); // define a linear list
trace(workerList.count); // displays 3
```

Die folgenden Anweisungen verwenden `ilk()` zum Bestimmen des Listentyps:

```
-- Lingo syntax
x = ["1", "2", "3"]
trace(x.ilk()) // returns #list

// JavaScript syntax
var x = list("1", "2", "3");
trace(x.ilk()) // returns #list
```

Die folgenden Anweisungen verwenden `max()` und `min()` zum Bestimmen des größten und kleinsten Wertes in einer Liste:

```
-- Lingo syntax
workerList = ["Bruno", "Heather", "Carlos"] -- define a linear list
trace(workerList.max()) -- displays "Heather"
trace(workerList.min()) -- displays "Bruno"

// JavaScript syntax
var workerList = list("Bruno", "Heather", "Carlos"); // define a linear list
trace(workerList.max()); // displays Heather
trace(workerList.min()); // displays Bruno
```

Die folgenden Anweisungen verwenden `findPos` zum Abrufen der Indexposition einer angegebenen Eigenschaft in einer Eigenschaftsliste:

```
-- Lingo syntax
-- define a property list
foodList = [#breakfast:"Waffles", #lunch:"Tofu Burger"]
trace(foodList.findPos(#lunch)) -- displays 2

// JavaScript syntax
// define a property list
var foodList = propList("breakfast", "Waffles", "lunch", "Tofu Burger");
trace(foodList.findPos("breakfast")); // displays 1
```

Hinzufügen und Löschen von Listeneinträgen

Auf folgenden Weisen können Sie Einträge einer Liste hinzufügen oder aus einer Liste löschen:

- Wenn Sie einen Eintrag am Ende der Liste hinzufügen möchten, verwenden Sie die Methode `append()`.
- Wenn Sie einen Eintrag an der ordnungsgemäßen Stelle in eine sortierte Liste einordnen möchten, verwenden Sie die Methode `add()` oder `addProp()`.
- Wenn Sie einen Eintrag an einer bestimmten Stelle in eine lineare Liste einfügen möchten, verwenden Sie die Methode `addAt()`.
- Wenn Sie einen Eintrag an einer bestimmten Stelle in eine Eigenschaftsliste einfügen möchten, verwenden Sie die Methode `addProp()`.
- Wenn Sie einen Eintrag aus einer Liste löschen möchten, verwenden Sie die Methode `deleteAt()`, `deleteOne()` oder `deleteProp()`.
- Wenn Sie einen Eintrag in einer Liste ersetzen möchten, verwenden Sie die Methode `setAt()` oder `setaProp()`..

Die folgenden Anweisungen verwenden `append()`, um einen Eintrag am Listenende hinzuzufügen:

```
-- Lingo syntax
workerList = ["Bruno", "Heather", "Carlos"] -- define a linear list
workerList.append("David")
trace(workerList) -- displays ["Bruno", "Heather", "Carlos", "David"]

// JavaScript syntax
var workerList = list("Bruno", "Heather", "Carlos"); // define a linear list
workerList.append("David");
trace(workerList); // displays ["Bruno", "Heather", "Carlos", "David"]
```

Die folgenden Anweisungen verwenden `addProp()`, um eine Eigenschaft und den dazugehörigen Wert einer Eigenschaftsliste hinzuzufügen:

```
-- Lingo syntax
-- define a property list
foodList = [#breakfast:"Waffles", #lunch:"Tofu Burger"]
foodList.addProp(#dinner, "Spaghetti") -- adds [#dinner: "Spaghetti"]

// JavaScript syntax
// define a property list
var foodList = propList("breakfast", "Waffles", "lunch", "Tofu Burger");
foodList.addProp("dinner", "Spaghetti"); // adds ["dinner": "Spaghetti"]
```

Sie brauchen eine Liste, die Sie nicht mehr benötigen, nicht ausdrücklich zu entfernen. Listen werden automatisch entfernt, wenn in keiner Variablen mehr auf sie verwiesen wird. Objekte anderer Typen müssen explizit entfernt werden, indem Sie Variablen, die auf sie verweisen, auf `VOID` (Lingo) oder `null` (JavaScript) festlegen.

Kopieren von Listen

Wenn Sie eine Liste, die Sie einer Variablen zugeordnet haben, einer zweiten Variablen zuordnen, wird keine separate Kopie der Liste erstellt. Die erste der folgenden Anweisungen erstellt beispielsweise eine Liste mit den Namen zweier Kontinente und weist die Liste der Variablen `landList` zu. Die zweite Anweisung weist dieselbe Liste der neuen Variablen `continentList` zu. In der dritten Anweisung wird bei Hinzufügen von `Australia` zu `landList` der Eintrag `Australia` automatisch auch der Liste `continentList` zugewiesen, da beide Variablen auf dasselbe List-Objekt im Speicher verweisen. Dasselbe Verhalten tritt bei Verwenden eines Arrays in JavaScript-Syntax ein.

```
-- Lingo syntax
landList = ["Asia", "Africa"]
continentList = landList
landList.add("Australia") -- this also adds "Australia" to continentList
```

So erstellen Sie eine Kopie einer Liste, die von einer anderen Liste unabhängig ist

❖ Verwenden Sie die Methode `duplicate()`.

Die folgenden Anweisungen erstellen beispielsweise eine Liste und erstellen anschließend eine unabhängige Kopie der Liste:

```
-- Lingo syntax
oldList = ["a", "b", "c"]
newList = oldList.duplicate() -- makes an independent copy of oldList

// JavaScript syntax
var oldList = list("a", "b", "c");
var newList = oldList.duplicate(); // makes an independent copy of oldList
```

Nachdem Sie die Liste `newList` angelegt haben, können Sie die Listen `oldList` und `newList` unabhängig voneinander bearbeiten.

Sortieren von Listen

Listen werden in alphanumerischer Reihenfolge (Zahlen vor Zeichenfolgen) sortiert. Zeichenfolgen werden hierbei unabhängig von der Anzahl ihrer Zeichen nach Anfangsbuchstabe sortiert. Sortierte Listen lassen sich etwas rascher verarbeiten als nicht sortierte Listen.

Eine lineare Liste wird nach den Werten in der Liste sortiert. Eine Eigenschaftsliste wird nach den Eigenschaftsnamen in der Liste oder im Array sortiert.

Nachdem die Werte in einer linearen oder Eigenschaftsliste sortiert wurden, bleiben sie sortiert, auch wenn Listenwerte hinzugefügt oder entfernt werden.

So sortieren Sie eine Liste

❖ Verwenden Sie die `sort()`-Methode.

Die folgenden Anweisungen sortieren beispielsweise eine nicht sortierte alphabetische Liste:

```
-- Lingo syntax
oldList = ["d", "a", "c", "b"]
oldList.sort() -- results in ["a", "b", "c", "d"]

// JavaScript syntax
var oldList = list("d", "a", "c", "b");
oldList.sort(); // results in ["a", "b", "c", "d"]
```

Erstellen mehrdimensionaler Listen

Sie können auch mehrdimensionale Listen erstellen, mit deren Hilfe Sie gleichzeitig mit Werten in mehr als einer Liste arbeiten können.

Im folgenden Beispiel erstellen die ersten beiden Anweisungen die getrennten linearen Listen `list1` und `list2`. Die dritte Anweisung erstellt eine mehrdimensionale Liste und weist diese `mdList` zu. Um auf die Werte in einer mehrdimensionalen Liste zuzugreifen, verwenden die vierte und fünfte Anweisung eckige Klammern für den Zugriff auf die Listenwerte. Die erste eckige Klammer bietet Zugriff auf eine angegebene Liste, die zweite auf den Wert an einer angegebenen Indexposition in der Liste.

```
-- Lingo syntax
list1 = list(5,10)
list2 = list(15,20)
mdList = list(list1, list2)
trace(mdList[1][2]) -- displays 10
trace(mdList[2][1]) -- displays 15

// JavaScript syntax
var list1 = list(5,10);
var list2 = list(15,20);
var mdList = list(list1, list2);
trace(mdList[1][2]); // displays 10
trace(mdList[2][1]); // displays 15
```

JavaScript-Arrays

JavaScript-Arrays ähneln linearen Listen in Lingo dahingehend, dass jedes Element in einem Array einen einzelnen Wert darstellt. Einer der Hauptunterschiede zwischen JavaScript-Arrays und linearen Lingo-Listen ist, dass der Index eines Arrays stets mit 0 beginnt.

Sie erstellen eine JavaScript-Array mithilfe des Array-Objekts. Zum Erstellen eines Arrays können Sie entweder eckige Klammern (`[ ]`) oder den Konstruktor `Array` verwenden. Die beiden folgenden Anweisungen erstellen ein Array mit zwei Werten:

```
// JavaScript syntax
var myArray = [10, 15]; // using square brackets
var myArray = new Array(10, 15); // using the Array constructor
```

Sie können auch leere Arrays erstellen (siehe die beiden folgenden Anweisungen):

```
// JavaScript syntax
var myArray = [];
var myArray = new Array();
```

Hinweis: *Die Director-Skriptreferenz bietet keine vollständige Übersicht über die „Array"-Objekte in JavaScript. Weitere Informationen zu „Array"-Objekten finden Sie in den vielen Ressourcen anderer Anbieter zu diesem Thema.*

Überprüfen der Elemente in Arrays

Auf folgenden Weisen können Sie die Merkmale eines Arrays sowie die Anzahl der Elemente feststellen:

- Wenn Sie den Inhalt eines Arrays anzeigen möchten, verwenden Sie die Funktion `put()` oder `trace()`, und übergeben Sie die Variable mit dem Array als Parameter.
- Wenn Sie die Anzahl der Elemente in einem Array feststellen möchten, verwenden Sie die `length`-Eigenschaft des Array-Objekts.
- Wenn Sie den Typ eines Arrays feststellen möchten, verwenden Sie die Eigenschaft `constructor`.

Das folgende Beispiel veranschaulicht das Festlegen der Anzahl der Elemente in einem Array mithilfe der `length`-Eigenschaft und anschließende Zurückgeben des Objekts mithilfe der `constructor`-Eigenschaft:

```
// JavaScript syntax
var x = ["1", "2", "3"];
trace(x.length) // displays 3
trace(x.constructor == Array) // displays true
```

Hinzufügen und Löschen von Array-Elementen

Auf folgende Weisen können Sie Elemente einem Array hinzufügen oder aus einem Array löschen:

- Wenn Sie ein Element am Ende der Liste hinzufügen möchten, verwenden Sie die Methode `push()` des Array-Objekts.
- Wenn Sie ein Element an der ordnungsgemäßen Stelle in ein sortiertes Array einordnen möchten, verwenden Sie die Methode `splice()` des Array-Objekts.
- Wenn Sie ein Element an einer bestimmten Stelle in ein Array einordnen möchten, verwenden Sie die Methode `splice()` des Array-Objekts.
- Wenn Sie ein Element aus einem Array löschen möchten, verwenden Sie die Methode `splice()` des Array-Objekts.

- Wenn Sie ein Element in einem Array ersetzen möchten, verwenden Sie die Methode `splice()` des Array-Objekts.

Das folgende Beispiel veranschaulicht das Verwenden der Methode `splice()` des Array-Objekts zum Hinzufügen von Elementen zu, Löschen von Elementen aus und Ersetzen von Elementen in einem Array:

```
// JavaScript syntax
var myArray = new Array("1", "2");
trace(myArray); displays 1,2

myArray.push("5"); // adds the value "5" to the end of myArray
trace(myArray); // displays 1,2,5

myArray.splice(3, 0, "4"); // adds the value "4" after the value "5"
trace(myArray); // displays 1,2,5,4

myArray.sort(); // sort myArray
trace(myArray); // displays 1,2,4,5

myArray.splice(2, 0, "3");
trace(myArray); // displays 1,2,3,4,5

myArray.splice(3, 2); // delete two values at index positions 3 and 4
trace(myArray); // displays 1,2,3

myArray.splice(2, 1, "7"); // replaces one value at index position 2 with "7"
trace(myArray); displays 1,2,7
```

Kopieren von Arrays

Wenn Sie ein Array, das Sie einer Variablen zugeordnet haben, einer zweiten Variablen zuordnen, wird keine separate Kopie des Arrays erstellt.

Die erste der folgenden Anweisungen erstellt beispielsweise ein Array mit den Namen zweier Kontinente und weist das Array der Variablen `landList` zu. Die zweite Anweisung weist dieselbe Liste der neuen Variablen `continentList` zu. In der dritten Anweisung wird bei Hinzufügen von `Australia` zu `landList` der Eintrag `Australia` automatisch auch dem Array `continentList` zugewiesen. Dies geschieht, weil beide Variablen auf dasselbe Array-Objekt im Speicher verweisen.

```
// JavaScript syntax
var landArray = new Array("Asia", "Africa");
var continentArray = landArray;
landArray.push("Australia"); // this also adds "Australia" to continentList
```

So erstellen Sie eine Kopie eines Arrays, das von einem anderen Array unabhängig ist

❖ Verwenden Sie die `slice()`-Methode des Array-Objekts.

Die folgenden Anweisungen erstellen beispielsweise ein Array und erstellen anschließend mithilfe von `slice()` eine unabhängige Kopie des Arrays:

```
// JavaScript syntax
var oldArray = ["a", "b", "c"];
var newArray = oldArray.slice(); // makes an independent copy of oldArray
```

Nachdem Sie das Array `newArray` angelegt haben, können Sie die Arrays `oldArray` und `newArray` unabhängig voneinander bearbeiten.

Sortieren von Arrays

Arrays werden in alphanumerischer Reihenfolge (Zahlen vor Zeichenfolgen) sortiert. Zeichenfolgen werden hierbei unabhängig von der Anzahl ihrer Zeichen nach Anfangsbuchstabe sortiert.

So sortieren Sie ein Array

❖ Verwenden Sie die `sort()`-Methode des Array-Objekts.

Die folgenden Anweisungen sortieren ein nicht sortiertes alphabetisches Array:

```
// JavaScript syntax
var oldArray = ["d", "a", "c", "b"];
oldArray.sort(); // results in a, b, c, d
```

Die folgenden Anweisungen sortieren ein nicht sortiertes alphanumerisches Array:

```
// JavaScript syntax
var oldArray = [6, "f", 3, "b"];
oldArray.sort(); // results in 3, 6, b, f
```

Das Sortieren eines Arrays führt zu einem neuen sortierten Array.

Erstellen mehrdimensionaler Arrays

Sie können auch mehrdimensionale Arrays erstellen, mit deren Hilfe Sie gleichzeitig mit Werten in mehr als einem Array arbeiten können.

Im folgenden Beispiel erstellen die ersten beiden Anweisungen die getrennten Arrays `array1` und `array2`. Die dritte Anweisung erstellt ein mehrdimensionales Array und weist dieses `mdArray` zu. Um auf die Werte in einem mehrdimensionalen Array zuzugreifen, verwenden die vierte und fünfte Anweisung eckige Klammern für den Zugriff auf die Array-Werte. Die erste eckige Klammer bietet Zugriff auf ein angegebenes Array, die zweite auf den Wert an einer angegebenen Indexposition im Array.

```
// JavaScript syntax
var array1 = new Array(5,10);
var array2 = [15,20];
var mdArray = new Array(array1, array2);
trace(mdArray[0][1]); // displays 10
trace(mdArray[1][0]); // displays 15
```

Kapitel 3: Schreiben von Skripts in Director

Scripts in Director® 11 bieten in Filmen Funktionalität, die auf andere Weise nicht möglich wäre. Beim Schreiben von Skripts werden Sie zur Unterstützung der gewünschten komplexeren Interaktivität in Ihren Director-Filmen zunehmend anspruchsvollere Skripts benötigen. Nachfolgend erhalten Sie Informationen zu Konzepten und Methoden zum Erstellen dieser Skripts und zur objektorientierten Skripterstellung in Director.

Wenn Sie noch keine Skripte in Director erstellt haben, sollten Sie als Vorbereitung auf die folgenden Themen die Einführung „Grundlagen der Skripterstellung in Director“ auf Seite 4 lesen.

Auswählen zwischen Lingo- und JavaScript-Syntax

Lingo und JavaScript bieten Zugriff auf dieselben Objekte, Ereignisse und Skripterstellungs-APIs. Deshalb spielt es keine Rolle, welche Sprache Sie zum Schreiben von Skripts wählen. Sie können sich also für die Sprache entscheiden, mit der Sie am meisten vertraut sind.

Um zu verstehen, wie Skripterstellungssprachen in der Regel bei einem gegebenen Objekt- und Ereignismodell in Director funktionieren, berücksichtigen Sie Folgendes:

- Im Allgemeinen „umhüllt“ eine Skripterstellungssprache wie Lingo oder JavaScript ein bestimmtes Objekt- und Ereignismodell, um Zugriff auf die dazugehörigen Objekte und Ereignisse bereitzustellen.
- JavaScript ist eine Implementierung des ECMAScript-Standards, die das Objekt- und Ereignismodell eines Webbrowsers umhüllt, um Zugriff auf dessen Objekte und Ereignisse zu bieten.
- ActionScript ist eine Implementierung des ECMAScript-Standards, die das Objekt- und Ereignismodell von Adobe® Flash® umhüllt, um Zugriff auf Flash-Objekte und -Ereignisse zu bieten.
- Die Director-Implementierung von JavaScript ist eine Implementierung des ECMAScript-Standards, die das Objekt- und Ereignismodell von Director umhüllt, um Zugriff auf Director-Objekte und -Ereignisse zu bieten.
- Lingo ist eine benutzerdefinierte Syntax, die das Objekt- und Ereignismodell von Director umhüllt, um Zugriff auf Director-Objekte und -Ereignisse zu bieten.

Lingo und JavaScript sind die beiden Sprachen, über die Sie auf das Objekt- und Ereignismodell von Director zugreifen können. In einer Sprache geschriebene Skripts weisen dieselben Merkmale wie in der anderen Sprache geschriebene Skripts auf.

Wenn Sie also wissen, wie Sie in der einen Sprache auf die Skripterstellungs-APIs zugreifen, wissen Sie im Wesentlichen auch, wie der Zugriff in der anderen Sprache erfolgt. Über JavaScript-Code können Sie beispielsweise auf Lingo-Datentypen wie Symbole, lineare Listen, Eigenschaftslisten usw. zugreifen, Lingo-Parent-Skripts und -Verhalten erstellen und aufrufen, Xtra-Erweiterungen erstellen und aufrufen und Lingo-Chunk-Ausdrücke verwenden. Sie können auch JavaScript- und Lingo-Skripts gemeinsam in einem Film verwenden, wobei jedoch ein einzelner Skriptdarsteller nur jeweils eine Syntax enthalten darf.

Es gibt zwei Hauptunterschiede zwischen Lingo und JavaScript:

- Beide Sprachen weisen besondere terminologische und syntaktische Konventionen auf. Beispielsweise unterscheidet sich die Syntax für Ereignisprozeduren:

```
-- Lingo syntax
on mouseDown
    ...
end

// JavaScript syntax
function mouseDown() {
    ...
}
```

Weitere Informationen zu den terminologischen und syntaktischen Konventionen der beiden Sprachen finden Sie unter „Skriptterminologie“ auf Seite 5 und„Skriptsyntax“ auf Seite 7.

- Der Zugriff auf einige der Skripterstellungs-APIs ist in beiden Sprachen geringfügig anders. Sie müssen z. B. unterschiedliche Konstrukte verwenden, um auf das zweite Wort im ersten Absatz eines Textdarstellers zuzugreifen:

```
-- Lingo syntax
member("News Items").paragraph[1].word[2]

// JavaScript syntax
member("News Items").getPropRef("paragraph", 1).getProp("word", 2);
```

Skripterstellung im Punktsyntaxformat

Beim Schreiben von Skripts mit Lingo- oder JavaScript-Syntax wählen Sie das Punktsyntaxformat, über das Sie auf die zu einem Objekt gehörenden Eigenschaften und Methoden zugreifen. Ein Objekt, gefolgt von einem Punkt (.) und dem Namen der Eigenschaft, Methode oder des Textblocks, die/den Sie angeben möchten. Jeder Punkt in einer Anweisung stellt im Wesentlichen einen Wechsel von einer höheren, allgemeineren Ebene zu einer niedrigeren, spezifischeren Ebene in der Objekthierarchie dar.

Das folgende Beispiel erstellt z. B. einen Verweis auf die Besetzungsbibliothek mit dem Namen „News Stories“ und verwendet anschließend Punktsyntax, um auf die Anzahl der Darsteller in dieser Besetzungsbibliothek zuzugreifen.

Geben Sie zur Bezeichnung von Textblöcken (Chunks) nach dem Punkt Begriffe ein, die sich auf spezifischere Elemente im Text beziehen. Die nachfolgende erste Anweisung verweist beispielsweise auf den ersten Absatz des Textdarstellers mit dem Namen „News Items“. Die zweite Anweisung verweist auf das zweite Wort im ersten Absatz.

```
-- Lingo syntax
member("News Items").paragraph[1]
member("News Items").paragraph[1].word[2]

// JavaScript syntax
member("News Items").getPropRef("paragraph", 1);
member("News Items").getPropRef("paragraph", 1).getProp("word", 2);
```

Bei einigen Objekten, die einen abgestuften Eigenschaftenzugriff auf entweder Daten oder einen bestimmten Darstellertyp verarbeiten (siehe die beiden vorherigen Anweisungen), wird der Zugriff auf die Eigenschaften nicht über normale JavaScript-Syntax unterstützt. Um mit JavaScript-Syntax auf abgestufte Eigenschaften zugreifen zu können, müssen Sie die Methoden `getPropRef()` und `getProp()` verwenden.

Bei dieser JavaScript-Ausnahme müssen verschiedene Dinge beachtet werden:

- Diese Vorgehensweise gilt für 3D-Objekte, Textdarsteller, Felddarsteller und XML Parser-Xtra-Erweiterungen, auf die über JavaScript-Syntax zugegriffen wird.

- Sie müssen die `getPropRef()`-Methode zum Speichern eines Verweises auf eines der zuvor erwähnten Objekte bzw. dessen Eigenschaften mithilfe von JavaScript-Syntax verwenden.
- Sie müssen die `getProp()`-Methode zum Abrufen eines Eigenschaftenwertes eines der zuvor erwähnten Objekte bzw. dessen Eigenschaften mithilfe von JavaScript-Syntax verwenden.
- Der Zugriff auf 3D-Objekte und -Eigenschaften muss über deren vollständig qualifizierte Namen in JavaScript-Syntax erfolgen. In Lingo kann beispielsweise die `shader`-Eigenschaft als Kurzform der `shaderList[1]`-Eigenschaft verwendet werden. In JavaScript muss dagegen die `shaderList[1]`-Eigenschaft stets verwendet werden.

Einführung in die Director-Objekte

Objekte sind grundsätzlich logische Gruppierungen benannter Daten, die auch Methoden enthalten können, die auf diese Daten angewendet werden. In dieser Version von Director wurden die Skripterstellungs-APIs in Objekten gruppiert, wobei der Zugriff über diese Objekte erfolgt. Jedes Objekt bietet Zugriff auf einen bestimmten Satz benannter Daten und Funktionstypen. Das „Sprite“-Objekt bietet beispielsweise Zugriff auf die Daten und Funktionen eines Sprites, das „Movie“-Objekt auf die Daten und Funktionen eines Films usw.

Die in Director verwendeten Objekte gliedern sich in vier Kategorien – Kernobjekte, Medientypen, Skriptobjekte und 3D-Objekte. Je nach Art der Funktionalität, die Sie hinzufügen möchten, und des Teils eines Films, dem Sie Funktionalität hinzufügen, verwenden Sie Objekte aus einer oder mehreren dieser Kategorien.

Core-Objekte

Diese Objektkategorie bietet Zugriff auf die Hauptfunktionen in Director, z. B. die Director-Wiedergabe-Engine, Filmfenster, Sprites, Sounds usw. Diese Objekte bilden die Basisebene, über die auf alle APIs und anderen Objektkategorien zugegriffen wird.

Es gibt auch eine Gruppe mit Methoden und Eigenschaften oberster Ebene, über die Sie direkt auf alle Core-Objekte zugreifen können, ohne für den Zugriff auf ein bestimmtes Core-Objekt die Objekthierarchie durchlaufen zu müssen.

Eine Übersicht über die verfügbaren Core-Objekte und deren APIs finden Sie unter „Core-Objekte in Director“ auf Seite 100.

Medientypen

Diese Objektkategorie ermöglicht den Zugriff auf die Funktionalität der verschiedenen Medientypen wie RealMedia, DVD, animiertes GIF usw., die als Darsteller zu Filmen hinzugefügt werden.

Genau genommen handelt es sich bei Medientypen nicht um Objekte, sondern um Darsteller mit einem bestimmten Medientyp. Wenn Sie einen Medientyp als Darsteller zu einem Film hinzufügen, erbt er nicht nur die Funktionalität des Core-Objekts „Member“, sondern erweitert das Member-Objekt um zusätzliche Funktionen, die nur für bestimmte Medientypen zur Verfügung stehen. Ein RealMedia-Darsteller hat beispielsweise Zugriff auf die Methoden und Eigenschaften des Member-Objekts, verfügt jedoch auch über zusätzliche, für RealMedia spezifische Methoden und Eigenschaften. Bei allen anderen Medientypen verhält sich dies genauso.

Eine Übersicht über die verfügbaren Medientypen und deren APIs finden Sie unter „Medientypen“ auf Seite 120.

Skriptobjekte

Diese Objektkategorie, auch *Xtra-Erweiterungen* genannt, bietet Zugriff auf die Funktionalität der mit Director® installierten Softwarekomponenten (z. B. XML Parser, Fileio, SpeechXtra usw.) und erweitert die Director-Kernfunktionalität. Die vordefinierten Xtra-Erweiterungen stellen Funktionen bereit, mit denen Sie zum Beispiel Filter importieren oder eine Internetverbindung herstellen können. Wenn Sie mit der Programmiersprache C vertraut sind, können Sie eigene Xtra-Erweiterungen entwickeln.

Eine Übersicht über die verfügbaren Skriptobjekte und deren APIs finden Sie unter „Skriptobjekte“ auf Seite 140.

3D-Objekte

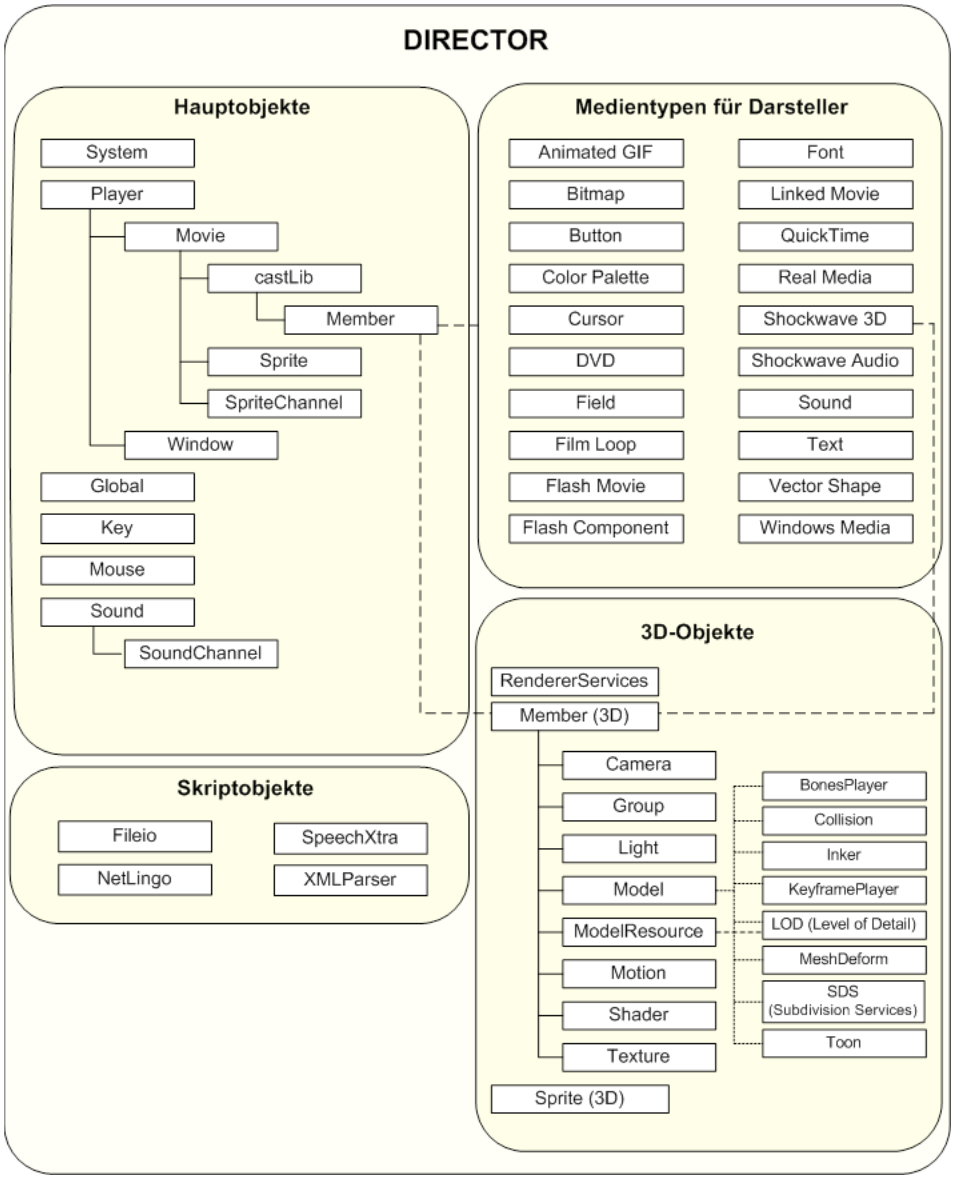
Diese Objektkategorie bietet Zugriff auf die Funktionalität der Darsteller und von Text zum Erstellen von 3D-Filmen.

Weitere Informationen zu 3D-Filmen finden Sie in den Themen des Director-Benutzerhandbuchs im Director-Hilfefenster.

Eine Übersicht über die verfügbaren 3D-Objekte und deren APIs finden Sie unter„3D-Objekte“ auf Seite 166.

Objektmodelldiagramme

Die folgenden Diagramme veranschaulichen die grundlegenden Beziehungen zwischen den Objektgruppen und ihrer Hierarchien in Director. Informationen zu Objekterstellung, Eigenschaften und Methoden sowie anderen APIs finden Sie in den entsprechenden API-Referenzthemen.



Objektgruppen und ihre Hierarchie

Funktionen und Eigenschaften oberster Ebene

Es gibt verschiedene Funktionen und Eigenschaften oberster Ebene, die einen direkten Zugriff auf Core-Objekte und -Funktionen in Director bieten. Wenn Sie Verweise auf Core-Objekte, neue Bilder, Listen usw. erstellen, werden Sie viele dieser Funktionen und Eigenschaften umfassend nutzen. Die Eigenschaft oberster Ebene `_movie` verweist beispielsweise direkt auf das Core-Objekt Movie, und die Funktion oberster Ebene `list()` erstellt eine lineare Liste.

In den folgenden Tabellen sind die Funktionen und Eigenschaften oberster Ebene aufgelistet.

Funktionen oberster Ebene	
castLib()	randomVector()
channel() (Top-Level)	rect()
color()	script()
date() (Formate), date() (System)	showLocals()
image()	sound()
isBusy()	sprite()
list()	symbol()
member()	timeout()
point()	trace()
propList()	vector()
put()	window()
random()	xtra()

Eigenschaften oberster Ebene	
_global	_player
_key	_sound
_mouse	_system
_movie	

Einführung in die objektorientierte Programmierung in Director

Mithilfe von Lingo- oder JavaScript-Syntax können Sie objektorientierte Programmierungsgrundsätze bei Ihren Skripts umsetzen. Das Umsetzen objektorientierter Grundsätze erleichtert die Programmierung im Allgemeinen, da Sie weniger Code schreiben müssen, zum Erledigen von Aufgaben eine einfachere Logik verwenden können und die Wiederverwendbarkeit und Modularität Ihres Codes verbessern können.

Je nach verwendeter Skriptsprache setzen Sie diese Grundsätze mittels zweier verschiedener Modelle um:

- In Lingo verwenden Sie Parent-Skripts, Ancestor-Skripts und Child-Objekte zur Simulierung der objektorientierten Programmierung.
- In JavaScript verwenden Sie standardmäßige JavaScript-konforme objektorientierte Programmierungstechniken zum Erstellen von Klassen und Unterklassen.

Die beiden Modelle ermöglichen das Übernehmen der Vorteile der objektorientierten Programmierung in Ihre Skripts, sodass es keine Rolle spielt, welche Skriptsprache Sie verwenden. Sie setzen lediglich die Grundsätze auf verschiedene Weisen um.

Da die Skriptsprachen unterschiedliche Modelle zum Umsetzen objektorientierter Grundsätze verwenden, funktionieren die für eine Sprache beschriebenen Techniken nicht in der anderen Sprache. Deshalb müssen Sie sich nur mit den Inhalten beschäftigen, die für die von Ihnen genutzte Skriptsprache gelten:

- Weitere Informationen über das Simulieren objektorientierter Programmierung in Lingo finden Sie unter „Objektorientierte Programmierung mithilfe von Lingo-Syntax“ auf Seite 50.
- Weitere Informationen über das Simulieren objektorientierter Programmierung in JavaScript-Syntax finden Sie unter „Objektorientierte Programmierung mithilfe von JavaScript“ auf Seite 60.

Objektorientierte Programmierung mithilfe von Lingo-Syntax

In Lingo bieten Parent-Skripts die Vorteile der objektorientierten Programmierung. Sie können Parent-Skripts verwenden, um Skriptobjekte zu erzeugen, die sich ähnlich verhalten und sich dennoch unabhängig voneinander einsetzen lassen.

In Lingo können mehrere Kopien (oder Instanzen) eines Parent-Skripts erstellt werden. Die einzelnen Instanzen eines Parent-Skripts werden als Child-Objekte bezeichnet. Child-Objekte lassen sich während der Filmwiedergabe je nach Bedarf erstellen. Hierbei ist die Anzahl der Child-Objekte, die auf der Grundlage eines Parent-Skripts erstellt werden können, nur durch die auf Ihrem Computer verfügbare Menge an Arbeitsspeicher begrenzt.

Director kann mehrere Child-Objekte eines Parent-Skripts ebenso erstellen wie mehrere Instanzen eines Verhaltens für unterschiedliche Sprites. Hierbei fungiert das Parent-Skript praktisch als Vorlage, gemäß der die einzelnen Child-Objekte angelegt werden.

Die Erläuterung von Parent-Skripts und Child-Objekte in Lingo dient der Beschreibung der Grundlagen beim Schreiben von Parent-Skripts und Erstellen und Verwenden von Child-Objekten sowie der Bereitstellung von Skriptbeispielen. Die grundlegenden Konzepte der objektorientierten Programmierung werden an dieser Stelle nicht behandelt. Um Parent-Skripts und Child-Objekte erfolgreich nutzen zu können, sollten Sie jedoch mit den Grundsätzen der objektorientierten Programmierung vertraut sein. Eine Einführung in die Grundlagen der objektorientierten Programmierung finden Sie in zahlreichen Ressourcen zu diesem Thema.

Ähnlichkeiten mit anderen objektorientierten Programmiersprachen

Wenn Sie mit objektorientierten Programmiersprachen wie Java oder C++ vertraut sind, dürften Ihnen die Konzepte, die der Erstellung von Parent-Skripts zugrunde liegen, bereits bekannt sein, wenn auch unter anderem Namen.

Die Begriffe der Director-Terminologie, mit denen Parent-Skripts und Child-Objekte beschrieben werden, entsprechen den folgenden allgemein bekannten Begriffen aus der objektorientierten Programmierung:

Parent-Skripts in Director entsprechen Klassen in der objektorientierten Programmierung.

Child-Objekte in Director entsprechen Instanzen in der objektorientierten Programmierung.

Eigenschaftsvariablen entsprechen den Instanz- oder Member-Variablen in der objektorientierten Programmierung.

Prozeduren in Director entsprechen Methoden in der objektorientierten Programmierung.

Ancestor-Skripts in Director entsprechen Super- oder Basisklassen in der objektorientierten Programmierung.

Grundlegendes zu Parent-Skripts und Child-Objekten

Ein *Parent-Skript* in Lingo besteht aus einer Reihe von Prozeduren und Eigenschaften, die ein Child-Objekt definieren, ohne selbst ein Child-Objekt zu sein. Ein *Child-Objekt* ist eine in sich abgeschlossene, unabhängige Instanz eines Parent-Skripts. Da die verschiedenen Child-Objekte eines Parent-Skripts dieselben Prozeduren und Eigenschaften aufweisen, reagieren Child-Objekte in einer Gruppe auf Ereignisse und Nachrichten häufig auf ganz ähnliche Weise.

In der Regel werden Parent-Skripts verwendet, um Child-Objekte zu erstellen, die die Verwaltung der Filmlogik vereinfachen. Diese Child-Objekte sind insbesondere dann von Nutzen, wenn bei der Wiedergabe eines Films dieselbe Logik mehrmals mit verschiedenen Parametern ausgeführt werden muss. Sie können ein Child-Objekt auch in die `scriptInstanceList` eines Sprites oder die `actorList` des Filmobjekts aufnehmen und zur Steuerung von Animationen verwenden.

Da alle Child-Objekte eines Parent-Skripts identische Prozeduren enthalten, zeigen sie bei denselben Ereignissen oft ähnliche Reaktionen. Jedes Child-Objekt weist jedoch seine eigenen Werte für die im Parent-Skript definierten Eigenschaften auf, sodass es sich auch vollkommen anders verhalten kann als seine Geschwisterobjekte, obwohl es sich jeweils um Instanzen desselben Parent-Skripts handelt.

Sie können zum Beispiel ein Parent-Skript erstellen, das bearbeitbare Textfelder als Child-Objekte definiert, von denen jedes seine eigenen Eigenschaftseinstellungen sowie Text- und Farbeigenschaften aufweist. Durch Ändern der Eigenschaftswerte bestimmter Child-Objekte können Sie die entsprechenden Eigenschaften während der Filmwiedergabe ändern, ohne dass sich dies auf die übrigen Child-Objekte auswirkt, die auf demselben Parent-Skript basieren.

In ähnlicher Weise lassen sich Eigenschaften eines Child-Objekts unabhängig von der jeweils entsprechenden Einstellung verwandter Child-Objekte auf `TRUE` oder `FALSE` einstellen.

Ein Parent-Skript verweist auf den Namen eines Skriptdarstellers, der die Eigenschaftsvariablen und Prozeduren enthält. Ein anhand eines Parent-Skripts erstelltes Child-Objekt ist im Wesentlichen eine neue Instanz des Skriptdarstellers.

Unterschiede zwischen Child-Objekten und Verhalten

Child-Objekte und Verhalten ähneln sich, da beide über mehrere Instanzen verfügen können, weisen jedoch auch einige wesentliche Unterschiede auf. Der größte Unterschied zwischen Child-Objekten und Verhalten besteht darin, dass Verhalten sich auf bestimmte Positionen im Drehbuch beziehen, da sie Sprites zugeordnet sind. Verhaltensobjekte werden von den im Drehbuch gespeicherten Initialisierern automatisch erstellt, wenn der Abspielkopf auf seinem Weg von Bild zu Bild auf Sprites trifft, denen Verhalten zugeordnet sind. Im Gegensatz dazu müssen die Child-Objekte eines Parent-Skripts in einer Prozedur ausdrücklich erstellt werden.

Verhalten und Child-Objekte unterscheiden sich auch in der Art und Weise, wie sie Sprites zugeordnet werden. Director weist ein Verhalten automatisch dem Sprite zu, dem es zugeordnet ist. Ein Child-Objekt hingegen muss explizit mit einem Sprite verbunden werden. Child-Objekte erfordern keine Sprite-Referenzen und existieren ausschließlich im Arbeitsspeicher.

Grundlagen von Ancestor-Skripts

Parent-Skripts können *Ancestor-Skripts* deklarieren. Dies sind zusätzliche Skripts, deren Prozeduren und Eigenschaften von einem Child-Objekt abgerufen und verwendet werden können.

Mit Ancestor-Skripts können Sie Prozeduren und Eigenschaften erstellen, die sich anschließend in mehrere Parent-Skripts einbinden und wiederverwenden lassen.

Ein Parent-Skript erklärt ein anderes Parent-Skript zu seinem Ancestor, indem es das Skript seiner `ancestor`-Eigenschaft zuordnet. Die folgende Anweisung beispielsweise definiert das Skript „What_Everyone_Does“ als Ancestor des Parent-Skripts, in dem die Anweisung enthalten ist:

```
-- Lingo syntax
ancestor = new(script "What_Everyone_Does")
```

Wenn in einem Child-Objekt keine Prozeduren und Eigenschaften definiert sind, sucht Director die Prozedur oder Eigenschaft in den Ancestor-Skripts des Child-Objekts. Hierbei wird zunächst das Parent-Skript durchsucht. Ist beim Aufruf einer Prozedur oder beim Testen einer Eigenschaft im Parent-Skript keine entsprechende Definition zu finden, durchsucht Director das Ancestor-Skript und übernimmt, sofern vorhanden, dessen Definition.

Jedes Child-Objekt kann jeweils immer nur ein Ancestor-Skript aufweisen. Da dieses Ancestor-Skript jedoch seinerseits über ein Ancestor-Skript und dieses wiederum über ein eigenes Ancestor-Skript verfügen kann, lassen sich im Prinzip beliebig viele Parent-Skripts erstellen, deren Prozeduren von einem Child-Objekt genutzt werden können.

Schreiben von Parent-Skripts

Ein Parent-Skript enthält den Code, der für das Erstellen von Child-Objekten und das Definieren ihrer Verhalten und Eigenschaften erforderlich ist. Zunächst müssen Sie festlegen, wie sich das Child-Objekt verhalten soll. Anschließend können Sie ein Parent-Skript schreiben, das folgende Aufgaben ausführt:

- Es deklariert optional die betreffenden Eigenschaftsvariablen. Diese Variablen stehen für Eigenschaften, deren Wert sich von Child-Objekt zu Child-Objekt unterscheiden kann.
- Es legt die Anfangswerte für die Eigenschaften und Variablen der Child-Objekte in der Prozedur `on new` fest.
- Es enthält zusätzliche Prozeduren, die die Aktionen der Child-Objekte steuern.

Deklarieren von Eigenschaftsvariablen

Alle Child-Objekte eines bestimmten Parent-Skripts weisen anfangs dieselben Eigenschaftsvariablenwerte auf. Der Wert einer Eigenschaftsvariablen gilt jeweils nur für dasjenige Child-Objekt, dem sie zugeordnet ist. Eine Eigenschaftsvariable und ihr Wert bestehen nur so lange, wie das Child-Objekt existiert. Der Anfangswert für die Eigenschaftsvariable wird normalerweise in der Prozedur `on new` eingestellt. Geschieht dies nicht, erhält die Variable den Anfangswert `VOID`.

So deklarieren Sie eine Eigenschaftsvariable

❖ Verwenden Sie das Schlüsselwort `property` am Anfang des Parent-Skripts.

So werden Eigenschaftsvariablen außerhalb des Child-Objekts festgelegt und getestet

❖ Eigenschaftsvariablen werden genau wie alle anderen Eigenschaften in Skripts festgelegt und getestet. Verwenden Sie zu diesem Zweck die Syntax *objectRef.propertyName*.

Die folgende Anweisung stellt beispielsweise die Eigenschaft `speed` des Objekts `car1` ein:

```
car1.speed = 55
```

Erstellen der „new“-Prozedur

In der Regel enthält jedes Parent-Skript eine `on new`-Prozedur. Diese Prozedur erstellt das neue Child-Objekt, wenn ein anderes Skript einen Befehl `new(script` *Parent-Skriptname*`)` aufruft, der das betreffende Parent-Skript anweist, ein Child-Objekt von sich selbst zu erstellen. Bei Bedarf lässt sich mit der `on new`-Prozedur im Parent-Skript auch der anfängliche Eigenschaftswert des Child-Objekts einstellen.

Die `on new`-Prozedur beginnt immer mit dem Ausdruck `on new,` gefolgt von der Variablen `me` und Parametern, die an das neue Child-Objekt übergeben werden.

Die folgende `on new`-Prozedur erstellt ein neues Child-Objekt anhand des Parent-Skripts und initialisiert seine `spriteNum`-Eigenschaft mit dem im Parameter `aSpriteNum` übergebenen Wert. Die Anweisung `return me` gibt das Child-Objekt an die Prozedur zurück, in der die `on new`-Prozedur ursprünglich aufgerufen wurde.

```
-- Lingo syntax
property spriteNum

on new me, aSpriteNum
    spriteNum = aSpriteNum
    return me
end
```

Weitere Informationen über das Aufrufen der `on new`-Prozedur finden Sie unter „Erstellen von Child-Objekten“ auf Seite 54.

Hinzufügen weiterer Prozeduren

Sie legen das Verhalten eines Child-Objekts fest, indem Sie die Prozeduren, die das gewünschte Verhalten produzieren, in das Parent-Skript aufnehmen. Sie können beispielsweise eine Prozedur hinzufügen, welche die Sprite-Farbe ändert.

Das folgende Parent-Skript definiert einen Wert für die Eigenschaft `spriteNum` und enthält eine zweite Prozedur, die die Eigenschaft `foreColor` des Sprites ändert:

```
-- Lingo syntax
property spriteNum

on new me, aSpriteNum
    spriteNum = aSpriteNum
    return me
end

on changeColor me
    spriteNum.foreColor = random(255)
end
```

Verweisen auf das aktuelle Objekt

In der Regel erstellt ein Parent-Skript viele Child-Objekte, von denen jedes mehrere Prozeduren enthält. Die besondere Parametervariable `me` weist die Prozeduren im Child-Objekt an, die Eigenschaften dieses Objekts und nicht diejenigen anderer Child-Objekte zu bearbeiten. Dadurch verwendet eine Prozedur in einem Child-Objekt, die auf Eigenschaften verweist, als Werte für diese Eigenschaften diejenigen ihres eigenen Child-Objekts.

Das Element `me` muss in den einzelnen Prozedurdefinitionen eines Parent-Skripts immer als erste Parametervariable angegeben werden. Aus diesem Grunde ist es immer wichtig, `me` als ersten Parameter eines Parent-Skripts zu definieren und diesen Parameter zu übergeben, wenn Sie andere Prozeduren im selben Parent-Skript aufrufen, da es sich bei diesen um die Prozeduren in den einzelnen Child-Objekten des Skripts handelt.

Wenn Sie auf Eigenschaften verweisen, die in Ancestor-Skripts definiert werden, müssen Sie den Parameter `me` als Verweisquelle verwenden. Der Grund hierfür ist, dass es sich bei der Eigenschaft um eine Eigenschaft des Child-Objekts handelt, auch wenn sie im Ancestor-Skript definiert wird. Die folgende Anweisung verwendet beispielsweise den Parameter `me`, um auf ein Objekt zu verweisen und auf die in einem Ancestor-Skript dieses Objekts definierten Eigenschaften zuzugreifen:

```
-- Lingo syntax
x = me.y -- access ancestor property y
```

Da die Variable `me` in sämtlichen Prozeduren des Child-Objekts vorhanden ist, steuern alle diese Prozeduren dasselbe Child-Objekt.

Erstellen von Child-Objekten

Child-Objekte existieren nur im Arbeitsspeicher und werden nicht zusammen mit einem Film gespeichert. Auf dem Datenträger befinden sich nur Parent- und Ancestor-Skripts.

Wenn Sie ein neues Child-Objekt erstellen möchten, verwenden Sie die `new()`-Methode und weisen dem Child-Objekt einen Variablennamen oder eine Position in einer Liste zu, damit Sie es später identifizieren und mit ihm arbeiten können.

Verwenden Sie beim Erstellen eines Child-Objekts und Zuordnen zu einer Variablen die folgende Syntax:

```
-- Lingo syntax
variableName = new(script "scriptName", parameter1, parameter2, ...)
```

Der Parameter *scriptName* ist der Name des Parent-Skripts. *parameter1*, *parameter2*, ... sind Parameter, die Sie an die `on new`-Prozedur des Child-Objekts übergeben. Die `new()`-Methode erstellt ein Child-Objekt mit dem Ancestor-Skript *Skriptname*. Anschließend ruft sie die `on new`-Prozedur im Child-Objekt auf und übergibt an sie die angegebenen Parameter.

Sie können eine `new()`-Anweisung von jeder beliebigen Position im Film aus erteilen und die Anfangseinstellungen des Child-Objekts anpassen, indem Sie die Werte der in der `new()`-Anweisung übergebenen Parameter ändern.

Jedes Child-Objekt benötigt nur so viel Speicherplatz, dass es die aktuellen Werte seiner Eigenschaften und Variablen und einen Verweis auf das Parent-Skript aufzeichnen kann. Aus diesem Grund lassen sich in den meisten Fällen beliebig viele Child-Objekte erstellen und verwalten.

Sie können weitere Child-Objekte eines Parent-Skripts erzeugen, indem Sie zusätzliche `new()`-Anweisungen aufrufen.

Wenn Sie ein Child-Objekt erstellen, seine Eigenschaftsvariablen jedoch nicht sofort initialisieren möchten, verwenden Sie die Methode `rawNew()`. Die `rawNew()`-Methode erstellt das Child-Objekt, ohne die `on new`-Prozedur des Parent-Skripts aufzurufen. In Situationen, in denen Sie eine größere Anzahl Child-Objekte benötigen, können Sie die Objekte mit `rawNew()` vorab erstellen und die Eigenschaftswerte für die einzelnen Objekte erst dann zuweisen, wenn Sie sie benötigen.

Die folgende Anweisung erstellt ein Child-Objekt anhand des Parent-Skripts „Car“, ohne seine Eigenschaftswerte zu initialisieren, und ordnet es der Variablen `car1` zu:

```
-- Lingo syntax
car1 = script("Car").rawNew()
```

Wenn Sie die Eigenschaften eines dieser Child-Objekte ändern möchten, rufen Sie dessen `on new`-Prozedur auf.

```
car1.new
```

Prüfen von Child-Objekteigenschaften

Wenn Sie die Werte bestimmter Eigenschaftsvariablen in einzelnen Child-Objekten überprüfen möchten, verwenden Sie eine einfache *Objektname.Eigenschaftsname*-Syntax. Die folgende Anweisung zum Beispiel weist der Variablen `x` den Wert der Eigenschaft `carSpeed` des Child-Objekts in der Variablen `car1` zu:

```
-- Lingo syntax
x = car1.carSpeed
```

Durch Abfragen von Objekteigenschaften außerhalb der jeweiligen Objekte lassen sich auf einfache Art und Weise Informationen zu bestimmten Gruppen von Objekten ermitteln, so zum Beispiel die Durchschnittsgeschwindigkeit aller Fahrzeuge in einem Autorennspiel. Außerdem können Sie anhand der Eigenschaften eines Objekts oftmals feststellen, wie sich andere von ihm abhängige Objekte verhalten werden.

Neben den zugewiesenen Eigenschaften können Sie auch überprüfen, ob ein Child-Objekt eine bestimmte Prozedur enthält, oder feststellen, von welchem Parent-Skript es stammt. Dies ist beispielsweise dann sinnvoll, wenn Sie über mehrere Objekte aus verschiedenen Parent-Skripts verfügen, die trotz großer Ähnlichkeit untereinander sehr geringfügige Unterschiede aufweisen.

Angenommen, Sie möchten ein Szenario erstellen, in dem eines von mehreren Parent-Skripts verwendet werden kann, um ein Child-Objekt anzulegen. In diesem Fall können Sie mit der Funktion `script()` den Namen des Parent-Skripts abrufen, von dem ein bestimmtes Child-Objekt abstammt.

Die folgenden Anweisungen überprüfen, ob das Objekt `car1` vom Parent-Skript `Car` erstellt wurde:

```
-- Lingo syntax
if car1.script = script("Car") then
    _sound.beep()
end if
```

Sie können auch eine Liste der Prozeduren in einem Child-Objekt abrufen, indem Sie die Methode `handlers()` verwenden, oder mit der Methode `handler()` prüfen, ob ein Child-Object eine bestimmte Prozedur enthält.

Die folgende Anweisung platziert eine Liste der Prozeduren im Child-Objekt `car1` in der Variablen `myHandlerList`:

```
-- Lingo syntax
myHandlerList = car1.handlers()
```

Die resultierende Liste sieht in etwa folgendermaßen aus:

```
[#start, #accelerate, #stop]
```

Die folgenden Anweisungen überprüfen mit der Methode `handler()`, ob das Child-Objekt `car1` die Prozedur `on accelerate` enthält:

```
-- Lingo syntax
if car1.handler(#accelerate) then
    put("The child object car1 contains the handler named on accelerate.")
end if
```

Entfernen von Child-Objekten

Sie können ein Child-Objekt aus einem Film entfernen, indem Sie alle Variablen, die auf das Child-Objekt verweisen, auf einen anderen Wert einstellen. Wenn das Child-Objekt einer Liste (z. B. `actorList`) zugeordnet wurde, müssen Sie es außerdem aus dieser Liste entfernen.

So entfernen Sie ein Child-Objekt und die Variablen, die auf das Objekt verweisen

❖ Stellen Sie alle Variablen auf `VOID` ein.

Director löscht das Child-Objekt, wenn keine Verweise auf das Objekt mehr vorhanden sind. Im folgenden Beispiel enthält `ball1` den einzigen Verweis auf ein bestimmtes Child-Objekt und ist auf `VOID` festgelegt, um das Objekt aus dem Arbeitsspeicher zu löschen:

```
-- Lingo syntax
ball1 = VOID
```

So entfernen Sie ein Objekt aus „actorList“

❖ Verwenden Sie die `delete()`-Methode, um das Element aus der Liste zu löschen.

Verwenden von „scriptInstanceList“

Mithilfe der Eigenschaft `scriptInstanceList` können Sie einem Sprite neue Verhalten dynamisch hinzufügen. Normalerweise handelt es sich bei der Eigenschaft `scriptInstanceList` um die Liste der Verhaltensinstanzen, die von dem im Drehbuch definierten Initialisierer erstellt wurden. Wenn Sie dieser Liste Child-Objekte hinzufügen, die aus Parent-Skripts erstellt wurden, erhalten die Child-Objekte die an andere Verhalten gesendeten Nachrichten.

Die folgende Anweisung fügt beispielsweise der Eigenschaft `scriptInstanceList` von Sprite10 ein Child-Objekt hinzu:

```
-- Lingo syntax
add(sprite(10).scriptInstanceList, new(script "rotation", 10))
```

Das Parent-Skript, auf das diese Anweisung verweist, kann folgendermaßen aussehen:

```
-- Lingo syntax parent script "rotation"
property spriteNum

on new me, aSpriteNum
    spriteNum = aSpriteNum
    return me
end

on prepareFrame me
    sprite(spriteNum).rotation = sprite(spriteNum).rotation + 1
end
```

Beim Hinzufügen von Child-Objekten zur `scriptInstanceList` muss die Eigenschaft `spriteNum` des Child-Objekts initialisiert werden. Normalerweise erfolgt dies über einen mit der Prozedur `on new` übergebenen Parameter.

Hinweis: *Die Nachricht `beginSprite` wird nicht an dynamisch hinzugefügte Child-Objekte gesendet.*

Weiterführende Informationen zu `scriptInstanceList` finden Sie unterscriptInstanceList.

Verwenden von „actorList“

Sie können eine spezielle Liste mit Child-Objekten (oder anderen Objekten) erstellen, die immer dann eine eigene Nachricht erhält, wenn der Abspielkopf in ein Bild eintritt oder die Bühne mit der Methode `updateStage()` aktualisiert wird.

Bei dieser speziellen Liste handelt es sich um die `actorList`, die nur solche Objekte enthält, die ihr ausdrücklich hinzugefügt wurden.

Diese `stepFrame`-Nachricht wird nur dann gesendet, wenn der Abspielkopf in ein Bild eintritt oder der Befehl `updateStage()` aufgerufen wird.

Die in der `actorList` aufgeführten Objekte erhalten bei jedem Bild anstelle einer `enterFrame`-Nachricht eine `stepFrame`-Nachricht. Wenn die Objekte über eine `on stepFrame`-Prozedur verfügen, werden die in der Prozedur enthaltenen Skriptanweisungen immer dann ausgeführt, wenn der Abspielkopf in ein neues Bild eintritt oder die Bühne mit der `updateStage()` aktualisiert wird.

Die Prozeduren `actorList` und `stepFrame` können beispielsweise verwendet werden, um Child-Objekte zu animieren, die als Sprites genutzt werden, oder einen Zähler zu aktualisieren, der verfolgt, wie oft der Abspielkopf in ein Bild eintritt.

Beides wäre auch mit einer `on enterFrame`-Prozedur möglich, doch die Eigenschaft `actorList` und die Prozedur `stepFrame` sind speziell für den Einsatz in Director optimiert. Objekte in der `actorList` reagieren auf `stepFrame`-Nachrichten wesentlich effizienter als auf `enterFrame`- oder benutzerdefinierte Nachrichten, die auf eine `updateStage()`-Methode hin gesendet werden.

So nehmen Sie Objekte in „actorList“ auf

❖ Verwenden Sie die `actorList`-Eigenschaft wie folgt, wobei *childObject* ein Verweis auf das hinzuzufügende Child-Objekt ist:

```
-- Lingo syntax
_movie.actorList.add(childObject)
```

Diese Anweisung bewirkt, dass die `stepFrame`-Prozedur des Objekts in seinem Parent- oder Ancestor-Skript automatisch jedesmal ausgeführt wird, wenn der Abspielkopf vorrückt. Das Objekt wird als der erste Parameter (`me`) an die `on stepFrame`-Prozedur übergeben.

Der Inhalt von `actorList` wird beim Verzweigen zu einem anderen Film nicht gelöscht, was in dem neuen Film zu unvorhersehbaren Verhaltensweisen führen kann. Wenn Sie verhindern möchten, dass Child-Objekte aus dem aktuellen Film in den neuen Film übertragen werden, fügen Sie eine Anweisung ein, die `actorList` in der `on prepareMovie`-Prozedur des neuen Films löscht.

So löschen Sie Child-Objekte aus „actorList“

❖ Stellen Sie `actorList` auf `[ ]` und damit auf eine leere Liste ein.

Weiterführende Informationen zu `actorList` finden Sie unter actorList

Erstellen von Timeout-Objekten

Ein Timeout-Objekt ist ein Skriptobjekt, das als Timer fungiert und nach Ablauf einer bestimmten Zeitspanne eine Nachricht sendet. Ein solcher Timer wird in Szenarien benötigt, in denen in regelmäßigen Zeitabständen oder nach Ablauf einer zuvor festgelegten Zeit bestimmte Dinge geschehen sollen.

Timeout-Objekte können Nachrichten versenden, mit denen Prozeduren in Child-Objekten oder Filmskripten aufgerufen werden, und werden mit dem Schlüsselwort `new()` erstellt. Hierbei müssen Sie einen Namen für das Objekt, eine aufzurufende Prozedur sowie die Frequenz angeben, mit der die Prozedur aufgerufen werden soll. Mit dem ersten Timeout-Objekt, das Sie anlegen, richtet Director eine Liste aktiver Timeout-Objekte namens `timeOutList` ein, die laufend aktualisiert wird.

Die nachfolgend beschriebene Syntax ist für alle neuen in Adobe Director 11 erstellten Filme bzw. für ältere mit Adobe Director 11 wiederzugebende Filme erforderlich, deren `scriptExecutionStyle`-Eigenschaft auf den Wert 10 festgelegt ist. Bei in Director MX und früher erstellten Filmen ist die`scriptExecutionStyle`-Eigenschaft auf 9 festgelegt, wodurch die Verwendung der Syntax für Director MX und früher ermöglicht wird.

So erstellen Sie Timeout-Objekte

```
-- Lingo syntax when scriptExecutionStyle is set to 9
variableName = timeout().new(timeoutName, timeoutPeriod, #timeoutHandler, {, targetObject})

-- Lingo syntax when scriptExecutionStyle is set to 10
variableName = timeout().new(timeoutName, timeoutPeriod, timeoutHandler, targetObject)
variableName = new timeout(timeoutName, timeoutPeriod, timeoutHandler, targetObject)

// JavaScript syntax
variableName = new timeout(timeoutName, timeoutPeriod, timeoutFunction, targetObject)
```

Diese Anweisung enthält die folgenden Elemente:

- *variableName* ist der Name der Variablen, in der Sie das Timeout-Objekt platzieren.
- `timeOut` gibt den Typ des Lingo-Objekts an, das Sie erstellen.
- *timeoutName* ist der von Ihnen zugewiesene Name des Timeout-Objekts, unter dem es in der `timeOutList` aufgeführt wird. Bestimmt die Eigenschaft `#name` des Objekts.
- `new` erstellt ein neues Objekt.
- *intMilliseconds* gibt an, in welchem Intervall das Timeout-Objekt die angegebene Prozedur aufrufen soll. Bestimmt die Eigenschaft `#period` des Objekts. Beim Wert 2000 wird beispielsweise die angegebene Prozedur alle zwei Sekunden aufgerufen.
- #*handlerName* ist der Name der Prozedur, die das Objekt aufrufen soll. Bestimmt die Eigenschaft `#timeOutHandler` des Objekts. Der Name ist durch das vorangestellte Rautezeichen # als Symbol gekennzeichnet. Eine Prozedur namens `on accelerate` wird beispielsweise als `#accelerate` angegeben.
- *targetObject* gibt an, welche Child-Objektprozedur aufgerufen werden soll. Bestimmt die Eigenschaft `#target` des Objekts. Dieser Parameter bezeichnet ein bestimmtes Child-Objekt, wenn mehrere Child-Objekte dieselben Prozeduren enthalten. Wenn Sie diesen Parameter nicht übergeben, sucht Director im Filmskript nach der angegebenen Prozedur.

Die folgende Anweisung erstellt ein Timeout-Objekt namens `timer1`, das in Abständen von jeweils zwei Sekunden die Prozedur `on accelerate` im Child-Objekt `car1` aufruft:

```
-- Lingo syntax
myTimer = timeOut("timer1").new(2000, #accelerate, car1)
```

Wenn Sie feststellen möchten, wann die nächste Timeout-Nachricht von einem bestimmten Timeout-Objekt gesendet wird, fragen Sie dessen Eigenschaft `#time` ab. Der Rückgabewert gibt den Zeitpunkt in Millisekunden an, zu dem die nächste Timeout-Nachricht gesendet wird. Die folgende Anweisung ermittelt beispielsweise den Zeitpunkt, zu dem das Timeout-Objekt `timer1` die nächste Timeout-Nachricht senden wird, und zeigt das Ergebnis im Nachrichtenfenster an:

```
-- Lingo syntax
put(timeout("timer1").time)
```

Verwenden von „timeOutList“

Mithilfe von `timeOutList` können Sie feststellen, wie viele Timeout-Objekte zum jeweiligen Zeitpunkt aktiv sind.

Die folgende Anweisung stellt die Variable `x` mithilfe der `count`-Eigenschaft auf die Anzahl der Objekte in `timeOutList` ein:

```
-- Lingo syntax
x = _movie.timeoutList.count
```

Sie können für einen Verweis auf ein bestimmtes Timeout-Objekt auch dessen Nummer in der Liste verwenden.

Die folgende Anweisung löscht das zweite Timeout-Objekt in `timeOutList` mithilfe der `forget()`-Methode:

```
-- Lingo syntax
timeout(2).forget()
```

Weiterleiten von Systemereignissen mit Timeout-Objekten

Wenn Sie Timeout-Objekte erstellen, die bestimmte Child-Objekte zum Ziel haben, können diese Child-Objekte Systemereignisse empfangen. Die Ereignisse werden von den Timeout-Objekten an die entsprechenden Child-Objekte weitergeleitet. Folgende Systemereignisse können von Child-Objekten empfangen werden: `prepareMovie`, `startMovie`, `stopMovie`, `prepareFrame` und `exitFrame`. Durch Einbinden von Prozeduren für diese Ereignisse in Ihre Child-Objekte können Sie festlegen, in welcher Weise diese auf die Ereignisse reagieren. Die Systemereignisse, die die Child-Objekte empfangen, werden auch Filmskripten, Bildskripten und anderen für die Reaktion auf Systemereignisse konzipierte Skripts gemeldet.

Das folgende Parent-Skript enthält eine Prozedur für das Systemereignis `exitFrame` sowie die benutzerdefinierte Prozedur `slowDown`:

```
-- Lingo syntax
property velocity

on new me
    velocity = random(55)
end

on exitFrame
    velocity = velocity + 5
end

on slowDown mph
    velocity = velocity - mph
end
```

Verknüpfen benutzerdefinierter Eigenschaften mit Timeout-Objekten

Wenn Sie benutzerdefinierte Eigenschaften mit einem Timeout-Objekt verknüpfen möchten, können Sie ein Timeout-Objekt erstellen, das als Ziel ein anderes Element als einen Verweis auf ein Skriptinstanzobjekt nutzt. Bei dieser Vorgehensweise werden die Zieldaten mit dem Timeout-Objekt verknüpft, wodurch sie in Ihrer Timeout-Prozedur verwendet werden können.

Dies wird in der folgenden Abbildung veranschaulicht:

```
-- Lingo syntax
-- initialize a timeout object and pass it a data property list (tData)
-- instead of a reference to a script instance object
tData = [#beta: 0]
tTO = timeout("betaData").new(50,#targetHandler,tData)

-- within a movie script, create the targetHandler handler
on targetHandler (aData)
    -- increment and display the beta property
    tData.beta = tData.beta + 1
    put(tData.beta)
end targetHandler
```

Im vorherigen Beispiel wird die Eigenschaft `beta` stetig erhöht. Dies bedeutet, dass Sie mehrere Timeout-Objekte initialisieren können, die alle dieselbe Filmskriptprozedur aufrufen. Jedem Timeout-Objekt kann eine eigene Datenliste zugeordnet sein.

Beachten Sie im Allgemeinen Folgendes:

- Bei Verwenden eines Verweises auf eine Skriptinstanz als Ziel wird die Zielprozedur in dieser bestimmten Skriptinstanz aufgerufen. Dieses Verfahren lässt keine benutzerdefinierten Eigenschaften zu.

- Bei Verwenden eines Verweises auf ein anderes Element als eine Skriptinstanz (z. B. eine Eigenschaftenliste) als Ziel wird die Zielprozedur in einem Filmskript aufgerufen. Dieses Verfahren lässt benutzerdefinierten Eigenschaften zu.

Objektorientierte Programmierung mithilfe von JavaScript

Die objektorientierte Programmierung in JavaScript unterscheidet sich von anderen objektorientierten, klassenbasierten Sprachen wie Java und C++, da JavaScript prototypbasiert ist.

Nachfolgend werden klassenbasierte Sprachen und prototypbasierte Sprachen wie JavaScript aus übergeordneter Sicht verglichen und die Unterschiede herausgestellt:

- Bei klassenbasierten Sprachen erstellen Sie Klassendefinitionen, die die anfänglichen Eigenschaften und Methoden festlegen und für alle anhand dieser Klassen erstellten Instanzen gelten. Eine Klassendefinition enthält spezielle *Konstruktormethoden* genannte Methoden, mit deren Hilfe die Instanzen der jeweiligen Klasse erstellt werden. Wenn eine Instanz mit dem Operator `new` in Verbindung mit einer bestimmten Konstruktormethode erstellt wird, übernimmt diese Instanz alle Eigenschaften ihrer übergeordneten Klasse. Diese Instanz kann auch abhängig vom aufgerufenen Konstruktor andere Verarbeitungsaufgaben ausführen, die für diese Instanz spezifisch sind.

  In einer Klassendefinition erstellen Sie eine Unterklasse, die alle Eigenschaften der übergeordneten Klassen übernimmt, und können daneben neue Eigenschaften festlegen und übernommene Eigenschaften optional ändern. Die übergeordnete Klasse, aus der eine Unterklasse erstellt wird, heißt auch *Superklasse*.

- Bei prototypbasierten Sprachen wie JavaScript wird nicht zwischen Klassen, Instanzen, Unterklassen usw. unterschieden, die allesamt als Objekte bezeichnet werden. Anstatt Klassendefinitionen werden in JavaScript-Syntax *Prototypobjekte* als Vorlage verwendet, anhand der neue Objekte erstellt werden. Ähnlich wie bei klassenbasierten Sprachen erstellen Sie in JavaScript ein neues Objekt mithilfe des Operators `new` in Verbindung mit einer Konstruktorfunktion.

  Anstatt mit Super- und Unterklassen zu arbeiten, verknüpfen Sie in JavaScript Prototypobjekte mit Konstruktorfunktionen, um die Übernahme von Eigenschaften zu erreichen. Dieser Prozess ähnelt sehr dem Arbeiten mit Super- und Unterklassen, lediglich mit anderer Terminologie.

  Im Gegensatz zu klassenbasierten Sprachen können Sie in JavaScript auch Eigenschaften zur Laufzeit einem Objekt bzw. einer Objektgruppe hinzufügen oder aus diesem/dieser entfernen. Wenn Sie beispielsweise eine Eigenschaft zur Laufzeit einem Prototypobjekt hinzufügen, erhalten alle Instanzobjekte, für die dieses ein Prototyp ist, ebenfalls diese Eigenschaft.

Objektorientierte Terminologie

Da alle Typen in JavaScript Objekte genannt werden, gibt es für klassenbasierte Begriffe wie *Superklasse*, *Unterklasse*, *Klasse*, *Instanz* usw. in JavaScript keine Entsprechungen. Diese Begriffe werden jedoch im Wesentlichen Objekten in JavaScript zugeordnet und eignen sich zur allgemeinen Verwendung, um auf die verschiedenen Typen von JavaScript-Objekten zu verweisen. Deshalb werden diese klassenbasierten Begriffe im Verlauf der Diskussion der objektorientierten Programmierung in JavaScript synonym mit *Objekt* verwendet und bedeuten Folgendes:

Superklasse Klasse, anhand der Unterklassen (Objekte) erstellt werden (Parent-Klasse).

Unterklasse Beliebige Klasse, die aus einer Superklasse (einem Objekt) erstellt wird (Child-Klasse).

Klasse Allgemeiner Begriff für Super- oder Unterklasse, eine Parent- oder Child-Klasse.

Instanz oder Objektinstanz Ein einzelnes Objekt, dass von einer Superklasse abgeleitet wurde.

Benutzerdefinierte Klassen

Einer der Hauptvorteile der objektorientierten Programmierung ist die Fähigkeit, eigene benutzerdefinierte Klassen zu erstellen, über die Sie Ihren Skripts selbstdefinierte Funktionalität hinzufügen können. Die in JavaScript vordefinierten Klassen wie Object, String, Math usw. sind in manchen Fällen nützlich, bieten aber ggf. nicht die Funktionalität, die Sie zum Erledigen der gewünschten Aufgabe benötigen. Angenommen, Sie möchten, dass einige Objekte in Ihrem Film verschiedene Transportmöglichkeiten wie Autos, Boote, Flugzeuge usw. darstellen und dass für jede Kategorie eindeutige Merkmale und Funktionen angezeigt werden. Weder die vordefinierten JavaScript-Klassen noch die vordefinierten Director-Objekte bieten dazu unmittelbar die benötigte Funktionalität. Deshalb können Sie für jede Transportmöglichkeit eine neue Klasse mit eindeutigen Merkmalen für jeden Typ erstellen.

Beachten Sie, dass Sie beim Erstellen benutzerdefinierter JavaScript-Klassen weiterhin Zugriff auf alle Funktionen der vordefinierten Director-Objekte haben. Wenn also die vordefinierten Director-Objekte die gewünschte Funktionalität nicht direkt bieten sollten, können Sie sie dennoch in Ihren benutzerdefinierten Klassen für den Zugriff auf ihre Werte und die vordefinierte Funktionalität nutzen.

Konstruktorfunktionen

In JavaScript stellt eine Konstruktorfunktion die Klasse dar, welche die Vorlage enthält, anhand der neue Objektinstanzen erstellt werden. Konstruktorfunktionen dienen der Erstellung und Initialisierung (d. h. Festlegung des Standardstatus) von Eigenschaften in den neuen Objekten.

Das Format von Konstruktorfunktionen ist im Wesentlichen mit dem herkömmlicher JavaScript-Methodenfunktionen identisch. Der Unterschied zwischen einer Konstruktor- und einer Methodenfunktion besteht darin, dass eine Konstruktorfunktion das besondere Schlüsselwort `this` verwendet, um einen Verweis auf das neue Objekt abzubilden, das initialisiert wird. Eine Methodenfunktion wendet zumeist nur eine Aktion auf eine angegebene Menge von Objektdaten an.

Das folgende Beispiel veranschaulicht eine Möglichkeit zum Erstellen der Konstruktorfunktion Rectangle zum Initialisieren der Höhe und Breite neuer Rectangle-Objekte:

```
function Rectangle(w, h) {
    this.width = w;
    this.height = h;
}
```

Sie können eine Konstruktorfunktion auch über die *Funktion/Literal*-Syntax erstellen. Funktion/Literal-Syntax bietet dieselbe Funktionalität wie die zuvor beschriebene Syntax und stellt lediglich eine Alternative beim Erstellen des Konstruktors dar. Das folgende Beispiel veranschaulicht das Verwenden von Funktion/Literal-Syntax zum Erstellen der Konstruktorfunktion Rectangle ähnlich wie im zuvor beschriebenen Beispiel:

```
Rectangle = function(w, h) {
    this.width = w;
    this.height = h;
}
```

Hinweis: *Beim Definieren von Konstruktorfunktionen, die auf einen Film angewendet werden sollen, müssen Sie diese in einem Filmskript ablegen, damit sie global verfügbar sind.*

Bei der Skripterstellung wird empfohlen, Konstruktorfunktionen Namen mit großem Anfangsbuchstaben zu geben, die ihre Funktionalität abbilden, z. B. `Rectangle` oder `Circle`.

Konstruktorfunktionen dienen zumeist nur zum Initialisieren neuer Objekte, können aber nach Wunsch auch das Objekt zurückgeben. Wenn Sie das initialisierte Objekte zurückgeben, wird das zurückgegebene Objekt zum Wert des Ausdrucks `new`.

Objektinstanzen

Die Erstellung einer neuen Objektinstanz erfolgt üblicherweise über den Operator `new`, gefolgt vom Namen einer Konstruktorfunktion. In den folgenden Beispielen werden neue Objektinstanzen erstellt:

```
var objRandom = new Object(); // assigns a reference to an Object object
var objString = new String(); // assigns a reference to a String object
```

Eine Konstruktorfunktion kann optional Parameter definieren, die eine neue Objektinstanz an sie übergibt, um den Status der Objektinstanz zu initialisieren. Wenn eine Konstruktorfunktion Parameter definiert, die bei der Initialisierung neuer Objektinstanzen verwendet werden, werden die Eigenschaftswerte wie folgt initialisiert:

- Wenn während der Initialisierung Werte an die Konstruktorfunktion übergeben werden, werden die Eigenschaften, die Werte empfangen haben, auf diese Werte festgelegt.
- Wenn während der Initialisierung keine Werte an die Konstruktorfunktion übergeben werden, werden die Eigenschaften, die keine Werte empfangen haben, auf `undefined` festgelegt.

Bei Erstellung neuer Objektinstanzen dient das Schlüsselwort `this` in der dazugehörigen Konstruktorfunktion zum Verweisen auf die neue Objektinstanz. Deshalb wird eine neue Objektinstanz mit allen Eigenschaften initialisiert, die mithilfe der Syntax `this.`*Eigenschaftsname* definiert wurden.

Im folgenden Beispiel verwendet die Konstruktorfunktion `Circle` das Schlüsselwort `this` zum Angeben der Namen dreier Eigenschaften, die neuen Objektinstanzen zugeordnet werden. Die auf den Konstruktor folgende Anweisung initialisiert eine neue Objektinstanz durch Übergeben von Werten an den Konstruktor. Diese Werte dienen als Ausgangswerte der Eigenschaften, die vom Schlüsselwort `this` angegeben werden.

```
// Circle constructor function
function Circle(x, y, r) {
    this.xCoord = x;
    this.yCoord = y;
    this.radius = r;
}

// xCoord = 10, yCoord = 15, radius = 5
var objCircle = new Circle(10, 15, 5);
```

Nachdem `objCircle` initialisiert wurde, können Sie auf dessen Eigenschaften zugreifen. Mithilfe der zuvor erstellten Instanz `objCircle` können Sie verschiedene Variablen auf die Werte ihrer Eigenschaften festlegen.

```
var theXCoord = objCircle.xCoord; // assigns the value 10 to theXCoord
var theYCoord = objCircle.yCoord; // assigns the value 15 to theYCoord
var theRadius = objCircle.radius; // assigns the value 5 to theRadius
```

Hinweis: *Weitere Informationen zum Verwenden der Punktsyntax für den Zugriff auf die Eigenschaften und Methoden eines Objekts finden Sie unter* *„Skripterstellung im Punktsyntaxformat“ auf Seite 45.*

Bei der Skripterstellung wird empfohlen, neuen Objekten Namen mit kleinem Anfangsbuchstaben zu geben, die ihre Funktionalität abbilden, z. B. `objRectangle` oder `objCircle`.

Sie können eine Objektinstanz auch mithilfe der *Objekt/Literal*-Syntax erstellen, bei welcher der Operator `new` und eine Konstruktorfunktion nicht erforderlich sind. Sie gehen im Allgemeinen nur so vor, wenn Sie nur eine Instanz eines Objekts benötigen, das in der Konstruktorfunktion nicht definiert wurde. Das folgende Beispiel erstellt eine Objektinstanz mit `x = 1`, `y = 2` und `radius = 2`:

```
var objSmallCircle = { x:1, y:2, radius:2 };
```

Objektübernahme

Zusätzlich zur Fähigkeit, eigene benutzerdefinierte Klassen zu erstellen, besteht ein weiterer großer Vorteil der objektorientierten Programmierung darin, dass Unterklassen die Eigenschaften und Methoden der Superklassen, anhand denen Sie erstellt wurden, zu übernehmen. Mithilfe dieser Übernahmefunktionalität können Sie Objekte erstellen, die über vordefinierte Eigenschaften und Funktionen verfügen.

In JavaScript gibt es eine Superklasse, die als Basisklasse aller anderen erstellten Unterklassen fungiert: die Superklasse „Object", die einige grundlegende Eigenschaften und Methoden enthält. Die anhand von Object als Vorlage erstellten Unterklassen übernehmen stets diese grundlegenden Eigenschaften und Methoden und definieren zumeist eigene Eigenschaften und Methoden. Unterklassen dieser Klassen übernehmen Elemente von Object, von ihren Superklassen usw. Alle weiteren Objekte, die Sie erstellen, setzen diese Übernahmekette fort.

Object enthält beispielsweise die `constructor`-Eigenschaft und die `toString()`-Methode. Wenn Sie die neue Klasse `SubObj1`erstellen, ist diese eine Unterklasse von „Object" und übernimmt deshalb automatisch die `constructor`-Eigenschaft und die Methode `toString()` der Klasse „Object". Wenn Sie anschließend die weitere Klasse `SubObj2` unter Verwendung von `SubObj1` als Superklasse erstellen, übernimmt `SubObj2` ebenfalls die `constructor`-Eigenschaft und die `toString()`-Methode von Object sowie zusätzlich alle benutzerdefinierten Eigenschaften und Methoden, die Sie in `SubObj1` definiert haben.

Zwei der wichtigen Eigenschaften, die Ihre benutzerdefinierte Konstruktorfunktionen von der Superklasse Object übernehmen, sind `prototype` und `constructor`. Die Eigenschaft `prototype` stellt das Prototypobjekt einer Klasse dar, welches das Hinzufügen von Variablen (Eigenschaften) und Methoden zu Objektinstanzen ermöglicht. Sie dient auch zum Implementieren der Übernahmefunktion in JavaScript. Die Eigenschaft `constructor` stellt die Konstruktorfunktion selbst dar. Die Verwendung dieser Eigenschaft wird in den folgenden Abschnitten erklärt.

Prototypobjekte

Wie zuvor erwähnt, übernimmt eine von Ihnen erstellte Unterklasse automatisch die Eigenschaften und Methoden der Superklasse, auf der sie basiert. In JavaScript wird diese Übernahme im Allgemeinen über Prototyp-Objekte implementiert. Eine Unterklasse übernimmt ihre Eigenschaft und Methoden vom Prototypobjekt ihrer Superklasse und nicht von der Superklasse selbst. Dies erweist sich als ausgesprochener Vorteil, da alle Eigenschaften und Methoden nicht explizit aus einer Klasse in eine Objektinstanz der Klasse kopiert werden müssen, wodurch der Arbeitsspeicherbedarf neuer Objektinstanzen drastisch reduziert wird.

Jede Klasse in JavaScript, einschließlich der vordefinierten Klasse Object, enthält nur ein Prototypobjekt. Jede anhand einer Klasse erstellte Objektinstanz hat Zugriff auf die Eigenschaften und Methoden im Prototypobjekt der jeweiligen Klasse. Deshalb ist das Prototypobjekt einer Klasse in der Regel das einzige Objekt, in dem die Eigenschaften und Methoden der jeweiligen Klasse tatsächlich gespeichert werden. Eine Objektinstanz enthält nur die Eigenschaften, die für die Initialisierung der jeweiligen Instanz erforderlich sind.

In Ihrem enthält jede Objektinstanz anscheinend tatsächlich diese Eigenschaften und Methoden, da Sie aus jeder Objektinstanz darauf zugreifen können, doch die Instanz nutzt für den Zugriff tatsächlich das Prototypobjekt. Das Prototypobjekt einer Klasse wird automatisch bei ihrer Erstellung angelegt. Sie greifen über die `prototype`-Eigenschaft der Klasse auf das Prototypobjekt zu.

Da ein Prototypobjekt einer Klasse Eigenschaften speichert, die von allen Objektinstanzen gemeinsam genutzt werden, eignen sich diese ideal zum Definieren von Eigenschaften und Methoden, deren Werte von allen Objektinstanzen gemeinsam genutzt werden. Durch das gemeinsame Nutzen von Eigenschaften und Methoden durch Objektinstanzen können Sie mühelos Instanzen erstellen, die ein definiertes Standardverhalten aufweisen, und können auch Instanzen anpassen, die vom Standardverhalten abweichen.

Prototypobjekte sind im Allgemeinen nicht zum Definieren von Eigenschaften und Methoden geeignet, deren Werte bei verschiedenen Objektinstanzen unterschiedlich sind. In Fällen, in denen dies zutrifft, legen Sie zumeist diese Eigenschaften und Methoden in der Klasse selbst fest.

Um den Gültigkeitsbereich einer benutzerdefinierten Eigenschaft oder Methode festzulegen, wird diese als Instanzvariable, Instanzmethode, Klassenvariable oder Klassenmethode definiert.

Instanzvariablen

Instanzvariablen sind Variablen (Eigenschaften), die in einer Konstruktorfunktion definiert und in jede Objektinstanz des jeweiligen Konstruktors kopiert werden. Alle Objektinstanzen verfügen über eigene Kopien von Instanzvariablen. Wenn es fünf Objektinstanzen der Klasse `Circle` gibt, sind in der Klasse fünf Kopien aller Instanzvariablen definiert. Da jede Objektinstanz über eine eigene Kopie einer Instanzvariablen verfügt, kann jede Objektinstanz einer Instanzvariablen einen eindeutigen Wert zuweisen, ohne die Werte anderer Kopien der Instanzvariablen zu ändern. Sie können aus den enthaltenden Objektinstanzen direkt auf Instanzvariablen zugreifen.

Im folgenden Beispiel werden vier Instanzvariablen in einer Konstruktorfunktion definiert: `make`, `model`, `color` und `speed`. Diese vier Instanzvariablen stehen in allen Objektinstanzen des Konstruktors `Car` direkt zur Verfügung:

```
function Car(make, model, color) { // define a Car class
    this.make = make;
    this.model = model;
    this.color = color;
    this.speed = 0;
}
```

Die folgende Objektinstanz `objCar` enthält alle vier Instanzvariablen. Obgleich kein Wert für die Instanzvariable `speed` an den Konstruktor `Car` übergehen wird, verfügt `objCar` weiter über die Eigenschaft `speed`, deren Ausgangswert 0 ist, da die Variable `speed` im Konstruktor `Car` definiert ist.

```
// objCar.make="Subaru", objCar.model="Forester",
// objCar.color="silver", objCar.speed = 0
var objCar = new Car("Subaru", "Forester", "silver");
```

Instanzmethoden

Instanzmethoden sind Methoden, auf die über eine Objektinstanz zugegriffen werden kann. Objektinstanzen verfügen nicht über eigene Kopien von Instanzmethoden. Stattdessen werden Instanzmethoden zuerst als Funktionen definiert. Anschließend werden Eigenschaften des Prototypobjekts der Konstruktorfunktion auf die Funktionswerte festgelegt. Instanzmethoden arbeiten mit dem Schlüsselwort `this` in der definierenden Konstruktorfunktion, um auf die Objektinstanz zu verweisen, auf die sie angewendet werden. Obgleich eine gegebene Objektinstanz nicht über eine Kopie einer Instanzmethode verfügt, können Sie dennoch direkt aus den dazugehörigen Objektinstanzen auf Instanzmethoden zugreifen.

Das folgende Beispiel definiert die Funktion `Car_increaseSpeed()`. Der Funktionsname wird anschließend der Eigenschaft `increaseSpeed` des Prototypobjekts der Klasse `Car` zugewiesen:

```
// increase the speed of a Car
function Car_increaseSpeed(x) {
    this.speed += x;
    return this.speed;
}
Car.prototype.increaseSpeed = Car_increaseSpeed;
```

Eine Objektinstanz von `Car` kann anschließend auf die `increaseSpeed()`-Methode zugreifen und deren Wert über die folgende Syntax einer Variablen zuweisen:

```
var objCar = new Car("Subaru", "Forester", "silver");
var newSpeed = objCar.increaseSpeed(30);
```

Sie können eine Instanzmethode auch über die Funktion/Literal-Syntax erstellen. Bei Verwenden der Funktion/Literal-Syntax muss keine Funktion definiert und kein Eigenschaftsname dem Funktionsnamen zugewiesen werden.

Das folgende Beispiel verwendet Funktion/Literal-Syntax zum Definieren der Methode `increaseSpeed()`, welche dieselbe Funktionalität wie die zuvor beschriebene Funktion `increaseSpeed()` bietet:

```
// increase the speed of a Car
Car.prototype.increaseSpeed = function(x) {
    this.speed += x;
    return this.speed;
}
```

Klassenvariablen

Klassenvariablen, die auch *statische* Variablen genannt werden, sind Variablen (Eigenschaften), die einer Klasse und nicht einer Objektinstanz zugeordnet sind. Unabhängig von der Anzahl der Objektinstanzen, die anhand einer Klasse erstellt werden, gibt es stets nur eine Kopie einer Klassenvariablen. Klassenvariablen verwenden zur Implementierung der Übernahmefunktion nicht das Prototypobjekt. Sie greifen auf eine Klassenvariable direkt über die Klasse und nicht über eine Objektinstanz zu. Sie müssen eine Klasse in einer Konstruktorfunktion definieren, bevor Sie Klassenvariablen definieren können.

Das folgende Beispiel definiert die beiden Klassenvariablen `MAX_SPEED` und `MIN_SPEED`:

```
function Car() { // define a Car class
    ...
}

Car.MAX_SPEED = 165;
Car.MIN_SPEED = 45;
```

Sie können auf die Klassenvariablen `MAX_SPEED` und `MIN_SPEED` direkt aus der Klasse `Car` zugreifen.

```
var carMaxSpeed = Car.MAX_SPEED; // carMaxSpeed = 165
var carMinSpeed = Car.MIN_SPEED; // carMinSpeed = 45
```

Klassenmethoden

Klassenmethoden, die auch *statische* Methoden genannt werden, sind Methoden, die einer Klasse und nicht einer Objektinstanz zugeordnet sind. Unabhängig von der Anzahl der Objektinstanzen, die anhand einer Klasse erstellt werden, gibt es stets nur eine Kopie einer Klassenmethode. Klassenmethoden verwenden zur Implementierung der Übernahmefunktion nicht das Prototypobjekt. Sie greifen auf eine Klassenvariable direkt über die Klasse und nicht über eine Objektinstanz zu. Sie müssen eine Klasse in einer Konstruktorfunktion definieren, bevor Sie Klassenvariablen definieren können.

Das folgende Beispiel definiert die Funktion `setInitialSpeed()`, welche die Standardgeschwindigkeit neuer Instanzen von „car“ ändern kann. Der Funktionsname wird der Eigenschaft `setInitialSpeed` der Klasse `Car` zugewiesen:

```
function Car(make, model, color) { // define a Car class
    this.make = make;
    this.model = model;
    this.color = color;
    this.speed = Car.defaultSpeed;
}
Car.defaultSpeed = 10; // initial speed for new Car instances
// increase the speed of a Car
function Car_setInitialSpeed(x) {
    Car.defaultSpeed = x;
}
Car.setInitialSpeed = Car_setInitialSpeed;
```

Sie können auf die Klassenmethode `setInitialSpeed()` direkt aus der Klasse `Car` zugreifen.

```
var newSpeed = Car.setInitialSpeed(30);
```

Sie können eine Klassenmethode auch über die Funktion/Literal-Syntax erstellen. Das folgende Beispiel verwendet Funktion/Literal-Syntax zum Definieren der Methode `setInitialSpeed()`, welche dieselbe Funktionalität wie die zuvor beschriebene Funktion `setInitialSpeed()` bietet:

```
// increase the speed of a Car
Car.setInitialSpeed = function(x) {
    Car.defaultSpeed = x;
}
```

Empfohlene Schritte bei der Klassendefinition

Die folgende Auflistung enthält die empfohlenen Schritte bei der Definition einer Klasse:

1. Definieren Sie eine Konstruktorfunktion, die als Vorlage dient, anhand der alle Objektinstanzen initialisiert werden. Sie können außerdem Instanzvariablen in der Konstruktorfunktion mithilfe des Schlüsselworts `this` definieren, um auf eine Objektinstanz zu verweisen.
2. Definieren Sie Instanzmethoden und möglicherweise zusätzliche Instanzvariablen, die im Prototypobjekt einer Klasse gespeichert werden. Diese Instanzmethoden und -variablen stehen alle Objektinstanzen zur Verfügung. Der Zugriff erfolgt über die Prototypobjekt der Klasse.
3. Definieren Sie Klassenmethoden, Klassenvariablen und Konstanten, die in der Klasse selbst gespeichert werden. Der Zugriff auf diese Klassenmethoden- und -variablen erfolgt nur über die Klasse selbst.

Wenn Sie in Ihrem Code auf eine Eigenschaft einer Objektinstanz zugreifen, durchsucht JavaScript die Objektinstanz selbst nach dieser Eigenschaft. Wenn die Instanz die Eigenschaft nicht enthält, durchsucht JavaScript das Prototypobjekt der Superklasse, anhand der die Instanz erstellt wurde. Da eine Objektinstanz vor dem Prototypobjekt der Klasse durchsucht wird, anhand der sie erstellt wurde, werden in den Objektinstanzeigenschaften die Eigenschaften des Prototypobjekts der jeweiligen Superklassen im Wesentlichen ausgeblendet. Dies bedeutet, dass sowohl eine Objektinstanz als auch deren Superklassen realistischerweise eine Eigenschaft mit demselben Namen, aber unterschiedlichen Werten definieren kann.

Löschen von Variablen

Sie können eine Klassen- oder Instanzvariable mit dem Operator `delete` löschen, was nachfolgend veranschaulicht wird.

```
function Car() { // define a Car constructor function
    ...
}
Car.color = "blue"; // define a color property for the Car class
Car.prototype.engine = "V8"; // define an engine property for the prototype

var objCar = new Car();

trace(Car.color); // displays "blue"
trace(objCar.engine); // displays "V8"

delete Car.color;
delete Car.prototype.engine;

trace(Car.color); // displays undefined
trace(objCar.engine); // displays undefined
```

Zugreifen auf die Konstruktoreigenschaft eines Prototypobjekts

Wenn Sie eine Klasse über die Erstellung einer Konstruktorfunktion definieren, erstellt JavaScript ein Prototypobjekt für die jeweilige Klasse. Bei Erstellen des Prototypobjekts enthält es anfänglich die Eigenschaft `constructor`, die auf die Konstruktorfunktion selbst verweist. Mithilfe der `constructor`-Eigenschaft eines Prototypobjekts können Sie den Typ eines Objekts bestimmen.

Im folgenden Beispiel enthält die `constructor`-Eigenschaft einen Verweis auf die Konstruktorfunktion, die zum Erstellen der Objektinstanz verwendet wird. Der Wert der `constructor`-Eigenschaft ist tatsächlich ein Verweis auf den Konstruktor selbst und keine Zeichenfolge, die den Namen des Konstruktors enthält:

```
function Car() { // define a Car class
    // initialization code here
}
var myCar = new Car(); // myCar.constructor == function Car() {}
```

Dynamisches Erstellen von Eigenschaften

Ein weiterer Vorteil der Verwendung von Prototypobjekten zum Implementieren der Übernahmefunktion ist, dass Eigenschaften und Methoden, die einem Prototypobjekt hinzugefügt werden, Objektinstanzen automatisch zur Verfügung gestellt werden. Dies gilt auch, wenn eine Objektinstanz vor dem Hinzufügen der Eigenschaften bzw. Methoden erstellt wurde.

Im folgenden Beispiel wird die Eigenschaft `color` dem Prototypobjekt der Klasse `Car` hinzugefügt, nachdem eine Objektinstanz von `Car` bereits erstellt wurde:

```
function Car(make, model) { // define a Car class
    this.make = make;
    this.model = model;
}
var myCar = new Car("Subaru", "Forester"); // create an object instance

trace(myCar.color); // returns undefined

// add the color property to the Car class after myCar was initialized
Car.prototype.color = "blue";

trace(myCar.color); // returns "blue"
```

Sie können auch Eigenschaft zu Objektinstanzen hinzufügen, nachdem die Instanzen erstellt wurden. Wenn Sie eine Eigenschaft einer bestimmten Objektinstanz hinzufügen, steht diese Eigenschaft nur dieser spezifischen Objektinstanz zur Verfügung. Mithilfe der zuvor erstellten Objektinstanz `myCar` fügen die folgenden Anweisungen die Eigenschaft `color` zu `myCar` hinzu, nachdem diese bereits erstellt wurde:

```
trace(myCar.color); // returns undefined

myCar.color = "blue"; // add the color property to the myCar instance

trace(myCar.color); // returns "blue"

var secondCar = new Car("Honda", "Accord"); // create a second object instance

trace(secondCar.color); // returns undefined
```

Schreiben von Skripts im Skriptfenster

Wenn Sie Skripts für einen Film schreiben, kann die An- und Vielzahl der Skripts enorm groß sein. Je komplexer Ihr Film ist, desto sorgfältiger sollten Sie planen und entscheiden, welche Methoden und Eigenschaften Sie benötigen, wie Sie die Skripts effektiv strukturieren und wo Sie sie sinnvoll platzieren.

Bevor Sie mit dem Schreiben von Skripts beginnen, sollten Sie ein klares Ziel vor Augen haben und wissen, was Sie erreichen möchten. Dies ist ebenso wichtig – und in der Regel ebenso zeitaufwändig – wie die Entwicklung von Storyboards für Ihre Arbeit.

Sobald Sie einen Gesamtplan für Ihren Film haben, können Sie damit beginnen, Skripts zu schreiben und zu testen. Dies nimmt einige Zeit in Anspruch. Bis ein Skript Ihren Vorstellungen entsprechend funktioniert, müssen Sie es nicht selten mehrmals schreiben, testen und debuggen.

Am besten beginnen Sie mit einfachen Skripts und testen Ihre Arbeit häufig. Wenn Sie einen Teil eines Skripts geschrieben haben, der wie gewünscht funktioniert, können Sie zum nächsten Teil übergehen. Auf diese Weise lassen sich Fehler leicht erkennen, und Sie können sicher sein, dass Ihre Skripts ordnungsgemäß sind, bevor Sie mit dem Schreiben komplexerer Skripts beginnen.

Das Skriptfenster bietet verschiedene Funktionen, mit deren Hilfe Sie Skripts erstellen und ändern können.

Im Skriptfenster von Director können Sie Filme um erweiterte, skriptbasierte Interaktivität ergänzen. Im Skriptfenster können Sie entweder mit Lingo- oder JavaScript-Syntax arbeiten. Bei Lingo handelt es sich um die traditionelle Skriptsprache von Director. JavaScript-Unterstützung wurde zuletzt hinzugefügt, um Multimediaentwicklern, die lieber mit JavaScript arbeiten, die Möglichkeit dazu zu bieten.

Durch Erstellen von Skripts im Skriptfenster können viele der Aufgaben ausgeführt werden, die auch auf der grafischen Benutzeroberfläche von Director erfolgen können, z. B. das Verschieben von Sprites auf der Bühne oder Wiedergeben von Sounds. Der wesentliche Vorteil von Skripts liegt jedoch in der Flexibilität, die sie einem Film verleihen. Statt eine Reihe von Bildern exakt wie im Drehbuch vorgeschrieben abzuspielen, kann ein Skript den Film anhand bestimmter Bedingungen und Ereignisse steuern.

Hinweis: *Als Ergänzung zum Skriptfenster, in dem Sie eigene Skripts erstellen können, bietet Director verschiedene vordefinierte Anweisungen (die Verhalten genannt werden), die Sie einfach auf Sprites und Bilder ziehen bzw. anwenden können. Auf diese Weise können Sie die Interaktivität von Skripts nutzen, ohne selbst Skripts schreiben zu müssen. Weitere Informationen zu Verhalten finden Sie unter dem Thema „Verhalten" im Director-Hilfefenster.*

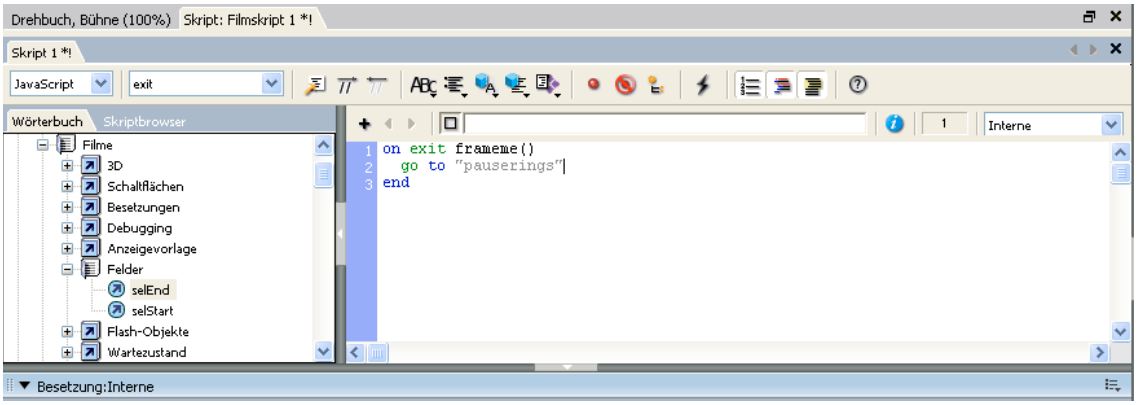
Das Skriptfenster enthält neben dem Skript-Editor einen Explorer-Bereich, der standardmäßig links neben dem Skript-Editor angezeigt wird. Sie können den Explorer-Bereich in der Wörterbuch- oder Skript-Browser-Ansicht anzeigen. Um die Standardposition des Explorer-Bereichs einzustellen, öffnen Sie das Dialogfeld „Skriptfenster-Voreinstellungen“ („Bearbeiten“ > „Voreinstellungen“ > „Skript“).

Wörterbuchansicht

In der Wörterbuchansicht wird eine Liste vordefinierter Lingo-/JavaScript-Funktionen in einer Baumstruktur angeordnet angezeigt. Die Funktionen sind basierend auf ihrer Kategorie und alphabetisch als Index klassifiziert.

In der Wörterbuchansicht können Sie folgende Aufgaben ausführen:

- Die vordefinierten Funktionen für Lingo- und JavaScript-Skripts durchsuchen
- Mithilfe der vordefinierten Funktionen Skripts erstellen



So werden in der Wörterbuchansicht Funktionen durchsucht und Skripts erstellt

1. Wählen Sie „Fenster„ > „Skript“. Das Skriptfenster wird angezeigt.
2. Wählen Sie die Registerkarte „Wörterbuch“ aus.
3. Wählen Sie im Popupmenü „Lingo“ oder „JavaScript“ aus. Im Feld unten werden die entsprechenden vordefinierten Funktionen angezeigt.

   Lingo: Zeigt Funktionen für Lingo-Skripts, 3D-Lingo-Skripts und Skript-Xtras an, die im aktuellen Film verwendet werden. Die Funktionen sind in Kategorien aufgeteilt („Global“, „Movie“, „Player“ usw.).

   JavaScript: Zeigt Funktionen für JavaScript-Skripts, 3D-JavaScript-Skripts und Skript-Xtras an, die im aktuellen Film verwendet werden. Die Funktionen sind in Kategorien aufgeteilt („Global“, „Movie“, „Player“ usw.).
4. Blenden Sie die einzelnen Kategorien ein, um deren dazugehörige Funktionen anzuzeigen, indem Sie auf das Pluszeichen (+) daneben klicken. Um Funktionen in alphabetischer Reihenfolge anzuzeigen, blenden Sie die Indexkategorie ein.
5. Um dem Skript-Editor eine Funktion zum Erstellen von Skripts hinzuzufügen, doppelklicken Sie auf die Funktion.
6. Speichern Sie das Skript.
7. Klicken Sie auf die Schaltfläche „Alle geänderten Skripts rekompilieren“.

Skript-Browser-Ansicht

In der Skript-Browser-Ansicht werden die Skripts und dazugehörigen Prozeduren angezeigt, die im Film verwendet wurden. In dieser Ansicht können Sie neue Skripts und Prozeduren erstellen.

In der Skript-Browser-Ansicht können Sie folgende Aufgaben ausführen:

- Skripts und Prozeduren im aktuellen Film in einer Struktur oder Liste durchsuchen
- Skripts basierend auf dem Skriptnamen, dem Darstellernamen, der Darstellernummer oder dem Skripttyp sortieren
- Eine Prozedur im Skript-Editor bestimmen
- Skripts unter den einzelnen Skripttypen oder -darstellern erstellen

So werden in der Skript-Browser-Ansicht Skripts durchsucht und erstellt

1 Wählen Sie „Fenster“ > „Skript“, um das Skriptfenster zu öffnen.

2 Klicken Sie auf die Registerkarte „Skript-Browser“. Die Skripts werden im Feld unten in einer Baumstruktur angezeigt. Um die Skripts als Liste anzuzeigen, klicken Sie auf die Schaltfläche „Skript-Browser-Ansicht“.

Baumansicht: Skripts werden auf der Basis des Skripttyps kategorisiert (beispielsweise Verhaltensskripte, Filmskripte, Parent-Skripts oder Besetzungsbibliothek), in der Sie erstellt wurden. Auch Darstellerskripts werden hier aufgelistet. Um ein Skript im Skript-Editor zu öffnen, doppelklicken Sie darauf. Zum Skript gehörende Prozeduren werden als Baumstruktur unter dem entsprechenden Skriptknoten angezeigt.

3 Um eine Liste kompilierter Prozeduren im Skript anzuzeigen, erweitern Sie den Knoten „Skriptname“. Nicht kompilierte Prozeduren werden nicht angezeigt.

4 Um eine Prozedur im Skript-Editor zu lokalisieren, doppelklicken Sie darauf. Die Prozedur wird im Skript-Editor hervorgehoben. Sie können auch in der Skriptleiste auf das Symbol „Gehe zu Prozedur“ klicken. Weitere Informationen hierzu finden Sie unter „Suchen von Prozeduren und Text in Skripts“ auf Seite 76.

5 Um einen Kommentar im Skript hinzuzufügen oder zu entfernen, klicken Sie in der Skriptleiste auf das Symbol „Kommentar“ oder „Kommentar entfernen“.

6 Um einen Haltepunkt ein- oder auszuschalten, klicken Sie in der Skriptleiste auf „Haltepunkt ein-/ausschalten“. Sie können auch F9 drücken oder neben dem Code auf die blaue Leiste klicken.

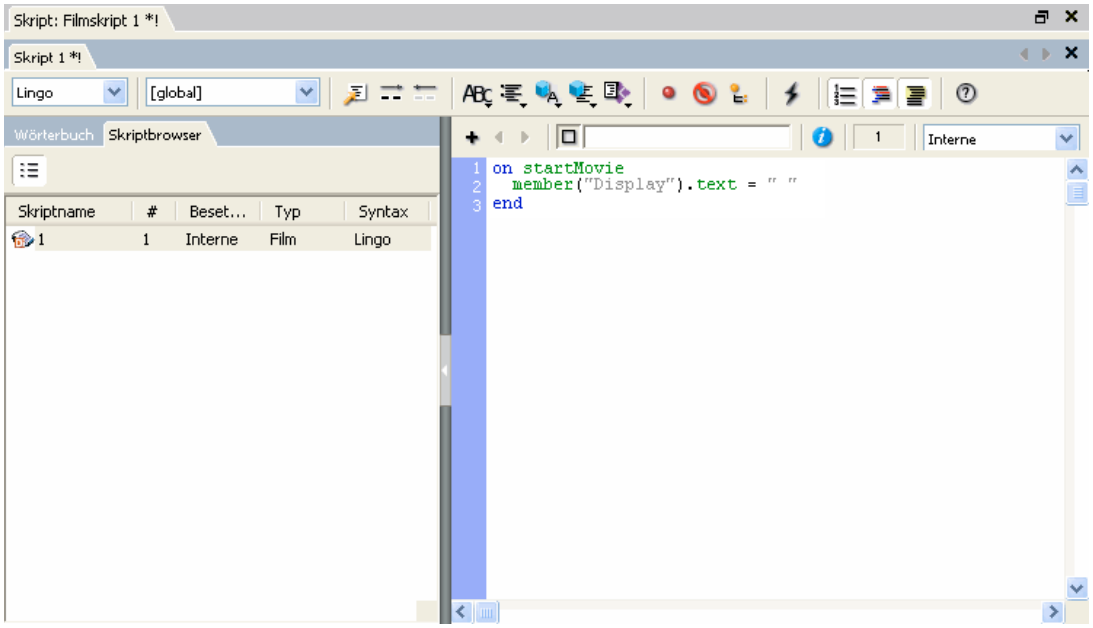
7 **Listenansicht**Skripts sind in einer Liste in einer Spalte basierend auf dem Skriptnamen, dem Darstellernamen, der Darstellernummer und dem Typ organisiert. Um eine Liste in einer Spalte zu sortieren, klicken Sie auf die gewünschte Spaltenüberschrift.

8 Um in der Baumansicht ein neues Skript zu erstellen, klicken Sie mit der rechten Maustaste auf einen Skripttyp, und wählen Sie im Kontextmenü „Neuen Skripttyp hinzufügen“ aus. Wenn Sie im Skript-Editor einen Namen für das Skript eingeben, wird der Name des Skripts in der Liste angezeigt.

9 Speichern Sie das Skript.

10 Klicken Sie auf die Schaltfläche „Alle geänderten Skripts rekompilieren“.

Darstellerskripts werden in der Baumansicht im Ordner „Darstellerskripten“ und in der Listenansicht als Skriptnamen angezeigt. Da diese Skripts einem bestimmten Darsteller und nicht tatsächlichen Darstellern zugeordnet sind, werden sie nur aus dem Skript-Browser entfernt, wenn Sie den dazugehörigen Darsteller löschen.



Informationen zu weiteren Möglichkeiten zum Erstellen und Öffnen von Skripts finden Sie unter „Ausführen allgemeiner Aufgaben“ auf Seite 77.

Hinweis: *Um eine Skriptregisterkarte zu schließen, klicken Sie auf das X auf der Registerkarte oder mit der rechten Maustaste auf den Registerkartenbereich. Wählen Sie dann die Option „<Skripttyp:Skriptname> schließen“.*

Öffnen und Schließen mehrerer Skripts

Im Skriptfenster können mehrere Skripts in Form verschiedener Registerkarten geöffnet werden. Da in das Skriptfenster nur eine feste Anzahl von Registerkarten passt, werden einige der Registerkarten ggf. ausgeblendet. Über die Symbole „“ und „“ können Sie zu den ausgeblendeten Registerkarten wechseln.

So öffnen Sie mehrere Skriptfenster

❖ Führen Sie einen der folgenden Schritte aus:

- Wählen Sie im Skriptfenster „Fenster Neues Skriptfenster“.
- Drücken Sie ALT+W+N.

Über diese Tastenkombination können Sie weitere Instanzen eines beliebigen aktiven Fensters öffnen. Wenn Sie beispielsweise das Fenster „Vektor“ geöffnet haben, können Sie über die Tastenkombination ALT+W+N weitere Fenster dieses Typs öffnen.

So schließen Sie eine Skriptregisterkarte

1 Klicken Sie auf die Registerkarte des Skriptfensters, das Sie schließen möchten.

2 Führen Sie einen der folgenden Schritte aus:

- Klicken Sie auf die zu schließende Registerkarte (falls nicht bereits die aktive Registerkarte) und dann auf die Schaltfläche „X“.
- Klicken Sie mit der rechten Maustaste auf die Registerkarte oder den Bereich neben der Registerkarte, und wählen Sie „<Skripttyp:Skriptname> schließen“.

Skriptfenster-Voreinstellungen festlegen

Im Skriptfenster können Sie die Textschriftart ändern und den verschiedenen Codekomponenten unterschiedliche Farben zuordnen. Im Dialogfeld „Skriptfenster-Voreinstellungen“ können Sie die Standardschriftart für den Text im Skriptfenster und die Farbe der verschiedenen Codeelemente ändern. Sofern Sie die Option Automatische Farbkodierung nicht deaktiviert haben, zeigt Director die unterschiedlichen Codeelemente automatisch in verschiedenen Farben an.

Um den Explorer-Bereich in den anderen Fenstern zu öffnen, klicken Sie auf das Pfeilsymbol auf der Trennleiste zwischen dem Skript-Editor und dem Explorer-Bereich.

So legen Sie Skriptfenster-Voreinstellungen fest

1 Wählen Sie „Bearbeiten“ > „Voreinstellungen“ > „Skript“.

2 Klicken Sie auf die Schaltfläche Schrift, und wählen Sie im Dialogfeld Schrift die gewünschte Standardschriftart aus.

3 Wählen Sie im Menü Farbe die Standardfarbe für den Text im Skriptfenster aus.

4 Wählen Sie im Menü Hintergrund die Hintergrundfarbe für das Skriptfenster aus.

5 Wenn in einem neuen Skriptfenster bestimmte Codeelemente automatisch farbig dargestellt werden sollen, aktivieren Sie das Kontrollkästchen „Automatische Farbkodierung“. Diese Option ist in der Standardeinstellung aktiviert. Wenn diese Option deaktiviert ist, wird der gesamte Text in der Standardfarbe angezeigt.

6 Wenn Ihre Skripts in einem neuen Skriptfenster automatisch mit den richtigen Einrückungen formatiert werden sollen, aktivieren Sie das Kontrollkästchen „Automatische Formatierung“. Diese Option ist in der Standardeinstellung aktiviert.

Hinweis: *Die Funktionen zur automatische Farbkodierung und Formatierung gelten nicht für JavaScript-Code. Wenn Sie Skripts in JavaScript-Syntax erstellen, sind die Schaltflächen „Automatische Farbkodierung“ und „Automatische Formatierung“ im Skriptfenster deaktiviert, und Begriffe wie* `function`*,* `var` *oder* `this` *werden in der Standardtextfarbe angezeigt.*

7 Wenn Ihre Skripts in einem neuen Skriptfenster mit Zeilennummern versehen werden sollen, aktivieren Sie das Kontrollkästchen „Zeilennummerierung“. Diese Option ist in der Standardeinstellung aktiviert.

8 Wenn Sie die Option Auto-Farbe aktiviert haben, stellen Sie in den entsprechenden Farbmenüs die Farben für die folgenden Codeelemente ein:

- Schlüsselwörter
- Kommentare
- Literale
- Benutzerdefiniert (Begriffe, die Sie in Ihrem eigenen Code definiert haben)

9 Klicken Sie auf das Farbmenü „Zeilennummern“, und stellen Sie die gewünschte Hintergrundfarbe für die Zeilennummernspalte ein.

10 Legen Sie im Menü „Debugger-Bereich“ fest, ob die Prozeduraufruflisten-, Variablen- und Watcher-Bereiche links, oben, rechts oder unten im Debugger-Fenster angezeigt werden.

11 Wählen Sie „Lingo“ oder „JavaScript“ im Popupmenü „Standardskripttyp“ aus. Director verwendet die gewählte Option beim Öffnen des Explorer-Bereichs als Standardeinstellung.

Hinweis: *Dies ist eine Änderung auf Anwendungsebene, die nach Schließen und erneutem Öffnen von Director beibehalten wird.*

12 Um den Explorer-Bereich an eine andere Position neben dem Skript-Editor im Skriptfenster zu bewegen, wählen Sie im Popupmenü „Explorer-Bereich“ entweder „Links“, „Oben“ > „Rechts“ oder „Unten“ aus. Der Explorer-Bereich wird standardmäßig links neben dem Skript-Editor angezeigt.

Hinweis: *Beim Debuggen des Skripts wird eine getrennte Instanz des Skript-Editors neben dem Debugger-Bereich angezeigt. Das Dokumentfenster kann in der Anwendung zwischen anderen geöffneten Fenstern neu angeordnet werden.*

Löschen von Skripts

Im Explorer-Bereich können Sie Skripts löschen.

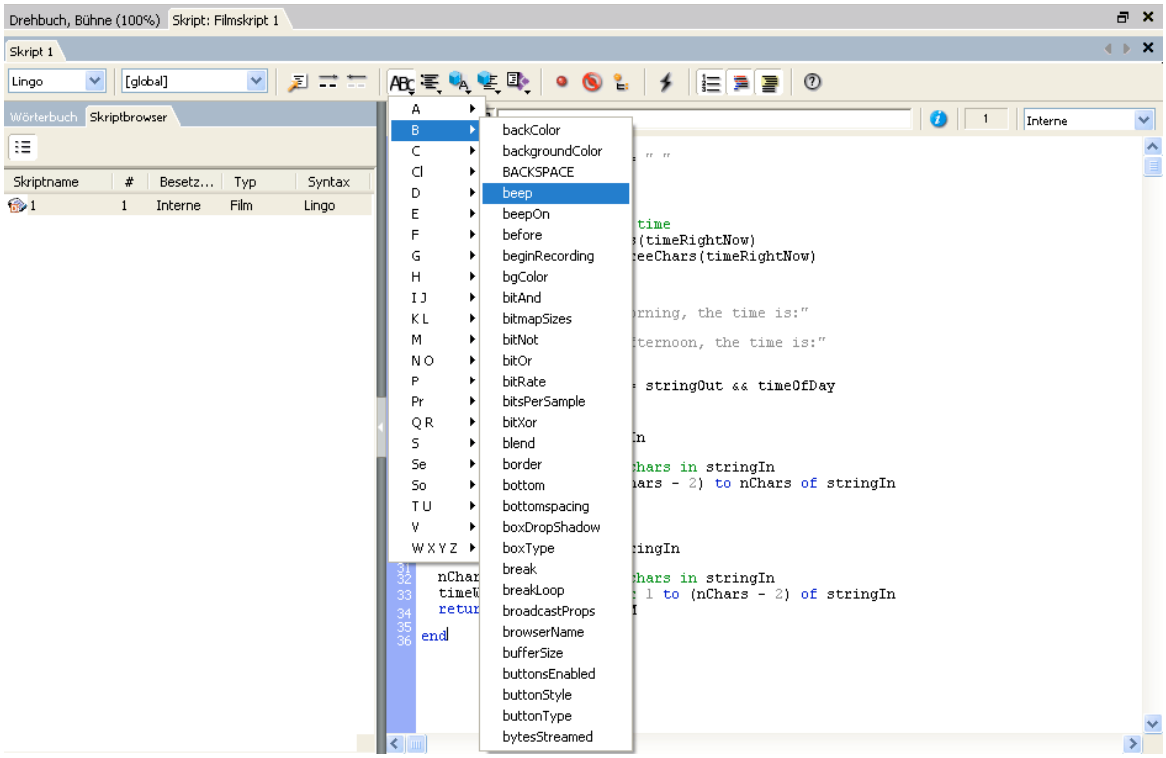
❖ Führen Sie einen der folgenden Schritte aus:

- Klicken Sie im Skript-Browser mit der rechten Maustaste auf einen Skripteintrag, und wählen Sie „Löschen“.
- Wählen Sie in der Listenansicht im Skript-Explorer ein Skript aus, und drücken Sie die ENTF-TASTE.

Hinweis: *Wenn Sie ein geöffnetes Skript löschen, wird die zum Skript gehörende Registerkarte ebenfalls aus dem Skriptfenster entfernt. Wenn das gelöschte geöffnete Skript das einzige Skript in Ihrem Film war, wird das Skriptfenster deaktiviert, und auf der Registerkarte, auf der das Skript enthalten war, wird der Text Skript angezeigt. Um den Texteditor zu aktivieren, klicken Sie im Skript-Editor auf das Pluszeichen (+).*

Einfügen häufig verwendeter Skriptbegriffe

Das Skriptfenster enthält Popupmenüs mit häufig verwendeten Skriptbegriffen, die Sie in ein Skript übernehmen können. Dieselben Menüs werden auch im Nachrichtenfenster angezeigt.



Einfügen häufig verwendeter Skriptbegriffe

Sowohl im Skriptfenster als auch im Nachrichtenfenster können Sie die Skriptsyntax für ein bestimmtes Skript auswählen.

So wählen Sie die Skriptsyntax aus

❖ Wählen Sie im Popupmenü „Skriptsyntax“ entweder „Lingo“ oder „JavaScript“ aus.

Geben Sie nach Auswahl der Skriptsyntax Code in der gewählten Syntax ein. Wenn Sie versuchen, ein Skript in einer anderen Syntax als der gewählten zu kompilieren, tritt ein Skriptfehler auf.

Beim Eingeben von Skripts in das Skriptfenster können Sie über die Schaltflächen „Kommentar“ oder „Kommentar entfernen“ eine oder mehrere Codezeilen mit Kommentar versehen oder Kommentar entfernen. Je nach gewählter Skriptsyntax zeigen diese Schaltfläche die ordnungsgemäßen Kommentarmarkierungen für die jeweilige Sprache an. Lingo verwendet doppelte Bindestriche (`--`), JavaScript doppelte Schrägstriche (`//`).

So kommentieren Sie Code

❖ Markieren Sie die Codezeilen, die Sie kommentieren möchten, und klicken Sie auf „Kommentar“.

Hinweis: *Bei Verwenden der Schaltfläche „Kommentar“ zum Kommentieren mehrerer JavaScript-Codezeilen fügt Director vor jeder Zeile doppelte Schrägstriche ein. Sie können mehrere Codezeilen auch kommentieren, indem Sie „/*“ vor die erste und „*/“ hinter die letzte kommentierte Codezeile setzen, was allerdings manuell erfolgen muss.*

So entfernen Sie Codekommentare

❖ Markieren Sie die Codezeilen, deren Kommentar Sie entfernen möchten, und klicken Sie auf „Kommentar entfernen“.

Sowohl das Skript- als auch das Nachrichtenfenster enthalten die folgenden Menüs:

- Im Menü Lingo in alphabetischer Reihenfolge sind alle Elemente, mit Ausnahme der 3D-Lingo-Elemente, in alphabetischer Reihenfolge aufgeführt.
- Im Menü Lingo nach Kategorie sind die Lingo-Elemente nach Funktionskategorien geordnet aufgeführt. Dieses Menü enthält keine 3D-Lingo-Elemente.
- Im Menü 3D-Lingo alphabetisch geordnet sind alle 3D-Lingo-Elemente in alphabetischer Reihenfolge aufgeführt.
- Im Menü 3D-Lingo nach Kategorie sind alle 3D-Lingo-Elemente nach Funktionskategorien geordnet aufgeführt.
- Das Popupmenü „Skript-Xtras“ enthält die Methoden und Eigenschaften aller gefundenen Xtra-Skripterweiterungen von Adobe und anderen Anbietern.

Hinweis: *Die im Popupmenü „Skript-Xtras“ aufgeführten Xtra-Skripterweiterungen sind ausschließlich diejenigen, welche die* `Interface()`*-Methode unterstützen und deren Namen tatsächlich im Popupmenü angezeigt werden. Obgleich einige Darstellermedientypen wie 3D und DVD auch die* `Interface()`*-Methode unterstützen, werden sie nicht im Popupmenü „Skript-Xtras“ angezeigt, da sie nicht in Director als Xtras-Skripterweiterungen implementiert sind.*

Elemente, die Sie in einem der Popupmenüs auswählen, werden jeweils an der Position der Einfügemarke in das Skriptfenster übernommen.

Wenn ein Element zusätzliche Parameter benötigt, werden Platzhalternamen eingefügt, die auf die noch fehlenden Informationen verweisen. Sind noch mehrere Argumente oder Parameter erforderlich, wird das erste Argument bzw. der erste Parameter hervorgehoben, sodass Sie den Eintrag lediglich zu überschreiben brauchen. Die übrigen Parameter müssen Sie dann selbst auswählen und ändern.

Einige Darstellertypen und Xtra-Skripterweiterungen bieten zusätzliche Begriffe, die in den Popupmenüs nicht enthalten sind. Diese Darstellertypen und Xtra-Erweiterungen werden oft mit eigener Dokumentation geliefert, und möglicherweise finden Sie auch in Director einige hilfreiche Informationen.

So zeigen Sie eine Liste der verfügbaren Xtra-Erweiterungen an

❖ Rufen Sie im Nachrichtenfenster entweder `put(_player.xtraList)` oder `trace(_player.xtraList)` auf.

So zeigen Sie eine Liste der verfügbaren Xtra-Skripterweiterungen an

❖ Rufen Sie im Nachrichtenfenster entweder `put(_player.scriptingXtraList)` oder `trace(_player.scriptingXtraList)` auf.

So zeigen Sie eine Liste der Methoden für eine Xtra-Erweiterung an

❖ Zeigen Sie im Popupmenü „Skript-Xtras“ auf eine Xtra-Erweiterung, und klicken Sie im Untermenü auf „Put Interface“. Die Methoden und Eigenschaften für diese Xtra-Erweiterung werden im Nachrichtenfenster angezeigt.

Eingeben und Bearbeiten von Text

In einem Skriptfenster lässt sich Text in ähnlicher Weise eingeben und bearbeiten wie in jedem anderen Feld.

Die folgende Liste bietet einen Überblick über allgemeine Bearbeitungsaufgaben, die Sie im Skriptfenster ausführen können:

- Zum Auswählen eines Wortes doppelklicken Sie auf das gewünschte Wort.
- Zum Auswählen eines gesamten Skripts wählen Sie im Menü „Bearbeiten“ den Befehl „Alles auswählen“.
- Zum Beginnen einer neuen Zeile geben Sie eine Zeilenschaltung ein.

- Zum Umbrechen einer langen Codezeile mit einem Fortsetzungszeichen bewegen Sie in Lingo die Einfügemarke an die gewünschte Stelle und drücken ALT+EINGABETASTE (Windows®) bzw. WAHL-+EINGABETASTE (Mac®). Das an dieser Stelle eingefügte Fortsetzungszeichen (\\) gibt an, dass die Anweisung in der nächsten Zeile fortgesetzt wird.

  Um in JavaScript eine lange Codezeile umzubrechen, fügen Sie durch Drücken der EINGABETASTE einen normalen Zeilenumbruch ein. Das Lingo-Fortsetzungszeichen verursacht in JavaScript-Syntax einen Skriptfehler.

- Zum Auffinden einer Prozedur im aktuellen Skript wählen Sie im Popupmenü „Gehe zu Prozedur“ den gewünschten Prozedurnamen aus.
- Zum Kompilieren geänderter Skripts klicken Sie im Skriptfenster auf die Schaltfläche „Alle geänderten Skripts rekompilieren“, oder schließen Sie das Skriptfenster. Beim Bearbeiten eines Skripts wird in der Titelleiste des Skriptfensters ein Sternchen als Hinweis darauf angezeigt, dass das Skript rekompiliert werden muss.
- Um alle Skripts in einem Film zu rekompilieren, wählen Sie im Menü „Steuerung“ den Befehl „Alle Skripts rekompilieren“ aus.
- Zum Neuformatieren eines Skripts mit den gewünschten Einrückungen drücken Sie im Skriptfenster die TAB-TASTE.

  Sofern die Syntax ordnungsgemäß ist, rückt Director die Anweisungen automatisch ein. Wenn eine Zeile nicht richtig eingerückt wird, enthält dieser Zeile einen Syntaxfehler.

- Zum Öffnen eines zweiten Skriptfensters klicken Sie bei gedrückter ALT-TASTE (Windows) bzw. WAHLTASTE (Macintosh) im Skriptfenster auf die Schaltfläche „Neuer Darsteller“. Auf diese Weise können Sie zwei verschiedene Abschnitte eines längeren Skripts mühelos gleichzeitig bearbeiten.
- Zum Ein- und Ausschalten der automatischen Zeilennummerierungen klicken Sie auf die Schaltfläche „Zeilennummerierungen“.
- Zum Ein- und Ausschalten der automatischen Farbkodierung klicken Sie auf die Schaltfläche Automatische Farbkodierung. Die automatische Farbkodierung bewirkt, dass die verschiedenen Arten von Lingo-Elementen (Eigenschaften, Befehle usw.) in unterschiedlichen Farben dargestellt werden.
- Zum Ein- und Ausschalten der automatischen Formatierung klicken Sie auf die Schaltfläche Automatische Formatierung. Die automatische Formatierung bewirkt, dass eine Skriptzeile, die Sie mit einer Zeilenschaltung oder der Tab-Taste beenden, automatisch korrekt eingerückt wird.

  Hinweis: *Die Funktionen zur automatische Farbkodierung und Formatierung gelten nicht für JavaScript-Code. Wenn Sie Skripts in JavaScript-Syntax erstellen, sind die Schaltflächen „Automatische Farbkodierung“ und „Automatische Formatierung“ im Skriptfenster deaktiviert, und Begriffe wie* `function`*,* `var` *und* `this`*werden in der Standardtextfarbe angezeigt.*

Suchen von Prozeduren und Text in Skripts

Mit dem Befehl Suchen im Menü Bearbeiten können Sie Prozeduren und Textabschnitte auffinden, die Sie bearbeiten möchten.

So suchen Sie in Skripts nach Prozeduren

1 Wählen Sie „Bearbeiten“ > „Suchen“ > „Prozedur“.

Das Dialogfeld Prozedur suchen wird angezeigt.

In der linken Spalte des Dialogfelds Prozedur suchen sind die Namen der im Film enthaltenen Prozeduren aufgeführt. In der mittleren Spalte erscheint die Nummer des Darstellers, der dem Skript der jeweiligen Prozedur zugeordnet ist, sowie der Name des Darstellers. In der rechten Spalte ist die Besetzung angegeben, aus der der betreffende Darsteller stammt.

2 Wählen Sie den gesuchten Darsteller aus.

3 Klicken Sie auf Suchen.

Die Prozedur erscheint im Skriptfenster.

In der Titelleiste des Skriptfensters wird der Skripttyp angezeigt.

So suchen Sie in Skripts nach Text

1 Aktivieren Sie das Skriptfenster.

2 Wählen Sie „Bearbeiten“ > „Suchen“ > „Text“.

Das Dialogfeld Text suchen wird angezeigt.

3 Geben Sie den gesuchten Text in das Feld „Suchen“ ein, und klicken Sie auf die Schaltfläche Suchen.

Standardmäßig unterscheidet „Suchen“ nicht zwischen Groß- und Kleinschreibung: `ThisHandler`, `thisHandler` und `THISHANDLER` sind für den Suchprozess das Gleiche. Aktivieren Sie das Kontrollkästchen „Groß-/Kleinschreibung“, um Groß-/Kleinschreibung bei der Suche zu unterscheiden.

So geben Sie an, welche Darsteller durchsucht werden sollen

❖ Wählen Sie unter „Suchen: Skripts“ die gewünschte Option.

So setzen Sie die Suche am Anfang des Skripts fort, wenn das Skriptende erreicht ist

❖ Aktivieren Sie die Option Suchrundlauf.

So beschränken Sie die Suche auf ganze Wörter und ignorieren übereinstimmende Wortfragmente

❖ Aktivieren Sie die Option Nur ganze Wörter.

So gelangen Sie zur nächsten Fundstelle des im Feld „Suchen“ angegebenen Textes

❖ Wählen Sie „Bearbeiten“ > „Erneut suchen“.

So gelangen Sie zu weiteren Fundstellen eines ausgewählten Textes

1 Markieren Sie den gewünschten Text.

2 Wählen Sie „Bearbeiten“ > „Suchen“ > „Auswahl“.

Ausführen allgemeiner Aufgaben

Im Folgenden werden allgemeine Methoden zum Erstellen, Zuordnen und Öffnen von Skripts beschrieben.

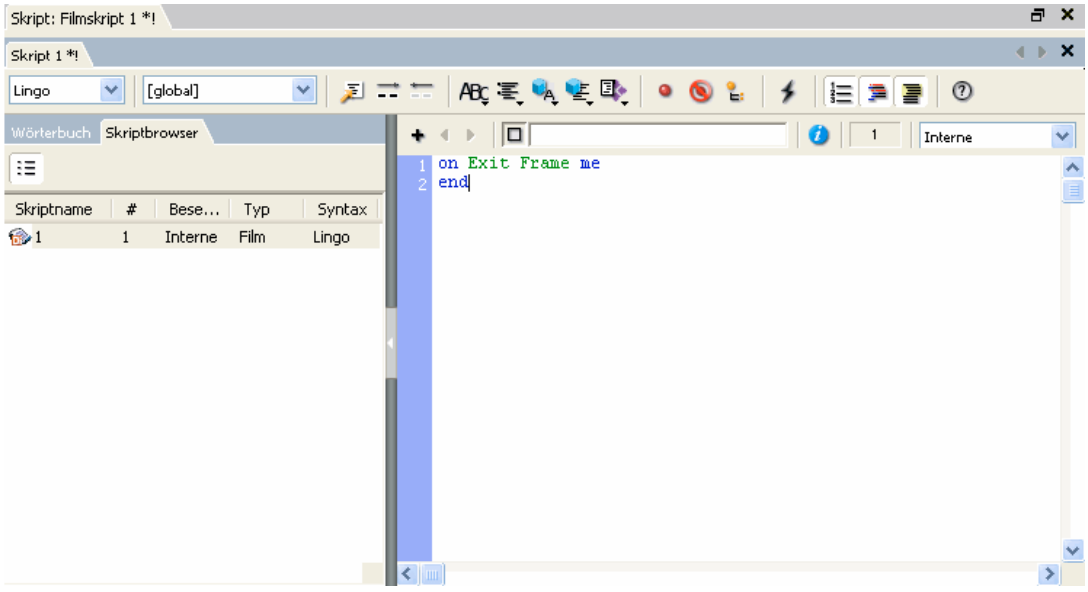
So erstellen Sie ein Bildverhalten (ein einem Bild zugeordnetes Skript)

❖ Doppelklicken Sie in dem Bild, dem das Verhalten zugeordnet werden soll, auf den Verhaltenskanal.

Verhaltenskanäle

Wenn Sie ein neues Verhalten erstellen, wird diesem die Darstellerbibliotheksnummer der ersten verfügbaren Position im aktuellen Besetzungsfenster zugeordnet.

Wenn Sie ein neues Bildverhalten erstellen, wird das Skriptfenster geöffnet, das bereits die Lingo-Prozedur `on exitFrame` enthält. Die erste Zeile enthält `on exitFrame`, gefolgt von einer Zeile mit einer blinkenden Einfügemarke und einer Zeile mit dem Wort `end`. Auf diese Weise können Sie schnell ein allgemeines Lingo-Verhalten dem Bild zuordnen. Damit diese Prozedur mit JavaScript-Syntax funktioniert, ersetzen Sie `on exitFrame` durch `function exitFrame()` { und `end` durch }.



Eines der am häufigsten verwendeten Bildverhalten sorgt dafür, dass der Abspielkopf ein einzelnes Bild in Schleife wiedergibt. Dies ist beispielsweise dann sinnvoll, wenn der Film in einem Bild abwarten soll, bis der Benutzer auf eine Schaltfläche klickt oder ein Digitalvideo oder ein Sound abgelaufen ist.

So halten Sie den Abspielkopf in einem Einzelbild

❖ Fügen Sie die folgende Anweisung in die auf die Anweisung `on exitFrame` (Lingo) oder `function exitFrame()` (JavaScript) folgende Zeile eines Bildverhaltens ein:

```
-- Lingo syntax
_movie.go(_movie.frame)

// JavaScript syntax
_movie.go(_movie.frame);
```

Die `frame`-Eigenschaft des Movie-Objekts verweist auf das aktuelle Bild am Abspielkopf. Diese Anweisung bedeutet dem Abspielkopf praktisch, dass er zum aktuellen Bild zurückkehren soll.

So erstellen Sie ein Sprite-Verhalten (ein einem Sprite zugeordnetes Skript)

❖ Wählen Sie das Sprite, dem das Verhalten zugeordnet werden soll, im Drehbuch oder auf der Bühne aus. Wählen Sie anschließend „Fenster“ > „Verhaltensinspektor“ und im Popupmenü Verhalten den Befehl „Neues Verhalten“.

Wenn Sie ein neues Sprite-Verhalten erstellen, wird das Skriptfenster geöffnet, das bereits die Lingo-Prozedur `on mouseUp` enthält. Die erste Zeile enthält `on mouseUp`, gefolgt von einer Zeile mit einer blinkenden Einfügemarke und einer Zeile mit dem Wort `end`. Auf diese Weise können Sie schnell ein allgemeines Verhalten dem Sprite zuordnen. Damit diese Prozedur mit JavaScript-Syntax funktioniert, ersetzen Sie `on mouseUp` durch `function mouseUp() {` und `end` durch `}`.

So öffnen Sie ein Verhalten zur Bearbeitung

1 Doppelklicken Sie im Besetzungsfenster auf das Verhalten.

Der Verhaltensinspektor wird geöffnet.

2 Klicken Sie auf das Symbol Skriptfenster im Verhaltensinspektor.

Im Skriptfenster wird das Verhalten angezeigt.

Eine andere Möglichkeit besteht darin, das Skriptfenster zu öffnen und die Skripts der Reihe nach aufzurufen, bis Sie das gewünschte Verhalten erreicht haben.

So entfernen Sie ein Verhalten von einer Position im Drehbuch

❖ Wählen Sie die gewünschte Position aus, und löschen Sie das Skript in der Liste im Eigenschafteninspektor (Registerkarte Verhalten).

So ordnen Sie Sprites oder Bildern vorhandene Verhalten zu

❖ Führen Sie einen der folgenden Schritte aus:

- Ziehen Sie das gewünschte Verhalten aus einer Besetzung auf ein Sprite oder Bild im Drehbuch bzw. auf ein Sprite auf der Bühne.
- Wählen Sie im Drehbuch das Sprite oder Bild aus, dem das Verhalten zugeordnet werden soll. Wählen Sie anschließend „Fenster“ > „Verhaltensinspektor“, und geben Sie im Popupmenü „Verhalten“ das gewünschte Verhalten an.

So erstellen Sie ein Filmskript (ein einem Film zugeordnetes Skript)

Führen Sie einen der folgenden Schritte aus:

- So erstellen Sie ein Filmskript in der Baumansicht des Skript-Explorers
  1 Wählen Sie unter der Besetzung, der Sie ein Filmskript hinzufügen möchten, einen Filmknoten aus.
  2 Klicken Sie mit der rechten Maustaste, und wählen Sie „Neues Filmskript hinzufügen“.

- Wenn es sich bei dem aktuellen Skript im Skriptfenster um ein Filmskript handelt, klicken Sie im Skriptfenster auf die Schaltfläche Neues Skript. (Durch Anklicken der Schaltfläche Neues Skript wird immer ein Skript des Typs erstellt, der als aktuelles Skript im Fenster angezeigt wird.)
- Wenn es sich beim aktuellen Skript im Skriptfenster nicht um ein Filmskript handelt, klicken Sie auf die Schaltfläche „Neues Skript“, und ändern Sie dann den Typ im Popupmenü „Skripttyp“ auf der Registerkarte „Skript“ des Eigenschafteninspektors.
- Wenn in der Besetzung, im Drehbuch oder auf der Bühne keine Sprites oder Skripts ausgewählt sind, öffnen Sie ein neues Skriptfenster. Dadurch wird standardmäßig ein neues Filmskript erstellt.

So öffnen Sie ein Film- oder Parent-Skript zur Bearbeitung

❖ Doppelklicken Sie nach Öffnen des Skripts im Skript-Explorer im Besetzungsfenster auf das Skript.

So ändern Sie den Skripttyp

1 Wählen Sie das Skript im Besetzungsfenster aus, oder öffnen Sie es im Skriptfenster.

2 Klicken Sie auf die Registerkarte „Skript“ des Eigenschafteninspektors, und wählen Sie im Popupmenü „Typ“ einen anderen Skripttyp aus.

So zeigen Sie die Skripts im Skriptfenster der Reihe nach an

❖ Klicken Sie auf die Pfeile Vorheriger Darsteller und Nächster Darsteller oben im Skriptfenster, um zum jeweils folgenden oder vorhergehenden Darsteller zu gelangen.

So duplizieren Sie ein Skript

❖ Wählen Sie das Skript im Besetzungsfenster aus, und klicken Sie im Menü Bearbeiten auf Duplizieren.

Wenn Sie ein Skript erstellen möchten, das automatisch allen auf einem bestimmten Darsteller basierenden Sprites zugeordnet wird, ordnen Sie das Skript dem Darsteller selbst zu.

So erstellen Sie ein einem Darsteller zugeordnetes Skript oder öffnen ein vorhandenes Skript

❖ Führen Sie einen der folgenden Schritte aus:

- Klicken Sie mit der rechten Maustaste (Windows) oder bei gedrückter CTRL-TASTE (Macintosh) auf einen Darsteller im Besetzungsfenster, und wählen Sie im Kontextmenü die Option „Darstellerskript“.
- Wählen Sie im Besetzungsfenster einen Darsteller aus, und klicken Sie auf die Schaltfläche Darstellerskript.

Verwenden verknüpfter Skripts

Skripts lassen sich nicht nur als interne Darsteller, sondern auch in externen Textdateien speichern, die mit dem Director-Film verknüpft sind. Diese verknüpften Skripts funktionieren ähnlich wie Grafik- oder Digitalvideodateien, die Sie als verknüpfte Darsteller in Director-Filme importieren.

Verknüpfte Skripts bieten folgende Vorteile:

- Die Director-Datei und das Skript können von verschiedenen Benutzern gleichzeitig bearbeitet werden.
- Skripts können mühelos mit anderen Benutzern ausgetauscht werden.
- Sie können Skripts getrennt von der Director-Datei in einer Quellcodesteuerungsanwendung wie Microsoft® Visual SourceSafe® oder Perforce® von Perforce Software steuern. Anwendungen dieser Art verhindern, dass Programmierer, die an einem gemeinsamen Director-Projekt arbeiten, ihre Arbeit gegenseitig überschreiben.

Director verwendet verknüpfte Skripts nur während der Anwendungserstellung. Zur Laufzeit benutzen Director-Projektoren und Adobe® Shockwave® eine spezielle interne Kopie der Skriptdaten, die im Film gespeichert ist. Auf diese Weise müssen Ihre verknüpften Skripts nicht mit Ihren Filmen verteilt und können von den Endbenutzern nicht kopiert werden.

So importieren Sie ein Skript als verknüpfte Textdatei

1. Wählen Sie „Datei" > „Importieren".
2. Wählen Sie im Popupmenü Dateityp den Eintrag Skript.
3. Wählen Sie die Skriptdateien aus, die Sie importieren möchten.

   Sie können Dateien mit der Erweiterung .txt, .ls oder .js importieren. Die Erweiterung .ls wird von Director für verknüpfte Skripts verwendet.

   Mithilfe der Schaltflächen Hinzufügen und Alle hinzufügen können Sie eine Liste von Dateien anlegen, die Sie importieren möchten. Dies ist besonders dann hilfreich, wenn Sie Skripts aus verschiedenen Speicherorten importieren möchten.
4. Wählen Sie im Popupmenü Media die Option Mit externer Datei verknüpfen.
5. Klicken Sie auf Importieren.

Sie können verknüpfte Skripts auf übliche Weise im Director-Skriptfenster bearbeiten. Hierbei werden Ihre Änderungen beim Speichern des Director-Films automatisch in die externen Dateien übernommen. (Wenn Sie das verknüpfte Skript von einem UNIX®-Server importiert haben, werden die UNIX-Zeilenendungen beibehalten.) Ein importiertes Skript, dessen Textdatei gesperrt ist, lässt sich in Director nicht bearbeiten.

Benutzerdefinierte Textfarben können auf verknüpfte Skripts im Skriptfenster nicht angewendet werden. Die automatische Farbkodierung hingegen funktioniert auch bei verknüpften Skripts.

So wandeln Sie einen internen Skriptdarsteller in einen externen, verknüpften Skriptdarsteller um

1. Wählen Sie den internen Darsteller aus, und wechseln Sie im Eigenschafteninspektor zur Registerkarte Skript.
2. Klicken Sie auf die Schaltfläche Verknüpfen.
3. Geben Sie im Dialogfeld Skriptdatei speichern unter einen Namen für die Skriptdatei an.
4. Klicken Sie auf Speichern.

So laden Sie ein verknüpftes Skript nach der Bearbeitung neu

- Verwenden Sie die `unload()`-Methode des Member-Objekts.

Wenn Sie ein verknüpftes Skript außerhalb von Director bearbeiten, können Sie es mithilfe des Befehls `unload()` im Nachrichtenfenster neu laden. Die folgende Anweisung bewirkt, dass das Skript `myScript` entladen und anschließend neu geladen wird:

```
-- Lingo syntax
member("myScript").unload()

// JavaScript syntax
member("myScript").unload();
```

Kapitel 4: Debuggen von Skripts in Director

Information zur Fehlebehebung in Skripts

Skripts funktionieren nicht immer auf Anhieb. Buchstabendreher oder Auslassungen führen oft zu Syntaxfehlern im Skript. Es kann auch vorkommen, dass ein Skript zwar funktioniert, aber nicht das erwartete Ergebnis liefert. Beim Schreiben von Skripts schleichen sich fast immer Fehler oder Unstimmigkeiten ein. Daher sollten Sie stets genügend Zeit für die Fehlerbehebung einplanen, wenn Sie Multimediainhalte entwickeln.

Während Sie im Laufe der Zeit mehr und mehr Erfahrung mit der Erstellung von Skripts sammeln, werden Sie es wahrscheinlich immer wieder mit neuen Problemen zu tun haben, je nachdem, mit welchem Bereich der Anwendungsentwicklung Sie sich gerade beschäftigen. Die in diesem Abschnitt beschriebenen grundlegenden Verfahren zur Fehlerbehebung können jedoch von Einsteigern und fortgeschrittenen Benutzern gleichermaßen befolgt werden.

Auf welche Weise sich ein Skriptfehler am besten beheben lässt, hängt von der jeweiligen Situation ab. Universelle Standardverfahren zur Lösung aller möglichen Probleme gibt es nicht. Vielmehr müssen Sie sich verschiedener Werkzeuge und Ansätze bedienen, die vor allem zwei Dinge voraussetzen:

- Verstehen der Wechselwirkungen und Interaktionen zwischen den einzelnen Skripts im Film
- Vertrautheit und Erfahrung mit allgemeinen Debugging-Methoden

Die folgenden Werkzeuge helfen Ihnen, Probleme in Skripts zu erkennen:

- Im **Nachrichtenfenster** wird bei aktivierter Verfolgungsfunktion ein Protokoll der im Film abgespielten Bilder und ausgeführten Prozeduren angezeigt.
- Im **Debugger-Fenster** werden die Werte globaler Variablen, Eigenschaften des gerade ausgeführten Skripts, die Abfolge der bis zum aktuellen Zeitpunkt ausgeführten Prozeduren sowie die von Ihnen ausgewählten Variablen- und Ausdruckswerte angezeigt.
- Im **Skriptfenster** können Sie Kommentare eingeben, Haltepunkte in das Skript einfügen und Variablen auswählen, deren Werte im Objektinspektor angezeigt werden.
- Im **Objektinspektor** können Sie die Werte ausgewählter Objekte und Eigenschaften anzeigen und festlegen.

Bewährte Verfahren bei der Skripterstellung

Mithilfe bewährter Skripterstellungsverfahren lassen sich viele Skriptprobleme von vornherein vermeiden.

- Schreiben Sie Skripts in kleineren Anweisungsgruppen, die Sie unmittelbar testen, nachdem Sie sie erstellt haben. Auf diese Weise können Sie auftretende Probleme in einem übersichtlicheren Kontext eingrenzen.
- Fügen Sie Kommentare ein, die den Zweck der Skriptanweisungen und Werte in Ihrem Skript erläutern. Diese Kommentare helfen Ihnen und anderen, sich zu einem späteren Zeitpunkt wieder im Skript zurechtzufinden. Die Kommentare in den folgenden Anweisungen erläutern beispielsweise den Zweck der `if...then`-Anweisung und der Wiederholungsschleife:

```
-- Lingo syntax
-- Loop until the "s" key is pressed
repeat while not(_key.keyPressed("s"))
    _sound.beep()
end repeat

// JavaScript syntax
// Loop until the "s" key is pressed
while(!_key.keyPressed("s")) {
    _sound.beep();
}
```

- Achten Sie auf eine ordnungsgemäße Skriptsyntax. Nutzen Sie die vorformatierten Skriptelemente in den Popupmenüs des Skriptfensters. Konsultieren Sie die API-Themen in dieser Referenz, um zu prüfen, ob Ihre Anweisungen ordnungsgemäß sind.
- Wählen Sie aussagekräftige Variablennamen, die den Zweck der Variablen beschreiben. Zu einer Variablen, die eine Zahl enthält, passt ein Name wie `newNumber` besser als eine Buchstabenfolge wie `ABC`.

Debugging-Grundlagen

Das Debuggen von Anwendungen ist kein schematischer Standardvorgang, sondern erfordert strategisches und analytisches Denken. Die in diesem Abschnitt beschriebenen grundlegenden Debugging-Ansätze sind nicht nur auf Lingo- oder JavaScript-Code, sondern auf alle Arten von Anwendungscode anwendbar.

Bevor Sie tief greifende Änderungen an einem Film vornehmen, sollten Sie stets eine Sicherungskopie anlegen. Es ist meist hilfreich, die Kopien numerisch aufsteigend zu benennen, z. B. „Dateiname_01.dir“, „Dateiname_02.dir“, „Dateiname_03.dir“ usw., um die verschiedenen Entwicklungsphasen eines Films unterscheiden zu können.

Ermitteln des Problems

So banal es klingt, besteht der erste Schritt beim Debuggen von Anwendungen darin, das Problem zu ermitteln. Führt eine Schaltfläche die verkehrte Funktion aus? Springt der Film zum falschen Bild? Lässt sich ein Feld nicht bearbeiten, das bearbeitbar sein sollte?

Sie möchten ggf. auch bestimmen, welche Aufgabe ein Skript erledigen soll, und anschließend Ihre Erwartung mit dem vergleichen, was das Skript tatsächlich leistet. Auf diese Weise können Sie das Ziel klar festlegen und feststellen, welche Zielvorgaben nicht erfüllt werden.

Wenn Sie ein Skript oder einen Teil eines Skripts aus einem anderen Film oder einem Codebeispiel übernommen haben, überprüfen Sie, ob das Skript für besondere Bedingungen geschrieben wurde. Möglicherweise setzt es voraus, dass ein Sprite-Kanal bereits mit einem Skript versehen ist. Ggf. müssen die Darstellernamen bestimmten Stilvorgaben folgen.

Lokalisieren des Problems

Befolgen Sie diese Ansätze zum Lokalisieren eines Problems:

- Verfolgen Sie die Anweisungen Schritt für Schritt zurück, und überlegen Sie, wo sich das unerwartete Verhalten zum ersten Mal bemerkbar machte.
- Überprüfen Sie im Nachrichtenfenster, welche Bilder der Film durchläuft und welche Prozeduren Ihre Skripts ausführen.

- Stellen Sie fest, was die Skripts bewirken sollen, und überlegen Sie, welche Elemente dieser Anweisungen mit dem Problem zusammenhängen könnten. Wenn sich beispielsweise ein Textdarsteller, der bearbeitbar sein sollte, nicht bearbeiten lässt, finden Sie heraus, an welchen Stellen im Film das Skript die Eigenschaft `editable` des Darstellers einstellt (bzw. nicht einstellt).
- Wenn ein Sprite sich auf der Bühne nicht wie erwartet ändert, ist möglicherweise irgendwo die Methode `updateStage()` erforderlich.
- Macht sich das Problem nur bei bestimmten Computern bemerkbar, während die Anwendung auf anderen Computern fehlerfrei funktioniert? Tritt es nur auf, wenn der Bildschirm auf Millionen von Farben eingestellt ist? Möglicherweise verursachen bestimmte Einstellungen des Computers einen Konflikt mit der Anwendung.

Sie können bestimmte Skriptzeilen genauer untersuchen, indem Sie Haltepunkte in eine Zeile einfügen, an denen die Skriptausführung unterbrochen und das Debugger-Fenster aufgerufen wird. Auf diese Weise haben Sie die Möglichkeit, die Bedingungen an diesem Punkt zu analysieren, bevor das Skript weiter verarbeitet wird. Weitere Informationen über das Einfügen von Haltepunkten finden Sie unter „Debuggen im Debugger-Fenster“ auf Seite 93.

Beheben einfacher Probleme

Wenn Sie einen Fehler feststellen, sollten Sie zunächst einmal die einfachen Fehlerquellen überprüfen.

Der erste Debugging-Test erfolgt, wenn Sie das Skript kompilieren. Sie können Ihr Skript wie folgt kompilieren:

- Klicken Sie im Skriptfenster auf „Alle geänderten Skripts rekompilieren“.
- Klicken Sie im Menü „Steuerung“ auf „Alle Skripts rekompilieren“.
- Drücken Sie UMSCHALTASTE+F8.
- Schließen Sie das Skriptfenster.

In der Regel sollten Skripts über eine der ersten drei Optionen kompiliert werden. Die vierte Option erfordert, dass Sie das Skriptfenster jedes Mal schließen müssen, wenn Sie ein Skript kompilieren möchten.

Wenn Sie ein Skript kompilieren, gibt Director® eine Fehlermeldung aus, sollte das Skript nicht ordnungsgemäße Syntax enthalten. In dieser Meldung ist in der Regel die Zeile angegeben, in der das Problem erstmals erkannt wurde. Die fehlerhafte Stelle in der Zeile wird mit einem Fragezeichen gekennzeichnet.

Die erste Zeile in der vorherigen Fehlermeldung informiert Sie, dass der betreffende Skriptfehler ein Syntaxfehler ist, der anschließend näher beschrieben wird. Die zweite Zeile in der Fehlermeldung enthält die tatsächliche Codezeile mit dem Syntaxfehler.

Suchen nach Syntaxfehlern

Syntaxfehler sind meist die häufigste Ursache von Problemen bei der Skripterstellung. Wenn ein Skript nicht ordnungsgemäß funktioniert, sollten Sie zunächst folgende Punkte überprüfen:

- Haben Sie Begriffe richtig geschrieben, Leerzeichen richtig gesetzt und notwendige Satzzeichen eingefügt? Director kann Anweisungen mit fehlerhafter Syntax nicht interpretieren.
- Stehen die Namen der Darsteller, Beschriftungen und Strings in der Anweisung in Anführungszeichen?
- Wurden alle erforderlichen Parameter angegeben? Welche Parameter im Einzelnen benötigt werden, hängt vom jeweiligen Element ab. Informationen zu den von den verschiedenen Elementen benötigten Parametern finden Sie in den API-Themen in dieser Referenz.

Suchen nach weiteren einfachen Fehlern

Auch wenn das Skript ohne Fehlermeldung kompiliert wird, enthält es möglicherweise einen Fehler. Wenn Ihr Skript nicht tut, was Sie erwarten, überprüfen Sie die folgenden Punkte:

- Sind die Werte der Parameter ordnungsgemäß? Wenn Sie beispielsweise einen falschen Wert für die Anzahl der Warntöne eingegeben haben, die die Methode `beep()` erzeugen soll, sind während der Wiedergabe zu wenig oder zu viel Töne zu hören.
- Weisen veränderliche Objekte wie Variablen oder Inhalte der Textdarsteller die gewünschten Werte auf? Sie können die Werte im Objektinspektor anzeigen, indem Sie den Namen des gewünschten Objekts auswählen und im Skriptfenster auf „Objekt inspizieren“ klicken oder über die Funktion`put()` oder `trace()` im Nachrichtenfenster ausgeben.
- Tun die Skriptelemente das, was sie tun sollen? Sie können ihr Verhalten anhand der API-Themen in dieser Referenz überprüfen.
- Ist die Schreibung ordnungsgemäß (bei JavaScript-Skripts)? JavaScript unterscheidet Groß-/Kleinschreibung, was bedeutet, dass alle Methoden, Funktionen, Eigenschaften und Variablen in der ordnungsgemäßen Schreibung angegeben werden müssen.

  Wenn Sie eine Methode oder Funktion mit falscher Schreibung aufrufen, wird ein Skriptfehler angezeigt.

  Wenn Sie auf eine Variable oder Eigenschaft unter Verwendung der falschen Schreibung zugreifen, wird vielleicht keine Fehlermeldung angezeigt, doch ggf. funktioniert das Skript nicht wie gewünscht. Die folgende Prozedur `mouseUp` enthält beispielsweise eine Anweisung, die versucht, mit falscher Schreibung auf die Eigenschaft `itemLabel` zuzugreifen. Dieses Skript generiert keinen Skriptfehler, erstellt jedoch dynamisch eine neue Variable mit falscher Schreibung. Der Wert der neu erstellten Variablen ist `undefined`.

```
// JavaScript syntax
function beginSprite() {
    this.itemLabel = "Blue prints";
}

function mouseUp() {
    trace(this.itemlabel) // creates the itemlabel property
}
```

Debugging im Skriptfenster

Das Skriptfenster enthält mehrere Funktion zum Debuggen von Skripts.

So öffnen Sie das Skriptfenster

❖ Wählen Sie „Fenster„ > „Skript“.

So wandeln Sie die aktuelle Lingo-Zeile in einen Kommentar um

❖ Klicken Sie auf Kommentar.

So entfernen Sie den Kommentar von der aktuellen Codezeile

❖ Klicken Sie auf Kommentar entfernen.

So aktivieren und deaktivieren Sie die Haltepunkte in der aktuellen Lingo-Zeile

❖ Klicken Sie auf Breakpoint ein-/ausschalten.

So deaktivieren Sie alle Haltepunkte

❖ Klicken Sie auf Breakpoints übergehen.

So nehmen Sie den ausgewählten Ausdruck oder die ausgewählte Variable in den Objektinspektor auf

❖ Klicken Sie auf Objekt inspizieren.

Debuggen im Nachrichtenfenster

Im Nachrichtenfenster können Sie Skriptbefehle testen und verfolgen, was während der Filmwiedergabe in Ihren Skripts abläuft.

So öffnen Sie das Nachrichtenfenster

❖ Wählen Sie „Fenster“ > „Nachricht“.

Verwalten des Nachrichtenfensters

Das Nachrichtenfenster besteht aus einem Eingabe- und einem Ausgabebereich. Der Eingabebereich ist bearbeitbar. Der Ausgabebereich ist schreibgeschützt. Die einzige Möglichkeit zum Anzeigen von Text im Ausgabebereich ist das Aufrufen der Funktion `put()` oder `trace()`.

Sie können die Größe des Eingabe- und des Ausgabebereichs ändern, indem Sie den horizontalen Trennbalken zwischen den beiden Bereichen nach oben oder unten verschieben.

So ändern Sie die Größe des Ausgabebereichs

❖ Ziehen Sie den horizontalen Trennbalken an eine andere Position.

So blenden Sie den Ausgabebereich aus

❖ Klicken Sie auf die Schaltfläche Ein-/Ausblenden in der Mitte des horizontalen Trennbalkens.

Bei ausgeblendetem Ausgabebereich wird die Ausgabe während der Lingo-Ausführung im Eingabebereich angezeigt.

So blenden Sie den Ausgabebereich wieder ein

❖ Klicken Sie erneut auf die Schaltfläche Ein-/Ausblenden.

So löschen Sie den Inhalt des Nachrichtenfensters

❖ Klicken Sie auf die Schaltfläche Löschen.

Wenn der Ausgabebereich eingeblendet ist, wird sein Inhalt gelöscht.

Wenn der Ausgabebereich nicht eingeblendet ist, wird der Inhalt des Eingabebereichs gelöscht.

So löschen Sie einen Teil des Inhalts im Ausgabebereich

1 Markieren Sie den Text, den Sie löschen möchten.

2 Drücken Sie die RÜCKTASTE bzw. ENTF-TASTE.

So kopieren Sie Text in den Eingabe- oder den Ausgabebereich

1 Markieren Sie den gewünschten Text.

2 Wählen Sie „Bearbeiten“ > „Kopieren“.

Testen von Skripts im Nachrichtenfenster

Sie können die Funktionsweise von Lingo- und JavaScript-Anweisungen testen, indem Sie sie in das Nachrichtenfenster eingeben und das Ergebnis überprüfen. Ein Befehl, den Sie in das Nachrichtenfenster eingeben, wird von Director sofort ausgeführt, unabhängig davon, ob gerade ein Film wiedergegeben wird oder nicht.

Bevor Sie die zu testenden Anweisungen eingeben, müssen Sie zuerst angeben, welche Skriptsyntax (Lingo oder JavaScript) Sie testen möchten.

So wählen Sie die Skriptsyntax aus

❖ Wählen Sie im Popupmenü „Skriptsyntax“ entweder „Lingo“ oder „JavaScript“ aus.

So testen Sie eine einzeilige Anweisung

1 Geben Sie die Anweisung direkt in das Nachrichtenfenster ein.

2 Drücken Sie die Eingabetaste (Windows®) bzw. den Zeilenschalter (Macintosh®). Director führt die Anweisung aus.

Ist die Anweisung gültig, wird ihr Ergebnis im Ausgabebereich in der unteren Hälfte des Nachrichtenfensters angezeigt. Ist das Skript ungültig, erscheint ein entsprechender Warnhinweis.

Geben Sie beispielsweise die folgende Anweisung in das Nachrichtenfenster ein:

```
-- Lingo syntax
put(50+50)

// JavaScript syntax
trace(50+50);
```

Wenn Sie die Eingabetaste drücken, erscheint das Ergebnis im Ausgabebereich:

```
-- Lingo syntax
-- 100

// JavaScript syntax
// 100
```

Geben Sie die folgende Anweisung in das Nachrichtenfenster ein:

```
-- Lingo syntax
_movie.stage.bgColor = 255

// JavaScript syntax
_movie.stage.bgColor = 255;
```

Wenn Sie die Eingabetaste (Windows®) oder die Return-Taste (Macintosh®) drücken, wird die Bühne schwarz.

Sie können mehrere Codezeilen gleichzeitig testen, indem Sie die Anweisungen kopieren und in das Nachrichtenfenster einfügen oder die einzelnen Zeilen durch Drücken von UMSCHALTTASTE+EINGABETASTE abschließen.

So führen Sie mehrere Codezeilen durch Kopieren und Einfügen aus

1. Kopieren Sie die gewünschten Codezeilen in die Zwischenablage.
2. Geben Sie eine leere Zeile in das Nachrichtenfenster ein.
3. Fügen Sie den Code aus der Zwischenablage in den Eingabebereich des Nachrichtenfensters ein.
4. Bewegen Sie die Einfügemarke an das Ende der letzten Codezeile.
5. Drücken Sie STRG-EINGABETASTE (Windows) oder CTRL+EINGABETASTE (Mac). Director findet die erste leere Zeile oberhalb der Einfügemarke und führt die auf die leere Zeile folgenden Codezeilen der Reihe nach aus.

So geben Sie mehrere Codezeilen manuell ein

1. Geben Sie eine leere Zeile in das Nachrichtenfenster ein.
2. Geben Sie die erste Codezeile ein.
3. Schließen Sie die Zeile mit UMSCHALTTASTE+EINGABETASTE ab.
4. Wiederholen Sie die Schritte 2 und 3, bis Sie alle gewünschten Codezeile eingegeben haben.
5. Drücken Sie STRG-EINGABETASTE (Windows) oder CTRL+EINGABETASTE (Mac). Director findet die erste leere Zeile oberhalb der Einfügemarke und führt die auf die leere Zeile folgenden Codezeilen der Reihe nach aus.

Sie können eine Prozedur testen, ohne den Film wiederzugeben, indem Sie sie in ein Filmskript- oder Verhaltensskriptfenster eingeben und anschließend vom Nachrichtenfenster aus aufrufen.

So testen Sie eine Prozedur

1. Fügen Sie entweder durch Kopien und Einfügen oder manuell eine mehrzeilige Prozedur in das Nachrichtenfenster ein (siehe die beiden zuvor beschriebenen Verfahren).
2. Bewegen Sie die Einfügemarke an das Ende der letzten Codezeile.
3. Drücken Sie die EINGABETASTE. Die Prozedur wird ausgeführt.

   Die Ausgaben der in der Prozedur enthaltenen `put()`- oder `trace()`-Anweisungen werden im Nachrichtenfenster angezeigt.

Ebenso wie das Skriptfenster enthält auch das Nachrichtenfenster Popupmenüs mit Skriptbefehlen. Wenn Sie in einem dieser Popupmenüs einen Befehl auswählen, erscheint dieser Befehl im Nachrichtenfenster, wobei das erste anzugebende Argument hervorgehoben wird. Über die verschiedenen Menüs können Sie jederzeit mühelos auf sämtliche Skriptbegriffe zugreifen.

Folgende Popupmenüs stehen zur Verfügung:

- Im Menü „Lingo in alphabetischer Reihenfolge“ sind alle Elemente, mit Ausnahme der 3D-Lingo-Elemente, in alphabetischer Reihenfolge aufgeführt.
- Im Menü „Lingo nach Kategorie“ sind alle Elemente, mit Ausnahme der 3D-Lingo-Elemente, nach Funktionskategorien geordnet aufgeführt.
- Im Menü „3D-Lingo alphabetisch geordnet“ sind alle 3D-Lingo-Elemente in alphabetischer Reihenfolge aufgeführt.
- Im Menü „3D-Lingo nach Kategorie“ sind alle 3D-Lingo-Elemente nach Funktionskategorien geordnet aufgeführt.

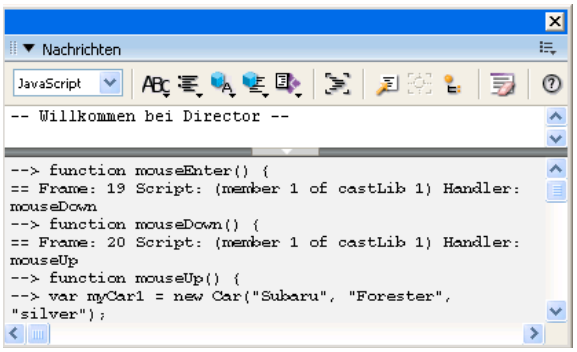
- Das Popupmenü „Skript-Xtras“ enthält die Methoden und Eigenschaften aller gefundenen Xtra-Skripterweiterungen von Adobe® und anderen Anbietern.

Hinweis: *Die im Popupmenü „Skript-Xtras“ aufgeführten Xtra-Skripterweiterungen sind ausschließlich diejenigen, die die* `Interface()`*-Methode unterstützen und deren Namen tatsächlich im Popupmenü angezeigt werden. Obgleich einige Darstellermedientypen wie 3D und DVD auch die* `Interface()`*-Methode unterstützen, werden sie nicht im Popupmenü „Skript-Xtras“ angezeigt, da sie nicht in Director als Xtras-Skripterweiterungen implementiert sind.*

Überwachen von Skripts im Nachrichtenfenster

Sie können den Ausgabebereich des Nachrichtenfensters so einrichten, dass ein Protokoll der Anweisungen angezeigt wird, die ein Film bei seiner Wiedergabe ausführt. Anhand dieses Protokolls können Sie den Ablauf des Codes verfolgen und die Ergebnisse bestimmter Anweisungen überprüfen. Dazu haben Sie zwei Möglichkeiten.

- Klicken Sie im Nachrichtenfenster auf die Schaltfläche „Ablaufverfolgung“.
- Legen Sie die `traceScript`-Eigenschaft des Movie-Objekts auf `TRUE` fest.



Einträge, denen ein doppeltes Ist-gleich-Zeichen (==) vorangestellt ist, geben an, was im Film geschieht, also beispielsweise in welches Bild der Film eingetreten ist, welches Skript ausgeführt wird und welches Resultat eine Funktion oder das Einstellen eines Werts ergeben hat.

Die folgende Zeile beispielsweise enthält verschiedene Informationen:

```
== Frame: 39 Script: 1 Handler: mouseUp
```

- Der Film ist in Bild 39 eingetreten.
- Der Film hat Skript 1, das erste dem Bild zugeordnete Skript, ausgeführt.
- Der Film hat die Prozedur `mouseUp` in Skript 1 ausgeführt, nachdem er in das Bild eingetreten ist.

Einträge, denen ein Pfeil aus zwei Bindestrichen und einer spitzen Klammer (`-->`) vorangestellt ist, geben die Codezeilen an, die ausgeführt wurden. Die folgenden Lingo-Zeilen:

```
--> _sound.fadeOut(1, 5*60)
--> if leftSide < 10 then
--> if leftSide < 200 then
--> _movie.go("Game Start")
```

Diese Lingo-Anweisungen wurden ausgeführt. Angenommen, Sie versuchen festzustellen, warum der Abspielkopf nicht in das Bild mit der Bezeichnung „Game Start“ eingetreten ist. Wenn die Zeile `--> _movie.go("Game Start")` im Nachrichtenfenster nicht angezeigt wird, kann dies bedeuten, dass die Bedingung in der vorherigen Anweisung entgegen Ihrer Erwartung nicht erfüllt war.

Bei aktivierter Option Verfolgen können im Ausgabebereich des Nachrichtenfensters rasch große Mengen Text angezeigt werden. Mithilfe der Schaltfläche Löschen können Sie den Inhalt des Ausgabebereichs löschen. Wenn der Ausgabebereich nicht eingeblendet ist, wird der Inhalt des Eingabebereichs gelöscht.

Sie können die Werte von Variablen und anderen Objekten verfolgen, indem Sie im Nachrichtenfenster den Namen des gewünschten Objekts auswählen und auf die Schaltfläche Objekt inspizieren klicken. Auf diese Weise nehmen Sie das Objekt in den Objektinspektor auf, in dem sein Wert während der Filmwiedergabe angezeigt und laufend aktualisiert wird. Weitere Informationen hierzu finden Sie unter „Debuggen im Objektinspektor“ auf Seite 90.

Im Debugging-Modus können Sie den Wert einer Variablen beobachten, indem Sie sie im Nachrichtenfenter auswählen und auf die Schaltfläche Ausdruck verfolgen klicken. Auf diese Weise nehmen Sie die Variable in den Watcher-Bereich des Debugger-Fensters auf, in dem ihr Wert während der Arbeit im Debugger-Fenster angezeigt und laufend aktualisiert wird. Weitere Informationen zum Watcher-Bereich finden Sie unter „Debuggen im Debugger-Fenster“ auf Seite 93.

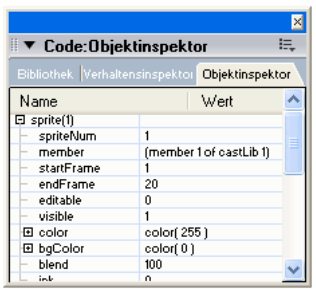
Debuggen im Objektinspektor

Mit dem Objektinspektor können Sie die Eigenschaften vieler Objekttypen überprüfen und einstellen, die nicht im Eigenschafteninspektor angezeigt werden. Hierzu gehören Skriptobjekte wie globale Variablen, Listen, Child-Objekte von Parent-Skripts, alle 3D-Darstellereigenschaften, Sprite-Eigenschaften, Skriptausdrücke und vieles mehr. Außerdem können Sie mit dem Objektinspektor Änderungen an Objekteigenschaften verfolgen, die während der Filmwiedergabe von Skripts oder veränderten Sprite-Eigenschaften verursacht werden. Derartige Änderungen werden im Eigenschafteninspektor während der Filmwiedergabe nicht angezeigt.

Damit Werte von JavaScript-Variablen im Objektinspektor angezeigt werden, müssen Sie diese ohne vorangestelltes *var* deklarieren.

So öffnen Sie den Objektinspektor

❖ Wählen Sie „Fenster“ > „Objektinspektor“.



Objektinspektor

Grundlegendes zur Objektstruktur

Der Objektinspektor eignet sich besonders zum Untersuchen der Struktur komplexer Objekte. 3D-Darsteller weisen zum Beispiel eine komplexe Eigenschaftenhierarchie auf, die sich dank der grafischen Darstellung der geschachtelten Struktur dieser Eigenschaften und ihrer Beziehungen untereinander im Objektinspektor wesentlich leichter durchschauen lässt. Das Verstehen der Eigenschaftenstrukturen von Director-Objekten ist beim Schreiben von Skripts von großer Bedeutung.

Da Sie beobachten können, wie sich die Werte der Eigenschaften während der Filmwiedergabe verändern, können Sie besser nachvollziehen, was im Film passiert. Besonders hilfreich ist dies beim Testen und Debuggen von Lingo-Skripts, da Sie die Auswirkungen Ihrer Skripts auf die einzelnen Werte mitverfolgen können.

Diese Informationen werden auch im Debugger-Fenster von Director angezeigt, das allerdings nur im Debugging-Modus verfügbar ist. Weitere Informationen zum Debugging finden Sie unter „Erweiterte Debugging-Funktionen“ auf Seite 99.

Anzeigbare Objekte

Es folgen einige Beispiele für Objekte, die Sie in den Objektinspektor eingeben können:

- Sprites, z. B. `sprite(3)`
- Darsteller, z. B. `member("3d")`
- Globale Variablen, z. B. `gMyList`
- Child-Objekte, z. B. `gMyChild`
- Adobe® Flash®-Objekte, z. B. `gMyFlashObject`. Weitere Informationen zur Verwendung von Flash-Objekten in Director finden Sie in den Themen des Director-Benutzerhandbuchs im Director-Hilfefenster.
- Skriptausdrücke, z. B. `sprite(7).blend`

Anzeigen von Objekten

Sie können ein Objekt auf drei Arten im Objektinspektor anzeigen, indem Sie es entweder mit der Maus direkt in den Objektinspektor ziehen, seinen Namen manuell in den Objektinspektor eingeben oder im Nachrichten- oder Skriptfenster auf die Schaltfläche „Objekt inspizieren“ klicken.

So ziehen Sie ein Objekt in den Objektinspektor

❖ Führen Sie einen der folgenden Schritte aus:

- Wählen Sie im Drehbuchfenster ein Sprite aus, und ziehen Sie es in den Objektinspektor.
- Wählen Sie im Besetzungsfenster einen Darsteller aus, und ziehen Sie ihn in den Objektinspektor.
- Wählen Sie im Skript-, Nachrichten- oder Textfenster den Namen eines Objekts aus, und ziehen Sie ihn in den Objektinspektor.

So geben Sie ein Objekt manuell in den Objektinspektor ein

1. Doppelklicken Sie auf die erste leere Zelle in der Spalte Name des Objektinspektors.
2. Geben Sie den Namen des Objekts in die Zelle ein. Verwenden Sie hierbei denselben Namen, mit dem Sie in Skripts auf das Objekt verweisen.
3. Drücken Sie die EINGABETASTE. Wenn das Objekt über Untereigenschaften verfügt, wird links neben dem Eintrag ein Pluszeichen (+) angezeigt.
4. Klicken Sie auf das Pluszeichen. Die Eigenschaften des Objekts werden darunter eingeblendet. Eigenschaften mit Untereigenschaften sind mit einem Pluszeichen links neben dem Eintrag gekennzeichnet. Klicken Sie auf die Pluszeichen, um die entsprechenden Untereigenschaften einzublenden.

So zeigen Sie ein Objekt durch Klicken auf die Schaltfläche „Objekt inspizieren“ an

1. Markieren Sie im Skriptfenster den Teil einer Anweisung, der auf ein Objekt verweist.
2. Klicken Sie im Skriptfenster auf „Objekt inspizieren“. Wenn das Objekt über Untereigenschaften verfügt, wird links neben dem Eintrag ein Pluszeichen (+) angezeigt.

3 Klicken Sie auf das Pluszeichen. Die Eigenschaften des Objekts werden darunter eingeblendet. Eigenschaften mit Untereigenschaften sind mit einem Pluszeichen links neben dem Eintrag gekennzeichnet. Klicken Sie auf die Pluszeichen, um die entsprechenden Untereigenschaften einzublenden.

Hinweis: *Das Inspizieren sehr vieler oder großer Einzelobjekte im Objektinspektor kann die Systemleistung während der Filmerstellung spürbar ausbremsen, insbesondere wenn „Automatische Abstimmung" aktiviert ist. Eine Liste mit beispielsweise 10.000 Einträgen kann bewirken, dass Director blockiert erscheint, während die Anzeige aktualisiert wird.*

Durchsuchen von Objekten

Sie können die Einträge im Objektinspektor auch mithilfe der Pfeiltasten Ihrer Tastatur durchsuchen.

So wählen Sie den jeweils nächsten oder vorhergehenden Eintrag in der Liste aus

❖ Drücken Sie die NACH-OBEN- bzw. NACH-UNTEN-TASTE.

So zeigen Sie die Untereigenschaften eines Objekts an

❖ Wählen Sie den Eintrag aus, und drücken Sie die NACH-RECHTS-TASTE.

So blenden Sie die Untereigenschaften eines Objekts aus

❖ Wählen Sie den Eintrag aus, und drücken Sie die NACH-LINKS-TASTE.

Verwenden von „Automatische Abstimmung"

Systemeigenschaften, wie `milliseconds` und `colorDepth`, werden im Objektinspektor nur aktualisiert, wenn die Option „Automatische Abfrage" aktiviert ist. Da diese Option den Prozessor stark beansprucht, kann das Hinzufügen einer größeren Anzahl von Systemeigenschaften zum Objektinspektor die Leistung Ihres Films beeinträchtigen.

So aktivieren Sie die automatische Abstimmungfrage

1 Klicken Sie mit der rechten Maustaste (Windows) oder bei gedrückter CTRL-TASTE (Mac) auf den Objektinspektor. Das Kontextmenü des Objektinspektors wird angezeigt.

2 Wählen Sie im Kontextmenü die Option Automatische Abfrage Wenn Sie diese Option aktiviert haben, ist der Eintrag Automatische Abfrage im Kontextmenü mit einem Häkchen gekennzeichnet.

So deaktivieren Sie die automatische Abfrage

❖ Wählen Sie im Kontextmenü erneut die Option „Automatische Abfrage".

Ändern von Objekt- und Eigenschaftswerten

Sie können den Wert eines Objekts oder einer Eigenschaft im Objektinspektor festlegen, indem Sie in das Feld rechts neben dem Objekt- oder Eigenschaftsnamen einen neuen Wert eintragen.

So legen Sie einen Objekt- oder Eigenschaftswert fest

1 Doppelklicken Sie auf den Wert rechts neben dem Elementnamen.

2 Geben Sie den neuen Wert für das Element ein.

3 Drücken Sie die EINGABETASTE. Der neue Wert wird eingestellt und der Film automatisch aktualisiert.

Sie können auch einen Skriptausdruck als Wert angeben und beispielsweise den Wert von `sprite(3).locH` auf den Ausdruck `sprite(8).locH + 20` einstellen.

Entfernen von Objekten

Sie können auch Elemente aus dem Objektinspektor entfernen.

So entfernen Sie ein einzelnes Element aus dem Objektinspektor

❖ Wählen Sie das Element aus, und drücken Sie die RÜCKSCHRITTTASTE (Windows) oder LÖSCHTASTE (Macintosh).

So löschen Sie den gesamten Inhalt des Objektinspektors

❖ Klicken Sie mit der rechten Maustaste (Windows) oder bei gedrückter CTRL-TASTE (Macintosh) auf den Objektinspektor, und wählen Sie im Kontextmenü „Alle löschen".

Wenn Sie in dem Film, an dem Sie arbeiten, einen anderen Film öffnen, werden die Objekte, die Sie in den Objektinspektor eingegeben haben, nicht entfernt. Auf diese Weise können Sie verschiedene Versionen eines Films mühelos miteinander vergleichen. Beim Beenden von Director wird der Inhalt des Objektinspektors gelöscht.

Debuggen im Debugger-Fenster

Beim Debugger-Fenster handelt es sich um eine speziellen Modus des Skriptfensters. Es bietet eine Reihe von Werkzeugen, mit denen sich die Ursachen von Problemen in Skripts feststellen lassen. Mithilfe des Debugger-Fensters können Sie rasch und mühelos die Abschnitte Ihres Codes ermitteln, die für auftretende Probleme verantwortlich sind. Die Funktionen des Debugger-Fensters ermöglichen es, Skripts zeilenweise auszuführen, geschachtelte Prozeduren zu überspringen, Skripttext zu bearbeiten sowie die Werte von Variablen und anderen Objekten zu überwachen. Durch den Umgang mit den Werkzeugen im Debugger-Fenster haben Sie die Gelegenheit, so manches über effizientes Programmieren zu lernen.

Im Debugger-Fenster können Sie Fehler in Skripts ermitteln und beheben. Es bietet eine Reihe von Werkzeugen, mit denen sich folgende Aufgaben ausführen lassen:

- Den Skriptabschnitt anzeigen, der die aktuelle Codezeile enthält.
- Die Folge der Prozeduren nachvollziehen, die vor der aktuellen Prozedur aufgerufen wurden.
- Ausgewählte Teile der aktuellen Prozedur ausführen.
- Ausgewählte Teile geschachtelter Prozeduren ausführen, die von der aktuellen Prozedur aufgerufen werden.
- Die Werte aller lokalen und globalen Variablen sowie der für den untersuchten Codeabschnitt relevanten Eigenschaften anzeigen.

Aktivieren des Debugging-Modus

Um das Debugger-Fenster zu öffnen, muss die Skriptausführung angehalten werden. Dies geschieht, wenn Director auf einen Skriptfehler oder Haltepunkt im Skript trifft.

Wenn ein Skriptfehler auftritt, wird das Dialogfeld „Skriptfehler" angezeigt. Dieses Dialogfeld enthält Informationen zur Art des aufgetretenen Fehlers. Außerdem können Sie angeben, ob Sie das Skript debuggen, im Skriptfenster bearbeiten oder den Vorgang abbrechen möchten.

So aktivieren Sie den Debugging-Modus

❖ Führen Sie einen der folgenden Schritte aus:

- Klicken Sie im Dialogfeld Skriptfehler auf Debuggen.

- Fügen Sie einen Haltepunkt in ein Skript ein.

Wenn Director bei der Ausführung eines Skripts auf einen Haltepunkt trifft, wird das Skript angehalten und das Skriptfenster in den Debugging-Modus versetzt. Der Film läuft weiter, doch die Ausführung Ihrer Skripts wird unterbrochen, bis Sie Director über das Debugger-Fenster mitteilen, wie die Verarbeitung fortgesetzt werden soll. Wenn Sie mehrere Skriptfenster geöffnet haben, sucht Director das Skript, das den Haltepunkt enthält, und versetzt das entsprechende Skriptfenster in den Debugging-Modus.

Drücken Sie die Tastenkombination Strg+F11, um das Debugging-Fenster in den Vordergrund zu bringen.

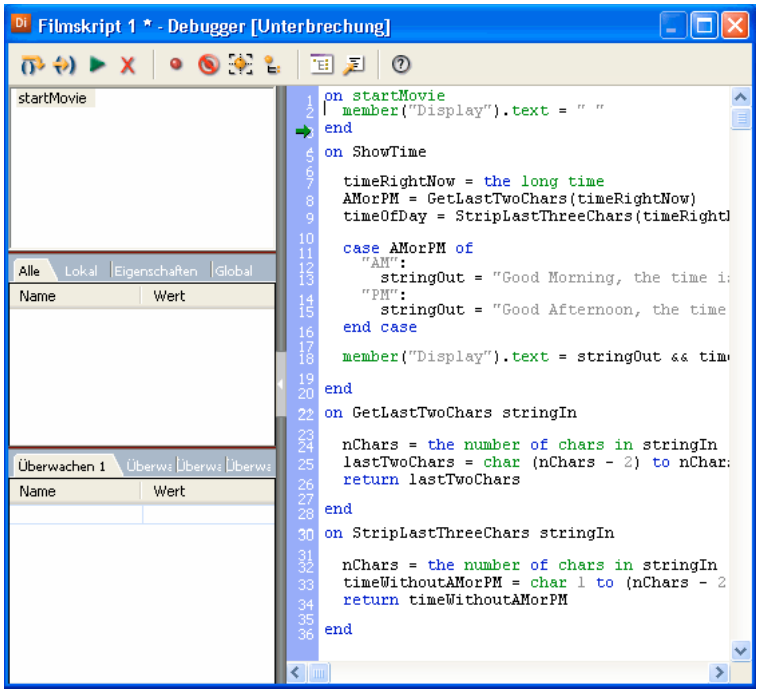
Hinweis: *Sie müssen den maximierten Registerkartenmodus verlassen, um nach dem Starten des Films zum Fehlerbehebungsfenster zurückzugelangen. Für diesen Vorgang kann nicht die Tastenkombination Strg+Tab verwendet werden.*

So fügen Sie einen Haltepunkt hinzu, um das Debugger-Fenster zu öffnen

1 Öffnen Sie im Skriptfenster das Skript, das den Haltepunkt enthalten soll.

2 Klicken Sie auf den linken Rand des Skriptfensters neben der Codezeile, in der Sie den Haltepunkt wünschen, oder platzieren Sie die Einfügemarke auf der Codezeile, und klicken Sie auf „Haltepunkt ein-/ausschalten“. Die Ausführung des Codes wird am Anfang dieser Zeile unterbrochen und das Skriptfenster in den Debugging-Modus versetzt.

Wenn das Skriptfenster geöffnet ist, während Director auf einen Skriptfehler oder Haltepunkt trifft, wird es auf dem Bildschirm durch das Debugger-Fenster ersetzt.



Debugger-Fenster

So beenden Sie das Debugging

❖ Führen Sie einen der folgenden Schritte aus:

- Klicken Sie im Debugger-Fenster auf die Schaltfläche Skript ausführen. Dadurch wird die normale Skriptausführung fortgesetzt.

- Klicken Sie im Debugger-Fenster auf die Schaltfläche Debugging stoppen. Dadurch werden die Debugging-Sitzung und der Film beendet.

  Anstelle des Debugger-Fensters wird wieder das Skriptfenster angezeigt.

Wenn das Debugger-Fenster geöffnet ist, werden die aktuelle Codezeile und eine Liste der Möglichkeiten angezeigt, die Verarbeitung fortzusetzen.

So ermitteln Sie die aktuelle Codezeile

❖ Suchen Sie im Skriptbereich nach dem grünen Pfeil neben einer der Codezeilen.

Der grüne Pfeil zeigt auf die aktuelle Zeile. Das Auswählen einer anderen Codezeile im Skriptbereich ist nicht möglich.

Anzeigen der Prozeduraufrufliste im Debugger-Fenster

In der Prozeduraufrufliste wird die Abfolge der geschachtelten Prozeduren angezeigt, die bis zur aktuellen Codezeile ausgeführt wurden. Anhand der Prozeduraufrufliste können Sie die Struktur Ihres Codes während des Debugging mitverfolgen. Wenn Sie in der Prozeduraufrufliste auf einen Prozedurnamen klicken, werden die dieser Prozedur zugeordneten Variablen im Variablenbereich angezeigt.

Anzeigen von Variablen im Debugger-Fenster

Im Variablenbereich des Debugger-Fensters werden die der aktuellen Prozedur zugeordneten Variablen angezeigt. Die aktuelle Prozedur selbst erscheint im Skriptbereich und ist als letzter Eintrag in der Prozeduraufrufliste aufgeführt. Sie haben auch die Möglichkeit, die den vorhergehenden Prozeduren in der Prozeduraufrufliste zugeordneten Variablen anzuzeigen. Beim Durchlaufen eines Skripts werden Variablen, deren Werte sich geändert haben, rot dargestellt. Weitere Informationen über den Einzelschrittaufruf von Skripts finden Sie unter „Schrittweises Durchlaufen von Skripts im Debugger-Fenster“ auf Seite 97.

So zeigen Sie die einer Prozedur in der Prozeduraufrufliste zugeordneten Variablen an

❖ Klicken Sie in der Prozeduraufrufliste auf den Namen der gewünschten Prozedur. Die Variablen werden im Variablenbereich angezeigt.

Der Variablenbereich enthält vier Registerkarten für die Anzeige der Variablen:

Alle – Auf dieser Registerkarte werden sowohl globale als auch lokale Variablen angezeigt, die der aktuellen Prozedur zugeordnet sind.

Lokal – Auf dieser Registerkarte werden nur die lokalen Variablen der ausgewählten Prozedur angezeigt.

Eigenschaft – Auf dieser Registerkarte werden die durch das aktuelle Skript deklarierten Eigenschaften angezeigt.

Global – Auf dieser Registerkarte werden nur die globalen Variablen der ausgewählten Prozedur angezeigt.

So sortieren Sie die Variablen im Variablenbereich

- Zum Sortieren der Variablen nach Namen klicken Sie auf das Wort *Name* in der Kopfzeile der Variablenliste.
- Zum Umkehren der Sortierfolge klicken Sie ein zweites Mal auf das Wort *Name*.

Sie können die Werte lokaler Variablen der aktuellen Prozedur und globaler Variablen im Variablenbereich ändern. Die Werte lokaler Variablen, die nicht mit der aktuellen Prozedur verknüpft sind, lassen sich auf diese Weise nicht ändern.

So ändern Sie den Wert einer Variablen im Variablenbereich

1 Doppelklicken Sie auf den Wert der Variablen in der Spalte „Wert“.

2 Geben Sie den neuen Wert für die Variable ein.

3 Drücken Sie die EINGABETASTE.

Anzeigen von Objekten im Debugger-Fenster

Im Watcher-Bereich des Debugger-Fensters können Sie Variablen und andere Datenobjekte der aktuellen Prozedur sowie Objekte anderer Prozeduren anzeigen. Indem Sie Objekte dem Watcher-Bereich hinzufügen, können Sie beobachten, wie sich ihre Werte während der Ausführung von Skripts ändern. Wenn der Wert eines Objekts in einer Codezeile geändert wird, zeigt Director den Namen des betreffenden Objekts im Watcher-Bereich rot an.

Der Watcher-Bereich enthält nur die Objekte, die Sie ihm hinzugefügt haben. Er weist vier Registerkarten auf, mit deren Hilfe Sie die Objekte im Watcher-Bereich zu Gruppen zusammenfassen können.

So nehmen Sie ein Objekt in den Watcher-Bereich auf, dessen Name im Skriptbereich angezeigt wird

1 Wählen Sie den Namen des Objekts im Skriptbereich aus.

2 Klicken Sie auf die Schaltfläche Ausdruck verfolgen.

So nehmen Sie ein Objekt in den Watcher-Bereich auf, dessen Name nicht im Skriptbereich angezeigt wird

1 Doppelklicken Sie auf die erste leere Zelle in der Objektspalte des Watcher-Bereichs.

2 Geben Sie den Namen des gewünschten Objekts in die Zelle ein, und drücken Sie die EINGABETASTE.

Wenn das Objekt über Eigenschaften verfügt, erscheint neben dem Objektnamen ein Pluszeichen (+).

So zeigen Sie die Eigenschaften eines Objekts an

❖ Klicken Sie auf das Pluszeichen neben dem Objektnamen.

Die Objekte im Watcher-Bereich lassen sich auf verschiedene Weise sortieren und gliedern.

Organisieren von Objekten im Watcher-Bereich

❖ Führen Sie einen der folgenden Schritte aus:

- Zum Sortieren der Objekte im Watcher-Fenster klicken Sie auf den Titel der Spalte Name auf der linken Seite der Liste. Die Objektnamen in der Spalte werden in alphabetischer Reihenfolge angezeigt.
- Zum Umkehren der Sortierfolge klicken Sie ein zweites Mal auf den Spaltentitel Name.
- Zum Gruppieren der Objekte verwenden Sie die Registerkarten im Watcher-Bereich. Wenn Sie ein Objekt einer bestimmten Registerkarte hinzufügen möchten, klicken Sie auf die gewünschte Registerkarte, bevor Sie das Objekt hinzufügen.
- Zum Löschen des Inhalts einer Registerkarte im Watcher-Bereichs wählen Sie die gewünschte Registerkarte aus, klicken mit der rechten Maustaste (Windows) oder bei gedrückter CTRL-TASTE (Macintosh) auf den Watcher-Bereich und wählen „Alle löschen“.

Schrittweises Durchlaufen von Skripts im Debugger-Fenster

Das Debugger-Fenster bietet eine Reihe von Werkzeugen, mit deren Hilfe Sie ein Skript langsam durchlaufen und die Auswirkungen der einzelnen Codezeilen auf Ihren Film beobachten können. Hierbei können Sie den Code zeilenweise ausführen und jeweils angeben, ob die geschachtelten Prozeduren ebenfalls zeilenweise oder im Ganzen ausgeführt werden sollen.

So führen Sie nur die mit dem grünen Pfeil gekennzeichnete, aktuelle Codezeile aus

❖ Klicken Sie auf die Schaltfläche Skript schrittweise ausführen.

Viele Prozeduren enthalten Anweisungen, mit denen andere Prozeduren aufgerufen werden. Sie können diese geschachtelten Prozeduren ebenfalls genauer untersuchen oder sie ignorieren und sich ganz auf den Code der aktuellen Prozedur konzentrieren.

Wenn Sie sicher sind, dass die geschachtelten Prozeduren ordnungsgemäß funktionieren, und sich auf den Code der aktuellen Prozedur konzentrieren möchten, können Sie die geschachtelten Prozeduren im Debugger-Fenster überspringen und direkt zur jeweils nächsten Codezeile der aktuellen Prozedur springen. Hierbei wird die verschachtelte Prozedur zwar ausgeführt, ihr Code jedoch nicht angezeigt und die Verarbeitung nicht unterbrochen.

So überspringen Sie geschachtelte Prozeduren

❖ Klicken Sie im Debugger-Fenster auf die Schaltfläche Skript schrittweise ausführen.

Die Schaltfläche „Skript schrittweise ausführen“ führt die aktuelle Codezeile aus, verarbeitet alle geschachtelten Prozeduren, die in dieser Zeile aufgerufen werden, und hält bei der nächsten Zeile der aktuellen Prozedur an.

Wenn Sie vermuten, dass die geschachtelten Prozeduren nicht ordnungsgemäß funktionieren, und ihr Verhalten genauer untersuchen möchten, können Sie im Debugger-Fenster auch geschachtelte Prozeduren zeilenweise ausführen.

So führen Sie geschachtelte Prozeduren zeilenweise aus

❖ Klicken Sie im Debugger-Fenster auf die Schaltfläche Im Skript schrittweise vorangehen.

Die aktuelle Codezeile und alle in dieser Zeile aufgerufenen geschachtelten Prozeduren werden in der üblichen Abfolge zeilenweise ausgeführt. Nach Beendigung einer geschachtelten Prozedur hält der Debugger an der nächsten Codezeile in der übergeordneten Prozedur an.

Wenn Sie mit dem Debugging fertig sind, können Sie den Debugger jederzeit beenden.

So wird die normale Codeausführung fortgesetzt und das Debugger-Fenster geschlossen

❖ Klicken Sie auf die Schaltfläche „Skript ausführen“.

So wird der Debugger geschlossen und die Filmwiedergabe beendet

❖ Klicken Sie auf die Schaltfläche „Debugging beenden“.

Skripts im Debugging-Modus bearbeiten

Im Debugging-Modus können Sie Ihre Skripts direkt im Debugger-Fenster bearbeiten. Auf diese Weise ist es möglich, erkannte Fehler sofort zu beheben und direkt mit dem Debuggen fortzufahren.

So bearbeiten Sie ein Skript im Debugger-Fenster

1 Klicken Sie auf den Skriptbereich, und bewegen Sie die Einfügemarke an die Stelle, an der Sie mit der Eingabe beginnen möchten.

2 Geben Sie die Änderungen in das Skript ein.

   Sie können direkt zu einer bestimmten Prozedur springen, indem Sie ihren Namen auswählen und auf die Schaltfläche „Gehe zu Prozedur“ klicken.

3 Wenn Sie mit dem Debuggen und Bearbeiten des Skripts fertig sind, klicken Sie auf die Schaltfläche Debugging stoppen. Das Skriptfenster kehrt in den Skriptmodus zurück.

4 Klicken Sie auf die Schaltfläche Alle geänderten Skripts rekompilieren.

Debuggen von Projektoren und Shockwave-Filmen

In diesem Abschnitt wird das Debugging zur Laufzeit in Director-Projektoren und -Filmen mit Adobe® Shockwave®-Inhalten behandelt. Sie können entweder das Nachrichtenfenster verwenden oder vollständige Dialogfelder vom Typ „Skriptfehler“ aktivieren, um Fehler in Projektoren und Filmen mit Shockwave-Inhalten zu beheben.

So debuggen Sie im Nachrichtenfenster

❖ Legen Sie die `debugPlaybackEnabled`-Eigenschaft des Player-Objekts auf `TRUE` fest.

   Ist diese Eigenschaft auf `TRUE` festgelegt, wird bei Wiedergabe eines Projektors oder Films mit Shockwave-Inhalt ein Nachrichtenfenster (Windows) oder eine Nachrichtentextdatei (Mac) geöffnet. Ergebnisse von Aufrufen der Funktionen `put()` und `trace()` werden in diesen Format ausgegeben.

   Wird im Verlauf der Filmwiedergabe die Eigenschaft `debugPlaybackEnabled` auf `FALSE` festgelegt, wird das Nachrichtenfenster bzw. die Textdatei geschlossen und kann während der Wiedergabesitzung nicht mehr geöffnet werden, auch wenn im weiteren Verlauf der Sitzung `debugPlaybackEnabled` wieder auf `TRUE` gesetzt wird.

So führen Sie ein Debugging durch, indem Sie vollständige Dialogfelder vom Typ „Skriptfehler“ aktivieren

❖ Legen Sie in einer INI-Datei für einen Projektor oder Film mit Shockwave-Inhalt die Eigenschaft `DisplayFullLingoErrorText` auf 1 fest.

   Dadurch wird im Dialogfeld „Skriptfehler“ ein aussagekräftigerer Text als allgemein üblich angezeigt. Eine allgemeine Fehlermeldung sieht beispielsweise so aus:

```
Script error: Continue?
```

   Durch Festlegen der Eigenschaft `DisplayFullLingoErrorText` auf 1 sieht die Fehlermeldung wie folgt aus:

```
Script error: list expected
```

   Informationen zum Erstellen und Ändern einer entsprechenden INI-Datei für einen Projektor oder Film mit Shockwave-Inhalt finden Sie in der Vorlagedatei „Director.ini“ im Installationsstammverzeichnis von Director.

Erweiterte Debugging-Funktionen

Wenn das Problem nicht leicht zu ermitteln ist, probieren Sie es mit den folgenden Methoden:

- Stellen Sie fest, in welchem Bereich das Problem auftritt. Wenn eine Schaltfläche beispielsweise die falsche Funktion ausführt, überprüfen Sie das dieser Schaltfläche zugeordnete Skript.

  Wenn sich ein Sprite nicht so verhält, wie es sollte, überprüfen Sie, ob seine Sprite-Eigenschaften auf die gewünschten Werte eingestellt sind.

- Betrachten Sie den Skriptablauf. Wenn in einem Abschnitt des Films nicht das geschieht, was Sie erwarten, gehen Sie zunächst die Abfolge der Filmereignisse im Kopf durch. Sehen Sie sich die übrigen Skripts in der Nachrichtenhierarchie an, und vergewissern Sie sich, dass Director die richtige Prozedur ausführt.

- Nutzen Sie die Verfolgungsfunktion im Nachrichtenfenster. Sie zeigt an, welche Bilder der Film durchläuft und welche Prozeduren während der Filmwiedergabe aufgerufen werden.

- Verwenden Sie die Funktionen Skript schrittweise ausführen und Im Skript schrittweise vorangehen im Debugger-Fenster, und überprüfen Sie, ob die Ergebnisse Ihren Erwartungen entsprechen.

- Überprüfen Sie Variablen und Ausdrücke, und analysieren Sie, wie sich ihre Werte während der Filmwiedergabe verändern. Prüfen Sie, ob sich die Werte vielleicht zum falschen Zeitpunkt oder überhaupt nicht ändern. Wenn eine Variable in mehreren Prozeduren verwendet wird, vergewissern Sie sich, dass sie in allen diesen Prozeduren als globale Variable deklariert ist.

  Sie können die Werte von Variablen und Ausdrücken im Watcher-Bereich des Debugger-Fensters oder im Objektinspektor verfolgen.

- Nehmen Sie Änderungen Schritt für Schritt vor. Scheuen Sie sich nicht, Änderungen an einer Prozedur vorzunehmen, um festzustellen, ob dadurch das Problem behoben oder ein Ergebnis erzielt wird, anhand dessen sich das Problem eingrenzen lässt.

  Achten Sie hierbei jedoch darauf, keine neuen Probleme zu verursachen. Nehmen Sie jeweils immer nur eine Änderung vor, und machen Sie sie wieder rückgängig, wenn das Problem dadurch nicht behoben wird. Wenn Sie zu viele Änderungen auf einmal vornehmen, haben Sie das Problem letztendlich zwar vielleicht behoben, können unter Umständen jedoch nicht genau sagen, wodurch es tatsächlich verursacht wurde, und laufen sogar Gefahr, neue Probleme zu erzeugen.

- Erstellen Sie den Abschnitt neu. Wenn Sie das Problem nicht finden können, versuchen Sie, den Abschnitt noch einmal neu zu erstellen. Falls sich ein Sprite beispielsweise nicht wie erwartet verhält, wenn sich der Cursor über ihm befindet, erstellen Sie einen einfachen Film, der nur das Sprite und die Prozedur mit der `rollOver()`-Anweisung enthält.

  Übernehmen Sie Skripts nicht einfach durch Kopieren und Einfügen, da Sie hierbei möglicherweise nur das Problem kopieren. Indem Sie den Abschnitt neu erstellen, können Sie die Logik in einem Minimalkontext neu aufbauen und sich vergewissern, dass Director wie erwartet arbeitet. Funktioniert der neu erstellte Abschnitt noch immer nicht richtig, so ist der Fehler wahrscheinlich in der Logik selbst zu suchen.

  Wenn er hingegen wie erwartet funktioniert, vergleichen Sie ihn mit dem entsprechenden Abschnitt in Ihrem ursprünglichen Film, und stellen Sie fest, wo sich die beiden Abschnitte unterscheiden. Sie können auch versuchen, den neuen Abschnitt in den ursprünglichen Film zu kopieren, um festzustellen, ob das Problem damit behoben ist.

Kapitel 5: Core-Objekte in Director

Die Core-Objekte in Director® geben Zugriff auf die Kernfunktionalität und -merkmale in Director, Projektoren und Adobe® Shockwave® Player. Zu den Core-Objekten gehören die Director-Player-Engine, Filmfenster, Sprites, Sounds usw. Sie stellen die Basisebene dar, über die der Zugriff auf fast alle APIs und anderen Objektkategorien erfolgt; Ausnahmen hierbei bilden die Skriptobjekte, welche die Kernfunktionalität von Director erweitern.

Das Diagramm unter „Objektmodelldiagramme“ auf Seite 47 zeigt, wie die Core-Objekte zueinander und zu anderen Objekten in Director in Bezug stehen.

Cast Library

Entspricht einer Besetzungsbibliothek innerhalb eines Filmes.

Ein Film kann aus einer oder mehreren Besetzungsbibliotheken bestehen. Eine Besetzungsbibliothek kann einen oder mehrere Darsteller umfassen, unter denen man die in einem Film verwendeten Medien wie Sounds, Text, Grafiken und andere Filme versteht.

Zum Erstellen eines Verweises auf eine Besetzungsbibliothek können Sie entweder die Top-Level-Funktion `castLib()` oder die Eigenschaft `castLib` des Movie-Objekts verwenden. Enthält ein Film z. B. eine Besetzungsbibliothek namens `scripts`, könnten Sie folgendermaßen einen Verweis auf diese Besetzungsbibliothek erstellen:

- Verwenden Sie die Top-Level-Methode castLib().

```
-- Lingo syntax
libScript = castLib("scripts")

// JavaScript syntax
var libScript = castLib("scripts");
```

- Verwenden Sie die Eigenschaft „castLib“ des Filmobjekts.

```
-- Lingo syntax
libScript = _movie.castLib["scripts"]

// JavaScript syntax
var libScript = _movie.castLib["scripts"];
```

Übersicht: Methoden des Cast Library-Objekts

Methode
findEmpty()

Übersicht: Eigenschaften des Cast Library-Objekts

Eigenschaft
fileName (Besetzung)
member (Besetzung)
name
number (Besetzung)
preLoadMode
selection

Siehe auch

`castLib`, `castLib()`, `Member`, `Movie`, `Player`, `Sprite`, `Window`

Global

Bietet einen Speicherort, an dem globale Variablen gespeichert und aufgerufen werden können. Diese Variablen stehen sowohl für Lingo als auch JavaScript-Syntax zur Verfügung.

Sie können über die Top-Level-Eigenschaft `_global` auf das Global-Objekt zugreifen. Sie haben die Möglichkeit, die Eigenschaft `_global` einer Variablen zuzuweisen oder die Eigenschaft `_global` direkt für den Zugriff auf die Methoden des Global-Objekts und eventuell definierte globale Variablen zu verwenden.

- Weisen Sie `_global` einer Variablen zu.

```
-- Lingo syntax
objGlobal = _global

// JavaScript syntax
var objGlobal = _global;
```

- Verwenden Sie die Eigenschaft „_global“ direkt.

```
-- Lingo syntax
_global.showGlobals()

// JavaScript syntax
_global.showGlobals();
```

- Greifen Sie auf eine globale Variable zu.

```
-- Lingo syntax
global gSuccess

on mouseDown
    gSuccess = "Congratulations!"
    put(gSuccess) -- displays "Congratulations!"
end

// JavaScript syntax
_global.gSuccess = "Congratulations!";

function mouseDown() {
    trace(_global.gSuccess); // displays "Congratulations!"
}
```

Übersicht: Methoden des Global-Objekts

Methode
clearGlobals()
showGlobals()

Siehe auch

_global

Taste

Dient dazu, die Tastaturaktivität von Benutzern zu überwachen.

Sie können über die Top-Level-Eigenschaft _key auf das Key-Objekt zugreifen. Sie haben die Möglichkeit, _key einer Variablen zuzuweisen oder die Eigenschaft _key direkt für den Zugriff auf die Methoden und Eigenschaften des Key-Objekts zu verwenden.

- Weisen Sie _key einer Variablen zu.

```
-- Lingo syntax
objKey = _key

// JavaScript syntax
var objKey = _key;
```

- Verwenden Sie die Eigenschaft „_key“ direkt.

```
-- Lingo syntax
isCtrlDown = _key.controlDown

// JavaScript syntax
var isCtrlDown = _key.controlDown;
```

Übersicht: Methoden des Key-Objekts

Methode
keyPressed()

Übersicht: Eigenschaften des Key-Objekts

Eigenschaft
commandDown
controlDown
key
keyCode
optionDown
shiftDown

Siehe auch

`_key`

Member

Entspricht einem Darsteller in einer Besetzungsbibliothek. Darsteller sind die Medien und Skriptelemente in Filmen. Zu Mediendarstellern gehören Text, Bitmaps, Formen usw. Zu Skriptdarstellern zählen Verhalten, Filmskripten usw.

Darsteller können anhand ihrer Nummer oder ihres Namens referenziert werden.

- Wenn Sie anhand der Nummer auf einen Darsteller verweisen, sucht Director in einer bestimmten Besetzungsbibliothek nach diesem Darsteller und ruft die zugehörigen Daten ab. Diese Methode ist schneller als der Darstellerverweis anhand des Namens. Da Director Verweise auf Darstellernummern in Skripts jedoch nicht automatisch aktualisiert, wird ein Nummernverweis auf einen Darsteller, der in der zugehörigen Besetzungsbibliothek an eine andere Position verschoben wurde, ungültig.
- Wenn Sie anhand des Namens auf einen Darsteller verweisen, durchsucht Director sämtliche Besetzungsbibliotheken des Films und ruft die Darstellerdaten ab, wenn der genannte Darsteller gefunden wurde. Diese Methode ist zeitaufwändiger als der Darstellerverweis anhand der Nummer. Dies gilt besonders bei Verweisen auf umfangreiche Filme mit vielen Besetzungsbibliotheken und Darstellern. Bei Namensverweisen auf Darsteller bleiben Verweise jedoch intakt, selbst wenn der Darsteller in der zugehörigen Besetzungsbibliothek an eine anderePosition verschoben wird.

Zum Erstellen eines Verweises auf einen Darsteller verwenden Sie entweder die Top-Level-Funktion `member()` oder die Eigenschaft `member` des Cast-, Movie- oder Sprite-Objekts.

Das folgende Beispiel zeigt, wie Sie einen Verweis auf einen Darsteller erstellen.

- Verwenden Sie die Top-Level-Funktion `member()`.

```
-- Lingo syntax
objTree = member("bmpTree")

// JavaScript syntax
var objTree = member("bmpTree");
```

- Verwenden Sie die Eigenschaft `member` des Sprite-Objekts.

```
-- Lingo syntax
objTree = sprite(1).member;

// JavaScript syntax
var objTree = sprite(1).member;
```

Übersicht: Methoden des Member-Objekts

Methode
copyToClipBoard()
duplicate() (Darsteller)
erase()
importFileInto()
move()
pasteClipBoardInto()
preLoad() (Darsteller)
unLoad() (Darsteller)

Übersicht: Eigenschaften des Member-Objekts

Eigenschaft	
castLibNum	modifiedDate
comments	name
creationDate	number (Darsteller)
fileName (Darsteller)	purgePriority, Eigenschaft
height	rect (Darsteller)
hilite	regPoint
linked	scriptText
loaded	size
media	thumbNail
mediaReady	type (Darsteller)
member	width
modified	
modifiedBy	

Siehe auch

`Medientypen`, `member()`, `member (Besetzung)`, `member (Film)`, `member (Sprite)`, `Movie`, `Player`, `Skriptobjekte`, `Sprite`, `Window`

Mixer

Ein Mixer ist ein Behälter zum Mischen aller in ihm enthaltenen Soundobjekte, der die resultierende Ausgabe abspielt. Da mehrere Audioquellen in eine einzige Audioquelle zusammengeführt werden, sparen Mixer durch das Verringern der durch die Soundkarte abgespielten Daten Ressourcen.

Übersicht: Methoden des Mixer-Objekts

Methode	
createSoundObject	deleteSoundObject
getSoundObject()	getSoundObjectList
mute (Mixer)	pause() (Mixer)
play() (Mixer)	reset (Mixer)
save (Mixer)	startSave (Mixer)
stop() (Mixer)	stopSave (Mixer)
unmute (Mixer)	

Übersicht: Eigenschaften des Mixer-Objekts

Eigenschaft	
bufferSize (Mixer)	bitDepth (Mixer)
channel	channelCount (Mixer)
elapsedTime (Mixer)	filterList (Mixer)
isSaving (Mixer)	name (Mixer)
numBuffersToPreload	panMatrix (Mixer)
sampleRate (Mixer)	status (Mixer)
soundObjectList	toChannels
useMatrix (Mixer)	volume (Mixer)

Mouse

Bietet Zugriff auf die Mausaktivität eines Benutzers, einschließlich Mausbewegungen und -klicks.

Sie können über die Top-Level-Eigenschaft `_mouse` auf das Mouse-Objekt zugreifen. Sie haben die Möglichkeit, `_mouse` einer Variablen zuzuweisen oder die Eigenschaft `_mouse` direkt für den Zugriff auf die Eigenschaften des Mouse-Objekts zu verwenden.

- Weisen Sie `_mouse` einer Variablen zu.

```
-- Lingo syntax
objMouse = _mouse

// JavaScript syntax
var objMouse = _mouse;
```

- Verwenden Sie die Eigenschaft „_mouse“ direkt.

```
-- Lingo syntax
isDblClick = _mouse.doubleClick

// JavaScript syntax
var isDblClick = _mouse.doubleClick;
```

Übersicht: Eigenschaften des Mouse-Objekts

Eigenschaft	
clickLoc	mouseLoc
clickOn	mouseMember
doubleClick	mouseUp
mouseChar	mouseV
mouseDown	mouseWord
mouseH	rightMouseDown
mouseItem	rightMouseUp
mouseLine	stillDown

Siehe auch

`_mouse`

Movie

Steht für einen Film, der im Director-Player wiedergegeben wird.

Der Director-Player kann einen oder mehrere Filme enthalten. Ein Film kann aus einer oder mehreren Besetzungsbibliotheken bestehen. Eine Besetzungsbibliothek kann einen oder mehrere Darsteller umfassen, unter denen man die in einem Film verwendeten Medien und Skriptelemente versteht. Zu Mediendarstellern gehören Text, Bitmaps, Formen usw. Zu Skriptdarstellern zählen Verhalten, Filmskripten usw. Sprites werden aus Darstellern erstellt und auf der Filmbühne eingesetzt.

Sie können mithilfe der Top-Level-Eigenschaft `_movie` auf den momentan aktiven Film Bezug nehmen. Über die Eigenschaft `movie` des zugehörigen Window-Objekts sind Verweise auf beliebige Filme im Player möglich.

- Nehmen Sie Bezug auf den momentan aktiven Film.

```
-- Lingo syntax
objMovie = _movie

// JavaScript syntax
var objMovie = _movie;
```

- Nutzen Sie die Eigenschaft „movie“ des Window-Objekts, um auf den Film in einem bestimmten Fenster zuzugreifen.

```
-- Lingo syntax
objMovie = _player.window[2].movie

// JavaScript syntax
var objMovie = _player.window[2].movie;
```

Filmverweise ermöglichen nicht nur den Zugriff auf die Methoden und Eigenschaften eines Films, sondern auch den Aufruf von Lingo- und JavaScript-Syntaxprozeduren sowie den Zugriff auf die Darsteller und Sprites des Filmes einschließlich ihrer Methoden und Eigenschaften. In früheren Versionen von Director musste der Befehl `tell` für die Arbeit mit Filmen verwendet werden. Das Movie-Objekt erleichtert die Arbeit mit Filmen.

Übersicht: Methoden des Movie-Objekts

Methode	
beginRecording()	marker()
call	mergeDisplayTemplate()
callAncestor	newMember()
cancelIdleLoad()	preLoad() (Film)
clearFrame()	preLoadMember()
constrainH()	preLoadMovie()
constrainV()	printFrom()
delay()	puppetPalette()
deleteFrame()	puppetSprite()
duplicateFrame()	puppetTempo()
endRecording()	puppetTransition()
finishIdleLoad()	ramNeeded()
frameReady() (Film)	rollOver()
getNthFileNameInFolder()	saveMovie()
go()	sendAllSprites()
goLoop()	sendSprite()
goNext()	stopEvent()
goPrevious()	unLoad() (Film)
halt()	unLoadMember()
idleLoadDone()	unLoadMovie()

Methode	
insertFrame()	updateFrame()
label()	updateStage()
loadPolicyFile()	

Übersicht: Eigenschaften des Movie-Objekts

Eigenschaft	
aboutInfo	idleReadChunkSize
active3dRenderer	imageCompression
activeCastLib	imageQuality
actorList	keyboardFocusSprite
allowTransportControl	lastChannel
allowVolumeControl	lastFrame
allowZooming	markerList
beepOn	member (Film)
buttonStyle	name
castLib	paletteMapping
centerStage	path (Film)
channel	preferred3dRenderer
copyrightInfo (Film)	preLoadEventAbort
displayTemplate	selEnd
editShortCutsEnabled	selection
enableFlashLingo	selStart
exitLock	score
fileFreeSize	scoreSelection
fileSize	script
fileVersion	sprite (Film)
fixStageSize	stage
frame	timeoutList
frameLabel	traceLoad
framePalette	traceLogFile
frameScript	traceScript
frameSound1	updateLock
frameSound2	updateMovieEnabled
frameTempo	useFastQuads

Eigenschaft	
frameTransition	window
idleHandlerPeriod	xtraList (Film)
idleLoadMode	xtraList (Film)
idleLoadPeriod	xtraList (Film)
idleLoadTag	xtraList (Film)

Siehe auch

`_movie`, `Cast Library`, `Member`, `movie`, `Player`, `Sprite`, `Window`

Player

Repräsentiert die Hauptwiedergabe-Engine für die Verwaltung und Ausführung der Autorenumgebung, von Filmen in einem Fenster (Movies in a Window, MIAW), Projektoren und Shockwave Player.

Das Player-Objekt bietet Zugriff auf alle Filme und Fenster, die mit ihm verwaltet werden, sowie auf alle verfügbaren Xtra-Erweiterungen.

Zum Erstellen eines Verweises auf das Player-Objekt können Sie die Top-Level-Eigenschaft `_player` verwenden.

- Weisen Sie `_player` einer Variablen zu.

```
-- Lingo syntax
objPlayer = _player

// JavaScript syntax
var objPlayer = _player;
```

- Verwenden Sie die Eigenschaft `_player` direkt.

```
-- Lingo syntax
_player.alert("The movie has ended.")

// JavaScript syntax
_player.alert("The movie has ended.");
```

Übersicht: Methoden des Player-Objekts

Methode	
alert()	getPref() (Player)
appMinimize()	installMenu
cursor()	open() (Player)
externalParamName()	quit()
externalParamValue()	setPref() (Player)
flushInputEvents()	windowPresent()

Übersicht: Eigenschaften des Player-Objekts

Eigenschaft	
activeWindow	netThrottleTicks
alertHook	organizationName
applicationName	productName
applicationPath	productVersion
currentSpriteNum	runMode
debugPlaybackEnabled	safePlayer
digitalVideoTimeScale	scriptingXtraList
disableImagingTransformation	searchCurrentFolder
emulateMultibuttonMouse	searchPathList
externalParamCount	serialNumber
frontWindow	sound (Player)
inlineImeEnabled	switchColorDepth
itemDelimiter	toolXtraList
lastClick	transitionXtraList
lastEvent	userName
activeWindow	window
lastKey	windowList
lastRoll	xtra
mediaXtraList	xtraList (Player)
netPresent	

Siehe auch

`_player`, `Cast Library`, `Member`, `Movie`, `Sprite`, `Window`

Sound

Das Soundobjekt von Direktor steuert die Audiowiedergabe in allen sechzehn verfügbaren Soundkanälen.

Alter Arbeitsablauf

Das Sound-Objekt besteht aus Sound Channel-Objekten, die einzelne Soundkanäle darstellen.

Zum Erstellen eines Verweises auf das Sound-Objekt können Sie die Top-Level-Eigenschaft `_sound` verwenden.

- Weisen Sie `_sound` einer Variablen zu.

```
-- Lingo syntax
objSound = _sound

// JavaScript syntax
var objSound = _sound;
```

- Nutzen Sie die Eigenschaft „_sound", um auf die Eigenschaft „_soundDevice“ des Soundobjekts zuzugreifen.

```
-- Lingo syntax
objDevice = _sound.soundDevice

// JavaScript syntax
var objDevice = _sound.soundDevice;
```

Übersicht: Methoden des Soundobjekts (Alter Arbeitsablauf)

Methode
beep()
channel() (Sound)

Übersicht: Eigenschaften des Soundobjekts (Alter Arbeitsablauf)

Eigenschaft
soundDevice
soundDeviceList
soundEnabled
soundKeepDevice
soundLevel
soundMixMedia

Siehe auch

_sound, Sound Channel

Neuer Arbeitsablauf

Soundobjekte werden zum Mixer hinzugefügt. Ein Mixer stellt eine Sammlung von Soundobjekten dar, in denen die Sounds einzeln oder alle zusammen wiedergegeben werden können. Sie können mithilfe des Mixers, mit dem das Soundobjekt verknüpft ist, einen Verweis zu diesem Soundobjekt erstellen.

❖ Weisen sie dem Soundobjekt eine Variable zu:

```
--- Lingo syntax
objSound = mixerobjectref.getSoundObject(soundobjectname)

// JavaScript syntax
var objSound = mixerobjectref.getSoundObject(soundobjectname);
```

Übersicht: Methoden des Soundobjekts (Neuer Arbeitsablauf)

Methode	
breakLoop (Soundobjekt)	Save (Soundobjekt)
moveTo	seek (Soundobjekt)
mute (Soundobjekt)	startSave (Soundobjekt)
pause (Soundobjekt)	stop (Soundobjekt)
play (Soundobjekt)	stopSave (Soundobjekt)
registerByteArrayCallback	unmute (Soundobjekt)
registerCuePointCallback	unregisterByteArrayCallback
registerEndOfSpoolCallback()	unregisterCuePointCallback
replaceMember	unregisterEndOfSpoolCallback()

Übersicht: Eigenschaften des Soundobjekts (Neuer Arbeitsablauf)

Eigenschaft	
bitDepth (Soundobject)	channelCount (Soundobjekt)
connectionStatus (Soundobjekt)	currentTime (Soundobjekt)
elapsedTime (Soundobjekt)	endTime (Soundobjekt)
filterList (Soundobjekt)	isSaving (Soundobjekt)
loopCount (Soundobjekt)	loopEndTime (Soundobjekt)
loopsRemaining (Soundobjekt)	loopStartTime (Soundobjekt)
member (Soundobjekt)	mixer
mostRecentCuePoint (Soundobjekt)	name (Soundobjekt)
percentStreamed (Soundobjekt)	panMatrix
playRate (Soundobject)	sampleCount (Soundobjekt)
sampleRate (Soundobjekt)	startTime (Soundobjekt)
status (Soundobjekt)	toChannels
useMatrix	volume (Soundobjekt)

Siehe auch

Mixer

Sound Channel

Repräsentiert einen einzelnen Soundkanal innerhalb des Sound-Objekts.

Acht Soundkanäle stehen zur Verfügung. Sie können ein Sound Channel-Objekt in einem Skript verwenden, um auf beliebige der acht Soundkanäle zuzugreifen und sie zu ändern.

Hinweis: *Nur die ersten zwei Soundkanäle können im Drehbuch der Director-Oberfläche geändert werden.*

Zum Erstellen eines Verweises auf ein Sound Channel-Objekt können Sie die Top-Level-Methode `sound()`, die Eigenschaft `sound` des Player-Objekts oder die Methode `channel()` des Sound-Objekts verwenden. Sie können Soundkanal 2 beispielsweise wie folgt referenzieren:

- Verwenden Sie die Top-Level-Methode `sound()`.

```
-- Lingo syntax
objSoundChannel = sound(2)

// JavaScript syntax
var objSoundChannel = sound(2);
```

- Verwenden Sie die Eigenschaft `sound` des Player-Objekts.

```
-- Lingo syntax
objSoundChannel = _player.sound[2]

// JavaScript syntax
var objSoundChannel = _player.sound[2];
```

- Verwenden Sie die Methode `channel()` des Sound-Objekts.

```
-- Lingo syntax
objSoundChannel = _sound.channel(2)

// JavaScript syntax
var objSoundChannel = _sound.channel(2);
```

Übersicht: Methoden des Sound Channel-Objekts

Methode	
breakLoop()	play() (Soundkanal)
fadeIn()	playFile()
fadeOut()	playNext() (Soundkanal)
fadeTo()	queue()
getPlayList()	rewind() (Soundkanal)
isBusy()	setPlayList()
pause() (Soundkanal)	stop() (Soundkanal)

Übersicht: Eigenschaften des Sound Channel-Objekts

Eigenschaft	
channelCount (Soundkanal)	member (Soundkanal)
elapsedTime	pan
endTime (Soundkanal)	sampleCount (Soundkanal)
loopCount	sampleRate (Soundkanal)

Eigenschaft	
loopEndTime (Soundkanal)	startTime (Sound Channel)
loopsRemaining	status
loopStartTime	volume (Soundkanal)

Siehe auch

`channel() (Sound)`, `sound (Player)`, `sound()`, `Sound`

Sprite

Steht für ein Vorkommen eines Darstellers in einem Sprite-Kanal des Drehbuchs.

Ein Sprite-Objekt umfasst einen Sprite-Einschluss, d. h., einen Bildbereich in einem bestimmten Sprite-Kanal. Ein Sprite Channel-Objekt entspricht einem ganzen Sprite-Kanal, unabhängig davon, wie viele Sprites dieser enthält.

Hinweis: *Der Zugriff auf Flash®-Darstellerkomponenten (Flash-Sprites), die sich auf der Bühne befinden, aber unsichtbar sind, ist nur über das Member-Objekt möglich. Wenn Sie ein Sprite-Objekt für den Zugriff auf ein unsichtbares Flash-Sprite verwenden, verursacht dies einen Fehler.*

Sprites können anhand ihrer Nummer oder ihres Namens referenziert werden.

- Wenn Sie anhand der Nummer auf ein Sprite Bezug nehmen, sucht Director alle Sprites im aktuellen Bild des Drehbuchs, wobei beim Kanal mit der niedrigsten Nummer begonnen wird, und ruft die Sprite-Daten ab, wenn es das Sprite mit der angegebenen Nummer findet. Diese Methode ist schneller als der Sprite-Verweis anhand des Namens. Da Director Verweise auf Sprite-Nummern in Skripts jedoch nicht automatisch aktualisiert, wird ein Nummernverweis auf ein Sprite, das auf der Bühne an eine andere Position verschoben wurde, ungültig.
- Bei einem Sprite-Verweis anhand des Namens sucht Director alle Sprites im aktuellen Bild des Drehbuchs, wobei beim Kanal mit der niedrigsten Nummer begonnen wird, und ruft die Sprite-Daten ab, wenn es das Sprite mit dem angegebenen Namen findet. Diese Methode ist zeitaufwändiger als der Sprite-Verweis anhand der Nummer. Dies gilt besonders bei Verweisen auf umfangreiche Filme mit vielen Besetzungsbibliotheken, Darstellern und Sprites. Bei Namensverweisen auf Sprites bleiben Verweise jedoch intakt, selbst wenn das Sprite auf der Bühne an eine andere Position verschoben wird.

Zum Erstellen eines Verweises auf ein Sprite-Objekt können Sie die Top-Level-Funktion `sprite()`, die Eigenschaft `sprite` des Movie-Objekts oder die Eigenschaft `sprite` des Sprite Channel-Objekts verwenden.

- Verwenden Sie die Top-Level-Funktion `sprite()`.

```
-- Lingo syntax
objSprite = sprite(1)

// JavaScript syntax
var objSprite = sprite(1);
```

- Verwenden Sie die Eigenschaft `sprite` des Movie-Objekts.

```
-- Lingo syntax
objSprite = _movie.sprite["willowTree"]

// JavaScript syntax
var objSprite = _movie.sprite["willowTree"];
```

- Verwenden Sie die Eigenschaft `sprite` des Sprite Channel-Objekts.

```
-- Lingo syntax
objSprite = channel(3).sprite

// JavaScript syntax
var objSprite = channel(3).sprite;
```

Sie können eine Referenz auf ein Sprite-Objekt verwenden, um auf den Darsteller zuzugreifen, aus dem das Sprite erstellt wurde. Alle Änderungen an dem Darsteller, aus dem ein Sprite erstellt wurde, werden auch im Sprite widergespiegelt. Das folgende Beispiel zeigt, wie Sie den Text eines Textdarstellers ändern, aus dem Sprite 5 erstellt wurde. Diese Änderung am Darsteller findet sich auch in Sprite 5 wieder.

```
-- Lingo syntax
labelText = sprite(5)
labelText.member.text = "Weeping Willow"

// JavaScript syntax
var labelText = sprite(5);
labelText.member.text = "Weeping Willow";
```

Übersicht: Eigenschaften des Sprite-Objekts

Eigenschaft	
backColor	locV
blend (Sprite)	locZ
bottom	member (Sprite)
constraint	name (Sprite)
cursor	quad
editable	rect (Sprite)
endFrame	right
filterlist	rotation
flipH	skew
flipV	spriteNum
foreColor	startFrame
height	top
ink	width
left	
locH	

Siehe auch

Cast Library, Member, Movie, Player, sprite (Film), sprite (Sprite-Kanal), sprite(), Sprite Channel, Window

Sprite Channel

Steht für einen einzelnen Sprite-Kanal im Drehbuch.

Ein Sprite-Objekt umfasst einen Sprite-Einschluss, d. h., einen Bildbereich in einem bestimmten Sprite-Kanal. Ein Sprite Channel-Objekt entspricht einem ganzen Sprite-Kanal, unabhängig davon, wie viele Sprites dieser enthält.

Sprite-Kanäle werden standardmäßig vom Drehbuch gesteuert. Mithilfe des Sprite Channel-Objekts können Sie die Steuerung eines Sprite-Kanals während der Drehbuchaufzeichnung an ein Skript übergeben.

Sprite-Kanäle können anhand ihrer Nummer oder ihres Namens referenziert werden.

- Wenn Sie anhand der Nummer auf einen Sprite-Kanal Bezug nehmen, können Sie direkt auf den Kanal zugreifen. Diese Methode ist schneller als der Namensverweis auf den Sprite-Kanal. Da Director Verweise auf Sprite-Kanalnummern in Skripts jedoch nicht automatisch aktualisiert, wird ein Nummernverweis auf einen Sprite-Kanal, der im Drehbuch an eine andere Position verschoben wurde, ungültig.
- Bei einem Sprite-Kanalverweis anhand des Namens durchsucht Director alle Kanäle, wobei beim Kanal mit der niedrigsten Nummer begonnen wird, und ruft die Daten des Sprite-Kanals ab, wenn es den Sprite-Kanal mit dem angegebenen Namen findet. Diese Methode ist zeitaufwändiger als der Nummernverweis auf einen Sprite-Kanal. Dies gilt besonders bei Verweisen auf umfangreiche Filme mit vielen Besetzungsbibliotheken, Darstellern und Sprites. Bei Namensverweisen auf Sprite-Kanäle bleiben Verweise jedoch intakt, selbst wenn der Sprite-Kanal im Drehbuch an eine andere Position verschoben wird.

Zum Erstellen eines Verweises auf ein Sprite Channel-Objekt können Sie die Top-Level-Methode `channel()` verwenden und den Kanal entweder anhand der Nummer oder des Namens referenzieren.

```
-- Lingo syntax
objSpriteChannel = channel(2) -- numbered reference
objSpriteChannel = channel("background") -- named reference

// JavaScript syntax
var objSpriteChannel = channel(2); // numbered reference
var objSpriteChannel = channel("background"); // named reference
```

Mit einem Verweis auf ein Sprite Channel-Objekt können Sie auf das Sprite zuzugreifen, das gerade in einem bestimmten Sprite-Kanal verwendet wird. Das folgende Beispiel zeigt, wie Sie auf die Hintergrundfarbe des Sprites zugreifen, das momentan in Sprite-Kanal 2 verwendet wird.

```
-- Lingo syntax
labelSprite = channel(2).sprite.backColor

// JavaScript syntax
var labelSprite = channel(2).sprite.backColor;
```

Übersicht: Methoden des Sprite Channel-Objekts

Methode
makeScriptedSprite()
removeScriptedSprite()

Übersicht: Eigenschaften des Sprite Channel-Objekts

Eigenschaft
name (Sprite-Kanal)
number (Sprite-Kanal)
scripted
sprite (Sprite-Kanal)

Siehe auch

Cast Library, channel() (Top-Level), Member, Movie, Player, Sprite, Window

System

Ermöglicht den Zugriff auf System- und Umgebungsinformationen, einschließlich auf Systemebene ablaufender Methoden.

Zum Erstellen eines Verweises auf das System-Objekt können Sie die Top-Level-Eigenschaft `_system` verwenden.

- Weisen Sie `_system` einer Variablen zu.

```
-- Lingo syntax
objSystem = _system

// JavaScript syntax
var objSystem = _system;
```

- Verwenden Sie die Eigenschaft „_system“ direkt.

```
-- Lingo syntax
sysDate = _system.date()

// JavaScript syntax
var sysDate = _system.date();
```

Übersicht: Methoden des System-Objekts

Methode
date() (System)
getInstalledCharSets
getSystemCharSet
isCharSetInstalled
restart()
shutDown()
time() (System)

Übersicht: Eigenschaften des System-Objekts

Eigenschaft
colorDepth
deskTopRectList
environmentPropList
milliseconds

Siehe auch

_system

Window

Entspricht einem Fenster, in dem ein Film wiedergegeben wird, einschließlich des Bühnenfensters und etwaiger MIAW-Fenster, die verwendet werden.

Zum Erstellen eines Verweises auf ein Window-Objekt können Sie die Top-Level-Funktion `window()` oder die Eigenschaft `window` bzw. `windowList` des Player-Objekts verwenden.

- Verwenden Sie die Top-Level-Methode `window()`.

```
-- Lingo syntax
objWindow = window("Sun")

// JavaScript syntax
var objWindow = window("Sun");
```

- Verwenden Sie die Eigenschaft `window` des Player-Objekts.

```
-- Lingo syntax
objWindow = _player.window["Sun"]

// JavaScript syntax
var objWindow = _player.window["Sun"];
```

- Verwenden Sie die Eigenschaft `windowList` des Player-Objekts.

```
-- Lingo syntax
objWindow = _player.windowList[1]

// JavaScript syntax
var objWindow = _player.windowList[1];
```

Hinweis: *Wenn Sie mit der Top-Level-Funktion `window()` oder der Eigenschaft `window` des Player-Objekts einen Namensverweis auf ein Fenster erstellen, erfolgt die Erstellung dieser Referenz nur, wenn ein Fenster mit diesem Namen vorhanden ist. Ist kein Fenster mit diesem Namen vorhanden, enthält der Verweis die Angabe `VOID` (Lingo) bzw. `null` (JavaScript-Syntax).*

Die Eigenschaft `scriptExecutionStyle` des Movie-Objekts wird standardmäßig auf den Wert 10 eingestellt, und `windowType` wird standardmäßig zugunsten der Eigenschaftslisten `appearanceOptions` und `titlebarOptions` verworfen. Wird `scriptExecutionStyle` auf den Wert 9 eingestellt, hat `windowType` die volle Gültigkeit.

Übersicht: Methoden des Window-Objekts

Methode	
close()	moveToBack()
forget() (Fenster)	moveToFront()
maximize()	open() (Fenster)
mergeProps()	restore()
minimize()	

Übersicht: Eigenschaften des Window-Objekts

Eigenschaft	
appearanceOptions	resizable
bgColor (Fenster)	sizeState
dockingEnabled	sourceRect
drawRect	title (Fenster)
fileName (Fenster)	titlebarOptions
image (Fenster)	type (Fenster)
movie	visible
name	windowBehind
picture (Fenster)	windowInFront
rect (Fenster)	

Siehe auch

`Cast Library`, `Member`, `Movie`, `Player`, `Sprite`, `window()`, `window`, `windowList`

Kapitel 6: Medientypen

Die Medientypen in Director® ermöglichen den Zugriff auf die Funktionalität der verschiedenen Medientypen wie RealMedia®, MP4Media, FLV, DVD, animiertes GIF usw., die als Darsteller zu Filmen hinzugefügt werden.

Genau genommen handelt es sich bei Medientypen nicht um Objekte, sondern um Darsteller mit einem bestimmten Medientyp. Wenn Sie einen Medientyp als Darsteller zu einem Film hinzufügen, erbt er nicht nur die Funktionalität des Core-Objekts „Member“, sondern erweitert das Member-Objekt um zusätzliche Funktionen, die nur für bestimmte Medientypen zur Verfügung stehen. Ein RealMedia-Darsteller hat beispielsweise Zugriff auf die Methoden und Eigenschaften des Member-Objekts, verfügt jedoch auch über zusätzliche, für RealMedia spezifische Methoden und Eigenschaften. Bei allen anderen Medientypen verhält sich dies genauso.

Das Diagramm unter „Objektmodelldiagramme“ auf Seite 47 zeigt, wie die Medientypen von Darstellern zueinander und zu anderen Objekten in Director in Bezug stehen.

Animiertes GIF

Steht für einen animierten GIF-Darsteller.

Mit der Methode `newMember()` des Movie-Objekts können Sie einen animierten GIF-Darsteller zu einem Film hinzufügen.

```
-- Lingo syntax
_movie.newMember(#animgif)

// JavaScript syntax
_movie.newMember("animgif");
```

Einige der folgenden Methoden oder Eigenschaften gelten evtl. nur für Sprites, die aus einem animierten GIF-Darsteller erstellt werden.

Übersicht: Methoden des Medientyps „Animated GIF“

Methode
resume()
rewind() (Animiertes GIF, Flash)

Übersicht: Eigenschaften des Medientyps „Animated GIF“

Eigenschaft
directToStage
frameRate
linked
path (Film)
playBackMode

Siehe auch

`Member`

Bitmap

Steht für einen Bitmapdarsteller.

Mit Bitmap-Grafikobjekten können Sie einfache Operationen durchführen, die sich auf den Inhalt eines ganzen Bitmapdarstellers auswirken, wie beispielsweise das Ändern der Hintergrund- und Vordergrundfarben des Darstellers oder feinste Änderungen an den Pixeln einer Grafik etwa durch Zuschneiden, Zeichnen oder Kopieren von Pixeln.

Mit der Methode `newMember()` des Movie-Objekts können Sie einen Bitmapdarsteller zu einem Film hinzufügen.

```
-- Lingo syntax
_movie.newMember(#bitmap)

// JavaScript syntax
_movie.newMember("bitmap");
```

Einige der folgenden Methoden oder Eigenschaften gelten evtl. nur für Sprites, die aus einem Bitmapdarsteller erstellt werden.

Übersicht: Methoden des Medientyps „Bitmap“

Methode
crop() (Grafik)
pictureP()

Übersicht: Eigenschaften des Medientyps „Bitmap“

Eigenschaft	
alphaThreshold	imageCompression
backColor	imageQuality
blend (Sprite)	palette
depth (Bitmap)	picture (Darsteller)
dither	rect (Grafik)
foreColor	trimWhiteSpace
image (Grafik)	useAlpha

Siehe auch

`Member`

Button

Steht für einen Schaltflächen- oder Kontrollkästchendarsteller.

Mit der Methode `newMember()` des Movie-Objekts können Sie einen Schaltflächendarsteller zu einem Film hinzufügen.

```
-- Lingo syntax
_movie.newMember(#button)

// JavaScript syntax
_movie.newMember("button");
```

Übersicht: Eigenschaften des Medientyps „Button“

Eigenschaft
hilite

Siehe auch

`Member`

ByteArray

ByteArray, Darsteller

Der Darsteller `#byteArray` lässt sich in Director, Shockwave und Projektor nutzen. Dieser Nur-Skript-Darsteller hat begrenzte Unterstützung für die Benutzerschnittstelle.

Erstellen eines ByteArray-Darstellers über die Benutzerschnittstelle Wählen Sie „Einfügen“ > „Medien-Elemente“ > „Byte Array “, um einen `ByteArray`-Darsteller einzufügen.

Erstellen eines ByteArray-Darstellers mit Lingo Verwenden Sie die Methode `new`, um einen `ByteArray`-Darsteller zu erstellen.

Syntax

```
m = new(#byteArray)
```

Weitere Informationen hierzu finden Sie unter „ByteArray“ auf Seite 155.

Color Palette

Steht für die Farbpalette, die einem Bitmapdarsteller zugeordnet ist.

Farbpalettendarsteller verfügen nicht über Methoden oder Eigenschaften, auf die direkt zugegriffen werden kann. Die folgenden Methoden und Eigenschaften werden Farbpaletten lediglich zugeordnet.

Mit der Methode `newMember()` des Movie-Objekts können Sie einen Farbpalettendarsteller zu einem Film hinzufügen.

```
-- Lingo syntax
_movie.newMember(#palette)

// JavaScript syntax
_movie.newMember("palette");
```

Sie können einen Bitmapdarsteller mit einem Farbpalettendarsteller verknüpfen, indem Sie die Eigenschaft `palette` des Bitmapdarstellers festlegen. Im folgenden Beispiel wird für die Eigenschaft `palette` des Bitmapdarstellers `bmpMember` der Farbpalettendarsteller `colorPaletteMember` festgelegt. Der Wert der Eigenschaft `palette` steht dabei für die Nummer des Farbpalettendarstellers.

```
-- Lingo syntax
member("bmpMember").palette = member("colorPaletteMember")

// JavaScript syntax
member("bmpMember").palette = member("colorPaletteMember");
```

Wenn ein Bitmapdarsteller mit einem Farbpalettendarsteller verknüpft wurde, kann der Farbpalettendarsteller erst gelöscht werden, nachdem seine Verknüpfung mit dem Bitmapdarsteller aufgehoben wurde.

Übersicht: Methoden des Medientyps „Color Palette“

Methode
color()

Übersicht: Eigenschaften des Medientyps „Color Palette“

Eigenschaft
depth (Bitmap)
palette
paletteMapping

Siehe auch

Bitmap, Member, palette

Cursor

Steht für einen Cursordarsteller.

Mit der Methode `newMember()` des Movie-Objekts können Sie einen Cursordarsteller zu einem Film hinzufügen.

```
-- Lingo syntax
_movie.newMember(#cursor)

// JavaScript syntax
_movie.newMember("cursor");
```

Übersicht: Eigenschaften des Medientyps „Cursor“

Eigenschaft
castMemberList
cursorSize
hotSpot
interval

Siehe auch

Member

DVD

Steht für einen DVD-Darsteller.

Mit der Methode `newMember()` des Movie-Objekts können Sie einen DVD-Darsteller zu einem Film hinzufügen.

```
-- Lingo syntax
_movie.newMember(#dvd)

// JavaScript syntax
_movie.newMember("dvd");
```

Einige der folgenden Methoden oder Eigenschaften gelten evtl. nur für Sprites, die aus einem DVD-Darsteller erstellt werden.

Übersicht: Ereignisse des Medientyps „DVD“

Die folgenden DVD-Ereignisse werden immer von einer `DVDeventNotification`-Ereignisprozedur verarbeitet. Tritt eines dieser Ereignisse ein, empfängt die `DVDeventNotification`-Ereignisprozedur das Ereignis als Parameter. Einige dieser Ereignisse enthalten auch zusätzliche Informationen, die in Form eines zweiten oder dritten Parameters an `DVDeventNotification` übergeben werden. Weitere Informationen über die Verwendung der folgenden Ereignisse mit der `DVDeventNotification`-Ereignisprozedur finden Sie unter on DVDeventNotification.

Ereignis	
on DVDeventNotification	noFirstPlayChain
audioStreamChange	parentalLevelChange
buttonChange	playbackStopped
chapterAutoStop	playPeriodAutoStop
chapterStart	rateChange
diskEjected	stillOff
diskInserted	stillOn

Ereignis	
domainChange	titleChange
error	UOPchange
karaokeMode	warning

Übersicht: Methoden des Medientyps „DVD“

Methode	
activateAtLoc()	rootMenu()
activateButton()	selectAtLoc()
frameStep()	selectButton()
chapterCount()	selectButtonRelative()
pause() (DVD)	stop() (DVD)
play() (DVD)	subPictureType()
returnToTitle()	titleMenu()

Übersicht: Eigenschaften des Medientyps „DVD“

Eigenschaft	
angle (DVD)	duration (DVD)
angleCount	folder
aspectRatio	frameRate (DVD)
audio (DVD)	fullScreen
audioChannelCount	mediaStatus (DVD)
audioExtension	playRate (DVD)
audioFormat	resolution (DVD)
audioSampleRate	selectedButton
audioStream	startTimeList
audioStreamCount	stopTimeList
buttonCount	subPicture
chapter	subPictureCount
chapterCount	title (DVD)
closedCaptions	titleCount
currentTime (DVD)	videoFormat
domain	volume (DVD)

Siehe auch

`Member`

Feld

Steht für einen Felddarsteller.

Mit der Methode `newMember()` des Movie-Objekts können Sie einen Felddarsteller zu einem Film hinzufügen.

```
-- Lingo syntax
_movie.newMember(#field)

// JavaScript syntax
_movie.newMember("field");
```

Übersicht: Methoden des Medientyps „Field“

Methode	
charPosToLoc()	pointToItem()
lineHeight()	pointToLine()
linePosToLocV()	pointToParagraph()
locToCharPos()	pointToWord()
locVToLinePos()	scrollByLine()
pointToChar()	scrollByPage()

Übersicht: Eigenschaften des Medientyps „Field“

Eigenschaft	
alignment	fontStyle
autoTab	lineCount
border	margin
boxDropShadow	pageHeight
boxType	scrollTop
dropShadow	selEnd
editable	selStart
font	text
fontSize	wordWrap

Siehe auch

Member

Film Loop

Steht für einen Filmschleifendarsteller.

Mit der Methode `newMember()` des Movie-Objekts können Sie einen Filmschleifendarsteller zu einem Film hinzufügen.

```
-- Lingo syntax
_movie.newMember(#filmloop)

// JavaScript syntax
_movie.newMember("filmloop");
```

Übersicht: Eigenschaften des Medientyps „Film Loop“

Eigenschaft
media
regPoint

Siehe auch

`Member`

Flash Component

Steht für eine Adobe® Flash®-Komponente, die in einen Darsteller oder ein Sprite mit Flash-Inhalten eingebettet ist.

Flash-Komponenten stellen vordefinierte Funktionalität als Erweiterung der bestehenden Funktionen von Darstellern oder Sprites mit Flash-Inhalten bereit. Sie werden von der Director-Entwicklungscommunity erstellt und unterstützt.

Director unterstützt folgende Flash-Komponenten:

Flash-Komponente	Beschreibung
Button	Eine rechteckige Benutzeroberflächen-Schaltfläche, deren Größe angepasst werden kann
CheckBox	Ein grundlegender Bestandteil von Formularen und Webanwendungen, der immer dann verwendet wird, wenn eine Reihe von `true`- bzw. `false`-Werten erfasst werden muss, die sich nicht gegenseitig ausschließen
DateChooser	Ein Kalender, in dem Benutzer ein Datum auswählen können
Label	Eine einzige Textzeile
List	Ein rollfähiges Listenfeld mit Einzel- oder Mehrfachauswahlmöglichkeit
NumericStepper	Ermöglicht Benutzern, eine sortierte Zahlengruppe schrittweise durchlaufen
RadioButton	Ein grundlegender Bestandteil von Formularen und Webanwendungen, der überall dort eingesetzt wird, wo Sie vom Benutzer eine einzelne Auswahl aus einer Optionsgruppe erzwingen möchten
ScrollPane	Zeigt Movieclips, JPEG-Dateien und SWF-Dateien in einem rollbaren Bereich an
TextArea	Ein mehrzeiliges Textfeld
TextInput	Eine einzeilige Komponente, die das systemeigene ActionScript-Objekt „TextField“ umschließt
Tree	Ermöglicht Benutzern, hierarchische Daten anzuzeigen

Neben der komponentenspezifischen Funktionalität haben Flash-Komponenten Zugriff auf dieselben APIs wie normale Flash-Darsteller bzw. -Sprites. Weitere Informationen über die Verwendung dieser Flash-Komponenten finden Sie in den Themen des Director-Benutzerhandbuchs im Director-Hilfefenster.

Mit der Methode `newMember()` des Movie-Objekts können Sie einen Flash-Komponentendarsteller zu einem Film hinzufügen.

```
-- Lingo syntax
_movie.newMember(#flashcomponent)

// JavaScript syntax
_movie.newMember("flashcomponent");
```

Siehe auch

Flash Movie, Member

Flash Movie

Steht für einen Darsteller oder ein Sprite, der bzw. das Flash-Inhalte enthält.

Mit der Methode `newMember()` des Movie-Objekts können Sie einen Flash-Filmdarsteller zu einem Film hinzufügen.

```
-- Lingo syntax
_movie.newMember(#flash)

// JavaScript syntax
_movie.newMember("flash");
```

Flash-Filmdarsteller oder -sprites können auch Flash-Komponenten enthalten. Flash-Komponenten stellen vordefinierte Funktionalität als Erweiterung der bestehenden Funktionen von Flash-Filmdarstellern oder -sprites bereit. Weitere Informationen über die von Director unterstützten Flash-Komponenten finden Sie unter „Flash Component“ auf Seite 127.

Einige der folgenden Methoden oder Eigenschaften gelten evtl. nur für Sprites, die aus einem Flash-Filmdarsteller erstellt werden.

Übersicht: Methoden des Medientyps „Flash Movie“

Methode	
callFrame()	printAsBitmap()
clearAsObjects()	rewind() (Animiertes GIF, Flash)
clearError()	setCallback()
findLabel()	setFlashProperty()
flashToStage()	settingsPanel()
getFlashProperty()	setVariable()
getVariable()	showProps()
goToFrame()	stageToFlash()
hitTest()	stop() (Flash)

Methode	
hold()	stream()
newObject()	tellTarget()
print()	

Übersicht: Eigenschaften des Medientyps „Flash Movie“

Eigenschaft	
actionsEnabled	originPoint
broadcastProps	originV
bufferSize	playBackMode
buttonsEnabled	playing
bytesStreamed	posterFrame
centerRegPoint	quality
clickMode	rotation
defaultRect	scale (Darsteller)
defaultRectMode	scaleMode
eventPassMode	sound (Darsteller)
fixedRate	static
flashRect	streamMode
frameCount	streamSize
imageEnabled	viewH
linked	viewPoint
mouseOverButton	viewScale
originH	viewV
originMode	

Siehe auch

`Flash Component`, `Member`

Schrift

Steht für einen Schriftdarsteller.

Mit der Methode `newMember()` des Movie-Objekts können Sie einen Schriftdarsteller zu einem Film hinzufügen.

```
-- Lingo syntax
_movie.newMember(#font)

// JavaScript syntax
_movie.newMember("font");
```

Übersicht: Eigenschaften des Medientyps „Font“

Eigenschaft
bitmapSizes
characterSet
fontStyle
originalFont
recordFont

Siehe auch

Member

Linked Movie

Steht für einen verknüpften Filmdarsteller.

Mit der Methode `newMember()` des Movie-Objekts können Sie einen verknüpften Filmdarsteller zu einem Film hinzufügen.

```
-- Lingo syntax
_movie.newMember(#movie)

// JavaScript syntax
_movie.newMember("movie");
```

Übersicht: Eigenschaften des Medientyps „Linked Movie“

Eigenschaft
scriptsEnabled

Siehe auch

Member

MP4Media/FLV

Director unterstützt die H.264-kodierten Dateiformate MP4, F4V und FLV.

Übersicht der Eigenschaften für MP4Media/FLV-Ressourcen

Eigenschaft	
audio (MP4Media/FLV)	connectionStatus (MP4Media/FLV)
currentTime (MP4Media/FLV)	directToStage (MP4Media/FLV)
duration (MP4Media/FLV)	frameRate (MP4Media/FLV)
fileName (MP4Media/FLV)	height (MP4Media/FLV)
image (MP4Media/FLV)	isPlayable (MP4Media/FLV)
loop (MP4Media/FLV)	percentStreamed (MP4Media/FLV)
trackInfo	video (MP4Media/FLV)
volume (MP4Media/FLV)	width (MP4Media/FLV)

Übersicht der Eigenschaften für MP4Media/FLV-Sprites

Eigenschaft	
audio (MP4Media/FLV)	connectionStatus (MP4Media/FLV)
currentTime (MP4Media/FLV)	directToStage (MP4Media/FLV)
image (MP4Media/FLV)	isPlayable (MP4Media/FLV)
loop (MP4Media/FLV)	mediaStatus (MP4Media/FLV)
mixer	pausedAtStart (MP4Media/FLV)
percentStreamed (MP4Media/FLV)	video (MP4Media/FLV)
volume (MP4Media/FLV)	

Übersicht der Methoden für MP4Media/FLV-Ressourcen

Methode	
enableSoundTrack(trackNum)	pause (MP4Media/FLV)
play() (MP4Media/FLV)	seek(mSec) (MP4Media/FLV)
stop() (MP4Media/FLV)	

Übersicht der Methoden für MP4Media/FLV-Sprites

Methode	
pause (MP4Media/FLV)	play() (MP4Media/FLV)
rewind() (MP4Media/FLV)	seek(mSec) (MP4Media/FLV)
stop() (MP4Media/FLV)	

QuickTime

Steht für einen QuickTime®-Darsteller.

Mit der Methode `newMember()` des Movie-Objekts können Sie einen QuickTime-Darsteller zu einem Film hinzufügen.

```
-- Lingo syntax
_movie.newMember(#quicktimemedia)

// JavaScript syntax
_movie.newMember("quicktimemedia");
```

Einige der folgenden Methoden oder Eigenschaften gelten evtl. nur für Sprites, die aus einem QuickTime-Darsteller erstellt werden.

Übersicht: Methoden des Medientyps „QuickTime“

Methode	
enableHotSpot()	qtRegisterAccessKey()
getHotSpotRect()	qtUnRegisterAccessKey()
nudge()	setTrackEnabled()
ptToHotSpotID()	swing()
QuickTimeVersion()	

Übersicht: Eigenschaften des Medientyps „QuickTime“

Eigenschaft	
audio (RealMedia)	scale (Darsteller)
currentTime (QuickTime, AVI)	staticQuality
fieldOfView	tilt
hotSpotEnterCallback	trackCount (Darsteller)
hotSpotExitCallback	trackCount (Sprite)
invertMask	trackEnabled
isVRMovie	trackNextKeyTime
loopBounds	trackNextSampleTime
mask	trackPreviousKeyTime
motionQuality	trackPreviousSampleTime
mouseLevel	trackStartTime (Darsteller)
node	trackStartTime (Sprite)
nodeEnterCallback	trackStopTime (Darsteller)
nodeExitCallback	trackStopTime (Sprite)

Eigenschaft	
nodeType	trackText
pan (QTVR-Eigenschaft)	trackType (Darsteller)
percentStreamed (Darsteller)	trackType (Sprite)
playRate (QuickTime, AVI, MP4, FLV)	translation
preLoad (Darsteller)	triggerCallback
rotation	warpMode

Siehe auch

`Member`

RealMedia

Steht für einen RealMedia-Darsteller.

Mit der Methode `newMember()` des Movie-Objekts können Sie einen RealMedia-Darsteller zu einem Film hinzufügen.

```
-- Lingo syntax
_movie.newMember(#realmedia)

// JavaScript syntax
_movie.newMember("realmedia");
```

Einige der folgenden Methoden oder Eigenschaften gelten evtl. nur für Sprites, die aus einem RealMedia-Darsteller erstellt werden.

Übersicht: Methoden des Medientyps „RealMedia“

Methode
pause() (RealMedia, SWA, Windows Media)
play() (RealMedia, SWA, Windows Media)
realPlayerNativeAudio()
realPlayerPromptToInstall()
realPlayerVersion()
seek()
stop() (RealMedia, SWA, Windows Media)

Übersicht: Eigenschaften des Medientyps „RealMedia“

Eigenschaft	
audio (RealMedia)	password
currentTime (RealMedia)	pausedAtStart (RealMedia, Windows Media)
displayRealLogo	percentBuffered
duration (RealMedia, SWA)	soundChannel (RealMedia)
image (RealMedia)	state (RealMedia)
lastError	userName (RealMedia)
mediaStatus (RealMedia, Windows Media)	video (RealMedia, Windows Media)

Siehe auch

`Member`

Shockwave 3D

Steht für einen Adobe® Shockwave®-3D-Darsteller.

Shockwave 3D-Darsteller (auch nur 3D-Darsteller genannt) unterscheiden sich von anderen Darstellern insofern, als sie eine komplette 3D-Welt enthalten. Eine 3D-Welt enthält eine Reihe von Objekten, die es nur bei 3D-Darstellern gibt und Ihnen ermöglichen, 3D-Funktionalität zu einem Film hinzuzufügen.

Mit der Methode `newMember()` des Movie-Objekts können Sie einen 3D-Darsteller zu einem Film hinzufügen.

```
-- Lingo syntax
_movie.newMember(#shockwave3d)

// JavaScript syntax
_movie.newMember("shockwave3d");
```

Weitere Informationen über die Objekte und APIs, die 3D-Darstellern zur Verfügung stehen, finden Sie unter „3D-Objekte“ auf Seite 166.

Siehe auch

`Member`

Shockwave Audio

Steht für einen Shockwave Audio-Darsteller.

Mit der Methode `newMember()` des Movie-Objekts können Sie einen Shockwave Audio-Darsteller zu einem Film hinzufügen.

```
-- Lingo syntax
_movie.newMember(#swa)

// JavaScript syntax
_movie.newMember("swa");
```

Übersicht: Ereignisse des Medientyps „Shockwave Audio“

Ereignis
on cuePassed

Übersicht: Methoden des Medientyps „Shockwave Audio“

Methode
getError() (Flash, SWA)
getErrorString()
isPastCuePoint()
pause() (RealMedia, SWA, Windows Media)
play() (RealMedia, SWA, Windows Media)
preLoadBuffer()
stop() (RealMedia, SWA, Windows Media)

Übersicht: Eigenschaften des Medientyps „Shockwave Audio“

Eigenschaft	
bitRate	percentStreamed (Darsteller)
bitsPerSample	preLoadTime
channelCount (Soundkanal)	sampleRate (Soundkanal)
copyrightInfo (SWA)	sampleSize
cuePointNames	soundChannel (SWA)
cuePointTimes	state (Flash, SWA)
duration (RealMedia, SWA)	streamName
loop (Darsteller)	URL
mostRecentCuePoint	volume (Darsteller)
numChannels	

Siehe auch

Member

Sound

Steht für einen Darsteller, der zum Speichern und Referenzieren von Soundsamples verwendet wird.

Soundsamples werden von den Core-Objekten „Sound“ und „Sound Channel“ gesteuert. Sounddarsteller verfügen nicht über eigene APIs, sondern verwenden die APIs der Sound- und Sound Channel-Objekten, die ihr Verhalten steuern.

Mit der Methode `newMember()` des Movie-Objekts können Sie einen Sounddarsteller zu einem Film hinzufügen.

```
-- Lingo syntax
_movie.newMember(#sound)

// JavaScript syntax
_movie.newMember("sound");
```

Weitere Informationen über die Objekte und APIs zur Steuerung von Soundsamples finden Sie unter „Sound“ auf Seite 110 und „Sound Channel“ auf Seite 112.

Siehe auch

`Member`

Text

Steht für einen Textdarsteller.

Mit der Methode `newMember()` des Movie-Objekts können Sie einen Textdarsteller zu einem Film hinzufügen.

```
-- Lingo syntax
_movie.newMember(#text)

// JavaScript syntax
_movie.newMember("text");
```

Übersicht: Ereignisse des Medientyps „Text“

Ereignis
on hyperlinkClicked

Übersicht: Methoden des Medientyps „Text“

Methode
count()
pointInHyperlink()
pointToChar()
pointToItem()
pointToLine()
pointToParagraph()
pointToWord()

Übersicht: Eigenschaften des Medientyps „Text“

Eigenschaft	
antiAlias	hyperlink
antiAliasThreshold	hyperlinkRange
bottomSpacing	hyperlinks
charSpacing	hyperlinkState
firstIndent	kerning
fixedLineSpace	kerningThreshold
font	RTF
fontStyle	selectedText
HTML	useHypertextStyles

Siehe auch

`Member`

Vector Shape

Steht für einen Vektorformdarsteller.

Mit der Methode `newMember()` des Movie-Objekts können Sie einen Vektorformdarsteller zu einem Film hinzufügen.

```
-- Lingo syntax
_movie.newMember(#vectorshape)

// JavaScript syntax
_movie.newMember("vectorshape");
```

Einige der folgenden Methoden oder Eigenschaften gelten evtl. nur für Sprites, die aus einem Vektorformdarsteller erstellt werden.

Übersicht: Methoden des Medientyps „Vector Shape“

Methode
addVertex()
deleteVertex()
moveVertex()
moveVertexHandle()
newCurve()
showProps()

Übersicht: Eigenschaften des Medientyps „Vector Shape“

Eigenschaft	
antiAlias	imageEnabled
backgroundColor	originH
broadcastProps	originMode
centerRegPoint	originPoint
closed	originV
curve	regPointVertex
defaultRect	scale (Darsteller)
defaultRectMode	scaleMode
endColor	strokeColor
fillColor	strokeWidth
fillCycles	vertex
fillDirection	vertexList
fillMode	viewH
fillOffset	viewPoint
fillScale	viewScale
flashRect	viewV
gradientType	

Siehe auch

`Member`

Windows Media

Steht für einen Windows Media®-Darsteller.

Mit der Methode `newMember()` des Movie-Objekts können Sie einen Windows Media-Darsteller zu einem Film hinzufügen.

```
-- Lingo syntax
_movie.newMember(#windowsmedia)

// JavaScript syntax
_movie.newMember("windowsmedia");
```

Einige der folgenden Methoden oder Eigenschaften gelten evtl. nur für Sprites, die aus einem Windows Media-Darsteller erstellt werden.

Übersicht: Methoden des Medientyps „Windows Media“

Methode
pause() (RealMedia, SWA, Windows Media)
play() (RealMedia, SWA, Windows Media)
rewind() (Windows Media)
stop() (RealMedia, SWA, Windows Media)

Übersicht: Eigenschaften des Medientyps „Windows Media“

Eigenschaft	
audio (Windows Media)	pausedAtStart (RealMedia, Windows Media)
directToStage	playRate (Windows Media)
duration (Darsteller)	video (RealMedia, Windows Media)
height	volume (Windows Media)
loop (Windows Media)	width
mediaStatus (RealMedia, Windows Media)	

Siehe auch

`Member`

Kapitel 7: Skriptobjekte

Die in Director verfügbaren Skriptobjekte, auch Xtra-Erweiterungen genannt, bieten Zugriff auf die Funktionalität der mit Director® installierten Softwarekomponenten und erweitern die Director-Kernfunktionalität. Die vordefinierten Xtra-Erweiterungen stellen Funktionen bereit, mit denen Sie zum Beispiel Filter importieren oder eine Internetverbindung herstellen können. Wenn Sie mit der Programmiersprache C vertraut sind, können Sie eigene Xtra-Erweiterungen entwickeln.

Das Diagramm unter „Objektmodelldiagramme“ auf Seite 47 zeigt, wie die Skriptobjekte zueinander und zu anderen Objekten in Director in Bezug stehen.

Audiofilter

Audiofilter sind Plugin-Audioeditoren, mit denen Effekte auf Audiodateien wie MP3, WAV, MP4 usw. angewendet werden. Audiofilter wirken auf PCM-Samples.

Diese Filter haben folgende Syntax:

```
<audioFilter Object Reference> audioFilter (<symbol>, <paramList>)
```

Bei korrekter Angabe der Parameter liefern Filter Filterobjekte zurück. Falls die Parameter jedoch nicht korrekt festgelegt wurden oder Bereichsüberschreitungen aufweisen, wird ein Fehler zurückgegeben.

Übersicht: Eigenschaften von Audiofiltern

Eigenschaftsname
enabled (Filter)

EchoFilter

Ein Echo ist die Reflexion einer Schallwelle, die den Hörer mit einer Verzögerung erreicht. Ein Echo ist entweder eine einfache Reflexion oder eine mehrfache Reflexion eines Sounds.

Der Echofilter von Director fügt dem Sound eine Reihe wiederholter, schwächer werdender Echos hinzu.

Übersicht: Eigenschaften

Eigenschaft	Beschreibung	Bereich	Standard
echoLevel:Number	Verhältnis des bearbeiteten zum unbearbeiteten Signal.	0 - 1	0.8
feedback:Number	Prozentzahl des Ausgangs, der an den Eingang zurückgeführt wird.	0 - 1	0.5
delayTime:Number	Feedback-Verzögerung.	1 - 2000	1000

Beispiele

Die folgenden Beispiele wenden einen Echofilter mit Standardparametern auf ein Soundobjekt an:

```
-- Lingo syntax
on mouseUp me
mixref = new(#mixer)
soundobj = mixref.createsoundobject("so",member(2))
myfilter=audiofilter(#echofilter,[#echoLevel:0.8,#feedback:0.5,#delaytime:1000])
soundobj.filterlist.append(myfilter)
mixref.play()
end

// JavaScript syntax
function mouseUp()
{
mixer = _movie.newMember(symbol("mixer"));
soundobj = mixer.CreateSoundObject("so",member(2));
myfilter=audioFilter(symbol("Echofilter"),propList(symbol("echoLevel"),0.8,symbol("feedback"
),0.5,symbol("delaytime"),1000));
soundobj.filterlist.append(myfilter);
mixer.play();
}
```

FlangeFilter

Unter „Flange“ versteht man einen Echoeffekt, bei dem das Originalsignal und das Echo sehr kurz und zeitabhängig ist. Der Effekt wird auch als „überstreichender“ Sound bezeichnet.

Übersicht: Eigenschaften

Eigenschaft	Beschreibung	Bereich	Standard
mix	Stellt die Mischung aus Originalsignal (trocken) und Flangersignal (nass) ein. Einige Anteile von beiden Signalen sind erforderlich, um die charakteristische Signalunterdrückung und -Verstärkung zu erzielen, die während des Flanging auftritt. Bei einer Einstellung des Originalsignals auf 1 tritt kein Flanging auf. Bei einer Einstellung der Verzögerung auf 0 ist das Ergebnis ein schwankender Sound.	0 - 1	0.8
feedback	Bestimmt den Prozentsatz des Flangersignals, das an den Flanger zurückgemeldet wird. Ohne Feedback verwendet der Effekt nur das Originalsignal. Mit zusätzlichem Feedback verwendet der Effekt einen Prozentsatz des betroffenen Signals ab einem Punkt vor dem aktuellen Wiedergabepunkt.	0 - 1	0.5
delayTime	Minimale Verzögerung für die Kopie des Eingangssignals.	1 - 30	10
width	Maximale zusätzliche Verzögerung, die neben dem im Parameter „Delay“ festgelegten Verzögerung auf das Signal angewendet wird.	1-30	10
rate	Frequenz des Tiefpass-Oszillators, der auf das Originalsignal angewendet wird.	0.01 - 60	0.25
waveform	Die verwendete Signalform (Sinus, Dreieck, Logarithmus).	Sinus, Dreieck, Logarithmus	sine

Beispiele

```
-- Lingo syntax
on mouseUp me
mixref=new(#mixer)
soundobj=mixref.createsoundobject("so",member(2))
myfilter=audioFilter(#FlangeFilter, [#mix:0.8, #feedback:0.5, #delayTime:10, #width:10,
#rate:0.25, #waveform:#sine])
soundobj.filterlist.append(myfilter)
mixref.play()
end

// JavaScript syntax
function mouseUp()
{
mixer =_movie.newMember(symbol("mixer"));
soundobj =mixer.CreateSoundObject("so",member(2));
myfilter=audioFilter(symbol("FlangeFilter"),propList(symbol("Mix"),0.8,symbol("feedback"),0.
5,symbol("delaytime"),10,symbol("width"),10,symbol("frequency"),0.25,symbol("rate"),0.25,sym
bol("waveform"),symbol("sine")));
soundobj.filterlist.append(myfilter);
mixer.play();
}
```

DistortionFilter

Mit dem Verzerrungseffekt (Distortion) lassen sich defekte Kfz-Lautsprecher, übersteuerte Verstärker usw simulieren. Director unterstützt die Amplitudenverzerrung.

Die Amplitudenverzerrung wird erreicht, indem die Signalstärke mit benutzerdefinierten Werten (`distortionValues`) oder mit Zufallswerten innerhalb eines festgelegten Bereichs verändert wird.

Übersicht: Eigenschaften

Eigenschaft	Beschreibung	Bereich	Standard
percentage	Ob der angegebene Betrag ein Absolutwert oder ein prozentualer Wert ist.	0-1	1
amount	Zufallswerte, die unter dem Verzerrungswert liegen, werden den Daten des Audiosignals hinzugefügt, wodurch eine Verzerrung hervorgerufen wird. Die Werte müssen zwischen 0 und 100 liegen, falls es sich um Prozentwerte handelt. Andernfalls können die Werte zwischen 0 und 32000 liegen.	0-100/0-32000	25

Eigenschaft	Beschreibung	Bereich	Standard
useRandom	Ob Zufallswerte oder Werte der `distortionValues`-Matrix verwendet werden. Falls `useRandom` auf TRUE gesetzt ist, werden Werte zwischen 0 und den zuvor genannten Werten verwendet. Falls der Wert FALSE ist, werden die Werte der `distortionValues`-Matrix verwendet.	0-1	1
distortionValues	Ein zweidimensionales Array mit den Spalten `from decibel and to decibel` und einem Verzerrungsfaktor, der die Verzerrungswerte für unterschiedliche Dezibelwerte festlegt. Die Signalstärke im Bereich `distortionValues[x][0]` und `distortionValues[x][1]` werden mit dem Wert `distortionValues[x][2]` multipliziert.	Die Werte für distortion[x][2] liegen zwischen 0 und 10.	
numRange	Anzahl der Werte in `distortionValues` (Anzahl der Zeilen der Matrix).		

Beispiele

```
-- Lingo syntax
on mouseUp me
     mixref=new(#mixer)
     soundobj=mixref.createsoundobject("so",member(2))
     myfilter=audiofilter(#Distortionfilter) -- Creates the distortion filter
-- using default parameters.
     soundobj.filterlist.append(myfilter)
     rows=3
     columns=3
     freq=100
     amp=5
     myMatrix = newmatrix(rows,columns) -- Creates a 3X3 matrix.

     repeat with i = 1 to rows  -- Initializes the matrix.
       repeat with j = 1 to columns
         if j=3 then
           myMatrix.setval(i,j,amp)
           amp=amp+5
         else
           myMatrix.setval(i,j,freq)
           freq=freq+100
         end if
       end repeat
     end repeat

     myfilter.userandom=0
     myfilter.distortionvalues = myMatrix -- Assigns myMatrix to the Distortion matrix.
     myfilter.numrange=rows
     mixref.play()

end

// JavaScript syntax
function mouseUp()
{
    mixref =_movie.newMember(symbol("mixer"));
```

```
    soundobj=mixref.createSoundobject("so",member(2));
    myfilter=audioFilter(symbol("DistortionFilter"));// Creates the distortion filter
// using default parameters.
    soundobj.filterlist.append(myfilter);
    rows=3;
    columns=3;
    freq=100;
    amp=5;

     myMatrix = newMatrix(rows,columns ); //Creates a 3X3 matrix.

     for(i = 1;i<=rows;i++)
     {
        for(j = 1;j<=columns;j++)
        {
           if (j==3)
           {
            myMatrix.setval(i,j,amp);
            amp=amp+5;
            }
           else
           {
            myMatrix.setval(i,j,freq);
            freq=freq+100;
           }
        }
     }
    myfilter.userandom=0;
    myfilter.distortionvalues = myMatrix; // Assigns myMatrix to the Distortion matrix.
    myfilter.numrange=rows;
    mixref.play();

}
```

AmplifierFilter

Der Verstärkerfilter verstärkt das Audio-Eingangssignal oder schwächt es ab. Falls es beim Verstärkerfilter zu einer Verzerrung kommt, bleibt das Audiosignal dennoch unverzerrt.

Übersicht: Eigenschaften

Eigenschaft	Beschreibung	Bereich	Standard
Factor	Faktor, um den das Eingangssignal verstärkt wird.	0 bis 10	2

Beispiele

```
-- Lingo syntax
on mouseUp me
mixref=new(#mixer)
soundobj=mixref.createsoundobject("so",member(2))
myfilter=audioFilter(#AmplifierFilter, [#factor:2])
soundobj.filterlist.append(myfilter)
mixref.play()
end

// JavaScript syntax
function mouseUp()
{
mixer =_movie.newMember(symbol("mixer"));
soundobj =mixer.CreateSoundObject("so",member(2));
myfilter=audioFilter(symbol("AmplifierFilter"),propList(symbol("factor"),2));
soundobj.filterlist.append(myfilter);
mixer.play();
}
```

EnvelopeFilter

Der Oszillator-Filter überlagert dem Audiosignal eine oszillierende Schwingung (`sine wave, sawTooth` oder `rectangularwave`).

Syntax

```
audioFilter(#EnvelopeFilter, [#frequency:1, #waveType:#Sine])
```

Übersicht: Eigenschaften

Eigenschaft	Beschreibung	Bereich	Standard
Rate	Die Geschwindigkeit der Oszillator-Schwingung, die das Hauptaudiosignal überlagert.	0.01 bis 100	1
ByteArray	Legt durch das Erstellen eines Bytearrays eine benutzerdefinierte Hüllkurve fest. Jedes Byte des Bytearrays stellt eine Lautstärke der Hüllkurve dar. Die Lautstärke reicht von 0 (Stille) bis 255 (volle Lautstärke). Legen Sie die Signalform auf „#input“ fest, um dieses Byte-Array zu verwenden. Der Inhalt des Byte-Arrays wird als eine Wellenform behandelt und die Dauer wird über den Parameter „Frequenz“ festgelegt. Falls die Größe des Byte-Arrays 2 Byte ist und die Frequenz 1 Hz, dann wird jedes Byte des Byte-Arrays auf jeweils 0.5 Sekunden des Audiosignals angewendet.	0 bis 255	Der Standardwert für das Bytearray.
Wellenform	Typ der oszillierenden Schwingung.	sine, sawTooth, rectangular, triangular, input	sine

Beispiele

```
--Lingo syntax
on mouseUp me
mixref = new(#mixer )
soundobj = mixref.createSoundObject("so", member(2))
myfilter = audioFilter(#envelopeFilter, [ #rate:1, #waveform:#Sine])
soundobj.filterlist.append(myfilter)
mixref.play()
end

//JavaScript  syntax
function mouseUp()
{
mixer = _movie.newMe mber(symbol("m ixer"));
soundobj = mixer.CreateSoundObject("so",member(2));
myfilter=audioFilter(symbol("envelopeFilter"),propList(symbol("rate"),1,symbol("waveform"),s
ymbol("Sine")));
soundobj.filterlist.append(myfilter);
mixer.play();
}
```

Die Verwendung des Byte-Arrays ist wie folgt:

```
mix = new(#mixer)
so = mix.createSoundObject("1", member(1)) so.filterlist.append(audiofilter(#envelopefilter,
[#bytearray:b, #waveform:#input])) mix.play()
```

Das Bytearray wird im Filmskript wie folgt gefüllt:

```
on startmovie me
b = bytearray()
repeat with i = 1 to 51000
b[i] = i/200
end repeat
```

FadeOutFilter

Verringert die Lautstärke des Sounds kontinuierlich vom aktuellen Pegel auf Null.

Übersicht: Eigenschaften

Eigenschaft	Beschreibung	Bereich	Standard
duration	Die Zeitspanne, während derer die Lautstärke verringert wird. `duration` wird in Millisekunden angebenen.	1 to 1000000	1000
startDelay	Die erstmalige Verzögerung, nach welcher der Filter angewendet wird.	0 to 8000000	0

Beispiele

```
-- Lingo syntax
on mouseUp me
     mixref=new(#mixer)
     soundobj=mixref.createsoundobject("so",member(2))
     myfilter=audioFilter(#FadeOutFilter, [#duration:1000, #startDelay:0])
     soundobj.filterlist.append(myfilter)
     mixref.play()
end

// JavaScript syntax
function mouseUp()
{
mixer =_movie.newMember(symbol("mixer"));
soundobj =mixer.CreateSoundObject("so",member(2));
myfilter=audioFilter(symbol("FadeOutFilter"),propList(symbol("duration"),1000,
symbol("startdelay"),0));
soundobj.filterlist.append(myfilter);
mixer.play();
}
```

FadeInFilter

Erhöht die Lautstärke des Sounds kontinuierlich vom aktuellen Pegel auf den Maximalwert.

Übersicht: Eigenschaften

Eigenschaft	Beschreibung	Bereich	Standard
duration	Die Zeitspanne, während derer die Lautstärke auf den Maximalwert angehoben wird. `duration` wird in Millisekunden angebenen.	1 to 1000000	1000
startDelay	Die erstmalige Verzögerung, nach welcher der Filter angewendet wird. Während der Dauer von `startDelay` wird Stille ausgegeben.	0 to 8000000	0

Beispiele

```
-- Lingo syntax
on mouseUp me
     mixref=new(#mixer)
     soundobj=mixref.createsoundobject("so",member(2))
     myfilter=audioFilter(#FadeInFilter, [#duration:1000, #startDelay:0])
     soundobj.filterlist.append(myfilter)
     mixref.play()
end

// JavaScript syntax
function mouseUp()
{
mixer =_movie.newMember(symbol("mixer"));
soundobj =mixer.CreateSoundObject("so",member(2));
myfilter=audioFilter(symbol("FadeInFilter"),propList(symbol("duration"),1000,
symbol("startdelay"),0));
soundobj.filterlist.append(myfilter);
mixer.play();
}
```

FadeToFilter

Erhöht oder erniedrigt die Lautstärke des Audiosignals auf den angegebenen Wert.

Übersicht: Eigenschaften

Eigenschaft	Beschreibung	Bereich	Standard
duration	Die Zeitspanne, während derer die Lautstärke des Signals auf den angegebenen Wert geändert wird. `duration` wird in Millisekunden angebenen.	1 to 1000000	1000
toValue	Der Zielwert des Lautstärkepegels.	0 bis 255	0
startDelay	Die erstmalige Verzögerung, nach welcher der Filter angewendet wird.	0 to 8000000	0

Beispiele

```
-- Lingo syntax
on mouseUp me
     mixref=new(#mixer)
     soundobj=mixref.createsoundobject("so",member(2))
     myfilter=audioFilter(#FadeToFilter, [#toValue:0, #duration:1000, #startDelay:0])
     soundobj.filterlist.append(myfilter)
     mixref.play()
end

// JavaScript syntax
function mouseUp()
{
mixer =_movie.newMember(symbol("mixer"));
soundobj =mixer.CreateSoundObject("so",member(2));
myfilter=audioFilter(symbol("FadeToFilter"),propList(symbol("toValue"),0,symbol("duration"),
1000, symbol("startdelay"),0));
soundobj.filterlist.append(myfilter);
mixer.play();
}
```

LowPassFilter

Der Tiefpassfilter lässt nur Frequenzen durch, die unterhalb der angegebenen Trennfrequenz liegen. Andere Frequenzen werden vom Filter blockiert.

Übersicht: Eigenschaften

Eigenschaft	Beschreibung	Bereich	Standard
lowCutOffFreq	Trennfrequenz; Komponenten des Audiosignals unterhalb dieser Frequenz werden durchgelassen.	20 - 96000	1000

Beispiele

```
-- Lingo syntax
on mouseUp me
mixref=new(#mixer)
soundobj=mixref.createsoundobject("so",member(2))
myfilter=audioFilter(#LowPassFilter, [#lowCutOffFreq:1000])
soundobj.filterlist.append(myfilter)
mixref.play()
end

// JavaScript syntax
function mouseUp()
{
mixer =_movie.newMember(symbol("mixer"));
soundobj =mixer.CreateSoundObject("so",member(2));
myfilter=audioFilter(symbol("LowPassFilter"),propList(symbol("lowCutOffFreq"),1000));
soundobj.filterlist.append(myfilter);
mixer.play();
}
```

HighPassFilter

Der Hochpassfilter lässt nur Frequenzen durch, die oberhalb der angegebenen Trennfrequenz liegen. Andere Frequenzen werden vom Filter blockiert.

Übersicht: Eigenschaften

Eigenschaft	Beschreibung	Bereich	Standard
highCutOffFreq	Trennfrequenz; Komponenten des Audiosignals oberhalb dieser Frequenz werden durchgelassen.	20 - 96000	4000

Beispiele

```
-- Lingo syntax
on mouseUp me
mixref=new(#mixer)
soundobj=mixref.createsoundobject("so",member(2))
myfilter=audioFilter(#HighPassFilter, [#highCutOffFreq:4000])
soundobj.filterlist.append(myfilter)
mixref.play()
end

// JavaScript syntax
function mouseUp()
{
mixer =_movie.newMember(symbol("mixer"));
soundobj =mixer.CreateSoundObject("so",member(2));
myfilter=audioFilter(symbol("HighPassFilter"),propList(symbol("highCutOffFreq"),4000));
soundobj.filterlist.append(myfilter);
mixer.play();
}
```

BandPassFilter

Der Bandpassfilter lässt nur Frequenzen zwischen den beiden angegebenen Trennfrequenzen durch. Frequenzen außerhalb dieses Frequenzbandes werden blockiert.

Übersicht: Eigenschaften

Eigenschaft	Beschreibung	Bereich	Standard
lowCutOffFreq	Frequenzen zwischen der unteren und der oberen Trennfrequenz werden durchgelassen. Andere Frequenzen werden vom Filter blockiert.	20 - 96000	1000
highCutOffFreq	Frequenzen zwischen der unteren und der oberen Trennfrequenz werden durchgelassen. Andere Frequenzen werden vom Filter blockiert.	20 - 96000	4000

Beispiele

```
-- Lingo syntax
on mouseUp me
mixref=new(#mixer)
soundobj=mixref.createsoundobject("so",member(2))
myfilter=audioFilter(#BandPassFilter, [#lowCutOffFreq: 1000, #highCutOffFreq:4000])
soundobj.filterlist.append(myfilter)
mixref.play()
end

// JavaScript syntax
function mouseUp()
{
mixer =_movie.newMember(symbol("mixer"));
soundobj =mixer.CreateSoundObject("so",member(2));
myfilter=audioFilter(symbol("BandPassFilter"),propList(symbol("lowCutOffFreq"),1000,symbol("
highCutOffFreq"),4000));
soundobj.filterlist.append(myfilter);
mixer.play();
}
```

BandStopFilter

Der Bandstopp-Filter blockiert Frequenzen, die innerhalb der festgelegten Trennfrequenzen liegen. Andere Frequenzen werden durchgelassen.

Übersicht: Eigenschaften

Eigenschaft	Beschreibung	Bereich	Standard
lowCutOffFreq	Frequenzen innerhalb der oberen und unteren Trennfrequenz werden blockiert. Andere Frequenzen werden durchgelassen.	20 - 96000	1000
highCutOffFreq	Frequenzen innerhalb der oberen und unteren Trennfrequenz werden blockiert. Andere Frequenzen werden durchgelassen.	20 - 96000	4000

Beispiele

```
-- Lingo syntax
on mouseUp me
mixref=new(#mixer)
soundobj=mixref.createsoundobject("so",member(2))
myfilter=audioFilter(#BandStopFilter, [#lowCutOffFreq: 1000, #highCutOffFreq:4000])
soundobj.filterlist.append(myfilter)
mixref.play()
end

// JavaScript syntax
function mouseUp()
{
mixer =_movie.newMember(symbol("mixer"));
soundobj =mixer.CreateSoundObject("so",member(2));
myfilter=audioFilter(symbol("BandStopFilter"),propList(symbol("lowCutOffFreq"),1000,symbol("
highCutOffFreq"),4000));
soundobj.filterlist.append(myfilter);
mixer.play();
}
```

ReverbFilter

Der Hall-Filter erzeugt den Effekt eines diffusen Sounds oder Wiederhalls.

Übersicht: Eigenschaften

Eigenschaft	Beschreibung	Bereich	Standard
wetMix	Prozentzahl des bearbeiteten vom unbearbeiteten Signal.	0 - 1	0
dryMix	Prozentzahl des Originalsignals im Ausgang.	0 - 1	0
reverbWidth	Breite des Halls.	0.1 - 100	1
lowCutOffFreq	Frequenzen unterhalb dieses Schwellwerts werden abgeschwächt.	20 - 96000	500
highCutOffFreq	Frequenzen oberhalb dieses Schwellwerts werden abgeschwächt.	20 - 96000	5000
decayRate	Legt fest, wie schnell das Audiosignal nach der anfänglichen Reflexion schwächer wird.	0.01 - 0.99	0.2

Beispiele

```
-- Lingo syntax
on mouseUp me
     mixref=new(#mixer)
     soundobj=mixref.createsoundobject("so",member(2))
     myfilter=audioFilter(#ReverbFilter, [#wetMix:0.3, #dryMix:0, #reverbWidth:1,
#lowCutOffFreq:500, #highCutOffFreq:5000, #decayRate:0.2])
     soundobj.filterlist.append(myfilter)
     mixref.play()
end

// JavaScript syntax
function mouseUp()
{
mixref =_movie.newMember(symbol("mixer"));
soundobj =mixref.CreateSoundObject("so",member(2));
myfilter=audioFilter(symbol("ReverbFilter"),propList(symbol("wetMix"),0.3,symbol("dryMix"),0
,symbol("reverbWidth"),1,symbol("lowCutOffFreq"),500,symbol("highCutOffFreq"),5000),
symbol("decayRate"),0.2);
soundobj.filterlist.append(myfilter);
mixref.play();
}
```

PitchShiftFilter

Verschiebt die Frequenz des Audiosignals um den angegebenen Faktor.

Übersicht: Eigenschaften

Eigenschaft	Beschreibung	Bereich	Standard
Umschalt	Faktor, um den die Frequenz des Audiosignals verschoben wird.	0.1- 5	2

Beispiele

```
-- Lingo syntax
on mouseUp me
     mixref=new(#mixer)
     soundobj=mixref.createsoundobject("so",member(2))
     myfilter=audioFilter(#PitchShiftFilter, [#shift:2])
     soundobj.filterlist.append(myfilter)
     mixref.play()
end

// JavaScript syntax
function mouseUp()
{
mixer =_movie.newMember(symbol("mixer"));
soundobj =mixer.CreateSoundObject("so",member(2));
myfilter =audioFilter(symbol("PitchShiftFilter"),propList(symbol("shift"),2));
soundobj.filterlist.append(myfilter);
mixer.play();
}
```

ParamEqFilter

Beschreibung

Der parametrische Equalizer verstärkt Signale der angegebenen Frequenz oder schwächt sie ab.

Übersicht: Eigenschaften

Eigenschaft	Beschreibung	Bereich	Standard
centerFrequency	Mittenfrequenz	20 - 96000	8000
bandWidth	Bandbreite	20-96000	200
Gain	Verstärkung in dB.	-20 bis 20	0

Beispiele

```
-- Lingo syntax
on mouseUp me
     mixref=new(#mixer)
     soundobj=mixref.createsoundobject("so",member(2))
     myfilter=audiofilter(#ParamEqFilter,[#centerFrequency:8000, #bandWidth:200, #gain:0])
     soundobj.filterlist.append(myfilter)
     mixref.play()
end

//JavaScript syntax
function mouseUp()
{
mixer =_movie.newMember(symbol("mixer"));
soundobj = mixer.CreateSoundObject("so",member(2));
myfilter=audioFilter(symbol("ParamEqFilter"),propList(symbol("centerFrequency"),8000,symbol(
"bandWidth"),200,symbol("gain"),0));
soundobj.filterlist.append(myfilter);
mixer.play();
}
```

ByteArray

Das Lingo-Objekt `ByteArray` ermöglicht Ihnen das komfortable Lesen und Schreiben binärer Daten. `ByteArray` ist ein Core-Objekt und steht JavaScript und Xtras zur Verfügung.

ByteArray, Darsteller

Der Darsteller `#byteArray` lässt sich in Director, Shockwave und Projektor nutzen. Dieser Nur-Skript-Darsteller hat begrenzte Unterstützung für die Benutzerschnittstelle.

Erstellen eines ByteArray-Darstellers über die Benutzerschnittstelle Wählen Sie „Einfügen“ > „Medien-Elemente“ > „Byte Array “, um einen `ByteArray`-Darsteller einzufügen.

Erstellen eines ByteArray-Darstellers mit Lingo Verwenden Sie die Methode `new`, um einen `ByteArray`-Darsteller zu erstellen.

Syntax

```
m = new(#byteArray)
```

ByteArray-Darstellereigenschaften

`ByteArray`-Darsteller besitzen nur die Eigenschaft `byteArray`.

Übersicht der für das Objekt „ByteArray“ gültigen Operatoren

Operator	Rückgabetyp
Operator für wahlfreien Zugriff.	Ganzzahl
Zeichenfolge	Zeichenfolge

Übersicht: Eigenschaften des ByteArray-Objekts

Eigenschaftsname	Typ
BytesRemaining	Integer (Nur Lesen)
endian	Symbol. Mögliche Werte sind `#bigEndian` und `#littleEndian`. Der Standardwert ist `#littleEndian`.
length (byte array)	Integer (Nur Lesen)
position (byte array)	Ganzzahl

Übersicht der vom ByteArray-Objekt unterstützten Methoden

Methodennamen	
ByteArray	ByteArray(str)
readBoolean	writeBoolean
readInt8 readInt16 readInt32	writeInt8 writeInt16 writeInt32
readFloat32 readFloat64	writeFloat32 writeFloat64
readByteArray	writeByteArray
readString	writeString (byte array)
readRawString	writeRawString
compress()	uncompress()
toHexString	

Integration mit dem Lingo-Image-Objekt

Folgende Methoden lassen sich dazu verwenden, einen Bildpixel-Puffer als ByteArray anzusprechen und den Puffer mit aus dem ByteArray zu befüllen.

Methodennamen
getPixels()
setPixels()

Importiert ein ByteArray als Darsteller.

Die neue Methode `importByteArrayInto()` ähnelt der Funktion `importFileInto()` und wurde hinzugefügt, damit ByteArray-Inhalte als Medienausgabe hinzugefügt werden können.

Bytearray als Eingabe für ein Soundobjekt

Ein ByteArray-Objekt lässt sich einem Sound-Objekt übergeben. Wenn ein Byte-Array aus einer Datei generiert wird, liest das Byte-Array den Inhalt der Datei mithilfe von FileEA und übergibt den Inhalt des ByteArrays dem Sound-Objekt, sobald die Callback-Methode „True“ ergibt.

Hinweis: *Director-Audio unterstützt nur nicht signierte 8-Bit und signierte 16-Bit und 24-Bit Audiodateien für diesen Vorgang.*

Syntax

```
SoundObject Mixer.createSoundObject(SoundObjname,callbackFunction,[castMemRef],
[sampleRate,channelcount,bitDepth])
```

Parameter

Parameter	Beschreibung	Standardwert	Erforderlich/Optional
SoundObjname	Der mit dem Soundobjekt verbundene Name. Der Name der Soundobjekte muss eindeutig sein.		Erforderlich
callbackFunction	Wenn eine Lingo- JavaScriptfunktion an die Funktion `createSoundObject` übergeben wird, ruft das Soundobjekt die Funktion `callbackFunction` auf, sobald es zur Übernahme von Wiedergabedaten bereit ist.		Erforderlich
castMemRef	Übergibt eine Darstellerreferenz, wenn die Callbackmethode ein Darsteller des Parentskripts ist. Wenn die Callbackmethode in „movieScript“ ist, legen Sie keinen Parameter fest oder legen Sie den Parameter als „Leer“ fest.		Optional
sampleRate	Samplingfrequenz der Audiodaten.	48000 kHz	Optional
channelCount	Anzahl der Kanäle der Audiodaten.	2	Optional
bitDepth	Bittiefe der Audiodaten.	16	Optional

Weitere Informationen über „callbackFunction“

Wenn eine Lingo- JavaScriptfunktion an die Funktion `createSoundObject` übergeben wird, ruft das Soundobjekt die Funktion `callbackFunction` auf, sobald es zur Übernahme von Wiedergabedaten bereit ist.

Optional können Sie mit der Callbackfunktion auch eine Referenz auf einen Darsteller übergeben.

Beim Bereitstellen der Eingabe an `callbackFunction` gilt Folgendes:

- Die registrierte Callback-Funktion wird kurz nach dem Abspielen des Soundobjekts aufgerufen.

- Der Parameter für die Callback-Funktion ist ein Byte-Array mit der Länge 0. Dieses Byte-Array wird mit Audiodaten gefüllt. Die Länge der Daten, die in das Byte-Array kopiert werden können, ist nicht direkt begrenzt.
- Die folgende Callback-Funktion wird aufgerufen, nachdem die Daten abgespielt wurden.
- Der Rückgabewert der Callback-Funktion kann „True“ oder „False“ sein, je nachdem, ob das Soundobjekt die Callback-Funktion beim nächsten Mal erneut aufrufen soll („True“) oder nach dem Abspielen der Audiodaten stoppen soll („False“).
- Falls nötig können Sie das Soundobjekt in der Callbackfunktion manuell stoppen. Das manuelle Stoppen des Soundobjekts stoppt auch die Callback-Methode.

Die Syntax von `callbackFunction` ist wie folgt:

```
On callbackFunction outByteArray
--outByteArray is filled with bytearray samples.
end
```

Hinweis: *Wenn innerhalb der Callbackprozedur ein Skript Laufzeitfehler verursacht (weil z. B. eine Eigenschaft nicht gefunden wird), werden die Fehlermeldungen nicht angezeigt und die Prozedurausführung wird an dem Punkt abgebrochen, an welchem der Fehler auftrat. Alle nachfolgenden Anweisungen werden nicht ausgeführt.*

Das übergebene Bytearray an die Funktion wird mit der angeforderten Anzahl von Bytes aus den Sampledaten des Byte-Arrays gefüllt.

Hinweis: *Die PCM-Sampledaten müssen in dem Format vorliegen, dass in `createSoundObject` festgelegt wurde.*

Falls in `createSoundObject` kein bestimmtes Format festgelegt wurde, wird das Standardformat mit folgender Spezifikation verwendet:

Bittiefe	16
Samplingfrequenz	48000 Hz
Anzahl der Kanäle	2

Um das Standardformat außer Kraft zu setzen, müssen folgende Symbole der Eigenschaftsliste gesetzt werden:

```
#bitDepth --Bit depth of the audio.
#sampleRate --Sampling frequency of the audio.
#channelCount --Number of channels in the audio.
```

Beispiele

```
    -- BUTTON BEHAVIOR --
    global gSoundIOInstance --  instance  of FileIO for  reading sound file
    global Sound             -- soundObject
    global gMixer             -- mixer object
    on mouseUp(me)
    ---------------------------------------------------------------------
    -- ACTION: Opens a sound file in the same folder as the movie
    -- and creates a soundObject to play it back. The sound file
    -- will be read in when requested by a callback from the
    -- sound object. The callback handler HAS to be in a Movie Script, which
    -- means that globals MUST be used to play back the sound file.
    -- See the Movie Script for callback details.
    ---------------------------------------------------------------------
-- Open a sample file to read the data
    gSoundIOInstance = new xtra("fileIO")
    gSoundIOInstance.openFile(_movie.path & "RawSound.snd",0)
    gMixer = new(#mixer)
-- Create the sound object with the SampleRate, ChannelCount &
-- BitDepth. Specify the callback method.
    vName = "Imported sound mixer"
    vCallback  = #audioInput
    vByteArray = VOID -- Callback method is available in the Movie Script, can specify
    -- scriptReference also
    vSettings  = [#samplerate:48000, #channelcount:2, #bitdepth:16]
    gSound = gMixer.createSoundObject(\
    vName, \
    vCallback, \
    VOID, \
    vSettings)
-- Play the sound
    gMixer.play()
    end mouseUp
```

Callbackmethode (Filmskript)

```
-- MOVIE SCRIPT --
global gSoundIOInstance -- instance of FileIO for reading sound file
global gSound           -- soundObject
on audioInput(aByteArray)
----------------------------------------
-- SOURCE: Called back from gSound whenever gSound has finished
-- playing its current contents.
-- INPUT: <aByteArray> will be an empty byteArray which has to be
-- filled and the modified byteArray to send back.
-- ACTION: Transfers the (remaining) contents of the file to
-- aByteArray
-- OUTPUT: Returns TRUE if there was any data to pas to aByteArray
-- FALSE if not.
------------------------------------------------------------------
 -- Checking the if we've reached the end of the file
vFileLength = gSoundIOInstance.getLength()
vPosition   = gSoundIOInstance.getPosition()
if vPosition = vFileLength then
-- We've reach the end of the file.  Close it and tell the
-- soundObject to stop calling back.
gSoundIOInstance.closefile()
return FALSE -- stop calling the callback
end if
-- Read entire contents of the file into a byteArray
vByteArray   = gSoundIOInstance.readByteArray(vFileLength)
vArrayLength = vByteArray.length
put vArrayLength, vFileLength
-- Copy from the temporary byteArray to the one passed in as a
-- parameter
if vArrayLength <> 0 then
vResult = aByteArray.writeByteArray(vByteArray, 1, vArrayLength)
end if
-- Ensure this callback is made again when the sound has played out
return TRUE
end audioInput

on stopmovie
gSoundIOInstance.closefile()
end stopmovie
```

Bytearray als Ausgabe eines Soundobjekts

Verwenden Sie `registerByteArrayCallback` und `unregisterByteArrayCallback`, um den Inhalt des Soundobjekts als Bytearray zu erhalten. Wenn die Callbackmethode ausgeführt wird, wird das Bytearray mit Audiosampledaten aus dem Soundobjekt gefüllt Die Daten des Byte-Arrays lassen sich vor dem Abspielen des Soundobjekts ändern.

Fileio

Ermöglicht, Dateieingabe- und -ausgabeoperationen durchzuführen.

Mit dem Operator `new` können Sie einen Verweis auf ein Fileio-Objekt erstellen.

```
-- Lingo syntax
objFileio = new xtra("fileio")

// JavaScript syntax
var objFileio = new xtra("fileio");
```

FileIO Xtra kann beliebig kodierte Dateien lesen und schreiben. In der neuen Version lassen sich mittels Bytearrays are Binärdaten in eine Datei schreiben oder aus dieser lesen.

Außerdem unterstützt FileIO jetzt neben dem Schreiben von Bytearrays in externe Dateien auch das direkte Lesen von Daten aus einem Bytearray.

Übersicht: Methoden des Fileio-Objekts

Methode	
closeFile()	createFile()
delete() (FileIO)	deleteFile()
displayOpen()	displaySave()
error()	fileName()
getCharSet	getFinderInfo()
getLength()	getOSDirectory()
getPosition()	openFile()
readByteArray (FileIO Xtra)	readChar()
readFile()	readLine()
readToken()	readWord()
setFilterMask()	setFinderInfo()
setNewLineConversion()	setPosition()
status()	setCharSet
writeByteArray (FileIO Xtra)	writeChar()
writeReturn()	writeString()
version()	

Das Xtra MUI

Das Xtra MUI ermöglicht vollfunktionsfähige Dialogfelder, die auf die von Ihnen gewünschte Art eingerichtet werden. Diese Dialogfelder benötigen nicht den Arbeits- bzw. Festplattenspeicher eines Films in einem Fenster (MIAW), der ein Dialogfeld simuliert.

Mit dem Operator `new` können Sie einen Verweis auf ein Objekt des Xtras MUI erstellen.

```
-- Lingo syntax
objMui = new xtra("Mui")

// JavaScript syntax
var objMui = new xtra("Mui");
```

Übersicht: Methoden des XML Parser-Objekts

Methode
Alert()
fileOpen()
fileSave()
GetItemPropList
getURL()
GetWidgetList()
GetWindowPropList
Initialize
ItemUpdate()
run
stop
WindowOperation

NetLingo

Ermöglicht Ihnen die Durchführung von Netzwerkvorgängen: Sie können z. B. Medien aus einem Netzwerk abrufen oder streamen, die Netzwerkverfügbarkeit prüfen, den Verlauf eines Netzwerkvorgangs kontrollieren usw.

Mit dem Operator `new` können Sie einen Verweis auf ein NetLingo-Objekt erstellen.

```
-- Lingo syntax
objNetLingo = new xtra("netlingo")

// JavaScript syntax
var objNetLingo = new xtra("netlingo");
```

Director stellt Lingo- und JavaScript-Methoden zur Verfügung, um Bytearrays über das Internet senden und empfangen zu können.

Übersicht: Methoden des NetLingo-Objekts

Methode	
browserName()	cacheDocVerify()
cacheSize()	clearCache
downloadNetThing	externalEvent()

Methode	
getLatestNetID	getNetByteArray
getNetText()	getStreamStatus()
gotoNetPage	gotoNetMovie
netAbort	netByteArrayResult
netDone()	netError()
netLastModDate()	netMIME()
netStatus	netTextResult()
postNetByteArray	postNetText
preloadNetThing()	proxyServer
tellStreamStatus()	URLEncode

SpeechXtra

Ermöglicht Ihnen, Sprachausgabefunktionen zu einem Film hinzuzufügen.

Mit dem Operator `new` können Sie einen Verweis auf ein SpeechXtra-Objekt erstellen.

```
-- Lingo syntax
objSpeech = new xtra("speechxtra")

// JavaScript syntax
var objSpeech = new xtra("speechxtra");
```

Übersicht: Methoden des SpeechXtra-Objekts

Methode	
voiceCount()	voiceSet()
voiceGet()	voiceSetPitch()
voiceGetAll()	voiceSetRate()
voiceGetPitch()	voiceSetVolume()
voiceGetRate()	voiceSpeak()
voiceGetVolume()	voiceState()
voiceInitialize()	voiceStop()
voicePause()	voiceWordPos()
voiceResume()	

XML-Parser

Ermöglicht die Durchführung von XML-Parsing.

Mit dem Operator `new` können Sie einen Verweis auf ein XML Parser-Objekt erstellen.

```
-- Lingo syntax
objXml = new xtra("xmlparser")

// JavaScript syntax
var objXml = new xtra("xmlparser");
```

Übersicht: Methoden des XML Parser-Objekts

Methode
count()
doneParsing()
getError() (XML)
ignoreWhiteSpace()
makeList()
makeSubList()
parseString()
parseURL()

Übersicht: Eigenschaften des XML Parser-Objekts

Eigenschaft
attributeName
attributeValue
child (XML)
name (XML)

XML Xtra

XML-Dokumente lassen sich mit einer beliebigen Kodierung erstellen und die Encoding-Informationen werden in das XML-Deklarationstag geschrieben.

```
<?xml version="1.0" encoding="utf-8"?>
```

Zusätzlich lassen sich in XML-Dateien potenziell auch binäre Datenbereiche speichern, sodass ein XML-Dokument nicht nur UTF-8-Zeichen enthalten muss.

Der XML-Parser nutzt das XML-Deklarationstag des Dokuments zum Identifizieren der verwendeten Textkodierung.

Übersicht der Methoden von XML Xtra

Name der Methode
parseByteArray
parseString (XML Xtra)
parseURL (XML Xtra)

Kapitel 8: 3D-Objekte

Informationen zu 3D-Objekten

Mit den 3D-Objekten können Sie 3D-Funktionalität zu einem Film hinzufügen. Diese Objekte werden in Director sowohl für Lingo als auch JavaScript-Syntax, Projektoren und Adobe®-Shockwave®-Player zur Verfügung gestellt.

Der Zugriff auf diese 3D-Objekte erfolgt über Shockwave 3D-Darsteller (auch nur 3D-Darsteller genannt). Sie können auch 3D-Sprites aus den 3D-Darstellern erstellen. Sowohl 3D-Darsteller als auch 3D-Sprites enthalten für 3D-Darsteller und -Sprites spezifische Funktionalität. Zudem haben sie Zugriff auf die Funktionalität anderer Darsteller und Sprites, deren APIs vom Core-Objekt `Member` bzw. `Sprite` festgelegt werden.

3D-Darsteller unterscheiden sich von anderen Darstellern insofern, als sie eine komplette 3D-Welt enthalten. Eine 3D-Welt enthält die Objekte, die den Zugriff auf 3D-Funktionalität ermöglichen. Alle Objekte einer 3D-Welt basieren auf einem Basisobjekt, dem so genannten Knoten. Die einfachste Form eines Knotens in einer 3D-Welt ist ein Group-Objekt. Ein Group-Objekt ist im Prinzip der grundlegendste Knoten. Alle anderen Objekte in einer 3D-Welt basieren auf einem Group-Objekt, d. h., die anderen Objekte erben die Funktionalität eines Group-Objekts zusätzlich zu ihrer spezifischen Funktionalität.

Die Abbildung „Objektmodelldiagramme“ auf Seite 47 zeigt, wie die 3D-Objekte zueinander und zu anderen Objekten in Director in Bezug stehen.

Der Lieferumfang von Director® umfasst zwei Xtra-Erweiterungen, die den Zugriff auf die 3D-Objekte ermöglichen:

- Das 3D Asset-Xtra (3DAuth.x32 unter Windows®, 3D Auth-Xtra unter Macintosh®) bietet Unterstützung für das 3D-Medienfenster innerhalb von Director.
- Das 3D Media-Xtra (Shockwave 3D Asset.x32 unter Windows, 3D Asset-Xtra unter Mac) ermöglicht die Unterstützung der eigentlichen 3D-Medien.

  Damit Sie bei der Erstellung order zur Laufzeit auf die 3D-Objekte zugreifen können, muss der Film das 3DAsset-Xtra enthalten.

Kamera

Steht für ein Kameraobjekt.

Eine Kamera steuert die Sicht eines 3D-Sprites auf die 3D-Welt. Ein 3D-Sprite zeigt die Welt aus dem Blickwinkel einer bestimmten Kamera.

Mit der Eigenschaft `camera` des 3D-Objekts `Member` können Sie einen Verweis auf eine Kamera erstellen. Die Eigenschaft `camera` gilt für die Kamera, die sich an der angegebenen Indexposition in der Kameraliste befindet. In Lingo verwenden Sie die Eigenschaft `camera` direkt vom 3D-Objekt `Member` aus, um einen Verweis zu erstellen. In JavaScript-Syntax müssen Sie zum Erstellen eines Verweises dagegen die Methode `getPropRef()` verwenden.

Im folgenden Beispiel wird ein Verweis auf die zweite Kamera des 3D-Darstellers „family room“ erstellt und der Variablen `myCamera` zugewiesen.

```
-- Lingo syntax
myCamera = member("family room").camera[2]

// JavaScript syntax
var myCamera = member("family room").getPropRef("camera", 2);
```

Übersicht: Methoden des Camera-Objekts

Methode
addBackdrop
addOverlay
insertBackdrop
insertOverlay
removeBackdrop
removeOverlay

Übersicht: Eigenschaften des Camera-Objekts

Eigenschaft	
backdrop	fog.far (fog)
backdrop[].blend (3D)	fog.near (fog)
backdrop[].loc (backdrop and overlay)	hither
backdrop[].regPoint (3D)	orthoHeight
backdrop[].rotation (backdrop and overlay)	overlay
backdrop[].scale (3D)	overlay[].blend (3D)
backdrop[].source	overlay[].loc (backdrop and overlay)
backdrop.count (3D)	overlay[].regPoint (3D)
child (3D)	overlay[].rotation (backdrop and overlay)
colorBuffer.clearAtRender	overlay[].scale (3D)
colorBuffer.clearValue	overlay[].source
fieldOfView (3D)	overlay.count (3D)
fog.color()	projection
fog.decayMode	rootNode
fog.enabled (fog)	yon

Siehe auch

Gruppe, Light, Modell, Model Resource, Bewegung, Shader, Texture

Gruppe

Steht für ein Modell ohne Ressource oder Shader.

Die Gruppe ist der grundlegendste Knoten und stellt lediglich einen Punkt im Raum dar, der durch eine Transformation dargestellt wird. Diesem Knoten können unter- und übergeordnete Objekte zugewiesen werden, um Modelle, Lichtquellen, Kameras oder andere Gruppen zu gruppieren.

Die einfachste Gruppe ist die so genannte Welt; sie ist im Grunde synonym mit einem 3D-Darsteller.

Mit der Eigenschaft `group` des 3D-Objekts `Member` können Sie einen Verweis auf eine Gruppe erstellen. Die Eigenschaft `group` gilt für die Gruppe, die sich an der angegebenen Indexposition in der Gruppenliste befindet. In Lingo verwenden Sie die Eigenschaft `group` direkt vom 3D-Objekt `Member` aus, um einen Verweis zu erstellen. In JavaScript-Syntax müssen Sie zum Erstellen eines Verweises dagegen die Methode `getPropRef()` verwenden.

Im folgenden Beispiel wird ein Verweis auf die erste Gruppe des 3D-Darstellers `space` erstellt und der Variablen `myGroup` zugewiesen.

```
-- Lingo syntax
myGroup = member("space").group[1]

// JavaScript syntax
var myGroup = member("space").getPropRef("group", 1);
```

Übersicht: Methoden des Group-Objekts

Methode	
addChild	pointAt
addToWorld	registerScript()
clone	removeFromWorld
cloneDeep	rotate
getWorldTransform()	scale (Befehl)
isInWorld()	translate

Übersicht: Eigenschaften des Group-Objekts

Eigenschaft
name (3D)
parent
pointAtOrientation
transform (Eigenschaft)
userData
worldPosition

Siehe auch

Kamera, Light, Modell, Model Resource, Bewegung, Shader, Texture

Light

Steht für eine Lichtquelle in einer 3D-Welt.

Lichtquellen dienen zur Beleuchtung einer 3D-Welt. Ohne Lichtquellen wären die Objekte innerhalb der Welt nicht sichtbar.

Mit der Eigenschaft `light` des 3D-Objekts `Member` können Sie einen Verweis auf eine Lichtquelle erstellen. Die Eigenschaft `light` gilt für die Lichtquelle, die sich an der angegebenen Indexposition in der Lichtquellenliste befindet. In Lingo verwenden Sie die Eigenschaft `light` direkt vom 3D-Objekt `Member` aus, um einen Verweis zu erstellen. In JavaScript-Syntax müssen Sie zum Erstellen eines Verweises dagegen die Methode `getPropRef()` verwenden.

Im folgenden Beispiel wird ein Verweis auf die dritte Lichtquelle des 3D-Darstellers „film room“ erstellt und der Variablen `myLight` zugewiesen.

```
-- Lingo syntax
myLight = member("film room").light[3]

// JavaScript syntax
var myLight = member("film room").getPropRef("light", 3);
```

Übersicht: Eigenschaften des Light-Objekts

Eigenschaft
attenuation
color (light)
specular (light)
spotAngle
spotDecay
type (light)

Siehe auch

`Kamera`, `Gruppe`, `Modell`, `Model Resource`, `Bewegung`, `Shader`, `Texture`

Member

Steht für einen Shockwave 3D-Darsteller.

Shockwave 3D-Darsteller (auch nur 3D-Darsteller genannt) enthalten eine komplette 3D-Welt. Eine 3D-Welt enthält die Objekte, mit denen Sie 3D-Funktionalität zu einem Film hinzuzufügen.

Zum Erstellen eines Verweises auf einen 3D-Darsteller verwenden Sie entweder die Top-Level-Funktion `member()` oder die Eigenschaft `member` des Movie- oder Sprite-Objekts. Mit diesen Verfahren werden auch Verweise auf Nicht-3D-Darsteller erstellt.

- Verwenden Sie die Top-Level-Funktion `member()`.

```
-- Lingo syntax
3dMember = member("magic")

// JavaScript syntax
var 3dMember = member("magic");
```

- Verwenden Sie die Eigenschaft `member` des Sprite-Objekts.

```
-- Lingo syntax
3dMember = sprite(1).member;

// JavaScript syntax
var 3dMember = sprite(1).member;
```

Übersicht: Methoden des Member-Objekts

Methode	
camera()	model (3D)
cloneModelFromCastmember	modelResource
cloneMotionFromCastmember	motion()
deleteCamera	newCamera
deleteGroup	newGroup
deleteLight	newLight
deleteModel	newMesh
deleteModelResource	newModel
deleteMotion	newModelResource
deleteShader	newShader
deleteTexture	newTexture
extrude3D	resetWorld
group()	revertToWorldDefaults
light()	
loadFile()	

Übersicht: Eigenschaften des Member-Objekts

Eigenschaft	
ambientColor	loop (3D)
animationEnabled	model
bevelDepth	modelResource
bevelType	motion
bytesStreamed (3D)	percentStreamed (3D)
camera	preLoad (3D)

Eigenschaft	
cameraPosition	reflectivity
cameraRotation	shader
diffuseColor	smoothness
directionalColor	specularColor
directionalPreset	state (3D)
directToStage	streamSize (3D)
displayFace	texture
displayMode	textureMember
group	textureType
light	tunnelDepth

Siehe auch

Kamera, Gruppe, Light, Modell, Model Resource, Bewegung, Shader, Sprite, Texture

Modell

Steht für ein sichtbares Objekt, das Benutzer innerhalb einer 3D-Welt sehen.

Modelle nutzen eine Modellressource und nehmen eine bestimmte Position und Ausrichtung in der 3D-Welt ein. Bei Modellressourcen handelt es sich um Elemente der 3D-Geometrie, die zum Zeichnen von 3D-Modellen verwendet werden können. Modelle definieren außerdem das Erscheinungsbild der Modellressource, also etwa die verwendeten Texturen und Shader. Weitere Informationen über die Beziehung zwischen Modellen und Modellressourcen finden Sie in den Themen des Director-Benutzerhandbuchs im Director-Hilfefenster.

Mit der Eigenschaft `model` des 3D-Objekts `Member` können Sie einen Verweis auf ein Modell erstellen. Die Eigenschaft `model` gilt für das Modell, das sich an der angegebenen Indexposition in der Modellliste befindet. In Lingo verwenden Sie die Eigenschaft `model` direkt vom 3D-Objekt `Member` aus, um einen Verweis zu erstellen. In JavaScript-Syntax müssen Sie zum Erstellen eines Verweises dagegen die Methode `getPropRef()` verwenden.

Im folgenden Beispiel wird ein Verweis auf das zweite Modell des 3D-Darstellers `Transportation` erstellt und der Variablen `myModel` zugewiesen.

```
-- Lingo syntax
myModel = member("Transportation").model[2]

// JavaScript syntax
var myModel = member("Transportation").getPropRef("model", 2);
```

Ein Modell enthält zudem Modifizierer, die steuern, wie das Modell gerendert wird oder sich seine Animation verhält. Modifizierer werden einem Modell mit der Methode `addModifier()` zugewiesen. Sobald der Modifizierer am Modell angebracht ist, können seine Eigenschaften mit Skripts manipuliert werden.

Folgende Modifizierer stehen für Modelle zur Verfügung:

Modifizierer	**Beschreibung**
Knochen-Player	Verändert die Modellgeometrie über einen bestimmten Zeitraum hinweg.
Collision (Kollision)	Ermöglicht einem Modell, auf Kollisionen hingewiesen zu werden und entsprechend zu reagieren.
Inker (Farbwalze)	Fügt einem vorhandenen Modell Silhouetten-, Falten- und Umrisskanten hinzu.
Keyframe Player (Schlüsselbild-Player)	Verändert die `transform`-Eigenschaften eines Modells über einen bestimmten Zeitraum hinweg.
Level of Detail (LOD, Details)	Ermöglicht die Steuerung der für die Wiedergabe eines Modells verwendeten Anzahl von Polygonen auf Modellebene, basierend auf dem Abstand des Modells von der Kamera. Der Modifizierer „LOD“ kann auch für Modellressourcen verwendet werden.
Mesh Deform (Gitternetzdeformation)	Ändert die Geometrie einer vorhandenen Modellressource zur Laufzeit.
Subdivision Surfaces (SDS, Oberflächenunterteilung)	Bewirkt, dass das Modell in dem Bereich, der gegenwärtig von der Kamera gezeigt wird, mit zusätzlichen geometrischen Details gerendert wird.
Toon (Cartoon)	Ändert das Rendering eines Modells, so dass es im Cartoonstil dargestellt wird.

Weitere Informationen zu den Methoden, Eigenschaften und Ereignissen für Modifizierer finden Sie in den Themen des Director-Benutzerhandbuchs im Director-Hilfefenster.

Model Resource

Steht für ein Element der 3-Geometrie, das zum Zeichnen von 3D-Modellen verwendet wird.

Modelle nutzen eine Modellressource und nehmen eine bestimmte Position und Ausrichtung in der 3D-Welt ein. Modelle definieren außerdem das Erscheinungsbild der Modellressource, also etwa die verwendeten Texturen und Shader.

Weitere Informationen über die Beziehung zwischen Modellen und Modellressourcen sowie über ihre Verwendung finden Sie in den Themen des Director-Benutzerhandbuchs im Director-Hilfefenster.

Mit der Eigenschaft `modelResource` des 3D-Objekts `Member` können Sie einen Verweis auf eine Modellressource erstellen. Die Eigenschaft `modelResource` gilt für die Modellressource, die sich an der angegebenen Indexposition in der Liste der Modellressourcen befindet. In Lingo verwenden Sie die Eigenschaft `modelResource` direkt vom 3D-Objekt `Member` aus, um einen Verweis zu erstellen. In JavaScript-Syntax müssen Sie zum Erstellen eines Verweises dagegen die Methode `getPropRef()` verwenden.

Im folgenden Beispiel wird ein Verweis auf die zweite Modellressource des 3D-Darstellers `wheels` erstellt und der Variablen `myModelResource` zugewiesen.

```
-- Lingo syntax
myModelResource = member("wheels").modelResource[2]

// JavaScript syntax
var myModelResource = member("wheels").getPropRef("modelResource", 2);
```

Bewegung

Steht für eine vordefinierte Animationssequenz, bei der sich ein Modell oder eine Modellkomponentebewegt.

Die einzelnen Bewegungen können entweder für sich alleine oder in Verbindung mit anderen Bewegungen abgespielt werden. Eine Rennbewegung lässt sich beispielsweise mit einem Sprung kombinieren, um darzustellen, wie eine Person beim Rennen über eine Pfütze springt.

Mit der Eigenschaft `motion` des 3D-Objekts `Member` können Sie einen Verweis auf eine Bewegung erstellen. Die Eigenschaft `motion` gilt für die Bewegung, die sich an der angegebenen Indexposition in der Liste der Bewegungen befindet. In Lingo verwenden Sie die Eigenschaft `motion` direkt vom 3D-Objekt `Member` aus, um einen Verweis zu erstellen. In JavaScript-Syntax müssen Sie zum Erstellen eines Verweises dagegen die Methode `getPropRef()` verwenden.

Im folgenden Beispiel wird ein Verweis auf die vierte Bewegung des 3D-Darstellers `athlete` erstellt und der Variablen `myMotion` zugewiesen.

```
-- Lingo syntax
myMotion = member("athlete").motion[4]

// JavaScript syntax
var myMotion = member("athlete").getPropRef("motion", 4);
```

Renderer Services

Steht für das globale Objekt, das eine Eigenschaftsliste enthält, deren Werte sich auf die gemeinsamen Rendering-Eigenschaften für alle 3D-Darsteller und -Sprites auswirken.

Sie können über die Top-Level-Funktion `getRendererServices()` auf das globale Renderer Services-Objekt zugreifen.

Im folgenden Beispiel wird auf die Renderer-Eigenschaft des globalen Renderer Services-Objekts zugegriffen und der Wert der Variablen `myRenderer` zugewiesen.

```
-- Lingo syntax
myRenderer = getRendererServices().renderer

// JavaScript syntax
var myRenderer = getRendererServices().renderer;
```

Übersicht: Methoden des Renderer Services-Objekts

Methode
getHardwareInfo()

Übersicht: Eigenschaften des Renderer Services-Objekts

Eigenschaft
modifiers
primitives
renderer
rendererDeviceList
textureRenderFormat

Siehe auch

Member, Sprite

Shader

Steht für die Oberflächenfarbe eines Modells.

Sie können die Oberfläche eines Modells mit Grafiken versehen, indem Sie eine oder mehrere Texturen auf jeden Shader anwenden.

Mit der Eigenschaft `shader` des 3D-Objekts `Member` können Sie einen Verweis auf einen Shader erstellen. Die Eigenschaft `shader` gilt für den Shader, der sich an der angegebenen Indexposition in der Shader-Liste befindet. In Lingo verwenden Sie die Eigenschaft `shader` direkt vom 3D-Objekt `Member` aus, um einen Verweis zu erstellen. In JavaScript-Syntax müssen Sie zum Erstellen eines Verweises dagegen die Methode `getPropRef()` verwenden.

Im folgenden Beispiel wird ein Verweis auf den zweiten Shader des 3D-Darstellers `triangle` erstellt und der Variablen `myShader` zugewiesen.

```
-- Lingo syntax
myShader = member("triangle").shader[2]

// JavaScript syntax
var myShader = member("triangle").getPropRef("shader", 2);
```

Sprite

Steht für ein 3D-Sprite, das aus einem Shockwave 3D-Darsteller erstellt wurde.

Zum Erstellen eines Verweises auf ein 3D-Sprite können Sie die Top-Level-Funktion `sprite()`, die Eigenschaft `sprite` des Movie-Objekts oder die Eigenschaft `sprite` des Sprite Channel-Objekts verwenden. Mit diesen Verfahren werden auch Verweise auf Nicht-3D-Sprites erstellt.

- Verwenden Sie die Top-Level-Funktion `sprite()`.

  ```
  -- Lingo syntax
  3dSprite = sprite(1)

  // JavaScript syntax
  var 3dSprite = sprite(1);
  ```

- Verwenden Sie die Eigenschaft `sprite` des Movie-Objekts.

  ```
  -- Lingo syntax
  3dSprite = _movie.sprite["willowTree"]

  // JavaScript syntax
  var 3dSprite = _movie.sprite["willowTree"];
  ```

- Verwenden Sie die Eigenschaft `sprite` des Sprite Channel-Objekts.

  ```
  -- Lingo syntax
  3dSprite = channel(3).sprite

  // JavaScript syntax
  var 3dSprite = channel(3).sprite;
  ```

Übersicht: Methoden des Sprite-Objekts

Methode
addCamera
cameraCount()
deleteCamera

Übersicht: Eigenschaften des Sprite-Objekts

Eigenschaft
antiAliasingEnabled
backColor
camera
directToStage

Siehe auch

`Kamera`, `Member`

Texture

Steht für die auf einen Shader angewendete Textur.

Mit der Eigenschaft `texture` des 3D-Objekts `Member` können Sie einen Verweis auf eine Textur erstellen. Die Eigenschaft `texture` gilt für die Textur, die sich an der angegebenen Indexposition in der Texturenliste befindet. In Lingo verwenden Sie die Eigenschaft `texture` direkt vom 3D-Objekt `Member` aus, um einen Verweis zu erstellen. In JavaScript-Syntax müssen Sie zum Erstellen eines Verweises dagegen die Methode `getPropRef()` verwenden.

Im folgenden Beispiel wird ein Verweis auf die erste Textur des 3D-Darstellers `triangle` erstellt und der Variablen `myTexture` zugewiesen.

```
-- Lingo syntax
myTexture = member("triangle").texture[1]

// JavaScript syntax
var myTexture = member("triangle").getPropRef("texture", 1);
```

Kapitel 9: Konstanten

In diesem Abschnitt finden Sie eine alphabetische Liste aller in Director® verfügbarer Konstanten.

Der Großteil dieser Konstanten gilt nur für Lingo. In der JavaScript-Syntax gibt es einige Konstanten, die den hier aufgeführten Lingo-Konstanten ähneln. Deshalb werden, wo angebracht, auch JavaScript-Syntaxverwendung und -beispiele beschrieben, damit Sie die Funktionalität von Lingo-Konstanten ihren Entsprechungen in der JavaScript-Syntax zuordnen können. Weitere Informationen zu Konstanten für die JavaScript-Syntax finden Sie in den vielen, von Drittanbietern erstellten Materialien zu diesem Thema.

" (string)

Syntax

```
--Lingo syntax
"

// JavaScript syntax
"
```

Beschreibung

Die Anführungszeichen am Anfang und Ende eines Strings geben an, dass dieser ein Literal ist – keine Variable, kein numerischer Wert und auch kein Skriptelement. Literalnamen von Darstellern, Besetzungen, Fenstern und externen Dateien müssen immer in Anführungszeichen stehen.

Beispiel

Die folgende Anweisung verwendet Anführungszeichen, um anzugeben, dass der String „San Francisco“ ein Literalstring ist, in diesem Fall der Name eines Darstellers:

```
--Lingo syntax
put member("San Francisco").loaded

// JavaScript syntax
put(member("San Francisco").loaded);
```

Siehe auch

QUOTE

BACKSPACE

Syntax

```
-- Lingo syntax
BACKSPACE

// JavaScript syntax
51 // value of _key.keyCode
```

Beschreibung
Diese Konstante stellt die Rücktaste dar. Diese Taste hat die Bezeichnung „Rücktaste“ (Windows®) bzw. „Löschtaste“ (Macintosh®).

Beispiel
Die folgende `on keyDown`-Prozedur prüft, ob die Rücktaste gedrückt wurde, und ruft die Prozedur `clearEntry` auf, wenn dies der Fall ist:

```
--Lingo syntax
on keyDown
    if (_key.key = BACKSPACE) then clearEntry
    _movie.stopEvent()
end keyDown

// JavaScript syntax
function keyDown() {
    if (_key.keyCode == 51) {
        clearEntry();
        _movie.stopEvent();
    }
}
```

EMPTY

Syntax

```
--Lingo syntax
EMPTY

// JavaScript syntax
""
```

Beschreibung
Diese Zeichenkonstante steht für einen leeren String, `""`, also eine Zeichenfolge ohne Zeichen.

Beispiel
Die folgende Anweisung löscht alle Zeichen im Felddarsteller „Hinweis“, indem das Feld auf `EMPTY`eingestellt wird:

```
--Lingo syntax
member("Notice").text = EMPTY

// JavaScript syntax
member("Notice").text = "";
```

ENTER

Syntax

```
--Lingo syntax
ENTER

// JavaScript syntax
3 // value of _key.keyCode
```

Beschreibung

Diese Zeichenkonstante steht für die Eingabetaste bei einer Zeilenschaltung.

Auf einer PC-Tastatur verweist das Element `ENTER` nur auf die Eingabetaste desZiffernblocks.

Bei einem Film, der als Applet abgespielt wird, geben Sie mit `RETURN` die Eingabetaste sowohl in Windows als auch auf dem Macintosh an.

Beispiel

Die folgende Anweisung überprüft, ob die Eingabetaste gedrückt ist, und schickt den Abspielkopf zum Bild `addSum`, wenn dies der Fall ist:

```
-- Lingo syntax
on keyDown
    if (_key.key = ENTER) then _movie.go("addSum")
end

// JavaScript syntax
function keyDown() {
    if (_key.keyCode == 3) {
        _movie.go("addSum");
    }
}
```

Siehe auch

RETURN (constant)

FALSE

Syntax

```
-- Lingo syntax
FALSE

// JavaScript syntax
false
```

Beschreibung

Diese Konstante trifft auf einen Ausdruck zu, der logisch `FALSE` (falsch) ist, wie z. B. „2 > 3“. Als Zahlenwert hat `FALSE` den numerischen Wert 0. Umgekehrt kann auch 0 (Null) als `FALSE` behandelt werden.

Beispiel

Die folgende Anweisung deaktiviert die Eigenschaft `soundEnabled`, indem sie diese auf `FALSE` setzt:

```
-- Lingo syntax
_sound.soundEnabled = FALSE

// JavaScript syntax
_sound.soundEnabled = false;
```

Siehe auch

`if`, `not`, `TRUE`

PI

Syntax

```
-- Lingo syntax
PI

// JavaScript syntax
Math.PI
```

Beschreibung

Diese Konstante gibt den Wert von Pi (p) alsFließkommazahl zurück. (Pi ist das Verhältnis zwischen Kreisumfang und Kreisdurchmesser.) Der Wert wird auf die Anzahl von Dezimalstellen gerundet, die in der Eigenschaft `floatPrecision` festgelegt ist.

Beispiel

Die folgende Anweisung verwendet die Konstante `PI` als Teil einer Gleichung zur Berechnung der Kreisfläche:

```
-- Lingo syntax
vRadius = 3
vArea = PI*power(vRadius, 2)
trace(vArea) -- results in 28.2743

// JavaScript syntax
var vRadius = 3;
vArea = Math.PI*Math.pow(vRadius, 2);
trace(vArea); // results in 28.274333882308138
```

QUOTE

Syntax

```
--Lingo syntax
QUOTE

// JavaScript syntax
\"
```

Beschreibung
Diese Konstante stellt das Anführungszeichen dar und verweist auf ein literales Anführungszeichen in einem String, da das eigentliche Anführungszeichen von Lingo-Skripts zur Begrenzung von Strings verwendet wird.

Beispiel
Die folgende Anweisung fügt Anführungszeichen in einen String ein:

```
-- Lingo syntax
put("Can you spell" && QUOTE & "Adobe" & QUOTE & "?")

// JavaScript syntax
put("Can you spell \"Adobe\"?");
```

Dadurch wird das Wort Adobe® in Anführungszeichen gesetzt:

```
Can you spell "Adobe"?
```

RETURN (constant)

Syntax

```
-- Lingo syntax
RETURN

// JavaScript syntax
36 // value of _key.keyCode
\n // when used in a string
```

Beschreibung
Diese Konstante stellt eine Zeilenschaltung (Carriage Return) dar.

Beispiel
Die folgende Anweisung setzt einen unterbrochenen Film fort, sobald der Benutzer die Eingabetaste drückt:

```
-- Lingo syntax
if (_key.key = RETURN) then _movie.go(_movie.frame + 1)

// JavaScript syntax
if (_key.keyCode == 36) {
    _movie.go(_movie.frame + 1);
}
```

Die folgende Anweisung verwendet die Zeichenkonstante `RETURN` dazu, um in einer Hinweismeldung eine Zeilenschaltung zwischen zwei Zeilen einzufügen:

```
-- Lingo syntax
_player.alert("Last line in the file." & RETURN & "Click OK to exit.")

// JavaScript syntax
_player.alert("Last line in the file." + "\n" + " Click OK to exit");
```

In Windows wird in der Regel am Ende jeder Zeile ein zusätzliches Zeilenvorschubzeichen eingefügt. Die folgende Anweisung erstellt den aus zwei Zeichen bestehenden String CRLF, der für den zusätzlichen Zeilenvorschub sorgt:

```
CRLF = RETURN & numToChar(10)
```

SPACE

Syntax

```
-- Lingo syntax
SPACE

// JavaScript syntax
49 // value of _key.keyCode
```

Beschreibung

Diese Konstante ist ein schreibgeschützter Wert, der das Leerzeichen darstellt.

Beispiel

Die folgende Anweisung zeigt „Age Of Aquarius“ im Nachrichtenfenster an:

```
-- Lingo syntax
put("Age"&SPACE&"Of"&SPACE&"Aquarius")
```

TAB

Syntax

```
-- Lingo syntax
TAB

// JavaScript syntax
48 // value of _key.keyCode
```

Beschreibung

Diese Konstante stellt die Tabulatortaste dar.

Beispiel

Die folgende Anweisung prüft, ob das eingegebene Zeichen das Tabulatorzeichen ist, und ruft die Prozedur `doNextField` auf, wenn dies der Fall ist:

```
-- Lingo syntax
if (_key.key = TAB) then doNextField

// JavaScript syntax
if (_key.keyCode == 48) {
    doNextField();
}
```

Die folgenden Anweisungen verschieben den Abspielkopf vorwärts oder rückwärts, je nachdem, ob der Benutzer die Tabulatortaste oder die Umschalt- und Tabulatortaste drückt:

```
-- Lingo syntax
if (_key.key = TAB) then
    if (_key.shiftDown) then
        _movie.go(_movie.frame - 1)
    else
        _movie.go(_movie.frame + 1)
    end if
end if

// JavaScript syntax
if (_key.keyCode == 48) {
    if (_key.shiftDown) {
        _movie.go(_movie.frame - 1);
    } else {
        _movie.go(_movie.frame + 1);
    }
}
```

Siehe auch

BACKSPACE, EMPTY, RETURN (constant)

TRUE

Syntax

```
-- Lingo syntax
TRUE

// JavaScript syntax
true
```

Beschreibung

Diese Konstante stellt den Wert eines logisch wahren Ausdrucks dar, z. B. `2 < 3`. Sie hat den nummerischen Wert von 1, aber in einem Vergleich wird jeder Integerwert (außer 0) als `TRUE` ausgewertet.

Beispiel

Die folgende Anweisung setzt die Eigenschaft `soundEnabled` zur Aktivierung auf `TRUE`:

```
-- Lingo syntax
_sound.soundEnabled = TRUE

// JavaScript syntax
_sound.soundEnabled = true;
```

Siehe auch

FALSE, if

VOID

Syntax

```
-- Lingo syntax
VOID

// JavaScript syntax
null
```

Beschreibung

Diese Konstante gibt den Wert `VOID` an.

Beispiel

Die folgende Anweisung prüft, ob der Wert in der Variablen `currentVariableVOID` ist:

```
-- Lingo syntax
if currentVariable = VOID then
    put("This variable has no value")
end if

// JavaScript syntax
if (currentVariable == undefined) {
    put("This variable has no value");
}
```

Siehe auch

`voidP()`

Kapitel 10: Ereignisse und Nachrichten

In diesem Abschnitt finden Sie eine alphabetische Liste aller in Director® verfügbarer Ereignisse und Nachrichten.

on activateApplication

Syntax

```
-- Lingo syntax
on activateApplication
    statement(s)
end

// JavaScript syntax
function activateApplication() {
    statement(s);
}
```

Beschreibung

Diese integrierte Prozedur wird ausgeführt, wenn der Benutzer den Projektor in den Vordergrund holt. Dies ist insbesondere dann sinnvoll, wenn ein Projektor in einem Fenster ausgeführt wird, das der Benutzer in den Hintergrund stellen kann, um mit anderen Anwendungen zu arbeiten. Wenn er anschließend wieder zum Projektor umschaltet, wird diese Prozedur ausgeführt. MIAWs, die im Projektor laufen, können diese Prozedur ebenfalls nutzen.

Beim Authoring wird diese Prozedur nur aufgerufen, wenn im Dialogfeld Allgemeine Voreinstellungen die Option Im Hintergrund animieren aktiviert ist.

Unter Windows® wird diese Prozedur nicht aufgerufen, wenn der Benutzer den Projektor lediglich minimiert, ohne eine andere Anwendung in den Vordergrund zu holen.

Beispiel

Die folgende Prozedur spielt einen Sound ab, wenn der Benutzer den Projektor wieder in den Vordergrund holt:

```
-- Lingo syntax
on activateApplication
    sound(1).queue(member("openSound"))
    sound(1).play()
end

// JavaScript syntax
function activateApplication() {
    sound(1).queue(member("openSound"));
    sound(1).play();
}
```

Siehe auch

on deactivateApplication, activeCastLib, on deactivateWindow

on activateWindow

Syntax

```
-- Lingo syntax
on activateWindow
        statement(s)
end

// JavaScript syntax
function activateWindow()
    statement(s);
}
```

Beschreibung

Diese Systemnachricht und Ereignisprozedur enthält Anweisungen, die ausgeführt werden, wenn ein Film in einem Fenster abläuft und der Benutzer das Fenster in den Vordergrund holt.

Sie können eine `on activateWindow`-Prozedur in einem Skript verwenden, das immer dann ausgeführt werden soll, wenn der Film aktiv wird.

Wenn Sie auf den Hauptfilm (die Hauptbühne) klicken, wird keine `on activateWindow`-Prozedur erstellt.

Beispiel

Die folgende Prozedur spielt den Sound „Hurra“ ab, wenn das Fenster, in dem der Film läuft, aktiv wird:

```
-- Lingo syntax
on activateWindow
    sound(2).play(member("Hurray"))
end

// JavaScript syntax
function activateWindow() {
    sound(2).play(member("Hurray"));
}
```

Siehe auch

activeWindow, close(), on deactivateWindow, frontWindow, on moveWindow, open() (Fenster)

on beginSprite

Syntax

```
-- Lingo syntax
on beginSprite
    statement(s)
end

// JavaScript syntax
function beginSprite() {
    statement(s);
}
```

Beschreibung

Diese Systemnachricht und Ereignisprozedur enthält Anweisungen, die dann ausgeführt werden, wenn der Abspielkopf zu einem Bild übergeht, das ein erstmalig auftretendes Sprite enthält. Wie im Falle von `endSprite` wird auch dieses Ereignis nur einmal generiert, selbst wenn der Abspielkopf in einem Bild in Schleife abgespielt wird, da der Auslöser ein Sprite ist, auf das der Abspielkopf bis dahin nicht gestoßen ist. Das Ereignis wird vor `prepareFrame` generiert.

Director erstellt Instanzen aller dem Sprite zugewiesener Verhaltensskripten, wenn die Nachricht `beginSprite` gesendet wird.

Der Objektbezug `me` wird an dieses Ereignis übergeben, wenn es in einem Verhalten verwendet wird. Die Nachricht wird an Verhalten und Bildskripten gesendet.

Wenn ein Sprite im ersten Bild eines Films beginnt, wird die Nachricht `beginSprite` nach der Nachricht `prepareMovie`, jedoch noch vor den Nachrichten `prepareFrame` und `startMovie` gesendet.

Hinweis: *Auf gewisse Sprite-Eigenschaften, wie z. B. auf die Sprite-Eigenschaft `rect`, kann in einer beginSprite-Prozedur eventuell nicht zugegriffen werden. Der Grund dafür ist, dass die Eigenschaft berechnet werden muss, was erst nach dem Zeichnen des Sprites möglich ist.*

Die Befehle `go`, `play` und `updateStage` sind in einer `on beginSprite`-Prozedur deaktiviert.

Beispiel

Die folgende Prozedur spielt den Sounddarsteller „Stevie Wonder“ ab, wenn das Sprite beginnt:

```
-- Lingo syntax
on beginSprite me
    sound(1).play(member("Stevie Wonder"))
end

// JavaScript syntax
function beginSprite() {
    sound(1).play(member("Stevie Wonder"));
}
```

Siehe auch

`on endSprite`, `on prepareFrame`, `scriptInstanceList`

on closeWindow

Syntax

```
-- Lingo syntax
on closeWindow
    statement(s)
end

// JavaScript syntax
function closeWindow() {
    statement(s);
}
```

Beschreibung

Diese Systemnachricht und Ereignisprozedur enthält Anweisungen, die ausgeführt werden, wenn der Benutzer das Fenster für einen Film schließt, indem er auf die entsprechende Schaltfläche in der Titelleiste des Fensters klickt.

Die Prozedur `on closeWindow` eignet sich für Lingo-Befehle, die beim Schließen des Filmfensters grundsätzlich ausgeführt werden sollen.

Beispiel

Die folgende Prozedur weist Director mit `forget` an, das aktuelle Fenster aus dem Speicher zu entfernen, wenn der Benutzer das Wiedergabefenster schließt.

```
-- Lingo syntax
on closeWindow
    -- perform general housekeeping here
    window(1).forget()
end

// JavaScript syntax
function closeWindow() {
    // perform general housekeeping here
    window(1).forget();
}
```

on cuePassed

Syntax

```
-- Lingo syntax
on cuePassed({me,} channelID, cuePointNumber,cuePointName)
    statement(s)
end

// JavaScript syntax
function cuePassed(channelID, cuePointNumber,cuePointName) {
    statement(s);
}
```

Beschreibung

Diese Systemnachricht und Ereignisprozedur enthält Anweisungen, die immer dann ausgeführt werden, wenn ein Sound oder ein Sprite in seinen Medien einen Aufrufpunkt passiert.

- `me` Der optionale Parameter `me` ist der `scriptInstanceRef`-Wert des aufgerufenen Skripts. Sie müssen diesen Parameter angeben, wenn Sie die Nachricht in einem Verhalten verwenden. Wenn Sie den Parameter weglassen, werden die anderen Argumente nicht richtig verarbeitet.
- *`channelID`* – Die Nummer des Sound- oder Sprite-Kanals für die Datei, in der der Aufrufpunkt enthalten ist.
- *`cuePointNumber`* – Die Ordnungszahl des Aufrufpunkts in der Liste der Aufrufpunkte des Darstellers, der das Ereignis auslöst.
- *`cuePointName`* – Der Name des gefundenen Aufrufpunkts.

Die Nachricht wird – in dieser Reihenfolge – an Sprite-, Darsteller-, Bild- und Filmskripten übergeben. Damit das Sprite das Ereignis empfangen kann, muss es sich bei ihm um die Quelle des Sounds handeln, z. B. um einen QuickTime-Film oder einen SWA-Darsteller.® Verwenden Sie die Eigenschaft `isPastCuePoint`, um Aufrufe in an Sprites angebrachten Verhalten zu prüfen, die keine Sounds erzeugen.

Beispiel

Die folgende Prozedur in einem Film- oder Bildskript zeigt alle Aufrufpunkte in Soundkanal 1 im Nachrichtenfenster an:

```
-- Lingo syntax
on cuePassed channel, number, name
    if (channel = #Sound1) then
        put("CuePoint" && number && "named" && name && "occurred in sound 1")
    end if
end

// JavaScript syntax
function cuePassed(channel, number, name) {
    if (channel == symbol("Sound1")) {
        put("CuePoint " + number + " named " + name + "occurred in sound 1");
    }
}
```

Siehe auch

`scriptInstanceList`, `cuePointNames`, `cuePointTimes`, `isPastCuePoint()`

on deactivateApplication

Syntax

```
-- Lingo syntax
on deactivateApplication
    statement(s)
end

// JavaScript syntax
function deactivateApplication() {
    statement(s);
}
```

Beschreibung

Diese integrierte Prozedur wird ausgeführt, wenn der Benutzer den Projektor in den Hintergrund stellt. Dies ist insbesondere dann sinnvoll, wenn ein Projektor in einem Fenster ausgeführt wird, das der Benutzer in den Hintergrund stellen kann, um mit anderen Anwendungen zu arbeiten. MIAWs, die im Projektor laufen, können diese Prozedur ebenfalls nutzen.

Beim Authoring wird diese Prozedur nur aufgerufen, wenn im Dialogfeld Allgemeine Voreinstellungen die Option Im Hintergrund animieren aktiviert ist.

Unter Windows wird diese Prozedur nicht aufgerufen, wenn der Benutzer den Projektor lediglich minimiert, ohne eine andere Anwendung in den Vordergrund zu holen.

Beispiel

Die folgende Prozedur spielt einen Sound ab, wenn der Benutzer den Projektor in den Hintergrund stellt:

```
-- Lingo syntax
on deactivateApplication
    sound(1).queue(member("closeSound"))
    sound(1).play()
end

// JavaScript syntax
function deactivateApplication() {
    sound(1).queue(member("closeSound"));
    sound(1).play();
}
```

Siehe auch

add (3D-Textur), activeCastLib, on deactivateWindow

on deactivateWindow

Syntax

```
-- Lingo syntax
on deactivateWindow
    statement(s)
end

// JavaScript syntax
function deactivateWindow() {
    statement(s);
}
```

Beschreibung

Diese Systemnachricht und Ereignisprozedur enthält Anweisungen, die dann ausgeführt werden, wenn das Wiedergabefenster des Films deaktiviert wird. Die Ereignisprozedur `on deactivate` eignet sich gut für Lingo-Code, der immer dann ausgeführt werden soll, wenn ein Fenster deaktiviert wird.

Beispiel

Die folgende Prozedur spielt den Sound „Schnarchen“ ab, wenn das Wiedergabefenster des Films deaktiviert wird.

```
-- Lingo syntax
on deactivateWindow
    sound(2).play(member("Snore"))
end

// JavaScript syntax
function deactivateWindow() {
    sound(2).play(member("Snore"));
}
```

on DVDeventNotification

Syntax

```
-- Lingo syntax
on DVDeventNotification objectRef, event {, eventArg1} {, eventArg2} {, eventArg3}
    statement(s)
end DVDeventNotification

// JavaScript syntax
function DVDeventNotification (objectRef, event {, eventArg1} {, eventArg2} {, eventArg3}) {
    statement(s);
}
```

Beschreibung

Diese vom Autor angegebene DVD-Ereignisprozedur enthält Anweisungen, die als Reaktion auf Ereignisse bei der DVD-Wiedergabe ausgeführt werden.

Mit dieser Prozedur können sämtliche DVD-Ereignisse verfolgt werden. In den obigen Skriptbeispielen handelt es sich bei *objectRef*, dem ersten an die DVDeventNotification-Prozedur übergebenen Parameter, um einen Verweis auf das DVDeventNotification-Objekt als solches. Das konkret aufgetretene Ereignisse wird immer als der zweite Parameter, *event*, übergeben. Manche Ereignisse enthalten zusätzliche Ereignisinformationen, die in einem dritten Parameter, *eventArg1*, übergeben werden. Manchmal enthalten auch ein vierter und fünfter Parameter, *eventArg2* und *eventArg3*, zusätzliche Ereignisinformationen.

Die folgende Tabelle enthält eine Liste der Ereignisse, die bei der DVD-Wiedergabe auftreten können.

Ereignis	**Beschreibung**
angleChange	Tritt auf, wenn sich die Anzahl verfügbarer Winkel oder die Nummer des aktuellen Benutzerwinkels ändert. Bei diesem Ereignis werden die folgenden, zusätzlichen Informationen an `DVDeventNotification` übergeben: • *eventArg2* – Eine Ganzzahl, welche die Anzahl der verfügbaren Winkel angibt. Beträgt die Anzahl der verfügbaren Winkel 1, ist das aktuelle Video nicht mehrwinklig. • *eventArg3* – Eine Ganzzahl, welche die Nummer des aktuellen Benutzerwinkels angibt.
audioStreamChange	Tritt auf, wenn sich die Nummer des aktuellen Benutzer-Audiostreams für den Haupttitel ändert. Bei diesem Ereignis werden die folgenden, zusätzlichen Informationen an `DVDeventNotification` übergeben: • *eventArg2* – Eine Ganzzahl, welche die Nummer des neuen Benutzer-Audiostreams angibt. Stream `0xFFFFFFFF` bedeutet, dass kein Stream ausgewählt ist.
buttonChange	Tritt auf, wenn sich die Anzahl verfügbarer Schaltflächen oder die Nummer der aktuell ausgewählten Schaltfläche ändert. Bei diesem Ereignis werden die folgenden, zusätzlichen Informationen an `DVDeventNotification` übergeben: • *eventArg2* – Eine Ganzzahl, welche die Anzahl der verfügbaren Schaltflächen angibt. • *eventArg3* – Eine Ganzzahl, welche die Nummer der aktuell ausgewählten Schaltfläche angibt. Wird hier der Wert 0 angegeben, bedeutet dies, dass keine Schaltfläche ausgewählt ist.
chapterAutoStop	Tritt auf, wenn die Wiedergabe aufgrund eines automatischen Stopps angehalten wird.

Ereignis	**Beschreibung**
chapterStart	Tritt auf, wenn die Wiedergabe eines neuen Programm in der Domäne `title` beginnt. Bei diesem Ereignis werden die folgenden, zusätzlichen Informationen an `DVDeventNotification` übergeben: • *eventArg2* – Eine Ganzzahl, welche die neue Kapitelnummer angibt.
diskEjected	Tritt auf, wenn eine DVD ausgeworfen wird.
diskInserted	Tritt auf, wenn eine DVD eingelegt wird.
domainChange	Tritt auf, wenn sich die Domäne des DVD-Players ändert. Bei diesem Ereignis werden die folgenden, zusätzlichen Informationen an `DVDeventNotification` übergeben: • *eventArg1*: Ein Wert, welcher die neue Domäne angibt. Als neue Domäne wird einer der folgenden Werte angegeben: • `firstPlay`: Der DVD-Navigator führt eine Standardinitialisierung einer DVD durch. • `videoManagerMenu`: Der DVD-Navigator zeigt Menüs für die gesamte DVD an. • `videoTitleSetMenu`: Der DVD-Navigator zeigt Menüs für den aktuellen Titelsatz an. • `title`. Der DVD-Navigator zeigt den aktuellen Titel an. • `stop`. Der DVD-Navigator befindet sich in der `stop`-Domäne.
error	Tritt auf, wenn ein DVD-Fehlerzustand festgestellt wird. Bei diesem Ereignis werden die folgenden, zusätzlichen Informationen an `DVDeventNotification` übergeben: • *eventArg2*: Ein Wert, der einen Fehlerzustand angibt. Als Fehlerzustand wird einer der folgenden Werte angegeben: • `copyProtectFail`: Schlüsselaustausch für „DVD-Kopierschutz fehlgeschlagen“. Die Wiedergabe wird angehalten. • `invalidDVD1_0Disc`: Die DVD-Videodisk wurde nicht ordnungsgemäß für Spezifikation, Version 1.*x* erstellt. Die Wiedergabe wird angehalten. • `invalidDiscRegion`: Die DVD-Videodisk kann nicht wiedergegeben werden, da sie nicht für die Wiedergabe im Systembereich erstellt wurde. • `lowParentalLevel`: Die Kindersicherungsstufe des Players ist niedriger als die niedrigstmögliche Kindersicherungsstufe des DVD-Inhalts. Die Wiedergabe wird angehalten. • `macrovisionFail`: Fehler bei Macrovision-Schutz. Die Wiedergabe wird angehalten. • `incompatibleSystemAndDecoderRegions`. Die DVD-Wiedergabe ist nicht möglich, da der Systembereich nicht mit dem Decoderbereich übereinstimmt. • `incompatibleDiscAndDecoderRegions`. Die DVD-Wiedergabe ist nicht möglich, da die DVD nicht für die Wiedergabe im Decoderbereich erstellt wurde. • `unexpected`. Ein unerwarteter Fehler trat auf. Eventuell wurde der DVD-Inhalt nicht korrekt erstellt. Die Wiedergabe wird angehalten.
karaokeMode	Tritt auf, wenn der Audiomodus auf `karaoke` eingestellt ist.
noFirstPlayChain	Tritt auf, wenn die DVD keine First Play Program Chain (FP_PGC) enthält und der DVD-Navigator nicht automatisch jede PGC lädt und die Wiedergabe startet.

Ereignis	**Beschreibung**
parentalLevelChange	Tritt auf, wenn sich die Kindersicherungsstufe des erstellten Inhalts demnächst ändert. Bei diesem Ereignis werden die folgenden, zusätzlichen Informationen an `DVDeventNotification` übergeben: • *eventArg2*: Eine Ganzzahl, welche die neue, im Player eingestellte Kindersicherungsstufe angibt.
playbackStopped	Tritt auf, wenn die Wiedergabe angehalten wird. Der DVD-Navigator hat die Wiedergabe der PGC abgeschlossen und fand keine weiteren Verzweigungsanweisungen für die nachfolgende Wiedergabe.
playPeriodAutoStop	Tritt auf, wenn die Wiedergabe aufgrund eines automatischen Stopps angehalten wird.
rateChange	Tritt auf, wenn sich die Wiedergabegeschwindigkeit ändert. Bei diesem Ereignis werden die folgenden, zusätzlichen Informationen an `DVDeventNotification` übergeben: • *eventArg2*: Eine Ganzzahl, welche die neue Wiedergabegeschwindigkeit angibt. Ein Wert unter (<) 0 bedeutet Rückwärtswiedergabe. Ein Wert über (>) 0 bedeutet Vorwärtswiedergabe. Dieser Wert ist die tatsächliche Wiedergabegeschwindigkeit multipliziert mit 10.000.
stillOff	Tritt am Ende eines Standbilds auf (PGC, Cell oder VOBU).
stillOn	Tritt am Anfang eines Standbilds auf (PGC, Cell oder VOBU). Bei diesem Ereignis werden die folgenden, zusätzlichen Informationen an `DVDeventNotification` übergeben: • *eventArg2* – Ein Boolescher Wert, der angibt, ob Schaltflächen zur Verfügung stehen. Null (0) bedeutet, dass Schaltflächen verfügbar sind. Eins (1) gibt an, dass keine Schaltflächen verfügbar sind. • *eventArg3* – Eine Ganzzahl oder Adresse, welche die Dauer des Standbilds in Sekunden angibt. `0xFFFFFFFF` gibt ein unendliches Standbild an.

Ereignis	Beschreibung
titleChange	Tritt auf, wenn sich die aktuelle Titelnummer ändert. Bei diesem Ereignis werden die folgenden, zusätzlichen Informationen an `DVDeventNotification` übergeben: • *eventArg2* – Eine Ganzzahl oder Adresse, welche die neue Titelnummer angibt.
UOPchange	Tritt auf, wenn sich einer der verfügbaren Wiedergabe- oder Suchmechanismen ändert. Bei diesem Ereignis werden die folgenden, zusätzlichen Informationen an `DVDeventNotification` übergeben: • *eventArg2* – Eine Ganzzahl oder Adresse, welche angibt, welche Wiedergabe- oder Suchmechanismen die DVD expliziert deaktiviert.
warning	Tritt auf, wenn ein DVD-Warnzustand festgestellt wird. Bei diesem Ereignis werden die folgenden, zusätzlichen Informationen an `DVDeventNotification` übergeben: • *eventArg2*– Eine Ganzzahl oder Adresse, welche den Warnzustand angibt. Als Warnzustand wird einer der folgenden Werte angegeben: • `invalidDVD1_0Disc`: Die DVD-Videodisk wurde nicht ordnungsgemäß erstellt. Die Wiedergabe kann zwar fortgesetzt werden, es kann jedoch zu unerwartetem Verhalten kommen. • `formatNotSupported`: Ein Decoder unterstützt das aktuelle Format nicht. Die Wiedergabe eines Streams funktioniert möglicherweise nicht. • `illegalNavCommand`: Der interne DVD-Navigator versucht, einen unzulässigen Befehl zu verarbeiten. • `open` • `seek` • `read`

Siehe auch

DVD

on endSprite

Syntax

```
-- Lingo syntax
on endSprite
    statement(s)
end

// JavaScript syntax
function endSprite() {
    statement(s);
}
```

Beschreibung

Diese Systemnachricht und Ereignisprozedur enthält Lingo-Code, der ausgeführt wird, wenn der Abspielkopf ein Sprite verlässt und zu einem Bild geht, in dem das Sprite nicht vorhanden ist. Sie wird nach Ablauf von `exitFrame` erzeugt.

Stellen Sie `on endSprite`-Prozeduren in ein Verhaltensskript.

Director löscht sofort die Instanzen aller am Sprite angebrachten Verhaltensskripts, sobald das Ereignis `endSprite` auftritt.

Der Ereignisprozedur wird das Verhalten oder der Bildskriptbezug `me` übergeben, falls sie in einem Verhalten verwendet wird. Diese `endSprite`-Nachricht wird nach der `exitFrame`-Nachricht gesendet, wenn der Abspielkopf bis zum Ende des Bildes läuft.

Die Methoden `go()`, `play()` und `updateStage()` sind in einer `on endSprite`-Prozedur deaktiviert.

Beispiel

Die folgende Prozedur läuft ab, wenn der Abspielkopf ein Sprite verlässt:

```
-- Lingo syntax
on endSprite me
    -- clean up
    gNumberOfSharks = gNumberOfSharks - 1
    sound(5).stop()
end

// JavaScript syntax
function endSprite() {
    // clean up
    gNumberOfSharks--;
    sound(5).stop();
}
```

Siehe auch

`on beginSprite`, `on exitFrame`

on enterFrame

Syntax

```
-- Lingo syntax
on enterFrame
    statement(s)
end

// JavaScript syntax
function enterFrame() {
    statement(s);
}
```

Beschreibung

Diese Systemnachricht und Ereignisprozedur enthält Anweisungen, die ausgeführt werden, wenn der Abspielkopf in das Bild eintritt.

Stellen Sie die Prozedur `on enterFrame` wie folgt in Verhaltens-, Bild- oder Filmskripten:

- Wenn die Prozedur einem einzelnen Sprite zugeordnet werden soll, stellen Sie die Prozedur in ein am Sprite angebrachtes Verhalten.
- Wenn die Prozedur einem einzelnen Bild zugeordnet werden soll, stellen Sie die Prozedur in das Bildskript.
- Soll die Prozedur jedem Bild zugeordnet werden (falls Sie dem Film nicht ausdrücklich andere Weisungen geben), stellen Sie die Prozedur `on enterFrame` in ein Filmskript. Die Prozedur wird immer dann ausgeführt, wenn der Abspielkopf in ein Bild eintritt, es sei denn, das Bildskript verfügt über eine eigene Prozedur. Wenn das Bildskript eine eigene Prozedur besitzt, setzt die `on enterFrame`-Prozedur im Bildskript die `on enterFrame`-Prozedur im Filmskript außer Kraft.

Die Reihenfolge der Bildereignisse lautet `stepFrame`, `prepareFrame`, `enterFrame` und `exitFrame`.

Diesem Ereignis wird der Objektbezug `me` übergeben, wenn es in einem Verhalten verwendet wird.

Beispiel

Die folgende Prozedur schaltet die Puppet-Bedingung für die Sprites 1 bis 5 jedes Mal aus, wenn der Abspielkopf in das Bild eintritt:

```
-- Lingo syntax
on enterFrame
    repeat with i = 1 to 5
        _movie.puppetSprite(i, FALSE)
    end repeat
end

// JavaScript syntax
function enterFrame() {
    for (i=1;i<=5;i++) {
        _movie.puppetSprite(i, false);
    }
}
```

on EvalScript

Syntax

```
-- Lingo syntax
on EvalScript aParam
    statement(s)
end

// JavaScript syntax
function EvalScript(aParam) {
    statement(s);
}
```

Beschreibung

Diese Systemnachricht und Ereignisprozedur in einem Film mit Adobe® Shockwave®-Inhalt enthält Anweisungen, die ausgeführt werden, wenn die Prozedur eine `EvalScript`-Nachricht von einem Browser erhält. Der Parameter ist ein String, der vom Browser übergeben wird.

- Die Nachricht `EvalScript` kann einen String enthalten, den Director als Lingo-Anweisung interpretiert. Lingo kann keine verschachtelten Strings akzeptieren. Wenn die von Ihnen aufgerufene Prozedur einen String als Parameter erwartet, muss der Parameter als Symbol übergeben werden.
- Die Prozedur `on EvalScript` wird durch die Skripterstellungsmethode `EvalScript()` in einem JavaScript- oder VBScript-Skript in einem Browser aufgerufen.

Schließen Sie nur die Verhalten in `on EvalScript` ein, die vom Benutzer gesteuert werden sollen. Geben Sie aus Sicherheitsgründen nicht den Zugriff auf alle Verhalten frei.

Hinweis: *Wenn Sie eine return-Funktion an das Ende der EvalScript-Prozedur stellen, kann der zurückgegebene Wert von JavaScript im Browser verwendet werden.*

Beispiel

Das folgende Beispiel zeigt, wie Sie den Abspielkopf auf ein bestimmtes Bild setzen können, das als Parameter übergeben wird:

```
-- Lingo syntax
on EvalScript aParam
    _movie.go(aParam)
end

// JavaScript syntax
function EvalScript(aParam) {
    _movie.go(aParam);
}
```

Die folgende Prozedur führt die Anweisung `_movie.go(aParam)` aus, wenn diese eine `EvalScript`-Nachricht empfängt, die „Hund“, „Katze“ oder „Baum“ als Argument enthält:

```
-- Lingo syntax
on EvalScript aParam
    case aParam of
        "dog", "cat", "tree": _movie.go(aParam)
    end case
end

// JavaScript syntax
function EvalScript(aParam) {
    switch(aParam) {
        case "dog", "cat", "tree": _movie.go(aParam);
    }
}
```

In JavaScript wäre eine mögliche Aufrufanweisung dafür `EvalScript ("dog")`.

Das Argument der folgenden Prozedur kann eine Zahl oder ein Symbol sein:

```
-- Lingo syntax
on EvalScript aParam
    if word 1 of aParam = "myHandler" then
        _movie.go(aParam)
    end if
end

// JavaScript syntax
function EvalScript(aParam) {
    if (aParam.indexOf("myHandler",0)) {
        _movie.go(aParam);
    }
}
```

Die folgende Prozedur erfordert normalerweise einen String als Argument. Das Argument wird als Symbol empfangen und dann durch die Funktion `string` in einen String in der Prozedur konvertiert:

```
-- Lingo syntax
on myHandler aParam
    _movie.go(string(aParam))
end

// JavaScript syntax
function myHandler(aParam) {
    _movie.go(aParam.toString());
}
```

Siehe auch

`externalEvent()`, `return (Schlüsselwort)`

on exitFrame

Syntax

```
-- Lingo syntax
on exitFrame
    statement(s)
end

// JavaScript syntax
function exitFrame() {
    statement(s);
}
```

Beschreibung

Diese Systemnachricht und Ereignisprozedur enthält Anweisungen, die immer dann ausgeführt werden, wenn der Abspielkopf das Bild verlässt, an dem die Prozedur `on exitFrame` angebracht ist. Die Prozedur `on exitFrame` eignet sich besonders für Lingo-Code, der Bedingungen zurücksetzt, die nach dem Verlassen des Bildes nicht mehr zutreffen.

Stellen Sie `on exitFrame`-Prozeduren wie folgt in Verhalten-, Bild- oder Filmskripten:

- Wenn die Prozedur einem einzelnen Sprite zugeordnet werden soll, stellen Sie die Prozedur in ein am Sprite angebrachtes Verhalten.
- Wenn die Prozedur einem einzelnen Bild zugeordnet werden soll, stellen Sie die Prozedur in das Bildskript.

- Wenn die Prozedur jedem Bild zugeordnet werden soll, das nicht ausdrücklich davon ausgeschlossen wurde, stellen Sie die Prozedur in ein Filmskript. Die Prozedur `on exitFrame` wird jedes Mal ausgeführt, wenn der Abspielkopf das Bild verlässt, es sei denn, das Bildskript verfügt über eine eigene `on exitFrame`-Prozedur. In diesem Fall setzt die `on exitFrame`-Prozedur im Bildskript die `on exitFrame`-Prozedur im Filmskript außer Kraft.

Diesem Ereignis wird der Sprite-Skript- oder Bildskript-Bezug `me` übergeben, wenn es in einem Verhalten verwendet wird. Die Reihenfolge der Bildereignisse lautet `prepareFrame`, `enterFrame` und `exitFrame`.

Beispiel

Die folgende Prozedur schaltet alle Puppet-Bedingungen aus, wenn der Abspielkopf das Bild verlässt:

```
-- Lingo syntax
on exitFrame me
    repeat with i = 48 down to 1
        sprite(i).scripted = FALSE
    end repeat
end

// JavaScript syntax
function exitFrame() {
    for (i=48; i>=1; i--);
        sprite(i).scripted = false;
    }
}
```

Die folgende Prozedur schickt den Abspielkopf zum angegebenen Bild, wenn der Wert in der globalen Variablen `vTotal` beim Verlassen des Bildes größer als 1000 ist:

```
// JavaScript syntax
function exitFrame() {
    if (_global.vTotal > 1000) {
        _movie.go("Finished");
    }
}
```

Siehe auch

on enterFrame

on getBehaviorDescription

Syntax

```
-- Lingo syntax
on getBehaviorDescription
    statement(s)
end

// JavaScript syntax
function getBehaviorDescription() {
    statement(s);
}
```

Beschreibung

Diese Systemnachricht und Ereignisprozedur enthält Lingo-Code, der einen String zurückgibt, der bei Auswahl eines Verhaltens im Beschreibungsfeld des Verhaltensinspektors erscheint.

Der Beschreibungsstring ist optional.

Director sendet die `getBehaviorDescription`-Nachricht an die Verhalten, die an einem Sprite angebracht sind, wenn der Verhaltensinspektor geöffnet wird. Stellen Sie die Prozedur `on getBehaviorDescription` in ein Verhalten.

Die Prozedur kann Zeilenumbrüche zur Formatierung mehrzeiliger Beschreibungen enthalten.

Beispiel

Die folgende Anweisung zeigt „Vertikale mehrzeilige Textfeld-Bildlaufleiste“ im Beschreibungsfeld an.

```
-- Lingo syntax
on getBehaviorDescription
    return "Vertical Multiline textField Scrollbar"
end

// JavaScript syntax
function getBehaviorDescription() {
    return "Vertical Multiline textField Scrollbar";
}
```

Siehe auch

`on getPropertyDescriptionList`, `on getBehaviorTooltip`, `on runPropertyDialog`

on getBehaviorTooltip

Syntax

```
-- Lingo syntax
on getBehaviorTooltip
    statement(s)
end

// JavaScript syntax
function getBehaviorTooltip() {
    statement(s);
}
```

Beschreibung

Diese Systemnachricht und Ereignisprozedur enthält Lingo-Code, der den String zurückgibt, der in der QuickInfo eines Skripts in der Bibliothekspalette erscheint.

Director sendet die `getBehaviorTooltip`-Nachricht an das Skript, wenn sich der Cursor in der Bibliothekspalette über dem betreffenden Eintrag befindet. Stellen Sie die Prozedur `on getBehaviorTooltip` in das Verhalten.

Die Verwendung der Prozedur ist optional. Wird keine Prozedur angegeben, erscheint stattdessen der Darstellername in der QuickInfo.

Die Prozedur kann Zeilenumbrüche zur Formatierung mehrzeiliger Beschreibungen enthalten.

Beispiel

Die folgende Anweisung zeigt den String „Jigsaw puzzle piece“ im Beschreibungsfeld an.

```
-- Lingo syntax
on getBehaviorTooltip
    return "Jigsaw puzzle piece"
end

// JavaScript syntax
function getBehaviorTooltip() {
    return "Jigsaw puzzle piece";
}
```

Siehe auch

on getPropertyDescriptionList, on getBehaviorDescription, on runPropertyDialog

on getPropertyDescriptionList

Syntax

```
-- Lingo syntax
on getPropertyDescriptionList
    statement(s)
end

// JavaScript syntax
function getPropertyDescriptionList() {
    statement(s);
}
```

Beschreibung

Diese Systemnachricht und Ereignisprozedur enthält Lingo-Code, der eine Liste mit Definitionen und Beschriftungen für die Parameter erstellt, die im Dialogfeld „Parameter“ eines Verhaltens aufgeführt sind.

Stellen Sie die Prozedur `on getPropertyDescriptionList` in ein Verhaltensskript. Verhalten, die keine `on getPropertyDescriptionList`-Prozedur enthalten, werden nicht im Dialogfeld Parameter angezeigt und können auch nicht über die Benutzeroberfläche von Director bearbeitet werden.

Die `on getPropertyDescriptionList`-Nachricht wird gesendet, wenn der Verhaltensinspektor geöffnet wurde, weil der Benutzer ein Verhalten in das Drehbuch gezogen oder im Verhaltensinspektor auf ein Verhalten doppelgeklickt hat.

Die Einstellungen `#default`, `#format` und `#comment` sind für jeden Parameter obligatorisch. Folgende Werte für diese Einstellungen sind möglich:

`#default`	Die Anfangseinstellung des Parameters.

`#format`	`#integer #float #string #symbol #member #bitmap #filmloop #field #palette #picture #sound #button #shape #movie #digitalvideo #script #richtext #ole #transition #xtra #frame #marker #ink #boolean`
`#comment`	Ein Beschreibungsstring, der links neben dem Bearbeitungsfeld für den Parameter im Dialogfeld „Parameter“ erscheint.
#range	Ein Bereich mit möglichen Werten, die einer Eigenschaft zugeordnet werden können. Dieser Bereich wird als lineare Liste mit mehreren Werten oder als Mindest- und Höchstwert in Form einer Eigenschaftsliste angegeben: `[#min: minValue, #max: maxValue]`.

Beispiel

Die folgende Prozedur definiert die Parameter eines Verhaltens, die im Dialogfeld Parameter aufgeführt sind. Jede Anweisung, die mit `addProp` beginnt, fügt der Liste namens „description“ einen Parameter hinzu. Jedes der Liste hinzugefügte Element definiert eine Eigenschaft sowie die `#default`-, `#format`- und `#comment`-Werte der Eigenschaft:

```
on getPropertyDescriptionList
    description = [:]
    description.addProp(#dynamic, [#default:1, #format:#boolean, #comment:"Dynamic"])
    description.addProp(#fieldNum, [#default:1, #format:#integer,#comment:"Scroll which
sprite:"])
    description.addProp(#extentSprite, [#default:1,#format:#integer,#comment: "Extend Sprite:"])
    description.addProp(#proportional, [#default:1,#format:#boolean, #comment: "Proportional:"])
    return description
end
```

Siehe auch

`addProp`, `on getBehaviorDescription`, `on runPropertyDialog`

on hyperlinkClicked

Syntax

```
-- Lingo syntax
on hyperlinkClicked me, data, range
    statement(s)
end

// JavaScript syntax
function hyperlinkClicked(data, range) {
    statement(s);
}
```

Beschreibung

Diese Systemnachricht und Ereignisprozedur stellt fest, wann tatsächlich auf einen Hyperlink geklickt wird.

Diese Ereignisprozedur hat folgende Parameter:

- `me` – Identifiziert die Sprite-Instanz in einem Verhalten.
- `data` – Die konkreten Hyperlinkdaten werden bei der Bearbeitung des Textdarstellers im Textinspektor eingegeben.

- `range` – Der Zeichenbereich des Hyperlinks im Text. (Durch Verwendung der Syntax member Ref.char[range[1]..range[2]] kann der eigentliche Text des Zeichenbereichs angezeigt werden.)

Die folgende Prozedur sollte an einem Sprite als Verhaltensskript angebracht und nicht in ein Darstellerskript gestellt werden.

Beispiel

Das folgende Verhalten zeigt einen Link, der den angeklickten Hyperlink untersucht, gegebenenfalls zu einer URL springt und dann den eigentlichen Text des Links im Nachrichtenfenster anzeigt:

```
property spriteNum
on hyperlinkClicked(me, data, range)
    if data starts "http://" then
        gotoNetPage(data)
    end if
    currentMember = sprite(spriteNum).member
    anchorString = currentMember.char[range[1]..range[2]]
    put("The hyperlink on"&&anchorString&&"was just clicked.")
end
// JavaScript syntax
function hyperlinkClicked(data, range) {
    var st = data.slice(0,7);
    var ht = "http://";
    if (st = ht) {
        gotoNetPage(data);
    }
    var currentMember = sprite(this.spriteNum).member;
    var r1 = currentMember.getPropRef("char", range[1]).hyperlinkRange;
    var a = r1[1] - 1;
    var b = r1[2];
    var st = new String(currentMember.text);
    var anchorString = st.slice(a, b);
    put("The hyperlink on " + anchorString + " was just clicked.");
}
```

on idle

Syntax

```
-- Lingo syntax
on idle
    statement(s)
end

// JavaScript syntax
function idle() {
    statement(s);
}
```

Beschreibung

Diese Systemnachricht und Ereignisprozedur enthält Anweisungen, die immer dann ausgeführt werden, wenn der Film keine anderen Ereignisse zu verarbeiten hat. Sie eignet sich insbesondere als Position für Lingo-Anweisungen, die so oft wie möglich ausgeführt werden sollen, wie z. B. Anweisungen, die Werte in globalen Variablen aktualisieren und die aktuellen Filmbedingungenanzeigen.

Da Anweisungen in `on idle`-Prozeduren sehr häufig ablaufen, sollten Sie möglichst keinen Lingo-Code mit langen Verarbeitungszeiten in **on idle**-Prozeduren stellen.

Oftmals ist es besser, `on idle`-Prozeduren in Bildskripten anstatt in Filmskripten zu stellen, damit die `on idle`-Prozedur nur in geeigneten Situationen genutzt wird.

Director kann Darsteller aus einer internen oder externen Besetzung während eines `idle`-Ereignisses laden. Verknüpfte Darsteller können während eines `idle`-Ereignisses jedoch nicht geladen werden.

Die `idle`-Nachricht wird nur an Bilds- und Filmskripten gesendet.

Beispiel

Die folgende Prozedur aktualisiert die im Film angezeigte Zeit immer dann, wenn keine anderen Ereignisse verarbeitet werden müssen:

```
-- Lingo syntax
on idle
    member("Time").text = _system.time()
end idle

// JavaScript syntax
function idle() {
    member("Time").text = _system.time();
}
```

Siehe auch

idleHandlerPeriod

on isOKToAttach

Syntax

```
-- Lingo syntax
on isOKToAttach me, aSpriteType, aSpriteNum
    statement(s)
end

// JavaScript syntax
function isOKToAttach(aSpriteType, aSpriteNum) {
    statement(s)
}
```

Beschreibung

Diese integrierte Prozedur kann zu einem Verhalten hinzugefügt werden, um den Typ des Sprites zu überprüfen, an dem das Verhalten angebracht werden soll. Auf diese Weise lässt sich verhindern, dass das Verhalten an einem ungeeigneten Sprite-Typ angebracht wird.

Wenn Sie das Verhalten an einem Sprite anbringen, führt Director die Prozedur aus und übergibt ihr den Typ und die Nummer des betreffenden Sprites. Das Argument `me` enthält einen Bezug auf das Verhalten, das am Sprite angebracht werden soll.

Diese Prozedur wird vor der Prozedur `on getPropertyDescriptionList` ausgeführt.

Mit Lingo kann geprüft werden, ob es sich um einen der folgenden zwei Sprite-Typen handelt: `#graphic` umfasst alle Grafikdarsteller wie Formen, Bitmaps, Digitalvideo, Text usw. `#script` gibt an, dass das Verhalten am Skriptkanal angebracht wurde. In diesem Fall ist der Wert von*spriteNum* 1.

Bei beiden Sprite-Typen muss die Prozedur `TRUE` oder `FALSE` zurückgeben. Der Wert `TRUE` gibt an, dass das Verhalten am jeweiligen Sprite angebracht werden kann. Der Wert `FALSE` gibt an, dass das Verhalten nicht am Sprite angebracht werden kann.

Wenn das Verhalten keine `on isOKToAttach`-Prozedur enthält, kann es an jedem beliebigen Sprite oder Bild angebracht werden.

Diese Prozedur wird beim erstmaligen Anbringen des Verhaltens am Sprite- oder Skriptkanal aufgerufen und ebenfalls beim Anbringen eines neuen Verhaltens an einem Sprite mithilfe des Verhaltensinspektors.

Beispiel

Die folgende Anweisung prüft den Typ des Sprites, an dem das Verhalten angebracht werden soll, und gibt bei allen Grafik-Sprites mit Ausnahme von Formen `TRUE` und beim Skriptkanal `FALSE` zurück:

```
-- Lingo syntax
on isOKToAttach me, aSpriteType, aSpriteNum
    case aSpriteType of
        #graphic: -- any graphic sprite type
            return sprite(aSpriteNum).member.type <> #shape
            -- works for everything but shape cast members
        #script: --the frame script channel
            return FALSE -- doesn't work as a frame script
    end case
end

// JavaScript syntax
function isOKToAttach(aSpriteType, aSpriteNum) {
    switch (aSpriteType) {
        case symbol("graphic"): // any graphic sprite type
            return sprite(aSpriteNum).member.type != symbol("shape");
            // works for everything but shape cast members
        case symbol("script"): // the frame script channel
            return false; // doesn't work as a frame script
    }
}
```

on keyDown

Syntax

```
-- Lingo syntax
on keyDown
    statement(s)
end

// JavaScript syntax
function keyDown() {
    statement(s);
}
```

Beschreibung

Diese Systemnachricht und Ereignisprozedur enthält Anweisungen, die ausgeführt werden, wenn der Benutzer eine Taste drückt.

Sobald eine Taste gedrückt wird, durchsucht Director für einen `on keyDown`-Handler folgende Speicherorte in der angegebenen Reihenfolge: primärer Ereignis-Handler, bearbeitbares Feld „Sprite-Script“, Feld „Darsteller-Script“, Feld „Bild“ und Filmskript. Bei Sprites und Darstellern können `on keyDown`-Prozeduren nur für bearbeitbare Text- und Felddarsteller verwendet werden. Auf einen anderen Darstellertyp, wie z. B. eine Bitmap, hat ein `keyDown`-Ereignis keinen Einfluss. (Falls im gesamten Film stets dieselbe Reaktion auf die Betätigung einer Taste erfolgen soll, verwenden Sie `keyDownScript`.)

Director hält die Suche an der ersten Position an, die eine `on keyDown`-Prozedur enthält, es sei denn, die Prozedur schließt den Befehl `pass` ein, mit dem die `keyDown`-Nachricht ausdrücklich an die nächste Position weitergegeben wird.

Die Ereignisprozedur `on keyDown` eignet sich insbesondere für Lingo-Code, mit dem Tastaturkurzbefehle oder andere Benutzeroberflächenfunktionen implementiert werden, die bei Betätigung einer Taste durch den Benutzer ausgeführt werden sollen.

Wenn der Film als Applet abgespielt wird, fängt die Prozedur `on keyDown` stets Tastenanschläge ab, selbst wenn die Prozedur leer ist. Falls der Benutzer in einem bearbeitbaren Feld eine Taste betätigt, muss eine an diesem Feld angebrachte `on keyDown`-Prozedur den Befehl `pass` enthalten, damit das entsprechende Zeichen im Feld erscheint.

Die Position, an der Sie eine `on keyDown`-Prozedur einfügen, kann sich darauf auswirken, wann sie ausgeführt wird.

- Wenn die Prozedur auf ein bestimmtes bearbeitbares Feld-Sprite angewendet werden soll, stellen Sie die Prozedur in ein Sprite-Skript.
- Wenn die Prozedur generell auf einen bearbeitbaren Felddarsteller angewendet werden soll, stellen Sie die Prozedur in ein Darstellerskript.
- Wenn die Prozedur auf ein ganzes Bild angewendet werden soll, stellen Sie die Prozedur in ein Bildskript.
- Wenn die Prozedur während des gesamten Films angewendet werden soll, stellen Sie die Prozedur in ein Filmskript.

Sie können eine `on keyDown`-Prozedur vorübergehend außer Kraft setzen, indem Sie eine andere `on keyDown`-Prozedur an eine Position stellen, die von Lingo überprüft wird, bevor die Prozedur erreicht wird, die außer Kraft gesetzt werden soll. Sie können zum Beispiel eine `on keyDown`-Prozedur, die einem Darsteller zugeordnet ist, außer Kraft setzen, indem Sie eine andere `on keyDown`-Prozedur in ein Sprite-Skript stellen.

Beispiel

Die folgende Prozedur prüft, ob die Eingabetaste gedrückt wurde, und schickt in diesem Fall den Abspielkopf zu einem anderen Bild:

```
-- Lingo syntax
on keyDown
    if (_key.key = RETURN) then _movie.go("AddSum")
end keyDown

// JavaScript syntax
function keyDown() {
    if (_key.keyCode == 36) {
        _movie.go("AddSum");
    }
}
```

Siehe auch

charToNum(), keyDownScript, keyUpScript, key, keyCode, keyPressed()

on keyUp

Syntax

```
-- Lingo syntax
on keyUp
    statement(s)
end

// JavaScript syntax
function keyUp() {
    statement(s);
}
```

Beschreibung

Diese Systemmeldung und Ereignisprozedur enthält Anweisungen, die ausgeführt werden, wenn eine Taste losgelassen wird. Die Prozedur `on keyUp` ähnelt der Prozedur `on keyDown`, mit dem Unterschied, dass dieses Ereignis nach dem Erscheinen eines Zeichens eintritt, wenn ein Feld- oder Text-Sprite auf dem Bildschirm bearbeitbar ist.

Sobald eine Taste losgelassen wird, durchsucht Director für einen `on keyUp`-Handler folgende Speicherorte in der angegebenen Reihenfolge: primärer Ereignis-Handler, bearbeitbares Feld „Sprite-Script“, Feld „Darsteller-Script“, Feld „Bild“ und Filmskript. Bei Sprites und Darstellern können die `on keyUp`-Prozeduren nur für bearbeitbare Strings verwendet werden. Auf andere Darstellertypen, wie zum Beispiel eine Bitmap, hat das Ereignis `keyUp` keinen Einfluss. Wenn im gesamten Film stets dieselbe Reaktion auf das Loslassen einer Taste erfolgen soll, verwenden Sie `keyUpScript`.

Lingo hält die Suche an der ersten Position an, die eine `on keyUp`-Prozedur enthält, es sei denn, die Prozedur schließt den Befehl `pass` ein, mit dem die `keyUp`-Nachricht ausdrücklich an die nächste Position weitergegeben wird.

Die Ereignisprozedur `on keyUp` eignet sich insbesondere für Lingo-Code, mit dem Tastaturkurzbefehle oder andere Benutzeroberflächenfunktionen implementiert werden, die ausgeführt werden sollen, wenn der Benutzer Tasten loslässt.

Wenn der Film als Applet abgespielt wird, fängt die Prozedur `on keyUp` stets Tastenanschläge ab, selbst wenn die Prozedur leer ist. Falls der Benutzer in einem bearbeitbaren Feld eine Taste drückt, muss eine an diesem Feld angebrachte `on keyUp`-Prozedur den Befehl `pass` enthalten, damit die Taste im Feld erscheint.

Die Position, an der Sie eine `on keyUp`-Prozedur einfügen, kann sich darauf auswirken, wann sie ausgeführt wird.

- Wenn die Prozedur auf ein bestimmtes bearbeitbares Feld-Sprite angewendet werden soll, stellen Sie die Prozedur in ein Verhalten.
- Wenn die Prozedur generell auf einen bearbeitbaren Felddarsteller angewendet werden soll, stellen Sie die Prozedur in ein Darstellerskript.
- Wenn die Prozedur auf ein ganzes Bild angewendet werden soll, stellen Sie die Prozedur in ein Bildskript.
- Wenn die Prozedur im gesamten Film angewendet werden soll, stellen Sie die Prozedur in ein Filmskript.

Sie können ein `on keyUp`-Ereignis außer Kraft setzen, indem Sie eine andere `on keyUp`-Prozedur an eine Position stellen, die von Lingo überprüft wird, bevor die Prozedur erreicht wird, die außer Kraft gesetzt werden soll. So lässt sich zum Beispiel eine `on keyUp`-Prozedur, die einem Darsteller zugeordnet ist, außer Kraft setzen, indem Sie eine andere `on keyUp`-Prozedur in ein Sprite-Skript stellen.

Beispiel

Die folgende Prozedur prüft, ob die Eingabetaste losgelassen wurde, und schickt in diesem Fall den Abspielkopf zu einem anderen Bild:

```
-- Lingo syntax
on keyUp
    if (_key.key = RETURN) then _movie.go("AddSum")
end keyUp

// JavaScript syntax
function keyUp() {
    if (_key.keyCode == 36) {
        _movie.go("AddSum");
    }
}
```

Siehe auch

on keyDown, keyDownScript, keyUpScript

on mouseDown (Ereignisprozedur)

Syntax

```
-- Lingo syntax
on mouseDown
    statement(s)
end

// JavaScript syntax
function mouseDown() {
    statement(s);
}
```

Beschreibung

Diese Systemnachricht und Ereignisprozedur enthält Anweisungen, die ausgeführt werden, wenn die Maustaste gedrückt wird.

Sobald die Maustaste gedrückt wird, durchsucht Director für einen`onmouseDown`-Handler folgende Speicherorte in der angegebenen Reihenfolge: primärer Ereignis-Handler, bearbeitbares Feld „Sprite-Script“, Feld „Darsteller-Script“, Feld „Bild“ und Filmskript. Lingo hält die Suche an der ersten Position mit einer `on mouseDown`-Prozedur an, es sei denn, die Prozedur enthält den Befehl `pass`, mit dem die `mouseDown`-Nachricht ausdrücklich an die nächste Position weitergegeben wird.

Wenn im gesamten Film die gleiche Reaktion auf das Drücken der Maustaste erfolgen soll, verwenden Sie `mouseDownScript` oder stellen einen „mouseDown“-Handler in ein Filmskript.

Die Ereignisprozedur `on mouseDown` eignet sich besonders für Lingo-Code, mit dem Grafiken eingeblendet, Soundeffekte ausgelöst oder Sprites bewegt werden, wenn der Benutzer die Maustaste drückt.

Die Position, an der Sie eine `on mouseDown`-Prozedur einfügen, kann sich darauf auswirken, wann sie ausgeführt wird.

- Wenn die Prozedur auf ein bestimmtes Sprite angewendet werden soll, stellen Sie die Prozedur in ein Sprite-Skript.
- Wenn die Prozedur generell auf einen Darsteller angewendet werden soll, stellen Sie die Prozedur in ein Darstellerskript.
- Wenn die Prozedur auf ein ganzes Bild angewendet werden soll, stellen Sie die Prozedur in ein Bildskript.
- Wenn die Prozedur im gesamten Film angewendet werden soll, stellen Sie die Prozedur in ein Filmskript.

Sie können eine `on mouseDown`-Prozedur außer Kraft setzen, indem Sie eine andere `on mouseDown`-Prozedur an einer Stelle einfügen, die von Lingo überprüft wird, bevor es die Prozedur erreicht, die außer Kraft gesetzt werden soll. Sie können die einem Darsteller zugeordnete `on mouseDown`-Prozedur zum Beispiel außer Kraft setzen, indem Sie eine `on mouseDown`-Prozedur in ein Sprite-Skript stellen.

Wenn diese Prozedur in einem Verhalten verwendet wird, wird der Sprite- oder Bildskriptbezug`me` an dieses Ereignis übergeben.

Beispiel

Die folgende Prozedur prüft, ob der Benutzer auf die Bühne klickt, und schickt, falls ein Klick erfolgt, den Abspielkopf zu einem anderen Bild:

```
-- Lingo syntax
on mouseDown
    if (_mouse.clickOn = 0) then _movie.go("AddSum")
end

// JavaScript syntax
function mouseDown() {
    if (_mouse.clickOn == 0) {
        _movie.go("AddSum");
    }
}
```

Wenn die folgende Prozedur einem Sprite-Skript zugeordnet ist, spielt sie einen Sound ab, wenn der Benutzer auf das Sprite klickt:

```
-- Lingo syntax
on mouseDown
    sound(1).play(member("Crickets"))
end

// JavaScript syntax
function mouseDown() {
    sound(1).play(member("Crickets"));
}
```

Siehe auch

clickOn, mouseDownScript, mouseUpScript

on mouseEnter

Syntax

```
-- Lingo syntax
on mouseEnter
    statement(s)
end

// JavaScript syntax
function mouseEnter() {
    statement(s);
}
```

Beschreibung

Diese Systemnachricht und Ereignisprozedur enthält Anweisungen, die ausgeführt werden, wenn der Mauszeiger erstmals mit dem aktiven Bereich des Sprites in Kontakt kommt. Die Maustaste muss dazu nicht gedrückt sein.

Falls das Sprite ein Bitmapdarsteller ist, auf den der Farbeffekt „Matt“ angewendet wird, ist der aktive Bereich der angezeigte Teil der Grafik. Andernfalls ist der aktive Bereich das Begrenzungsrechteck des Sprites.

Wenn diese Prozedur in einem Verhalten verwendet wird, wird der Sprite- oder Bildskriptbezug `me` an dieses Ereignis übergeben.

Beispiel

Die folgende Anweisung ist ein einfaches Schaltflächenverhalten, das die Bitmap der Schaltfläche wechselt, wenn sich die Maus über die Schaltfläche und dann von ihr herunter bewegt:

```
-- Lingo syntax
property spriteNum

on mouseEnter me
    -- Determine current cast member and switch to next in cast
    currentMember = sprite(spriteNum).member.number
    sprite(spriteNum).member = currentMember + 1
end

on mouseLeave me
    -- Determine current cast member and switch to previous in cast
    currentMember = sprite(spriteNum).member.number
    sprite(spriteNum).member = currentMember - 1
end

// JavaScript syntax
var spriteNum;

function mouseEnter() {
    // Determine current cast member and switch to next in cast
    currentMember = sprite(spriteNum).member.number;
    sprite(spriteNum).member = currentMember + 1;
}

function mouseLeave() {
    // Determine current cast member and switch to previous in cast
    currentMember = sprite(spriteNum).member.number;
    sprite(spriteNum).member = currentMember - 1;
}
```

Siehe auch

on mouseLeave, on mouseWithin

on mouseLeave

Syntax

```
-- Lingo syntax
on mouseLeave
    statement(s)
end

// JavaScript syntax
function mouseLeave() {
    statement(s);
}
```

Beschreibung

Diese Systemnachricht und Ereignisprozedur enthält Anweisungen, die ausgeführt werden, wenn die Maus den aktiven Bereich des Sprites verlässt. Die Maustaste muss dazu nicht gedrückt sein.

Falls das Sprite ein Bitmapdarsteller ist, auf den der Farbeffekt „Matt“ angewendet wird, ist der aktive Bereich der angezeigte Teil der Grafik. Andernfalls ist der aktive Bereich das Begrenzungsrechteck des Sprites.

Wenn diese Prozedur in einem Verhalten verwendet wird, wird der Sprite- oder Bildskriptbezug`me` an dieses Ereignis übergeben.

Beispiel

Die folgende Anweisung zeigt ein einfaches Schaltflächenverhalten, mit dem die Bitmap der Schaltfläche ausgewechselt wird, wenn sich der Mauszeiger über die Schaltfläche und dann von ihr herunter bewegt:

```
-- Lingo syntax
property spriteNum

on mouseEnter me
    -- Determine current cast member and switch to next in cast
    currentMember = sprite(spriteNum).member.number
    sprite(spriteNum).member = currentMember + 1
end

on mouseLeave me
    -- Determine current cast member and switch to previous in cast
    currentMember = sprite(spriteNum).member.number
    sprite(spriteNum).member = currentMember - 1
end

// JavaScript syntax
var spriteNum;

function mouseEnter() {
    // Determine current cast member and switch to next in cast
    currentMember = sprite(spriteNum).member.number;
    sprite(spriteNum).member = currentMember + 1;
}

function mouseLeave() {
    // Determine current cast member and switch to previous in cast
    currentMember = sprite(spriteNum).member.number;
    sprite(spriteNum).member = currentMember - 1;
}
```

Siehe auch

on mouseEnter, on mouseWithin

on mouseUp (Ereignisprozedur)

Syntax

```
-- Lingo syntax
on mouseUp
    statement(s)
end

// JavaScript syntax
function mouseUp() {
    statement(s);
}
```

Beschreibung

Diese Systemnachricht und Ereignisprozedur enthält Anweisungen, die beim Loslassen der Maustaste aktiviert werden.

Sobald die Maustaste losgelassen wird, durchsucht Lingo für einen `on mouseUp`-Handler folgende Speicherorte in der angegebenen Reihenfolge: primärer Ereignis-Handler, bearbeitbares Feld „Sprite-Script“, Feld „Darsteller-Script“, Feld „Bild“ und Filmskript Lingo hält die Suche an der ersten Position mit einer `on mouseUp`-Prozedur an, es sei denn, die Prozedur enthält den Befehl `pass`, mit dem die Meldung `mouseUp` ausdrücklich an die nächste Position weitergegeben wird.

Um im gesamten Film die gleiche Reaktion zu erhalten, wenn der Benutzer die Maustaste loslässt, verwenden Sie `the mouseUpScript`.

Die Ereignisprozedur `on mouseUp` eignet sich für Lingo-Code, mit dem das Erscheinungsbild von Objekten (wie zum Beispiel Schaltflächen) geändert wird, nachdem sie angeklickt wurden. Tauschen Sie dazu den dem Sprite zugeordneten Darsteller aus, nachdem auf das Sprite geklickt und die Maustaste losgelassen wurde.

Die Position, an der Sie eine `on mouseUp`-Prozedur einfügen, kann sich folgendermaßen darauf auswirken, wann sie ausgeführt wird:

- Wenn die Prozedur auf ein bestimmtes Sprite angewendet werden soll, stellen Sie die Prozedur in ein Sprite-Skript.
- Wenn die Prozedur generell auf einen Darsteller angewendet werden soll, stellen Sie die Prozedur in ein Darstellerskript.
- Wenn die Prozedur auf ein ganzes Bild angewendet werden soll, stellen Sie die Prozedur in ein Bildskript.
- Wenn die Prozedur im gesamten Film angewendet werden soll, stellen Sie die Prozedur in ein Filmskript.

Sie können eine `on mouseUp`-Prozedur außer Kraft setzen, indem Sie eine andere `on mouseUp`-Prozedur an einer Stelle einfügen, die von Lingo überprüft wird, bevor es die Prozedur erreicht, die außer Kraft gesetzt werden soll. So können Sie zum Beispiel eine einem Darsteller zugeordnete `on mouseUp`-Prozedur außer Kraft setzen, indem Sie eine `on mouseUp`-Prozedur in ein Sprite-Skript stellen.

Wenn diese Prozedur in einem Verhalten verwendet wird, wird der Sprite- oder Bildskriptbezug`me` an dieses Ereignis übergeben.

Beispiel

Die folgende Prozedur, die Sprite 10 zugeordnet ist, tauscht den Sprite 10 zugeordneten Darsteller aus, wenn der Benutzer nach dem Klicken auf das Sprite die Maustaste loslässt:

```
-- Lingo syntax
on mouseUp
    sprite(10).member = member("Dimmed")
end

// JavaScript syntax
function mouseUp() {
    sprite(10).member = member("Dimmed");
}
```

Siehe auch

`on mouseDown (Ereignisprozedur)`

on mouseUpOutside

Syntax

```
-- Lingo syntax
on mouseUpOutside me
    statement(s)
end

// JavaScript syntax
function mouseUpOutside() {
    statement(s);
}
```

Beschreibung

Diese Systemnachricht und Ereignisprozedur wird abgeschickt, wenn die Maustaste über einem Sprite gedrückt wird, aber erst dann losgelassen wird, wenn sich der Zeiger nicht mehr über dem Sprite befindet.

Beispiel

Die folgende Anweisung spielt einen Sound ab, wenn die Maustaste über einem Sprite geklickt und außerhalb des Begrenzungsrechtecks des Sprites losgelassen wird:

```
-- Lingo syntax
on mouseUpOutside me
    sound(1).play(member("Professor Long Hair"))
end

// JavaScript syntax
function mouseUpOutside() {
    sound(1).play(member("Professor Long Hair"));
}
```

Siehe auch

on mouseEnter, on mouseLeave, on mouseWithin

on mouseWithin

Syntax

```
-- Lingo syntax
on mouseWithin
    statement(s)
end

// JavaScript syntax
function mouseWithin() {
    statement(s);
}
```

Beschreibung

Diese Systemnachricht und Ereignisprozedur enthält Anweisungen, die ausgeführt werden, wenn sich der Mauszeiger im aktiven Bereich des Sprites befindet. Die Maustaste muss dazu nicht gedrückt sein.

Falls das Sprite ein Bitmapdarsteller ist, auf den der Farbeffekt „Matt“ angewendet wird, ist der aktive Bereich der angezeigte Teil der Grafik. Andernfalls ist der aktive Bereich das Begrenzungsrechteck des Sprites.

Wenn diese Prozedur in einem Verhalten verwendet wird, wird der Sprite- oder Bildskriptbezug`me` an dieses Ereignis übergeben.

Beispiel

Die folgende Anweisung zeigt die Mausposition an, wenn sich der Mauszeiger über einem Sprite befindet:

```
-- Lingo syntax
on mouseWithin
    member("Display").text = string(_mouse.mouseH)
end

// JavaScript syntax
function mouseWithin() {
    member("Display").text = _mouse.mouseH.toString();
}
```

Siehe auch

`on mouseEnter`, `on mouseLeave`

on moveWindow

Syntax

```
-- Lingo syntax
on moveWindow
    statement(s)
end

// JavaScript syntax
function moveWindow() {
    statement(s);
}
```

Beschreibung

Diese Systemmeldung und Ereignisprozedur enthält Anweisungen, die beim Verschieben eines Fensters ausgeführt werden, z. B. wenn ein Film an eine neue Position auf der Bühne gezogen wird. Platzieren Sie hier Lingo-Code, der immer dann ausgeführt werden soll, wenn das Fenster eines Films seine Position ändert.

Beispiel

Die folgende Prozedur zeigt eine Meldung im Nachrichtenfenster an, wenn das Fenster, in dem ein Film abgespielt wird, verschoben wird:

```
-- Lingo syntax
on moveWindow
    put("Just moved window containing" && _movie.name)
end

// JavaScript syntax
function moveWindow() {
    put("Just moved window containing " + _movie.name);
}
```

Siehe auch

activeWindow, name (3D), windowList

on openWindow

Syntax

```
-- Lingo syntax
on openWindow
    statement(s)
end

// JavaScript syntax
function openWindow() {
    statement(s);
}
```

Beschreibung

Diese Systemnachricht und Ereignisprozedur enthält Anweisungen, die ausgeführt werden, wenn Director den Film als Film in einem Fenster öffnet. Sie eignet sich besonders für Lingo-Anweisungen, die bei jedem Öffnen des Films in einem Fenster ausgeführt werden sollen.

Beispiel

Die folgende Prozedur spielt beim Öffnen des Filmfensters die Sounddatei „Hurra“ ab:

```
-- Lingo syntax
on openWindow
    sound(2).play(member("Hurray"))
end

// JavaScript syntax
function openWindow() {
    sound(2).play(member("Hurray"));
}
```

on prepareFrame

Syntax

```
-- Lingo syntax
on prepareFrame
    statement(s)
end

// JavaScript syntax
function prepareFrame {
    statement(s);
}
```

Beschreibung

Diese Systemnachricht und Ereignisprozedur enthält Anweisungen, die unmittelbar vor dem Zeichnen des aktuellen Bildes ausgeführt werden.

Im Gegensatz zu den Ereignissen `beginSprite` und `endSprite` wird das Ereignis `prepareFrame` jedes Mal neu generiert, wenn der Abspielkopf in ein Bild eintritt.

Die Prozedur `on prepareFrame` eignet sich insbesondere für Änderungen der Sprite-Eigenschaften, bevor das Sprite gezeichnet wird.

Bei Verwendung in einem Verhalten empfängt die Prozedur `on prepareFrame` den Bezug `me`.

Die Befehle `go`, `play` und `updateStage` sind in der Prozedur `on prepareFrame` deaktiviert.

Beispiel

Die folgende Prozedur stellt die Eigenschaft `locH` für das Sprite ein, an dem das Verhalten angebracht ist:

```
-- Lingo syntax
on prepareFrame me
    sprite(me.spriteNum).locH= _mouse.mouseH
end

// JavaScript syntax
function prepareFrame() {
    sprite(spriteNum).locH= _mouse.mouseH;
}
```

Siehe auch

`on enterFrame`

on prepareMovie

Syntax

```
-- Lingo syntax
on prepareMovie
    statement(s)
end

// JavaScript syntax
function prepareMovie() {
    statement(s);
}
```

Beschreibung

Diese Systemnachricht und Ereignisprozedur enthält Anweisungen, die ausgeführt werden, nachdem der Film die Darsteller vorausgeladen hat, aber bevor der Film:

- Verhaltensinstanzen erstellt, die an Sprites im zuerst abgespielten Bild angebracht sind;
- das erste abgespielte Bild vorbereitet. Das schließt das Zeichnen des Bildes, das Abspielen von Sounds und die Ausführung von Übergängen und Paletteneffekten mit ein.

Neue globale Variablen, die für Sprite-Verhalten im ersten Bild verwendet werden, sollten in der Prozedur `onprepareMovie` initialisiert werden. Globale Variablen, die bereits durch den vorherigen Film eingestellt wurden, brauchen nicht erneut eingestellt zu werden.

Die Prozedur `on prepareMovie` eignet sich insbesondere für Lingo-Anweisungen, die globale Variablen erstellen, Variablen initialisieren und Sounds abspielen, während der restliche Film in den Speicher geladen wird. Auch können über diese Prozedur Computerbedingungen wie die Farbtiefe geprüft und angepasst werden.

Die Befehle `go`, `play` und `updateStage` sind in der Prozedur `on prepareMovie` deaktiviert.

Beispiel

Die folgende Prozedur erstellt bei Beginn des Films eine globale Variable:

```
-- Lingo syntax
on prepareMovie
    global currentScore
    currentScore = 0
end

// JavaScript syntax
function prepareMovie() {
    _global.currentScore = 0;
}
```

Siehe auch

`on enterFrame`, `on startMovie`

on resizeWindow

Syntax

```
-- Lingo syntax
on resizeWindow
    statement(s)
end

// JavaScript syntax
function resizeWindow() {
    statement(s);
}
```

Beschreibung

Diese Systemnachricht und Ereignisprozedur enthält Anweisungen, die ausgeführt werden, wenn ein Film in einem Fenster (MIAW) abgespielt wird und der Benutzer die Fenstergröße ändert, indem er das Größenänderungsfeld des Fensters bzw. an einer Fensterkante zieht.

Die Ereignisprozedur `on resizeWindow` ist besonders für Lingo-Anweisungen geeignet, die sich auf Fensterabmessungen beziehen, wie z. B. Lingo zum Positionieren von Sprites oder Zuschneiden von Digitalvideos.

Beispiel

Die folgende Prozedur verschiebt Sprite 3 auf die in der Variablen `centerPlace` gespeicherten Koordinaten, wenn das Fenster, in dem der Film abgespielt wird, vergrößert oder verkleinert wird:

```
-- Lingo syntax
on resizeWindow centerPlace
    sprite(3).loc = centerPlace
end

// JavaScript syntax
function resizeWindow(centerPlace) {
    sprite(3).loc = centerPlace;
}
```

Siehe auch

drawRect, sourceRect

on rightMouseDown (Ereignisprozedur)

Syntax

```
-- Lingo syntax
on rightMousedown
    statement(s)
end

// JavaScript syntax
function rightMouseDown() {
    statement(s);
}
```

Beschreibung

Diese Systemnachricht und Ereignisprozedur gibt in Windows die Anweisungen an, die beim Drücken der rechten Maustaste ausgeführt werden. Auf Macintosh-Computern werden diese Anweisungen ausgeführt, sobald Maus- und Steuertaste gleichzeitig gedrückt werden und die Eigenschaft `emulateMultiButtonMouse` auf `TRUE` gesetzt ist. Wenn diese Eigenschaft auf `FALSE` gesetzt ist, hat die Ereignisprozedur auf dem Macintosh keine Wirkung.

Beispiel

Die folgende Prozedur öffnet das Hilfefenster, sobald der Benutzer in Windows mit der rechten Maustaste klickt:

```
-- Lingo syntax
on rightMousedown
    window("Help").open()
end

// JavaScript syntax
function rightMouseDown() {
    window("Help").open();
}
```

on rightMouseUp (Ereignisprozedur)

Syntax

```
-- Lingo syntax
on rightMouseUp
    statement(s)
end

// JavaScript syntax
function rightMouseUp() {
    statement(s);
}
```

Beschreibung

Diese Systemnachricht und Ereignisprozedur gibt in Windows Anweisungen an, die beim Loslassen der rechten Maustaste ausgeführt werden. Auf Macintosh-Computern werden diese Anweisungen ausgeführt, sobald die Maustaste bei gedrückter Steuertaste losgelassen wird und die Eigenschaft `emulateMultiButtonMouse` auf `TRUE` gesetzt ist. Wenn diese Eigenschaft auf `FALSE` gesetzt ist, hat diese Ereignisprozedur auf dem Macintosh keine Wirkung.

Beispiel

Die folgende Prozedur öffnet das Fenster „Hilfe“, sobald der Benutzer in Windows die rechte Maustaste loslässt:

```
-- Lingo syntax
on rightMouseUp
    window("Help").open()
end

// JavaScript syntax
function rightMouseUp() {
    window("Help").open();
}
```

on runPropertyDialog

Syntax

```
-- Lingo syntax
on runPropertyDialog me, currentInitializerList
    statement(s)
end

// JavaScript syntax
function runPropertyDialog(currentInitializerList) {
    statement(s);
}
```

Beschreibung

Diese Systemnachricht und Ereignisprozedur enthält Lingo-Code, der im Parameterdialogfeld bestimmte Werte für die Parameter eines Verhaltens definiert. Die Nachricht `runPropertyDialog` wird immer dann gesendet, wenn dasVerhalten an einem Sprite angebracht ist oder der Benutzer die anfänglichen Eigenschaftswerte eines Sprite-Verhaltens ändert.

Die aktuellen Einstellungen für die Anfangseigenschaften eines Verhaltens werden als Eigenschaftsliste an die Prozedur weitergegeben. Wenn die Prozedur `on runPropertyDialog` im Verhalten nicht definiert ist, führt Director einen Verhaltensanpassungsdialog aus, und zwar auf Basis der durch die Prozedur `on getPropertyDescriptionList` zurückgegebenen Eigenschaftsliste.

Beispiel

Die folgende Prozedur setzt die im Dialogfeld „Parameter“ eingestellten Verhaltenswerte außer Kraft. Neue Werte sind in der Liste `currentInitializerList` enthalten. Normalerweise kann der Benutzer im Dialogfeld „Parameter“ die Masse- und Schwerekonstanten einstellen. Die folgende Prozedur ordnet diesen Parametern jedoch konstante Werte zu, ohne dass ein Dialogfeld angezeigt wird:

```
-- Lingo syntax
property mass
property gravitationalConstant

on runPropertyDialog me, currentInitializerList
    --force mass to 10
    currentInitializerList.setaProp(#mass, 10)
    -- force gravitationalConstant to 9.8
    currentInitializerList.setaProp(#gravitationalConstant, 9.8)
    return currentInitializerList
end

// JavaScript syntax
function runPropertyDialog(currentInitializerList) {
    //force mass to 10
    currentInitializerList.setaProp("mass", 10)
    //force gravitationalConstant to 9.8
    currentInitializerList.setaProp("gravitationalConstant", 9.8)
    return(currentInitializerList)
}
```

Siehe auch

on getBehaviorDescription, on getPropertyDescriptionList

on savedLocal

Syntax

```
-- Lingo syntax
on savedLocal
    statement(s)
end

// JavaScript syntax
function savedLocal() {
    statement(s);
}
```

Beschreibung

Diese Systemnachricht und Ereignisprozedur dient möglichen Funktionsverbesserungen in zukünftigen Versionen vonShockwave Player.

Siehe auch

allowSaveLocal

on sendXML

Syntax

```
-- Lingo syntax
on sendXML "sendxmlstring", "window", "postdata"
    statement(s)
end

// JavaScript syntax
function sendXML(sendxmlstring, window, postdata) {
    statement(s);
}
```

Beschreibung

Diese Ereignisprozedur funktioniert ähnlich wie die Skriptmethode `getURL`, die in der Adobe® Flash®-Asset-Xtra-Erweiterung verfügbar ist. Die Prozedur `on sendXML` wird in Lingo aufgerufen, wenn die ActionScript-Methode *XMLobject*`.send` in einem Flash-Sprite oder Flash-XML-Objekt ausgeführt wird.

In ActionScript übergibt die Methode *XMLobject*`.send` zwei Parameter zusätzlich zu den XML-Daten im XML-Objekt. Hierbei handelt es sich um die folgenden zwei Parameter:

- *url* – URL, an die die XML-Daten gesendet werden sollen. In der Regel ist dies die URL eines Serverskripts, das auf die Verarbeitung der XML-Daten wartet.
- *window* – Browserfenster, in dem die Antwort des Servers angezeigt werden soll.

Die ActionScript-Methode *XMLobject*`.send` kann in Director entweder von einem Flash-Sprite oder einem per Lingo erstellten globalen Flash-XML-Objekt aufgerufen werden. In diesem Fall werden die Lingo-Prozedur `on sendXML` aufgerufen und die gleichen Parameter an die Prozedur übergeben.

Der folgende Lingo-Code veranschaulicht den Empfang der Parameter durch die Prozedur `on sendXML`:

```
on sendXML me, theURL, targetWindow, XMLdata
```

Diese Parameter entsprechen wie folgt den Parametern der Methode *XMLobject*.send:

- theURL – URL, an die die XML-Daten gesendet werden sollen.
- targetWindow – Browserfenster, in dem die Antwort des Servers angezeigt werden soll.
- XMLdata – XML-Daten im Flash-XML-Objekt.

  Durch Erstellung einer on sendXML-Prozedur in einem Director-Film können Sie in einem Flash-Sprite oder einem globalen Flash-Objekt generierte *XMLobject*.send-Ereignisse verarbeiten.

  Flash-Sprites können auch externe XML-Daten laden oder interne XML-Daten parsen. DasFlash Asset-Xtraverarbeitet diese Funktionen genauso wie Flash 5- oder Flash MX-Inhalte in IhremBrowser.

Beispiel

Der folgende Lingo-Befehl ruft Informationen zur Methode *XMLobject*.send aus einem Flash-Sprite ab, verweist den Browser zur URL und überträgt die XML-Daten an die URL:

```
-- Lingo syntax
on sendXML me, theURL, targetWindow, xmlData
    gotoNetPage(theURL, targetWindow)
    postNetText(theURL, xmlData)
end

// JavaScript syntax
function sendXML(theURL, targetWindow, xmlData) {
    gotoNetPage(theURL, targetWindow);
    postNetText(theURL, xmlData);
}
```

on startMovie

Syntax

```
-- Lingo syntax
on startMovie
    statement(s)
end

// JavaScript syntax
function startMovie() {
    statement(s);
}
```

Beschreibung

Diese Systemnachricht und Ereignisprozedur enthält Anweisungen, die unmittelbar vor Eintritt des Abspielkopfes in das erste Bild des Films ausgeführt werden. Das startMovie-Ereignis findet nach dem prepareFrame-Ereignis und vor dem enterFrame-Ereignis statt.

Eine on startMovie-Prozedur eignet sich für Lingo-Code, der Sprites im ersten Bild eines Films initialisiert.

Beispiel

Die folgende Prozedur macht Sprites zu Filmbeginn unsichtbar:

```
-- Lingo syntax
on startMovie
    repeat with counter = 10 to 50
        sprite(counter).visible = 0
    end repeat
end startMovie

// JavaScript syntax
function startMovie() {
    for(counter=10;counter<=50;counter++) {
        sprite(counter).visible = 0;
    }
}
```

Siehe auch

on prepareMovie

on stepFrame

Syntax

```
-- Lingo syntax
on stepFrame
    statement(s)
end

// JavaScript syntax
function stepFrame() {
    statement(s);
}
```

Beschreibung

Diese Systemnachricht und Ereignisprozedur funktioniert in Skriptinstanzen in der actorList, da diese die einzigen Objekte sind, die on stepFrame-Nachrichten erhalten. Diese Ereignisprozedur wird ausgeführt, wenn der Abspielkopf in ein Bild eintritt oder die Bühne aktualisiert wird.

Eine on stepFrame-Prozedur eignet sich für Lingo-Code, den Sie regelmäßig für einen bestimmten Satz von Objekten ausführen möchten. Ordnen Sie die Objekte der actorList zu, wenn Lingo in der on stepFrame-Prozedur ausgeführt werden soll; entfernen Sie die Objekte aus der actorList, wenn Lingo nicht ausgeführt werden soll. Während sich die Objekte in der actorList befinden, werden die on stepFrame-Prozeduren der Objekte jedes Mal ausgeführt, wenn der Abspielkopf in ein Bild eintritt oder der Befehl updateStage ausgeführt wird.

Die stepFrame-Nachricht wird vor der prepareFrame-Nachricht gesendet.

Ordnen Sie die Objekte der actorList zu, damit sie auf stepFrame-Nachrichten reagieren. Objekte müssen über eine on stepFrame-Prozedur verfügen, um diese integrierte Funktionalität mit actorList verwenden zu können.

Die Befehle go, play und updateStage sind in einer on stepFrame-Prozedur deaktiviert.

Beispiel

Wenn actorList ein Child-Objekt zugeordnet wird, aktualisiert die in diesem Parent-Skript enthaltene on stepFrame-Prozedur die Position des in der mySprite gespeicherten Sprites, und zwar immer dann, wenn der Abspielkopf in ein Bild eintritt:

```
-- Lingo syntax
property mySprite

on new me, theSprite
    mySprite = theSprite
    return me
end

on stepFrame me
    sprite(mySprite).loc = point(random(640),random(480))
end

// JavaScript syntax
// define a constructor class that contains the mySprite property
function Frame(theSprite) {
    this.mySprite = theSprite;
}

function stepFrame() {
    var myFrame = new Frame(sprite(spriteName).spriteNum);
    sprite(myFrame.mySprite).loc = point(random(640),random(480));
end
```

on stopMovie

Syntax

```
-- Lingo syntax
on stopMovie
    statement(s)
end

// JavaScript syntax
function stopMovie() {
    statement(s);
}
```

Beschreibung

Diese Systemnachricht und Ereignisprozedur enthält Anweisungen, die ausgeführt werden, wenn die Wiedergabe des Films gestoppt wird.

Eine `on stopMovie`-Prozedur eignet sich für Lingo-Anweisungen, die nach Filmende Aufräumarbeiten erledigen, wie z. B. Ressourcendateien schließen, globale Variablen und Felder löschen und Objekte entfernen.

Eine `on stopMovie`-Prozedur in einem MIAW wird nur dann aufgerufen, wenn der Film bis zum Ende abgespielt wird oder in einen anderen Film verzweigt. Sie wird nicht aufgerufen, wenn das Fenster geschlossen oder mit dem Befehl `forget window` gelöscht wird.

Beispiel

Die folgende Prozedur erstellt beim Anhaltendes Films eine globale Variable:

```
-- Lingo syntax
global gCurrentScore

on stopMovie
    gCurrentScore = 0
end

// JavaScript syntax
_global.gCurrentScore;

function stopMovie() {
    _global.gCurrentScore = 0;
}
```

Siehe auch

`on prepareMovie`

on streamStatus

Syntax

```
-- Lingo syntax
on streamStatus URL, state, bytesSoFar, bytesTotal, error
    statement(s)
end

// JavaScript syntax
function streamStatus(URL, state, bytesSoFar, bytesTotal, error) {
    statement(s);
}
```

Beschreibung

Diese Systemnachricht und Ereignisprozedur wird in regelmäßigen Abständen aufgerufen, um festzustellen, welcher Bereich eines Objekts bereits aus dem Internet heruntergeladen wurde. Die Prozedur wird nur aufgerufen, wenn `tellStreamStatus (TRUE)` aufgerufen und die Prozedur zum Filmskript hinzugefügt wurde.

Die Ereignisprozedur `on streamStatus` hat folgende Parameter:

URL	Zeigt die Internetadresse der zu ladenden Daten an.
state	Zeigt den Stream-Status der Daten an, die heruntergeladen werden. Mögliche Werte: `Connecting`, `Started`, `InProgress`, `Complete` und `Error`.
bytesSoFar	Zeigt die Anzahl von Bytes an, die bis zu diesem Zeitpunkt aus dem Netzwerk abgerufen wurden.
bytesTotal	Zeigt die Gesamtzahl von Bytes im Stream an, sofern bekannt. Der Wert kann 0 sein, wenn der HTTP-Server die Länge des Inhalts nicht im MIME-Header angibt.
error	Zeigt einen leeren String ("") an, wenn der Downloadvorgang noch nicht abgeschlossen ist, *OK*, wenn der Downloadvorgang erfolgreich abgeschlossen wurde, oder einen Fehlercode, wenn der Downloadvorgang nicht erfolgreich war.

Diese Parameter werden automatisch von Director mit Informationen zum Downloadvorgang bestückt. Die Prozedur wird von Director automatisch aufgerufen; Sie haben keine Kontrolle darüber, wann der nächste Aufruf erfolgt. Wenn Sie Informationen über einen bestimmten Vorgang benötigen, rufen Sie `getStreamStatus()` auf.

Netzwerk-Streams können mit Lingo-Befehlen eingeleitet werden, aber auch über URL-Links zu Medien oder durch Verwendung eines externen Darstellers aus einer URL. Eine `streamStatus`-Prozedur wird mit Informationen über alle Netzwerk-Streams aufgerufen.

Stellen Sie die `streamStatus`-Prozedur in ein Filmskript.

Beispiel

Die folgende Prozedur bestimmt den Status eines gestreamten Objekts und zeigt die URL des Objekts an:

```
-- Lingo syntax
on streamStatus URL, state, bytesSoFar, bytesTotal
    if state = "Complete" then
        put(URL && "download finished")
    end if
end streamStatus

// JavaScript syntax
function streamStatus(URL, state, bytesSoFar, bytesTotal) {
    if (state == "Complete") {
        put(URL + " download finished");
    }
}
```

Siehe auch

`getStreamStatus()`, `tellStreamStatus()`

on timeOut

Syntax

```
-- Lingo syntax
on timeOut
    statement(s)
end

// JavaScript syntax
function timeOut() {
    statement(s);
}
```

Beschreibung

Diese Systemnachricht und Ereignisprozedur enthält Anweisungen, die ausgeführt werden, wenn eine bestimmte Zeit lang in `timeOutLength` keine Tastatur- oder Mausaktivität stattfindet. Die Prozedur `on timeOut` ist immer in ein Filmskript zu stellen.

Damit ein Timeout immer dieselbe Wirkung in einem Film hat, sollten Sie `the timeoutScript` verwenden, um das Timeout-Verhalten zentral zu steuern.

Beispiel

Die folgende Prozedur spielt den Film „Attract-Schleife“ ab, nachdem der Benutzer während des in der Eigenschaft `timeOutLength` festgelegten Zeitraums keine Aktion durchgeführt hat. Diese Prozedur kann eine Reaktion erzeugen, wenn der Benutzer den Computer verlässt.

```
-- Lingo syntax
on timeOut
    _movie.play("Attract Loop")
end timeOut

// JavaScript syntax
function timeOut() {
    _movie.play("Attract Loop");
}
```

trayIconMouseDoubleClick

Syntax

```
-- Lingo syntax
on trayIconMouseDoubleClick
    statement(s)
end

// JavaScript syntax
function trayIconMouseDoubleClick() {
    statement(s);
}
```

Beschreibung

Diese Film- und Fenstereignisprozedur (nur Microsoft Windows) enthält Anweisungen, die ausgeführt werden, wenn der Benutzer auf das Symbol des Infobereichs doppelklickt.

Das Ereignis `trayIconMouseDoubleClick` wird nur an die Prozedur gesendet, wenn die Eigenschaft `systemTrayIcon` auf `TRUE` eingestellt ist.

Beispiel

Die folgende Prozedur unterbricht einen Film, wenn der Benutzer auf das Symbol des Infobereichs doppelklickt.

```
-- Lingo syntax
on trayIconMouseDoubleClick
    _movie.delay(500)
end

// JavaScript syntax
function trayIconMouseDoubleClick() {
    _movie.delay(500);
}
```

Siehe auch

Movie, systemTrayIcon, trayIconMouseDown, trayIconRightMouseDown, Window

trayIconMouseDown

Syntax

```
-- Lingo syntax
on trayIconMouseDown
    statement(s)
end

// JavaScript syntax
function trayIconMouseDown() {
    statement(s);
}
```

Beschreibung

Diese Film- und Fenstereignisprozedur (nur Microsoft Windows) enthält Anweisungen, die ausgeführt werden, wenn der Benutzer auf das Symbol des Infobereichs klickt.

Das Ereignis `trayIconMouseDown` wird nur an die Prozedur gesendet, wenn die Eigenschaft `systemTrayIcon` auf `TRUE` eingestellt ist.

Beispiel

Die folgende Prozedur unterbricht einen Film, wenn der Benutzer die Maustaste drückt, während sich der Mauszeiger auf dem Symbol des Infobereichs befindet.

```
-- Lingo syntax
on trayIconMouseDown
    _movie.delay(500)
end

// JavaScript syntax
function trayIconMouseDown() {
    _movie.delay(500);
}
```

Siehe auch

Movie, systemTrayIcon, trayIconMouseDoubleClick, trayIconRightMouseDown, Window

trayIconRightMouseDown

Syntax

```
-- Lingo syntax
on trayIconRightMouseDown
    statement(s)
end

// JavaScript syntax
function trayIconRightMouseDown() {
    statement(s);
}
```

Beschreibung

Diese Film- und Fensterereignisprozedur (nur Microsoft Windows) enthält Anweisungen, die ausgeführt werden, wenn der Benutzer mit der rechten Maustaste auf das Symbol des Infobereichs klickt.

Das Ereignis `trayIconRightMouseDown` wird nur an die Prozedur gesendet, wenn die Eigenschaft `systemTrayIcon` auf `TRUE` eingestellt ist.

Beispiel

Die folgende Prozedur unterbricht einen Film, wenn der Benutzer mit der rechten Maustaste auf das Symbol des Infobereichs klickt.

```
-- Lingo syntax
on trayIconRightMouseDown
    _movie.delay(500)
end

// JavaScript syntax
function trayIconRightMouseDown() {
    _movie.delay(500);
}
```

Siehe auch

Movie, systemTrayIcon, trayIconMouseDoubleClick, trayIconMouseDown, Window

on zoomWindow

Syntax

```
-- Lingo syntax
on zoomWindow
    statement(s)
end

// JavaScript syntax
function zoomWindow() {
    statement(s);
}
```

Beschreibung

Diese Systemnachricht und Ereignisprozedur enthält Anweisungen, die ausgeführt werden, wenn die Größe eines Films, der als Film in einem Fenster (MIAW) läuft, geändert wird. Dies geschieht, wenn der Benutzer entweder auf die Schaltflächen zum Minimieren oder Maximieren (Windows) oder auf die Zoom-Taste (Macintosh) klickt. Das Betriebssystem bestimmt die Dimensionen nach Änderung der Fenstergröße.

Die Ereignisprozedur `on zoomWindow` ist eine geeignete Stelle für Lingo-Code, der Sprites neu anordnet, nachdem sich die Fensterdimensionen geändert haben.

Beispiel

Die folgende Prozedur verschiebt Sprite 3 auf die in der Variablen `centerPlace` gespeicherten Koordinaten, wenn das Fenster, in dem der Film abgespielt wird, vergrößert oder verkleinert wird:

```
-- Lingo syntax
on zoomWindow
    centerPlace = point(10, 10)
    sprite(3).loc = centerPlace
end

// JavaScript syntax
function zoomWindow() {
    var centerPlace = point(10, 10);
    sprite(3).loc = centerPlace;
}
```

Siehe auch

drawRect, sourceRect, on resizeWindow

Kapitel 11: Schlüsselwörter

In diesem Abschnitt finden Sie eine alphabetische Liste aller in Adobe® Director® verfügbarer Schlüsselwörter.

Diese Schlüsselwörter gelten nur für Lingo. In der JavaScript-Syntax gibt es einige Schlüsselwörter und Konstrukte mit ähnlichen Funktionen wie die aufgeführten Lingo-Schlüsselwörter, doch diese werden hier nicht dokumentiert. Weitere Informationen zu den Schlüsselwörtern und Konstrukten für die JavaScript-Syntax finden Sie unter „Grundlagen der Skripterstellung in Director“ auf Seite 4.

\ (Fortsetzung)

Syntax

```
-- Lingo syntax
first part of a statement on this line \
second part of the statement \
third part of the statement
```

Beschreibung

Fortsetzungssymbol: Als letztes Zeichen in einer Zeile gibt es an, dass die Anweisung in der nächsten Zeile fortgesetzt wird. Lingo interpretiert die Zeilen dann als eine fortlaufende Anweisung.

Beispiel

Die folgende Anweisung verwendet das Zeichen \, um die Anweisung auf zwei Zeilen umzubrechen:

```
-- Lingo syntax
if sprite("mySprite").member = member("myMember") then \
_player.alert("The sprite was created from myMember")
```

case

Syntax

```
-- Lingo syntax
case expression of
    expression1: Statement
    expression2: Statement(s)
    expression3, expression4: Statement
    {otherwise: Statement(s)}
end case
```

Beschreibung

Dieses Schlüsselwort startet eine logische Struktur mit mehrfachen Verzweigungen, die leichter zu schreiben ist als die wiederholte Verwendung von `if...then`-Anweisungen.

Lingo vergleicht den Wert in *case expression* mit den Ausdrücken in den nachfolgenden Zeilen. Der Vergleich beginnt am Anfang und wird der Reihenfolge nach in jeder Zeile fortgesetzt, bis Lingo auf einen Ausdruck stößt, der mit *case expression* übereinstimmt.

Wenn Lingo einen passenden Ausdruck findet, führt es die entsprechende Anweisung oder Anweisungen aus, die nach dem Doppelpunkt hinter dem passenden Ausdruck stehen. Wenn nur eine Anweisung auf den passenden Ausdruck folgt, können dieser Ausdruck und seine entsprechende Anweisung auf derselben Zeile stehen. Mehrere Anweisungen müssen in eingerückten Zeilen unmittelbar unter dem passenden Ausdruck stehen.

Mehrere mögliche Übereinstimmungen könnten Lingo veranlassen, dieselben Anweisungen auszuführen. In einem solchem Fall müssen die Ausdrücke durch ein Komma voneinander getrennt werden. (Die Syntaxzeile, die *expression3* und *expresssion4* enthält, ist ein Beispiel dafür.)

Nachdem Lingo den ersten passenden Ausdruck gefunden hat, hört es mit der Suche nach weiteren übereinstimmenden Ausdrücken auf.

Wenn die optionale Anweisung `otherwise` am Ende der case-Struktur enthalten ist, werden die auf `otherwise` folgenden Anweisungen ausgeführt, falls keine passenden Ausdrücke gefunden wurden.

Beispiel

Die folgende Prozedur prüft, welche Taste der Benutzer zuletzt gedrückt hat, und reagiert entsprechend.

- Wenn der Benutzer A gedrückt hat, geht der Film zum Bild mit der Beschriftung „Apple“.
- Hat der Benutzer B oder C gedrückt, führt der Film den angegebenen Übergang aus und geht dann zum Bild mit der Beschriftung „Oranges“.
- Hat der Benutzer eine andere Taste gedrückt, gibt der Computer einen Warnton aus.

```
on keyDown
    case (_key.key) of
        "a": _movie.go("Apple")
        "b", "c":
             _movie.puppetTransition(99)
             _movie.go("Oranges")
        otherwise: _sound.beep()
    end case
end keyDown
```

Diese case-Anweisung prüft, ob sich der Cursor über Sprite 1, 2 oder 3 befindet, und führt die entsprechenden Anweisungen aus, wenn dies der Fall ist:

```
case _movie.rollOver() of
    1: sound(1).play(member("Horn"))
    2: sound(1).play(member("Drum"))
    3: sound(1).play(member("Bongos"))
end case
```

char...of

Syntax

```
-- Lingo syntax
textMemberExpression.char[whichCharacter]
char whichCharacter of fieldOrStringVariable
textMemberExpression.char[firstCharacter..lastCharacter]
char firstCharacter to lastCharacter of fieldOrStringVariable
```

Beschreibung

Dieses Schlüsselwort identifiziert ein Zeichen oder einen Zeichenbereich in einem Chunk-Ausdruck. Ein Chunk-Ausdruck ist ein Zeichen, Wort, Gegenstand oder eine Zeile in einer beliebigen Textquelle (wie z. B. Felddarstellern und Variablen), die einen String enthält.

- Ein Ausdruck, der *whichCharacter* verwendet, identifiziert ein bestimmtes Zeichen.
- Ein Ausdruck, der *firstCharacter* (erstes Zeichen) und *lastCharacter* (letztes Zeichen) verwendet, identifiziert einen Zeichenbereich.

Diese Ausdrücke müssen Ganzzahlen sein, die ein Zeichen oder einen Zeichenbereich im Chunk angeben. Zeichen können Buchstaben, Zahlen, Satzzeichen, Leerzeichen und Steuerzeichen wie Tabulator und Eingabetaste sein.

Sie können das Schlüsselwort `char...of` testen, aber nicht einstellen. Verwenden Sie den Befehl `put...into`, um Zeichen in einem String zu modifizieren.

Beispiel

Die folgende Anweisung zeigt das erste Zeichen des Strings $9.00 an:

```
//Lingo
put(("$9.00").char[1..1])
-- "$"

// Javascript
trace(("$9.00").substring(0,1))
-- "$"
```

Die folgende Anweisung zeigt den vollständigen String $9.00 an:

```
//Lingo
put(("$9.00").char[1..5])
-- "$9.00"

// Javascript
trace(("$9.00").substring(0))
-- "$9.00"
```

Die folgende Anweisung ändert die ersten fünf Zeichen des zweiten Wortes in der dritten Zeile eines Textdarstellers:

```
//Lingo
member("quiz").line[3].word[2].char[1..5] = "?????"

// Javascript
var s = member(1).getPropRef("line",3).getProp("word",2)
s=s.replace(s.substring(0),"????")
member(1).getPropRef("line",3).setProp("word",2,s)
```

Siehe auch

mouseMember, mouseItem, mouseLine, mouseWord

end

Syntax

```
-- Lingo syntax
end
```

Beschreibung
Dieses Schlüsselwort markiert das Ende von Prozeduren und mehrzeiligen Befehlsstrukturen.

Beispiel
Die folgende mouseDown-Prozedur endet mit einer „end mouseDown“-Anweisung.

```
on mouseDown
     _player.alert("The mouse was pressed")
end mouseDown
```

end case

Syntax

```
-- Lingo syntax
end case
```

Beschreibung
Dieses Schlüsselwort beendet eine `case`-Anweisung.

Beispiel
Die folgende Prozedur verwendet das Schlüsselwort end case zum Beenden der Anweisung case:

```
on keyDown
    case (_key.key) of
        "a": _movie.go("Apple")
        "b", "c":
             _movie.puppetTransition(99)
             _movie.go("Oranges")
        otherwise: _sound.beep()
    end case
end keyDown
```

Siehe auch
case

exit

Syntax

```
-- Lingo syntax
exit
```

Beschreibung
Dieses Schlüsselwort weist Lingo an, eine Prozedur zu verlassen und zu der Stelle zurückzukehren, an der die Prozedur aufgerufen wurde. Wenn die Prozedur in einer anderen Prozedur verschachtelt ist, kehrt Lingo zur Hauptprozedur zurück.

Beispiel

Die erste Anweisung des folgenden Skripts überprüft, ob der Monitor auf Schwarzweiß gesetzt ist, und wird beendet, wenn dies der Fall ist:

```
on setColors
    if _system.colorDepth = 1 then exit
    sprite(1).foreColor = 35
end
```

Siehe auch

abort, halt(), quit(), pass, return (Schlüsselwort)

exit repeat

Syntax

```
-- Lingo syntax
exit repeat
```

Beschreibung

Dieses Schlüsselwort weist Lingo an, eine Wiederholungsschleife zu verlassen und zu der Anweisung zu gehen, die auf die Anweisung `end repeat` folgt, ohne jedoch die aktuelle Prozedur oder Methode zu beenden.

Das Schlüsselwort `exit repeat` ist geeignet, um eine Wiederholungsschleife abzubrechen, wenn eine angegebene Bedingung – beispielsweise das Vorhandensein von zwei identischen Werten oder einer Variable mit einem bestimmen Wert – vorliegt.

Beispiel

Die folgende Prozedur sucht nach der Position des ersten Vokals in einem String, der durch die Variable testString dargestellt wird. Sobald der erste Vokal gefunden wurde, weist der Befehl exit repeat Lingo an, die Wiederholungsschleife zu verlassen und zur Anweisung return i zu gehen:

```
on findVowel testString
    repeat with i = 1 to testString.char[testString.char.count]
        if "aeiou" contains testString.char[i] then exit repeat
    end repeat
    return i
end
```

Siehe auch

repeat while, repeat with

field

Syntax

```
field(whichField)
```

Beschreibung

Dieses Schlüsselwort verweist auf den Felddarsteller, der in *whichField* angegeben ist.

- Wenn es sich bei *whichField* um einen String handelt, wird dieser als Darstellername verwendet.
- Wenn *whichField* eine Ganzzahl ist, wird diese als Darstellernummer verwendet.

  Strings und Chunk-Ausdrücke können aus dem Feld ausgelesen oder in das Feld geschrieben werden.

  Der Begriff `field` wurde in früheren Versionen von Director verwendet und wird aus Gründen der Abwärtskompatibilität beibehalten. In neuen Filmen sollten Sie mit `member` auf Felddarsteller verweisen.

Beispiel

Die folgende Anweisung setzt die Zeichen 5 bis 10 des Feldnameneintrags in die Variable myKeyword:

```
myKeyword = field("entry").char[5..10]
```

Die folgende Anweisung überprüft, ob der Benutzer das Wort *Schreibtisch* eingegeben hat, und geht in diesem Fall zum Bild „SchreibtischGebot“:

```
if member("bid") contains "desk" then _movie.go("deskBid")
```

Siehe auch

`char...of`, `item...of`, `line...of`, `word...of`

global

Syntax

```
global variable1 {, variable2} {, variable3}...
```

Beschreibung

Dieses Schlüsselwort definiert eine Variable als globale Variable, damit sie auch von anderen Prozeduren oder Filmen verwendet werden kann.

Jede Prozedur, die den Inhalt einer globalen Variablen untersucht oder ändert, muss die Variable mit dem Schlüsselwort `global` als globale Variable kennzeichnen. Andernfalls wird die Variable von der Prozedur als lokale Variable behandelt, selbst wenn sie in einer anderen Prozedur als globale Variable deklariert wurde.

Hinweis: *Damit globale Variablen im gesamten Film verfügbar sind, müssen sie in der Prozedur* `prepareMovie` *deklariert und initialisiert werden. Wenn Sie anschließend den Film verlassen und dann aus einem anderen Film zu diesem zurückkehren, werden die globalen Variablen auf die Anfangswerte zurückgesetzt, falls Sie nicht zuerst überprüfen, ob sie nicht bereits eingestellt sind.*

Eine globale Variable kann in jeder Prozedur und in jedem Skript deklariert werden. Ihr Wert kann von anderen Prozeduren oder Skripts verwendet werden, welche die Variable ebenfalls als global deklarieren. Wenn das Skript den Wert der Variablen ändert, steht der neue Wert allen anderen Prozeduren zur Verfügung, welche die Variable als globale Variable behandeln.

Eine globale Variable ist in jedem Skript oder Film verfügbar, unabhängig davon, wo sie zuerst deklariert wurde, und wird nicht automatisch gelöscht, wenn Sie zu einem anderen Bild, Film oder Fenster wechseln.

Alle im Nachrichtenfenster manipulierten Variablen sind automatisch global, auch wenn sie nicht ausdrücklich als solche deklariert wurden.

Im Internet abgespielte Filme mitShockwave®-Inhalt können nicht auf globale Variablen in anderen Filmen zugreifen, selbst wenn diese Filme auf derselben HTML-Seite abgespielt werden. Filme können globale Variablen nur dann gemeinsam verwenden, wenn ein eingebetteter Film zu einem anderen Film gelenkt wird und sich dann selbst mit einem der Befehle `goToNetMovie` oder `go movie` ersetzt.

Beispiel

Die folgende Anweisung stellt die globale Variable StartingPoint auf den Anfangswert 1 ein, sofern sie nicht bereits einen Wert enthält. Dies ermöglicht das Navigieren zu und aus dem Film, ohne dass gespeicherte Daten verloren gehen.

```
//Lingo
global gStartingPoint

on prepareMovie
    if voidP(gStartingPoint) then gStartingPoint = 1
end

// Javascript
function prepareMovie(){
    if (_global.gStartingPoint==null) then _global.gStartingPoint = 1
}
```

Siehe auch

showGlobals(), property, gotoNetMovie

if

Syntax

```
if logicalExpression then statement
if logicalExpression then statement
else statement
end if
if logicalExpression then
        statement(s)
end if
if logicalExpression then
        statement(s)
else
        statement(s)
end if
if logicalExpression1 then
        statement(s)
else if logicalExpression2 then
        statement(s)
else if logicalExpression3 then
        statement(s)
end if
if logicalExpression1 then
        statement(s)
else logicalExpression2
end if
```

Beschreibung

Dieses Schlüsselwort mit der Struktur `if...then` wertet den durch *logicalExpression* angegebenen logischen Ausdruck aus.

- Wenn die Bedingung `TRUE` ist, führt Lingo die auf `then` folgende(n) Anweisung(en) aus.
- Wenn die Bedingung `FALSE` ist, führt Lingo die auf `else` folgende(n) Anweisung(en) aus. Folgen keine Anweisungen auf `else`, beendet Lingo die `if...then`-Struktur.
- Alle Teile der Bedingung müssen ausgewertet werden – die Ausführung der Anweisung wird nicht an der ersten erfüllten oder unerfüllten Bedingung beendet. Ein schnellerer Code kann erstellt werden, wenn die `if...then`-Anweisungen in getrennte Zeilen und nicht alle in die erste auszuwertende Zeile gesetzt werden.

Wenn es sich bei der Bedingung um eine Eigenschaft handelt, prüft Lingo automatisch, ob die Eigenschaft `TRUE` ist. Sie brauchen deshalb nicht explizit den Ausdruck `= TRUE` nach der Eigenschaft hinzufügen.

Der `else`-Teil der Anweisung ist optional. Wenn Sie mehr als eine *then*- oder *else*-Anweisung verwenden möchten, müssen Sie die Struktur mit dem Format `end if` abschließen.

Der `else`-Teil stimmt immer mit der vorherigen `if`-Anweisung überein. Aus diesem Grund müssen Sie manchmal eine `else nothing`-Anweisung einfügen, um ein `else`-Schlüsselwort mit dem entsprechenden `if`-Schlüsselwort zu verbinden.

Hinweis: *Durch Drücken der Tabulatortaste im Skriptfenster lässt sich rasch feststellen, ob ein Skript ordnungsgemäß gepaart wurde. Hierbei wird Director angewiesen, das geöffnete Skriptfenster zu überprüfen und die Einrückung des Inhalts anzuzeigen. Jegliche Nichtübereinstimmung wird hierdurch sofort erkennbar.*

Beispiel

Die folgende Anweisung prüft, ob die Eingabetaste gedrückt wurde, und geht zum nächsten Bild, wenn dies der Fall ist:

```
if the key = RETURN then go the frame + 1
```

Die folgende Prozedur prüft, ob die Befehlstaste und die Taste "Q" gleichzeitig gedrückt wurden, und führt in diesem Fall die nachfolgenden Anweisungen aus:

```
on keyDown
    if (_key.commandDown) and (_key.key = "q") then
        cleanUp
        quit
    end if
end keyDown
```

Vergleichen Sie die folgenden beiden Strukturen und Leistungsergebnisse. Die erste Struktur wertet beide Bedingungen aus und muss deshalb das Zeitmaß bestimmen; dies kann etwas dauern. Die zweite Struktur wertet die erste Bedingung aus und überprüft die zweite Bedingung nur, wenn die erste Bedingung TRUE ist.

```
spriteUnderCursor = rollOver()
if (spriteUnderCursor > 25) and MeasureTimeSinceIStarted() then
    _player.alert("You found the hidden treasure!")
end if
```

Die alternative und schnellere Struktur lautet folgendermaßen:

```
spriteUnderCursor = rollOver()
if (spriteUnderCursor > 25) then
    if MeasureTimeSinceIStarted() then
        _player.alert("You found the hidden treasure!")
    end if
end if
```

Siehe auch

case

INF

Syntax

```
-- Lingo syntax
INF
```

Beschreibung

Dieser Rückgabewert gibt an, dass der angegebene Lingo-Ausdruck bei Auswertung eine unendliche Zahl ergibt.

Siehe auch

NAN

item...of

Syntax

```
-- Lingo syntax
textMemberExpression.item[whichItem]
item whichItem of fieldOrStringVariable
textMemberExpression.item[firstItem..lastItem]
item firstItem to lastItem of fieldOrStringVariable
```

Beschreibung

Dieses Schlüsselwort bezeichnet ein Element oder einen Bereich von Elementen in einem Chunk-Ausdruck. In diesem Fall ist ein Element eine beliebige Folge von Zeichen, die durch den aktuellen Begrenzer, der durch die Eigenschaft `itemDelimiter` bestimmt wird, begrenzt sind.

Die Begriffe *whichItem*, *firstItem* und *lastItem* müssen ganzzahlige Werte oder Ausdrücke sein, die auf die Position der Elemente im Chunk verweisen.

Chunk-Ausdrücke verweisen auf beliebige Zeichen, Wörter, Elemente oder Zeilen in einer beliebigen Stringquelle. Zu den Stringquellen gehören unter anderem Feld- und Textdarsteller sowie Variablen, die Strings enthalten.

Wenn die Zahl, die das letzte Element bezeichnet, größer als die Position des Elements im Chunk-Ausdruck ist, wird stattdessen das tatsächliche letzte Element angegeben.

Beispiel

Die folgende Anweisung sucht nach dem dritten Element im Chunk-Ausdruck, der aus Farbbezeichnungen besteht, und zeigt das Ergebnis dann im Nachrichtenfenster an:

```
put("red,yellow,blue green,orange".item[3])
-- "blue green"
```

Das Ergebnis ist der gesamte Chunk "blaugrün", da der ganze Chunk zwischen den Kommas steht.

Die folgende Anweisung sucht nach dem dritten bis fünften Element im Chunk-Ausdruck. Da der Chunk-Ausdruck nur vier Elemente enthält, werden nur das dritte und vierte Element zurückgegeben. Das Ergebnis wird im Nachrichtenfenster angezeigt.

```
put("red,yellow,blue green,orange".item[3..5])
-- "blue green, orange"
put item 5 of "red, yellow, blue green, orange"
-- ""
```

Die folgende Anweisung fügt das Element „Schreibtisch“ als viertes Element in der zweiten Zeile des Felddarstellers „Alle Gebote“ ein:

```
member("All Bids").line[2].item[4] = "Desk"
```

Siehe auch

char...of, itemDelimiter, number of members, word...of

line...of

Syntax

```
-- Lingo syntax
textMemberExpression.line[whichLine]
line whichLine of fieldOrStringVariable
textMemberExpression.line[firstLine..lastLine]
line firstLine to lastLine of fieldOrStringVariable
```

Beschreibung

Dieses Schlüsselwort gibt eine Zeile oder einen Zeilenbereich in einem Chunk-Ausdruck an. Ein Zeilen-Chunk ist eine beliebige Folge von Zeichen, die durch Zeilenschaltungen begrenzt sind und nicht durch normalen Textumbruch entstehen.

Die Ausdrücke *whichLine*, *firstLine* und *lastLine* müssen Ganzzahlen sein, die eine Zeile im Chunk angeben.

Chunk-Ausdrücke verweisen auf beliebige Zeichen, Wörter, Elemente oder Zeilen in beliebigen Zeichenquellen. Zu Zeichenquellen gehören Felddarsteller und Variablen, die Strings enthalten.

Beispiel

Die folgende Anweisung ordnet die ersten vier Zeilen der Variablen „Action“ dem Felddarsteller „To Do“ zu:

```
member("To Do").text = Action.line[1..4]
```

Die folgende Anweisung fügt das Wort „and“ nach dem zweiten Wort in der dritten Zeile des Strings ein, welcher der Variablen „Notes“ zugeordnet ist:

```
put "and" after Notes.line[3].word[2]
```

Siehe auch

char...of, item...of, word...of, number of members

loop (Schlüsselwort)

Syntax

```
-- Lingo syntax
_movie.goLoop()
```

Beschreibung

Dieses Schlüsselwort verweist auf die Markierung.

Beispiel

Die folgende Prozedur bewirkt, dass der Film zwischen der vorherigen Markierung und dem aktuellen Bild in Schleife abgespielt wird:

```
on exitFrame
    _movie.goLoop()
end exitFrame
```

me

Syntax

```
-- Lingo syntax
me
```

Beschreibung

Diese spezielle Variable wird in übergeordneten Skripts und Verhalten verwendet, um auf das aktuelle Objekt zu verweisen, das eine Instanz des übergeordneten Skripts bzw. Verhaltens oder eine Variable ist, welche die Speicheradresse des Objektsenthält.

Der Begriff selbst hat in Lingo keine vordefinierte Bedeutung. Der Begriff `me` wird wie allgemein üblich verwendet.

Ein Beispiel für `me` in einem fertigen Film finden Sie im Film „Parent Scripts“ im Ordner „Learning\\Lingo_Examples“ im Director-Anwendungsordner.

Beispiel

Die folgende Anweisung stellt das Objekt `myBird1` auf das Skript namens „Bird“ ein. Das Schlüsselwort `me` akzeptiert das Parameterskript „Vogel“ und wird zur Rückgabe dieses Parameters verwendet.

```
myBird1 = new script("Bird")
```

Dies ist die Prozedur `on new` des Skripts „Vogel“:

```
on new me
    return me
end
```

Ein Parent-Skript besteht aus den beiden folgenden Sätzen von Prozeduren. Der erste Satz verwendet `me`, um auf das untergeordnete Objekt Bezug zu nehmen. Der zweite Satz verwendet die Variable `myAddress`, um auf das untergeordnete Objekt Bezug zu nehmen. In allen übrigen Aspekten sind die übergeordneten Skriptenidentisch.

Dies ist der erste Satz:

```
property myData

on new me, theData
    myData = theData
    return me
end

on stepFrame me
    ProcessData me
end
```

Dies ist der zweite Satz:

```
property myData

on new myAddress, theData
    myData = theData
    return myAddress
end

on stepFrame myAddress
    ProcessData myAddress
end
```

Siehe auch

new(), ancestor

menu

Syntax

```
-- Lingo syntax
menu: menuName
itemName | script
itemName | script
...
[more menus]
```

Beschreibung

Dieses Schlüsselwort gibt in Verbindung mit dem Befehl `installMenu` den konkreten Inhalt benutzerdefinierter Menüs an. Felddarsteller enthalten Menüdefinitionen, und Sie können mit dem Darstellernamen oder der Darstellernummer auf dieseverweisen.

Unmittelbar auf das Schlüsselwort `menu` folgen ein Doppelpunkt, eine Leerstelle und der Name des Menüs. In den nachfolgenden Zeilen geben Sie die Menüelemente für dieses Menü an. Sie können ein Skript einstellen, das ausgeführt wird, wenn der Benutzer ein Menüelement auswählt, indem Sie das Skript nach dem vertikalen Strichsymbol (|) angeben. Ein neues Menü wird beim nachfolgenden Auftreten des Schlüsselworts `menu` definiert.

Hinweis: *Im Shockwave Player sind keine Menüs verfügbar.*

Auf dem Macintosh können Sie benutzerdefinierte Menüs mithilfe von Sonderzeichen definieren. Bei diesen Sonderzeichen wird zwischen Groß- und Kleinschreibung unterschieden. Wenn beispielsweise ein Menüeintrag fett dargestellt werden soll, muss der Buchstabe *B* als Großbuchstabe eingegeben werden.

Sonderzeichen sollten auf den Namen des Menüelements folgen und vor dem vertikalen Strichsymbol (|) stehen. Sie können zur Definition eines Menüs außerdem mehr als ein Sonderzeichen verwenden. Wenn Sie zum Beispiel <B<U verwenden, wird der Stil auf „Fett“ und „Unterstrichen“ gesetzt.

Formatieren Sie möglichst keine plattformübergreifenden Filme mit Sonderzeichen, weil diese nicht von allen Windows®-Computern unterstützt werden.

Symbol	Beispiel	Beschreibung
@	menu: @	*Erstellt auf dem Macintosh® das Apple®-Symbol und aktiviert Einträge in der Macintosh-Menüleiste, wenn Sie ein Apple-Menü definieren.
!Ã	!ÃEasy Select	*Versieht auf dem Macintosh das Menü mit einem Häkchen (Wahl+V).
<B	Bold<B	*Stellt auf dem Macintosh den Schriftstil des Menüeintrags auf „Fett“ ein.
<I	Italic<I	*Stellt auf dem Macintosh den Schriftstil auf „Kursiv“ein.
<U	Underline<U	*Stellt auf dem Macintosh den Schriftstil auf „Unterstrichen“ ein.
<O	Outline<O	*Stellt auf dem Macintosh den Schriftstil auf „Kontur“ ein.
<S	Shadow<S	*Stellt auf dem Macintosh den Schriftstil auf „Schatten“ ein.
|	Open/O | go to frame "Open"	Weist dem Menüeintrag ein Skript zu.
/	Quit/Q	Definiert ein Befehlstastenäquivalent.
(	Save(	Deaktiviert den Menüeintrag.
(-	(-	Erstellt eine deaktivierte Zeile im Menü.

*Kennzeichnet die Formatierungs-Tags, die nur auf dem Macintosh funktionieren.

Beispiel

Dieses Beispiel ist der Text eine Felddarstellers namens „CustomMenu2“, mit dem der Inhalt eines benutzerdefinierten Dateimenüs angegeben werden kann. Verwenden Sie zum Installieren dieses Menüs bei laufendem Film die Anweisung installMenu member("CustomMenu2"). Das Menüelement "Konvertieren" führt die benutzerdefinierte Prozedur convertThis aus.

```
menu: File
Open/O | _movie.go("Open")
Close/W | _movie.go("Close")
Convert/C | convertThis
    (-
Quit/Q | _movie.go("Quit")
```

Siehe auch

installMenu, name, number (Menüelemente), checkMark, enabled, script

NAN

Syntax

```
-- Lingo syntax
NAN
```

Beschreibung

Dieser Rückgabewert gibt an, dass der jeweilige Lingo-Ausdruck keine Zahl ist.

Die folgende Anweisung versucht, die Quadratwurzel von -1 (keine gültige Zahl) im Nachrichtenfenster anzuzeigen:

```
-- Lingo syntax
put((-1).sqrt) -- NAN
```

Siehe auch

INF

next

Syntax

```
-- Lingo syntax
next
```

Beschreibung

Dieses Schlüsselwort bezieht sich auf die nächste Markierung im Film und hat die gleiche Bedeutung wie die Phrase `the marker (+ 1)`.

Beispiel

Die folgende Anweisung schickt den Abspielkopf zur nächsten Markierung im Film:

```
go next
```

Die folgende Prozedur verschiebt den Film zur nächsten Markierung im Drehbuch, wenn die rechte Pfeiltaste gedrückt wird, und zur vorherigen Markierung, wenn die linke Pfeiltaste gedrückt wird:

```
on keyUp
    if (_key.keyCode = 124) then _movie.goNext()
    if (_key.keyCode = 123) then _movie.goPrevious()
end keyUp
```

Siehe auch

loop (Schlüsselwort), goPrevious()

next repeat

Syntax

```
-- Lingo syntax
next repeat
```

Beschreibung

Dieses Schlüsselwort sendet Lingo zum nächsten Schritt in einer Wiederholungsschleife in einem Skript. Diese Funktion unterscheidet sich von der des Schlüsselworts `exit repeat`.

Beispiel

Bei der folgenden Wiederholungsschleife werden im Nachrichtenfenster nur ungerade Zahlen angezeigt:

```
repeat with i = 1 to 10
    if (i mod 2) = 0 then next repeat
    put(i)
end repeat
```

on

Syntax

```
-- Lingo syntax
on handlerName {argument1}, {arg2}, {arg3} ...
    statement(s)
end handlerName
```

Beschreibung

Dieses Schlüsselwort kennzeichnet den Anfang einer Prozedur, d. h. einer Sammlung von Lingo-Anweisungen, die Sie unter Verwendung des Prozedurnamens ausführen können. Eine Prozedur kann Argumente als Eingabewerte akzeptieren und gibt als Funktionsergebnis einen Wert zurück.

Prozeduren können in Verhalten, Film- und Darstellerskripten definiert werden. Eine in einem Darstellerskript befindliche Prozedur kann nur von anderen Prozeduren im selben Skript aufgerufen werden. Eine Prozedur in einem Filmskript kann jedoch von überall aufgerufen werden.

Sie können die gleiche Prozedur in mehreren Filmen verwenden, indem Sie das Skript der Prozedur in eine gemeinsam genutzte Besetzung aufnehmen.

otherwise

Syntax

```
-- Lingo syntax
otherwise statement(s)
```

Beschreibung

Dieses Schlüsselwort steht vor den Anweisungen, die Lingo ausführt, wenn keine der vorangegangenen Bedingungen in einer `case`-Anweisung erfüllt sind.

Dieses Schlüsselwort kann verwendet werden, um Benutzer vor außerhalb des Gültigkeitsbereichs liegenden Eingaben sowie vor ungültigen Typen zu warnen. Außerdem kann dieses Schlüsselwort während der Entwicklung beim Debugging sehr nützlich sein.

Beispiel

Die folgende Prozedur prüft, welche Taste der Benutzer zuletzt gedrückt hat, und reagiert entsprechend:

- Hat der Benutzer A, B oder C gedrückt, führt der Film die auf das Schlüsselwort of folgende Aktion aus.
- Hat der Benutzer dagegen eine andere Taste gedrückt, führt der Film die auf das Schlüsselwort otherwise folgende Anweisung aus. In diesem Fall besteht die Anweisung aus einem einfachen Warnhinweis.

```
//Lingo
on keyDown
     case (_key.key) of
        "a": _movie.go("Apple")
        "b", "c":
             _movie.puppetTransition(99)
             _movie.go("Oranges")
        otherwise: _player.alert("That is not a valid key.")
    end case
end keyDown

// Javascript
function keyDown()
{
    switch(_key.key)
    {
        case "a":
        _movie.go("Apple");
        break;
        case "b":
        case "c":
        _movie.puppetTransition(99);
        _movie.go("Oranges");
        break;
        default:
        _player.alert("That is not a valid key.");
    }
}
```

property

Syntax

```
-- Lingo syntax
property {property1}{, property2} {,property3} {...}
```

Beschreibung

Dieses Schlüsselwort deklariert die in *property1*, *property2* usw. angegebenen Eigenschaften als Eigenschaftsvariablen.

Eigenschaftsvariablen müssen zu Beginn des Parent-Skripts oder des Verhaltensskripts deklariert werden. Sie können auf diese Skripts auch von außerhalb des Parent-Skripts oder Verhaltensskripts über den Operator `the` zugreifen.

Hinweis: *Die Eigenschaft* `spriteNum` *ist für sämtliche Verhalten verfügbar. Sie brauchen diese Eigenschaft nur zu deklarieren, um darauf zugreifen zu können.*

Sie können eine Eigenschaft im übergeordneten oder Verhaltensskript referenzieren, ohne das Schlüsselwort `me` verwenden zu müssen. Um jedoch eine Eigenschaft eines Ancestor-Skripts des übergeordneten Skripts referenzieren zu können, müssen Sie die Form `me.property` verwenden.

Bei Verhalten sind Eigenschaften, die in einem Verhaltensskript definiert wurden, auch für andere Verhalten verfügbar, die am selben Sprite angebracht sind.

Die Eigenschaft eines untergeordneten Objekts kann außerhalb des übergeordneten Skripts des Objekts direkt manipuliert werden, und zwar mit Hilfe einer Syntax, die der zum Manipulieren anderer Eigenschaften verwendeten sehr ähnlich ist. Die folgende Anweisung stellt beispielsweise die Eigenschaft `motionStyle` für ein untergeordnetes Objekt ein:

```
set the motionStyle of myBouncingObject to #frenetic
```

Mit der Funktion `count` kann die Anzahl der Eigenschaften im Parent-Skript eines Child-Objekts bestimmt werden. Sie können die Namen dieser Eigenschaften über `getPropAt` abrufen. Wenn Sie einem Objekt Eigenschaften hinzufügen möchten, müssen Sie `setaProp()` verwenden.

Ein Beispiel für `property` in einem fertigen Film finden Sie im Film „Parent Scripts“ im Ordner „Learning\\Lingo_Examples“ im Director-Anwendungsordner.

Beispiel

Die folgende Anweisung sorgt dafür, dass jedes aus einem übergeordneten Skript erstellte untergeordnete Objekt eine eigene Positions- und Geschwindigkeitseinstellung aufweisen kann:

```
property location, velocity
```

Die folgende Parent-Skript-Prozedur deklariert pMySpriteNum als Eigenschaft, damit sie zur Verfügung steht:

```
-- script Elder
property pMyChannel
on new me, whichSprite
    me.pMyChannel = whichSprite
    return me
end
```

Das ursprüngliche Verhaltensskript erstellt den Ancestor und gibt die Eigenschaft spriteNum an alle Verhalten weiter:

```
property spriteNum

property ancestor

on beginSprite me
    ancestor = new script("Elder", spriteNum)
end
```

Siehe auch

`end`, `ancestor`, `spriteNum`

put...after

Syntax

```
-- Lingo syntax
put expression after chunkExpression
```

Beschreibung

Dieser Befehl wertet einen Lingo-Ausdruck aus, konvertiert den Wert in einen String und fügt diesen dann nach dem angegebenen Chunk in den Container ein, ohne dass der Containerinhalt dadurch ersetzt wird. (Falls *chunkExpression* einen nicht vorhandenen Ziel-Chunk angibt, wird der Wert des Strings dementsprechend in den Container eingefügt.)

Chunk-Ausdrücke können beliebige Zeichen, Wörter, Elemente oder Zeilen in einem Container sein. Zu Containern zählen Felddarsteller, Textdarsteller, Variablen, die Strings enthalten, sowie spezifische Zeichen, Wörter, Elemente, Zeilen und Bereiche in Containern.

Beispiel

Die folgende Anweisung fügt den String „fox dog cat“ nach dem Inhalt des Felddarstellers „Animal List“ hinzu:

```
put("fox dog cat") after member("Animal List")
```

Das Gleiche kann durch folgende Syntax erreicht werden:

```
put "fox dog cat" after member("Animal List").line[1]
```

Siehe auch

char...of, item...of, line...of, paragraph, word...of, put...before, put...into

put...before

Syntax

```
-- Lingo syntax
put expression before chunkExpression
```

Beschreibung

Dieser Befehl wertet einen Lingo-Ausdruck aus, konvertiert den Wert in einen String und fügt diesen dann vor dem angegebenen Chunk in den Container ein, ohne dass der Containerinhalt dadurch ersetzt wird. (Falls *chunkExpression* einen nicht vorhandenen Ziel-Chunk angibt, wird der Wert des Strings dementsprechend in den Container eingefügt.)

Chunk-Ausdrücke können beliebige Zeichen, Wörter, Elemente oder Zeilen in einem Container sein. Zu Containern zählen Felddarsteller, Textdarsteller, Variablen, die Strings enthalten, sowie spezifische Zeichen, Wörter, Elemente, Zeilen und Bereiche in Containern.

Beispiel

Die folgende Anweisung setzt die Variable "animalList" auf den String "fox dog cat" und fügt dann das Wort *elk* vor dem zweiten Wort in der Liste ein:

```
put "fox dog cat" into animalList
put "elk " before word 2 of animalList
```

Das Ergebnis ist der String "fox elk dog cat".

Das Gleiche kann durch folgende Syntax erreicht werden:

```
put "fox dog cat" into animalList
put "elk " before animalList.word[2]
```

Siehe auch

char...of, item...of, line...of, paragraph, word...of, put...after, put...into

put...into

Syntax

```
-- Lingo syntax
put expression into chunkExpression
```

Beschreibung

Dieser Befehl wertet einen Lingo-Ausdruck aus, konvertiert den Wert in einen String und ersetzt durch diesen dann den angegebenen Chunk in einem Container. (Falls *chunkExpression* einen nicht vorhandenen Ziel-Chunk angibt, wird der Wert des Strings dementsprechend in den Container eingefügt.)

Chunk-Ausdrücke können beliebige Zeichen, Wörter, Elemente oder Zeilen in einem Container sein. Zu Containern zählen Felddarsteller, Textdarsteller, Variablen, die Strings enthalten, sowie spezifische Zeichen, Wörter, Elemente, Zeilen und Bereiche in Containern.

Wenn ein Film als Applet abgespielt wird, ersetzt der Befehl `put...into` den gesamten Text im Container und nicht nur Text-Chunks.

Um Variablen Werte zuzuordnen, müssen Sie den Befehl `set` verwenden.

Beispiel

Die folgende Anweisung ändert die zweite Zeile des Felddarstellers „Review Comments“ in „Reviewed by Agnes Gooch“:

```
put "Reviewed by Agnes Gooch" into line 2 of member("Review Comments")
```

Das Gleiche kann durch einen Textdarsteller unter Verwendung folgender Syntax erreicht werden:

```
put "Reviewed by Agnes Gooch" into member("Review Comments").line[2]

// Javascript
member("Review Comments").setProp("line",2,"Reviewed by Agnes Gooch")
repeat while:
```

Siehe auch

`char...of`, `item...of`, `line...of`, `paragraph`, `word...of`, `put...before`, `put...after`, `set...to`, `set...=`

repeat while

Syntax

```
-- Lingo syntax
repeat while testCondition
    statement(s)
end repeat
```

Beschreibung

Dieses Schlüsselwort bewirkt, dass die *Anweisung(en)* so lange wiederholt ausgeführt werden, wie die in *testCondition* angegebene Bedingung `TRUE` ist. Diese Struktur kann in Lingo-Segmenten verwendet werden, die entweder Strings bis zum Ende einer Datei lesen, Elemente bis zum Ende einer Liste prüfen oder eine Aktion wiederholt ausführen, bis der Benutzer mit der Maustaste klickt oder diese loslässt.

Während sich Lingo in einer Wiederholungsschleife befindet, werden alle anderen Ereignisse ignoriert. Um zu prüfen, welche Taste in einer Wiederholungsschleife gedrückt wurde, müssen Sie die Eigenschaft `keyPressed` verwenden.

Es kann jeweils immer nur eine Prozedur ausgeführt werden. Falls sich Lingo sehr lange in einer Wiederholungsschleife befindet, stauen sich schnell die anderen Ereignisse, die noch ausgewertet werden müssen. Wiederholungsschleifen sollten deshalb möglichst nur für kurze, schnelle Vorgänge verwendet werden oder, wenn Sie genau wissen, dass der Benutzer nichts anderes ausführt.

Wenn die Verarbeitung eines Elements länger als ein paar Sekunden dauert, sollten Sie die Funktion in einer Schleife mit einem Zähler oder Test auswerten, um den Fortschritt verfolgen zu können.

Wenn die Stoppbedingung nie erreicht wird oder kein Ausstieg aus der Wiederholungsschleife vorgesehen ist, können Sie Director mit Strg+Alt+Punkt (Windows) bzw. Befehl+Punkt (Macintosh) zum Abbruch zwingen.

Beispiel

Die folgende Prozedur startet den Zähler, setzt ihn auf 0 zurück und lässt ihn dann auf 60 Millisekunden hochzählen:

```
on countTime
    _system.milliseconds
    repeat while _system.milliseconds < 60
    -- waiting for time
    end repeat
end countTime

// Javascript
function countTime()
{
    _system.milliseconds
    while (_system.milliseconds < 60)
    {
        -- waiting for time
    }
}
```

Siehe auch

`exit`, `exit repeat`, `repeat with`, `keyPressed()`

repeat with

Syntax

```
-- Lingo syntax
repeat with counter = start to finish
    statement(s)
end repeat
```

Beschreibung

Dieses Schlüsselwort führt die angegebenen Lingo-Anweisungen so oft aus, wie in *counter* festgelegt ist. Der Wert von *counter* besteht aus der Differenz zwischen den Werten *start* und *finish*. Der Zähler wird bei jedem Durchlauf der Wiederholungsschleife um 1 erhöht.

Die Struktur `repeat with` eignet sich besonders dazu, einer Reihe von Sprites wiederholt die gleichen Effekte zuzuordnen oder eine Zahlenfolge zu berechnen (z. B. um eine Zahl zu potenzieren).

Während sich Lingo in einer Wiederholungsschleife befindet, werden alle anderen Ereignisse ignoriert. Um zu prüfen, welche Taste in einer Wiederholungsschleife gedrückt wurde, müssen Sie die Eigenschaft `keyPressed` verwenden.

Es kann jeweils immer nur eine Prozedur ausgeführt werden. Falls sich Lingo sehr lange in einer Wiederholungsschleife befindet, stauen sich schnell die anderen Ereignisse, die noch ausgewertet werden müssen. Wiederholungsschleifen sollten deshalb möglichst nur für kurze, schnelle Vorgänge verwendet werden oder, wenn Sie genau wissen, dass der Benutzer nichts anderes ausführt.

Wenn die Verarbeitung eines Elements länger als ein paar Sekunden dauert, sollten Sie die Funktion in einer Schleife mit einem Zähler oder Test auswerten, um den Fortschritt verfolgen zu können.

Wenn die Stoppbedingung nie erreicht wird oder kein Ausstieg aus der Wiederholungsschleife vorgesehen ist, können Sie Director mit Strg+Alt+Punkt (Windows) bzw. Befehl+Punkt (Macintosh) zum Abbruch zwingen.

Beispiel

Die folgende Prozedur wandelt die Sprites 1 bis 30 in Puppets um:

```
//Lingo
on puppetize
    repeat with channel = 1 to 30
    _movie.puppetSprite(channel, TRUE)
    end repeat
end puppetize

// Javascript
function puppetize()
{
    for(var channel=1;channel<30;channel++)
    {
    _movie.puppetSprite(channel, true);
    }
}
```

Siehe auch

exit, exit repeat, repeat while, repeat with...down to, repeat with...in list

repeat with...down to

Syntax

```
-- Lingo syntax
repeat with variable = startValue down to endValue
```

Beschreibung

Dieses Schlüsselwort zählt abwärts von *startValue* bis *endValue*, und zwar in Schritten von 1.

Es kann jeweils immer nur eine Prozedur ausgeführt werden. Falls sich Lingo sehr lange in einer Wiederholungsschleife befindet, stauen sich schnell die anderen Ereignisse, die noch ausgewertet werden müssen. Wiederholungsschleifen sollten deshalb möglichst nur für kurze, schnelle Vorgänge verwendet werden oder wenn Sie genau wissen, dass der Anwender nichts anderes ausführt.

Während sich Lingo in einer Wiederholungsschleife befindet, werden alle anderen Ereignisse ignoriert. Um zu prüfen, welche Taste in einer Wiederholungsschleife gedrückt wurde, müssen Sie die Eigenschaft `keyPressed` verwenden.

Wenn die Verarbeitung eines Elements länger als ein paar Sekunden dauert, sollten Sie die Funktion in einer Schleife mit einem Zähler oder Test auswerten, um den Fortschritt verfolgen zu können.

Wenn die Stoppbedingung nie erreicht wird oder kein Ausstieg aus der Wiederholungsschleife vorgesehen ist, können Sie Director mit Strg+Alt+Punkt (Windows) bzw. Befehl+Punkt (Macintosh) zum Abbruch zwingen.

Beispiel
Die folgende Prozedur enthält eine Wiederholungsschleife, die von 20 auf 15 herunterzählt:

```
on countDown
    repeat with i = 20 down to 15
        sprite(6).member = 10 + i
        _movie.updateStage()
    end repeat
end
```

repeat with...in list

Syntax

```
-- Lingo syntax
repeat with variable in someList
```

Beschreibung
Dieses Schlüsselwort ordnet der Variablen aufeinanderfolgende Werte aus der angegebenen Liste zu.

Während sich Lingo in einer Wiederholungsschleife befindet, werden alle anderen Ereignisse außer Tastenaktionen ignoriert. Um zu prüfen, welche Taste in einer Wiederholungsschleife gedrückt wurde, müssen Sie die Eigenschaft `keyPressed` verwenden.

Es kann jeweils immer nur eine Prozedur ausgeführt werden. Falls sich Lingo sehr lange in einer Wiederholungsschleife befindet, stauen sich schnell die anderen Ereignisse, die noch ausgewertet werden müssen. Wiederholungsschleifen sollten deshalb möglichst nur für kurze, schnelle Vorgänge verwendet werden oder, wenn Sie genau wissen, dass der Benutzer nichts anderes ausführt.

Wenn die Verarbeitung eines Elements länger als ein paar Sekunden dauert, sollten Sie die Funktion in einer Schleife mit einem Zähler oder Test auswerten, um den Fortschritt verfolgen zu können.

Wenn die Stoppbedingung nie erreicht wird oder kein Ausstieg aus der Wiederholungsschleife vorgesehen ist, können Sie Director mit Strg+Alt+Punkt (Windows) bzw. Befehl+Punkt (Macintosh) zum Abbruch zwingen.

Beispiel
Die folgende Anweisung zeigt im Nachrichtenfenster vier Werte an:

```
//Lingo
repeat with i in [1, 2, 3, 4]
    put(i)
end repeat

// Javascript
var nl = new Array()
nl =[1,2,3,4]
var i
for(i in nl)
{
    trace(nl[i])
}
```

return (Schlüsselwort)

Syntax

```
-- Lingo syntax
return expression
```

Beschreibung

Dieses Schlüsselwort gibt den Wert für *expression* zurück und verlässt dann die Prozedur. Beim Argument *expression* kann es sich um einen beliebigen Lingo-Wert handeln.

Wenn Sie eine Prozedur aufrufen, die als benutzerdefinierte Funktion dient und einen Rückgabewert besitzt, müssen Sie die Argumentelisten in Klammern setzen. Dies ist selbst dann notwendig, wenn gar keine Argumente vorhanden sind, wie z. B. bei der Funktionsprozedur `diceRoll`, die bereits unter der Funktion `result` erörtert wurde.

Das Schlüsselwort `return` funktioniert ähnlich wie der Befehl `exit`, aber bei `return` wird außerdem ein Wert an die aufrufende Anweisung zurückgegeben. Mit anderen Worten, der Befehl `return` verlässt die Prozedur unverzüglich, kann aber an die aufrufende Lingo-Anweisung einen Wert zurückgeben.

Die Verwendung von `return` bei der objektorientierten Skripterstellung kann zu Anfang schwer verständlich sein. Es ist einfacher, `return` anfänglich erst einmal zur Erstellung von Funktionen undAusstiegsprozeduren zu verwenden. Später werden Sie dann von selbst erkennen, dass Ihnen die Zeile `return me` in der Prozedur `on new` die Möglichkeit gibt, einen Bezug auf ein Objekt zurückzugeben, das erstellt wurde, damit ihm ein Variablenname zugeordnet werden kann.

Das Schlüsselwort `return` unterscheidet sich von der Zeichenkonstanten `RETURN`, die eine Zeilenschaltung angibt. Der Gebrauch der Funktion hängt ganz vom Kontext ab.

Um einen zurückgegebenen Wert abzurufen, müssen Sie in der Aufrufsanweisung nach dem Prozedurnamen Klammern setzen, um darauf hinzuweisen, dass es sich bei der benannten Prozedur um eine Funktion handelt.

Ein Beispiel für `return (keyword)` in einem fertigen Film finden Sie im Film „Parent Scripts“ im Ordner „Learning\\Lingo_Examples“ im Director-Anwendungsordner.

Beispiel

Die folgende Prozedur gibt nach dem Zufallsprinzip ein Vielfaches von 5 im Bereich 5 bis 100 zurück:

```
//Lingo
on getRandomScore
    theScore = 5 * random(20)
    return theScore
end getRandomScore

// Javascript
function getRandomScore()
{
    theScore = 5 * random(20);
    return theScore;
}
```

Sie können diese Prozedur z. B. mit folgender Anweisung aufrufen:

```
thisScore = getRandomScore()
```

Im folgenden Beispiel wird der Variablen `thisScore` der Rückgabewert aus der Funktion `getRandomScore()` zugeordnet. Ein übergeordnetes Skript nimmt dieselbe Funktion wahr: durch die Rückgabe einer Objektreferenz stellt die Variable „name“ im aufrufenden Code einen Anhaltspunkt für nachfolgende Bezugnamen auf das Objekt dar.

Siehe auch

`result`, `RETURN (constant)`

set...to, set...=

Syntax

```
-- Lingo syntax
lingoProperty = expression
variable = expression
```

Beschreibung

Dieser Befehl wertet einen Ausdruck aus und setzt das Resultat in die durch *lingoProperty* angegebene Eigenschaft oder in die durch *variable* angegebene Variable.

Beispiel

Diese Anweisung setzt den Namen von Darsteller 3 auf „Sunset“:

```
member(3).name = "Sunset"
```

Die folgende Anweisung kehrt die Eigenschaft soundEnabled in das Gegenteil ihres aktuellen Status um. Wenn soundEnabled den Wert TRUE hat, d. h. der Sound eingeschaltet ist, schaltet ihn diese Anweisung aus. Wenn soundEnabled auf FALSE gesetzt ist, d. h. der Sound ausgeschaltet ist, schaltet ihn diese Anweisung ein.

```
_sound.soundEnabled = not(_sound.soundEnabled)
```

Die folgende Anweisung stellt die Variable "vowels" auf den String "aeiou" ein:

```
vowels = "aeiou"
```

Siehe auch

`property`

sprite...intersects

Syntax

```
-- Lingo syntax
sprite(sprite1).intersects(sprite2)
sprite sprite1 intersects sprite2
```

Beschreibung

Dieses Schlüsselwort ist ein Operator, der die Position von zwei Sprites vergleicht, um festzustellen, ob das Begrenzungsrechteck von *sprite1* das Begrenzungsrechteck von *sprite2* berührt (TRUE) oder nicht berührt (FALSE).

Falls beide Sprites über den Farbeffekt „Matt“ verfügen, werden nicht ihre Begrenzungsrechtecke, sondern ihre eigentlichen Umrisse zum Vergleich verwendet. Der Umriss eines Sprites ist durch die nicht-weißen Pixel definiert, die seinen Rand bilden.

Dieser Vergleichsoperator hat die Prioritätsstufe 5.

Hinweis: *Der Punktoperator wird verwendet, wenn „sprite1“ kein einzelner Ausdruck ist, sondern z. B. ein Ausdruck mit einer mathematischen Operation.*

Beispiel

Die folgende Anweisung prüft, ob zwei Sprites sich schneiden, und ändert den Inhalt des Felddarstellers „Notice“ in „You placed it correctly.“, wenn dies der Fall ist.:

```
// Lingo Syntax
if sprite i intersects j then put("You placed it correctly.") into member("Notice")

// Javascript
if (sprite(i).intersects(sprite(j)))
{
    member("Notice").text="You placed it correctly";
}
```

Siehe auch

sprite...within, quad

sprite...within

Syntax

```
-- Lingo syntax
sprite(sprite1).within(sprite2)
sprite sprite1 within sprite2
```

Beschreibung

Dieses Schlüsselwort ist ein Operator, der die Position von zwei Sprites vergleicht und feststellt, ob sich das Begrenzungsrechteck von *sprite1* vollständig innerhalb des Begrenzungsrechtecks von *sprite2* befindet (TRUE) oder nicht (FALSE).

Falls beide Sprites über den Farbeffekt „Matt“ verfügen, werden nicht ihre Begrenzungsrechtecke, sondern ihre eigentlichen Umrisse zum Vergleich verwendet. Der Umriss eines Sprites ist durch die nicht-weißen Pixel definiert, die seinen Rand bilden.

Dieser Vergleichsoperator hat die Prioritätsstufe 5.

Hinweis: *Der Punktoperator wird verwendet, wenn „sprite1“ kein einzelner Ausdruck ist, sondern z. B. ein Ausdruck mit einer mathematischen Operation.*

Beispiel

Die folgende Anweisung prüft, ob sich ein Sprite innerhalb eines anderen befindet, und ruft die Prozedur doInside auf, wenn dies der Fall ist:

```
//Lingo
if sprite(3).within(2) then doInside

// Javascript
if (sprite(3).within(2))
{
    doInside();
}
```

Siehe auch

sprite...intersects, quad

version

Syntax

```
-- Lingo syntax
_player.productVersion
```

Beschreibung

Dieses Schlüsselwort ist eine Systemvariable, die die Version von Director enthält. Derselbe String erscheint im Fenster „Info“ im Macintosh-Finder.

Beispiel

Die folgende Anweisung zeigt die Version von Director im Nachrichtenfenster an:

```
put(_player.productVersion)
```

word...of

Syntax

```
-- Lingo syntax
member(whichCastMember).word[whichWord]
textMemberExpression.word[whichWord]
chunkExpression.word[whichWord]
word whichWord of fieldOrStringVariable
fieldOrStringVariable. word[whichWord]
textMemberExpression.word[firstWord..lastWord]
member(whichCastMember).word[firstWord..lastWord]
word firstWord to lastWord of chunkExpression
chunkExpression.word[whichWord..lastWord]
```

Beschreibung

Dieser Chunk-Ausdruck gibt ein Wort oder einen Wortbereich in einem Chunk-Ausdruck an. Ein Wort-Chunk ist eine beliebige Folge von Zeichen, die durch Leerzeichen begrenzt sind. (Alle unsichtbaren Zeichen, wie z. B. Tabulator oder Zeilenschaltung, werden als Leerzeichen betrachtet.)

Die Ausdrücke *whichWord*, *firstWord* und *lastWord* müssen Ganzzahlen ergeben, die für ein Wort in einem Chunk stehen.

Chunk-Ausdrücke verweisen auf beliebige Zeichen, Wörter, Elemente oder Zeilen in beliebigen Zeichenquellen. Zeichenquellen können auch Feld- und Textdarsteller sowie Variablen sein, die Strings enthalten.

Ein Beispiel für `word...of` in einem fertigen Film finden Sie im Film „Text“ im Ordner „Learning/Lingo_Examples“ im Director-Anwendungsordner.

Beispiel

Die folgenden Anweisungen setzen die Variable „animalList“ auf den String „fox dog cat“ und fügen dann das Wort „elk“ vor dem zweiten Wort in der Liste ein:

```
animalList = "fox dog cat"
put "elk" before animalList.word[2]
```

Das Ergebnis ist der String "fox elk dog cat".

Die folgende Anweisung veranlasst Director, das fünfte Wort aus demselben String im Nachrichtenfenster anzuzeigen:

```
put "fox elk dog cat".word[5]
```

Da dieser String kein fünftes Wort enthält, zeigt das Nachrichtenfenster zwei Anführungszeichen an (""), die für einen leeren String stehen.

Siehe auch

`char...of`, `line...of`, `item...of`, `count()`, `number (Wörter)`

Kapitel 12: Methoden

In diesem Abschnitt finden Sie eine alphabetische Liste aller in Director® verfügbaren Methoden.

_system.gc()

Syntax

```
_system.gc();
```

Beschreibung

Ruft die JavaScript-Garbage-Collection-Methode auf. `_system.gc()` wird nur von JavaScript unterstützt. Mit dieser Methode können Sie festlegen, wann Garbage-Collection-Vorgänge ausgelöst werden. Sie können diese Methode auch ausführen, wenn Sie nur Lingo-Code schreiben und globale Variablen nutzen. Die Speicherverwaltung der beiden Skriptsprachen unterscheidet sich: das JavaScript-Runtime-Modul verwendet Garbage Collection; das Lingo-Runtime-Modul nicht. Nicht mehr benötigte Speicherblöcke, die von JavaScript reserviert wurden, stehen dem System erst wieder zur Verfügung, wenn sie durch den Java Script-Garbage Collection-Vorgang freigegeben werden. Der Java Script-Garbage Collection-Vorgang lässt sich durch den Aufruf der Routine `_system.gc()` herbeiführen. Auch das JavaScript-Modul führt zur Laufzeit periodisch Garbage-Collection-Vorgänge aus. Wenn allerdings nur Lingo-Code ausgeführt wird, wird der `_system.gc()`-Aufruf nicht verwendet. Auch in reinem Lingo-Code wird für eine als global deklarierte Variable ein JavaScript-Referenzobjekt erstellt, damit ein möglicherweise später ausgeführter JavaScript-Code auf die Variable zugreifen kann. Sobald Lingo die Variable freigibt, kann das JavaScript-Referenzobjekt und die Variablendaten durch die Garbage-Collection-Routine gelöscht werden.

Falls das JavaScript nicht zur Ausführung kommt, wird auch `_system.gc()` nicht aufgerufen und der gesamte Speicher kann durch Objekte verbraucht werden, die auf eine Rückführung zum System warten. Schreiben Sie als Abhilfe ein einfaches JavaScript-Programm in einem Filmskript, das Sie von Ihrem Lingo-Code aus aufrufen.

```
function clearJSMemory()
{
  _system.gc()
}
```

abort

Syntax

```
--Lingo syntax
abort

// JavaScript syntax
abort();
```

Beschreibung

Dieser Befehl veranlasst Lingo, die aktuelle Prozedur und alle Prozeduren, welche die aktuelle Prozedur aufgerufen haben, zu beenden, ohne die noch verbleibenden Anweisungen in der Prozedur auszuführen. Damit unterscheidet er sich vom Schlüsselwort `exit`, das zur Prozedur zurückkehrt, von der die aktuelle Prozedur aufgerufen wurde.

Der Befehl `abort` beendet Director nicht.

Parameter
Keiner

Beispiel
Die folgende Anweisung veranlasst Lingo, die aktuelle Prozedur und alle Prozeduren, die diese aufgerufen haben, zu beenden, wenn der verfügbare Speicherplatz weniger als 50 KB beträgt:

```
-- Lingo syntax
if the freeBytes < 50*1024 then abort

// JavaScript syntax
if (_player.freeBytes < 50*1024) {
    abort()
}
```

Siehe auch
`exit`, `halt()`, `quit()`

abs()

Syntax

```
--Lingo syntax
abs (numericExpression)

// JavaScript syntax
Math.abs (numericExpression)
```

Beschreibung
Die Funktion `abs()` hat mehrere Verwendungsmöglichkeiten. Sie vereinfacht die Verfolgung von Maus- und Sprite-Bewegungen, indem sie Koordinatendifferenzen (die positiv oder negativ sein können) in Entfernungen (die immer positiv sind) konvertiert. Die Funktion `abs()` ist auch für mathematische Funktionen wie `sqrt()` und `log()` nützlich

In JavaScript-Syntax wird die `abs()`-Funktion des Math-Objekts verwendet.

Parameter
numericExpression Erforderlich. Eine Ganz- oder Fließkommazahl, aus der ein absoluter Wert berechnet wird. Wenn *numericExpression* eine Ganzzahl ist, ist der absolute Wert auch eine Ganzzahl. Wenn *numericExpression* eine Fließkommazahl ist, ist der absolute Wert auch eine Fließkommazahl.

Beispiel
Die folgende Anweisung bestimmt, ob der absolute Wert der Differenz zwischen der aktuellen Mausposition und dem Wert der Variablen startV größer als 30 ist (die Entfernung kann schließlich keine negative Zahl sein). Wenn dies der Fall ist, wird die Vordergrundfarbe von Sprite 6 geändert.

```
-- Lingo syntax
if (the mouseV - startV).abs > 30 then sprite(6).forecolor = 95

// JavaScript syntax
if ((_mouse.mouseV - Math.abs(_mouse.startV)) > 30) {
    sprite(6).foreColor = 95;
}
```

activateAtLoc()

Syntax

```
-- Lingo syntax
dvdObjRef.activateAtLoc(point(x, y))

// JavaScript syntax
dvdObjRef.activateAtLoc(point(x, y));
```

Beschreibung

Diese DVD-Methode aktiviert die Eigenschaft „hilite“ des eingebetteten DVD-Menüelements, das sich an einer angegebenen Bühnenposition befindet.

Bei Erfolg gibt diese Methode 0 zurück.

Parameter

`point`*(x, y)* Erforderlich. Ein Punkt in Bühnenkoordinaten, der die Position des eingebetteten DVD-Menüelements angibt.

Beispiel

Folgende Anweisung aktiviert die Eigenschaft „hilite“ des Menüelements an der angegebenen Bühnenposition:

```
-- Lingo syntax
member("movie1").activateAtLoc(point(100, 200))

// JavaScript syntax
member("movie1").activateAtLoc(point(100, 200));
```

Siehe auch

DVD

activateButton()

Syntax

```
-- Lingo syntax
dvdObjRef.activateButton()

// JavaScript syntax
dvdObjRef.activateButton();
```

Beschreibung
Diese DVD-Methode aktiviert die aktuell ausgewählte Menüschaltfläche.

Bei Erfolg gibt diese Methode 0 zurück.

Hinweis: *Diese Methode wird auf Macintosh®-Intel® nicht unterstützt.*

Parameter
Keiner

Beispiel
Folgende Anweisung aktiviert die Menüschaltfläche an einem angegebenen Darsteller:

```
-- Lingo syntax
sprite(1).member.activateButton()

// JavaScript syntax
sprite(1).member.activateButton();
```

Siehe auch
DVD

add

Syntax

```
-- Lingo syntax
linearList.add(value)

// JavaScript syntax
array.push(value);
```

Beschreibung
Dieser Listenbefehl ist nur auf lineare Listen anwendbar und fügt einer linearen Liste einen Wert hinzu. In einer geordneten Liste wird der Wert an der vorgesehenen Stelle eingefügt. In einer nicht geordneten Liste wird der Wert ans Ende der Liste gesetzt.

Wenn dieser Befehl mit einer Eigenschaftsliste verwendet wird, gibt er eine Fehlermeldung zurück.

Hinweis: *Verwechseln Sie den Befehl* `add` *nicht mit dem Additionsoperator* `+` *oder mit dem* `&`*-Operator, der zur Verkettung von Strings verwendet wird.*

Parameter
value Erforderlich. Ein der linearen Liste hinzuzufügender Wert.

Beispiel
Die folgenden Anweisungen fügen den Wert 2 zur Liste mit der Bezeichnung bids hinzu.

```
-- Lingo syntax
bids = [3, 4, 1]
bids.add(2)

// JavaScript syntax
bids = new Array(3,4,1);
bids.push(2);
```

Die folgende Anweisung fügt den Wert 2 zur geordneten linearen Liste [3, 4, 1] hinzu. Die resultierende Liste ist [3, 4, 1, 2].

```
-- Lingo syntax
-- sort the list using Lingo
bids.sort()
bids.add(2)

// JavaScript syntax
// sort the list using JavaScript
bids.sort();
bids.push(2);
```

„bids“ wird nach Wert sortiert und die Ergebnisliste ist [1, 2, 3, 4].

Siehe auch

`sort`, `addAt`, `append`

add (3D-Textur)

Syntax

```
--Lingo syntax
member(whichCastmember).model(whichModel).meshdeform.mesh[index].textureLayer.add()

// JavaScript syntax
member(whichCastmember).model(whichModel).meshdeform.mesh[index].textureLayer.add()
```

Beschreibung

Dieser Befehl für den 3D-Modifizierer `meshdeform` fügt zum Gitternetz des Modells eine leere Texturebene hinzu.

Mit dem folgenden Code können Sie Texturkoordinaten zwischen Ebenen kopieren:

```
modelReference.meshdeform.texturelayer[a].texturecoordinatelist =
modelReference.meshdeform.texturelayer[b].texturecoordinatelist
```

Parameter

Keiner

Beispiel

Die folgende Anweisung erstellt eine neue Texturebene für das erste Gitternetz des Modells "Ohr“:

```
--Lingo syntax
member("Scene").model("Ear").meshdeform.mesh[1].textureLayer.add()

// JavaScript syntax
member("Scene").getprop("model","Ear").meshdeform.mesh[1].textureLayer.add();
```

Siehe auch

meshDeform (Modifizierer), textureLayer, textureCoordinateList

addAt

Syntax

```
list.AddAt(position, value)
```

Beschreibung

Dieser Listenbefehl ist nur auf lineare Listen anwendbar und fügt einer Liste an einer angegebenen Position einen Wert hinzu.

Wenn dieser Befehl mit einer Eigenschaftsliste verwendet wird, gibt er eine Fehlermeldung zurück.

Parameter

position Erforderlich. Ein Ganzzahlwert, der die Position in der Liste angibt, an der der durch *value* angegebene Wert hinzugefügt wird.

value Erforderlich. Ein der Liste hinzuzufügender Wert.

Beispiel

Die folgende Anweisung fügt den Wert 8 an der vierten Position der Liste bids ein, die [3, 2, 4, 5, 6, 7] lautet:

```
--Lingo
bids = [3, 2, 4, 5, 6, 7]
bids.addAt(4,8)

// Javascript
bids = list(3, 2, 4, 5, 6, 7)
bids.addAt(4,8)
```

Der resultierende Wert von "bids" lautet [3, 2, 4, 8, 5, 6, 7].

addBackdrop

Syntax

```
-- Lingo syntax
sprite(whichSprite).camera{(index)}.addBackdrop(texture, locWithinSprite, rotation)
member(whichCastmember).camera(whichCamera).addBackdrop(texture, locWithinSprite, rotation)

// JavaScript syntax
sprite(whichSprite).camera{(index)}.addBackdrop(texture, locWithinSprite, rotation);
member(whichCastmember).camera(whichCamera).addBackdrop(texture, locWithinSprite, rotation);
```

Beschreibung
Dieser 3D-Kamerabefehl fügt am Ende der Hintergrundliste der Kamera einen weiteren Hintergrund hinzu.

Parameter
texture Erforderlich. Die auf den Hintergrund anzuwendende Textur.

locWithinSprite Erforderlich. Eine 2D-Position, an der der Hintergrund im 3D-Sprite angezeigt wird. Die Position wird von der linken oberen Ecke des Sprites an gemessen.

rotation Erforderlich. Eine Ganzzahl, die die Gradzahl angibt, um die die Textur gedreht werden soll.

Beispiel
Die erste Zeile der folgenden Anweisung erstellt eine Textur namens „Rau“ aus dem Darsteller „Zeder“ und speichert sie in der Variablen „t1“. Die zweite Zeile wendet die Textur am Punkt (220, 220) in Sprite 5 als Hintergrund an. Die Textur ist um 0° gedreht. Die letzte Zeile wendet die gleiche Textur als Hintergrund für Kamera 1 des Darstellers "Szene“ am Punkt (20, 20) mit einem Drehwinkel von 45° an.

```
t1 = member("Scene").newTexture("Rough", #fromCastMember,member("Cedar"))
sprite(5).camera.addBackdrop(t1, point(220, 220), 0)
member("Scene").camera[1].addBackdrop(t1, point(20, 20), 45)

// Javascript
var t1 = member("Scene").newTexture("Rough", symbol("fromCastMember"),member("Cedar"))
sprite(5).camera.addBackdrop(t1, point(220, 220), 0) ;
member("Scene").getPropRef("camera",1).addBackdrop(t1, point(20, 20), 45) ;
```

Siehe auch
`removeBackdrop`

addCamera

Syntax
```
-- Lingo syntax
sprite(whichSprite).addCamera(whichCamera, index)
-- JavaScript syntax
sprite(whichSprite).addCamera(whichCamera, index);
```

Beschreibung
Dieser 3D-Befehl fügt der Kameraliste für das Sprite eine Kamera hinzu. Die Ansicht von den einzelnen Kameras aus erscheint über der Ansicht von Kameras mit niedrigeren *index*-Positionen. Sie können für jede Kamera die Eigenschaft `rect` einstellen, um mehrere Ansichten im Sprite darzustellen.

Parameter
whichCamera Erforderlich. Ein Verweis auf die Kamera, die der Kameraliste für das Sprite hinzugefügt werden soll.

index Erforderlich. Eine Ganzzahl, die die Indexposition in der Kameraliste angibt, an der *whichCamera* hinzugefügt wird. Wenn *index* größer ist als der aktuelle Wert von `cameraCount()`, wird die Kamera am Ende der Liste hinzugefügt.

Beispiel

Die folgende Anweisung fügt die Kamera `FlightCam` an der fünften Indexposition in die Kameraliste von Sprite 12 ein:

```
--Lingo syntax
sprite(12).addCamera(member("scene").camera("FlightCam"), 5)

// JavaScript syntax
sprite(12).addCamera(member("scene").getPropRef("camera", i), 5);
// where i is the number index for the camera "FlightCam".
```

Siehe auch

`cameraCount()`, `deleteCamera`

addChild

Syntax

```
-- Lingo syntax
member(whichCastmember).node(whichParentNode).addChild(member(whichCastmember).node(whichChi
ldNode) {,#preserveWorld})

// JavaScript syntax
member(whichCastmember).node(whichParentNode).addChild(member(whichCastmember).node(whichChi
ldNode) {,symbol(preserveWorld)})
```

Beschreibung

Dieser 3D-Befehl fügt der Cild-Liste eines Nodes einen Node hinzu und entfernt diesen aus der Child-Liste seines früheren Parents.

Das Gleiche erreichen Sie auch, indem Sie die Eigenschaft `parent` des Child-Nodes auf den Parent-Node festlegen.

Parameter

addMemberRef Erforderlich. Ein Verweis auf den Darsteller, der den hinzuzufügenden Node enthält.

addNodeRef Erforderlich. Ein Verweis auf den hinzuzufügenden Node. Dieser Node kann ein Modell, eine Gruppe, eine Kamera oder ein Licht sein.

symPreserveParentOrWorld Optional. Ein Verweis auf die Kamera, die der Kameraliste für das Sprite hinzugefügt werden soll. Gültige Werte sind: `#preserveWorld` und `#preserveParent`. Wenn das Child unter Angabe von `#preserveParent` hinzugefügt wird, bleibt die auf sein Parent-Objekt bezogene Transformation unverändert, und das Child springt im Raum des neuen Parents zu dieser Transformation. Die Welttransformation des Child wird neu berechnet. Wenn das Child unter Angabe von `#preserveWorld` hinzugefügt wird, bleibt seine Welttransformation unverändert, und das Child springt nicht zu seiner Transformation im Raum des neuen Parents. Seine Parent-bezogene Transformation wird neu berechnet.

Beispiel

Die folgende Anweisung fügt das Modell "Reifen" zur Child-Liste des Modells "Auto" hinzu:

```
-- Lingo syntax
member("3D").model("Car").addChild(member("3D").model("Tire"))

// JavaScript syntax
member("3D").getProp("model" , i ).addChild(member("3D").getProp("model" , j));
// where i is the number index for model "Car" and j is the number index for model "Tire".
```

Die folgende Anweisung fügt das Modell `Bird` zur Child-Liste der Kamera `MyCamera` hinzu und verwendet das Argument `#preserveWorld`, um die Weltposition von „Vogel“ beizubehalten:

```
-- Lingo syntax
member("3D").camera("MyCamera").addChild(member("3D").model
("Bird"), #preserveWorld)

// JavaScript syntax
member("3D").getPropRef("camera",j).addChild(member("3D").getProp("model",i),symbol("preserv
eWorld"))
// where i the number index of the model "Bird" and j is the number index of the camera
"MyCamera"
```

Siehe auch

parent, addToWorld, removeFromWorld

addModifier

Syntax

```
-- Lingo syntax
member(whichCastmember).model(whichModel).addModifier(#modifierType)

// JavaScript syntax
member(whichCastmember).model(whichModel).addModifier(symbol(modifierType));
```

Beschreibung

Dieser 3D-Modellbefehl fügt einen angegebenen Modifizierer zum Modell hinzu. Für diesen Befehl gibt es keinen Standardwert.

Parameter

symbolModType Erforderlich. Ein Symbol, das den hinzuzufügenden Modifiziere angibt. Es gibt folgende Modifizierer:

- `#bonesPlayer`
- `#collision`
- `#inker`
- `#keyframePlayer`
- `#lod`
- `#meshDeform`
- `#sds`
- `#toon`

Weitere Informationen zu diesen Modifizierern finden Sie in den einzelnen Beschreibungen.

Beispiel

Die folgende Anweisung fügt den Modifizierer "#toon" zum Modell "Box" hinzu:

```
-- Lingo syntax
member("shapes").model("Box").addModifier(#toon)

// JavaScript syntax
member("shapes").getPropRef("model" , a ).addModifier(symbol("toon"));
// where a is the number index for the "Box" model.
```

Siehe auch

bonesPlayer (Modifizierer), collision (modifier), inker (Modifizierer), keyframePlayer (Modifizierer), lod (Modifizierer), meshDeform (Modifizierer), sds (Modifizierer), toon (Modifizierer), getRendererServices(), removeModifier, modifier, modifier[], modifiers

addOverlay

Syntax

```
-- Lingo syntax
sprite(whichSprite).camera{(index)}.addOverlay(texture, locWithinSprite, rotation)
member(whichCastmember).camera(whichCamera).addOverlay(texture, locWithinSprite, rotation)

// JavaScript syntax
sprite(whichSprite).camera{(index)}.addOverlay(texture, locWithinSprite, rotation)
member(whichCastmember).camera(whichCamera).addOverlay(texture, locWithinSprite, rotation)
```

Beschreibung

Dieser 3D-Kamerabefehl fügt am Ende der Überlagerungsliste einer Kamera eine weitere Überlagerung hinzu.

Parameter

texture Erforderlich. Die auf die Überlagerung anzuwendende Textur.

locWithinSprite Erforderlich. Eine 2D-Position, an der die Überlagerung im 3D-Sprite angezeigt wird. Die Position wird von der linken oberen Ecke des Sprites an gemessen.

rotation Erforderlich. Eine Ganzzahl, die die Gradzahl angibt, um die die Textur gedreht werden soll.

Beispiel

Die erste Zeile der folgenden Anweisung erstellt eine Textur namens „Rau“ aus dem Darsteller `Cedar` und speichert sie in der Variablen „t1“. Die zweite Zeile wendet die Textur am Punkt (220, 220) in Sprite 5 als Überlagerung an. Die Textur ist um 0° gedreht. Die letzte Zeile dieser Anweisung wendet die gleiche Textur als Überlagerung für Kamera 20 des Darstellers „Szene“ am Punkt (20, 45) an.

```
-- Lingo syntax
t1 = member("Scene").newTexture("Rough", #fromCastMember,member("Cedar"))
sprite(5).camera.addOverlay(t1, point(220, 220), 0)
member("Scene").camera[1].addOverlay(t1, point(20, 20), 45)

// JavaScript syntax
t1 = member("Scene").newTexture("Rough", symbol("fromCastMember"),member("Cedar"));
sprite(5).camera.addOverlay(t1, point(220, 220), 0);
member("Scene").getPropRef("camera",1).addOverlay(t1, point(20, 20), 45);
```

Siehe auch

removeOverlay, overlay

addProp

Syntax

```
list.addProp(property, value)
addProp list, property, value
```

Beschreibung

Dieser Eigenschaftslistenbefehl ist nur auf Eigenschaftslisten anwendbar und fügt einer Eigenschaftsliste eine angegebene Eigenschaft mit einem dazugehörigen Wert hinzu.

In einer nicht geordneten Liste wird der Wert ans Ende der Liste gesetzt. In einer geordneten Liste wird der Wert an der vorgesehenen Stelle eingefügt.

Falls die Eigenschaft bereits in der Liste vorhanden ist, erstellen sowohl Lingo als auch JavaScript-Syntax ein Duplikat der Eigenschaft. Sie können Duplikate vermeiden, indem Sie mit dem Befehl `setaProp()` die Eigenschaft des neuen Eintrags ändern.

Wenn dieser Befehl mit einer linearen Liste verwendet wird, gibt er eine Fehlermeldung zurück.

Parameter

property Erforderlich. Die der Liste hinzuzufügende Eigenschaft.

value Erforderlich. Der Wert der der Liste hinzuzufügenden Eigenschaft.

Beispiel

Die folgende Anweisung fügt die Eigenschaft mit der Bezeichnung `Kayne` und ihren zugeordneten Wert 1 in die Eigenschaftsliste `bids` ein, die die folgenden Einträge enthält: [`#gee: 4, #ohasi:1`]. Da die Liste geordnet ist, wird der neue Eintrag in alphabetischer Reihenfolge eingefügt:

```
--Lingo
bids.addProp(#kayne, 3)

// Javascript
bids.addProp("kayne",3)
```

Das Ergebnis ist die Liste: [`#gee: 4, #kayne: 3, #ohasi: 1`].

Die Anweisung fügt den Eintrag `kayne: 7` zur Liste der benannten Gebote hinzu, die damit folgende Einträge enthält [`#gee: 4, #kayne: 3,#ohasi: 1`]. Da die Liste bereits die Eigenschaft `Kayne` enthält, erstellt Lingo ein Duplikat der Eigenschaft:

```
--Lingo
bids.addProp(#kayne, 7)

// Javascript
bids.addProp("kayne",7)
```

Das Ergebnis ist die Liste: [`#gee: 4, #kayne: 3, #kayne: 7, #ohasi: 1`].

addToWorld

Syntax

```
-- Lingo syntax
member(whichCastmember).model(whichModel).addToWorld()
member(whichCastmember).group(whichGroup).addToWorld()
member(whichCastmember).camera(whichCamera).addToWorld()
member(whichCastmember).light(whichLight).addToWorld()

// JavaScript syntax
member(whichCastmember).model(whichModel).addToWorld()
member(whichCastmember).group(whichGroup).addToWorld()
member(whichCastmember).camera(whichCamera).addToWorld()
memberwhichCastmember).light(whichLight).addToWorld()
```

Beschreibung

Dieser 3D-Befehl fügt das Modell, die Gruppe, die Kamera oder das Licht in die 3D-Welt des Darstellers als Child der Gruppe "World" ein.

Wenn ein Modell, eine Gruppe, eine Kamera oder ein Licht erstellt oder geklont wird, wird es bzw. sie automatisch zur Welt hinzugefügt. Mit dem Befehl `removeFromWorld` können Sie ein Modell, eine Gruppe, eine Kamera oder ein Licht aus der 3D-Welt entfernen, ohne es bzw. sie zu löschen. Mit dem Befehl `isInWorld()` können Sie testen, ob ein Modell, eine Gruppe, eine Kamera oder ein Licht in die Welt eingefügt bzw. aus ihr entfernt wurde.

Parameter

Keiner

Beispiel

Die folgende Anweisung fügt das Modell `gbCyl` zur 3D-Welt des Darstellers`Scene`.

```
-- Lingo syntax
member("Scene").model("gbCyl").addToWorld()

// JavaScript syntax
member("Scene").getProp("model","gbCyl").addToWorld();
```

Siehe auch

`isInWorld()`, `removeFromWorld`

addVertex()

Syntax

```
-- Lingo syntax
memberObjRef.addVertex(indexToAddAt, pointToAddVertex {,[ horizControlLocV, \ vertControlLocV
], [ horizControlLocH, vertControlLocV ]})

// JavaScript syntax
memberObjRef.addVertex(indexToAddAt, pointToAddVertex {,[ horizControlLocV, vertControlLocV
], [ horizControlLocH, vertControlLocV ]});
```

Beschreibung

Dieser Vektorformbefehl fügt einem Vektorformdarsteller an der angegebenen Position einen neuen Scheitelpunkt hinzu.

Die horizontale und vertikale Position ist relativ zur Anfangsposition des Vektorformdarstellers.

Mit den letzten zwei optionalen Parametern können Sie die Position der Anfasser für den Scheitelpunkt bestimmen. Die Anfasserposition ist relativ zum Scheitelpunkt, das heißt, wenn keine Position angegeben ist, befindet er sich horizontal und vertikal an der Position 0.

Parameter

indexToAddAt Erforderlich. Eine Ganzzahl, die die Indexposition angibt, an der der Darsteller hinzugefügt werden soll.

pointToAddVertex Erforderlich. Ein Punkt, der die Position angibt, an der der Darsteller hinzugefügt werden soll.

horizControlLocH Optional. Eine Ganzzahl, die die Position des horizontalen Teils der horizontalen Steuerziehpunkts angibt.

horizControlLocV Optional. Eine Ganzzahl, die die Position des vertikalen Teils der horizontalen Steuerziehpunkts angibt.

vertControlLocH Optional. Eine Ganzzahl, die die Position des horizontalen Teils der vertikalen Steuerziehpunkts angibt.

vertControlLocV Optional. Eine Ganzzahl, die die Position des vertikalen Teils der vertikalen Steuerziehpunkts angibt.

Beispiel

Die folgende Anweisung fügt zwischen zwei vorhandenen Scheitelpunkten in der Vektorform "Archie" einen Scheitelpunkt an der Position 25 horizontal und 15 vertikal hinzu:

```
-- Lingo syntax
member("Archie").addVertex(2, point(25, 15))

// JavaScript syntax
member("Archie").addVertex(2, point(25, 15));
```

Siehe auch

vertexList, moveVertex(), deleteVertex(), originMode

alert()

Syntax

```
-- Lingo syntax
_player.alert(displayString)

// JavaScript syntax
_player.alert(displayString);
```

Beschreibung

Diese Player-Methode löst einen Systemwarnton aus und zeigt ein Warndialogfeld mit einer angegebenen Zeichenfolge an.

Die Warnmeldung muss eine Zeichenfolge vom Typ „string“ sein. Wenn Sie eine Zahlenvariable in eine Warnmeldung einfügen möchten, müssen Sie die Variable erst in einen String konvertieren, bevor sie an `alert()` übergeben wird.

Parameter

displayString Erforderlich. Eine Zeichenfolge, die den im Warndialogfeld angezeigten Text darstellt. Die Zeichenfolge kann bis zu 255 Zeichen umfassen.

Beispiel

Die folgende Anweisung erstellt einen Warnhinweis, der Sie darauf aufmerksam macht, dass kein CD-ROM-Laufwerk angeschlossen ist:

```
-- Lingo syntax
_player.alert("There is no CD-ROM drive connected.")

// JavaScript syntax
_player.alert("There is no CD-ROM drive connected.");
```

Die folgende Anweisung erstellt einen Warnhinweis, der besagt, dass eine Datei nicht gefunden wurde:

```
-- Lingo syntax
_player.alert("The file" && QUOTE & filename & QUOTE && "was not found.")

// JavaScript syntax
_player.alert("The file \"" + filename + "\" was not found.");
```

Siehe auch

Player

Alert()

Syntax

```
Alert( MUIObject,alertPropertiesList)
```

Beschreibung

Dieser Befehl zeigt ein Warndialogfeld an, das aus einer Instanz des Xtras MUI erzeugt wurde. Diese Funktion ist eine Ergänzung der einfachen Warnhinweise, die mithilfe des Befehls „alert“ generiert werden.

Das Xtra MUI stellt modale Warnhinweise bereit. Der Warnhinweis kann verschiebbar bzw. nicht verschiebbar sein. Um den Warnhinweis zu erzeugen, erstellen Sie ein Objekt des Xtras MUI, und rufen Sie anschließend den Befehl „Alert“ mit einer Liste, die Definitionen von Warnhinweiseigenschaften enthält, als zweiten Parameter auf.

Es folgen Eigenschaften, die Sie angeben müssen, und ihre möglichen Werte:

Eigenschaft	Mögliche Werte	Angabe
`#buttons`	`#Ok#OkCancel#AbortRetryIgnore#YesNoCancel#YesNo#RetryCancel`	Die Reihe der Schaltflächen, die im Warnhinweis angezeigt werden. Die Schaltflächen werden in der Reihenfolge angezeigt, in der sie in jedem Symbol benannt sind.
`#default`	Die Ordnungszahl der Schaltfläche, die zur Standardeinstellung wird. Wenn beispielsweise die Schaltflächen im Warnhinweis „OK“ und „Abbrechen“ heißen, gibt „2“ die Schaltfläche „Abbrechen“ an. Ist keine Standardeinstellung gewünscht, geben Sie „0“ an.	Die ausgewählte Standardschaltfläche
`#icon`	`#stop #note #caution #question #error`	Der Symboltyp, der im Warnhinweis angezeigt wird. Ist kein Symbol gewünscht, geben Sie „0“ an.
`#message`	Zeichenfolge	Die Meldung, die im Warnhinweis angezeigt wird
`#movable`	TRUE oder FALSE	Gibt an, ob der Warnhinweis verschiebbar ist
`#title`	Zeichenfolge	Der Titel der Warnmeldung

Sie müssen die Eigenschaften des Warnhinweises explizit angeben. Das Xtra MUI stellt keine Liste mit standardmäßigen Warnhinweiseigenschaften bereit. Lingo gibt einen Wert für die Schaltfläche zurück, auf die der Benutzer klickt.

Warnhinweise können nahezu den gesamten Bildschirm einnehmen. Falls gewünscht, können Sie eine ausführliche Beschreibung anzeigen.

Die folgenden Anweisungen dienen zum Erstellen und Anzeigen eines Warndialogfelds.

- Die erste Anweisung erzeugt eine Instanz des Xtras MUI, bei der es sich um das Objekt handelt, das als Dialogfeld verwendet wird.
- Die zweite Anweisung richtet eine Liste der Eigenschaften des Warnhinweises ein.
- Die letzten Anweisungen verwenden den Befehl „Alert“, um den Warnhinweis anzuzeigen und zu melden, auf welche Schaltflächen der Benutzer klickt.

Beispiel

```
-- Lingo syntax
set alertObj = new(xtra "MUI")
set alertInitList = [ #buttons : #YesNo, ¬
#title : "Alert Test", #message : "This shows Yes and No buttons",\
#movable : TRUE]
if objectP ( alertObj ) then
        set result = Alert( alertObj, alertInitList )
        case result of
        1 : -- the user clicked yes
        2 : -- the user clicked no
        otherwise : -- something is seriously wrong
    end case
end if
```

append

Syntax

```
list.append(value)
append list, value
```

Beschreibung

Dieser Listenbefehl ist nur auf lineare Listen anwendbar und fügt den angegebenen Wert am Ende der linearen Liste an. Damit unterscheidet er sich vom Befehl `add`, der einer geordneten Liste an der vorgesehenen Stelle einen Wert hinzufügt.

Wenn dieser Befehl mit einer Eigenschaftsliste verwendet wird, gibt er einen Skriptfehler zurück.

Eigenschaften

value Erforderlich. Der am Ende der linearen Liste hinzuzufügende Wert.

Beispiel

Die folgende Anweisung setzt den Wert 2 ans Ende der geordneten Liste "bids", die [1, 3, 4] enthält, obwohl dies nicht der Sortierreihenfolge der Liste entspricht:

```
--Lingo
set bids = [1, 3, 4]
bids.append(2)

// Javascript
bids = list(1, 3, 4)
bids.append(2)
```

Der resultierende Wert für bids ist [1, 3, 4, 2].

Siehe auch

`add (3D-Textur)`, `sort`

applyFilter()

Syntax

```
-- Lingo syntax
imageObjRef.applyFilter(filterObjRef {, rectObj})

// JavaScript syntax
imageObjRef.applyFilter(filterObjRef {, rectObj});
```

Beschreibung

Diese Bild-Methode wendet den angegebenen Filter auf ein Bildobjekt an. Verwenden Sie den Parameter `rectObj` (optional), um den Filter auf einen Bildausschnitt anzuwenden.

`applyFilter()` ändert das Bildobjekt direkt. Die Methode verändert allerdings nicht die Bildgröße. Filtereffekt, die über die Grenzen von `rectObj` hinausgehen, werden ignoriert.

Wichtig: *Die über `applyFilter()` angewendeten Filtereffekte lassen sich nicht umkehren.*

Parameter

Parameter	Beschreibung	Erforderlich/Optional
filterObjRef	Filter, der auf das angegebene Bildobjekt angewendet werden soll. Verwenden Sie die Methode `filter()` zum Erstellen des `filterObjRef`.	Erforderlich
rectObj	Die rechteckige Fläche innerhalb des Bildes, auf die der Filter angewendet wird. Die Koordinaten des Rechtecks sind relativ zu den Bildkoordinaten. Falls kein Zielrechteck festgelegt wird, bezieht sich der Filter auf das gesamte Bild.	Optional

Beispiele

Das folgende Beispiel wendet `blurFilter` auf einen Ausschnitt von `Earth` (einem Bitmapdarsteller) an.

```
-- Lingo
member("Earth").image.applyFilter(filter(#blurfilter), rect(50, 50, 100, 100))

// Javascript
member("Earth").image.applyFilter(filter(#blurfilter), rect(50, 50, 100, 100));
```

Siehe auch

`rect (Grafik)`, `image (Grafik)`

appMinimize()

Syntax

```
-- Lingo syntax
_player.appMinimize()

// JavaScript syntax
_player.appMinimize();
```

Beschreibung

Unter Windows verkleinert diese Player-Methode einen Projektor zu einem Symbol in der Windows-Taskleiste. Auf dem Mac wird ein Projektor dadurch ausgeblendet.

Ein auf dem Mac ausgeblendeter Projektor kann über das Mac-Anwendungsmenü wieder eingeblendet werden.

Diese Methode ist nützlich bei Projektoren und MIAWs, die ohne Titelleiste angezeigt werden.

Parameter

Keiner

Beispiel

```
--Lingo syntax
on mouseUp me
    _player.appMinimize()
end


// JavaScript syntax
function mouseUp() {
     _player.appMinimize();
}
```

Siehe auch

Player

atan()

Syntax

```
-- Lingo syntax
(number).atan
atan (number)

// JavaScript syntax
Math.atan(number);
```

Beschreibung

Diese mathematische Funktion (nur Lingo) berechnet den Arkustangens, d. h. den Winkel, dessen Tangens die angegebene Zahl ist. Das Ergebnis ist ein Bogenmaßwert zwischen -pi/2 und +pi/2.

In JavaScript-Syntax wird die `atan()`-Funktion des Math-Objekts verwendet.

Parameter

Keiner

Beispiel

Die folgende Anweisung zeigt den Arkustangens von 1 an:

```
(1).atan
```

Das auf vier Dezimalstellen gerundete Ergebnis ist 0,7854, was ungefähr pi/4 entspricht.

Die meisten trigonometrischen Funktionen verwenden das Bogenmaß. Aus diesem Grunde ist zu empfehlen, Gradwerte ins Bogenmaß zu konvertieren.

Mit der folgenden Prozedur können Sie Grade ins Bogenmaß umwandeln:

```
-- Lingo syntax
on DegreesToRads degreeValue
    return degreeValue * PI/180
end

// JavaScript syntax
function DegreesToRads(degreeValue) {
    return degreeValue * PI/180
}
```

Die Prozedur zeigt die Umwandlung von 30° ins Bogenmaß im Nachrichtenfenster an:

```
put DegreesToRads(30)
-- 0.5236
```

Siehe auch

cos(), PI, sin()

beep()

Syntax

```
-- Lingo syntax
_sound.beep({intBeepCount})

// JavaScript syntax
_sound.beep({intBeepCount});
```

Beschreibung

Diese Sound-Methode bewirkt, dass der Lautsprecher des Computers die durch *intBeepCount* festgelegte Anzahl von Warntönen ausgibt. Wenn *intBeepCount* fehlt, ertönt der Warnton einmal.

- In Windows ist der Warnton der im Dialogfeld Soundeigenschaften zugeordnete Sound.
- Auf dem Macintosh ist der Warnton der im Kontrollfeld „Ton“ ausgewählte Warnton. Wenn die Lautstärke im Kontrollfeld Ton auf 0 eingestellt ist, blinkt stattdessen die Menüleiste.

Parameter

intBeepCount Optional. Eine Ganzzahl, die die Anzahl der über die Computerlautsprecher wiederzugebenden Warntöne angibt.

Beispiel

```
-- Lingo syntax
on mouseUp me
    _sound.beep(1)
end mouseUp
// JavaScript syntax
function mouseUp() {
    _sound.beep(1);
}
```

Siehe auch

Sound

beginRecording()

Syntax

```
-- Lingo syntax
_movie.beginRecording()

// JavaScript syntax
_movie.beginRecording();
```

Beschreibung

Diese Filmmethode beginnt eine Erstellungssitzung für ein Drehbuch.

Wenn Sie `beginRecording()` aufrufen, wird der Abspielkopf automatisch um ein Bild vorgeschoben. In diesem Bild beginnt dann die Aufzeichnung. Zur Vermeidung dieses Verhaltens beginnen Sie die Aufzeichnung in dem Bild, aus dem `beginRecording()` aufgerufen wird, und fügen eine Anweisung wie `_movie.go(_movie.frame - 1)` zwischen den Aufrufen von `beginRecording()` und `endRecording()` ein.

Es kann jeweils immer nur eine Aktualisierungssitzung in einem Film aktiv sein.

Jeder Aufruf von `beginRecording()` muss mit einem Aufruf von `endRecording()` gepaart werden, der die Drehbucherstellungssitzung beendet.

Parameter

Keiner

Beispiel

Wenn das Schlüsselwort `beginRecording` in der folgenden Prozedur verwendet wird, beginnt es eine `Score`-Erstellungssitzung, die den Darsteller `Ball` animiert. Dabei wird der Darsteller dem Sprite-Kanal 20 zugeordnet, und das Sprite wird über eine Reihe von Bildern hinweg horizontal und vertikal bewegt. Die Anzahl der Bilder wird durch das Argument `numberOfFrames` festgelegt.

```
-- Lingo syntax
on animBall(numberOfFrames)
    _movie.beginRecording()
    horizontal = 0
    vertical = 100
    repeat with i = 1 to numberOfFrames
        _movie.go(i)
        sprite(20).member = member("Ball")
        sprite(20).locH = horizontal
        sprite(20).locV = vertical
        sprite(20).foreColor = 255
        horizontal = horizontal + 3
        vertical = vertical + 2
        _movie.updateFrame()
    end repeat
    _movie.endRecording()
end animBall

// JavaScript syntax
function animBall(numberOfFrames) {
    _movie.beginRecording();
    var horizontal = 0;
    var vertical = 100;
    for (var i = 1; i <= numberOfFrames; i++) {
        _movie.go(1);
        sprite(20).member = member("Ball")
        sprite(20).locH = horizontal;
        sprite(20).locV = vertical;
        sprite(20).foreColor = 255;
        horizontal = horizontal + 3;
        vertical = vertical + 2;
        _movie.updateFrame();
    }
    _movie.endRecording();
}
```

Siehe auch

endRecording(), Movie, updateFrame()

bitAnd()

Syntax

```
bitAnd(integer1, integer2)
```

Beschreibung

Diese Funktion (nur Lingo) wandelt die beiden angegebenen Ganzzahlen in 32-Bit-Binärzahlen um und gibt eine Binärzahl mit Einsen an den Stellen, an denen bei beiden Zahlen eine 1 stand, und Nullen an allen anderen Stellen zurück. Das Ergebnis ist eine neue Binärzahl, die von Lingo als dezimale Ganzzahl dargestellt wird.

Ganzzahl	Binärzahl (gekürzt)
6	00110
7	00111
Ergebnis	
6	00110

In JavaScript-Syntax verwenden Sie den bitweisen Operator "&".

Parameter

integer1 Erforderlich. Die erste Ganzzahl.

integer2 Erforderlich. Die zweite Ganzzahl.

Beispiel

Die folgende Anweisung vergleicht die Binärwerte der Ganzzahlen 6 und 7 und gibt das Ergebnis als Ganzzahl zurück:

```
--Lingo
put bitAnd(6, 7)
-- 6



// Javascript
trace ( 6 & 7)
// 6
```

Siehe auch

`bitNot()`, `bitOr()`, `bitXor()`

bitNot()

Syntax

```
(integer).bitNot
bitNot(integer)
```

Beschreibung

Diese Funktion (nur Lingo) wandelt die angegebene Ganzzahl in eine 32-Bit-Binärzahl um und kehrt den Wert der einzelnen Binärziffern um, indem sie alle Einsen durch Nullen und alle Nullen durch Einsen ersetzt. Das Ergebnis ist eine neue Binärzahl, die von Lingo als dezimale Ganzzahl dargestellt wird.

Ganzzahl	Binärzahl
1	00000000000000000000000000000001
Ergebnis	
-2	11111111111111111111111111111110

In JavaScript-Syntax verwenden Sie den bitweisen Operator "~".

Parameter
Keiner

Beispiel
Die folgende Anweisung kehrt den Binärwert der Ganzzahl 1 um und gibt eine neue Zahl zurück.

```
--Lingo
put (1).bitNot
-- -2

// Javascript
trace(~1)
// -2
```

Siehe auch
`bitAnd()`, `bitOr()`, `bitXor()`

bitOr()

Syntax
```
bitOr(integer1, integer2)
```

Beschreibung
Diese Funktion (nur Lingo) wandelt die beiden angegebenen Ganzzahlen in 32-Bit-Binärzahlen um und gibt eine Binärzahl mit Einsen an den Stellen, an denen bei beiden Zahlen eine 1 stand, und Nullen an allen anderen Stellen zurück. Das Ergebnis ist eine neue Binärzahl, die von Lingo als dezimale Ganzzahl dargestellt wird.

Ganzzahl	**Binärzahl (gekürzt)**
5	0101
6	0110
Ergebnis	
7	0111

In JavaScript-Syntax verwenden Sie den bitweisen Operator "|".

Parameter
integer1 Erforderlich. Die erste Ganzzahl.

integer2 Erforderlich. Die zweite Ganzzahl.

Beispiel
Die folgende Anweisung vergleicht den 32-Bit-Binärwert der Ganzzahlen 5 und 6 und gibt das Ergebnis als Ganzzahl zurück:

```
-- Lingo
put bitOr(5, 6)
-- 7

// Javascript
trace(5|6)
// 7
```

Siehe auch

`bitNot()`, `bitAnd()`, `bitXor()`

bitXor()

Syntax

```
bitXor(integer1, integer2)
```

Beschreibung

Diese Funktion wandelt die beiden angegebenen Ganzzahlen in 32-Bit-Binärzahlen um und gibt eine Binärzahl mit Einsen an den Stellen, an denen die Ziffern der beiden Nummern nicht übereinstimmen, und Nullen an den Stellen, an denen die Ziffern identisch sind, zurück. Das Ergebnis ist eine neue Binärzahl, die von Lingo als dezimale Ganzzahl dargestellt wird.

Ganzzahl	**Binärzahl (gekürzt)**
5	0101
6	0110
Ergebnis	
3	0011

In JavaScript-Syntax verwenden Sie den bitweisen Operator "^".

Parameter

integer1 Erforderlich. Die erste Ganzzahl.

integer2 Erforderlich. Die zweite Ganzzahl.

Beispiel

Die folgende Anweisung vergleicht den 32-Bit-Binärwert der Ganzzahlen 5 und 6 und gibt das Ergebnis als Ganzzahl zurück:

```
-- Lingo
put bitXor(5, 6)
-- 3

// Javascript
trace(5^6)
// 3
```

Siehe auch

bitNot(), bitOr(), bitAnd()

breakLoop()

Syntax

```
-- Lingo syntax
soundChannelObjRef.breakLoop()

// JavaScript syntax
soundChannelObjRef.breakLoop();
```

Beschreibung

Diese Soundkanalmethode bewirkt, dass der aktuell in Schleife abgespielte Sound auf Kanal *soundChannelObjRef* die Schleife beendet und bis zu seiner `endTime` wiedergegeben wird.

Außerhalb einer Schleife hat diese Methode keine Wirkung.

Parameter

Keiner

Beispiel

Die folgende Prozedur bewirkt, dass die Hintergrundmusikschleife auf Soundkanal 2 abgebrochen und der Sound bis zum Ende wiedergegeben wird:

```
-- Lingo syntax
on continueBackgroundMusic
    sound(2).breakLoop()
end

// JavaScript syntax
function continueBackgroundMusic() {
    sound(2).breakLoop();
}
```

Siehe auch

endTime (Soundkanal), Sound Channel

breakLoop (Soundobjekt)

Syntax

```
soundObject.breakLoop()
```

Beschreibung

Diese Soundobjektmethode stoppt die Wiedergabeschleife und spielt das Soundobjekt bis `endTime` ab.

Beispiele

```
--Lingo syntax
on mouseUp me
     soundObjRef.breakloop() -- Stops the looping of the currently looping sound object.
end

// JavaScript syntax
function mouseUp(){
soundObjRef.breakLoop(); // Stops the looping of the currently looping sound object.
}
```

Siehe auch

`endTime (Soundobjekt)`

browserName()

Syntax

```
browserName pathName
browserName()
browserName(#enabled, trueOrFalse)
```

Beschreibung

Dies ist eine Systemeigenschaft, ein Befehl und eine Funktion, die den Pfad oder die Position des Browsers angibt. Mit dem `FileIO Xtra` können Sie ein Dialogfeld anzeigen, mit dem der Benutzer nach einem Browser suchen kann. Die Methode `displayOpen()` des `FileIO Xtra` kann zum Anzeigen des Dialogfelds „Öffnen“ verwendet werden.

Die Form `browserName()` gibt den Namen des aktuellen Browsers zurück. Wenn Sie einen Pfadnamen wie den mit dem `FileIO Xtra` gefundenen als Argument in die Form `browserName(fullPathToApplication)` setzen, kann die Eigenschaft eingestellt werden. Die Form `browserName(#enabled,` *trueOrFalse*`)` legt fest, ob der angegebene Browser automatisch gestartet wird, wenn der Befehl `goToNetPage` erteilt wird.

Dieser Befehl ist nur für das Abspielen in einem Projektor oder in Director nützlich und hat keine Wirkung, wenn der Film in einem Browser abgespielt wird.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Die folgende Anweisung verweist auf die Position des Firefox®-Browsers:

```
browserName "My Disk:My Folder:FireFox"
```

Die folgende Anweisung zeigt den Namen des Browsers in einem Nachrichtenfenster an:

```
put browserName()
```

ByteArray

Syntax

```
ByteArray([intSize],[intInitialValue])
```

Beschreibung
Diese Bytearray-Methode erstellt ein Bytearray der Größe `intSize`. Diese Methode wird mit dem Wert `intInitialValue` initialisiert.

Parameter

Parameter	Beschreibung	Standardwert
`intSize`	Größe des Byte-Arrays.	0
`intInitialValue`	Das Byte-Array wird mit diesem Wert initialisiert, der in allen Bytes gespeichert wird.	0

Beispiele
```
--Lingo syntax
bArray=byteArray(10,1)

//JavaScript syntax
bArray=byteArray(10,1);
```

ByteArray(str)

Syntax
```
ByteArray(str)
```

Beschreibung
Diese Bytearray-Methode erstellt aus dem übergebenen String ein Bytearray.

Parameter

Parameter	Beschreibung	Standardwert
str	Inhalt des Byte-Arrays	Void

Beispiele
```
--Lingo syntax
bArray=byteArray("Director")

//JavaScript syntax
bArray=byteArray("Director");
```

build()

Syntax
```
-- Lingo syntax
member(whichCastmember).modelResource(whichModelResource).build()

// JavaScript syntax
member(whichCastmember).modelResource(whichModelResource).build();
```

Beschreibung

Dieser 3D-Gitternetzbefehl dient zum Erstellen eines Gitternetzes. Dieser Befehl wird nur bei Modellressourcen vom Typ `#mesh` verwendet.

Sie müssen den Befehl `build()` zu Beginn der Gitternetzerstellung, nach Änderung einer `face`-Eigenschaft des Gitternetzes und nach Aufruf des Befehls `generateNormals()` verwenden.

Parameter

Keiner

Beispiel

In diesem Beispiel werden eine einfache Modellressource vom Typ `#mesh` erstellt, seine Eigenschaften festgelegt und dann ein neues Modell anhand der Modellressource erstellt. Die genaue Verfahrensweise wird in der folgenden Erläuterung des Beispielcodes beschrieben:

In Zeile 1 wird ein Gitternetz namens "Ebene" erstellt, das eine Fläche, drei Scheitelpunkte und bis zu drei Farben aufweist. Die Anzahl von Normalen und die Anzahl von Texturkoordinaten sind nicht festgelegt. Die Normalen werden mit dem Befehl `generateNormals` erstellt.

In Zeile 2 werden die Vektoren definiert, die als Scheitelpunkte für das Gitternetz "Ebene" verwendet werden sollen.

In Zeile 3 werden die Vektoren den Scheitelpunkten der ersten Fläche des Gitternetzes "Ebene" zugeordnet.

In Zeile 4 werden die drei Farben definiert, die für den Befehl `newMesh` gelten sollen.

In Zeile 5 werden die Farben der ersten Fläche des Gitternetzes "Ebene" zugeordnet. Die dritte Farbe in der Farbliste wird auf den ersten Scheitelpunkt von "Ebene" angewendet, die zweite Farbe auf den zweiten Scheitelpunkt und die erste Farbe auf den dritten Scheitelpunkt. Die Farben werden in Form von Verläufen über die erste Fläche von "Ebene" verteilt.

In Zeile 6 werden mit dem Befehl `generateNormals()` die Normalenvektoren für „Ebene“ erstellt.

In Zeile 7 wird der Befehl build() zur Erstellung des Gitternetzes aufgerufen.

```
-- Lingo syntax
nm = member("Shapes").newMesh("Plane",1,3,0,3,0)
nm.vertexList = [vector(0,0,0), vector(20,0,0), vector(20, 20, 0)]
nm.face[1].vertices = [1,2,3]
nm.colorList = [rgb(255,255,0), rgb(0, 255, 0), rgb(0,0,255)]
nm.face[1].colors = [3,2,1]
nm.generateNormals(#smooth)
nm.build()
nm = member("Shapes").newModel("TriModel", nm)

// JavaScript syntax
nm = member("Shapes").newMesh("Plane",1,3,0,3,0);
nm.vertexList = [vector(0,0,0), vector(20,0,0), vector(20, 20, 0)];
nm.face[1].vertices = [1,2,3];
nm.colorList = [rgb(255,255,0), rgb(0, 255, 0), rgb(0,0,255)];
nm.face[1].colors = [3,2,1];
nm.generateNormals(#smooth);
nm.build();
nm = member("Shapes").newModel("TriModel", nm);
```

Siehe auch

`generateNormals()`, `newMesh`, `face[ ]`

cacheDocVerify()

Syntax

```
-- Lingo syntax
cacheDocVerify #setting
cacheDocVerify()

// JavaScript syntax
cacheDocVerify symbol(setting);
cacheDocVerify();
```

Beschreibung

Diese Funktion bestimmt, wie oft der Inhalt einer Seite im Internet mit Informationen aus dem Projektor-Cache aktualisiert wird.

Die Form `cacheDocVerify()` gibt die aktuelle Einstellung des Cache-Speichers zurück.

Die Funktion `cacheDocVerify` ist nur für Filme gültig, die in Director oder als Projektoren abgespielt werden. Sie kann nicht für Filme mit Adobe® Shockwave®-Inhalten eingesetzt werden, da diese die Netzwerkeinstellungen des Browsers verwenden, in dem sie abgespielt werden.

```
-- Lingo syntax
on resetCache
    current = cacheDocVerify()
    if current = #once then
        alert "Turning cache verification on"
        cacheDocVerify #always
    end if
end

// JavaScript syntax
function resetCache() {
    current = cacheDocVerify();
    if (current == symbol("once")) {
        alert("Turning cache verification on");
        cacheDocVerify(symbol("always"))
    }
}
```

Parameter

cacheSetting Optional. Ein Symbol, das angibt, wie oft der Inhalt einer Seite im Internet aktualisiert wird. Mögliche Werte sind `#once` (Standard) und `#always`. Der Wert `#once` veranlasst einen Film, eine Datei einmal vom Internet zu laden und sie dann bei Bedarf aus dem Cache-Speicher abzurufen, ohne nach einer aktualisierten Version im Internet zu suchen. Der Wert `#always` veranlasst den Film, bei jedem Aufruf einer URL-Adresse nach einer aktualisierten Version der Datei zu suchen.

Siehe auch

`cacheSize()`, `clearCache`

cacheSize()

Syntax

```
-- Lingo syntax
cacheSize Size
cacheSize()

// JavaScript syntax
cacheSize(Size);
cacheSize();
```

Beschreibung

Diese Funktion und dieser Befehl stellen die Größe des Cache-Speichers von Director ein.

Die Funktion `cacheSize` ist nur für Filme gültig, die in Director oder als Projektoren abgespielt werden. Sie kann nicht für Filme mit Shockwave-Inhalten eingesetzt werden, da diese die Netzwerkeinstellungen des Browsers verwenden, in dem sie abgespielt werden.

Parameter

newCacheSize Optional. Eine Ganzzahl, die die Cachegröße in KB angibt.

Beispiel

Die folgende Prozedur prüft, ob die Cache-Einstellung des Browsers geringer als 1 MB ist. Wenn dem so ist, zeigt die Prozedur einen Warnhinweis an und stellt die Cache-Größe auf 1 MB ein:

```
-- Lingo syntax
on checkCache if
    cacheSize()<1000 then
        alert "increasing cache to 1MB"
        cacheSize 1000
    end if
end

// JavaScript syntax
function checkCache() {
    if (cacheSize() < 1000) {
        alert("increasing cache to 1MB");
        cacheSize(1000);
    }
}
```

Siehe auch

`cacheDocVerify()`, `clearCache`

call

Syntax

```
call #handlerName, script, {args...}
call (#handlerName, scriptInstance, {args...})
```

Beschreibung

Dieser Befehl sendet eine Nachricht, die eine Prozedur in einem angegebenen Skript oder einer Liste mit Skripts aufruft.

Der Befehl `call` kann eine Variable als Namen der Prozedur verwenden. Nachrichten, die unter Verwendung von `call` übergeben werden, werden weder an andere am Sprite angebrachte Skripts noch an Darstellerskripten, Bildskripten oder Filmskripten übertragen.

Parameter

symHandlerName Erforderlich. Ein Symbol, das die zu aktivierende Prozedur angibt.

scriptInstance Erforderlich. Ein Verweis auf das Skript oder die Liste mit Skripts, das bzw. die die Prozedur enthält. Wenn *script* eine einzelne Skriptinstanz ist und die Prozedur nicht im Ancestor-Skript des Skripts definiert wurde, erscheint ein Warnhinweis. Wenn *script* eine Liste von Skriptinstanzen ist, wird die Nachricht nacheinander an jedes Element in der Liste gesendet. Wenn die Prozedur nicht im Ancestor-Skript definiert ist, erfolgt kein Warnhinweis.

args Optional. Alle an die Prozedur zu übergebenden optionalen Parameter.

Beispiel

Die folgende Prozedur schickt die Nachricht `bumpCounter` an das erste Verhalten, das an Sprite 1 angebracht ist:

```
-- Lingo syntax
on mouseDown me
    -- get the reference to the first behavior of sprite 1
    set xref = getAt (the scriptInstanceList of sprite 1,1)
    -- run the bumpCounter handler in the referenced script,
    -- with a parameter
    call (#bumpCounter, xref, 2)
end

// JavaScript syntax
function mouseDown() {
    // get the reference to the first behavior of sprite 1
    xref = getAt(sprite(1).script(1));
    // run the bumpCounter handler in the referenced script
    call(symbol("bumpcounter"), xref, 2);
}
```

Das folgende Beispiel zeigt, wie eine call-Anweisung Prozeduren in einem Verhalten oder einem Parent-Skript und seinem Ancestor-Skript aufruft.

- Dies ist das Parent-Skript:

```
-- Lingo syntax
-- script Man
property ancestor

on new me
    set ancestor = new(script "Animal", 2)
    return me
end
on run me, newTool
    put "Man running with "&the legCount of me&" legs"
end
```

- Dies ist das Ancestor-Skript:

```
-- script Animal
property legCount

on new me, newLegCount
    set legCount = newLegCount
    return me
end
on run me
    put "Animal running with "& legCount &" legs"
end
on walk me
    put "Animal walking with "& legCount &" legs"
end
```

- Die folgenden Anweisungen verwenden das Parent- und das Ancestor-Skript.

  Die folgende Anweisung erstellt eine Instanz des Parent-Skripts:

  ```
  set m = new(script "man")
  ```

  Die folgende Anweisung lässt den Menschen gehen:

  ```
  call #walk, m
  -- "Animal walking with 2 legs"
  ```

  Die folgende Anweisung lässt den Menschen laufen:

  ```
  set msg = #run
  call msg, m
  -- "Man running with 2 legs and rock"
  ```

  Die folgende Anweisung erstellt die zweite Instanz des Parent-Skripts:

  ```
  set m2 = new(script "man")
  ```

  Die folgende Anweisung schickt eine Nachricht an beide Instanzen des Parent-Skripts:

  ```
  call msg, [m, m2]
  -- "Man running with 2 legs "
  -- "Man running with 2 legs "
  ```

callAncestor

Syntax

```
callAncestor handlerName, script, {args...}
```

Beschreibung

Dieser Befehl sendet eine Nachricht an das Ancestor-Skript eines Child-Objekts.

Ancestor-Skripts können selbst Ancestor-Skripts haben.

Wenn Sie `callAncestor` verwenden, kann der Name der Prozedur eine Variable sein, und Sie können die Prozeduren im ersten Skript ausdrücklich übergehen und direkt zum Ancestor-Skript gehen.

Parameter

symHandlerName Erforderlich. Ein Symbol, das die zu aktivierende Prozedur angibt.

scriptInstance Erforderlich. Ein Verweis auf das Skript oder die Liste mit Skripts, das bzw. die die Prozedur enthält. Wenn *script* eine einzelne Skriptinstanz ist und die Prozedur nicht im Ancestor-Skript des Skripts definiert wurde, erscheint ein Warnhinweis. Wenn *script* eine Liste von Skriptinstanzen ist, wird die Nachricht nacheinander an jedes Element in der Liste gesendet. Wenn die Prozedur nicht im Ancestor-Skript definiert ist, erfolgt kein Warnhinweis.

args Optional. Alle an die Prozedur zu übergebenden optionalen Parameter.

Beispiel

Dieses Beispiel zeigt, wie eine `callAncestor`-Anweisung Prozeduren im Ancestor-Skript eines Verhaltens oder eines Parent-Skripts aufrufen kann.

- Dies ist das Parent-Skript:

```
-- script "man"
property ancestor

on new me, newTool
    set ancestor = new(script "Animal", 2)
    return me
end
on run me
    put "Man running with "&the legCount of me&"legs"
end
```

- Dies ist das Ancestor-Skript:

```
-- script "animal"
property legCount

on new me, newLegCount
    set legCount = newLegCount
    return me
end
on run me
    put "Animal running with "& legCount &" legs"
end
on walk me
    put "Animal walking with "& legCount &" legs"
end
```

- Die folgenden Anweisungen verwenden das Parent- und das Ancestor-Skript.

  Die folgende Anweisung erstellt eine Instanz des Parent-Skripts:

```
set m = new(script "man")
```

  Die folgende Anweisung lässt den Menschen gehen:

```
call #walk, m
-- "Animal walking with 2 legs"
```

  Die folgende Anweisung lässt den Menschen laufen:

```
set msg = #run
callAncestor msg, m
-- "Animal running with 2 legs"
```

  Die folgende Anweisung erstellt die zweite Instanz des Parent-Skripts:

```
set m2 = new(script "man")
```

Die folgende Anweisung sendet eine Nachricht an das Ancestor-Skript für beide Menschen:

```
callAncestor #run,[m,m2]
-- "Animal running with 2 legs"
-- "Animal running with 2 legs"
```

Siehe auch

ancestor, new()

callFrame()

Syntax

```
-- Lingo syntax
spriteObjRef.callFrame(flashFrameNameOrNum)

// JavaScript syntax
spriteObjRef.callFrame(flashFrameNameOrNum);
```

Beschreibung

Mit diesem Befehl wird eine Reihe von Aktionen aufgerufen, die sich in einem Bild eines Flash®-Film-Sprites befinden.

Dieser Befehl sendet eine Nachricht an die ActionScript-Engine von Flash® und löst die Aktionen aus, die im Flash-Film auszuführen sind.

Parameter

flashFrameNameOrNum Erforderlich. Eine Zeichenfolge oder Zahl, die den Namen oder die Nummer des aufzurufenden Bildes angibt.

Beispiel

Die folgende Lingo-Anweisung führt die Aktionen aus, die an Bild 10 des Flash-Films in Sprite 1 angebracht sind:

```
-- Lingo syntax
sprite(1).callFrame(10)

// JavaScript syntax
sprite(1).callFrame(10);
```

camera()

Syntax

```
member(whichCastMember).camera(whichCamera)
member(whichCastMember).camera[index]
member(whichCastMember).camera(whichCamera).whichCameraProperty
member(whichCastMember).camera[index].whichCameraProperty
sprite(whichSprite).camera{(index)}
sprite(whichSprite).camera{(index)}.whichCameraProperty
```

Beschreibung

Dieses 3D-Element ist ein Objekt an einer Vektorposition, von der aus die 3D-Welt betrachtet wird.

Jedes Sprite verfügt über eine Liste mit Kameras. Die Ansicht von den einzelnen Kameras aus erscheint über der Ansicht von Kameras mit niedrigeren *index*-Positionen. Sie können für jede Kamera die Eigenschaft `rect (camera)` einstellen, um mehrere Ansichten im Sprite darzustellen.

Kameras werden in der Kamerapalette des Darstellers gespeichert. Mit den Befehlen `newCamera` und `deleteCamera` können Sie Kameras in einem 3D-Darsteller erstellen und löschen.

Bei einem Sprite bezieht sich die Eigenschaft `camera` auf die erste Kamera in der Kameraliste des Sprites. Mit `sprite(`*whichSprite*`).camera` wird auf dieselbe Kamera Bezug genommen wie mit `sprite(`*whichSprite*`).camera(1)`. Mit den Befehlen `addCamera` und `deleteCamera` können Sie bei einem 3D-Sprite die Kameraliste erstellen.

Beispiel

Die folgende Anweisung setzt die Kamera von Sprite 1 auf die Kamera `TreeCam` des Darstellers `Picnic`:

```
sprite(1).camera = member("Picnic").camera("TreeCam")
```

Die folgende Anweisung setzt die Kamera von Sprite 1 auf Kamera 2 des Darstellers „Picknick“:

```
sprite(1).camera = member("Picnic").camera[2]
```

Siehe auch

`bevelDepth`, `overlay`, `modelUnderLoc`, `spriteSpaceToWorldSpace`, `fog`, `clearAtRender`

cameraCount()

Syntax

```
-- Lingo syntax
sprite(whichSprite).cameraCount()

// JavaScript syntax
sprite(whichSprite).cameraCount();
```

Beschreibung

Dieser 3D-Befehl gibt die Anzahl von Einträgen in der Kameraliste des Sprites zurück.

Parameter

Keiner

Beispiel

Die folgende Anweisung zeigt, dass Sprite 5 drei Kameras enthält:

```
-- Lingo syntax
put sprite(5).cameraCount()
-- 3

// JavaScript syntax
put(sprite(5).cameraCount());
// 3
```

Siehe auch

addCamera, deleteCamera

cancelIdleLoad()

Syntax

```
-- Lingo syntax
_movie.cancelIdleLoad(intLoadTag)

// JavaScript syntax
_movie.cancelIdleLoad(intLoadTag);
```

Beschreibung

Diese Filmmethode bricht den Ladevorgang aller Darsteller ab, die das angegebene Ladekennzeichen haben.

Parameter

intLoadTag Erforderlich. Eine Ganzzahl, die eine Gruppe von Darstellern angibt, die im Ruhezustand des Computers auf das Vorausladen warten.

Beispiel

Die folgende Anweisung bricht den Ladevorgang der Darsteller ab, die das Wartezustand-Ladekennzeichen20haben:

```
-- Lingo syntax
_movie.cancelIdleLoad(20)

// JavaScript syntax
_movie.cancelIdleLoad(20);
```

Siehe auch

idleLoadTag, Movie

castLib()

Syntax

```
-- Lingo syntax
castLib(castNameOrNum)

// JavaScript syntax
castLib(castNameOrNum);
```

Beschreibung

Diese Top-Level-Funktion gibt einen Verweis auf eine angegebene Besetzungsbibliothek zurück.

Die Standardbesetzungsbibliothek ist Nr. 1. Um einen Darsteller in einer anderen Besetzungsbibliothek als Besetzung 1 anzugeben, stellen Sie `castLib()` auf die andere Besetzungsbibliothek ein.

Parameter

castNameOrNum Erforderlich. Eine Zeichenfolge, die den Namen der Besetzungsbibliothek angibt, oder eine Ganzzahl, die die Nummer der Besetzungsbibliothek angibt.

Beispiel

Die folgende Anweisung setzt die Variable parts auf die zweite Besetzungsbibliothek fest:

```
-- Lingo syntax
parts = castLib(2)

// JavaScript syntax
var parts = castLib(2);
```

Siehe auch

`Cast Library`, `castLibNum`

channel() (Top-Level)

Syntax

```
-- Lingo syntax
channel(soundChannelNameOrNum)

// JavaScript syntax
channel(soundChannelNameOrNum);
```

Beschreibung

Diese Top-Level-Funktion gibt einen Verweis auf ein Sound Channel-Objekt zurück.

Parameter

soundChannelNameOrNum Erforderlich. Eine Zeichenfolge, die den Namen eines Soundkanals angibt, oder eine Ganzzahl, die die Indexposition eines Soundkanals angibt.

Mit dem Ausdruck `_movie.channel[1]` lässt sich in Lingo oder JavaScript der Kanal 1 eines Filmes ansprechen.

Beispiel

Die folgende Anweisung setzt die Variable `newChannel` auf den Soundkanal 9:

```
-- Lingo syntax
newChannel = channel(9)

// JavaScript syntax
var newChannel = channel(9);
```

Siehe auch

`Sound Channel`

channel() (Sound)

Syntax

```
-- Lingo syntax
_sound.channel(intChannelNum)

// JavaScript syntax
_sound.channel(intChannelNum);
```

Beschreibung

Diese Soundmethode gibt einen Verweis auf einen angegebenen Soundkanal zurück.

Die Funktionalität dieser Methode ist mit der der Top-Level-Methode `sound()` identisch.

Parameter

intChannelNum Erforderlich. Eine Ganzzahl, die den Soundkanal angibt, auf den verwiesen werden soll.

Beispiel

Die folgende Anweisung setzt die Variable namens `myChannel` auf den Soundkanal 2:

```
-- Lingo syntax
myChannel = _sound.channel(2)

// JavaScript syntax
var myChannel = _sound.channel(2);
```

Siehe auch

`Sound`, `sound()`, `Sound Channel`

chapterCount()

Syntax

```
-- Lingo syntax
dvdObjRef.chapterCount({intTitle})

// JavaScript syntax
dvdObjRef.chapterCount({intTitle});
```

Beschreibung

Diese DVD-Methode gibt die Anzahl verfügbarer Kapitel in einem Titel an.

Parameter

intTitle Optional. Eine Ganzzahl, die den Titel angibt, der die zu zählenden Kapitel enthält. Wird sie ausgelassen, gibt `chapterCount()` die Anzahl der verfügbaren Kapitel im aktuellen Titel zurück.

Beispiel

Die folgende Anweisung gibt die Anzahl der Kapitel im aktuellen Titel zurück:

```
-- Lingo syntax
trace (member(1).chapterCount)-- 17

// JavaScript syntax
trace (member(1). chapterCount);// 17
```

Siehe auch

chapterCount, DVD

charPosToLoc()

Syntax

```
--Lingo syntax
memberObjRef.charPosToLoc(nthCharacter)

// JavaScript syntax
memberObjRef.charPosToLoc(nthCharacter);
```

Beschreibung

Diese Feldfunktion gibt den Punkt im gesamten Felddarsteller (nicht nur in dem Teil, der auf der Bühne erscheint) aus, der einem angegebenen Zeichen am nächsten liegt. Dies ist nützlich, wenn Sie die Position einzelner Zeichen bestimmen möchten.

Werte für `charPosToLoc` werden in Pixel von der linken oberen Ecke des Felddarstellers aus angegeben. Der Parameter *nthCharacter* ist 1 für das erste Zeichen im Feld, 2 für das zweite Zeichenusw.

Parameter

nthCharacter Erforderlich. Das zu testende Zeichen.

Beispiel

Die folgende Anweisung gibt den Punkt an, an dem das fünfzigste Zeichen im Felddarsteller "Schlagzeile" erscheint und ordnet das Ergebnis der Variablen location zu:

```
-- Lingo syntax
location = member("Headline").charPosToLoc(50)

// JavaScript syntax
var location = member("Headline").charPosToLoc(50);
```

chars()

Syntax

```
chars(stringExpression, firstCharacter, lastCharacter)
```

Beschreibung

Diese Funktion (nur Lingo) identifiziert eine Teilzeichenfolge (Substring) in einem Ausdruck.

Die Ausdrücke *firstCharacter* und *lastCharacter* müssen eine Position im String angeben.

Wenn *firstCharacter* und *lastCharacter* identisch sind, wird ein einzelnes Zeichen aus dem String zurückgegeben. Wenn *lastCharacter* größer ist als die Anzahl der Zeichen im String, wird nur ein Substring bis zur Länge des Strings angegeben. Wenn *lastCharacter* vor *firstCharacter* steht, gibt die Funktion den Wert `EMPTY` zurück.

Ein Beispiel für `chars()` in einem fertigen Film finden Sie im Film „Text“ im Ordner „Learning\\Lingo_“ im Director-Anwendungsordner.

In JavaScript-Syntax wird die `substr()`-Funktion des String-Objekts verwendet.

Parameter

stringExpression Erforderlich. Eine Zeichenfolge, die den Ausdruck angibt, aus dem ein Substring zurückgegeben wird.

firstCharacter Erforderlich. Eine Ganzzahl, die die Position angibt, an der der Substring beginnt.

lastCharacter Erforderlich. Eine Ganzzahl, die die Position angibt, an der der Substring endet.

Beispiel

Die folgende Anweisung identifiziert das zweite Zeichen im Wort *Adobe*:

```
put chars("Adobe", 2, 2)
-- "d"
```

Die folgende Anweisung identifiziert das zweite bis fünfte Zeichen im Wort *Adobe*:

```
put chars("Adobe", 2, 5)
-- "dobe"
```

Die folgende Anweisung versucht, das sechste bis zwanzigste Zeichen im Wort "Adobe" zu identifizieren. Da das Wort aber nur 10 Zeichen hat, zeigt das Ergebnis nur das sechste bis zehnte Zeichen an.

```
put chars ("Adobe", 2, 20)
-- "dobe"
```

Siehe auch

`char...of`, `length()`, `offset()` (Zeichenfolgenfunktion), `number` (Zeichen)

charToNum()

Syntax

```
(stringExpression).charToNum
charToNum(stringExpression)
```

Beschreibung

Diese Funktion (nur Lingo) gibt den ASCII-Code zurück, der dem ersten Zeichen eines Ausdrucks entspricht.

Die Funktion `charToNum()` ist besonders nützlich, wenn Sie den ASCII-Wert von Zeichen prüfen möchten, die sich durch Tastenkombinationen wie der Strg-Taste mit einer weiteren alphanumerischen Taste ergeben.

Director behandelt Groß- und Kleinbuchstaben gleich, wenn Sie sie unter Verwendung des Gleichheitszeichen-Operators (=) vergleichen. Zum Beispiel gibt die Anweisung `put ("M" = "m")` als Ergebnis 1 oder `TRUE` zurück.

Sie können Probleme vermeiden, wenn Sie mit `charToNum()` den ASCII-Code eines Zeichens zurückgeben und dann mit dem ASCII-Code auf das Zeichen verweisen.

In JavaScript-Syntax wird die `charCodeAt()`-Funktion des String-Objekts verwendet.

Parameter

stringExpression Erforderlich. Eine Zeichenfolge, die den zu testenden Ausdruck angibt.

Beispiel

Die folgende Anweisung zeigt den ASCII-Code für den Buchstaben A an:

```
put ("A").charToNum
-- 65
```

Der folgende Vergleich stellt fest, ob der eingegebene Buchstabe ein großes A ist und navigiert daraufhin entweder zur Sequenz "Richtige Antwort" oder "Falsche Antwort" im Drehbuch:

```
-- Lingo syntax
on CheckKeyHit theKey
    if (theKey).charToNum = 65 then
        go "Correct Answer"
    else
        go "Wrong Answer"
    end if
end

// JavaScript syntax
function CheckKeyHit(theKey) {
    if (theKey.charToNum() == 65)
        go("Correct Answer");
    } else {
        go("Wrong Answer");
    }
}
```

Siehe auch

numToChar()

clearAsObjects()

Syntax

```
-- Lingo syntax
clearAsObjects()

// JavaScript syntax
clearAsObjects();
```

Beschreibung

Dieser Befehl setzt den für ActionScript-Objekte verwendeten globalen Flash Player zurück und entfernt alle ActionScript-Objekte aus dem Speicher. In Lingo gespeicherte Bezüge auf diese Objekte werden hierbei weder gelöscht noch zurückgesetzt. Lingo-Bezüge sind auch weiterhin vorhanden, verweisen aber auf nicht mehr existierende Objekte. Jeder Bezug muss separat auf `VOID` gesetzt werden.

Der Befehl `clearAsObjects()` wirkt sich nur auf globale Objekte aus, wie z. B. das in der folgenden Anweisung erstellte Array:

```
-- Lingo syntax
myGlobalArray = newObject(#array)

// JavaScript syntax
myGlobalArray = new Array();
```

Der Befehl `clearAsObjects()` hat keinerlei Auswirkungen auf Objekte, die in Sprite-Bezügen wie dem folgenden erstellt wurden:

```
myArray = sprite(2).newObject(#array)
```

Parameter
Keiner

Beispiel
Die folgende Anweisung löscht alle global erstellten ActionScript-Objekte aus dem Speicher:

```
-- Lingo syntax
clearAsObjects()

// JavaScript syntax
clearAsObjects();
```

Siehe auch
`newObject()`, `setCallback()`

clearCache

Syntax
`clearCache`

Beschreibung
Dieser Befehl leert den Netzwerk-Cache von Director.

Der Befehl `clearCache` leert nur den Cache von Director, der vom Zwischenspeicher des Browsers unabhängig ist.

Wenn eine Datei gerade bearbeitet wird, verbleibt sie im Zwischenspeicher, bis sie nicht mehr gebraucht wird.

Parameter
Keiner

Beispiel
Die folgende Prozedur löscht den Cache, wenn die Filmwiedergabe beginnt:

```
-- Lingo syntax
on startMovie
    clearCache
end

// JavaScript syntax
function startMovie() {
    clearCache();
}
```

Siehe auch

`cacheDocVerify()`, `cacheSize()`

clearError()

Syntax

```
-- Lingo syntax
memberObjRef.clearError()

// JavaScript syntax
memberObjRef.clearError();
```

Beschreibung

Dieser Flash-Befehl setzt den Fehlerzustand eines gestreamten Flash-Darstellers auf 0 zurück.

Wenn beim Streaming eines Darstellers in den Speicher ein Fehler auftritt, stellt Director die Eigenschaft `state` des Darstellers auf –1 ein, um auf den Fehler hinzuweisen. In diesem Fall können Sie mit der Funktion `getError` den Fehlertyp feststellen und anschließend mit dem Befehl `clearError` den Fehlerzustand des Darstellers auf 0 zurücksetzen. Nachdem Sie den Fehlerzustand des Darstellers gelöscht haben, versucht Director, den Darsteller zu öffnen, wenn er wieder im Director-Film gebraucht wird. Wenn Sie die Eigenschaften `pathName`, `linked` und `preload` des Darstellers einstellen, wird der Fehlerzustand ebenfalls automatisch gelöscht.

Parameter

Keiner

Beispiel

Die folgende Anweisung prüft, ob beim Streamen des Flash-Darstellers `Dali` in den Speicher ein Fehler aufgrund fehlenden Speicherplatzes auftrat. Falls sich ein Speicherfehler ereignete, versucht das Skript mit dem Befehl `unloadCast` etwas Speicher freizugeben. Dann schickt es den Abspielkopf im Director-Film zum Bild `Artists`, in dem das Flash-Film-Sprite zuerst erscheint, damit Director erneut versuchen kann, den Flash-Film abzuspielen. Wenn ein Fehler anderer Art auftrat, geht das Skript zu einem Bild namens "Sorry“, in dem erklärt wird, dass der angeforderte Flash-Film nicht abgespielt werden kann.

```
-- Lingo syntax
on CheckFlashStatus
    if (member("Dali").getError() = #memory) then
        member("Dali").clearError()
        member("Dali").unload()
        unloadCast
    else
        _movie.go("Sorry")
    end if
end

// JavaScript syntax
function CheckFlashStatus() {
    var ge = member("Dali").getError();
    if (ge = "memory") {
        member("Dali").clearError();
        unloadCast;
        _movie.go("Artists");
    } else {
        _movie.go("Sorry");
    }
}
```

Siehe auch

state (Flash, SWA), getError() (Flash, SWA)

clearFrame()

Syntax

```
-- Lingo syntax
_movie.clearFrame()

// JavaScript syntax
_movie.clearFrame();
```

Beschreibung

Diese Filmmethode löscht alle Sprite-Kanäle in einem Bild während der Drehbuchaufzeichnung.

Parameter

Keiner

Beispiel

Die folgende Prozedur löscht den Inhalt aller Bilder, bevor sie dieses Bild während der Drehbucherstellung bearbeitet:

```
-- Lingo syntax
on newScore
    _movie.beginRecording()
    repeat with counter = 1 to 50
        _movie.clearFrame()
        _movie.frameScript = 25
        _movie.updateFrame()
    end repeat
    _movie.endRecording()
end

// JavaScript syntax
function newScore() {
    _movie.beginRecording();
    for (var i = 1; i <= 50; i++) {
        _movie.clearFrame();
        _movie.frameScript = 25;
        _movie.updateFrame();
    }
    _movie.endRecording();
}
```

Siehe auch

beginRecording(), endRecording(), Movie, updateFrame()

clearGlobals()

Syntax

```
-- Lingo syntax
_global.clearGlobals()

// JavaScript syntax
_global.clearGlobals();
```

Beschreibung

Diese globale Methode stellt alle globalen Variablen auf `VOID` (Lingo) oder `null` (JavaScript-Syntax) ein.

Diese Methode kann nützlich sein, wenn Sie globale Variablen initialisieren oder einen neuen Film öffnen, der einen neuen Satz globaler Variablen benötigt.

Parameter

Keiner

Beispiel

Die folgenden Prozeduren stellen alle globalen Variablen auf `VOID` (Lingo) oder `null` (JavaScript-Syntax) ein:

```
-- Lingo syntax
on mouseDown
    _global.clearGlobals()
end

// JavaScript syntax
function mouseDown() {
    _global.clearGlobals();
}
```

Siehe auch

Global

clone

Syntax

```
member(whichCastmember).model(whichModel).clone(cloneName)
member(whichCastmember).group(whichGroup).clone(cloneName)
member(whichCastmember).light(whichLight).clone(cloneName)
member(whichCastmember).camera(whichCamera).clone(cloneName)
```

Beschreibung

Dieser 3D-Befehl erstellt eine Kopie des Modells, der Gruppe, des Lichts bzw. der Kamera sowie aller zugehörigen Child-Nodes. Der Klon besitzt den gleichen Parent-Node wie das Modell, die Gruppe, das Licht bzw. die Kamera, aus dem/der er geklont wurde.

Ein Klon eines Modells verwendet die gleiche Modellressource und wird der gleichen `shaderList`-Liste zugeordnet wie das ursprüngliche Modell.

Wenn Sie den Parameter *cloneName* weglassen oder `""` angeben, wird der Klon zwar mit der Methode `count` nicht erfasst, erscheint aber trotzdem in der Szene.

Parameter

cloneName Erforderlich. Gibt den Namen des neuen Klons an.

Beispiel

Die folgende Anweisung erstellt einen Klon namens `Teapot2` aus dem Modell `Teapot` und gibt einen Bezug auf das neue Modell zurück:

```
-- Lingo
teapotCopy = member("3D World").model("Teapot").clone("Teapot2")

// Javascript
teapotCopy = member("3D World").getProp("model","Teapot").clone("Teapot2")
```

Siehe auch

cloneDeep, cloneModelFromCastmember, cloneMotionFromCastmember, loadFile()

cloneDeep

Syntax

```
member(whichCastmember).model(whichModel).cloneDeep(cloneName)
member(whichCastmember).group(whichGroup).cloneDeep(cloneName)
member(whichCastmember).light(whichLight).cloneDeep(cloneName)
member(whichCastmember).camera(whichCamera).cloneDeep(cloneName)
```

Beschreibung

Dieser 3D-Befehl erstellt eine Kopie des Modells, der Gruppe, des Lichts bzw. der Kamera sowie der folgenden Elemente:

- der vom ursprünglichen Modell bzw. der ursprünglichen Gruppe verwendeten Modellressourcen, Shader und Texturen
- der Child-Objekte des Modells, der Gruppe, des Lichts bzw. der Kamera
- der von den Child-Objekten verwendeten Modellressourcen, Shader und Texturen

Diese Methode nimmt mehr Speicherplatz in Anspruch und ist zeitaufwändiger ist als der Befehl `clone`.

Parameter

cloneName Erforderlich. Gibt den Namen des neuen Klons an.

Beispiel

Die folgende Anweisung erstellt eine Kopie des Modells `Teapot`, seiner Child-Objekte und der von `Teapot` und seinen Child-Objekten verwendeten Modellressourcen, Shader und Texturen. Die Variable `teapotCopy` ist ein Bezug auf das geklonte Modell.

```
-- Lingo
teapotCopy = member("3D World").model("Teapot").cloneDeep("Teapot2")

// Javascript
teapotCopy =member("3DWorld").getProp("model","Teapot").cloneDeep("Teapot2")
```

Siehe auch

`clone`, `cloneModelFromCastmember`, `cloneMotionFromCastmember`, `loadFile()`

cloneModelFromCastmember

Syntax

```
member(whichCastmember).cloneModelFromCastmember(newModelName, sourceModelName,
sourceCastmember)
```

Beschreibung

Dieser 3D-Befehl kopiert ein Modell aus einem Darsteller, benennt es um und fügt es als Child-Node seiner 3D-Welt in einen Darsteller ein.

Dieser Befehl kopiert außerdem die Child-Nodes von *sourceModelName* sowie die vom Modell und seinen Child-Nodes verwendeten Modellressourcen, Shader und Texturen.

Dieser Befehl funktioniert nur dann ordnungsgemäß, wenn der Quelldarsteller vollständig geladen ist.

Parameter
newModelName Required. Gibt den Namen des neu geklonten Modells an.

sourceModelName Erforderlich. Gibt das zu klonende Modell an.

sourceCastMember Erforderlich. Gibt den Darsteller an, der das zu klonende Modell enthält.

Beispiel
Die folgende Anweisung erstellt eine Kopie des Modells `Pluto` des Darstellers `Scene` und fügt diese dem Darsteller `Scene2` unter dem neuen Namen `Planet` hinzu. Ebenfalls importiert werden die Child-Nodes von `Pluto` sowie die von `Pluto` und seinen Child-Nodes verwendeten Modellressourcen, Shader und Texturen.

```
--Lingo
member("Scene2").cloneModelFromCastmember("Planet", "Pluto", member("Scene"))

// Javascript
member("Scene2").cloneModelFromCastmember("Planet", "Pluto", member("Scene"));
```

Siehe auch
`cloneMotionFromCastmember`, `clone`, `cloneDeep`, `loadFile()`

cloneMotionFromCastmember

Syntax
```
member(whichCastmember).cloneMotionFromCastmember(newMotionName, sourceMotionName,
sourceCastmember)
```

Beschreibung
Dieser 3D-Befehl kopiert eine Bewegung aus einem Darsteller, benennt sie um und fügt sie in einen Darsteller ein.

Dieser Befehl funktioniert nur dann ordnungsgemäß, wenn der Quelldarsteller vollständig geladen ist.

Parameter
newMotionName Erforderlich. Gibt den Namen der neu geklonten Bewegung an.

sourceMotionName Erforderlich. Gibt die zu klonende Bewegung an.

sourceCastMember Erforderlich. Gibt den Darsteller an, der die zu klonende Bewegung enthält.

Beispiel
Die folgende Anweisung kopiert die Bewegung „Gehen“ im Darsteller `ParkScene`, benennt die Kopie `FunnyWalk` und stellt sie in den Darsteller `gbMember`.

```
--Lingo
member("gbMember").cloneMotionFromCastmember("FunnyWalk", "Walk", member("ParkScene"))

// Javascript
member("gbMember").cloneMotionFromCastmember("FunnyWalk", "Walk", member("ParkScene"));
```

Siehe auch

map (3D), cloneModelFromCastmember, clone, cloneDeep, loadFile()

close()

Syntax

```
-- Lingo syntax
windowObjRef.close()

// JavaScript syntax
windowObjRef.close();
```

Beschreibung

Diese Fenstermethode schließt ein Fenster.

Das Schließen eines bereits geschlossenen Fensters hat keine Wirkung.

Beachten Sie bitte, dass ein Film beim Schließen des Fensters weder angehalten noch aus dem Speicher entfernt wird. Diese Methode schließt lediglich das Fenster, in dem der Film abgespielt wird. Sie können das Fenster mit der Methode `open() (Window)` sofort wieder öffnen. Dies ermöglicht den schnellen Zugriff auf Fenster, die verfügbar bleiben sollen.

Wenn Sie ein Fenster endgültig schließen und aus dem Speicher entfernen möchten, verwenden Sie die Methode `forget()`. Überprüfen Sie zuvor jedoch, dass keine Verweise auf den Film in diesem Fenster vorhanden sind, ehe Sie die Methode `forget()` verwenden, da ansonsten Fehler auftreten, wenn Skripts mit dem gelöschten Fenster kommunizieren oder interagieren möchten.

Parameter

Keiner

Beispiel

Die folgende Anweisung schließt das Fenster „Bedienfeld“, das sich im Unterordner „MIAW-Quellen“ im Ordner des aktuellen Films befindet:

```
-- Lingo syntax
window(_movie.path & "MIAW Sources\Panel").close()

// JavaScript syntax
window(_movie.path + "MIAW Sources\\Panel").close();
```

Die folgende Anweisung schließt das Fenster Nummer 5 in `windowList`:

```
-- Lingo syntax
window(5).close()

// JavaScript syntax
window(5).close();
```

Siehe auch

forget() (Fenster), open() (Fenster), Window

closeFile()

Syntax

```
-- Lingo syntax
fileioObjRef.closeFile()

// JavaScript syntax
fileioObjRef.closeFile();
```

Beschreibung

Diese Datei-E/A-Methode (Fileio) schließt eine Datei.

Parameter

Keiner

Siehe auch

`Fileio`

closeXlib

Syntax

```
closeXlib whichFile
```

Beschreibung

Dieser Befehl schließt eine Xlibrary-Datei.

In den `Xlibrary`-Dateien sind Xtra-Erweiterungen gespeichert. `Xlibrary`-Dateien sind Ressourcendateien, die `Xtra`-Erweiterungen enthalten.`HyperCard XCMDs` und `XFCNs` können ebenfalls in `Xlibrary`-Dateien gespeichert werden.

Der Befehl `closeXlib` kann nicht für URLs verwendet werden.

In Windows braucht die Erweiterung "DLL" für Xtra-Erweiterungen nicht ausdrücklich angegeben zu werden.

Es ist ratsam, jede geöffnete Datei sofort wieder zu schließen, wenn sie nicht mehr benötigt wird.

Hinweis: *Dieser Befehl wird im Shockwave Player nicht unterstützt.*

Parameter

whichFile Optional. Gibt die zu schließende Xlibrary-Datei an. Wenn sich *whichFile* in einem anderen Ordner befindet als der aktuelle Film, muss *whichFile* einen Pfadnamen enthalten. Wenn *whichFile* nicht angegeben ist, werden alle offenen Xlibrarys geschlossen.

Beispiel

Die folgende Anweisung schließt alle geöffneten XBibliothek-Dateien:

```
closeXlib
```

Die folgende Anweisung schließt die Xlibrary „Video Disc Xlibrary“, wenn sie sich im selben Ordner befindet wie der Film selbst:

```
closeXlib "Video Disc Xlibrary"
```

Die folgende Anweisung schließt die `Xlibrary Transporter Xtra`-Erweiterungen im Ordner `New Xtras`, der sich im selben Ordner befindet wie der Film. Das Laufwerk wird mit der Variablen `currentDrive` angegeben:

```
closeXlib "@:New Xtras:Transporter Xtras"
```

Siehe auch

`Interface()`, `openXlib`

color()

Syntax

```
-- Lingo syntax
color(intPaletteIndex)
color(intRed, intGreen, intBlue)

// JavaScript syntax
color(intPaletteIndex);
color(intRed, intGreen, intBlue);
```

Beschreibung

Diese Top-Level-Funktion und dieser Datentyp geben unter Verwendung von RGB- oder 8-Bit-Palettenindexwerten ein Color-Datenobjekt zurück.

Das resultierende Farbobjekt kann auf Darsteller, Sprites sowie die Bühne, falls möglich, angewendet werden.

Parameter

intPaletteIndex Erforderlich, wenn 8-Bit-Palettenwerte verwendet werden. Eine Ganzzahl, die den zu verwendenden 8-Bit-Palettenwert angibt. Gültige Werte: 0 bis 255. Alle anderen Werte werden abgeschnitten.

intRed Erforderlich, wenn RGB-Werte verwendet werden. Eine Ganzzahl, die die Rot-Farbkomponente in der aktuellen Palette angibt. Gültige Werte: 0 bis 255. Alle anderen Werte werden abgeschnitten.

intGreen Erforderlich, wenn RGB-Werte verwendet werden. Eine Ganzzahl, die die Grün-Farbkomponente in der aktuellen Palette angibt. Gültige Werte: 0 bis 255. Alle anderen Werte werden abgeschnitten.

intBlue Erforderlich, wenn RGB-Werte verwendet werden. Eine Ganzzahl, die die Blau-Farbkomponente in der aktuellen Palette angibt. Gültige Werte: 0 bis 255. Alle anderen Werte werden abgeschnitten.

Beispiel

Die folgenden Anweisungen zeigen die Farbe von Sprite 6 im Nachrichtenfenster an und stellen sie anschließend auf einen neuen Wert ein:

```
-- Lingo syntax
put(sprite(6).color) -- paletteIndex(255)
sprite(6).color = color(137)
put(sprite(6).color) -- paletteIndex(137)

// JavaScript syntax
put(sprite(6).color) // paletteIndex(255);
sprite(6).color = color(137);
put(sprite(6).color) // paletteIndex(137);
```

compress()

Syntax

```
byteArrayObject.compress()
```

Beschreibung

Diese Bytearray-Methode komprimiert den Inhalt des Bytearrays mit dem herkömmlichen Zlib-Verfahren. Nach dem Komprimieren wird die Position auf 1 gesetzt.

Beispiele

```
--Lingo syntax
bArray.compress()

//JavaScript syntax
bArray.compress();
```

constrainH()

Syntax

```
-- Lingo syntax
_movie.constrainH(intSpriteNum, intPosn)

// JavaScript syntax
_movie.constrainH(intSpriteNum, intPosn);
```

Beschreibung

Diese Filmmethode gibt eine Ganzzahl zurück, deren Wert von den horizontalen Koordinaten der linken und rechten Ränder eines Sprites abhängig ist.

Die zurückgegebene Ganzzahl kann einen der drei folgenden Werte annehmen.

- Liegt der Parameter *intPosn* zwischen den Werten der linken und rechten Koordinaten des Sprites, entspricht die zurückgegebene Ganzzahl *intPosn*.
- Ist der Parameter *intPosn* kleiner als der Wert der linken Koordinate des Sprites, wird die zurückgegebene Ganzzahl auf den Wert der linken Koordinate des Sprites festgelegt.
- Ist der Parameter *intPosn* größer als der Wert der rechten Koordinate des Sprites, wird die zurückgegebene Ganzzahl auf den Wert der rechten Koordinate des Sprites festgelegt.

Diese Methode ändert die Sprite-Eigenschaften nicht.

Sowohl die `constrainH()`- als auch die `constrainV()`-Methode beschränkt jeweils nur eine Achse.

Parameter

intSpriteNum Erforderlich. Eine Ganzzahl, die das Sprite angibt, dessen horizontale Koordinaten mit *intPosn* abgeglichen werden.

intPosn Erforderlich. Eine Ganzzahl, die mit den horizontalen Koordinaten der linken und rechten Ränder des Sprites, das durch *intSpriteNum* angegeben ist, abgeglichen wird.

Beispiel

Die folgenden Anweisungen prüfen die Funktion constrainH für Sprite 1, wenn seine linke und rechte Koordinate 40 bzw. 60 ist:

```
-- Lingo syntax
put(constrainH(1, 20)) -- 40
put(constrainH(1, 55)) -- 55
put(constrainH(1, 100)) -- 60

// JavaScript syntax
put(constrainH(1, 20)); // 40
put(constrainH(1, 55)); // 55
put(constrainH(1, 100)); // 60
```

Die Anweisung beschränkt einen Schieberegler (Sprite 1) auf die Enden einer Skala (Sprite 2), wenn der Mauszeiger über das Ende des Reglers hinaus bewegt wird:

```
-- Lingo syntax
sprite(1).locH = _movie.constrainH(2, _mouse.mouseH)

// JavaScript syntax
sprite(1).locH = _movie.constrainH(2, _mouse.mouseH);
```

Siehe auch

constrainV(), Movie

constrainV()

Syntax

```
-- Lingo syntax
_movie.constrainV(intSpriteNum, intPosn)

// JavaScript syntax
_movie.constrainV(intSpriteNum, intPosn);
```

Beschreibung

Diese Filmmethode gibt eine Ganzzahl zurück, deren Wert von den vertikalen Koordinaten der oberen und unteren Ränder eines Sprites abhängig ist.

Die zurückgegebene Ganzzahl kann einen der drei folgenden Werte annehmen.

- Liegt der Parameter *intPosn* zwischen den Werten der oberen und unteren Koordinaten des Sprites, entspricht die zurückgegebene Ganzzahl *intPosn*.
- Ist der Parameter *intPosn* kleiner als der Wert der oberen Koordinate des Sprites, wird die zurückgegebene Ganzzahl auf den Wert der oberen Koordinate des Sprites festgelegt.
- Ist der Parameter *intPosn* größer als der Wert der unteren Koordinate des Sprites, wird die zurückgegebene Ganzzahl auf den Wert der unteren Koordinate des Sprites festgelegt.

Diese Methode ändert die Sprite-Eigenschaften nicht.

Sowohl die `constrainV()`- als auch die `constrainH()`-Methode beschränkt jeweils nur eine Achse.

Parameter

intSpriteNum Erforderlich. Eine Ganzzahl, die das Sprite angibt, dessen vertikale Koordinaten mit *intPosn* abgeglichen werden.

intPosn Erforderlich. Eine Ganzzahl, die mit den vertikalen Koordinaten der linken und rechten Ränder des Sprites, das durch *intSpriteNum* angegeben ist, abgeglichen wird.

Beispiel

Die folgenden Anweisungen prüfen die Funktion constrainV für Sprite 1, wenn seine obere und untere Koordinate 40 bzw. 60 ist:

```
-- Lingo syntax
put(constrainV(1, 20)) -- 40
put(constrainV(1, 55)) -- 55
put(constrainV(1, 100)) -- 60

// JavaScript syntax
put(constrainV(1, 20)); // 40
put(constrainV(1, 55)); // 55
put(constrainV(1, 100)); // 60
```

Die Anweisung beschränkt einen Schieberegler (Sprite 1) auf die Enden einer Skala (Sprite 2), wenn der Mauszeiger über das Ende des Reglers hinaus bewegt wird:

```
-- Lingo syntax
sprite(1).locV = _movie.constrainV(2, _mouse.mouseH)

// JavaScript syntax
sprite(1).locV = _movie.constrainV(2, _mouse.mouseH);
```

Siehe auch

constrainH(), Movie

copyPixels()

Syntax

```
-- Lingo syntax
imageObjRef.copyPixels(sourceImgObj, destRectOrQuad, sourceRect {, paramList})

// JavaScript syntax
imageObjRef.copyPixels(sourceImgObj, destRectOrQuad, sourceRect {, paramList});
```

Beschreibung

Bildmethode Diese Grafikmethode kopiert den Inhalt eines Rechtecks aus einem vorhandenen Grafikobjekt in ein neuesGrafikobjekt.

Wenn Sie Pixel aus einem Bereich eines Darstellers in einen anderen Bereich desselben Darstellers kopieren möchten, sollten Sie sie zunächst in ein dupliziertes Grafikobjekt kopieren und anschließend wieder in den ursprünglichen Darsteller einfügen. Das unmittelbare Kopieren aus einem Bereich in einen anderen Bereich derselben Grafik ist nicht zu empfehlen.

Um den Farbeffekt „Matt“ mit `copyPixels()` zu simulieren, erstellen Sie mit `createMatte()` ein Mattobjekt und übergeben es als Wert für den Parameter `#maskImage`von `copyPixels()`.

Ein Beispiel für quad in einem fertigen Film finden Sie im Film „Quad“ im Ordner „Learning\\Lingo_“ im Director-Anwendungsordner.

Weist bei der Verwendung von `copyPixel` die Eigenschaft `useAlpha` für die Quell- oder die Zielgrafik den Wert false auf, hat das Ziel keine Alphainformationen. Wenn das Ziel Alphainhalte aufweisen soll, muss die Eigenschaft `useAlpha` sowohl für die Quell- als auch die Zielgrafik den Wert true haben.

Parameter

sourceImgObj Erforderlich. Ein Verweis auf das Quellgrafikobjekt, aus dem die Pixel kopiert werden.

destRectOrQuad Erforderlich, wenn Pixel in ein Rechteck aus Bildschirmkoordinaten oder ein Fließkomma-Quad kopiert werden. Das Rechteck oder „quad“, in das Pixel kopiert werden.

sourceRect Erforderlich. Das Quellrechteck, aus dem Pixel kopiert werden.

paramList Optional. Eine Parameterliste, die zum Manipulieren kopierter Pixel vor deren Platzierung in *destRect* bzw. *destQuad* verwendet werden kann. Die Eigenschaftsliste kann beliebig viele der folgenden Parameter enthalten.

Eigenschaft	Verwendung und Wirkung
`#color`	Die Vordergrundfarbe für Farbeffekte. Die Standardfarbe ist schwarz.
`#bgColor`	Die Hintergrundfarbe für Farbeffekte oder Hintergrundtransparenz. Die Standardfarbe ist Weiß.
`#ink`	Die Art des auf die kopierten Pixel anzuwendenden Farbeffekts. Hierbei kann es sich um ein Farbeffektsymbol oder den entsprechenden numerischen Farbeffektwert handeln. Der Standardfarbeffekt ist `#copy`.
`#blendLevel`	Der Grad der auf die kopierten Pixel anzuwendenden Mischung (Transparenz). Die möglichen Werte liegen zwischen 0 und 255. Der Standardwert ist 255 (opak). Werte unter 255 bewirken, dass der Parameter `#ink` gegebenenfalls von `#backgroundTransparent` in `#blend` oder `#blendTransparent` geändert wird. `#blendLevel` könnte auch durch `#blend` ersetzt werden. In diesem Fall wird ein Wertebereich von 0 bis 100 verwendet.
`#dither`	Der Wert `TRUE` oder `FALSE` bestimmt, ob die kopierten Pixel beim Einfügen in das *destRect* einer 8- oder 16-Bit-Grafik gerastert werden. Der Standardwert ist `FALSE`, d. h die kopierten Pixel werden direkt in die Farbpalette von *imageObjRef* eingefügt.
`#useFastQuads`	Der Wert `TRUE` oder `FALSE` bestimmt, ob Quad-Berechnungen beim Kopieren von Pixeln in ein *destQuad* nach der schnelleren, aber weniger genauen Methode durchgeführt werden. Setzen Sie diesen Parameter auf `TRUE`, wenn Sie Quads für einfache Drehungs- und Neigungsoperationen verwenden. Für alle anderen Quads, etwa bei Perspektivänderungen, sollten Sie die Standardeinstellung `FALSE` beibehalten. Der Standardwert ist `FALSE`.
`#maskImage`	Gibt ein mit der Methode `creatMask()` bzw. `createMatte()` erstelltes Masken- oder Mattobjekt an, das als Maske für die kopierten Pixel verwendet werden soll. Auf diese Weise können Sie die Wirkung der Farbeffekte „Maske“ und „Matt“ duplizieren. Wenn die Quellgrafik über einen Alphakanal verfügt und ihre Eigenschaft `useAlpha` auf `TRUE` eingestellt ist, wird das angegebene Masken- oder Mattobjekt ignoriert und der Alphakanal verwendet. In der Standardeinstellung ist keine Maske angegeben.
`#maskOffset`	Ein Punkt, der den x- und y-Versatz der in `#maskImage` angegebenen Maske angibt. Die Koordinaten beziehen sich auf die linke obere Ecke der Quellgrafik. Der Standardversatz ist (0, 0).

Beispiel

Die folgende Anweisung kopiert die gesamte Grafik des Darstellers "Happy" in das Rechteck des Darstellers "Flower". Wenn die Darsteller unterschiedlich groß sind, wird die Grafik des Darstellers "Happy" in das Rechteck des Darstellers "Flower" eingepasst.

```
-- Lingo
member("flower").image.copyPixels(member("Happy").image,member("flower").image.rect,
member("Happy").image.rect)

// JavaScript syntax
member("flower").image.copyPixels(member("Happy").image,member("flower").image.rect,
member("Happy").image.rect);
```

Die folgende Anweisung kopiert einen Teil der Grafik des Darstellers „Happy“ in einen Teil des Darstellers „Flower“. Der kopierte Teil des Darstellers "Happy" befindet sich in „rectangle(0, 0, 200, 90)“. Er wird in das „rectangle(20, 20, 100, 40)“ innerhalb der Grafik des Darstellers "Flower" eingefügt. Hierbei wird kopierte Teil des Darstellers "Happy" in das Rechteck eingepasst.

```
-- Lingo
member("flower").image.copyPixels(member("Happy").image,
rect(20,20,100,40),rect(0,0,200,90))

// JavaScript syntax
member("flower").image.copyPixels(member("Happy").image,
rect(20,20,100,40),rect(0,0,200,90))
```

Die folgende Anweisung kopiert die gesamte Grafik des Darstellers „Happy“ in ein Rechteck innerhalb der Grafik des Darstellers „Flower“. Da das Rechteck, in das die kopierte Grafik des Darstellers "Happy" eingefügt wird, genauso groß ist wie das Rechteck des Darstellers "Happy", ist keine Größenanpassung erforderlich. Da die kopierte Grafik mit einem Mischungsgrad von 50 halbtransparent ist, scheint der Bereich des Darstellers "Grafik", in den sie eingefügt wird, durch.

```
-- Lingo
member("flower").image.copyPixels(member("Happy").image, rect(90,110,290,310),
member("Happy").image.rect, [#blendLevel: 50])

// JavaScript syntax
member("flower").image.copyPixels(member("Happy").image, rect(90,110,290,310),
member("Happy").image.rect, \propList(symbol("blendLevel"),50))
```

Siehe auch

color(), image()

copyToClipBoard()

Syntax

```
-- Lingo syntax
memberObjRef.copyToClipBoard()

// JavaScript syntax
memberObjRef.copyToClipBoard();
```

Beschreibung

Diese Darstellermethode kopiert einen angegebenen Darsteller in die Zwischenablage.

Für das Aufrufen dieser Methode muss das Besetzungsfenster nicht aktiv sein.

Diese Methode eignet sich besonders zum Kopieren von Darstellern aus einem Film in einen anderen oder aus einer Anwendung in eine andere.

Parameter
Keiner

Beispiel
Die folgende Anweisung kopiert den Darsteller „Stuhl“ in die Zwischenablage:

```
-- Lingo syntax
member("chair").copyToClipBoard()

// JavaScript syntax
member("chair").copyToClipBoard();
```

Die folgende Anweisung kopiert Darsteller Nr. 5 in die Zwischenablage:

```
--- Lingo syntax
member(5).copyToClipBoard()

// JavaScript syntax
member(5).copyToClipBoard();
```

Siehe auch
`Member`, `pasteClipBoardInto()`

cos()

Syntax
```
(angle).cos
cos (angle)
```

Beschreibung
Diese Funktion (nur Lingo) berechnet den Kosinus des angegebenen Winkels. Der Winkel muss im Bogenmaß angegebenwerden.

In JavaScript-Syntax wird die `cos()`-Funktion des Math-Objekts verwendet.

Parameter
angle Erforderlich. Eine Ganzzahl, die den zu testenden Winkel angibt.

Beispiel
Die folgende Anweisung berechnet den Kosinus von PI geteilt durch 2 und zeigt das Ergebnis im Nachrichtenfenster an:

```
put (PI/2).cos
```

Siehe auch
`atan()`, `PI`, `sin()`

count()

Syntax

```
-- Lingo syntax
list.count
object.count

// JavaScript syntax
list.count;
object.count;
```

Beschreibung

Diese Funktion gibt die Anzahl der Einträge in einer linearen oder Eigenschaftsliste, die Anzahl der Eigenschaften in einem Parent-Skript (ohne Eigenschaften des Ancestor-Skripts) oder die Anzahl der Chunks (Zeichen, Zeilen oder Wörter) in einem Textausdruck zurück.

Der Befehl `count` ist auf lineare und Eigenschaftslisten, auf Objekte, die mit einem Parent-Skript erstellt wurden, und auf die Eigenschaft `the globals` anwendbar.

Ein Beispiel für `count()` in einem fertigen Film finden Sie im Film „Text“ im Ordner „Learning/Lingo“ im Director-Anwendungsordner.

Verwenden von „Count“ als Funktion mit folgender Syntax

```
x = count(object)
x = object.count()
```

gibt die Anzahl der Eigenschaften im Basisobjekt zurück. Verwenden von „Count“ als Eigenschaft mit der Syntax

```
x = object.count
```

gibt die Anzahl der Eigenschaften des Vorgängers zurück.

Parameter

Keiner

Beispiel

Die folgende Anweisung zeigt die Zahl 3 als Anzahl der Einträge an:

```
--Lingo syntax
put([10,20,30].count) -- 3

// JavaScript syntax
put(list(10,20,30).count); // 3
```

Javascript unterstützt `count(object)` nicht.

Siehe auch

globals

createFile()

Syntax

```
-- Lingo syntax
fileioObjRef.createFile(stringFileName)

// JavaScript syntax
fileioObjRef.createFile(stringFileName);
```

Beschreibung

Diese Datei-E/A-Methode (Fileio) erstellt eine angegebene Datei.

Parameter

stringFileName Erforderlich. Eine Zeichenfolge, die den Pfad und Namen der zu erstellenden Datei angibt.

Beispiel

Das folgende Beispiel erstellt eine Datei namens „xtra.txt“ in „c:\“.

```
-- Lingo syntax
objFileio = new xtra("fileio")
objFileio.createFile("c:\xtra.txt")

// JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.createFile("c:\xtra.txt");
```

Siehe auch

`Fileio`

createMask()

Syntax

```
imageObject.createMask()
```

Beschreibung

Diese Funktion erstellt ein Maskenobjekt und gibt es für die Verwendung in der Funktion `copyPixels()` zurück.

Maskenobjekte sind keine Grafikobjekte und dienen lediglich zum Duplizieren des Sprite-Farbeffekts „Maske“ mithilfe der Funktion `copyPixels()`. Sie können beim mehrmaligen Verwenden derselben Grafik als Maske Zeit sparen, indem Sie das Maskenobjekt nur ein Mal erstellen und in einer Variablen speichern, die sich jederzeit erneut aufrufen lässt.

Beispiel

Die folgende Anweisung kopiert die gesamte Grafik des Darstellers „Happy“ in ein Rechteck innerhalb der Grafik des Darstellers „Braunes Quadrat“. Darsteller „gradient2“ wird als Maske auf das kopierte Bild angewendet. Die Maske wird gegenüber dem Rechteck, in das die Grafik des Darstellers „Happy“ eingefügt wird, um 10 Pixel nach links und oben verschoben.

```
member("brown square").image.copyPixels(member("Happy").image, rect(20, 20, 150, 108),
member("Happy").rect,[#maskImage:member("gradient2").image.createMask(), maskOffset:point(-
10, -10)])
```

Siehe auch

copyPixels(), createMatte(), ink

createMatte()

```
imageObject.createMatte({alphaThreshold})
```

Beschreibung

Diese Funktion erstellt ein Mattobjekt und gibt es für die Verwendung in der Funktion `copyPixels()` zurück, sodass Sie den Sprite-Farbeffekt "Matt" duplizieren können. Hierbei wird das Mattobjekt aus der Alphaebene des angegebenen Grafikobjekts erstellt. Der optionale Parameter *alphaThreshold* schließt alle Pixel des Mattobjekts aus, deren Alphakanalwert unterhalb des angegebenen Schwellenwerts liegt. Er wird nur in Verbindung mit 32-Bit-Grafiken verwendet, die über einen Alphakanal verfügen. Der zulässige Wertebereich für *alphaThreshold* liegt zwischen 0 und 255.

Mattobjekte sind keine Grafikobjekte und sind lediglich mit der Funktion `copyPixels()` nützlich. Sie können beim mehrmaligen Verwenden derselben Grafik als Mattobjekt Zeit sparen, indem Sie das Mattobjekt nur ein Mal erstellen und in einer Variablen speichern, die sich jederzeit erneut aufrufen lässt.

Beispiel

Die folgende Anweisung erstellt ein neues Mattobjekt aus der Alphaebene des Grafikobjekts `testImage`, wobei Pixel mit einem Alphawert unter 50% ignoriert werden:

```
newMatte = testImage.createMatte(128)
```

Siehe auch

copyPixels(), createMask()

createSoundObject

Syntax

```
SoundObject Mixer.createSoundObject (SoundObjname, filepath, [startTime, endTime, loopCount,
loopStartTime, loopEndTime, preLoadTime])
SoundObject Mixer.createSoundObject(SoundObjname, castMem, [startTime, endTime, loopCount,
loopStartTime, loopEndTime, preLoadTime])
SoundObject Mixer.createSoundObject(SoundObjname,callbackFunction,[castMemRef],
[sampleRate,channelcount,bitDepth]) -- For more information, see „Bytearray als Eingabe für
ein Soundobjekt“ auf Seite 157])
```

Rückgabewert

SoundObject

Beschreibung

Diese Audiomethode erzeugt ein Soundobjekt.

Parameter

Parameter	Beschreibung	Standardwert
SoundObjName	Der mit dem Soundobjekt verbundene Name. Der Name der Soundobjekte muss eindeutig sein.	
filepath	Name der Datei, die Direktor mithilfe des Soundsobjekts abspielen soll.	
castMem	Der Sound-Darsteller, den Director mithilfe des Soundsobjekts abspielen soll.	
proplist	Die Eigenschaftenliste, in der die Eigenschaften des Soundobjekts festgelegt werden.	
startTime	Legt die Startzeit (in Millisekunden) des aktuellen Soundobjekts fest.	Beginn des Sounds
endTime	Legt die Endzeit (in Millisekunden) des aktuellen Soundobjekts fest.	Ende des Sounds
loopCount	Die Häufigkeit, mit welcher der durch `#loopStartTime` und `#loopEndTime` festgelegte Bereich des Soundobjekts abgespielt werden soll.	1
loopStartTime	Startpunkt (in Millisekunden innerhalb des Soundobjekts) des wiederholten Bereichs des Soundobjekts.	
loopEndTime	Endpunkt (in Millisekunden innerhalb des Soundobjekts) des wiederholten Bereichs des Soundobjekts.	
preLoadTime	Die Länge des Sounds (in Millisekunden), die vor dem Beginn der Wiedergabe von Director geladen werden soll.	

Beispiele

Das folgende Beispiel erzeugt das Soundobjekt `soundObject1` mit der festgelegten `propertylist` aus `member(3)`. Das neue Soundobjekt wird `mixer1` hinzugefügt. Für das erzeugte Soundobjekt wird die Referenz `soundObjRef` zurückgegeben.

```
-- Lingo syntax
on mouseUp me
mixer.createSoundObject("SoundObj1", member(3), [ #startTime: 20000, #endTime: 80000,
#loopCount: 5, #loopStartTime: 50000 , #loopEndTime :75000, #preLoadTime:6000]
end

//JavaScript syntax
function mouseup()
{
soundObjRef=mixer1.createSoundObject("soundobj1",member(3),propList(symbol("startTime")
,10000,symbol("endTime") ,40000, symbol("loopcount") ,5 ,symbol("loopstarttime") ,20000,
symbol("loopendtime") ,30000, symbol("preLoadTime") ,6000));}
```

Siehe auch

Mixer

crop() (Grafik)

Syntax

```
-- Lingo syntax
imageObjRef.crop(rectToCropTo)

// JavaScript syntax
imageObjRef.crop(rectToCropTo);
```

Beschreibung

Diese Grafikmethode gibt ein neues Grafikobjekt zurück, das eine Kopie eines Quellgrafikobjekts enthält, das auf ein vorgegebenes Rechteck zugeschnitten wurde.

Durch das Aufrufen von `crop()` wird das Quellgrafikobjekt nicht verändert.

Das neue Grafikobjekt gehört zu keinem Darsteller und ist nicht mit der Bühne verknüpft. Legen Sie die Darstellereigenschaft `image` fest, um die neue Grafik einem Darsteller zuzuweisen.

Parameter

rectToCropTo Erforderlich. Das Rechteck, auf das die neue Grafik zugeschnitten wird.

Beispiel

Die folgende Anweisung veranlasst Director, alle Sprites zuzuschneiden, die auf den Digitalvideo-Darsteller "Interview" verweisen.

```
-- Lingo
Dot syntax:
member("Interview").crop = TRUE
Verbose syntax:
set the crop of member "Interview" to TRUE

// Javascript
member("Interview").crop=true
```

Siehe auch

`image (Grafik)`, `image()`, `rect (Grafik)`

crop() (Bitmap)

Syntax

```
-- Lingo syntax
memberObjRef.crop()

// JavaScript syntax
memberObjRef.crop();
```

Beschreibung

Mit diesem Bitmapbefehl kann ein Bitmapdarsteller auf eine bestimmte Größe zugeschnitten werden.

Sie können den Befehl `crop` verwenden, um einen vorhandenen Darsteller zuzuschneiden oder eine Momentaufnahme der Bühne zu erstellen und für die Anzeige zuzuschneiden.

Das Registrierungskreuz bleibt an derselben Position, sodass die Bitmap im Verhältnis zur Originalposition nicht verschoben wird.

Parameter

rectToCropTo Erforderlich. Gibt das Rechteck an, auf das ein Darsteller zugeschnitten wird.

Beispiel

Die folgende Anweisung stellt einen vorhandenen Bitmapdarsteller auf eine Momentaufnahme der Bühne ein und schneidet die resultierende Grafik auf ein Rechteck in der Größe von Sprite 10 zu:

```
-- Lingo syntax
stageImage = (_movie.stage).image
spriteImage = stageImage.crop(sprite(10).rect)
member("sprite snapshot").image = spriteImage

// JavaScript syntax
var stageImage = (_movie.stage).image;
var spriteImage = stageImage.crop(sprite(10).rect);
member("sprite snapshot").image = spriteImage;
```

Siehe auch

picture (Darsteller)

cross

Syntax

```
vector1.cross(vector2)
```

Beschreibung

Diese 3D-Vektormethode gibt einen Vektor zurück, der im 90° Winkel zu *vector1* und *vector2* verläuft.

Beispiel

Im folgenden Beispiel ist "pos1" ein Vektor auf der x-Achse und "pos2" ein Vektor auf der y-Achse. Von pos1.cross(pos2) wird der Wert vector(0.0000, 0.0000, 1.00000e4) zurückgegeben, der im 90° Winkel zu "pos1" und "pos2" verläuft.

```
ppos1 = vector(100, 0, 0)
pos2 = vector(0, 100, 0)

-- Lingo
put pos1.cross(pos2)
-- vector( 0.0000, 0.0000, 1.00000e4 )

// Javascript
trace(pos1.cross(pos2))
// vector( 0.0000, 0.0000, 1.00000e4 )
```

Siehe auch

crossProduct(), perpendicularTo

crossProduct()

Syntax

```
vector1.crossProduct(vector2)
```

Beschreibung

Diese 3D-Vektormethode gibt einen Vektor zurück, der im 90° Winkel zu *vector1* und *vector2* verläuft.

Beispiel

Im folgenden Beispiel ist `pos1` ein Vektor auf der x-Achse und `pos2` ein Vektor auf der y-Achse. Von `pos1.crossProduct(pos2)` wird der Wert `vector( 0.0000, 0.0000, 1.00000e4 )` zurückgegeben, der im 90° Winkel zu `pos1` und `pos2` verläuft.

```
pos1 = vector(100, 0, 0)
pos2 = vector(0, 100, 0)

-- Lingo
put pos1.crossProduct(pos2)
-- vector( 0.0000, 0.0000, 1.00000e4 )

// Javascript
trace(pos1.crossProduct(pos2))
// vector( 0.0000, 0.0000, 1.00000e4 )
```

Siehe auch

perpendicularTo, cross

cursor()

Syntax

```
-- Lingo syntax
_player.cursor(intCursorNum)
_player.cursor(cursorMemNum, maskMemNum)
_player.cursor(cursorMemRef)

// JavaScript syntax
_player.cursor(intCursorNum);
_player.cursor(cursorMemNum, maskMemNum);
_player.cursor(cursorMemRef);
```

Beschreibung

Diese Player-Methode ändert den Darsteller oder den integrierten Cursor, der für einen Cursor verwendet wird. Die Änderung bleibt so lange in Kraft, bis Sie den Cursor auf 0 einstellen und somit ausschalten.

- Mit der Syntax `_player.cursor(`*cursorMemNum*`, `*maskMemNum*`)` können Sie die Nummer eines Darstellers, der als Cursor verwendet werden soll, und seine optionale Maske angeben. Der Hotspot des Cursors ist das Registrierungskreuz des Darstellers.

  Bei dem angegebenen Darsteller muss es sich um einen 1-Bit-Darsteller handeln. Falls der Darsteller größer als 16 x 16 Pixel ist, schneidet Director ihn von der linken oberen Bildecke aus auf ein Quadrat von 16 x 16 Pixel zurecht. Der Hotspot des Cursors ist immer noch das Registrierungskreuz des Darstellers.

- Verwenden Sie die Syntax `_player.cursor(`*cursorMemRef*`)` für benutzerdefinierte Cursor, die über das Cursor-Xtra verfügbar sind.

  Hinweis: *Obwohl das Cursor-Xtra Cursor aus verschiedenen Besetzungsbibliothekstypen zulässt, können Textdarsteller nicht als Cursor verwendet werden.*

- Verwenden Sie die Syntax `_player.cursor(`*intCursorNum*`)`, um Standardsystemcursor anzugeben. Der Begriff *intCursorNum* muss einen der folgenden ganzzahligen Werte enthalten:

Wert	**Beschreibung**
-1, 0	Pfeil
1	Einfügemarke
2	Kreuz
3	Querstange
4	Uhr (Mac) oder Sanduhr (Windows)
5	Norden Süden Osten Westen (NSOW)
6	Norden Süden (NS)
200	Leer (Cursor ausgeblendet)
254	Hilfe
256	Bleistift
257	Radiergummi
258	Auswählen
259	Farbeimer
260	Hand
261	Rechteck, Werkzeug
262	Abgerundetes Rechteck, Werkzeug
263	Kreis, Werkzeug
264	Linie, Werkzeug
265	Richt-Text, Werkzeug
266	Textfeld, Werkzeug
267	Schaltfläche, Werkzeug

Wert	**Beschreibung**
268	Kontrollkästchen, Werkzeug
269	Optionsfeld, Werkzeug
270	Platzierung, Werkzeug
271	Registrierungskreuz, Werkzeug
272	Lasso
280	Finger
281	Pipette
282	Warten, Maustaste gedrückt 1
283	Warten, Maustaste gedrückt 2
284	Vertikale Größe
285	Horizontale Größe
286	Diagonale Größe
290	Geschlossene Hand
291	Nicht Ablegen, Hand
292	Kopieren (geschlossene Hand)
293	Umgekehrter Pfeil
294	Drehen
295	Neigen
296	Horizontaler Doppelpfeil
297	Vertikaler Doppelpfeil
298	Südwest Nordost, Doppelpfeil
299	Nordwest Südost, Doppelpfeil
300	Verwischen/Weichzeichnen, Pinsel
301	Airbrush
302	Vergrößern
303	Verkleinern
304	Zoom abbrechen
305	Startform
306	Punkt hinzufügen
307	Form schließen
308	Kamera zoomen
309	Kamera verschieben
310	Kamera drehen
457	Benutzerdefiniert

Beim Ablauf von Systemvorgängen, wie z. B. dem Laden einer Datei, ist es möglich, dass das Betriebssystem den Uhrcursor anzeigt und dann zu einem Pfeilcursor wechselt, wenn es die Steuerung wieder an die Anwendung zurückgibt. In diesem Fall werden die Einstellungen des Befehls `cursor` aus dem vorhergehenden Film außer Kraft gesetzt. Wenn Sie am Anfang eines neuen Films im Rahmen einer Präsentation mit mehreren Filmen, in denen jeweils eigene Cursor angezeigt werden, den Befehl `cursor()` verwenden möchten, sollten Sie die Ressourcennummern der Sondercursor daher in einer globalen Variablen speichern, die beim Filmwechsel nicht aus dem Speicher entladen wird.

Cursorbefehle können durch ein Xtra oder ein anderes externes Programm unterbrochen werden. Wenn der Cursor in Director auf einen bestimmten Wert eingestellt ist und ein Xtra oder ein externes Programm die Steuerung des Cursors übernimmt, hat das Zurücksetzen des Cursors auf den Ausgangswert keine Wirkung, da Director die Cursoränderung nicht erkennt. Sie können dies umgehen, indem Sie den Cursor ausdrücklich auf einen dritten Wert einstellen und ihn dann auf den Ausgangswert zurücksetzen.

Parameter

intCursorNum Erforderlich, wenn eine Ganzzahl zur Identifizierung eines Cursors verwendet wird. Eine Ganzzahl, die den integrierten Cursor angibt, der als Cursor verwendet werden soll.

cursorMemNum Erforderlich bei der Verwendung einer Darstellernummer mit optionaler Maske zur Identifizierung des Cursors. Eine Ganzzahl, die die Darstellernummer angibt, die als Cursor verwendet werden soll.

maskMemNum Erforderlich bei der Verwendung einer Darstellernummer mit optionaler Maske zur Identifizierung des Cursors. Eine Ganzzahl, die die Maskennummer von *cursorMemNum* angibt.

cursorMemRef Erforderlich bei der Verwendung eines Darstellerverweises zur Identifizierung des Cursors. Ein Verweis auf den Darsteller, der als Cursor verwendet werden soll.

Beispiel

Die folgende Anweisung ändert den Cursor in einen Uhrcursor (Mac) oder eine Sanduhr (Windows), wenn der Wert in der Variablen Status gleich 1 ist:

```
-- Lingo syntax syntax
if (status = 1) then
    _player.cursor(4)
end if

// JavaScript syntax
if (status == 1) {
    _player.cursor(4);
}
```

Die folgende Prozedur prüft, ob es sich bei dem der Variablen zugeordneten Darsteller um einen 1-Bit-Darsteller handelt, und verwendet ihn als Cursor, wenn dies der Fall ist:

```
-- Lingo syntax syntax
on myCursor(someMember)
    if (member(someMember).depth = 1) then
        _player.cursor(someMember)
    else
        _sound.beep()
    end if
end

// JavaScript syntax
function myCursor(someMember) {
    if (member(someMember).depth == 1) {
        _player.cursor(someMember);
    }
    else {
        _sound.beep();
    }
}
```

Siehe auch

Player

date() (Formate)

Syntax

```
-- Lingo syntax syntax
date({stringFormat})
date({intFormat})
date({intYearFormat, intMonthFormat, intDayFormat})

// JavaScript syntax
Date({"month dd, yyyy hh:mm:ss"});
Date({"month dd, yyyy"});
Date({yy,mm,dd,hh,mm,ss});
Date({yy,mm,dd});
Date({milliseconds});
```

Beschreibung

Diese Top-Level-Funktion und dieser Datentyp erstellen eine standardmäßige formatierte Datumsobjektinstanz, die mit anderen Datumsobjektinstanzen in arithmetischen Operationen und zur Manipulation von Datumsformaten über Plattformen hinweg und in internationalen Formaten verwendet werden kann.

Lingo-Datenobjekte und JavaScript-Syntax-Datenobjekte unterscheiden sich. Deshalb können keine Lingo-Datenobjekte mithilfe von JavaScript-Syntax und keine JavaScript-Syntax-Datenobjekte mithilfe von Lingo-Syntax erstellt werden.

Erstellen Sie ein neues Date-Objekt in JavaScript-Syntax mithilfe der `new Date()`-Syntax. In JavaScript-Syntax wird Groß- und Kleinschreibung unterschieden. So führt beispielsweise die Verwendung von `new date()` zu einem Laufzeitfehler.

Verwenden Sie bei der Erstellung des Datums vier Ziffern für das Jahr, zwei Ziffern für den Monat und zwei Ziffern für den Tag. Die folgenden Ausdrücke geben alle ein Datumsobjekt zurück, das dem 21. Oktober 2004 entspricht.

Datumsformat	Syntax
String	date(“20041021”)
Ganzzahl	date(20041021)
Durch Kommas getrennt	date(2004, 10, 21)

Es werden die folgenden einzelnen Eigenschaften des Datumsobjekts zurückgegeben.

Eigenschaft	Beschreibung
`#year`	Das Jahr in Form einer Ganzzahl
`#month`	Der Monat in Form einer Ganzzahl
`#day`	Der Tag des Monats in Form einer Ganzzahl

Additions- und Subtraktionsoperationen mit dem Datum werden als Addition und Subtraktion der Tage interpretiert.

Parameter

stringFormat Optional beim Erstellen eines Lingo-Datumsobjekts. Ein String, der das neue Datumsobjekt angibt.

intFormat Optional beim Erstellen eines Lingo-Datumsobjekts. Eine Ganzzahl, die das neue Datumsobjekt angibt.

intYearFormat Optional beim Erstellen eines Lingo-Datumsobjekts. Eine Ganzzahl, die das vierstellige Jahr des neuen Datumsobjekts angibt.

intMonthFormat Optional beim Erstellen eines Lingo-Datumsobjekts. Eine Ganzzahl, die den zweistelligen Monat des neuen Datumsobjekts angibt.

intDayFormat Optional beim Erstellen eines Lingo-Datumsobjekts. Eine Ganzzahl, die den zweistelligen Tag des neuen Datumsobjekts angibt.

month Optional beim Erstellen eines JavaScript-Syntax-Datumsobjekts. Eine Zeichenfolge, die den Monat des neuen Datumsobjekts angibt. Gültige Werte: 0 (Januar) bis 11 (Dezember).

dd Optional beim Erstellen eines JavaScript-Syntax-Datumsobjekts. Eine zweistellige Ganzzahl, die den Tag des neuen Datumsobjekts angibt. Gültige Werte: 0 (Sonntag) bis 6 (Samstag).

yyyy Optional beim Erstellen eines JavaScript-Syntax-Datumsobjekts. Eine vierstellige Ganzzahl, die das Jahr des neuen Datumsobjekts angibt.

hh Optional beim Erstellen eines JavaScript-Syntax-Datumsobjekts. Eine zweistellige Ganzzahl, die die Stunde des neuen Datumsobjekts angibt. Gültige Werte: 0 (0 Uhr) bis 23 (23 Uhr).

mm Optional beim Erstellen eines JavaScript-Syntax-Datumsobjekts. Eine zweistellige Ganzzahl, die die Minute des neuen Datumsobjekts angibt. Gültige Werte: 0 bis 59.

ss Optional beim Erstellen eines JavaScript-Syntax-Datumsobjekts. Eine zweistellige Ganzzahl, die die Sekunden des neuen Datumsobjekts angibt. Gültige Werte: 0 bis 59.

yy Optional beim Erstellen eines JavaScript-Syntax-Datumsobjekts. Eine zweistellige Ganzzahl, die das Jahr des neuen Datumsobjekts angibt. Gültige Werte: 0 bis 99.

milliseconds Optional beim Erstellen eines JavaScript-Syntax-Datumsobjekts. Eine Ganzzahl, die die Millisekunden des neuen Datumsobjekts angibt. Gültige Werte: 0 bis 999.

Beispiel

Die folgenden Anweisungen erstellen und bestimmen die Anzahl der Tage zwischen zwei Daten:

```
-- Lingo syntax syntax
myBirthday = date(19650712)
yourBirthday = date(19450529)
put("There are" && abs(yourBirthday - myBirthday) && "days between our birthdays.")

// JavaScript syntax
var myBirthday = new Date(1965, 07, 12);
var yourBirthday = new Date(1945, 05, 29);
put("There are " + Math.abs(((yourBirthday - myBirthday)/1000/60/60/24)) + " days between our
birthdays.");
```

Die folgenden Anweisungen greifen auf eine einzelne Eigenschaft eines Datums zu:

```
-- Lingo syntax syntax
myBirthday = date(19650712)
put("I was born in month number" && myBirthday.month)

// JavaScript syntax
var myBirthday = new Date(1965, 07, 12);
put("I was born in month number " + myBirthday.getMonth());
```

date() (System)

Syntax

```
-- Lingo syntax
_system.date({yyyymmdd})

// JavaScript syntax
_system.date({yyyymmdd});
```

Beschreibung

Diese Systemmethode gibt das aktuelle Datum der Systemuhr zurück.

Das von Director verwendete Format hängt dem auf dem jeweiligen Computer eingestellten Datumsformat ab.

- Unter Windows können Sie die Datumsanzeige im Dialogfeld „Ländereinstellungen“ in der Systemsteuerung anpassen. (Windows speichert das aktuelle Kurzformat des Datums in der Datei „System.ini“. Legen Sie mit diesem Wert fest, was die einzelnen Teile des kurzen Datums bedeuten.)
- Auf dem Macintosh können Sie das Datum im Kontrollfeld „Datum und Zeit“ anpassen.

Parameter

yyyymmdd Optional. Eine Zahl, die das vierstellige Jahr (*yyyy*), den zweistelligen Monat (*mm*) und den zweistelligen Tag (*dd*) des zurückgegebenen Datums angibt.

Beispiel

Die folgende Anweisung prüft, ob das aktuelle Datum der 1. Januar ist, indem sie feststellt, ob die ersten vier Zeichen des Datums 1.1. sind. Wenn dies der Fall ist, erscheint der Hinweis „Frohes Neues Jahr!“.

```
-- Lingo syntax
if (_system.date().char[1..4] = "1/1/") then
    _player.alert("Happy New Year!")
end if

// JavaScript syntax
if (_system.date().toString().substr(0, 4) == "1/1/") {
    _player.alert("Happy New Year!");
}
```

Siehe auch

System

delay()

Syntax

```
-- Lingo syntax
_movie.delay(intTicks)

// JavaScript syntax
_movie.delay(intTicks);
```

Beschreibung

Diese Filmmethode hält den Abspielkopf so lange wie angegeben an.

Während dieser Zeit besteht die einzig mögliche Maus- und Tastaturaktivität darin, den Film mit Strg+Alt+Punkt (Windows) bzw. Befehl+Punkt (Mac) zu stoppen. Da die `delay()`-Methode die Zeitdauer der einzelnen Bilder verlängert, ist sie nützlich, um die Abspielrate einer Sequenz von Bildern zu steuern.

Die `delay()`-Methode kann nur angewendet werden, solange sich der Abspielkopf bewegt. Prozeduren werden jedoch auch ausgeführt, wenn `delay()` aktiviert wurde, da die Methode nur den Abspielkopf, nicht aber das Skript selbst anhält. Stellen Sie Skripts, die die `delay()`-Methode verwenden, entweder in eine `enterFrame`- oder eine `exitFrame`-Prozedur.

Um bei stillstehendem Abspielkopf das Stoppverhalten in einer Prozedur zu simulieren, verwenden Sie die `milliseconds`-Eigenschaft des Systemobjekts und warten, bis die angegebene Zeit verstrichen ist, bevor Sie das Bild verlassen.

Parameter

intTicks Erforderlich. Eine Ganzzahl, die die Anzahl der Ticks angibt, für die der Abspielkopf angehalten werden soll. Ein Tick ist 1/60 Sekunde.

Beispiel

Die folgende Prozedur verzögert den Film um 2 Sekunden, wenn eine Taste gedrückt wird:

```
-- Lingo syntax
on keyDown
    _movie.delay(2*60)
end

// JavaScript syntax
function keyDown() {
    _movie.delay(2*60)
}
```

Die folgende Prozedur, die in ein Bildskript gesetzt werden kann, hält den Film eine nach dem Zufallsprinzip ausgewählte Anzahl von Ticks lang an:

```
-- Lingo syntax
on keyDown
    if (_key.key = "x") then
        _movie.delay(random(180))
    end if
end

// JavaScript syntax
function keyDown() {
    if (_key.key == "x") {
        _movie.delay(random(180));
    }
}
```

Siehe auch

endFrame, milliseconds, Movie

delete()

Syntax

```
delete chunkExpression
```

Beschreibung

Chunk-Ausdrücke sind beliebige Zeichen, Wörter, Elemente oder Zeilen in einem Zeichencontainer.

Parameter

Keiner

Beispiel

```
delete member(1).line[1]
```

Diese Anweisung löscht die erste Zeile des Textdarstellers.

```
delete member(1).word[1]
```

Diese Anweisung löscht das erste Wort des Textdarstellers.

```
delete member(1).char[1]
```

Diese Anweisung löscht das erste Zeichen des Textdarstellers.

delete() (FileIO)

Syntax

```
-- Lingo syntax
fileioObjRef.delete()

// JavaScript syntax
fileioObjRef.deleteFile();
```

Beschreibung

Diese Datei-E/A-Methode (Fileio) löscht eine Datei. Für JavaScript unterstützt Director 11 die neue Funktion „deleteFile()“.

Parameter

Keiner

Siehe auch

Fileio

deleteFile()

Syntax

```
-- Lingo syntax
fileioObjRef.deleteFile()

// JavaScript syntax
fileioObjRef.deleteFile();
```

Beschreibung

Diese Datei-E/A-Methode (Fileio) löscht eine Datei. Diese Methode wurde in Director 11 neu hinzugefügt.

Parameter

Keiner

Siehe auch

Fileio

deleteAt

Syntax

```
list.deleteAt(number)
deleteAt list, number
```

Beschreibung

Dieser Listenbefehl löscht einen Wert aus einer linearen oder Eigenschaftsliste.

Der Befehl `deleteAt` prüft, ob die Liste das betreffende Element enthält. Wenn Sie versuchen, ein Objekt zu löschen, das nicht in der Liste aufgeführt ist, zeigt Director einen Warnhinweis an.

Parameter
number Erforderlich. Gibt die Position des in der Liste zu löschenden Elements an.

Beispiel
Die folgende Anweisung löscht den zweiten Eintrag in der Liste "Designer" mit den Elementen [Gee, Kayne, Ohashi]:

```
--Lingo
designers = ["gee", "kayne", "ohashi"]
designers.deleteAt(2)

// Javascript
Designers = list("gee","kayne","ohashi");
Designers.deleteAt(2);
```

Das Ergebnis ist die Liste [Gee, Ohashi].

Die folgende Prozedur prüft, ob sich ein Objekt in der Liste befindet, bevor sie versucht, es zu löschen:

```
on myDeleteAt theList, theIndex
    if theList.count < theIndex then
        beep
    else
        theList.deleteAt(theIndex)
    end if
end
```

Siehe auch
addAt

deleteCamera

Syntax
```
member(whichCastmember).deleteCamera(cameraName)
member(whichCastmember).deleteCamera(index)
sprite(whichSprite).deleteCamera(cameraOrIndex)
```

Beschreibung
Bei einem Darsteller entfernt dieser 3D-Befehl die Kamera aus dem Darsteller und der 3D-Welt. Child-Objekte der Kamera werden zwar aus der 3D-Welt entfernt, aber nicht gelöscht.

Die Standardkamera des Darstellers kann nicht gelöscht werden.

Bei einem Sprite entfernt dieser Befehl die Kamera aus der Kameraliste des Sprites. Die Kamera wird nicht aus dem Darsteller gelöscht.

Parameter
cameraNameOrNum Erforderlich. Eine Zeichenfolge oder Ganzzahl, die den Namen oder die Indexposition der zu löschenden Kamera angibt.

Beispiel

Dieser Befehl löscht beide Kameras aus dem Darsteller „Room“, zunächst die Kamera „Camera06“ und dann „Camera 1“.

```
member("Room").deleteCamera("Camera06")
member("Room").deleteCamera(1)
```

Dieser Befehl löscht zwei Kameras aus der Kameraliste für „Sprite 5“, zunächst die zweite Kamera in der Liste, dann die Kamera mit dem Namen „Camera06“.

```
sprite(5).deleteCamera(2)
sprite(5).deleteCamera(member("Room").camera("Camera06"))
```

Siehe auch

`newCamera`, `addCamera`, `cameraCount()`

deleteFrame()

Syntax

```
-- Lingo syntax
_movie.deleteFrame()

// JavaScript syntax
_movie.deleteFrame();
```

Beschreibung

Diese Filmmethode löscht das aktuelle Bild und macht das nächste Bild zum aktuellen Bild. Sie kann nur während einer Sitzung zur Drehbucherstellung verwendet werden.

Parameter

Keiner

Beispiel

Die folgende Prozedur prüft, ob das Sprite in Kanal 10 des aktuellen Bildes über den rechten Rand einer 640 x 480 Pixel großen Bühne hinausgeht, und löscht das Bild, wenn dies der Fall ist:

```
-- Lingo syntax
on testSprite
    _movie.beginRecording()
    if (sprite(10).locH > 640) then
        _movie.deleteFrame()
    end if
    _movie.endRecording()
end

// JavaScript syntax
function testSprite() {
    _movie.beginRecording();
    if (sprite(10).locH > 640) {
        _movie.deleteFrame();
    }
    _movie.endRecording();
}
```

Siehe auch

beginRecording(), endRecording(), Movie, updateFrame()

deleteGroup

Syntax

```
member(whichCastmember).deleteGroup(whichGroup)
member(whichCastmember).deleteGroup(index)
```

Beschreibung

Dieser 3D-Befehl entfernt die Gruppe aus dem Darsteller und der 3D-Welt. Child-Objekte der Gruppe werden zwar aus der 3D-Welt entfernt, aber nicht gelöscht.

Die Gruppe "World" ist die Standardgruppe und kann nicht gelöscht werden.

Parameter

groupNameOrNum Erforderlich. Eine Zeichenfolge oder Ganzzahl, die den Namen oder die Indexposition der zu löschenden Gruppe angibt.

Beispiel

Durch die erste Zeile im folgenden Beispiel wird die Gruppe "Dummy16" aus dem Darsteller "Szene" gelöscht, durch die zweite Zeile die dritte Gruppe von "Szene":

```
member("Scene").deleteGroup("Dummy16")
member("Scene").deleteGroup(3)
```

Siehe auch

newGroup, child (3D), parent

deleteLight

Syntax

```
member(whichCastmember).deleteLight(whichLight)
member(whichCastmember).deleteLight(index)
```

Beschreibung

Dieser 3D-Befehl entfernt das Licht aus dem Darsteller und der 3D-Welt. Child-Objekte des Lichts werden zwar aus der 3D-Welt entfernt, aber nicht gelöscht.

Parameter

lightNameOrNum Erforderlich. Eine Zeichenfolge oder Ganzzahl, die den Namen oder die Indexposition des zu löschenden Lichts angibt.

Beispiel

In den folgenden Beispielen werden Lichter aus dem Darsteller "Raum" gelöscht:

```
member("Room").deleteLight("ambientRoomLight")
member("Room").deleteLight(6)
```

Siehe auch

newLight

deleteModel

Syntax

```
member(whichCastmember).deleteModel(whichModel)
member(whichCastmember).deleteModel(index)
```

Beschreibung

Dieser 3D-Befehl entfernt das Modell aus dem Darsteller und der 3D-Welt. Child-Objekte des Modells werden zwar aus der 3D-Welt entfernt, aber nicht gelöscht.

Parameter

modelNameOrNum Erforderlich. Eine Zeichenfolge oder Ganzzahl, die den Namen oder die Indexposition des zu löschenden Modells angibt.

Beispiel

Durch die erste Zeile im folgenden Beispiel wird das Modell `Player3` aus dem Darsteller `gbWorld` gelöscht. Die zweite Zeile löscht das neunte Modell aus `gbWorld`.

```
member("gbWorld").deleteModel("Player3")
member("gbWorld").deleteModel(9)
```

Siehe auch

newModel

deleteModelResource

Syntax

```
member(whichCastmember).deleteModelResource(whichModelResource)
member(whichCastmember).deleteModelResource(index)
```

Beschreibung

Dieser 3D-Befehl entfernt die Modellressource aus dem Darsteller und der 3D-Welt.

Modelle, die die gelöschte Modellressource verwenden, werden zwar unsichtbar, da sie ihre Geometrie verlieren, aber nicht aus der Welt entfernt oder gelöscht.

Parameter

resourceNameOrNum Erforderlich. Eine Zeichenfolge oder Ganzzahl, die den Namen oder die Indexposition der zu löschenden Ressource angibt.

Beispiel

In den folgenden Beispielen werden zwei Modellressourcen aus dem Darsteller `StreetScene` gelöscht:

```
member("StreetScene").deleteModelResource("HouseB")
member("StreetScene").deleteModelResource(3)
```

Siehe auch

`newModelResource`, `newMesh`

deleteMotion

Syntax

```
member(whichCastmember).deleteMotion(whichMotion)
member(whichCastmember).deleteMotion(index)
```

Beschreibung

Dieser 3D-Befehl entfernt die Bewegung aus dem Darsteller.

Parameter

motionNameOrNum Erforderlich. Eine Zeichenfolge oder Ganzzahl, die den Namen oder die Indexposition der zu löschenden Bewegung angibt.

Beispiel

Durch die erste Zeile im folgenden Beispiel wird die Bewegung `BackFlip` aus dem Darsteller `PicnicScene` gelöscht. Die zweite Zeile löscht die fünfte Bewegung aus `PicnicScene`.

```
member("PicnicScene").deleteMotion("BackFlip")
member("PicnicScene").deleteMotion(5)
```

Siehe auch

`newMotion()`, `removeLast()`

deleteOne

Syntax

```
list.deleteOne(value)
deleteOne list, value
```

Beschreibung

Dieser Listenbefehl entfernt einen Wert aus einer linearen oder Eigenschaftsliste. In einer Eigenschaftsliste löscht `deleteOne` auch die mit dem gelöschten Wert verknüpfte Eigenschaft. Ist der Wert mehrmals in der Liste vorhanden, wird nur der erste auftretende Wert von `deleteOne` gelöscht.

Eine Eigenschaft kann mit diesem Befehl nicht gelöscht werden.

Wenn Sie mithilfe der `add`- oder `append`-Methode der `filterlist` einen Filter hinzufügen, wird ein Duplikat erstellt und der Liste hinzugefügt. Methoden wie `deleteOne`, `getPos`, `findPos` und `getOne` verwenden den genauen Wert in der Liste und nicht den Duplikatwert.

In diesem Fall kann der `deleteOne`-Befehl wie folgt verwendet werden:

```
f = filter(#glowfilter)
sprite(1).filterlist.append(f)
f = sprite(1).filterlist[1]-- here we get the actual value added to the list.
sprite(1).filterlist.deleteOne(f)
```

Die dritte Zeile des Skripts fügt der Liste den Verweis des Filterwerts hinzu.

Parameter
value Erforderlich. Der aus der Liste zu löschende Wert.

Beispiel
Die erste Anweisung erstellt eine Liste, die aus den Tagen „Dienstag“, „Mittwoch“ und „Freitag“ besteht. Die zweite Anweisung entfernt die Bezeichnung „Mittwoch“ aus der Liste.

```
--Lingo
days = ["Tuesday", "Wednesday", "Friday"]
days.deleteOne("Wednesday")
put days

// Javascript
days = list("Tuesday", "Wednesday", "Friday");
days.deleteOne("Wednesday");
trace(days);
```

Die Anweisung put days bewirkt, dass das Ergebnis im Nachrichtenfenster angezeigt wird:

```
-- ["Tuesday", "Friday"].
```

deleteProp

Syntax

```
list.deleteProp(item)
deleteProp list, item
```

Beschreibung
Dieser Listenbefehl löscht den angegebenen Eintrag in der angegebenen Liste.

- In linearen Listen ersetzen Sie *item* durch die Zahl, die die Listenposition des zu löschenden Eintrags angibt. Der Befehl `deleteProp` für lineare Listen ist mit dem Befehl `deleteAt` identisch. Falls die Zahl größer ist als die Anzahl der Einträge in der Liste, wird ein Skriptfehler gemeldet.
- In Eigenschaftslisten ersetzen Sie *item* durch den Namen der zu löschenden Eigenschaft. Wenn Sie eine Eigenschaft löschen, wird auch der damit verbundene Wert gelöscht. Ist eine Eigenschaft mehrmals in der Liste vorhanden, wird nur die erste auftretende Eigenschaft in der Liste gelöscht.

Parameter
item Erforderlich. Das aus der Liste zu löschende Element.

Beispiel

Diese Anweisung löscht die Eigenschaft „Color“ aus der Liste: [`#height:100, #width: 200, #color: 34, #ink: 15`] mit Namen `spriteAttributes`:

```
spriteAttributes.deleteProp(#color)
```

Das Ergebnis ist die Liste: [`#height:100, #width: 200, #ink: 15`].

Siehe auch

`deleteAt`

deleteShader

Syntax

```
member(whichCastmember).deleteShader(whichShader)
member(whichCastmember).deleteShader(index)
```

Beschreibung

Dieser 3D-Befehl entfernt den Shader aus dem Darsteller.

Parameter

shaderNameOrNum Erforderlich. Eine Zeichenfolge oder Ganzzahl, die den Namen oder die Indexposition des zu löschenden Shaders angibt.

Beispiel

Durch die erste Zeile im folgenden Beispiel wird der Shader `Road` aus dem Darsteller `StreetScene` gelöscht. Die zweite Zeile löscht den dritten Shader aus `StreetScene`.

```
-- Lingo
member("StreetScene").deleteShader("Road")
member("StreetScene").deleteShader(3)

// Javascript
member("StreetScene").deleteShader("Road");
member("StreetScene").deleteShader(3);
```

Siehe auch

`shader`, `shaderList`, `newShader`

deleteSoundObject

Syntax

```
mixer.deleteSoundObject(soundObjRef)
mixer.deleteSoundObject(soundObjName)
```

Beschreibung

Diese Audiomethode löscht das Soundobjekt anhand der angebenen Referenz oder des Namens.

Parameter

Parameter	Beschreibung	Erforderlich/Optional
SoundObjRef	Soundobjekt-Referenz	Erforderlich
SoundObjName	Name des angegebenen Soundobjekts.	Erforderlich

Beispiele

```
-- Lingo syntax
on mouseUp me
     mixer1.deleteSoundObject(soundObjRef) --Deletes the sound object with the reference
soundobjRef.
--OR
     mixer1.deleteSoundObject("SoundObj1") --Deletes the sound object with the name "soundobj1".

end

// JavaScript syntax
function mouseup()
{
mixer1.deleteSoundObject(soundObjRef) //Deletes the sound object with the reference soundobjRef.
//OR
mixer1.deleteSoundObject("SoundObj1"); //Deletes the sound object with the reference
"soundobj1".

}
```

Siehe auch

Mixer

deleteTexture

Syntax

```
member(whichCastmember).deleteTexture(whichTexture)
member(whichCastmember).deleteTexture(index)
```

Beschreibung

Dieser 3D-Befehl entfernt eine Textur aus dem Darsteller.

Parameter

textureNameOrNum Erforderlich. Eine Zeichenfolge oder Ganzzahl, die den Namen oder die Indexposition der zu löschenden Textur angibt.

Beispiel

Durch die erste Zeile im folgenden Beispiel wird die Textur `Sky` aus dem Darsteller `PicnicScene` gelöscht. Die zweite Zeile löscht die fünfte Textur aus `PicnicScene`.

```
-- Lingo
member("PicnicScene).deleteTexture("Sky")
member("PicnicScene").deleteTexture(5)

// Javascript
member("PicnicScene).deleteTexture("Sky");
member("PicnicScene").deleteTexture(5)
```

Siehe auch

newTexture, texture

deleteVertex()

Syntax

```
-- Lingo syntax
memberObjRef.deleteVertex(indexToRemove)

// JavaScript syntax
memberObjRef.deleteVertex(indexToRemove);
```

Beschreibung

Dieser Vektorformbefehl entfernt einen vorhandenen Scheitelpunkt eines Vektorformdarstellers an der angegebenen Indexposition.

Parameter

indexToRemove Erforderlich. Eine Ganzzahl, die die Indexposition des zu löschenden Scheitelpunktes angibt.

Beispiel

Die folgende Zeile entfernt den zweiten Scheitelpunkt in der Vektorform "Archie“:

```
-- Lingo syntax
member("Archie").deleteVertex(2)

// JavaScript syntax
member("Archie").deleteVertex(2);
```

Siehe auch

addVertex(), moveVertex(), originMode, vertexList

displayOpen()

Syntax

```
-- Lingo syntax
fileioObjRef.displayOpen()

// JavaScript syntax
fileioObjRef.displayOpen();
```

Beschreibung
Diese Datei-E/A-Methode (Fileio) zeigt ein Dialogfeld „Öffnen“ an.

Diese Methode gibt den vollständigen Pfad und Namen der ausgewählten Datei an das Skript zurück.

Parameter
Keiner

Siehe auch
`Fileio`

displaySave()

Syntax
```
-- Lingo syntax
fileioObjRef.displaySave(stringTitle, stringFileName)

// JavaScript syntax
fileioObjRef.displaySave(stringTitle, stringFileName);
```

Beschreibung
Diese Datei-E/A-Methode (Fileio) zeigt ein Dialogfeld „Speichern“ an.

Diese Methode gibt den vollständigen Pfad und Namen der gespeicherten Datei an das Skript zurück.

Parameter
stringTitle Erforderlich. Eine Zeichenfolge, die den im Dialogfeld „Speichern“ angezeigten Titel angibt.

stringFileName Erforderlich. Eine Zeichenfolge, die den vollständigen Pfad und Namen der zu speichernden Datei angibt.

Siehe auch
`Fileio`

do

Syntax
```
do stringExpression
```

Beschreibung
Dieser Befehl wertet eine Zeichenfolge aus und führt das Ergebnis als Skriptanweisung aus. Dieser Befehl eignet sich zur Bewertung von Ausdrücken, die vom Benutzer eingegeben wurden, sowie zur Ausführung von Befehlen, die in Stringvariablen, Feldern, Arrays und Dateien gespeichert werden.

Die Verwendung nichtinitialisierter lokaler Variablen im Befehl `do` führt zu einem Kompilierungsfehler. Initialisieren Sie deshalb alle lokalen Variablen im Voraus.

Hinweis: *Mit diesem Befehl können keine globalen Variablen deklariert werden – diese Variablen müssen im Voraus deklariert werden.*

Der Befehl `do` funktioniert sowohl mit mehrzeiligen als auch mit einzeiligen Strings.

Parameter
stringExpression Erforderlich. Die auszuwertende Zeichenfolge.

Beispiel
Die folgende Anweisung führt die Anweisung in Anführungszeichen aus:

```
do "beep 2"
do commandList[3]
```

doneParsing()

Syntax
`parserObject.doneParsing()`

Beschreibung
Diese Funktion gibt `1 (TRUE)` zurück, wenn der Parser das Parsen eines Dokuments mit `parseURL()` beendet hat. Der Rückgabewert lautet `0 (FALSE)`, bis der Parsing-Vorgang abgeschlossen ist.

Parameter
Keiner

Siehe auch
`parseURL()`

dot()

Syntax
`vector1.dot(vector2)`

Beschreibung
Diese 3D-Vektormethode gibt die Summe der Produkte der x-, y- und z-Komponenten von zwei Vektoren zurück. Wenn beide Vektoren normiert sind, ist das Punktprodukt (`dot`) der Kosinus des Winkels zwischen den beiden Vektoren.

Um das Punktprodukt von zwei Vektoren manuell zu berechnen, multiplizieren Sie die x-Komponente von `vector1` mit der x-Komponente von `vector2`, die y-Komponente von `vector1` mit der y-Komponente von `vector2` und die z-Komponente von `vector1` mit der z-Komponente von `vector2`, und addieren Sie die drei Produkte.

Diese Methode ist mit der Funktion `dotProduct()` identisch.

Parameter
vector2 Erforderlich. Der zweite Vektor aus dem eine Summe zurückgegeben wird.

Beispiel

Im folgenden Beispiel beträgt der Winkel zwischen den Vektoren `pos5` und `pos6` ist 45°. Die Funktion `getNormalized` gibt die normierten Werte von `pos5` und `pos6` zurück und speichert sie in den Variablen `norm1` und `norm2`. Das Punktprodukt von `norm1` und `norm2` ist 0,7071, was dem Kosinus von 45° entspricht.

```
pos5 = vector(100, 100, 0)
pos6 = vector(0, 100, 0)
put pos5.angleBetween(pos6)
-- 45.0000
norm1 = pos5.getNormalized()
put norm1
-- vector( 0.7071, 0.7071, 0.0000 )
norm2 = pos6.getNormalized()
put norm2
-- vector( 0.0000, 1.0000, 0.0000 )
put norm1.dot(norm2)
-- 0.7071
```

Siehe auch

`dotProduct()`, `getNormalized`, `normalize`

dotProduct()

Syntax

```
vector1.dotProduct(vector2)
```

Beschreibung

Diese 3D-Vektormethode gibt die Summe der Produkte der x-, y- und z-Komponenten von zwei Vektoren zurück. Wenn beide Vektoren normiert sind, ist das Punktprodukt (`dotproduct`) der Kosinus des Winkels zwischen den beiden Vektoren.

Um das Punktprodukt von zwei Vektoren manuell zu berechnen, multiplizieren Sie die x-Komponente von `vector1` mit der x-Komponente von `vector2`, die y-Komponente von `vector1` mit der y-Komponente von `vector2` und die z-Komponente von `vector1` mit der z-Komponente von `vector2`, und addieren Sie die drei Produkte.

Diese Methode ist mit der Funktion `dot()` identisch.

Parameter

vector2 Erforderlich. Der zweite Vektor aus dem eine Summe zurückgegeben wird.

Beispiel

Im folgenden Beispiel beträgt der Winkel zwischen den Vektoren `pos5` und `pos6` 45°. Die Funktion `getNormalized` gibt die normierten Werte von `pos5` und `pos6` zurück und speichert sie in den Variablen `norm1` und `norm2`. Das `dotProduct` (Punktprodukt) von `norm1` und `norm2` ist 0,7071, was dem Kosinus von 45° entspricht.

```
pos5 = vector(100, 100, 0)
pos6 = vector(0, 100, 0)
put pos5.angleBetween(pos6)
-- 45.0000
norm1 = pos5.getNormalized()
put norm1
-- vector( 0.7071, 0.7071, 0.0000 )
norm2 = pos6.getNormalized()
put norm2
-- vector( 0.0000, 1.0000, 0.0000 )
put norm1.dotProduct(norm2)
-- 0.7071
```

Siehe auch

dot(), getNormalized, normalize

downloadNetThing

Syntax

```
downloadNetThing URL, localFile
```

Beschreibung

Dieser Befehl kopiert eine Datei aus dem Internet in eine Datei auf dem lokalen Datenträger, während der aktuelle Film weiter abgespielt wird. Verwenden Sie `netDone`, um herauszufinden, ob das Vorausladen bereits beendet ist.

Director-Filme im Authoring-Modus und Projektoren unterstützen den Befehl `downLoadNetThing`, der Shockwave-Player hingegen nicht. Dadurch wird verhindert, dass Benutzer ungewollt Dateien aus dem Internet kopieren.

Zwar können zahlreiche Netzwerkvorgänge gleichzeitig aktiv sein, es kommt allerdings zu hohen Leistungseinbußen, wenn mehr als vier Vorgänge parallel ausgeführt werden.

Das Verhalten des Befehls `downloadNetThing` wird weder von der Cachegröße noch von der Option "Dokumente überprüfen" beeinflusst.

Parameter

URL Erforderlich. Die URL eines beliebigen downloadbaren Objekts: beispielsweise auf einem FTP- oder HTTP-Server, eine Webseite, ein externer Darsteller, ein Director-Film oder eine Grafik.

localFile Erforderlich. Der Pfadname und der Dateiname für die Datei auf dem lokalen Datenträger

Beispiel

Die folgenden Anweisungen laden einen externen Darsteller von einer URL in den Director-Anwendungsordner und machen dann diese Datei zu einem externen Darsteller mit dem Namen "Besetzung der Tausende":

```
downLoadNetThing("http://www.cbDeMille.com/Thousands.cst",
the\applicationPath&"Thousands.cst")
castLib("Cast of Thousands").fileName = the applicationPath&"Thousands.cst"
```

Siehe auch

importFileInto(), netDone(), preloadNetThing()

draw()

Syntax

```
-- Lingo syntax
imageObjRef.draw(x1, y1, x2, y2, colorObjOrParamList)
imageObjRef.draw(point(x, y), point(x, y), colorObjOrParamList)
imageObjRef.draw(rect, colorObjOrParamList)

// JavaScript syntax
imageObjRef.draw(x1, y1, x2, y2, colorObjOrParamList);
imageObjRef.draw(point(x, y), point(x, y), colorObjOrParamList);
imageObjRef.draw(rect, colorObjOrParamList);
```

Beschreibung

Diese Grafikmethode zeichnet eine Linie oder eine nicht ausgefüllte Form mit einer angegebenen Farbe in einem rechteckigen Bereich eines vorgegebenen Grafikobjekts.

Wenn kein Fehler auftritt, gibt die Methode den Wert 1 zurück.

Wenn die optionale Parameterliste nicht angegeben wird, zeichnet `draw()` eine 1 Pixel breite Linie zwischen dem ersten und zweiten angegebenen Punkt oder zwischen der linken oberen und der rechten unteren Ecke des angegebenen Rechtecks.

Aus Leistungsgründen sollte das Farbobjekt bei Grafiken mit einer Farbtiefe bis einschließlich maximal 8Bit einen indizierten Farbwert enthalten. Für 16- oder 32-Bit-Grafiken verwenden Sie einen RGB-Farbwert.

Wenn Sie einen Bereich ausfüllen möchten, verwenden Sie die `fill()`-Methode.

Parameter

x1 Erforderlich beim Zeichnen einer Linie mithilfe von `x`- und `y`-Koordinaten. Eine Ganzzahl, die die `x`-Koordinate des Startpunktes der Linie angibt.

y1 Erforderlich beim Zeichnen einer Linie mithilfe von `x`- und `y`-Koordinaten. Eine Ganzzahl, die die `y`-Koordinate des Startpunktes der Linie angibt.

x2 Erforderlich beim Zeichnen einer Linie mithilfe von `x`- und `y`-Koordinaten. Eine Ganzzahl, die die `x`-Koordinate des Endpunktes der Linie angibt.

y2 Erforderlich beim Zeichnen einer Linie mithilfe von `x`- und `y`-Koordinaten. Eine Ganzzahl, die die `y`-Koordinate des Endpunktes der Linie angibt.

colorObjOrParamList Erforderlich. Ein Farbobjekt oder eine Parameterliste, das bzw. die die Farbe der Linie oder der Formumrandung angibt. Die Parameterliste kann anstelle eines einfachen Farbobjekts verwendet werden, um folgende Eigenschaften anzugeben.

Eigenschaft	Beschreibung
`#shapeType`	Einer der Symbolwerte `#oval`, `#rect`, `#roundRect` oder `#line`. Der Standard ist `#line`.
`#lineSize`	Die beim Zeichnen der Form zu verwendende Linienbreite.
`#color`	Ein Farbobjekt, das die Farbe der Formumrandung festlegt.

`point`*(x, y)*, `point`*(x, y)* Erforderlich beim Zeichnen einer Linie mithilfe von Punkten. Zwei Punkte, die die Start- und Endpunkte der Linie angeben.

rect Erforderlich beim Zeichnen einer Form. Ein Rechteck, das den rechteckigen Bereich angibt, in dem eine Form gezeichnet wird.

Beispiel

Die folgende Anweisung zeichnet eine dunkelrote Linie mit einer Breite von 1 Pixel zwischen dem Punkt (20, 20) und dem Punkt (20, 60) in der Grafik auf der Bühne.

```
-- Lingo
objImage = _movie.stage.image
objImage.draw(point(20, 20), point(20, 60), rgb(255, 0 ,0))

// Javascript
var objImage = _movie.stage.image ;
objImage.draw(point(20, 20), point(20, 60), color(255, 0 ,0)) ;
```

Siehe auch

`color()`, `copyPixels()`, `fill()`, `image()`, `setPixel()`

duplicate() (Grafik)

Syntax

```
-- Lingo syntax
imageObjRef.duplicate()

// JavaScript syntax
imageObjRef.duplicate();
```

Beschreibung

Diese Grafikmethode erstellt eine Kopie einer angegebenen Grafik und gibt sie zurück.

Die neue Grafik ist vollständig unabhängig vom Original und mit keinem Darsteller verknüpft. Wenn Sie umfangreiche Änderungen an einer Grafik vornehmen möchten, ist es von Vorteil, eine vom Darsteller unabhängige Kopie der Grafik zu erstellen.

Parameter

Keiner

Beispiel

Die folgende Anweisung erstellt ein neues Grafikobjekt aus der Grafik des Darstellers `Lunar Surface` und stellt das neue Grafikobjekt in die Variable `workingImage`:

```
workingImage = member("Lunar Surface").image.duplicate()
```

Siehe auch

`image()`

duplicate() (Listenfunktion)

Syntax

```
(oldList).duplicate()
duplicate(oldList)
```

Beschreibung

Diese Listenfunktion gibt eine Kopie der Liste zurück und kopiert verschachtelte Listen (d. h. Listenelemente, die selbst Listen sind) und deren Inhalt. Diese Funktion ist nützlich, um den aktuellen Inhalt einer Liste zu speichern.

Hinweis: *„filterlist()" kann mithilfe dieser Funktion nicht dupliziert werden.*

Wenn Sie eine Liste einer Variablen zuweisen, enthält die Variable einen Bezug auf die Liste und nicht die Liste selbst. Das heißt, dass sich alle Änderungen der Kopie auch auf die Originalliste auswirken.

Ein Beispiel für `duplicate() (list function)` in einem fertigen Film finden Sie im Film „Vektor Shapes" im Ordner „Learning\\Lingo_" im Director-Anwendungsordner.

Parameter

oldList Erforderlich. Gibt die zu duplizierende Liste an.

Beispiel

Die folgende Anweisung fertigt eine Kopie der Liste `CustomersToday` an und ordnet sie der Variablen `CustomerRecord` zu:

```
-- Lingo
CustomersToday = [1, 3, 5]
CustomerRecord = CustomersToday.duplicate()

// Javascript

CustomersToday = list(1, 3, 5);
CustomerRecord = CustomersToday.duplicate();
```

Siehe auch

`image()`

duplicate() (Darsteller)

Syntax

```
-- Lingo syntax
memberObjRef.duplicate({intPosn})

// JavaScript syntax
memberObjRef.duplicate({intPosn});
```

Beschreibung

Diese Darstellermethode erstellt eine Kopie eines angegebenen Darstellers.

Diese Methode sollte am besten nur beim Authoring und nicht zur Laufzeit verwendet werden, da sie einen zusätzlichen Darsteller im Speicher erstellt, was zu Speicherproblemen führen könnte.

Mithilfe dieser Methode können Sie Änderungen an Darstellern permanent mit der Datei speichern.

Parameter

intPosn Optional. Eine Ganzzahl, die das Besetzungsfenster für den duplizierten Darsteller angibt. Wird der Parameter nicht angegeben, wird der duplizierte Darsteller auf die erste leere Position im Besetzungsfenster platziert.

Beispiel

Die folgende Anweisung kopiert den Darsteller "Schreibtisch" an die erste leere Position im Besetzungsfenster:

```
-- Lingo syntax
member("Desk").duplicate()

// JavaScript syntax
member("Desk").duplicate();
```

Die folgende Anweisung kopiert den Darsteller „Schreibtisch“ an Position 125 im Besetzungsfenster:

```
-- Lingo syntax
member("Desk").duplicate(125)

// JavaScript syntax
member("Desk").duplicate(125);
```

Siehe auch

`Member`

duplicateFrame()

Syntax

```
-- Lingo syntax
_movie.duplicateFrame()

// JavaScript syntax
_movie.duplicateFrame();
```

Beschreibung

Diese Filmmethode erstellt ein Duplikat des aktuellen Bildes und seines Inhalts an, fügt das Bildduplikat hinter dem aktuellen Bild ein und macht es dann zum aktuellen Bild. Diese Methode kann nur bei der Drehbucherstellung verwendet werden.

Diese Methode hat dieselbe Funktionalität wie die `insertFrame()`-Methode.

Parameter

Keiner

Beispiel

In der folgenden Prozedur erstellt der Befehl `duplicateFrame` eine Reihe von Bildern, die den Darsteller `Ball` in der externen Besetzung `Toys` dem Sprite-Kanal 20 zuordnen. Die Anzahl der Bilder wird durch das Argument `numberOfFrames` bestimmt.

```
-- Lingo syntax
on animBall(numberOfFrames)
    _movie.beginRecording()
    sprite(20).member = member("Ball", "Toys")
    repeat with i = 0 to numberOfFrames
        _movie.duplicateFrame()
    end repeat
    _movie.endRecording()
end animBall

// JavaScript syntax
function animBall(numberOfFrames) {
    _movie.beginRecording();
    sprite(20).member = member("Ball", "Toys");
    for (var i = 0; i <= numberOfFrames; i++) {
        _movie.duplicateFrame();
    }
    _movie.endRecording();
}
```

Siehe auch

`insertFrame()`, `Movie`

enableHotSpot()

Syntax

```
-- Lingo syntax
spriteObjRef.enableHotSpot(hotSpotID, trueOrFalse)

// JavaScript syntax
spriteObjRef.enableHotSpot(hotSpotID, trueOrFalse);
```

Beschreibung

Dieser QTVR (QuickTime® VR)-Befehl bestimmt, ob ein Hotspot für ein bestimmtes QTVR-Sprite aktiviert (`TRUE`) oder deaktiviert ist (`FALSE`).

Parameter

hotSpotID Erforderlich. Gibt den im QTVR-Sprite zu testenden Hotspot an.

trueOrFalse Erforderlich. Ein `TRUE`- oder `FALSE`-Wert, der angibt, ob das QTVR-Sprite aktiviert ist.

enableSoundTrack(trackNum)

Syntax

```
trackInfo = member(1).trackInfo
member(1).enableSoundTrack(trackNumber)
```

Beschreibung

Diese Darstellermethode legt die Spurnummer des Soundtracks fest, der in einem Video mit mehreren Soundtracks abgespielt werden soll. Standardmäßig wird die erste Spur der Datei im Video abgespielt. Es kann jeweils nur eine Spur gleichzeitig aktiviert sein.

Hinweis: *Diese Eigenschaft ist nicht auf FLV-Darsteller anwendbar, da diese nur über einen Soundtrack verfügen.*

Parameter

Parameter	Beschreibung	Erforderlich/Optional
trackNum	Legt den Soundtrack fest, der für das Video abgespielt wird.	Erforderlich

Beispiele

```
-- Lingo syntax
trackInfo = member(1).trackInfo
member(1).enableSoundTrack(2)

// JavaScript syntax
trackInfo = member(1).trackInfo ;
    member(1).enableSoundTrack(2) ;
```

endRecording()

Syntax

```
-- Lingo syntax
_movie.endRecording()

// JavaScript syntax
_movie.endRecording();
```

Beschreibung

Diese Filmmethode beendet die Aktualisierungssitzung eines Drehbuchs.

Mithilfe von Skriptcode können Sie nach dem Aufrufen von `endRecording()` mit der Steuerung der Drehbuchkanäle fortfahren.

Parameter

Keiner

Beispiel

In der folgenden Prozedur beendet das Schlüsselwort `endRecording` die Erstellung des Drehbuchs:

```
-- Lingo syntax
on animBall(numberOfFrames)
    _movie.beginRecording()
    horizontal = 0
    vertical = 100
    repeat with i = 1 to numberOfFrames
        _movie.go(i)
        sprite(20).member = member("Ball")
        sprite(20).locH = horizontal
        sprite(20).locV = vertical
        sprite(20).foreColor = 255
        horizontal = horizontal + 3
        vertical = vertical + 2
        _movie.updateFrame()
    end repeat
    _movie.endRecording()
end animBall

// JavaScript syntax
function animBall(numberOfFrames) {
    _movie.beginRecording();
    var horizontal = 0;
    var vertical = 100;
    for (var i = 1; i <= numberOfFrames; i++) {
        _movie.go(1);
        sprite(20).member = member("Ball");
        sprite(20).locH = horizontal;
        sprite(20).locV = vertical;
        sprite(20).foreColor = 255;
        horizontal = horizontal + 3;
        vertical = vertical + 2;
        _movie.updateFrame();
    }
    _movie.endRecording();
}
```

Siehe auch

beginRecording(), Movie, updateFrame()

erase()

Syntax

```
-- Lingo syntax
memberObjRef.erase()

// JavaScript syntax
memberObjRef.erase();
```

Beschreibung

Diese Darstellermethode löscht einen angegebenen Darsteller und hält seine leere Position im Besetzungsfenster frei.

Zur Erzielung optimaler Ergebnisse sollten Sie diesen Befehl nur während der Filmerstellung und nicht in Projektoren verwenden. Die Verwendung dieser Methode in Projektoren kann zu Speicherproblemen führen.

Parameter
Keiner

Beispiel
Die folgende Anweisung löscht den Darsteller "Getriebe" in der Besetzung "Maschine":

```
-- Lingo syntax
member("Gear", "Hardware").erase()

// JavaScript syntax
member("Gear", "Hardware").erase();
```

Die folgende Prozedur löscht Darsteller, die von Anfang (start) bis Ende (finish) nummeriert sind:

```
-- Lingo syntax
on deleteMember start, finish
    repeat with i = start to finish
        member(i).erase()
    end repeat
end deleteMember

// JavaScript syntax
function deleteMember(start, finish) {
    for (var i=start; i<=finish; i++) {
        member(i).erase();
    }
}
```

Siehe auch
Member, new()

error()

Syntax

```
-- Lingo syntax
fileioObjRef.error(intError)

// JavaScript syntax
fileioObjRef.error(intError);
```

Beschreibung
Diese Datei-E/A-Methode (Fileio) gibt eine angegebene Fehlermeldung zurück.

Parameter
intError Erforderlich. Eine Ganzzahl, die den Fehler angibt. Gültige Werte sind: 0 („OK“) und 1 („Memory allocation failure“; Fehler bei der Arbeitsspeicherzuordnung). Alle anderen Werte geben „Unknown error“ (Unbekannter Fehler) zurück.

Beispiel
Das folgende Beispiel erstellt eine Datei namens "c:\check.txt". Kommt es beim Erstellen der Datei zu einem Fehler, wird diese Fehlermeldung ausgegeben.

```
-- Lingo syntax
objFileio = new xtra("fileio")
isok = objFileio.createFile("c:\check.txt")
if ( isok = 0 ) then
    alert( " file creation successful")
else
    alert( " file creation failed " & objFileio.error(isok))
end if

// JavaScript syntax
var objFileio = new xtra("fileio");
isok = objFileio.createFile("c:\check.txt");
if ( isok == 0 )
{
    _player.alert( " file creation successful");
}
else
{
    _player.alert( " file creation failed " + objFileio.error(isok));
}
```

Siehe auch

`Fileio`

externalEvent()

Syntax

```
externalEvent "string"
```

Beschreibung

Dieser Befehl sendet einen String an den Browser, der von diesem als Skriptanweisung interpretiert werden kann. Dadurch kann ein Film, der im Browser abgespielt wird, oder der Browser mit der HTML-Seite, in die der Film eingebettet ist, kommunizieren.

Dieser Befehl funktioniert nur bei Filmen, die in Browsern abgespielt werden.

Hinweis: *Der Befehl* `externalEvent` *gibt keinen Wert zurück. Deshalb kann nicht sofort festgestellt werden, ob der Browser das Ereignis verarbeitet oder ignoriert hat. Verwenden Sie* `on EvalScript` *im Browser, um eine Nachricht an den Film zurückzugeben.*

Parameter

string Erforderlich. Die an den Browser zu sendende Zeichenfolge. Die Zeichenfolge muss in einer Skriptsprache formuliert sein, die vom Browser unterstützt wird.

Beispiel

Die folgenden Anweisungen verwenden `externalEvent` in der LiveConnect-Skriptumgebung.

LiveConnect wertet den String, der durch `externalEvent` übergeben wurde, als Funktionsaufruf aus. JavaScript-Autoren müssen diese Funktion im HTML-Header definieren und benennen. Im Film werden der Funktionsname und die Parameter als String in `externalEvent` definiert. Da die Parameter vom Browser als separate Strings interpretiert werden müssen, wird jeder Parameter in einfache Anführungszeichen gesetzt.

Anweisungen im HTML-Code:

```
function MyFunction(parm1, parm2) {
    //script here
}
```

Anweisungen in einem Skript im Film:

```
externalEvent ("MyFunction('parm1','parm2')")
```

Die folgenden Anweisungen verwenden `externalEvent` in der ActiveX-Skriptumgebung, die von Internet Explorer® unter Windows® genutzt wird. ActiveX behandelt `externalEvent` als Ereignis und verarbeitet dieses Ereignis und dessen Stringparameter auf die gleiche Weise wie das Ereignis `onClick` in einem Schaltflächenobjekt.

Anweisungen im HTML-Code:

```
Sub
NameOfShockwaveInstance_externalEvent(aParam)
    'script here
End Sub
```

Eine andere Möglichkeit besteht darin, ein Skript für das Ereignis zu definieren:

```
<SCRIPT FOR="NameOfShockwaveInstance"
EVENT="externalEvent(aParam)"
LANGUAGE="VBScript">
    'script here
</SCRIPT>
```

Schließen Sie im Film die Funktion und alle Parameter als Teil des Strings für `externalEvent` ein:

```
externalEvent ("MyFunction ('parm1','parm2')")
```

Siehe auch

`on EvalScript`

extrude3D

Syntax

```
member(whichTextCastmember).extrude3D(member(which3dCastmember))
```

Beschreibung

Dieser 3D-Befehl erstellt eine neue `#extruder`-Modellressource in einem 3D-Darsteller aus dem Text in einem Textdarsteller:

Diese Vorgehensweise ist nicht mit der Verwendung der 3D-Eigenschaft `displayMode` eines Textdarstellers identisch.

So erstellen Sie ein Modell mit extrude3D:

1. Erstellen Sie eine neue `#extruder`-Modellressource in einem 3D-Darsteller:

   ```
   textResource = member("textMember").extrude3D(member("3DMember"))
   ```

2. Erstellen Sie ein neues Modell anhand der in Schritt 1 erzeugten Modellressource:

   ```
   member("3DMember").newModel("myText", textResource)
   ```

Parameter

which3dCastmember Erforderlich. Der Darsteller, in dem eine neue `#extruder`-Modellressource erstellt wird.

Beispiel

In diesem Beispiel ist "Logo" ein Textdarsteller und "Szene" ein 3D-Darsteller. Die erste Zeile erstellt in "Szene" eine Modellressource, bei der es sich um die 3D-Version des Texts in "Logo" handelt. Die zweite Zeile erstellt anhand dieser Modellressource ein Modell namens `3dLogo`.

```
-- Lingo
myTextModelResource =member("Logo").extrude3d(member("Scene"))
member("Scene").newModel("3dLogo", myTextModelResource)

// Javascript
myTextModelResource =member("Logo").extrude3d(member("Scene"));
member("Scene").newModel("3dLogo", myTextModelResource);
```

Siehe auch

`bevelDepth`, `bevelType`, `displayFace`, `smoothness`, `tunnelDepth`, `displayMode`

externalParamName()

Syntax

```
-- Lingo syntax
_player.externalParamName(paramNameOrNum)

// JavaScript syntax
_player.externalParamName(paramNameOrNum);
```

Beschreibung

Diese Player-Methode gibt den Namen eines bestimmten Parameters in der Liste der externen Parameter von einem HTML-Tag EMBED oder OBJECT zurück.

Wird ein Parameter anhand seines Namens angegeben, gibt diese Methode alle Parameternamen zurück, die mit *paramNameOrNum* übereinstimmen. Die Groß- und Kleinschreibung spielt dabei keine Rolle. Wird kein übereinstimmender Parametername gefunden, gibt diese Methode `VOID` (Lingo) oder `null` (JavaScript-Syntax) zurück.

Wird ein Parameter anhand der Zahl angegeben, gibt diese Methode den Parameternamen zurück, der sich an der *paramNameOrNum*-Position in der Parameterliste befindet. Wird keine übereinstimmende Parameterposition gefunden, gibt diese Methode `VOID` oder `null` zurück.

Diese Methode gilt nur für Filme mit Shockwave-Inhalt, die in einem Browser abgespielt werden. Sie kann nicht bei Director-Filmen oder Projektoren verwendet werden.

Die folgende Liste beschreibt die vordefinierten externen Parameter, die verwendet werden können.

Parameter	Definition
swAudio	Eine Zeichenfolge, die den Speicherort einer Shockwave Audio-Datei angibt, die mit dem Film abzuspielen ist. Der Wert ist eine vollqualifizierte URL.
swBackColor	Ein Farbwert, mit dem die Bühnenfarbeigenschaft des Films geändert werden soll. Der Wert ist ein Ganzzahlwert von 0 bis 255. Verwenden Sie 0 bis 255 für Filme mit 8-Bit-Farben und 0 bis 15 für Filme mit 4-Bit-Farben.
swBanner	Eine Zeichenfolge, die den Text angibt, der als Banner im Film verwendet werden soll.
swColor	Ein Farbwert, der zum Ändern der Farbe eines bestimmten Objekts verwendet wird. Der Wert ist eine Ganzzahl von 0 bis 255. Verwenden Sie 0 bis 255 für Filme mit 8-Bit-Farben und 0 bis 15 für Filme mit 4-Bit-Farben.
swForeColor	Ein neuer Vordergrundfarbwert. Text, der in Felddarsteller geschrieben wird, wird in der aktuell aktiven Vordergrundfarbe gerendert. Der Wert ist ein Ganzzahlwert von 0 bis 255. Verwenden Sie 0 bis 255 für Filme mit 8-Bit-Farben und 0 bis 15 für Filme mit 4-Bit-Farben.
swFrame	Ein Zeichenfolgenwert, der der Name ist, der einem vorgegebenen Bild im Film zugewiesen ist.
swList	Eine durch Kommas getrennte Liste von Elementen, die mit Skriptcode geparst werden kann. Listenwerte können Schlüssel/Wert-Paare, Boolesche Elemente, Ganzzahlen oder Zeichenfolgen sein.
swName	Ein Name, z. B. ein Benutzername, der in einem Film angezeigt oder verwendet werden soll.
swPassword	Ein Kennwort, z. B. zur Verwendung in Verbindung mit der `swName`-Eigenschaft, das im Film verwendet werden soll.
swPreloadTime	Ein Ganzzahlwert, der die Anzahl von Sekunden eines Audiodateisounds angibt, die vor dem Abspielen des Sounds vorgeladen werden sollen. Wird zusammen mit Shockwave Audio verwendet, um die Abspielleistung zu verbessern, indem die vor dem Abspielbeginn bereits heruntergeladene Audiomenge erhöht wird.
swSound	Ein Zeichenfolgenwert, der den Namen eines Sounds angeben kann, der im Director-Film wiedergegeben werden soll, bzw. ob überhaupt ein Sound abgespielt werden soll.
swText	Ein Zeichenfolgenwert, der den Text angibt, der im Film verwendet werden soll.
swURL	Eine Zeichenfolge-URL, die den Speicherort eines anderen Films mit Shockwave-Inhalt oder einer Shockwave Audio-Datei angeben kann.
swVolume	Ein Ganzzahlwert (empfohlen wird 0 bis 10), der zum Steuern der Lautstärke der Soundausgabe des Films verwendet wird. 0 ist aus (kein Sound), 10 ist maximale Lautstärke.
`sw1` bis `sw9`	Neun zusätzliche Eigenschaften für vom Autor definierte Parameter.

Parameter

paramNameOrNum Erforderlich. Eine Zeichenfolge, die den Namen des zurückzugebenden Parameternamens angibt, oder eine Ganzzahl, die die Indexposition des zurückzugebenden Parameternamens angibt.

Beispiel

Die folgende Anweisung stellt den Wert eines externen Parameters in die Variable `myVariable`:

```
-- Lingo syntax
if (_player.externalParamName("swURL") = "swURL") then
    myVariable = _player.externalParamValue("swURL")
end if

// JavaScript syntax
if (_player.externalParamName("swURL") == "swURL") {
    var myVariable = _player.externalParamName("swURL");
}
```

Siehe auch

externalParamValue(), Movie

externalParamValue()

Syntax

```
-- Lingo syntax
_player.externalParamValue(paramNameOrNum)

// JavaScript syntax
_player.externalParamValue(paramNameOrNum);
```

Beschreibung

Gibt den Namen eines bestimmten Parameters in der Liste der externen Parameter von einem HTML-Tag EMBED oder OBJECT zurück.

Wird ein Parameterwert anhand seines Namens angegeben, gibt diese Methode den Wert des ersten Parameters zurück, dessen Name mit *paramNameOrNum* übereinstimmt. Die Groß- und Kleinschreibung spielt dabei keine Rolle. Wird kein übereinstimmender Parameterwert gefunden, gibt diese Methode `VOID` (Lingo) oder `null` (JavaScript-Syntax) zurück.

Wird ein Parameterwert anhand der Indexposition angegeben, gibt diese Methode den Wert des Parameters zurück, der sich an der *paramNameOrNum*-Position in der Parameterliste befindet. Wird keine übereinstimmende Parameterposition gefunden, gibt diese Methode `VOID` oder `null` zurück.

Diese Methode gilt nur für Filme mit Shockwave-Inhalt, die in einem Browser abgespielt werden. Sie kann nicht bei Director-Filmen oder Projektoren verwendet werden.

Die folgende Liste beschreibt die vordefinierten externen Parameter, die verwendet werden können.

Parameter	Definition
swAudio	Eine Zeichenfolge, die den Speicherort einer Shockwave Audio-Datei angibt, die mit dem Film abzuspielen ist. Der Wert ist eine vollqualifizierte URL.
swBackColor	Ein Farbwert, mit dem die Bühnenfarbeigenschaft des Films geändert werden soll. Der Wert ist ein Ganzzahlwert von 0 bis 255. Verwenden Sie 0 bis 255 für Filme mit 8-Bit-Farben und 0 bis 15 für Filme mit 4-Bit-Farben.
swBanner	Eine Zeichenfolge, die den Text angibt, der als Banner im Film verwendet werden soll.
swColor	Ein Farbwert, der zum Ändern der Farbe eines bestimmten Objekts verwendet wird. Der Wert ist eine Ganzzahl von 0 bis 255. Verwenden Sie 0 bis 255 für Filme mit 8-Bit-Farben und 0 bis 15 für Filme mit 4-Bit-Farben.
swForeColor	Ein neuer Vordergrundfarbwert. Text, der in Felddarsteller geschrieben wird, wird in der aktuell aktiven Vordergrundfarbe gerendert. Der Wert ist ein Ganzzahlwert von 0 bis 255. Verwenden Sie 0 bis 255 für Filme mit 8-Bit-Farben und 0 bis 15 für Filme mit 4-Bit-Farben.
swFrame	Ein Zeichenfolgenwert, der der Name ist, der einem vorgegebenen Bild im Film zugewiesen ist.
swList	Eine durch Kommas getrennte Liste von Elementen, die mit Skriptcode geparst werden kann. Listenwerte können Schlüssel/Wert-Paare, Boolesche Elemente, Ganzzahlen oder Zeichenfolgen sein.
swName	Ein Name, z. B. ein Benutzername, der in einem Film angezeigt oder verwendet werden soll.
swPassword	Ein Kennwort, z. B. zur Verwendung in Verbindung mit der `swName`-Eigenschaft, das im Film verwendet werden soll.
swPreloadTime	Ein Ganzzahlwert, der die Anzahl von Sekunden eines Audiodateisounds angibt, die vor dem Abspielen des Sounds vorgeladen werden sollen. Wird zusammen mit Shockwave Audio verwendet, um die Abspielleistung zu verbessern, indem die vor dem Abspielbeginn bereits heruntergeladene Audiomenge erhöht wird.
swSound	Ein Zeichenfolgenwert, der den Namen eines Sounds angeben kann, der im Director-Film wiedergegeben werden soll, bzw. ob überhaupt ein Sound abgespielt werden soll.
swText	Ein Zeichenfolgenwert, der den Text angibt, der im Film verwendet werden soll.

Parameter	Definition
swURL	Eine Zeichenfolge-URL, die den Speicherort eines anderen Shockwave-Films oder einer Shockwave Audio-Datei angeben kann.
swVolume	Ein Ganzzahlwert (empfohlen wird 0 bis 10), der zum Steuern der Lautstärke der Soundausgabe des Films verwendet wird. 0 ist aus (kein Sound), 10 ist maximale Lautstärke.
`sw1` bis `sw9`	Neun zusätzliche Eigenschaften für vom Autor definierte Parameter.

Parameter

paramNameOrNum Erforderlich. Eine Zeichenfolge, die den Namen des zurückzugebenden Parameterwerts angibt, oder eine Ganzzahl, die die Indexposition des zurückzugebenden Parameterwerts angibt.

Beispiel

Die folgende Anweisung stellt den Wert eines externen Parameters in die Variable `myVariable`:

```
-- Lingo syntax
if (_player.externalParamName("swURL") = "swURL") then
    myVariable = _player.externalParamValue("swURL")
end if

// JavaScript syntax
if (_player.externalParamName("swURL") == "swURL") {
    var myVariable = _player.externalParamName("swURL");
}
```

Siehe auch

`externalParamName()`, `Movie`

extractAlpha()

Syntax

```
imageObject.extractAlpha()
```

Beschreibung

Diese Funktion kopiert den Alphakanal der angegebenen 32-Bit-Grafik und gibt sie als neues Grafikobjekt zurück. Das Ergebnis ist eine 8-Bit-Graustufengrafik, die den Alphakanal darstellt.

Diese Funktion dient zum Downsampling von 32-Bit-Grafiken mit Alphakanälen.

Beispiel

Die folgende Anweisung stellt den Alphakanal der Grafik von Darsteller 1 in die Variable `mainAlpha`.

```
-- Lingo
mainAlpha = member(1).image.extractAlpha()

// Javascript
mainAlpha = member(1).image.extractAlpha();
```

Siehe auch

`setAlpha()`

fadeIn()

Syntax

```
-- Lingo syntax
soundChannelObjRef.fadeIn({intMilliseconds})

// JavaScript syntax
soundChannelObjRef.fadeIn({intMilliseconds});
```

Beschreibung

Diese Soundkanalmethode setzt den Parameter `volume` eines Soundkanals sofort auf Null und anschließend im Laufe einer in Millisekunden angegebenen Zeitspanne schrittweise wieder auf den aktuellen Wert.

Die aktuelle Schwenkbereichseinstellung wird während des gesamten Überblendvorgangs beibehalten.

Parameter

intMilliseconds Optional. Eine Ganzzahl, die die Anzahl von Millisekunden angibt, über die die Lautstärke wieder zurück auf ihren Originalwert angehoben wird. Die Dauer dieser Zeitspanne beträgt standardmäßig 1000 Millisekunden (1 Sekunde), wenn kein Wert angegeben wird.

Beispiel

Die folgende Lingo-Anweisung blendet Soundkanal 3 ab dem Beginn des Sounddarstellers `introMusic2` über eine Zeitspanne von 3 Sekunden hinweg ein:

```
-- Lingo syntax
sound(3).play(member("introMusic2"))
sound(3).fadeIn(3000)

// JavaScript syntax
sound(3).play(member("introMusic2"));
sound(3).fadeIn(3000);
```

Siehe auch

`fadeOut()`, `fadeTo()`, `pan`, `Sound Channel`, `volume (Windows Media)`

fadeOut()

Syntax

```
-- Lingo syntax
soundChannelObjRef.fadeOut({intMilliseconds})

// JavaScript syntax
soundChannelObjRef.fadeOut({intMilliseconds});
```

Beschreibung

Diese Soundkanalmethode verringert den Parameter `volume` eines Soundkanals im Laufe einer in Millisekunden angegebenen Zeitspanne schrittweise auf Null.

Die aktuelle Schwenkbereichseinstellung wird während des gesamten Überblendvorgangs beibehalten.

Parameter

intMilliseconds Optional. Eine Ganzzahl, die die Anzahl von Millisekunden angibt, über die die Lautstärke auf Null verringert wird. Die Dauer dieser Zeitspanne beträgt standardmäßig 1000 Millisekunden (1 Sekunde), wenn kein Wert angegeben wird.

Beispiel

Die folgende Anweisung blendet Soundkanal 3 über eine Zeitspanne von 5 Sekunden hinweg aus:

```
-- Lingo syntax
sound(3).fadeOut(5000)

// JavaScript syntax
sound(3).fadeOut(5000);
```

Siehe auch

`fadeIn()`, `fadeTo()`, `pan`, `Sound Channel`, `volume (Windows Media)`

fadeTo()

Syntax

```
-- Lingo syntax
soundChannelObjRef.fadeTo(intVolume {, intMilliseconds})

// JavaScript syntax
soundChannelObjRef.fadeTo(intVolume {, intMilliseconds});
```

Beschreibung

Diese Soundkanalmethode verringert den Parameter `volume` eines Soundkanals im Laufe einer in Millisekunden angegebenen Zeitspanne schrittweise auf einen vorgegebenen Lautstärkewert.

Die aktuelle Schwenkbereichseinstellung wird während des gesamten Überblendvorgangs beibehalten.

Ein Beispiel für `fadeTo()` in einem fertigen Film finden Sie im Film „Sound Control“ im Ordner „Learning\\Lingo_“ im Director-Anwendungsordner.

Parameter

intVolume Erforderlich. Eine Ganzzahl, die die Lautstärke angibt, zu der gewechselt werden soll. Die möglichen Werte für die Lautstärke *intVolume* liegen zwischen 0 und 255.

intMilliseconds Optional. Eine Ganzzahl, die die Anzahl von Millisekunden angibt, über die die Lautstärke auf *intVolume* geändert wird. Die Dauer dieser Zeitspanne beträgt standardmäßig 1000 Millisekunden (1 Sekunde), wenn kein Wert angegeben wird.

Beispiel

Die folgende Anweisung stellt die Lautstärke von Soundkanal 4 im Laufe von 2 Sekunden schrittweise auf den Wert 150 ein. Je nach der ursprünglichen Lautstärkeneinstellung von Soundkanal 4 wird die Lautstärke hierbei höher oder niedriger gestellt.

```
-- Lingo syntax
sound(4).fadeTo(150, 2000)

// JavaScript syntax
sound(4).fadeTo(150, 2000);
```

Siehe auch

fadeIn(), fadeOut(), pan, Sound Channel, volume (Windows Media)

fileName()

Syntax

```
-- Lingo syntax
fileioObjRef.fileName()

// JavaScript syntax
fileioObjRef.fileName();
```

Beschreibung

Diese Datei-E/A-Methode (Fileio) gibt den vollständigen Pfad und Namen einer geöffneten Datei an.

Sie müssen zuerst durch Aufrufen von `openFile()` eine Datei öffnen, bevor Sie mithilfe von `fileName()` den Namen der Datei zurückgeben.

Parameter

Keiner

Beispiel

Die folgende Anweisung erstellt eine Datei und gibt deren Speicherort aus.

```
-- Lingo syntax
objFileio = new xtra("fileio")
objFileio.createFile(_player.ApplicationPath)
put objFileio.fileName()

// JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.createFile(_player.ApplicationPath);
trace(objFileio.fileName());
```

Siehe auch

Fileio , openFile()

fileOpen()

Syntax

```
FileOpen(MUIObject, string)
```

Beschreibung

Diese Funktion zeigt ein Standarddialogfeld zum Öffnen von Dateien an, das von einer Instanz des Xtras MUI bereitgestellt wird.

Der zweite Parameter gibt eine Zeichenfolge an, die im bearbeitbaren Feld angezeigt wird, wenn das Dialogfeld geöffnet wird. Der Benutzer kann die zu öffnende Datei angeben, indem er den Dateinamen in das bearbeitbare Feld eingibt. Wenn der Benutzer auf eine Schaltfläche klickt, wird der Text zurückgegeben.

- Wenn der Benutzer auf „Abbrechen“ klickt, entspricht der zurückgegebene Text dem eingegebenen Wert.
- Klickt der Benutzer auf „OK“, ist der zurückgegebene Text ein plattformspezifischer Pfad.

Parameter

Keiner

Beispiel

Diese Anweisungen dienen zum Erstellen und Anzeigen eines Standarddialogfelds zum Öffnen von Dateien.

- Die erste Anweisung erzeugt eine Instanz des Xtras MUI, bei der es sich um das Objekt handelt, das als Dialogfeld verwendet wird.
- Die zweite Anweisung weist der Variablen `fileString` eine Zeichenfolge zu, die später als zweiter Parameter des Befehls `FileOpen` verwendet wird.
- Die dritte Anweisung verwendet den Befehl `FileOpen` zum Generieren des Dialogfelds zum Öffnen von Dateien.
- Die letzten Anweisungen prüfen, ob die ursprüngliche Zeichenfolge, die mit dem Befehl `FileOpen` gesendet wurde, der Zeichenfolge entspricht, die zurückgegeben wurde, als der Benutzer auf eine Schaltfläche geklickt hat. Sind die Werte unterschiedlich, hat der Benutzer eine zu öffnende Datei ausgewählt.

```
-- Lingo syntax
set aMuiObj = new (xtra "MUI")
set fileString = "Open this file"
set result = fileOpen(aMuiObj, fileString)
-- Check to see if the dialog was canceled
if (result <> fileString) then
    -- The dialog wasn't canceled, do something with the new path data.
else
    put "ERROR - fileOpen requires a valid aMuiObj"
end if
```

Siehe auch

`Fileio` , `openFile()`

fileSave()

Syntax

```
FileSave( MUIObject, string, message )
```

Beschreibung
Diese Funktion zeigt ein Standarddialogfeld zum Speichern von Dateien an, das zum Speichern der aktuellen Datei dient. Das Dialogfeld wird aus einer Instanz des Xtras MUI erstellt.

- Der Zeichenfolgenparameter gibt die Zeichenfolge an, die im Dateinamenfeld des Dialogfelds angezeigt wird. Der Benutzer kann in dieses Feld einen neuen Dateinamen für die Datei eingeben. Wenn der Benutzer auf eine Schaltfläche klickt, gibt Lingo einen Wert für die Zeichenfolge zurück, die den Inhalt des Feldes enthält. Wenn der Benutzer auf „Abbrechen" klickt, entspricht die zurückgegebene Zeichenfolge der ursprünglichen Zeichenfolge.
- Der Nachrichtenparameter gibt die Zeichenfolge an, die über dem bearbeitbaren Feld des Dialogfelds angezeigt wird.

Parameter
Keiner

Beispiel
Diese Anweisungen dienen zum Erstellen und Anzeigen eines Standarddialogfelds zum Speichern von Dateien.

- Die erste Anweisung erzeugt eine Instanz des Xtras MUI, bei der es sich um das Objekt handelt, das als Dialogfeld verwendet wird.
- Die zweite Anweisung weist der Variablen `fileString` eine Zeichenfolge zu, die später als zweiter Parameter des Befehls `FileSave` verwendet wird.
- Die dritte Anweisung verwendet den Befehl `FileSave` zum Generieren des Dialogfelds zum Speichern von Dateien.
- Die letzten Anweisungen prüfen, ob das Ergebnis, nachdem der Benutzer auf eine Schaltfläche geklickt hat, der Zeichenfolge entspricht, die beim Öffnen des Dialogfelds gesendet wurde. Ist sie anders, hat der Benutzer auf eine andere Schaltfläche als „Abbrechen" geklickt.

```
-- Lingo syntax
set aMuiObj = new (xtra "MUI")
set fileString = "Save this file"
set result = fileSave(aMuiObj, fileString, "with this prompt")
-- Check to see if the dialog was canceled
if (result <> fileString) then
    -- The dialog wasn't canceled, do something with the new path data.
end if
```

fill()

Syntax

```
-- Lingo syntax
imageObjRef.fill(left, top, right, bottom, colorObjOrParamList)
imageObjRef.fill(point(x, y), point(x, y), colorObjOrParamList)
imageObjRef.fill(rect, colorObjOrParamList)

// JavaScript syntax
imageObjRef.fill(left, top, right, bottom, colorObjOrParamList);
imageObjRef.fill(point(x, y), point(x, y), colorObjOrParamList);
imageObjRef.fill(rect, colorObjOrParamList);
```

Beschreibung

Diese Grafikmethode füllt einen rechteckigen Bereich in einem angegebenen Grafikobjekt mit einer vorgegebenen Farbe.

Wenn kein Fehler auftritt, gibt die Methode den Wert 1 zurück, andernfalls wird der Wert 0 zurückgegeben.

Aus Leistungsgründen sollte das Farbobjekt bei Grafiken mit einer Farbtiefe bis einschließlich maximal 8Bit einen indizierten Farbwert enthalten. Für 16- oder 32-Bit-Grafiken verwenden Sie einen RGB-Farbwert.

Parameter

left Erforderlich zum Füllen eines durch Koordinaten angegebenen Bereichs. Eine Ganzzahl, die die linke Seite des zu füllenden Bereichs angibt.

top Erforderlich zum Füllen eines durch Koordinaten angegebenen Bereichs. Eine Ganzzahl, die die obere Seite des zu füllenden Bereichs angibt.

right Erforderlich zum Füllen eines durch Koordinaten angegebenen Bereichs. Eine Ganzzahl, die die rechte Seite des zu füllenden Bereichs angibt.

bottom Erforderlich zum Füllen eines durch Koordinaten angegebenen Bereichs. Eine Ganzzahl, die die untere Seite des zu füllenden Bereichs angibt.

colorObjOrParamList Erforderlich. Ein Farbobjekt oder eine Parameterliste, das bzw. die die Farbe zum Füllen des Bereichs angibt. Die Parameterliste kann anstelle eines einfachen Farbobjekts verwendet werden, um folgende Eigenschaften anzugeben.

Eigenschaft	Beschreibung
`#shapeType`	Einer der Symbolwerte `#oval`, `#rect`, `#roundRect` oder `#line`. Die Standardeinstellung ist `#line`.
`#lineSize`	Die beim Zeichnen der Form zu verwendende Linienbreite.
`#color`	Ein Farbobjekt, das die Füllfarbe des Bereichs festlegt.
`#bgColor`	Ein Farbobjekt, das die Farbe der Bereichsumrandung festlegt.

`point`*(x, y)*, `point`*(x, y)* Erforderlich beim Füllen eines Bereichs mithilfe von Punkten. Zwei Punkte, die die obere linke und untere rechte Ecke des zu füllenden Bereichs relativ zur oberen linken Ecke des vorgegebenen Grafikobjekts angeben.

rect Erforderlich beim Füllen eines Bereichs mithilfe eines Rechtecks. Ein Rechteck, das den zu füllenden rechteckigen Bereich angibt.

Beispiel

Die folgende Anweisung füllt einen durch die zwei Punkte festgelegten Bereich, die die obere linke und untere rechte Ecke des zu füllenden Bereichs relativ zur oberen linken Ecke des vorgegebenen Grafikobjekts „Bühne“ angeben.

```
-- Lingo
objImage = _movie.stage.image
objImage.fill(point(20, 20), point(30, 60), rgb(255, 0 ,0))

// Javascript
var objImage = _movie.stage.image ;
objImage.fill(point(20, 20), point(30, 60), color(255, 0 ,0)) ;
```

Siehe auch

`color()`, `draw()`, `image()`

filter()

Syntax

```
filter(filter symbol)
```

Beschreibung

Bitmap-Filtermethode zum Erstellen von Bitmap-Filtern, um diese auf ein Sprite anzuwenden. Die Funktion „filter()“ akzeptiert folgende Symbole:

#blurfilter Erstellt einen Weichzeichnerfilter

#glowfilter Erstellt einen Glühfilter

#bevelfilter Erzeugt einen Filter zur Abflachung von Kanten

#dropshadowfilter Erzeugt einen Schlagschattenfilter

#adjustcolorfilter Erzeugt einen Filter zur Farbanpassung

#gradientglowfilter Erzeugt den Filter „Farbverlauf/Glühen“

#gradientbevelfilter Erzeugt den Filter „Farbverlauf/Geschliffen“

#convolutionmatrixfilter Erzeugt einen Konvolutionsmatrixfilter

#displacementmapfilter Erzeugt einen Versatzabbildungsfilter.

Beispiel

Die erste Anweisung legt die Variable `myFilter` auf den Filter `Blur` fest. Die nächste Zeile wendet den Weichzeichnungsfilter („blur“) auf `sprite(1)` an.

```
--Lingo syntax
MyFilter=filter(#BlurFilter)
sprite(1).filterlist.append(MyFilter)
// JavaScript syntax
var MyFilter = filter(symbol("BlurFilter"));
sprite(1).filterlist.append(MyFilter);
```

Siehe auch

„Bitmap-Filter“ im Director-Benutzerhandbuch

findLabel()

Syntax

```
-- Lingo syntax
spriteObjRef.findLabel(whichLabelName)

// JavaScript syntax
spriteObjRef.findLabel(whichLabelName);
```

Beschreibung

Gibt die Bildnummer (im Flash-Film) zurück, die mit der angeforderten Beschriftung verbunden ist.

Der Wert 0 wird zurückgegeben, wenn die Beschriftung nicht existiert oder dieser Teil des Flash-Films noch nicht heruntergeladen wurde.

Parameter

whichLabelName Erforderlich. Gibt die zu suchende Grafikbeschriftung an.

Beispiel

Gibt die Bildnummer (im Flash-Film in Darsteller (1)) zurück, die der Beschriftung „GetMe“ zugeordnet ist.

```
-- Lingo syntax
sprite(1).findLabel("GetMe")

// JavaScript syntax
sprite(1).findLabel("GetMe");
```

findEmpty()

Syntax

```
-- Lingo syntax
castObjRef.findEmpty({memberObjRef})

// JavaScript syntax
castObjRef.findEmpty({memberObjRef});
```

Beschreibung

Diese Besetzungsbibliotheksmethode zeigt die nächste leere Darstellerposition oder die Position nach einem angegebenen Darsteller an.

Die Methode steht nur für die aktuelle Besetzungsbibliothek zur Verfügung.

Parameter

memberObjRef Optional. Ein Verweis auf den Darsteller, nach dem die nächste leere Darstellerposition angezeigt wird. Wird der Parameter nicht angegeben, wird die nächste leere Darstellerposition angezeigt.

Beispiel

Die folgende Anweisung sucht den ersten leeren Darsteller bei oder nach Darsteller 100:

```
-- Lingo syntax
trace(castLib(1).findEmpty(member(100)))

// JavaScript syntax
trace(castLib(1).findEmpty(member(100)));
```

Siehe auch

Cast Library, Member

findPos

Syntax

```
list.findPos(property)
findPos(list, property)
```

Beschreibung

Dieser Listenbefehl identifiziert die Position einer Eigenschaft in einer Eigenschaftsliste.

Wenn Sie `findPos` bei linearen Listen verwenden, wird eine falsche Zahl zurückgegeben, wenn der Wert von *property* eine Zahl ist, bzw. ein Skriptfehler, wenn der Wert von *property* ein String ist.

Der Befehl `findPos` führt dieselbe Funktion aus wie der Befehl `findPosNear`, mit dem Unterschied, dass `findPosVOID` ist, wenn die angegebene Eigenschaft nicht in der Liste enthalten ist.

Wenn Sie mithilfe der `add`- oder `append`-Methode der `filterlist` einen Filter hinzufügen, wird ein Duplikat erstellt und der Liste hinzugefügt. Methoden wie `deleteOne`, `getPos`, `findPos` und `getOne` verwenden den genauen Wert in der Liste und nicht den Duplikatwert.

In diesem Fall kann die `findPos`-Methode wie folgt verwendet werden:

```
f = filter(#glowfilter)
sprite(1).filterlist.append(f)
f = sprite(1).filterlist[1]-- here we get the actual value added to the list.
sprite(1).filterlist.findPos(f)
```

Die dritte Zeile in dem vorstehenden Skript fügt der Liste den Verweis des Filterwerts hinzu.

Parameter

property Erforderlich. Die Eigenschaft, deren Position identifiziert wird.

Beispiel

Die folgende Anweisung gibt die Position der Eigenschaft `c` in der Liste `Answers` an, die aus `[#a:10, #b:12, #c:15, #d:22]` besteht:

```
-- Lingo
Answers.findPos(#c)

// Javascript
Answers.findPos("c");
```

Das Ergebnis ist 3, da c in der Liste an dritter Stelle steht.

Siehe auch

`findPosNear`, `sort`

findPosNear

Syntax

```
sortedList.findPosNear(valueOrProperty)
findPosNear(sortedList, valueOrProperty)
```

Beschreibung
Dieser nur für geordnete Listen bestimmte Listenbefehl kennzeichnet die Position eines Elements in einer angegebenen geordneten Liste.

Der Befehl `findPosNear` funktioniert nur bei geordneten Listen. *valueOrProperty* kann bei geordneten linearen Listen durch einen Wert und bei geordneten Eigenschaftslisten durch eine Eigenschaft ersetzt werden.

Der Befehl `findPosNear` ist dem Befehl `findPos` ähnlich, mit dem Unterschied, dass der Befehl `findPosNear` die Position des Wertes mit dem ähnlichsten alphanumerischen Namen anzeigt, wenn die angegebene Eigenschaft nicht in der Liste enthalten ist. Dieser Befehl dient zum Auffinden eines Namens, der einem bestimmten Namen in einem geordneten Namensverzeichnis am ehesten entspricht.

Parameter
valueOrProperty Erforderlich. Der Wert oder die Eigenschaft, deren Position identifiziert wird.

Beispiel
Die folgende Anweisung gibt die Position einer Eigenschaft in der geordneten Liste „Atworten“ an, die aus [`#Nile:2, #Pharaoh:4, #Raja:0`] besteht:

```
Answers.findPosNear(#Ni)
```

Das Ergebnis ist 1, da „Ni“ dem Wort „Nil“ – der ersten Eigenschaft in der Liste – am ehesten entspricht.

Siehe auch
`findPos`

finishIdleLoad()

Syntax
```
-- Lingo syntax
_movie.finishIdleLoad(intLoadTag)

// JavaScript syntax
_movie.finishIdleLoad(intLoadTag);
```

Beschreibung
Dieser erzwingt das vollständige Herunterladen aller Darsteller, die mit dem angegebenen Ladekennzeichen („Load Tag“) markiert sind.

Parameter
intLoadTag Erforderlich. Eine Ganzzahl, die das Ladekennzeichen für die zu ladenden Darsteller angibt.

Beispiel
Die folgende Anweisung lädt alle Darsteller mit dem Ladekennzeichen 20 vollständig herunter:

```
-- Lingo syntax
_movie.finishIdleLoad(20)

// JavaScript syntax
_movie.finishIdleLoad(20);
```

Siehe auch

idleHandlerPeriod, idleLoadDone(), idleLoadMode, idleLoadPeriod, idleLoadTag, idleReadChunkSize, Movie

flashToStage()

Syntax

```
-- Lingo syntax
spriteObjRef.flashToStage(pointInFlashMovie)

// JavaScript syntax
spriteObjRef.flashToStage(pointInFlashMovie);
```

Beschreibung

Diese Funktion gibt die Koordinate der Director-Bühne zurück, die der angegebenen Koordinate in einem Flash-Film-Sprite entspricht. Die Funktion akzeptiert sowohl den Kanal als auch die Koordinate des Flash-Films und gibt die Koordinate der Director-Bühne als Director-Punktwerte („point“) zurück: point(300,300).

Flash-Filmkoordinaten werden in Flash-Film-Pixel angegeben, die durch die Originalgröße des Films bei seiner Erstellung in Flash bestimmt werden. Zum Errechnen von Flash-Filmkoordinaten liegt der Nullpunkt – point(0,0) – eines Flash-Films stets in der linken oberen Ecke. (Die Darstellereigenschaft `originPoint` wird nur zum Drehen und Skalieren verwendet und nicht zum Errechnen von Filmkoordinaten.)

Die Funktion `flashToStage` und die verwandte Funktion `stageToFlash` sind nützlich, wenn Sie feststellen möchten, welche Flash-Filmkoordinate sich direkt über einer Director-Bühnenkoordinate befindet. Sowohl bei Flash als auch bei Director ist „point(0,0)“ die linke obere Ecke der Bühne. Diese Koordinaten stimmen möglicherweise nicht mit der Director-Bühne überein, wenn ein Flash-Sprite gestreckt, skaliert oder gedreht wurde.

Parameter

pointInFlashMovie Erforderlich. Der Punkt im Flash-Film-Sprite, dessen Koordinaten zurückgegeben werden.

Beispiel

Die folgende Prozedur akzeptiert einen Punktwert und einen Sprite-Bezug als Parameter und setzt die obere linke Koordinate des angegebenen Sprites auf den angegebenen Punkt in einem Flash-Film-Sprite in Kanal 10:

```
-- Lingo syntax
on snapSprite(whichFlashPoint, whichSprite)
    sprite(whichSprite).loc = sprite(1).FlashToStage(whichFlashPoint)
    _movie.updatestage()
end

// JavaScript syntax
function snapSprite(whichFlashPoint, whichSprite) {
    sprite(whichSprite).loc = sprite(1).FlashToStage(whichFlashPoint);
    _movie.updateStage();
}
```

Siehe auch

stageToFlash()

float()

Syntax

```
(expression).float
float (expression)
```

Beschreibung

Diese Funktion (nur Lingo) konvertiert einen Ausdruck in eine Fließkommazahl. Die Anzahl der Ziffern nach dem Dezimalpunkt (dient lediglich der Anzeige, Berechnungen sind nicht davon betroffen) wird mit der Eigenschaft `floatPrecision` eingestellt.

In JavaScript-Syntax wird die `parseFloat()`-Funktion verwendet.

Parameter

expression Erforderlich. Der in eine Fließkommazahl zu konvertierende Ausdruck.

Beispiel

Die folgende Anweisung konvertiert die Ganzzahl 1 in die Fließkommazahl 1:

```
put (1).float
-- 1.0
```

Unter Verwendung von float können Rechenvorgänge ausgeführt werden. Ist eines der Elemente ein float-Wert, wird die gesamte Operation mit float ausgeführt:

```
put 2 + 2
-- 4
put (2).float + 2
-- 4.0
the floatPrecision = 4
put 22/7
-- 3
put (22).float / 7
-- 3.1429"
```

Siehe auch

`floatPrecision`, `ilk()`

floatP()

Syntax

```
(expression).floatP
floatP(expression)
```

Beschreibung

Diese Funktion (nur Lingo) gibt an, ob ein Ausdruck eine Fließkommazahl ist (1 oder `TRUE`) oder nicht (0 oder `FALSE`).

Der Buchstabe *P* in `floatP` steht für *predicate* (Prädikat).

Parameter
expression Erforderlich. Der zu testende Ausdruck.

Beispiel
Die folgende Anweisung prüft, ob 3.0 eine Fließkommazahl ist. Im Nachrichtenfenster wird die Zahl 1 angezeigt, was besagt, dass die Anweisung TRUE ist.

```
put (3.0).floatP
-- 1
```

Die folgende Anweisung prüft, ob 3 eine Fließkommazahl ist. Im Nachrichtenfenster wird die Zahl 0 angezeigt, was besagt, dass die Anweisung FALSE ist.

```
put (3).floatP
-- 0
```

Siehe auch
`float()`, `ilk()`, `integerP()`, `objectP()`, `stringP()`, `symbolP()`

flushInputEvents()

Syntax

```
-- Lingo syntax
_player.flushInputEvents()

// JavaScript syntax
_player.flushInputEvents();
```

Beschreibung
Diese Player-Methode entfernt alle anstehenden Maus- oder Tastaturereignisse aus der Director-Nachrichtenwarteschlange.

Sie dient im Allgemeinen dazu, Mausklicks oder Tastenbetätigungen zu unterdrücken, während Skriptcode eine enge Wiederholungsschleife durchläuft.

Diese Methode funktioniert nur zur Laufzeit während der Wiedergabe und ist während der Filmerstellung wirkungslos.

Parameter
Keiner

Beispiel
Die folgende Anweisung unterdrückt Maus- und Tastaturereignisse während der Ausführung einer Wiederholungsschleife:

```
-- Lingo syntax
repeat with i = 1 to 10000
    _player.flushInputEvents()
    sprite(1).loc = sprite(1).loc + point(1, 1)
end repeat

// JavaScript syntax
for (var i = 1; i <= 10000; i++) {
    _player.flushInputEvents();
    sprite(1).loc = sprite(1).loc + point(1, 1);
}
```

Siehe auch

on keyDown, on keyUp, on mouseDown (Ereignisprozedur), on mouseUp (Ereignisprozedur), Player

forget() (Fenster)

Syntax

```
-- Lingo syntax
windowObjRef.forget()

// JavaScript syntax
windowObjRef.forget();
```

Beschreibung

Diese Fenstermethode weist Skripts an, ein Fenster zu schließen und die Wiedergabe zu beenden, wenn es nicht mehr verwendet wird und keine anderen Variablen darauf verweisen.

Durch das Aufrufen von `forget()` für ein Fenster wird ebenfalls der Verweis dieses Fensters aus `windowList` entfernt.

Wenn die `forget()`-Methode aufgerufen wird, werden das Fenster und der Film in einem Fenster (MIAW) ausgeblendet, ohne dass die Prozeduren `stopMovie`, `closeWindow` oder `deactivateWindow` aufgerufen werden.

Wenn mehrere globale Bezüge auf den Film in einem Fenster verweisen, reagiert das Fenster nicht auf die `forget()`-Methode.

Parameter

Keiner

Beispiel

Die folgende Anweisung weist Lingo an, das Fenster "Steuerpult" zu löschen, wenn es vom Film nicht mehr verwendet wird:

```
-- Lingo syntax
window("Control Panel").forget()

// JavaScript syntax
window("Control Panel").forget();
```

Siehe auch

close(), open() (Fenster), Window, windowList

forget() (Timeout)

Syntax

```
timeout("timeoutName").forget()
forget(timeout("timeoutName"))
```

Beschreibung

Diese Timeout-Objekt-Funktion entfernt ein Timeout-Objekt aus der Liste `timeoutList` und verhindert, dass es weitere Timeout-Ereignisse sendet.

Parameter

Keiner

Beispiel

Die folgende Anweisung entfernt das `timeout`-Objekt `AlarmClock` aus der Liste `timeoutList`:

```
-- Lingo
timeout("AlarmClock").forget()

// Javascript
timeout("AlarmClock").forget();
```

Siehe auch

`timeout()`, `timeoutHandler`, `timeoutList`, `new()`

framesToHMS()

Syntax

```
framesToHMS(frames, tempo, dropFrame, fractionalSeconds)
```

Beschreibung

Diese Funktion konvertiert die angegebene Anzahl von Bildern in ihre entsprechende Länge in Stunden, Minuten und Sekunden. Diese Funktion ist nützlich, wenn Sie die tatsächliche Spielzeit eines Films ermitteln oder ein Videoabspielgerät steuern möchten.

Der resultierende String verwendet das Format `sHH:MM:SS.FFD`, wobei Folgendes gilt:

s	Wenn die Zeit kleiner als Null ist, wird ein Zeichen verwendet, ist sie größer oder gleich Null, wird ein Leerzeichen verwendet.
HH	Stunden.
MM	Minuten.
SS	Sekunden.
FF	Gibt den Bruchteil einer Sekunde an, wenn *fractionalSeconds* den Wert `TRUE` hat bzw. die Anzahl der Bilder (`frames`), wenn *fractionalSeconds*den Wert `FALSE` hat.
D	Ein `"d"` wird verwendet, wenn *dropFrame* den Wert `TRUE` hat bzw. ein Leerzeichen, wenn *dropFrame* den Wert `FALSE` hat.

Parameter

frames Erforderlich. Ein Ganzzahlausdruck, die die Anzahl der Bilder angibt.

tempo Erforderlich. Ein Ganzzahlausdruck, der die Wiedergabegeschwindigkeit in Bildern pro Sekunde angibt.

dropFrame Erforderlich. Kompensiert die Bildratenabweichung beim amerikanischen NTSC-System (Farbe), die nicht genau 30 Bilder pro Sekunde beträgt, und wird nur bei einer Bildrateneinstellung von 30 BpSbenötigt. Normalerweise ist dieser Parameter auf `FALSE` festgelegt.

fractionalSeconds Erforderlich. Stellt fest, ob die verbleibenden Bilder in das nächste Hundertstel einer Sekunde konvertiert werden (`TRUE`) oder die Anzahl der Bilder in Form eines ganzzahligen Wertes zurückgegeben wird (`FALSE`).

Beispiel

Die folgende Anweisung konvertiert einen Film mit 2710 Bildern und einer Bildrate von 30 BpS. Die Argumente `dropFrame` und `fractionalSeconds` sind beide deaktiviert:

```
put framesToHMS(2710, 30, FALSE, FALSE)
-- " 00:01:30.10 "
```

Siehe auch

`HMStoFrames()`

frameReady() (Film)

Syntax

```
-- Lingo syntax
_movie.frameReady({intFrameNum})
_movie.frameReady(frameNumA, frameNumB)

// JavaScript syntax
_movie.frameReady({intFrameNum});
_movie.frameReady(frameNumA, frameNumB);
```

Beschreibung

Diese Filmmethode ermittelt für Director-Filme, Projektoren und Filme mit Shockwave-Inhalt, ob die Darsteller eines Bildes oder Bildbereichs heruntergeladen wurden.

Die Methode gibt `TRUE` zurück, wenn die angegebenen Darsteller heruntergeladen wurden, und `FALSE`, wenn nicht.

Eine Demonstration der `frameReady()`-Methode in einem Director-Film sehen Sie im Beispielfilm „Streaming Shockwave“ in der Director-Hilfe.

Parameter

intFrameNum Optional beim Testen, ob die Darsteller eines Einzelbildes heruntergeladen wurden. Eine Ganzzahl, die das zu testende Einzelbild angibt. Wenn der Parameter ausgelassen wird, ermittelt `frameReady()`, ob die Darsteller in irgendeinem Bild des Drehbuchs heruntergeladen wurden.

frameNumA Erforderlich beim Testen, ob die Darsteller in einem Bildbereich heruntergeladen wurden. Eine Ganzzahl, die das erste zu testende Bild des Bereichs angibt.

frameNumB Erforderlich beim Testen, ob die Darsteller in einem Bildbereich heruntergeladen wurden. Eine Ganzzahl, die das letzte zu testende Bild des Bereichs angibt.

Beispiel

Die folgende Anweisung ermittelt, ob die Darsteller für Bild 20 heruntergeladen wurden und angezeigt werden können:

```
-- Lingo syntax
on exitFrame
    if (_movie.frameReady(20)) then
        _movie.go(20)
    else
        _movie.go(1)
    end if
end

// JavaScript syntax
function exitFrame() {
    if (_movie.frameReady(20)) {
        _movie.go(20);
    }
    else {
        _movie.go(1);
    }
}
```

Das folgende Bildskript prüft, ob Bild 25 eines Flash-Film-Sprites in Kanal 5 wiedergegeben werden kann. Wenn nein, läuft der Abspielkopf weiterhin im aktuellen Bild des Director-Films in einer Schleife. Sobald Bild 25 wiedergegeben werden kann, beginnt das Skript mit dem Abspielen des Films und lässt den Abspielkopf zum nächsten Bild des Director-Films übergehen.

Siehe auch

mediaReady, Movie

frameStep()

Syntax

```
-- Lingo syntax
dvdObjRef.frameStep(intFrames)

// JavaScript syntax
dvdObjRef.frameStep(intFrames);
```

Beschreibung

Diese DVD-Methode springt bei angehaltener Wiedergabe von der aktuellen Position aus um eine angegebene Anzahl von Bildern vor.

Das Zurückspringen wird für die DVD-Wiedergabe weder von Windows- noch von Mac-Systemsoftware unterstützt.

Parameter

intFrames Erforderlich. Eine Ganzzahl, die die Anzahl der Bilder angibt, um die nach vorne gesprungen werden soll.

Beispiel

Folgende Anweisung überspringt 100 Bilder:

```
-- Lingo syntax
member("drama").frameStep(100)

// JavaScript syntax
member("drama").frameStep(100);
```

Siehe auch

DVD

freeBlock()

Syntax

```
the freeBlock
```

Beschreibung

Diese Funktion gibt die Größe des größten zusammenhängenden freien Speicherblocks in Byte an. Ein Kilobyte (KB) entspricht 1024 Byte. Ein Megabyte (MB) entspricht 1024 Kilobyte. Damit ein Darsteller geladen werden kann, muss ein freier Speicherblock verfügbar sein, der mindestens so groß wie der Darsteller ist.

Parameter

Keiner

Beispiel

Die folgende Anweisung stellt fest, ob der größte zusammenhängende freie Speicherblock kleiner als 10 KB ist, und zeigt einen Warnhinweis an, wenn dies der Fall ist:

```
-- Lingo syntax
if (the freeBlock < (10 * 1024)) then alert "Not enough memory!"

// JavaScript syntax
if (freeBlock < (10 * 1024)) {
    alert("Not enough memory!")
}
```

Siehe auch

freeBytes(), memorySize, ramNeeded(), size

freeBytes()

Syntax

```
the freeBytes
```

Beschreibung
Diese Funktion zeigt die Gesamtmenge des freien Speichers in Byte an, wobei dieser Speicher nicht zusammenhängend sein muss. Ein Kilobyte (KB) entspricht 1024 Byte. Ein Megabyte (MB) entspricht 1024 Kilobyte.

Diese Funktion unterscheidet sich von `freeBlock` insofern, als dass sie angibt, wieviel freier Speicher insgesamt zur Verfügung steht, und sich dabei nicht auf den zusammenhängenden Speicher beschränkt.

Wenn Sie auf dem Mac die Option "Systemzwischenspeicher verwenden" in den allgemeinen Voreinstellungen von Director oder im Dialogfeld „Optionen“ eines Projektors auswählen, wird die Funktion `freeBytes` angewiesen, den gesamten freien und verfügbaren Speicher zurückzugeben. Dies entspricht der Menge des dem Programm zugeordneten Speichers, der im Dialogfenster „Information“ angezeigt wird, und dem Wert von „Größter freier Block“ im Dialogfenster „Über diesen Macintosh“.

Parameter
Keiner

Beispiel
Die folgende Anweisung prüft, ob mehr als 200KB Speicher verfügbar ist, und spielt in diesem Fall einen Farbfilm ab:

```
if (the freeBytes > (200 * 1024)) then play movie "colorMovie"
```

Siehe auch
`freeBlock()`, `memorySize`, `objectP()`, `ramNeeded()`, `size`

generateNormals()

Syntax
```
member(whichCastmember).modelResource(whichModelResource).generateNormals(style)
```

Beschreibung
Dieser 3D-Befehl für Modellressourcen vom Typ `#mesh` berechnet die `normal`en Vektoren aller Gitternetzscheitelpunkte.

Wenn der Parameter *style* auf `#flat` gesetzt ist, erhält jeder Scheitelpunkt für jede Fläche, zu der er gehört, eine Normale. Darüber hinaus besitzen alle drei Scheitelpunkte einer Fläche dieselbe Normale. Wenn beispielsweise alle Scheitelpunkte von `face[1]normal[1]` und alle Scheitelpunkte von `face[2]normal[2]` erhalten und die zwei Seiten `vertex[8]` gemeinsam benutzen, ist die Normale von `vertex[8]normal[1]` in `face[1]` und `normal[2]` in `face[2]`. Der Parameter `#flat` führt zu einer sehr klaren Trennung der Gitternetzflächen.

Wenn der Parameter *style* auf `#smooth` gesetzt ist, erhält jeder Scheitelpunkt nur eine Normale, unabhängig davon, wie vielen Flächen er angehört, und die drei Scheitelpunkte einer Fläche können unterschiedliche Normalen besitzen. Jede Scheitelpunktnormale ist der Durchschnitt der Flächennormalen aller Flächen, die den Scheitelpunkt gemeinsam benutzen. Die Verwendung des Parameters `#smooth` führt zu einem abgerundeteren Aussehen der Gitternetzflächen, außer an den Außenkanten der Seiten an der Gitternetzsilhouette, die auch weiterhin scharf sind.

Eine Scheitelpunktnormale ist ein Richtungsvektor, der die Vorwärtsrichtung eines Scheitelpunkts angibt. Wenn die Scheitelpunktnormale zur Kamera zeigt, werden die Farben, die im durch diese Normale gesteuerten Gitternetzbereich erscheinen, durch den Shader bestimmt. Wenn die Scheitelpunktnormale von der Kamera weg zeigt, ist der durch diese Normale gesteuerte Gitternetzbereich nicht sichtbar.

Nach Verwendung des Befehls `generateNormals()` müssen Sie das Gitternetz mit dem Befehl `build()` neu erstellen.

Parameter
style Erforderlich. Ein Symbol, das den Stil des Scheitelpunkts angibt.

Beispiel
Die folgende Anweisung berechnet Gitternetznormalen für die Modellressource `FloorMesh`. Der Parameter *style* wird auf `#smooth` gesetzt, sodass jeder Scheitelpunkt im `mesh` nur eine Normale erhält.

```
-- Lingo
member("Room").modelResource("FloorMesh").generateNormals(#smooth)

// Javascript
member("Room").getProp("modelResource", "index (of the
modelresource)").generateNormals("smooth");
```

Siehe auch
`build()`, `face[ ]`, `normalList`, `normals`, `flat`

getaProp

Syntax
```
propertyList.propertyName
getaProp(list, item)
list[listPosition]
propertyList [ #propertyName ]
propertyList [ "propertyName" ]
```

Beschreibung
Dieser Listenbefehl identifiziert bei linearen und Eigenschaftslisten den Wert, der mit dem in *item*, *listPosition* oder *propertyName* angegebenen Element in der in *list* angegebenen Liste verbunden ist.

- Wenn die Liste eine lineare Liste ist, ersetzen Sie *item* durch die Positionsnummer des Elements in der Liste, die durch *listPosition* angezeigt wird. Das Ergebnis ist der Wert an dieser Position.
- Wenn die Liste eine Eigenschaftsliste ist, ersetzen Sie *item* durch eine Eigenschaft in der Liste, wie z. B. *propertyName*. Das Ergebnis ist der Wert, der mit dieser Eigenschaft verbunden ist.

Der Befehl `getaProp` gibt `VOID` zurück, wenn der angegebene Wert nicht in der Liste enthalten ist.

Wird `getaProp` für lineare Listen verwendet, erfüllt er dieselbe Funktion wie der Befehl `getAt`.

Parameter
itemNameOrNum Erforderlich. Bei linearen Listen eine Ganzzahl, die die Indexposition des zurückzugebenden Werts in der Liste angibt, und bei Eigenschaftslisten ein Symbol (Lingo) oder eine Zeichenfolge (JavaScript-Syntax), das bzw. die die Eigenschaft angibt, deren Wert zurückgegeben wird.

Beispiel
Die folgende Anweisung kennzeichnet den Wert, der mit der Eigenschaft `#joe` in der Eigenschaftsliste „Alter“ verbunden ist, die aus [`#john:10, #joe:12, #cheryl:15, #barbara:22`] besteht:

```
put getaProp(ages, #joe)
```

Das Ergebnis ist 12, da dies der mit der Eigenschaft `#joe` verbundene Wert ist.

Das gleiche Ergebnis kann durch Listenzugriff mit eckigen Klammern auf diese Liste erzielt werden:

```
put ages[#joe]
```

Das Ergebnis ist wiederum 12.

Wenn Sie den Wert einer bestimmten Position in der Liste ermitteln möchten, können Sie ebenfalls den Listenzugriff mit eckigen Klammern nutzen. Verwenden Sie die folgende Syntax, um den dritten Wert in der Liste, der mit der dritten Eigenschaft verbunden ist, zu ermitteln:

```
put ages[3]
-- 15
```

Hinweis: *Im Gegensatz zum Befehl* `getAProp` *wird nicht* `VOID` *zurückgegeben, wenn eine Eigenschaft nicht vorhanden ist, sondern es tritt ein Skriptfehler auf, wenn der Zugriff mit eckigen Klammern verwendet wird und die Eigenschaft fehlt.*

Siehe auch

`getAt`, `getOne()`, `getProp()`, `setaProp`, `setAt`

getAt

Syntax

```
getAt(list, position)
list [position]
```

Beschreibung

Dieser Listenbefehl identifiziert das Element an einer vorgegebenen Position in einer angegebenen Liste. Wenn die Zahl der angegebenen Position größer ist als die Anzahl der in der Liste enthaltenen Elemente, tritt ein Skriptfehler auf.

Der Befehl `getAt` funktioniert bei linearen und Eigenschaftslisten. Er erfüllt dieselbe Funktion wie der Befehl `getaProp` für lineare Listen.

Dieser Befehl ist nützlich, wenn Sie eine Liste aus einer anderen Liste erstellen möchten, wie z. B. `deskTopRectList`.

Parameter

list Erforderlich. Gibt die Liste an, in der das Element vorhanden ist.

position Erforderlich. Gibt die Indexposition des Elements in der Liste an.

Beispiel

Die folgende Anweisung führt dazu, dass im Nachrichtenfenster das dritte Element in der Liste "Antworten" erscheint, die aus [10, 12, 15, 22] besteht:

```
put getAt(answers, 3)
-- 15
```

Dasselbe Ergebnis kann durch Listenzugriff erzielt werden:

```
put answers[3]
-- 15
```

Im folgenden Beispiel wird der erste Listeneintrag in einer Liste ermittelt, die aus zwei Einträgen besteht, welche den Namen, die Abteilung und die Kennnummer für Angestellte angeben. Anschließend wird das zweite Element der neu erstellten Liste zurückgegeben, das die Abteilung kennzeichnet, in der die erste Person in der Liste arbeitet. Das Format der Liste lautet [["Dennis", "Consulting", 510], ["Sherry", "Vertrieb", 973]] und die Liste heißt `employeeInfoList`.

```
firstPerson = getAt(employeeInfoList, 1)
put firstPerson
-- ["Dennis", "consulting", 510]
firstPersonDept = getAt(firstPerson, 2)
put firstPersonDept
-- "consulting"
```

Außerdem können `getAt`-Befehle verschachtelt werden, ohne Variablen in Zwischenschritten Werte zuzuordnen. Dieses Format lässt sich möglicherweise schwerer lesen und schreiben, ist aber prägnanter.

```
firstPersonDept = getAt(getAt(employeeInfoList, 1), 2)
put firstPersonDept
    -- "consulting"
```

Sie können auch den Listenzugriff durch eckige Klammern verwenden:

```
firstPerson = employeeInfoList[1]
put firstPerson
-- ["Dennis", "consulting", 510]
firstPersonDept = firstPerson[2]
put firstPersonDept
-- "consulting"
```

Genau wie bei `getAt` können eckige Klammern auch hier verschachtelt werden:

```
firstPersonDept = employeeInfoList[1][2]
```

Siehe auch

`getaProp`, `setaProp`, `setAt`

getCharSet

Syntax

```
FileIO.getCharSet()
```

Beschreibung

Diese FileIO Xtra-Methode gibt den aktiven `FileIO`-Zeichensatz zurück. Der Standardzeichensatz ist UTF-8.

Beispiele

```
--Lingo syntax
FileIO.getCharSet()

//JavaScript syntax
FileIO.getCharSet();
```

Siehe auch

`readChar()`, `readFile()`, `readLine()`, `readToken()`, `readWord()`, `writeChar()`, `writeString()`

getError() (Flash, SWA)

Syntax

```
-- Lingo syntax
memberObjRef.getError()

// JavaScript syntax
memberObjRef.getError();
```

Beschreibung

Diese Funktion für Shockwave Audio- (SWA) und Flash-Darsteller gibt an, ob ein Fehler beim Streaming des Darstellers in den Speicher aufgetreten ist, und gibt einen Wert zurück.

Mögliche ganzzahlige `getError()`-Werte und zugehörige `getErrorString()`-Nachrichten für Shockwave Audio-Darsteller:

`getError()`**-Wert**	`getErrorString()`**-Meldung**
0	OK
1	Arbeitsspeicher
2	network
3	playback device
99	other

Mögliche `getError`-Werte für Flash-Film-Darsteller:

- `FALSE` – Kein Fehler aufgetreten.
- `#memory` – Es steht nicht genügend Speicherplatz zum Laden des Darstellers zur Verfügung.
- `#fileNotFound` – Die Datei mit den Elementen des Darstellers wurde nicht gefunden.
- `#network` – Der Darsteller konnte aufgrund eines Netzwerkfehlers nicht geladen werden.
- `#fileFormat` – Die Datei wurde gefunden, scheint aber der falsche Typ zu sein, oder beim Lesen der Datei trat ein Fehler auf.
- `#other` – Anderer Fehler.

Wenn beim Laden eines Darstellers in den Speicher ein Fehler auftritt, setzt Director die Eigenschaft des Darstellers auf -1. Anhand der Funktion `getError` können Sie die Art des aufgetretenen Fehlers ermitteln.

Parameter

Keiner

Beispiel

Die folgende Prozedur stellt mit der Funktion `getError` fest, ob ein Fehler in Verbindung mit dem `Shockwave Audio`-Darsteller `Norma Desmond Speaks` aufgetreten ist. Wenn dies der Fall ist, wird der entsprechende Fehlerstring in einem Feld angezeigt:

```
-- Lingo syntax
on exitFrame
    if member("Norma Desmond Speaks").getError() <> 0 then
        member("Display Error Name").text = member("Norma Desmond \ Speaks").getErrorString()
    end if
end

// JavaScript syntax
function exitFrame() {
var memNor = member("Norma Desmond Speaks").getError();
    if (memNor != 0) {
        member("Display Error Name").text = member("Norma Desmond Speaks").getErrorString();
    }
}
```

Die folgende Prozedur prüft, ob beim Streaming des Flash-Darstellers `Dali` in den Speicher ein Fehler aufgetreten ist. Handelt es sich um einen Speicherfehler, versucht das Skript mit dem Befehl `unloadCast`, etwas Speicherplatz freizugeben, und schickt dann den Abspielkopf zum Bild `Artists` im Director-Film, in dem das Flash-Film-Sprite erstmalig erscheint, damit Director erneut versuchen kann, den Flash-Film zu laden und abzuspielen. Wenn ein Fehler anderer Art auftrat, geht das Skript zu einem Bild namens "Sorry“, in dem erklärt wird, dass der angeforderte Flash-Film nicht abgespielt werden kann.

```
-- Lingo syntax
on CheckFlashStatus
errorCheck = member("Dali").getError()
    if errorCheck <> 0 then
        if errorCheck = #memory then
            member("Dali").clearError()
            unloadCast()
            _movie.go("Artists")
        else
            _movie.go("Sorry")
        end if
    end if
end

// JavaScript syntax
function CheckFlashStatus() {
var errorCheck = member("Dali").getError();
    if (errorCheck != 0) {
        if (errorCheck == "memory") {
            member("Dali").clearError();
            unloadCast();
            _movie.go("Artists");
        } else {
            _movie.go("Sorry");
        }
    }
}
```

Siehe auch

clearError(), getErrorString(), state (Flash, SWA)

getError() (XML)

Syntax

```
parserObject.getError()
```

Beschreibung

Diese Funktion gibt einen String mit der Fehlerbeschreibung zur jeweiligen Fehlernummer zurück (einschließlich Nummer der Zeile und Spalte, in der der Fehler auftrat). Wenn kein Fehler vorliegt, gibt die Funktion `<VOID>` zurück.

Parameter

Keiner

Beispiel

Die folgenden Anweisungen prüfen, ob ein Fehler vorliegt, nachdem ein String mit XML-Daten geparst wurde:

```
-- Lingo
errCode = parserObject.parseString(member("XMLtext").text)
errorString = parserObject.getError()
if voidP(errorString) then
    -- Go ahead and use the XML in some way
else
    alert "Sorry, there was an error " & errorString
    -- Exit from the handler
    exit
end if

// Javascript
errCode = parserObject.parseString(member("XMLtext").text);
errorString = parserObject.getError();
if (errorString != null)
{
    // Go ahead and use the XML in some way
}else{
    _player.alert("Sorry, there was an error " + errorString);
    // Exit from the handler
}
```

getErrorString()

Syntax

```
-- Lingo syntax
memberObjRef.getErrorString()

// JavaScript syntax
memberObjRef.getErrorString();
```

Beschreibung

Diese Funktion für Shockwave Audio- (SWA) Darsteller gibt einen String mit einer Fehlermeldung zurück, die dem durch die Funktion `getError()` zurückgegebenen Fehlerwert entspricht.

Mögliche ganzzahlige `getError()`-Werte und zugehörige `getErrorString()`-Nachrichten:

`getError()`-Wert	`getErrorString()`-Meldung
0	OK
1	Arbeitsspeicher
2	network
3	playback device
99	other

Parameter
Keiner

Beispiel
Die folgende Prozedur stellt anhand von `getError()` fest, ob ein Fehler in Verbindung mit dem Shockwave Audio-Darsteller `Norma Desmond Speaks` aufgetreten ist. Wenn dies der Fall ist, wird mit `getErrorString` die Fehlermeldung abgerufen und einem Felddarsteller zugeordnet:

```
-- Lingo syntax
on exitFrame
    if member("Norma Desmond Speaks").getError() <> 0 then
        member("Display Error Name").text = member("Norma Desmond Speaks").getErrorString()
    end if
end

// JavaScript syntax
function exitFrame() {
    var memNor = member("Norma Desmond Speaks").getError();
    if (memNor != 0) {
        member("Display Error Name").text = member("Norma Desmond Speaks").getErrorString();
    }
}
```

Siehe auch
`getError() (Flash, SWA)`

getFinderInfo()

Syntax

```
-- Lingo syntax
fileioObjRef.getFinderInfo()

// JavaScript syntax
fileioObjRef.getFinderInfo();
```

Beschreibung
Diese Datei-E/A-Methode (Fileio; nur Mac) gibt die Finder-Informationen für eine offene Datei zurück.

Sie müssen zuerst durch Aufrufen von `openFile()` eine Datei öffnen, bevor Sie mithilfe von `getFinderInfo()` die Finder-Informationen der Datei zurückgeben.

Parameter
Keiner

Beispiel

```
-- Lingo syntax
objFileio = new xtra("fileio")
objFileio.openFile(stringFileName, intMode)
objFileio.getFinderInfo()

// JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.openFile(stringFileName, intMode) ;
objFileio.getFinderInfo() ;
```

Siehe auch
`Fileio`, `openFile()`

getFlashProperty()

Syntax

```
-- Lingo syntax
spriteObjRef.getFlashProperty(targetName, symProp)

// JavaScript syntax
spriteObjRef.getFlashProperty(targetName, symProp);
```

Beschreibung
Diese Funktion ermöglicht Lingo, die Flash-ActionScript-Funktion `getProperty()` für das angegebene Flash-Sprite aufzurufen. Diese Flash-ActionScript-Funktion dient zum Abrufen von Eigenschaftswerten für einen Movieclip oder eine Ebene in einem Flash-Film. Dies ist vergleichbar mit dem Testen von Sprite-Eigenschaften in Director.

Wenn Sie eine globale Eigenschaft des Flash-Sprites abrufen möchten, übergeben Sie einen leeren String als *targetName*. Folgende globale Flash-Eigenschaften lassen sich testen: `#focusRect` und `#spriteSoundBufferTime`.

Eine Beschreibung dieser Eigenschaften finden Sie in der Flash-Dokumentation.

Parameter
targetName Erforderlich. Eine Zeichenfolge, die den Namen des Movieclips oder der Ebene mit der Eigenschaft, die Sie im angegebenen Flash-Sprite abrufen möchten, angibt.

symProp Erforderlich. Ein Symbol, das den Namen der abzurufenden Eigenschaft angibt. Valid values include: `#posX`, `#posY`, `#scaleX`, `#scaleY`, `#visible`, `#rotate`, `#alpha`, `#name`, `#width`, `#height`, `#target`, `#url`, `#dropTarget`, `#totalFrames`, `#currentFrame`, `#cursor` und `#lastframeLoaded`.

Beispiel
Die folgende Anweisung ruft den Wert der Eigenschaft `#rotate` des Movieclips `Star` im Flash-Darsteller in `sprite 3` ab:

```
-- Lingo syntax
sprite(3).setFlashProperty("Star", #rotate)
sprite(3).getFlashProperty("Star")

// JavaScript syntax
sprite(3).setFlashProperty("Star", symbol("rotate"));
sprite(3).getFlashProperty("Star");
```

getFrameLabel()

Syntax

```
sprite(whichFlashSprite).getFrameLabel(whichFlashFrameNumber)
getFrameLabel(sprite whichFlashSprite, whichFlashFrameNumber)
```

Beschreibung

Diese Funktion gibt die Bildbeschriftung in einem Flash-Film zurück, die mit der angeforderten Bildnummer verbunden ist. Wenn die Beschriftung nicht existiert oder dieser Teil des Flash-Films noch nicht heruntergeladen wurde, gibt diese Funktion einen leeren String zurück.

Parameter

whichFlashFrameNumber Erforderlich. Gibt die Bildnummer an, die der Grafikbeschriftung zugeordnet ist.

Beispiel

Die folgende Prozedur stellt fest, ob die Markierung in Bild 15 des in Sprite 1 laufenden Flash-Films "Löwen“ heißt. Wenn ja, springt der Director-Film zum Bild "Löwen". Wenn nein, verbleibt der Director-Film im aktuellen Bild, und die Wiedergabe des Flash-Films wird fortgesetzt.

```
-- Lingo syntax
on exitFrame
    if sprite(1).getFrameLabel(15) = "Lions" then
        go "Lions"
    else
        go the frame
    end if
end

// JavaScript syntax
function exitFrame() {
    if (sprite(1).getFrameLabel(15) == "Lions") {
        _movie.go("Lions");
    } else {
        _movie.go(_movie.frame);
    }
}
```

getHardwareInfo()

Syntax

```
getRendererServices().getHardwareInfo()
```

Beschreibung

Diese 3D-Methode für `rendererServices` gibt eine Eigenschaftsliste mit Informationen über die Videokarte des Benutzers zurück. Die Liste enthält die folgenden Eigenschaften:

`#present` ist ein Boolescher Wert, der angibt, ob im Computer Hardware zur Videobeschleunigung installiert ist.

`#vendor` gibt den Namen des Videokartenherstellers an.

`#model` gibt das Videokartenmodell an.

`#version` gibt die Version des Videotreibers an.

`#maxTextureSize` ist eine lineare Liste mit der maximalen Breite und Höhe einer Textur (in Pixel). Texturen, die diese Abmessungen überschreiten, werden per Downsampling entsprechend verkleinert. Um auf das Sampling zurückzuführende Artefakte zu vermeiden, sollten Sie Texturen unterschiedlicher Größe erstellen und zur Laufzeit die verwenden, die den `#maxTextureSize`-Wert nicht überschreiten.

`#supportedTextureRenderFormats` ist eine lineare Liste mit den von der Videokarte unterstützten Texturpixelformaten. Einzelheiten finden Sie unter `textureRenderFormat`.

`#textureUnits` gibt die Anzahl von Textureinheiten an, die der Karte zur Verfügung stehen.

`#depthBufferRange` ist eine lineare Liste der Bittiefenauflösungen, auf die die Eigenschaft `depthBufferDepth` gesetzt werden kann.

`#colorBufferRange` ist eine lineare Liste der Bittiefenauflösungen, auf die die Eigenschaft `colorBufferDepth` gesetzt werden kann.

Beispiel

Die folgende Anweisung zeigt eine detaillierte Eigenschaftsliste mit Informationen über die Hardware des Benutzers an.

```
-- Lingo
put getRendererServices().getHardwareInfo()
-- [#present: 1, #vendor: "NVIDIA Corporation", #model: "32MB DDR NVIDIA GeForce2 GTS (Dell)",
#version: "4.12.01.0532", #maxTextureSize: [2048, 2048], #supportedTextureRenderFormats:
[#rgba8888, #rgba8880, #rgba5650, #rgba5551, #rgba5550, #rgba4444], #textureUnits: 2,
#depthBufferRange: [16, 24], #colorBufferRange: [16, 32]]

// Javascript
trace(getRendererServices().getHardwareInfo())
//<[#present: 1, #vendor: "NVIDIA Corporation", #model: "32MB DDR NVIDIA GeForce2 GTS (Dell)",
#version: "4.12.01.0532", #maxTextureSize: [2048, 2048], #supportedTextureRenderFormats:
[#rgba8888, #rgba8880, #rgba5650, #rgba5551, #rgba5550, #rgba4444], #textureUnits: 2,
#depthBufferRange: [16, 24], #colorBufferRange: [16, 32]]>
```

Siehe auch

`getRendererServices()`

getHotSpotRect()

Syntax

```
-- Lingo syntax
spriteObjRef.getHotSpotRect(hotSpotID)

// JavaScript syntax
spriteObjRef.getHotSpotRect(hotSpotID);
```

Beschreibung

Diese QuickTime VR-Funktion gibt ein ungefähres Begrenzungsrechteck eines Hotspots zurück. Wenn der Hotspot nicht existiert oder auf der Bühne nicht sichtbar ist, gibt diese Funktion „rect(0, 0, 0, 0)“ zurück. Wenn der Hotspot zum Teil sichtbar ist, gibt diese Funktion das Begrenzungsrechteck für den sichtbaren Teil zurück.

Parameter

hotSpotID Erforderlich. Gibt den Hotspot an, für den ein Begrenzungsrechteck zurückgegeben wird.

getInstalledCharSets

Syntax

```
_system.getInstalledCharSets()
```

Beschreibung

Diese Zeichensatzmethode gibt die Liste der auf dem Computer Installierten Zeichensätze zurück.

Beispiele

```
--Lingo syntax
put _system.getInstalledCharSets()

//JavaScript syntax
put(_system.getInstalledCharSets());
```

GetItemPropList

Syntax

```
GetItemPropList(MUIObject)
```

Beschreibung

Diese Funktion gibt eine Liste der vordefinierten Eigenschaften des Xtras MUI für Komponenten in einem allgemeinen Dialogfeld zurück und eignet sich zum Definieren neuer Komponenten in einem solchen Dialogfeld. Rufen Sie über die Funktion `GetItemPropList` eine umfassende Liste mit Eigenschaften und Werten ab, und bearbeiten Sie anschließend einzelne Eigenschaften den Anforderungen entsprechend.

Es folgt die Liste der Eigenschaften und Werte:

Eigenschaft	Standardwert
#value	0
#type	#checkBox
#attributes	[],
#title	"title"
#tip	"tip" (reserviert für eine mögliche Nutzung in späteren Versionen des Xtras MUI)
#locH	20
#locV	24
#width	200
#height	210
#enabled	1

Beispiel
Diese Anweisungen definieren den Anfang eines Dialogfeldfensters.

- Die erste Anweisung erzeugt eine Instanz des Xtras MUI, bei der es sich um das Objekt handelt, das als Dialogfeld verwendet wird.
- Die zweite Anweisung weist eine Liste mit Standardeinstellungen von Dialogfeldkomponenten der Variablen `tempItemProps` zu.
- Die dritte Anweisung macht die Komponente zum Anfang des Dialogfelds, indem deren Wert in `#windowBegin` geändert wird.

```
--Lingo syntax
set aMuiObj = new (Xtra "MUI")
set tempItemProps = GetItemPropList(aMuiObj)
set the type of tempItemProps = #windowBegin
```

getLast()

Syntax
```
list.getLast()
getLast(list)
```

Beschreibung
Diese Listenfunktion ermittelt den letzten Wert in einer durch *list* angegebenen linearen oder Eigenschaftsliste.

Parameter
Keiner

Beispiel
Die folgende Anweisung ermittelt das letzte Element, 22, in der Liste "Antworten", die aus [10, 12, 15, 22] besteht:

```
put Answers.getLast()
```

Die folgende Anweisung ermittelt das letzte Element, 850, in der Liste „Gebote“, die aus [#Gee:750, #Kayne:600, #Ohashi:850] besteht:

```
put Bids.getLast()
```

getLatestNetID

Syntax

```
getLatestNetID
```

Beschreibung

Diese Funktion gibt ein Kennzeichen für den zuletzt gestarteten Netzwerkvorgang zurück.

Das von `getLatestNetID` zurückgegebene Kennzeichen kann in den Funktionen `netDone, netError` und `netAbort` verwendet werden, um den zuletzt ausgeführten Netzwerkvorgang zu identifizieren.

Hinweis: *Diese Funktion wird aus Gründen der Rückwärtskompatibilität unterstützt. Es wird empfohlen, an Stelle von `getLatestNetID` die von einer Netzwerk-Lingo-Funktion zurückgegebene Netzwerk-ID zu verwenden. Wenn Sie jedoch `getLatestNetID` verwenden, sollten Sie diese Funktion unmittelbar nach dem Befehl `netLingo` aufrufen.*

Parameter

Keiner

Beispiel

Das folgende Skript weist dem Felddarsteller "Ergebnis" die Netzwerk-ID einer `getNetText`-Operation zu, damit die Ergebnisse dieser Operation später abgerufen werden können.

```
on startOperation
    global gNetID
    getNetText("url")
    set gNetID = getLatestNetID()
end
on checkOperation
    global gNetID
    if netDone(gNetID) then
        put netTextResult into member "Result"
    end if
end
```

Siehe auch

`netAbort`, `netDone()`, `netError()`

getLength()

Syntax

```
-- Lingo syntax
fileioObjRef.getLength()

// JavaScript syntax
fileioObjRef.getLength();
```

Beschreibung

Diese Datei-E/A-Methode (Fileio) gibt die Länge einer geöffneten Datei zurück.

Sie müssen zuerst durch Aufrufen von `openFile()` eine Datei öffnen, bevor Sie mithilfe von `getLength()` die Länge der Datei zurückgeben.

Parameter

Keiner

Beispiel

Das folgende Beispiel öffnet eine Datei namens „c:\check.txt“ und ruft deren Länge ab.

```
-- Lingo syntax
objFileio = new xtra("fileio")
objFileio.openFile("c:\check.txt",2)
put objFileio.getLength()

// JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.openFile("c:\check.txt",2);
trace(objFileio.getLength())
```

Siehe auch

`Fileio`, `openFile()`

getNetByteArray

Syntax

```
getNetByteArray(URL, [propertyList], [serverOSString])
```

Beschreibung

Diese Net Lingo-Methode lädt eine Webseite als Bytearray zurück. Diese Methode ist ähnlich wie `getNetText`.

Parameter

Parameter	Beschreibung	Erforderlich/Optional
URL	Die URL zur Datei, die das erforderliche Byte-Array enthält.	Erforderlich
propertyList	Gibt eine Eigenschaftsliste an, die für CGI-Abfragen verwendet wird.	Optional
serverOSString	Gibt die Kodierung der zurückgelieferten Zeichen in der Eigenschaftenliste `propertyList` zurück.	Optional

Beispiele

```
--Lingo syntax
id = getNetByteArray("http://www.bytearray.com/byte.cgi")

//JavaScript syntax
id = getNetByteArray("http://www.bytearray.com/byte.cgi");
```

getNetText()

Syntax

```
getNetText(URL {, serverOSString} {, characterSet})
getNetText(URL, propertyList {, serverOSString} {, characterSet})
```

Beschreibung

Diese Funktion beginnt, Text aus einer in der Regel auf einem HTTP- oder FTP-Server gespeicherten Datei abzurufen, oder leitet eine CGI-Abfrage ein.

Die erste angegebene Syntax startet den Textabruf. Sie können auf diese Weise HTTP-CGI-Abfragen übermitteln, die in der URL ordnungsgemäß kodiert sein müssen. Die zweite Syntax beinhaltet eine Eigenschaftsliste und übermittelt eine CGI-Abfrage in der korrekten URL-Kodierung.

Die Eigenschaftsliste wird mit dem optionalen Parameter *propertyList* in eine CGI-Abfrage übernommen. Die Eigenschaftsliste ist URL-kodiert und die gesendete URL lautet `(urlstring & "?" & encodedproplist)`.

Verwenden Sie den optionalen Parameter *serverOSString*, um die Return-Zeichen in *propertylist* zu kodieren. Der Standardwert lautet `UNIX`, kann jedoch auf `Win` oder `Mac` gesetzt werden. Er bewirkt, dass alle Zeilenschaltungen im Argument *propertylist* in die auf dem Server verwendeten Steuerzeichen umgewandelt werden. Bei den meisten Anwendungen ist diese Einstellung nicht erforderlich, da in Formularantworten normalerweise keine Zeilenumbrüche verwendet werden.

Der optionale Parameter *characterSet* wird nur benötigt, wenn der Benutzer Director auf einem `shiftJIS`-System (Japanisch) einsetzt. Die möglichen Zeichensatzeinstellungen sind `JIS`, `EUC`, `ASCII` und `AUTO`. Lingo konvertiert die abgerufenen Daten aus `Shift-JIS` in den angegebenen Zeichensatz. Bei der Einstellung „AUTO“ versucht Director zu bestimmen, welcher Zeichensatz für den abgerufenen Text verwendet wurde, und übersetzt diesen dann in den Zeichensatz des lokalen Computers. Die Standardeinstellung ist ASCII.

Ermitteln Sie unter Verwendung von `netDone`, wann der `getNetText`-Vorgang abgeschlossen ist, und stellen Sie mit `netError` fest, ob er erfolgreich war. Verwenden Sie `netTextResult`, um den mit `getNetText` abgerufenen Text anzuzeigen.

Die Funktion kann mit relativen URLs eingesetzt werden.

Ein Beispiel für `getNetText()` in einem fertigen Film finden Sie im Film „Forms and Post“ im Ordner „Learning\\Lingo_“ im Director-Anwendungsordner.

Parameter

URL Erforderlich. Die URL zu der Datei, die den abzurufenden Text enthält.

propertyList Optional. Gibt eine Eigenschaftsliste an, die für CGI-Abfragen verwendet wird.

serverOSString Optional. Gibt die Kodierung von Return-Zeichen (Zeilenumbrüchen) in der *propertyList* an.

characterSet Optional. Gibt Zeicheneinstellungen an

Beispiel

Das folgende Skript ruft Text von der URL "http://BigServer.com/sample.txt" ab und aktualisiert den Felddarsteller, auf den mit der Maus geklickt wird:

```
property spriteNum

property theNetID

on mouseUp me
    theNetID = getNetText ("http://BigServer.com/sample.txt")
end

on exitFrame me
    if netDone(theNetID) then
        sprite(spriteNum).member.text = netTextResult(theNetID)
    end if
end
```

Die folgende Anweisung ruft das Ergebnis einer CGI-Abfrage ab:

```
getNetText("http://www.yourserver.com/cgi-bin/query.cgi?name=Bill")
```

Die folgende Anweisung hat dieselbe Wirkung wie die im vorhergehenden Beispiel, übergibt die CGI-Abfrage jedoch in einer Eigenschaftsliste und führt die URL-Kodierung automatisch durch.

```
getNetText("http://www.yourserver.com/cgi-bin/query.cgi", [#name:"Bill"])
```

Siehe auch

`netDone()`, `netError()`, `netTextResult()`

getNormalized

Syntax

```
getNormalized(vector)
vector.getNormalized()
```

Beschreibung

Diese 3D-Vektormethode kopiert den Vektor und dividiert die x,- y- und z-Komponenten der Kopie durch die Länge des ursprünglichen Vektors. Der daraus resultierende Vektor weist eine Länge von 1 Welteinheit auf.

Diese Methode gibt die Kopie zurück und lässt den ursprünglichen Vektor unverändert. Mit dem Befehl `normalize` können Sie den ursprünglichen Vektor normieren.

Beispiel

Die folgende Anweisung speichert den normierten Wert des Vektors `MyVec` in der Variablen `Norm`. Der Wert von `Norm` ist vector (-0.1199, 0.9928, 0.0000), der Betrag von `Norm` ist 1.

```
-- Lingo
MyVec = vector(-209.9019, 1737.5126, 0.0000)
Norm = MyVec.getNormalized()
put Norm
-- vector( -0.1199, 0.9928, 0.0000 )
put Norm.magnitude
-- 1.0000

// Javascript
MyVec = vector(-209.9019, 1737.5126, 0.0000);
Norm = MyVec.getNormalized();
trace(Norm);
// vector( -0.1199, 0.9928, 0.0000 )
trace(Norm.magnitude);
// 1.0000
```

Siehe auch

`normalize`

getNthFileNameInFolder()

Syntax

```
getNthFileNameInFolder(folderPath, fileNumber)
```

Beschreibung

Diese Filmmethode gibt einen Dateinamen aus dem Verzeichnisordner auf der Grundlage des angegebenen Pfadnamens und der Dateinummer im Ordner zurück. Damit Director-Filme mit der Funktion `getNthFileNameInFolder` gefunden werden können, müssen sie in der Ordnerstruktur sichtbar gemacht werden. (Auf dem Mac werden andere Dateitypen gefunden, ganz gleich, ob sie sichtbar sind oder nicht.) Gibt diese Funktion einen leeren String zurück, haben Sie eine Zahl angegeben, die größer ist als die Anzahl der Dateien in diesem Ordner.

Die Funktion `getNthFileNameInFolder` kann nicht bei URLs eingesetzt werden.

Zur Angabe anderer Ordnernamen verwenden Sie den Operator `@ pathname` oder den vollständigen Pfadnamen im Format der Plattform, auf der der Film abgespielt wird. Beispiel:

- Unter Windows verwenden Sie einen Verzeichnispfad im Format "C:\\Director\\Filme".
- Auf dem Mac verwenden Sie einen Pfadnamen im Format "Festplatte:Director:Filme".Wenn Sie Dateien auf dem Mac-Desktop suchen, verwenden Sie den Pfad „Festplatte:Desktop Folder“.
- Diese Funktion ist im Shockwave Player nicht verfügbar.

Parameter

folderPath Erforderlich. Gibt den Pfad zu dem Ordner an, der die Datei enthält.

fileNumber Erforderlich. Gibt die Indexposition der Datei in dem Ordner an.

Beispiel

Die folgende Prozedur gibt eine Liste mit Dateinamen im Ordner des aktuellen Pfads zurück. ZumAufruf der Funktion verwenden Sie Klammern, wie in diesem Beispiel: put`currentFolder()`.

```
-- Lingo
on currentFolder
fileList = []
repeat with i = 1 to 100
        n=getNthFileNameInFolder(the moviePath, i)
        if n = EMPTY then exit repeat
        fileList.append(n)
end repeat
return fileList
end currentFolder

// Javascript
function currentFolder()
{
    fileList = list();
    var i=1;
    while(i<100)
    {
        n = _movie.getNthFileNameInFolder(_movie.path,i);
                if (n=="")
        {
            i=101;
        }
        else
        {
            fileList.append(n);
        }
        i++;
    }
    return fileList;
}
```

Siehe auch

@ (Pfadname), Movie

getOne()

Syntax

```
list.getOne(value)
getOne(list, value)
```

Beschreibung

Diese Listenfunktion identifiziert die Position (lineare Liste) oder Eigenschaft (Eigenschaftsliste), die einem Wert in einer Liste zugeordnet ist.

Ist ein Wert mehrmals in der Liste vorhanden, wird nur der erste aufgefundene Wert angezeigt. Der Befehl `getOne` gibt das Ergebnis 0 zurück, wenn der angegebene Wert nicht in der Liste enthalten ist.

Bei linearen Listen hat der Befehl `getOne` dieselbe Funktion wie der Befehl `getPos`.

Wenn Sie mithilfe der `add` oder `append`-Methode der Liste `filterlist` einen Filter hinzufügen, wird ein Duplikat erstellt und der Liste hinzugefügt. Methoden wie `deleteOne`, `getPos`, `findPos` und `getOne` verwenden den genauen Wert in der Liste und nicht den Duplikatwert.

In diesem Fall kann die `getOne`-Methode wie folgt verwendet werden:

```
f = filter(#glowfilter)
sprite(1).filterlist.append(f)
f = sprite(1).filterlist[1]-- here we get the actual value added to the list.
sprite(1).filterlist.getOne(f)
```

Die dritte Zeile in dem vorstehenden Skript fügt der Liste den Verweis des Filterwerts hinzu.

Parameter

value Erforderlich. Gibt den der Position oder Eigenschaft zugeordneten Wert an.

Beispiel

Die folgende Anweisung gibt die Position des Wertes "12" in der linearen Liste "Antworten" an, die aus [10, 12, 15, 22] besteht:

```
-- Lingo
put Answers.getOne(12)

// Javascript
trace(Answers.getOne(12));
```

Das Ergebnis ist 2, da 12 der zweite Wert in der Liste ist.

Die folgende Anweisung gibt die Eigenschaft an, die mit dem Wert „12“ in der Eigenschaftsliste `Answers` verbunden ist, die aus [`#a:10, #b:12, #c:15, #d:22`] besteht:

```
-- Lingo
put Answers.getOne(12)

// Javascript
trace(Answers.getOne(12));
```

Das Ergebnis ist `#b`, da dies die mit dem Wert „12“ verbundene Eigenschaft ist.

Siehe auch

`getPos()`

getOSDirectory()

Syntax

```
-- Lingo syntax
getOSDirectory()

// JavaScript syntax
getOSDirectory();
```

Beschreibung
Diese Funktion gibt den vollständigen Pfad zum Systemordner (Mac) oder dem Windows-Verzeichnis (Windows) zurück.

Parameter
Keiner

Beispiel
Die folgende Anweisung zeigt das Betriebssystemverzeichnis des Computers an („put“). In diesem Fall ist es „c:\windows“.

```
-- Lingo
Put getOSDirectory()

// Javascript
trace(getOSDirectory());
```

Siehe auch
`Fileio`

getPixel()

Syntax

```
-- Lingo syntax
imageObjRef.getPixel(x, y {, #integer})
imageObjRef.getPixel(point(x, y) {, #integer})

// JavaScript syntax
imageObjRef.getPixel(x, y {, #integer});
imageObjRef.getPixel(point(x, y) {, #integer});
```

Beschreibung
Diese Grafikmethode gibt eine indizierte oder RGB-Farbe des Pixels an der angegebenen Position in einer vorgegebenen Grafik zurück.

Der Index der Zeilen und Spalten der zurückgegebenen Grafik beginnt bei 0. Deshalb muss, um auf das oberste linke Pixel einer Grafik zuzugreifen, die Position als `(0,0)` und nicht als `(1,1)` angegeben werden. Ist eine vorgegebene Grafik *h* Pixel hoch und *w* Pixel breit, müssen Sie, um auf das unterste rechte Pixel der Grafik zuzugreifen, die Position als `(`*w*`,1)`, `(`*h*`,1)` angeben.

Diese Methode gibt einen Wert von 0 zurück, wenn sich das angegebene Pixel außerhalb der angegebenen Grafik befindet.

Um zahlreiche Pixel auf die Farbe eines anderen Pixels zu setzen, sollten Sie aus Zeitgründen besser reine Zahlen verwenden (unter Verwendung des optionalen Parameters `#integer`). Reine ganzzahlige Farbwerte haben außerdem den Vorteil, dass sie bei 32-Bit-Grafiken neben den Farbinformationen auch Alphakanalinformationen enthalten. Die Alphakanalinformationen lassen sich durch Dividieren des ganzzahligen Werts durch 2^8+8+8 aus der reinen Ganzzahl extrahieren.

Parameter

x Erforderlich beim Angeben eines Pixels mithilfe von `x`- und `y`-Koordinaten. Eine Ganzzahl, die die `x`-Koordinate des Pixels angibt.

Y Erforderlich beim Angeben eines Pixels mithilfe von `x`- und `y`-Koordinaten. Eine Ganzzahl, die die `y`-Koordinate des Pixels angibt.

#integer Optional. Ein Symbol, das die reine Zahl des zurückgegebenen Farbwerts angibt.

`point`*(x, y)* Erforderlich beim Angeben eines Pixels mithilfe eines Positionspunkts. Ein Punkt, der die Position des Pixels angibt.

Beispiel

Die folgende Anweisung ruft die Farbe des Pixels an der Position (20, 20) im Darsteller „Grafik“ auf der Bühne ab.

```
-- Lingo
objImage = _movie.stage.image
objImage.getPixel(20, 20)
put (objImage)

-- Javascript
var objImage = _movie.stage.image ;
objImage.getPixel(20, 20) ;
put (objImage) ;
```

Siehe auch

`color()`, `image()`, `power()`, `setPixel()`

getPixels()

Syntax

```
Bytearray image.getPixels(#symbolImageFormat)
Possible values form symImageFormat are:
#bgra8888 => 32 bit BGRA
#argb8888=> 32 bit ARGB
#rgba8888=> 32 bit RGBA
#rgb888=> 24bit RGB
#bgr888=> 24bit BGR
```

Beschreibung

Diese Byte-Array-Methode erzeugt ein Byte-Array, das mit Bilddaten initialisiert wird.

Hinweis: *Nur 32-Bit Lingo-Bildobjekte werden unterstützt.*

Beispiel

Mit dem folgenden Code-Abschnitt wird ein Bild mit roter Farbe gefüllt und der Alphawert auf 50 eingestellt.

```
--Lingo syntax
on mouseUp me
i32=image(128,128,32) -- Creating a 32 bit image
i32.useAlpha=1 -- Setting the Alpha to 1
ba=i32.getPixels(#RGBA8888) -- Getting the pixels of the image using 32 bit RGBA format
ba2=byteArray(128*128*4,0) -- Creating the bytearray
i=0
repeat with r=1 to 128
repeat with c=1 to 128
ba2[i+1]=255 -- Value for RED
ba2[i+2]=0 -- Value for GREEN
ba2[i+3]=0 -- Value for BLUE
ba2[i+4]=50 -- Value for Alpha
i=i+4
end repeat
end repeat
i32.setPixels(ba2,#RGBA8888) -- Setting the pixels of the image using 32 bit RGBA format
member("test").image=i32
end
//JavaScript Syntax
function mouseUp(me)
{
var i=0;
i32=image(128,128,32);
i32.useAlpha=1;
ba=i32.getPixels(symbol("RGBA8888"));
ba2=byteArray(128*128*4,0);
i=0;
for(r=1;r<=128;r++)
{
for(c=1;c<=128;c++)
{
ba2[i+1]=255;
ba2[i+2]=0;
ba2[i+3]=0;
ba2[i+4]=50;
i=i+4;
}
}
i32.setPixels(ba2,symbol("RGBA8888"));
member("test").image=i32;
}
```

Siehe auch

setPixels()

getPlayList()

Syntax

```
-- Lingo syntax
soundChannelObjRef.getPlayList()

// JavaScript syntax
soundChannelObjRef.getPlayList();
```

Beschreibung

Diese Soundkanalmethode gibt eine Kopie der Liste von Sounds zurück, die sich in der Warteschlange für einen Soundkanal befinden.

Der aktuelle Sound ist weder in der zurückgegebenen Liste aufgeführt, noch kann er direkt bearbeitet werden. Verwenden Sie zum Ändern der Liste die Funktion `setPlayList()`.

Die Abspielliste ist eine lineare Liste von Eigenschaftslisten. Jede Eigenschaftsliste entspricht einem Sounddarsteller in der Warteschlange. Zu jedem wartenden Sound können die folgenden Eigenschaften angegeben sein:

Eigenschaft	Beschreibung
`#member`	Der in die Warteschlange einzureihende Sounddarsteller. Diese Eigenschaft ist erforderlich; alle anderen sind optional.
`#startTime`	Der Zeitpunkt im Sound, zu dem die Wiedergabe beginnt, in Millisekunden. Standardmäßig ist dies der Anfangspunkt des Sounds. Weitere Informationen finden Sie unter `startTime`.
`#endTime`	Der Zeitpunkt im Sound, zu dem die Wiedergabe endet, in Millisekunden. Standardmäßig ist dies das Ende des Sounds. Weitere Informationen finden Sie unter`endTime`.
`#loopCount`	Angabe, wie oft eine Schleife abgespielt werden soll, die in `#loopStartTime` und `#loopEndTime` definiert ist. DerStandardwert ist1. Weitere Informationen finden Sie unter `loopCount`.
`#loopStartTime`	Der Zeitpunkt im Sound, zu dem die Schleifenwiedergabe beginnt, in Millisekunden. Weitere Informationen finden Sie unter `loopStartTime`.
`#loopEndTime`	Der Zeitpunkt im Sound, zu dem die Schleifenwiedergabe endet, in Millisekunden. Weitere Informationen finden Sie unter `loopEndTime`.
`#preloadTime`	Angabe, wie viel Sound vor Abspielbeginn in den Zwischenspeicher geladen werden soll, in Millisekunden. Weitere Informationen finden Sie unter `preloadTime`.

Parameter

Keiner

Beispiel

Die folgende Prozedur reiht zwei Sounds in die Warteschlange für Soundkanal 2 ein, startet ihre Wiedergabe und zeigt anschließend die Abspielliste `playList` im Nachrichtenfenster an. Beachten Sie, dass in der Abspielliste nur der zweite wartende Sound aufgeführt ist, weil der erste Sound bereits wiedergegeben wird.

```
-- Lingo syntax
on playMusic
    sound(2).queue(member("Chimes"))
    sound(2).queue([#member:member("introMusic"), #startTime:3000, #endTime:10000,
#loopCount:5, #loopStartTime:8000, #loopEndTime:8900])
    put(sound(2).getPlayList())
    sound(2).play()
end playMusic

// JavaScript syntax
function playMusic() {
    sound(2).queue(member("Chimes"));
    sound(2).queue(propList("member",member("introMusic"), "startTime",3000, "endTime",10000,
"loopCount",5, "loopStartTime",8000, "loopEndTime",8900));
    put(sound(2).getPlayList());
sound(2).play();
}
```

Siehe auch

endTime (Soundkanal), loopCount, loopEndTime (Soundkanal), loopStartTime, Member, member, preLoadTime, queue(), setPlayList(), Sound Channel, startTime (Sound Channel)

getPos()

Syntax

```
list.getPos(value)
getPos(list, value)
```

Beschreibung

Diese Listenfunktion identifiziert die Position eines Werts in einer Liste. Befindet sich der angegebene Wert nicht in der Liste, gibt der Befehl `getPos` den Wert 0 zurück.

Ist ein Wert mehrmals in der Liste vorhanden, wird nur der erste aufgefundene Wert angezeigt. Bei linearen Listen hat dieser Befehl die gleiche Funktion wie der Befehl `getOne`.

Wenn Sie mithilfe der `add`oder `append`-Methode der Filterliste `filterlist`einen Filter hinzufügen, wird ein Duplikat erstellt und der Liste hinzugefügt. Methoden wie `deleteOne`, `getPos`, `findPos` und `getOne` verwenden den genauen Wert in der Liste und nicht den Duplikatwert.

In diesem Fall kann die Methode `getPos` wie folgt verwendet werden:

```
f = filter(#glowfilter)
sprite(1).filterlist.append(f)
f = sprite(1).filterlist[1]-- here we get the actual value added to the list.
sprite(1).filterlist.getPos(f)
```

Die dritte Zeile in dem vorstehenden Skript ruft den Verweis des der Liste hinzugefügten Filterwerts ab.

Parameter

value Erforderlich. Gibt den der Position zugeordneten Wert an.

Beispiel

Die folgende Anweisung gibt die Position des Wertes 12 in der Liste `Answers` an, die aus [`#a:10, #b:12, #c:15, #d:22`] besteht:

```
-- Lingo
put Answers.getPos(12)

// Javascript
trace(Answers.getPos(12));
```

Das Ergebnis ist 2, da 12 der zweite Wert in der Liste ist.

Siehe auch

`getOne()`

getPosition()

Syntax

```
-- Lingo syntax
fileioObjRef.getPosition()

// JavaScript syntax
fileioObjRef.getPosition();
```

Beschreibung

Diese Datei-E/A-Methode (Fileio) gibt die Position einer Datei zurück.

Parameter

Keiner

Beispiel

Die folgende Anweisung öffnet die Datei „c:\xtra.txt“ und ruft die aktuelle Position ab.

```
-- Lingo syntax
objFileio = new xtra("fileio")
objFileio.openFile("c:\xtra.txt",0)
put objFileio.getPosition()

// JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.openFile("c:\xtra.txt",0);
trace(objFileio.getPosition());
```

Siehe auch

`Fileio`

getPref()

Syntax

```
getPref(prefFileName)
```

Beschreibung

Diese Funktion ruft den Inhalt der angegebenen Datei ab.

Ersetzen Sie bei Verwendung dieser Funktion *prefFileName* durch den Namen einer Datei, die von der Funktion `setPref` erstellt wurde. Wenn keine solche Datei existiert, gibt `getPref` den Wert `VOID` zurück.

Der für *prefFileName* verwendete Dateiname muss ein gültiger Dateiname sein, aber keine vollständige Pfadangabe, da Director den Pfad liefert. Der Dateipfad wird von Director festgelegt. Die einzigen gültigen Dateierweiterungen für *prefFileName* sind .txt und .htm. Alle anderen Erweiterungen werden zurückgewiesen.

Dieser Befehl darf nicht zum Zugriff auf schreibgeschützte oder gesperrte Medien verwendet werden.

Hinweis: *Von `setPref` geschriebene Daten sind in einem Browser nicht privat; jeder Film mit Shockwave-Inhalt kann diese Informationen lesen und auf einen Server hochladen. Vertrauliche Informationen sollten daher nicht mit `setPref` gespeichert werden.*

Ein Beispiel für `getPref()` in einem fertigen Film finden Sie im Film „Read and Write Text“ im Ordner „Learning\\Lingo_“ im Director-Anwendungsordner.

Parameter

prefFileName Erforderlich. Gibt die Datei an, aus der Inhalt abgerufen wird.

Beispiel

Die folgende Prozedur ruft den Inhalt der Datei "Test" ab und ordnet den enthaltenen Text dem Feld "Gesamtergebnis" zu:

```
on mouseUp
    theText = getPref("Test")
    member("Total Score").text = theText
end
```

Siehe auch

`setPref()`

getPref() (Player)

Syntax

```
-- Lingo syntax
_player.getPref(stringPrefName)

// JavaScript syntax
_player.getPref(stringPrefName);
```

Beschreibung

Diese Player-Methode ruft den Inhalt der angegebenen Datei ab.

Ersetzen Sie bei Verwendung dieser Methode *stringPrefName* durch den Namen einer Datei, die mit der Methode `setPref()` erstellt wurde. Wenn keine solche Datei vorhanden ist, gibt `getPref()` den Wert `VOID` (Lingo) oder `null` (JavaScript-Syntax) zurück.

Der für *stringPrefName* verwendete Dateiname muss ein gültiger Dateiname sein, aber keine vollständige Pfadangabe, da Director den Pfad liefert. Der Dateipfad wird von Director festgelegt. Dieeinzigen gültigen Dateierweiterungen für *stringPrefName1* sind TXT und HTM. Alle anderen Erweiterungen werden zurückgewiesen.

Diese Methode darf nicht zum Zugriff auf schreibgeschützte oder gesperrte Medien verwendet werden.

Hinweis: *Mit `setPref()` geschriebene Daten sind in einem Browser nicht geschützt. Jeder Film mit Shockwave-Inhalt ist in der Lage, diese Informationen zu lesen und sie auf einen Server hochzuladen. Vertrauliche Informationen sollten daher nicht mit `setPref()` gespeichert werden.*

Ein Beispiel für `getPref()` in einem fertigen Film finden Sie im Film „Read and Write Text“ im Ordner „Learning\\Lingo_“ im Director-Anwendungsordner.

Parameter

stringPrefName Erforderlich. Eine Zeichenfolge, die die Datei angibt, für die Inhalt abgerufen wird.

Beispiel

Die folgende Prozedur ruft den Inhalt der Datei "Test" ab und ordnet den enthaltenen Text dem Feld "Gesamtergebnis" zu:

```
-- Lingo syntax
on mouseUp
    theText = _player.getPref("Test")
    member("Total Score").text = theText
end

// JavaScript syntax
function mouseUp() {
    var theText = _player.getPref("Test");
    member("Total Score").text = theText;
}
```

Siehe auch

`Player`, `setPref()`

getProp()

Syntax

```
getProp(list, property)
list.property
```

Beschreibung

Diese Eigenschaftslistenfunktion identifiziert den einer Eigenschaft in einer Eigenschaftsliste zugeordneten Wert.

Nahezu identisch mit dem Befehl `getaProp` zeigt der Befehl `getProp` jedoch eine Fehlermeldung an, wenn die angegebene Eigenschaft nicht in der Liste enthalten ist oder eine lineare Liste angegeben wurde.

Parameter

list Erforderlich. Gibt die Eigenschaftsliste an, aus der die Eigenschaft *property* abgerufen wird.

property Erforderlich. Gibt die Eigenschaft an, der der angegebene Wert zugeordnet ist.

Beispiel

Die folgende Anweisung gibt den Wert an, der mit der Eigenschaft `#c` in der Eigenschaftsliste `Answers` verbunden ist, die aus `[#a:10, #b:12, #c:15, #d:22]` besteht:

```
-- Lingo
getProp(Answers, #c)

/ Javascript
Answers.getProp("c");
```

Das Ergebnis ist 15, da 15 den mit `#c` verbundenen Wert darstellt.

Siehe auch

`getOne()`

getPropAt()

Syntax

```
list.getPropAt(index)
getPropAt(list, index)
```

Beschreibung

Diese Eigenschaftslistenfunktion, die nur für Eigenschaftslisten verwendet werden kann, kennzeichnet den Eigenschaftsnamen, der einer angegebenen Position in einer Eigenschaftsliste zugeordnet ist. Befindet sich das angegebene Element nicht in der Liste oder wird `getPropAt()` für eine lineare Liste verwendet, tritt ein Skriptfehler auf.

Parameter

index Erforderlich. Gibt die Indexposition der Eigenschaft in der Eigenschaftsliste an.

Beispiel

Die folgende Anweisung gibt die zweite Eigenschaft in der Eigenschaftsliste `Answers` an, die aus `[#a:10, #b:12, #c:15, #d:22]` besteht:

```
-- Lingo
put Answers.getPropAt(2)
-- #b

// Javascript
trace(Answers.getPropAt(2))
// b
```

getPropRef() (nur JavaScript)

Syntax

```
getPropRef(list, property)
```

Beschreibung

Diese Eigenschaftslistenfunktion gibt die Referenz für eine Eigenschaft in einer Eigenschaftsliste zurück. Die Funktion dient zum Zugriff auf verschachtelte Eigenschaften in JavaScript-Syntax. Es lässt sich auch die Methode `getPropRef()` nutzen, um eine Referenz auf ein bereits bekanntes Objekt oder seine Eigenschaften zu speichern.

Parameter

Eigenschaft	Beschreibung
`list`	Erforderlich. Gibt die Eigenschaftsliste an, aus der die Eigenschaft property abgerufen wird.
`property`	Erforderlich. Gibt die Eigenschaft an, der der angegebene Wert zugeordnet ist.

Beispiel

```
put(member(1).getPropRef("paragraph",1))
//Retrieves the first paragraph from the text member.
```

getRendererServices()

Syntax

```
getRendererServices()
getRendererServices().whichGetRendererServicesProperty
```

Beschreibung

Dieser 3D-Befehl gibt das `rendererServices`-Objekt zurück. Dieses Objekt enthält Hardwareinformationen und -eigenschaften, die sich auf alle 3D-Sprites und -Darsteller auswirken.

Das `rendererServices`-Objekt weist folgende Eigenschaften auf:

- `renderer` gibt den Software-Rasterizer an, der zum Rendern aller 3D-Sprites verwendet wird.
- `rendererDeviceList` gibt eine Liste aller auf dem Benutzersystem verfügbaren Software-Rasterizer zurück. Mögliche Werte: `#openGL`, `#directX5_2`, `#directX7_0`, `#directx9` und `#software`. Die Eigenschaft `renderer` muss einen dieser Werte aufweisen. Diese Eigenschaft kann getestet, aber nicht eingestellt werden.
- `textureRenderFormat` gibt das vom Renderer verwendete Pixelformat an. Mögliche Werte: `#rgba8888`, `#rgba8880`, `#rgba5650`, `#rgba5550`, `#rgba5551` und `#rgba4444`. Die vier Ziffern in jedem Symbol geben an, wie viele Bits für jede Rot-, Grün-, Blau- und Alphakomponente verwendet werden.
- `depthBufferDepth` gibt die Bittiefe des Hardwareausgabepuffers an.
- `colorBufferDepth` gibt die Bittiefe des Farbpuffers an. Diese Eigenschaft kann getestet, aber nicht eingestellt werden.
- `modifiers` ist eine lineare Liste mit den Modifizierern, die von Modellen in 3D-Darstellern verwendet werden können. Mögliche Werte: `#collision`, `#bonesPlayer`, `#keyframePlayer`, `#toon`, `#lod`, `#meshDeform`, `#sds`, `#inker` und auf Xtras von Dritten basierende Modifizierer. Diese Eigenschaft kann getestet, aber nicht eingestellt werden.

- `primitives` ist eine lineare Liste mit den Primitiventypen, die bei der Erstellung neuer Modellressourcen verwendet werden können. Mögliche Werte: `#sphere`, `#box`, `#cylinder`, `#plane`, `#particle` und auf Xtras von Dritten basierende Primitiventypen. Diese Eigenschaft kann getestet, aber nicht eingestellt werden.

Hinweis: *Weitere Informationen zu diesen Eigenschaften finden Sie in den einzelnen Beschreibungen.*

Parameter
Keiner

Beispiel
Die folgende Anweisung ruft das `rendererServices`-Objekt und die Renderer-Informationen ab.

```
-- Lingo
Ro = getRendererServices()
Put Ro.renderer
-- #directX7_0

// Javascript
Var ro = getRendererServices();
trace(ro.renderer);
// #directX7_0
```

Siehe auch
`renderer`, `preferred3dRenderer`, `active3dRenderer`, `rendererDeviceList`

getSoundObject()

Syntax
```
soundObjRef = mixer.getSoundObject(SoundObjName)
```

Beschreibung
Diese Audiomethode gibt das mit dem Mixer verknüpfte Soundobjekt des betreffenden Namens zurück. Es gibt „Leer“ aus, wenn Director kein passendes Sound-Objekt findet.

Rückgabewert
`SoundObjRef`

Parameter

Parameter	Beschreibung	Erforderlich/Optional
SoundObjRef	Soundobjekt-Referenz	Erforderlich
SoundObjName	Names des zu ermittelnden Soundobjekts.	Erforderlich

Beispiele

```
-- Lingo syntax

 on mouseUp me

    mixer1.getSoundObject("SoundObj1") -- Returns the reference of the sound object named
"soundobj1".
 end

// JavaScript syntax
 function mouseup()
 {
 mixer1.getSoundObject("SoundObj1"); //Returns the reference of the sound object named "SoundObj1".

 }
```

Siehe auch

Mixer

getSoundObjectList

Syntax

```
soundObjList = mixer.getSoundObjectList()
```

Beschreibung

Diese Audiomethode gibt eine Liste aller Soundobjekte des Mixers zurück, die noch nicht gelöscht wurden.

Rückgabewert

Liste der Soundobjekte

Beispiele

```
-- Lingo syntax
on mouseUp me

    mixer1.getSoundObjectList() -- Returns the list of the sound objects in mixer1.
end

// JavaScript syntax

function mouseup()
{
mixer1.getSoundObjectList();  // Returns the list of sound objects in mixer1.


}
```

Siehe auch

Mixer

getStreamStatus()

Syntax

```
getStreamStatus(netID)
getStreamStatus(URLString)
```

Beschreibung

Diese Funktion gibt eine Eigenschaftsliste zurück, die dem Format für die global verfügbare Funktion `tellStreamStatus` entspricht, die zum Rückruf von Sprites oder Objekten verwendet werden kann. Die Liste enthält die folgenden Strings:

#URL	String mit der URL-Adresse, die zum Start des Netzwerkvorgangs verwendet wurde.
`#state`	Der String bestehend aus "Connecting" (Verbindung wird hergestellt), "Started" (Gestartet), "InProgress" (Läuft), "Complete" (Abgeschlossen), "Error" (Fehler) oder "NoInformation" (KeineInformation – der letzte String wird angezeigt, wenn entweder die Netzwerk-ID so alt ist, dass die Statusinformationen gelöscht wurden, oder wenn die in `URLString` angegebene URL im Cachespeicher nicht gefunden wurde).
`#bytesSoFar`	Anzahl von Byte, die bereits aus dem Netzwerk abgerufen wurden.
`#bytesTotal`	Gesamtzahl der zu streamenden Byte, sofern bekannt. Der Wert kann 0 sein, wenn der HTTP-Server die Länge des Inhalts nicht im MIME-Header angibt.
`#error`	String, der „" (`EMPTY`) enthält, wenn der Download unvollständig ist, „OK", wenn der Download erfolgreich abgeschlossen wurde, oder einen Fehlercode, wenn der Download mit einem Fehler endete.

Sie können zum Beispiel einen Netzwerkvorgang mit `getNetText()` starten und dann seinen Verlauf mit `getStreamStatus()` verfolgen.

Parameter

netID Erforderlich. Ein Netzwerkvorgang, der den zu verarbeitenden Textstream darstellt.

Beispiel

Die folgende Anweisung zeigt im Nachrichtenfenster den aktuellen Status eines Ladevorgangs an, der mit `getNetText()` begonnen wurde, sowie die resultierende Netzwerk-ID, die in die Variable `netID` gestellt wurde:

```
-- Lingo
put getStreamStatus(netID)
-- [#URL: "www.adobe.com", #state: "InProgress", #bytesSoFar: 250, #bytesTotal: 50000, #error:
EMPTY]

// Javascript
trace(getStreamStatus(netID))
// <[#URL: "www.adobe.com", #state: "InProgress", #bytesSoFar: 250, #bytesTotal: 50000,
#error: EMPTY]>
```

Siehe auch

`on streamStatus`, `tellStreamStatus()`

getSystemCharSet

Syntax

```
_system.getSystemCharSet()
```

Beschreibung

Diese Zeichensatzmethode gibt den Stammzeichensatz des Computers zurück.

Beispiele

```
--Lingo syntax
put _system.getSystemCharSet()

//JavaScript syntax
put(_system.getSystemCharSet());
```

getURL()

Syntax

```
GetUrl(MUIObject, message, MovableOrNot)
```

Beschreibung

Diese Funktion zeigt ein Dialogfeld für die Eingabe eines URL an und gibt den vom Benutzer eingegebenen URL zurück.

- *message* gibt die Nachricht an, die im Feld zur Eingabe eines URL angezeigt wird. Beim ersten Öffnen des Dialogfeldes wird diese Zeichenfolge als vordefinierter Wert gesendet. Wenn der Benutzer auf eine Schaltfläche klickt, gibt Lingo die Zeichenfolge zurück, die der Benutzer eingegeben hat. Wenn der Benutzer auf „Abbrechen“ klickt, entspricht die zurückgegebene Zeichenfolge der ursprünglichen Zeichenfolge.
- Auf dem Macintosh gibt *MovableOrNot* an, ob das Dialogfeld verschiebbar ist. Falls `TRUE` ist das Dialogfeld verschiebbar. Falls `FALSE` ist das Dialogfeld nicht verschiebbar. Unter Windows ist das `GetURL`-Dialogfeld stets verschiebbar.

Beispiel

Diese Anweisungen dienen zum Anzeigen eines Dialogfelds für die Eingabe eines URL.

- Die erste Anweisung erzeugt eine Instanz des Xtras MUI, bei der es sich um das Objekt handelt, das als Dialogfeld verwendet wird.
- Die zweite Anweisung zeigt mithilfe der Funktion `GetUrl` ein verschiebbares Dialogfeld für die Eingabe von URLs an und weist das Dialogfeld dem Variablenergebnis zu. Die Meldung „Enter a URL here“ wird im Feld zur Eingabe eines URL des Dialogfelds angezeigt.
- Die letzten Anweisungen prüfen, ob das Ergebnis, nachdem der Benutzer auf eine Schaltfläche geklickt hat, der Zeichenfolge entspricht, die beim Öffnen des Dialogfelds gesendet wurde. Ist es unterschiedlich, hat der Benutzer einen URL eingeben und auf „OK“ geklickt.

```
-- Lingo
set MUIObj = new (xtra "Mui")
set result = GetUrl(MUIObj, "Enter a URL", TRUE )
if objectP ( MUIObj) then
    set result = GetUrl( MUIObj, "Enter a URL", TRUE )
    if ( result <> "Enter a URL" ) then
        goToNetPage result
    end if
end if
```

getVal()

Syntax

```
<float> Matrix.getVal(whichRow, whichColumn)
```

Beschreibung

Matrixmethode. Ruft den Wert des angegebenen Elements in der angegebenen Matrix ab.

Parameter

whichRow Erforderlich. Zeilennummer des Elements, dessen Wert gelesen wird.

whichColumn Erforderlich. Spaltennummer des Elements, dessen Wert gelesen wird.

Beispiel

Die folgende Funktion verwendet die `getVal()`-Methode einer Matrix zum Umwandeln einer Matrix in eine lineare Liste.

```
--Lingo
on matrixToList(mat)
     rows = mat.numRows
     cols = mat.numColumns
     matrixList = []
     repeat with i = 1 to rows
       repeat with j = 1 to cols
         matrixList.append(mat.getVal(i,j))
       end repeat
     end repeat
     return matrixList
end
//Java Script
function matrixToList(mat)
{
     rows = mat.numRows;
     cols = mat.numColumns;
     matrixList = list();
     for( i = 1; i <= rows; i++)
     {
       for( j = 1; j <= cols; j++)
         matrixList.append(mat.getVal(i,j));
}
     return matrixList;
}
```

Siehe auch

`setVal()`, `numRows()`, `numColumns()`, `matrixAddition()`, `matrixMultiply()`, `matrixMultiplyScalar()`, `matrixTranspose()`, `newMatrix()`

getVariable()

Syntax

```
-- Lingo syntax
spriteObjRef.getVariable(variableName {, returnValueOrReference})

// JavaScript syntax
spriteObjRef.getVariable(variableName {, returnValueOrReference});
```

Beschreibung

Diese Funktion gibt den aktuellen Wert der angegebenen Variablen im angegebenen Flash-Sprite zurück. Flash-Variablen wurden in Version 4 von Flash eingeführt.

Diese Funktion kann auf zweierlei Weise verwendet werden.

Wenn Sie den optionalen Parameter *returnValueOrReference* auf `TRUE` setzen (Standard), wird der aktuelle Wert der Variablen als String zurückgegeben. Wenn Sie den optionalen Parameter *returnValueOrReference* auf `FALSE` setzen, wird der aktuelle Literalwert der Flash-Variablen zurückgegeben.

Wenn der Wert der Flash-Variablen ein Objektbezug ist, müssen Sie den Parameter *returnValueOrReference* auf `FALSE` setzen, damit der Rückgabewert als Objektbezug interpretiert wird. Bei Rückgabe als String ist er nicht mehr als Objektbezug gültig.

Parameter

variableName Erforderlich. Gibt den Namen der Variablen an, deren Wert zurückgegeben wird.

returnValueOrReference Optional. Gibt an, ob der zurückgegebene Wert eine Zeichenfolge (`TRUE`) oder ein Objektverweis (`FALSE`) ist.

Beispiel

Die folgende Anweisung setzt die Variable `tValue` auf den Stringwert der Flash-Variablen `gOtherVar` im Flash-Film in `sprite 3`:

```
-- Lingo syntax
tValue = sprite(3).getVariable("gOtherVar", TRUE)
put(tValue) -- "5"
// JavaScript syntax
var tValue = sprite(3).getVariable("gOtherVar", true);
trace(tValue); // 5
```

Die folgende Anweisung bewirkt, dass die Variable tObject auf das gleiche Objekt Bezug nimmt wie die Variable gVar im Flash-Film in Sprite 3:

```
-- Lingo syntax
tObject = sprite(3).getVariable("gVar",FALSE)
// JavaScript syntax
var tObject = sprite(3).getVariable("gVar",0);
```

Die folgende Anweisung gibt den Wert der Variablen `currentURL` aus dem Flash-Darsteller in `sprite 3` zurück und zeigt ihn im Nachrichtenfenster an:

```
-- Lingo syntax
put(sprite(3).getVariable("currentURL"))

// JavaScript syntax
trace(sprite(3).getVariable("currentURL"));
```

Siehe auch

`setVariable()`

GetWidgetList()

Syntax

```
GetWidgetList(MUIObject)
```

Beschreibung

Diese Funktion gibt eine lineare Liste mit Symbolen für Typen allgemeiner Dialogfeldkomponenten zurück, die für eine Instanz des Xtra MUI unterstützt werden.

Beispiel

Diese Anweisung zeigt eine Liste mit Oberflächenelementen an, die von „MUIObject“, einer Instanz des Xtra MUI, unterstützt werden:

```
-- Lingo
put GetWidgetList(MUIObject)
-- [#dividerV, #dividerH, #bitmap, #checkBox, #radioButton, #PopupList, #editText,
#WindowBegin, #WindowEnd, #GroupHBegin, #GroupHEnd, #GroupVBegin, #GroupVEnd, #label,
#IntegerSliderH, #FloatSliderH, #defaultPushButton, #cancelPushButton, #pushButton,
#toggleButton]
```

GetWindowPropList

Syntax

```
GetWindowPropList(MUIObject)
```

Beschreibung

Diese Funktion gibt eine Liste der vordefinierten Einstellungen des Xtras MUI für Fenster in einem allgemeinen Dialogfeld zurück.

Nutzen Sie beim Definieren eines neuen allgemeinen Dialogfelds die Funktion `GetWindowPropList`, um eine umfassende Liste mit Dialogfeldeigenschaften und -werten abzurufen und einzelne Eigenschaften anschließend den Anforderungen entsprechend zu bearbeiten. Dieser Schritt ist nicht nur zweckmäßig, sondern stellt auch Kompatibilität mit künftigen Versionen des Xtras MUI sicher, die ggf. über weitere Eigenschaften verfügen.

Es folgen die Fenstereigenschaften und vordefinierten Werte, die `GetWindowPropList` zurückgibt:

Eigenschaft	Vordefinierter Wert
#type	#normal
#name	"window"
#callback	"nothing"
#mode	#data
#xPosition	100
#yPosition	120
#width	200
#height	210
#modal	1
#toolTips	0
#closeBox	1
#canZoom	0

Beispiel

Diese Anweisungen dienen zur Definition eines neuen allgemeinen Dialogfelds. Die erste Anweisung weist eine Liste vordefinierter Eigenschaften der Variablen `thePropList` zu. Nachfolgende Anweisungen passen das Dialogfeld durch Ändern folgender Einstellungen an:

```
-- Lingo
set thePropList = GetWindowPropList(muiObject)
set the name of thePropList = "Picture Window"
set the callback of thePropList = "theWindowCallback"
set the mode of thePropList = #data
set the modal of thePropList = TRUE
set the closeBox of thePropList = FALSE
```

getWorldTransform()

Syntax

```
member(whichCastmember).node(whichNode).getWorldTransform()
member(whichCastmember).node(whichNode).getWorldTransform().position
member(whichCastmember).node(whichNode).getWorldTransform().rotation
member(whichCastmember).node(whichNode).getWorldTransform().scale
```

Beschreibung

Dieser 3D-Befehl gibt die auf die Welt bezogene Transformation des Modells, der Gruppe, der Kamera oder des Lichts zurück, das bzw. die durch `node` dargestellt ist.

Die Eigenschaft `transform` eines Node wird bezogen auf die Transformation ihres Parent-Node berechnet. Der Befehl `getWorldTransform()` berechnet die Transformation des Nodes bezogen auf den Ursprung der 3D-Welt (weltbezogen).

Mit `member(`*whichCastmember*`).node(`*whichNode*`).getWorldTransform().position` können Sie die Eigenschaft position der weltbezogenen Transformation des Nodes ermitteln. Sie können auch `worldPosition` als Kurzform für `getWorldTransform().position` verwenden.

Mit `member(`*whichCastmember*`).node(`*whichNode*`).getWorldTransform().rotation` können Sie die Eigenschaft rotation der weltbezogenen Transformation des Nodes ermitteln.

Mit `member(`*whichCastmember*`).node(`*whichNode*`).getWorldTransform().scale` können Sie die Eigenschaft scale der weltbezogenen Transformation des Nodes ermitteln.

Diese Eigenschaften können getestet, aber nicht gesetzt werden.

Beispiel

Die folgende Anweisung zeigt die weltbezogene Transformation des Modells "Box" gefolgt von ihren Positions- und Drehungseigenschaften:

```
put member("3d world").model("Box").getworldTransform()
-- transform(1.000000,0.000000,0.000000,0.000000, 0.000000,1.000000,0.000000,0.000000,
0.000000,0.000000,1.000000,0.000000, - 94.144844,119.012825,0.000000,1.000000)
put member("3d world").model("Box"). getworldTransform().position
-- vector(-94.1448, 119.0128, 0.0000)
put member("3d world").model("Box"). getworldTransform().rotation
--vector(0.0000, 0.0000, 0.0000)

// Javascript
trace(member("3d world").getProp("model","Box").getworldTransform()
// <transform(1.000000,0.000000,0.000000,0.000000, 0.000000,1.000000,0.000000,0.000000,
0.000000,0.000000,1.000000,0.000000, -94.144844,119.012825,0.000000,1.000000)>
trace(member("3d world").getProp("model","Box"). getworldTransform().position
// <vector(-94.1448, 119.0128, 0.0000)>
trace(member("3d world").getProp("model","Box"). getworldTransform().rotation
//<vector(0.0000, 0.0000, 0.0000)>
```

Siehe auch

worldPosition, transform (Eigenschaft)

go()

Syntax

```
-- Lingo syntax
_movie.go(frameNameOrNum {, movieName})

// JavaScript syntax
_movie.go(frameNameOrNum {, movieName});
```

Beschreibung

Diese Filmmethode lässt den Abspielkopf zu einem angegebenen Bild in einem vorgegebenen Film verzweigen.

Mithilfe der Methode können Sie den Abspielkopf anweisen, eine Schleife zur vorherigen Markierung zu bilden. Sie stellt außerdem ein bequemes Verfahren dar, um den Abspielkopf im selben Bereich des Films zu halten, solange Skriptcode aktiv ist.

Es ist besser, Markierungsbeschriftungen für *frameNameOrNum* statt Bildnummern zu verwenden, da sich Bildnummern bei der Bearbeitung eines Films ändern können. Außerdem lassen sich Skripts leichter lesen, wenn Markierungsbeschriftungen verwendet werden.

Das Aufrufen von `go()` mit dem Parameter *movieName* lädt Bild 1 des Films. Wird `go()` von einer Prozedur aus aufgerufen, wird die Prozedur, in der sich die Methode befindet, weiter ausgeführt.

Wenn Sie einen Film abspielen möchten, der sich in einem anderen Ordner befindet, müssen Sie den vollständigen Pfad angeben. Um einen möglichen Ladefehler zu vermeiden, sollten Sie hierbei allerdings die Dateierweiterung DIR, DXR oder DCR weglassen.

Wenn Sie auf schnellstem Weg auf einen Film unter einer URL zugreifen möchten, laden Sie zuerst die Filmdatei mit der `downloadNetThing()`-Methode auf einen lokalen Datenträger herunter und rufen dann mit der `go()`-Methode mit dem Parameter *movieName* den Film auf dem lokalen Datenträger auf.

Die `goLoop()`-Methode schickt den `playhead` (Abspielkopf) zur vorherigen Markierung in einem Film, wobei es sich um ein bequemes Verfahren handelt, mit dem der Abspielkopf im selben Bereich des Films gehalten werden kann, solange Lingo- bzw. JavaScript-Syntax-Code aktiv ist.

Folgende Variablen werden beim Laden eines Films zurückgesetzt: die Eigenschaften `beepOn` und `constraint`; `keyDownScript`, `mouseDownScript` und `mouseUpScript`; die Sprite-Eigenschaften `cursor` und `immediate`; die Methoden `cursor()` und `puppetSprite()` sowie benutzerdefinierte Menüs. Das `timeoutScript` wird hingegen beim Laden eines Filmes nicht zurückgesetzt.

Parameter

frameNameOrNum Erforderlich. Eine Zeichenfolge, die die Markierungsbeschriftung des Bildes angibt, zu dem der Abspielkopf verzweigt, oder eine Ganzzahl, die die Nummer des Bildes angibt, zu dem der Abspielkopf verzweigt.

movieName Optional. Eine Zeichenfolge, die den Film angibt, der das durch *frameNameOrNum* angegebene Bild enthält. Dieser Wert muss eine Filmdatei angeben. Befindet sich der Film in einem anderen Ordner, muss *movieName* ebenfalls den Pfad angeben.

Beispiel

Die folgende Anweisung schickt den Abspielkopf zur Markierung "Start":

```
-- Lingo syntax
_movie.go("start")

// JavaScript syntax
_movie.go("start");
```

Die folgende Anweisung schickt den Abspielkopf zur Markierung „Erinnerung“ im Film „Noh Tale to Tell“:

```
-- Lingo syntax
_movie.go("Memory", "Noh Tale to Tell")

// JavaScript syntax
_movie.go("Memory", "Noh Tale to Tell");
```

Die folgende Prozedur weist den Film an, das aktuelle Bild in Schleife abzuspielen. Dies ist nützlich, wenn der Film beim Abspielen in einem Bild warten soll, damit er auf Ereignisse reagieren kann.

```
-- Lingo syntax
on exitFrame
    _movie.go(_movie.frame)
end

// JavaScript syntax
function exitFrame() {
    _movie.go(_movie.frame);
}
```

Siehe auch

downloadNetThing, goLoop(), Movie

goLoop()

Syntax

```
-- Lingo syntax
_movie.goLoop()

// JavaScript syntax
_movie.goLoop();
```

Beschreibung

Diese Filmmethode schickt den Abspielkopf zur vorherigen Markierung im Film zurück, und zwar entweder zu einer Markierung vor dem aktuellen Bild, falls dieses keine Markierung enthält, oder zum aktuellen Bild, falls es eine Markierung enthält.

Wenn sich links neben Abspielkopf keine Markierungen befinden, springt der Abspielkopf zu einer der folgenden Positionen:

- zu der nächsten Markierung auf der rechten Seite, falls das aktuelle Bild keine Markierung aufweist
- zum aktuellen Bild, falls dieses eine Markierung aufweist
- zu Bild 1, falls der Film keine Markierungen enthält

Parameter

Keiner

Beispiel

Die folgende Anweisung bewirkt, dass der Film zwischen dem aktuellen Bild und der vorherigen Markierung in Schleife abgespielt wird:

```
-- Lingo syntax
_movie.goLoop()

// JavaScript syntax
_movie.goLoop();
```

Siehe auch

go(), goNext(), goPrevious(), Movie

goNext()

Syntax

```
-- Lingo syntax
_movie.goNext()

// JavaScript syntax
_movie.goNext();
```

Beschreibung

Diese Filmmethode schickt den Abspielkopf zur nächsten Markierung im Film.

Wenn sich rechts neben dem Abspielkopf keine Markierungen befinden, springt der Abspielkopf zur letzten Markierung im Film oder zu Bild 1, falls der Film keine Markierungen enthält.

Parameter

Keiner

Beispiel

Die folgende Anweisung schickt den Abspielkopf zur nächsten Markierung im Film:

```
-- Lingo syntax
_movie.goNext()

// JavaScript syntax
_movie.goNext();
```

Siehe auch

`go()`, `goLoop()`, `goPrevious()`, `Movie`

goPrevious()

Syntax

```
-- Lingo syntax
_movie.goPrevious()

// JavaScript syntax
_movie.goPrevious();
```

Beschreibung

Diese Filmmethode schickt den Abspielkopf zur vorherigen Markierung im Film.

Diese Markierung befindet sich zwei Markierungen vor dem aktuellen Bild, falls dieses keine Markierung enthält, bzw. eine Markierung vor dem aktuellen Bild, falls dieses eine Markierung aufweist.

Wenn sich links neben Abspielkopf keine Markierungen befinden, springt der Abspielkopf zu einer der folgenden Positionen:

- zu der nächsten Markierung rechts, falls das aktuelle Bild keine Markierung aufweist
- zum aktuellen Bild, falls dieses eine Markierung aufweist

- zu Bild 1, falls der Film keine Markierungen enthält

Parameter
Keiner

Beispiel
Die folgende Anweisung schickt den Abspielkopf zur vorherigen Markierung im Film:

```
-- Lingo syntax
_movie.goPrevious()

// JavaScript syntax
_movie.goPrevious();
```

Siehe auch
`go()`, `goLoop()`, `goNext()`, `Movie`

goToFrame()

Syntax

```
-- Lingo syntax
spriteObjRef.goToFrame(frameNameOrNum)

// JavaScript syntax
spriteObjRef.goToFrame(frameNameOrNum);
```

Beschreibung
Dieser Befehl spielt ein Flash-Film-Sprite ab dem im Parameter *frameNumber* angegebenen Bild ab. Sie können das Bild entweder durch eine Ganzzahl bezeichnen, die eine Bildnummer angibt, oder durch einen String, der einen Beschriftungsnamen kennzeichnet. Der Befehl `goToFrame` hat dieselbe Wirkung wie das Einstellen der Eigenschaft `frame` eines Flash-Film-Sprites.

Beispiel
Die folgende Prozedur verzweigt zu verschiedenen Punkten in einem Flash-Film in Kanal 5. Sie akzeptiert einen Parameter, der angibt, zu welchem Bild gesprungen werden soll.

```
-- Lingo syntax
on Navigate(whereTo)
    sprite(5).goToFrame(whereTo)
end

// JavaScript syntax
function Navigate(whereTo) {
    sprite(5).goToFrame(whereTo);
}
```

gotoNetMovie

Syntax

```
gotoNetMovie URL
gotoNetMovie (URL)
```

Beschreibung

Dieser Befehl ruft einen neuen Film mit Shockwave-Inhalt von einem HTTP- oder FTP-Server ab und spielt ihn ab. Die Wiedergabe des aktuellen Films wird so lange fortgesetzt, bis der neue Film verfügbar ist.

Als gültige Parameter werden nur URLs unterstützt. Die URL kann entweder einen Dateinamen oder eine Markierung innerhalb eines Films bezeichnen. Relative URLs können verwendet werden, wenn der Film sich auf einem Internetserver befindet. Sie müssen in diesem Fall allerdings auch die Erweiterung des Dateinamens angeben.

Beim Testen eines Films auf dem lokalen Datenträger oder Netzwerk müssen die Medien im Verzeichnis „dswmedia“ abgelegt sein.

Wenn Sie während eines `gotoNetMovie`-Vorgangs einen zweiten `gotoNetMovie`- Befehl erteilen, ehe der erste beendet ist, bricht der zweite Befehl den ersten ab.

Parameter

URL Erforderlich. Gibt die URL des abzuspielenden Shockwave-Inhalts an.

Beispiel

In der folgenden Anweisung gibt die URL einen Director-Dateinamen an:

```
gotoNetMovie "http://www.yourserver.com/movies/movie1.dcr"
```

In der folgenden Anweisung gibt die URL eine Markierung innerhalb eines Dateinamens an:

```
gotoNetMovie "http://www.yourserver.com/movies/buttons.dcr#Contents"
```

In der folgenden Anweisung wird `gotoNetMovie` als Funktion verwendet. Die Funktion gibt die Netzwerk-ID des Vorgangs zurück.

```
myNetID = gotoNetMovie ("http://www.yourserver.com/movies/buttons.dcr#Contents")
```

gotoNetPage

Syntax

```
gotoNetPage "URL", {"targetName"}
```

Beschreibung

Dieser Befehl öffnet einen Film mit Shockwave-Inhalt oder eine andere MIME-Datei im Browser.

Als gültige Parameter werden nur URLs unterstützt. Relative URLs können verwendet werden, wenn sich der Film auf einem HTTP- oder FTP-Server befindet.

In der Authoring-Umgebung ruft der Befehl `gotoNetPage` den bevorzugten Browser auf, falls dieser aktiviert ist. In Projektoren versucht dieser Befehl, den bevorzugten Browser aufzurufen, der im Dialogfeld „Netzwerk-Voreinstellungen“ oder mit dem Befehl `browserName` eingestellt wurde. Wenn der bevorzugte Browser mit keiner dieser Methoden eingestellt wurde, versucht der Befehl `goToNetPage`, einen Browser auf dem Computer zu finden.

Parameter

URL Erforderlich. Gibt die URL des abzuspielenden Films mit Shockwave-Inhalt oder der MIME-Datei an.

targetName Optional. Ein optionaler HTML-Parameter, der den Frame oder das Fenster kennzeichnet, in dem die Seite angezeigt werden soll.

- Wenn *targetName* ein Fenster oder Frame im Browser ist, ersetzt `gotoNetPage` den Inhalt dieses Fensters oder Frames.
- Wenn *targetName* kein Frame oder geöffnetes Fenster ist, öffnet `goToNetPage` ein neues Fenster. Bei Verwendung des Strings `_new` wird immer ein neues Fenster geöffnet.
- Wenn *targetName* nicht angegeben ist, ersetzt `gotoNetPage` die aktuelle Seite, wo immer sich diesebefindet.

Beispiel

Das folgende Skript lädt die Datei "NeueSeite.html" in den Frame oder das Fenster "frwin". Wenn ein Fenster oder ein Frame im aktuellen Fenster namens "frwin" existiert, wird dieses Fenster oder dieser Frame verwendet. Wenn das Fenster "frwin" nicht vorhanden ist, wird ein neues Fenster mit dem Namen "frwin" erstellt.

```
on keyDown
    gotoNetPage "Newpage.html", "frwin"
end
```

Die folgende Prozedur öffnet ein neues Fenster, unabhängig davon, welches Fenster gegenwärtig im Browser geöffnet ist:

```
on mouseUp
    gotoNetPage "Todays_News.html", "_new"
end
```

Siehe auch

`browserName()`, `netDone()`

group()

Syntax

```
member(whichCastmember).group(whichGroup)
member(whichCastmember).group[index]
```

Beschreibung

Dieses 3D-Element ist ein Node in der 3D-Welt, der einen Namen, eine Transformation, einen Parent-Node und Child-Nodes aufweist, aber keine anderen Eigenschaften.

Jeder 3D-Darsteller besitzt eine Standardgruppe namens "World", die nicht gelöscht werden kann. Die Parent-Hierarchie aller in der 3D-Welt vorhandenen Modelle, Lichtquellen, Kameras und Gruppen endet mit `group("world")`.

Beispiel

Die folgende Anweisung zeigt, dass die vierte Gruppe des Darstellers `newAlien` die Gruppe `Direct01` ist:

```
-- Lingo
put member("newAlien").group[4]

// Javascript
put member("newAlien").getPropRef("group",4) ;
-- group("Direct01")
```

Siehe auch

newGroup, deleteGroup, child (3D), parent

halt()

Syntax

```
-- Lingo syntax
_movie.halt()

// JavaScript syntax
_movie.halt();
```

Beschreibung

Diese Filmmethode beendet die aktuelle Prozedur und jede andere Prozedur, von der sie aufgerufen wurde, und hält während der Erstellung den Film an bzw. beendet einen laufenden Projektor.

Parameter

Keiner

Beispiel

Die folgende Anweisung überprüft, ob die Menge an freiem Speicher geringer als 50 KB ist. Ist dies der Fall, beendet sie alle Prozeduren, von denen aus sie aufgerufen wurde, und hält den Film an:

```
-- Lingo syntax
if (_system.freeBytes < (50*1024)) then
    _movie.halt()
end if

// JavaScript syntax
if (_system.freeBytes < (50*1024)) {
    _movie.halt();
}
```

Siehe auch

Movie

handler()

Syntax

```
scriptObject.handler(#handlerSymbol)
```

Beschreibung
Diese Funktion gibt `TRUE` zurück, wenn das angegebene *scriptObject* eine vorgegebene Prozedur enthält, oder `FALSE`, wenn dies nicht der Fall ist. Bei dem Skriptobjekt muss es sich um ein Parent-Skript, ein Child-Objekt oder ein Verhalten handeln.

Parameter
symHandler Erforderlich. Gibt den Namen der Prozedur an.

Beispiel
Der folgende Code ruft eine Prozedur für ein Objekt nur dann auf, wenn diese Prozedur vorhanden ist:

```
-- Lingo
if spiderObject.handler(#pounce) = TRUE then
    spiderObject.pounce()
end if

// Javascript
if (spiderObject.handler(symbol("pounce")) == true)
{
    spiderObject.pounce();
}
```

Siehe auch
`handlers()`, `new()`, `rawNew()`, `script()`

handlers()

Syntax
`scriptObject.handlers()`

Beschreibung
Diese Funktion gibt eine lineare Liste der Prozeduren im angegebenen *scriptObject* zurück. Die einzelnen Prozedurnamen sind in Form von Symbolen in der Liste aufgeführt. Diese Funktion dient zum Debugging von Filmen.

Ein direktes Abrufen der Prozeduren eines Skriptdarstellers ist nicht möglich. Stattdessen müssen Sie diese über die Eigenschaft `script` des jeweiligen Darstellers abrufen.

Parameter
Keiner

Beispiel
Die folgende Anweisung zeigt die Liste der Prozeduren im Child-Objekt `RedCar` im Nachrichtenfenster an:

```
put RedCar.handlers()
-- [#accelerate, #turn, #stop]
```

Die folgende Anweisung zeigt die Liste der Prozeduren im Parent-Skript-Darsteller `CarParentScript` im Nachrichtenfenster an:

```
put member("CarParentScript").script.handlers()
-- [#accelerate, #turn, #stop]
```

Siehe auch

handler(), script()

hilite (Befehl)

Syntax

```
fieldChunkExpression.hilite()
hilite fieldChunkExpression
```

Beschreibung

Dieser Befehl hebt den angegebenen Chunk im Feld-Sprite hervor (wählt ihn aus). Hierbei kann es sich um einen beliebigen Chunk handeln, der sich mit Lingo definieren lässt, wie z. B. ein Zeichen, Wort oder eine Zeile. Auf dem Mac wird die Hervorhebungsfarbe im Kontrollfeld „Farbe“ eingestellt.

Parameter

Keiner

Beispiel

Die folgende Anweisung hebt das vierte Wort im Felddarsteller "Kommentare" hervor, der den String "Gedanke für diesen Tag" enthält:

```
-- Lingo
member("Comments").word[4].hilite()

// Javascript
member("Comments").getPropRef("word",4).hilite() ;
```

Siehe auch

char...of, item...of, line...of, word...of, delete(), mouseChar, mouseLine, mouseWord, field, selection() (Funktion), selEnd, selStart

hitTest()

Syntax

```
-- Lingo syntax
spriteObjRef.hitTest(point)

// JavaScript syntax
spriteObjRef.hitTest(point);
```

Beschreibung
Diese Funktion gibt an, welcher Teil eines Flash-Films sich direkt über einer bestimmten Position auf der Director-Bühne befindet. Die Director-Bühnenposition wird als Director-Punktwert ausgedrückt, beispielsweise „point(100,50)“. Die Funktion `hitTest` gibt die folgenden Werte zurück:

- `#background` – Die angegebene Bühnenposition befindet sich im Hintergrund des Flash-Film-Sprites.
- `#normal` – Die angegebene Bühnenposition befindet sich in einem gefüllten Objekt.
- `#button` – Die angegebene Bühnenposition befindet sich im aktiven Bereich einer Schaltfläche.
- `#editText` – Die angegebene Bühnenposition befindet sich in einem in Flash bearbeitbaren Textfeld.

Parameter
point Erforderlich. Gibt den zu testenden Punkt an.

Beispiel
Das folgende Bildskript prüft, ob sich die Maus über einer Schaltfläche in einem Flash-Film-Sprite in Kanal 5 befindet, und stellt in diesem Fall ein Textfeld ein, das zur Anzeige einer Statusmeldung verwendet wird:

```
-- Lingo syntax
on exitFrame
    if sprite(5).hitTest(_mouse.mouseLoc) = #button then
        member("Message Line").text = "Click here to play the movie."
        _movie.updatestage()
    else
        member("Message Line").text = ""
    end if
    _movie.go(_movie.frame)
end

// JavaScript syntax
function exitFrame() {
    var hT = sprite(5).hitTest(_mouse.mouseLoc);
    if (hT.toString() == "#button")
    {
        member("Message Line").text = "Click here to play the movie.";
        _movie.updatestage();
    } else {
        member("Message Line").text = "";
    }
    _movie.go(_movie.frame)
}
```

HMStoFrames()

Syntax
```
HMStoFrames(hms, tempo, dropFrame, fractionalSeconds)
```

Beschreibung
Diese Funktion konvertiert Filme, die in Stunden, Minuten und Sekunden gemessen werden, in die entsprechende Anzahl von Bildern bzw. konvertiert eine bestimmte Anzahl von Stunden, Minuten und Sekunden in eine Abspielzeit, wenn Sie das Argument *tempo* auf 1 (1 Bild = 1 Sekunde) setzen.

Parameter

hms Erforderlich. Ein Zeichenfolgenausdruck, der die Zeit im Format `sHH:MM:SS.FFD` angibt, wobei die Buchstaben Folgendes bedeuten:

s	Wenn die Zeit kleiner als Null ist, wird ein Zeichen verwendet, ist sie größer oder gleich Null, wird ein Leerzeichen verwendet.
HH	Stunden.
MM	Minuten.
SS	Sekunden.
FF	Gibt den Bruchteil einer Sekunde an, wenn `fractionalSeconds` den Wert `TRUE` hat, bzw. die Anzahl der Bilder ("frames"), wenn `fractionalSeconds` den Wert `FALSE` hat.
D	Ein "d" wird verwendet, wenn *dropFrame* den Wert `TRUE` hat, bzw. ein Leerzeichen, wenn *dropFrame* den Wert `FALSE` hat.

tempo Erforderlich. Gibt die Wiedergabegeschwindigkeit in Bildern pro Sekunde an.

dropFrame Erforderlich. Logischer Ausdruck, der angibt, ob ein Bild ein dropFrame (`TRUE`) ist oder nicht (`FALSE`). Wenn der String *hms* mit einem *d* endet, wird die Zeit unabhängig vom *dropFrame*-Wert als dropFrame behandelt.

fractionalSeconds Erforderlich. Logischer Ausdruck, der die Bedeutung der Zahlen, die auf die Sekunden folgen, festlegt. Hierbei kann es sich entweder um den auf das nächste Hundertstel einer Sekunde gerundeten Sekundenbruchteil (`TRUE`) oder die Anzahl der verbleibenden Bilder (`FALSE`) handeln.

Beispiel

Die folgende Anweisung bestimmt die Anzahl der Bilder in einem Film, der 1 Minute und 30,1 Sekunden lang ist und ein Tempo von 30 BpS hat. Es werden weder das Argument `dropFrame` noch `fractionalSeconds` verwendet.

```
put HMStoFrames(" 00:01:30.10 ", 30, FALSE, FALSE)
-- 2710
```

Die folgende Anweisung konvertiert 600 Sekunden in Minuten:

```
put framesToHMS(600, 1,0,0)
-- " 00:10:00.00 "
```

Die folgende Anweisung konvertiert 1,5 Stunden in Sekunden:

```
put HMStoFrames("1:30:00", 1,0,0)
-- 5400
```

Siehe auch

`framesToHMS()`

hold()

Syntax

```
-- Lingo syntax
spriteObjRef.hitTest(point)

// JavaScript syntax
spriteObjRef.hitTest(point);
```

Beschreibung
Dieser Flash-Befehl hält ein Flash-Film-Sprite an, das gerade im aktuellen Bild abgespielt wird. Sounds werden jedoch weiter abgespielt.

Parameter
Keiner

Beispiel
Das folgende Bildskript hält die in Kanälen 5 bis 10 spielenden Flash-Film-Sprites an, während die Sounds dieser Kanäle weiter abgespielt werden:

```
-- Lingo syntax
on enterFrame
    repeat with i = 5 to 10
        sprite(i).hold()
    end repeat
end

// JavaScript syntax
function enterFrame() {
    var i = 5;
    while (i < 11) {
        sprite(i).hold();
        i++;
    }
}
```

Siehe auch
playRate (QuickTime, AVI, MP4, FLV)

importByteArrayInto()

Syntax
`memberRef.importByteArrayInto(ByteArrayRef,strFileType)`

Beschreibung
Diese Bytearraymethode import das Mitglied von den Bytearray-Inhalten.

Parameter
Keiner

Beispiel

```
--Lingo Syntax
--Read JPEG file from disk and import it as bitmap
file = new xtra("fileio")
file.openFile(_movie.path & "import.jpg", 1)
ba = file.readByteArray(file.getLength())
m=new(#bytearray)
m.bytearray=ba
member(5).importByteArrayInto(m.bytearray,"jpg")
//Javascript Syntax
file = new xtra("fileio");
file.openFile(_movie.path + "import.jpg", 1);
ba = file.readByteArray(file.getLength());
m=_movie.newMember(symbol("bytearray"));
m.bytearray=ba;
member(5).importByteArrayInto(m.bytearray,"jpg");
```

identity()

Syntax

```
member(whichCastmember).model(whichModel).transform.identity()
member(whichCastmember).group(whichGroup).transform.identity()
member(whichCastmember).camera(whichCamera).transform.identity()
sprite(whichSprite).camera{(index)}.transform.identity()
member(whichCastmember).light(whichLight).transform.identity()
transformReference.identity()
```

Beschreibung

Dieser 3D-Befehl setzt die Transformation auf die Identitätstransformation, d. h. auf `transform(1.0000,0.0000,0.0000,0.0000, 0.0000,1.0000,0.0000,0.0000, 0.0000,0.0000,1.0000,0.0000, 0.0000,0.0000,0.0000,1.0000).`

Die Eigenschaft `position` der Einheitsmatrix ist `vector(0, 0, 0)`.

Die Eigenschaft `rotation` der Einheitsmatrix ist `vector(0, 0, 0)`.

Die Eigenschaft `scale` der Einheitsmatrix ist `vector(1, 1, 1)`.

Die Einheitsmatrix ist Parent-bezogen.

Parameter

Keiner

Beispiel

Die folgende Anweisung setzt die Transformation des Modells "Box" auf die Einheitsmatrix:

```
-- Lingo
member("3d world").model("Box").transform.identity()

// Javascript
member("3d world").getProp("model",1).transform.identity()
```

Das folgende Skript setzt voraus, dass „Box“ das erste verfügbare Modell ist.

Siehe auch

transform (Eigenschaft), getWorldTransform()

idleLoadDone()

Syntax

```
-- Lingo syntax
_movie.idleLoadDone(intLoadTag)

// JavaScript syntax
_movie.idleLoadDone(intLoadTag);
```

Beschreibung

Diese Filmmethode meldet, ob alle Darsteller mit dem entsprechenden Kennzeichen geladen wurden (TRUE) oder noch darauf warten, geladen zu werden (FALSE).

Parameter

intLoadTag Erforderlich. Eine Ganzzahl, die das Ladekennzeichen für die zu testenden Darsteller angibt.

Beispiel

Die folgende Anweisung überprüft, ob alle Darsteller mit dem Ladekennzeichen ("Load Tag") 20 geladen wurden, und spielt in diesem Fall den Film "Kiosk" ab:

```
-- Lingo syntax
if (_movie.idleLoadDone(20)) then
    _movie.play(1, "Kiosk")
end if

// JavaScript syntax
if (_movie.idleLoadDone(20)) {
    _movie.play(1, "Kiosk");
}
```

Siehe auch

idleHandlerPeriod, idleLoadMode, idleLoadPeriod, idleLoadTag, idleReadChunkSize, Movie

ignoreWhiteSpace()

Syntax

```
XMLparserObject.ignoreWhiteSpace(trueOrFalse)
```

Beschreibung

Dieser XML-Befehl gibt an, ob der Parser Leerräume beim Generieren einer Lingo-Liste ignorieren oder beibehalten soll. Wenn ignoreWhiteSpace() auf TRUE gesetzt ist (Standardeinstellung), werden Leerräume ignoriert. Wenn der Befehl auf FALSE gesetzt ist, werden Leerräume beibehalten und als Daten behandelt.

Wenn ein Element getrennte Anfangs- und End-Tags aufweist, wie z. B. `<sample> </sample>`, werden Zeichendaten innerhalb des Elements ignoriert, allerdings nur, sofern sie ausschließlich aus Leerraum bestehen. Wenn das Element andere Zeichen enthält oder wenn `ignoreWhiteSpace()` auf `FALSE` gesetzt ist, wird ein CDATE-Node mit dem genauen Text (einschließlich Leerraum) erstellt.

Parameter

trueOrFalse Erforderlich. Ein Wert, der angibt, ob der Parser Leerraum ignorieren soll (`TRUE`) oder nicht (`FALSE`).

Beispiel

Die folgenden Lingo-Anweisungen lassen die Funktion `ignoreWhiteSpace()` auf den Standardwert `TRUE` gesetzt und schreiben den angegebenen XML-Text nach dem Parsen in eine Liste. Das Element „<sample>“ weist in der Liste keine untergeordneten Objekte auf.

```
XMLtext = "<sample> </sample>"
parserObj.parseString(XMLtext)
theList = parserObj.makelist()
put theList
-- ["ROOT OF XML DOCUMENT": ["!ATTRIBUTES": [:], "sample": ["!ATTRIBUTES": [:]]]]
```

Die folgenden Lingo-Anweisungen setzen die Funktion `ignoreWhiteSpace()` auf `FALSE` und schreiben den angegebenen XML-Text nach dem Parsen in eine Liste. Das Element „<sample>“ weist nun ein untergeordnetes Objekt auf, das ein Leerzeichen enthält.

```
XMLtext = "<sample> </sample>"
parserObj.ignorewhitespace(FALSE)
parserObj.parseString(XMLtext)
theList = parserObj.makelist()
put theList
-- ["ROOT OF XML DOCUMENT": ["!ATTRIBUTES": [:], "sample": ["!ATTRIBUTES": [:], "!CHARDATA":
" "]]]
```

Die folgenden Lingo-Anweisungen lassen die Funktion `ignoreWhiteSpace()` auf den Standardwert `TRUE` gesetzt und parsen den angegebenen XML-Text. Die Tags „<sample>“ und „<sub>“ weisen jeweils nur einen Child-Knoten auf.

```
XMLtext = "<sample> <sub> phrase 1 </sub></sample>"
parserObj.parseString(XMLtext)
theList = parserObj.makeList()
put theList
-- ["ROOT OF XML DOCUMENT": ["!ATTRIBUTES": [:], "sample": ["!ATTRIBUTES": [:], "sub":
["!ATTRIBUTES": [:], "!CHARDATA": " phrase 1 "]]]]
```

Die folgenden Lingo-Anweisungen setzen die Funktion `ignoreWhiteSpace()` auf `FALSE` und parsen den angegebenen XML-Text. Das Tag „<sample>“ besitzt jetzt zwei Child-Knoten; der erste davon ist ein einzelnes Leerzeichen.

```
XMLtext = "<sample> <sub> phrase 1 </sub></sample>"
gparser.ignoreWhiteSpace(FALSE)
gparser.parseString(XMLtext)
theList = gparser.makeList()
put theList
-- ["ROOT OF XML DOCUMENT": ["!ATTRIBUTES": [:], "sample": ["!ATTRIBUTES": [:], "!CHARDATA":
" ", "sub": ["!ATTRIBUTES": [:], "!CHARDATA": " phrase 1 "]]]]
```

ilk()

Syntax

```
ilk(object)
ilk(object, type)
```

Beschreibung

Diese Funktion gibt den Typ eines Objekts an.

Die folgende Tabelle enthält die Rückgabewerte der einzelnen Objekttypen, die von `ilk()` erkannt werden.

Objekttyp	ilk(object)-Rückgabe	ilk(object, type) gibt nur dann 1 zurück, wenn type =	Beispiel
Lineare Liste	`#list`	`#list or #linearlist`	`ilk ([1,2,3])`
Eigenschaftsliste	`#proplist`	`#list or #proplist`	`ilk ([#his: 1234, #hers: 7890])`
Ganzzahl	`#integer`	`#integer or #number`	`ilk (333)`
Fließkommazahl	`#float`	`#float or #number`	`ilk (123.456)`
String	`#string`	`#string`	`ilk ("asdf")`
rect	`#rect`	`#rect or #list`	`ilk (sprite(1).rect)`
Punkt	`#point`	`#point or #list`	`ilk (sprite(1).loc)`
Farbe	`#color`	`#color`	`ilk (sprite(1).color)`
Datum	`#date`	`#date`	`ilk (the systemdate)`
Symbol	`#symbol`	`#symbol`	`ilk (#hello)`
void	`#void`	`#void`	`ilk (void)`
picture	`#picture`	`#picture`	`ilk (member (2).picture)`
Parent-Skriptinstanz	`#instance`	`#object`	`ilk (new (script "blahblah"))`
Xtra-Instanz	`#instance`	`#object`	`ilk (new (xtra "fileio"))`
member	`#member`	`#object` oder `#member`	`ilk (member 1)`
Xtra	`#xtra`	`#object` oder `#xtra`	`ilk (xtra "fileio")`
Skript	`#script`	`#object` oder `#script`	`ilk (script "blahblah")`
Besetzung	`#castlib`	`#object` oder `#castlib`	`ilk (castlib 1)`
Sprite	`#sprite`	`#object` oder `#sprite`	`ilk (sprite 1)`
Sound	`#instance` oder `#sound` (wenn Soundsteuerungs-Xtra nicht vorhanden ist)	`#instance` oder `#sound`	`ilk (sound "yaddayadda")`
window	`#window`	`#object` oder `#window`	`ilk (the stage)`
Medien	`#media`	`#object` oder `#media`	`ilk (member (2).media)`
Timeout	`#timeout`	`#object` oder `#timeout`	`ilk (timeOut("intervalTimer"))`
image	`#image`	`#object` oder `#image`	ilk ((the stage).image)

Parameter
object Erforderlich. Gibt das zu testende Objekt an.

type Optional. Gibt den Typ an, mit dem *object* verglichen wird. Gehört das Objekt dem angegebenen Typ an, gibt die Funktion `ilk()TRUE` zurück. Gehört das Objekt nicht dem angegebenen Typ an, gibt die Funktion `ilk()FALSE` zurück.

Beispiel
Die folgende ilk-Anweisung identifiziert den Typ des Objekts "Gebote":

```
Bids = [:]
put ilk( Bids )
-- #proplist
```

Die folgende ilk-Anweisung testet, ob es sich bei der Variablen Summe um eine Liste handelt, und zeigt das Ergebnis im Nachrichtenfenster an:

```
Total = 2+2
put ilk( Total, #list )
-- 0
```

Da die Variable „Summe“ keine Liste ist, wird im Nachrichtenfenster 0 angezeigt, was das numerische Gegenstück zu FALSE ist.

Die folgende Anweisung testet eine Variable namens `myVariable`, um festzustellen, ob es sich tatsächlich um ein Datum handelt, und zeigt sie anschließend im Nachrichtenfenster an:

```
myVariable = the systemDate
if ilk(myVariable, #date) then put myVariable
-- date( 1999, 2, 19 )
```

ilk (3D)

Syntax

```
ilk(object)
ilk(object,type)
object.ilk
object.ilk(type)
```

Beschreibung
Diese Lingo-Funktion gibt den Typ eines Objekts an.

Die folgende Tabelle enthält die Rückgabewerte der einzelnen 3D-Objekttypen, die von `ilk()` erkannt werden. Eine Aufstellung der Rückgabewerte von Nicht-3D-Objekten, auf die in diesem Wörterbuch nicht eingegangen wird, finden Sie im Haupt-Lingo-Wörterbuch.

Objekttyp	ilk(object)-Rückgabe	ilk(object, type), wenn type =
Renderservices	`#renderer`	`#renderer`
Modellressource	`#modelresource, #plane, #box, #sphere, #cylinder, #particle, #mesh`	Das Gleiche wie `ilk(object)`, außer für `#modelresource`, das der `ilk` der anhand einer importierten W3D-Datei erzeugten Ressourcen ist.
Modell	`#model`	`#model`

Objekttyp	ilk(object)-Rückgabe	ilk(object, type), wenn type =
Bewegung	`#motion`	`#motion` oder `#list`
shader	`#shader`	`#shader` oder `#list`
Textur	`#texture`	`#texture` oder `#list`
Gruppe	`#group`	`#group`
camera	`#camera`	`#camera`
Kollisionsdaten	`#collisiondata`	`#collisiondata`
vector	`#vector`	`#vector`
transform	`#transform`	`#transform`

Parameter

object Erforderlich. Gibt das zu testende Objekt an.

type Optional. Gibt den Typ an, mit dem *object* verglichen wird. Gehört das Objekt dem angegebenen Typ an, gibt die Funktion `ilk()TRUE` zurück. Gehört das Objekt nicht dem angegebenen Typ an, gibt die Funktion `ilk()FALSE` zurück.

Beispiel

Die folgende Anweisung zeigt, dass `MyObject` ein Bewegungsobjekt ist:

```
put MyObject.ilk
-- #motion
```

Die folgende Anweisung testet, ob `MyObject` ein Bewegungsobjekt ist. Aus dem Rückgabewert 1 geht hervor, dass dies der Fall ist.

```
put MyObject.ilk(#motion)
-- 1
```

Siehe auch

`tweenMode`

image()

Syntax

```
-- Lingo syntax
image(intWidth, intHeight, intBitDepth, <optional:paletteRef>)

// JavaScript syntax
image(intWidth, intHeight, intBitDepth, <optional:paletteRef>);
```

Beschreibung

Diese Top-Level-Funktion erstellt eine neue Grafik mit vorgegebenen Abmessungen und gibt sie zurück.

Wenn Sie eine neue Grafik mithilfe der Top-Level-Funktion `image()` erstellen, ist die neue Grafik ein eigenständiger Grafikdatensatz, der von allen anderen Grafiken unabhängig ist. Deshalb haben an anderen Grafiken vorgenommene Änderungen keine Auswirkungen auf die neue Grafik.

Wenn Sie auf eine Grafik verweisen, indem Sie eine Variable auf eine Quellgrafik, wie z. B. einen Darsteller oder die Grafik der Bühne, festlegen, enthält diese Variable einen Verweis auf die Quellgrafik. Deshalb wird eine an der Grafik in entweder dem Quellobjekt oder der Variablen vorgenommene Änderung in der anderen Grafik ebenfalls übernommen.

Zur Vermeidung dieses Verhaltens und zum Erstellen einer Kopie einer Grafik, die von der Quellgrafik unabhängig ist, verwenden Sie die `duplicate()`-Methode. Die `duplicate()`-Methode gibt eine Kopie einer Quellgrafik zurück, die alle Werte der Quellgrafik erbt, aber nicht an die Quellgrafik gebunden ist. Deshalb wird eine an entweder der Quellgrafik oder der neuen Kopie der Quellgrafik vorgenommene Änderung in der anderen Grafik nicht übernommen.

Wenn Sie ein Grafikobjekt durch Verweisen auf einen Darsteller erstellen, enthält das neue Objekt einen Verweis auf die Grafik des Darstellers. Änderungen, die an der Grafik vorgenommen werden, werden ebenfalls am Darsteller als auch an allen Sprites vorgenommen, die aus diesem Darsteller erstellt werden.

Wenn Sie ein neues Grafikobjekt erstellen, ist der Standardwert der Hintergrundfarbe Weiß (color(255,255,255)) und der Alphakanal ist vollständig undurchsichtig (color(0,0,0)).

Die Alphakanalfarbe für 100%ige Transparenz ist Weiß (color(255,255,255)). Die Alphakanalfarbe für 100%ige Undurchsichtigkeit ist Schwarz (color(0,0,0)).

Ein Beispiel für `image()` in einem fertigen Film finden Sie im Film „Imaging“ im Ordner „Learning\\Lingo_“ im Director-Anwendungsordner.

Parameter

intWidth Erforderlich. Eine Ganzzahl, die die Breite der neuen Grafik angibt.

intHeight Erforderlich. Eine Ganzzahl, die die Höhe der neuen Grafik angibt.

intBitDepth Erforderlich. Eine Ganzzahl, die die Bittiefe der neuen Grafik angibt. Gültige Werte sind 1, 2, 4, 8, 16 und 32.

Beispiel

Das folgende Beispiel erstellt eine 8-Bit-Grafik, mit einer Breite von 200 Pixel und einer Höhe von 200 Pixel.

```
-- Lingo syntax
objImage = image(200, 200, 8)

// JavaScript syntax
var objImage = image(200, 200, 8);
```

Das folgende Beispiel erstellt eine Grafik durch Verweisen auf die Grafik auf der Bühne.

```
-- Lingo syntax
objImage = _movie.stage.image

// JavaScript syntax
var objImage = _movie.stage.image;
```

Siehe auch

duplicate() (Grafik), fill(), image (Grafik)

importFileInto()

Syntax

```
-- Lingo syntax
memberObjRef.importFileInto(fileOrUrlString, propertyList)

// JavaScript syntax
memberObjRef.importFileInto(fileOrUrlString, propertyList);
```

Beschreibung

Diese Darstellermethode ersetzt den Inhalt eines angegebenen Darstellers durch eine vorgegebene Datei.

Die `importFileInto()`-Methode ist in den folgenden Situationen nützlich.

- Wenn Sie einen Film fertig gestellt haben, können Sie mit dieser Methode extern verknüpfte Medien einbetten, sodass sie während der Projektarbeit bearbeitet werden können.
- Wenn Sie bei der Filmerstellung ein Drehbuch per Lingo oder JavaScript-Syntax erstellen, können Sie mit dieser Methode neuen Darstellern Inhalte zuordnen.
- Wenn Sie Dateien aus dem Internet herunterladen, können Sie mit diesem Befehl die Datei an einer bestimmten URL herunterladen und den Dateinamen für verknüpfte Medien festlegen.

  Hinweis: *Es ist in der Regel jedoch effizienter, eine Datei zuerst mit der `preloadNetThing()`-Methode von einer URL auf einen lokalen Datenträger herunterzuladen und diese dann vom Datenträger aus zu importieren. Die Verwendung von `preloadNetThing()` verringert außerdem mögliche Probleme beim Herunterladen.*
- Mit diesem Befehl können Sie sowohl RTF- als auch HTML-Dokumente in Textdarsteller importieren, wobei Formatierung und Links intakt bleiben.

Die Verwendung von `importFileInto()` in Projektoren kann den verfügbaren Speicher sehr schnell aufbrauchen. Deshalb sollten, wann immer möglich, die gleichen Darsteller für importierte Daten erneut verwendet werden.

In Director und Projektoren lädt `importFileInto()` die Datei automatisch herunter. Im Shockwave Player müssen Sie `preloadNetThing()` aufrufen und auf den erfolgreichen Abschluss des Ladevorgangs warten, bis Sie `importFileInto()` mit der Datei verwenden können.

Parameter

fileOrUrlString Erforderlich. Eine Zeichenfolge, die die Datei angibt, mit der der Inhalt des Darstellers ersetzt wird.

propertyList ist optional und unterstützt folgende Eigenschaften:

`#dither`: 0; nicht rastern (Standardeinstellung).

`#dither`: ungleich 0; rastern.

`#trimWhiteSpace`: 0; weiße Flächen an Außenrändern der Grafik nicht beschneiden.

`#trimWhiteSpace`: ungleich Null; weiße Flächen beschneiden (Standardeinstellung).

`#linked`: 0; als internen Darsteller importieren (Standardeinstellung).

`#linked`: ungleich Null; als verknüpften Darsteller importieren.

`#remapImageToStage`: 0; grafikeigene Tiefe verwenden.

`#remapImageToStage`: ungleich Null; Grafik der Bühnentiefe neu zuordnen (Standardeinstellung).

Beispiel

Die folgende Prozedur ordnet der Variablen `tempURL` eine URL mit einer GIF-Datei zu und importiert anschließend diese Datei mit dem Befehl `importFileInto` in einen neuen Bitmapdarsteller:

```
-- Lingo syntax
on exitFrame
    tempURL = "http://www.dukeOfUrl.com/crown.gif"
    _movie.newMember(#bitmap).importFileInto(tempURL)
end

// JavaScript syntax
function exitFrame() {
    var tempURL = "http://www.dukeOfUrl.com/crown.gif";
    _movie.newMember("bitmap").importFileInto(tempURL);
}
```

Die folgende Anweisung ersetzt den Inhalt des Sounddarstellers „Erinnerung“ durch die Sounddatei „Wind“:

```
-- Lingo syntax
member("Memory").importFileInto("Wind.wav")

// JavaScript syntax
member("Memory").importFileInto("Wind.wav");
```

Die folgenden Anweisungen laden eine externe Datei von einer URL in den Anwendungsordner von Director und importieren anschließend diese Datei in den Sounddarsteller „Norma Desmond Speaks“:

```
-- Lingo syntax
downLoadNetThing("http://www.cbDeMille.com/Talkies.AIF", _player.applicationPath &
"Talkies.AIF")
member("Norma Desmond Speaks").importFileInto(_player.applicationPath & "Talkies.AIF")

// JavaScript syntax
downLoadNetThing("http://www.cbDeMille.com/Talkies.AIF",_player.applicationPath +
"Talkies.AIF");
member("Norma Desmond Speaks").importFileInto(_player.applicationPath +"Talkies.AIF");
```

Siehe auch

`downloadNetThing`, `fileName (Fenster)`, `Member`, `preloadNetThing()`

Initialize

Syntax

```
Initialize (MUIObject, initialPropertyList)
```

Beschreibung

Dieser Befehl richtet ein allgemeines Dialogfeld aus einer Instanz des Xtras MUI ein. `initialPropertyList` ist eine Eigenschaftenliste, die angibt, wo Director Definitionen für die Attribute des Dialogfelds abruft.

- Die der Eigenschaft `#windowPropList` zugeordnete Eigenschaftenliste ist die Liste, die Director für die Definition der Attribute des allgemeinen Dialogfelds nutzt.

- Die der Eigenschaft `#windowItemList` zugeordnete lineare Liste ist die Liste, die Director für die Definition einzelner Komponenten nutzt. Jedes Element in der Liste ist eine Eigenschaftenliste, die eine Komponente definiert.

Beispiel

Diese Anweisung initialisiert ein allgemeines Dialogfeld, das anhand von `MUIObject`, einer Instanz des Xtra MUI, erstellt wird. Die Liste `aWindowPropList` enthält Definitionen für das allgemeine Dialogfeld. Die Liste `aWindowItemList` enthält Definitionen der einzelnen Komponenten des Dialogfelds:

```
-- Lingo
Initialize(MUIObj, [#windowPropList:aWindowPropList, \
#windowItemList:aWindowItemList])
```

insertBackdrop

Syntax

```
sprite(whichSprite).camera{(index)}.insertBackdrop(index, texture, locWithinSprite, rotation)
member(whichCastmember).camera(whichCamera).insertBackdrop(index, texture, locWithinSprite,
rotation)
```

Beschreibung

Dieser 3D-Kamerabefehl fügt an einer angegebenen Position in der Liste einen Hintergrund zur Hintergrundliste der Kamera hinzu.

Parameter

index Erforderlich. Gibt die Indexposition in der Hintergrundliste der Kamera an, an der der Hintergrund hinzugefügt wird.

texture Erforderlich. Gibt die Textur des hinzugefügten Hintergrunds an.

locWithinSprite Erforderlich. Eine 2D-Position, an der der Hintergrund im 3D-Sprite angezeigt wird. Die Position wird von der linken oberen Ecke des Sprites an gemessen.

rotation Optional. Gibt die Drehung des hinzugefügten Hintergrunds an.

Beispiel

In der ersten Zeile in diesem Beispiel wird eine Textur namens "Zeder" erstellt. In der zweiten Zeile wird diese Textur an der ersten Position in der Hintergrundliste der Kamera von Sprite 5 eingefügt. Der Hintergrund wird am Punkt (300, 120) positioniert (gemessen von der oberen linken Ecke des Sprite) und um 45° gedreht.

```
-- Lingo
t1 = member("scene").texture("Cedar")
sprite(5).camera.insertBackdrop(1, t1, point(300, 120), 45)

// Javascript
Var t1= member("scene").getProp("texture",1);
Sprite(5).getPropRef("camera",1).insertBackDrop(1,t1,point(300,120),45);
```

Siehe auch

removeBackdrop, bevelDepth, overlay, backdrop

insertFrame()

Syntax

```
-- Lingo syntax
_movie.insertFrame()

// JavaScript syntax
_movie.insertFrame();
```

Beschreibung

Mit dieser Filmmethode wird das aktuelle Bild und dessen Inhalt dupliziert.

Das duplizierte Bild wird nach dem aktuellen Bild eingefügt und wird dann zum aktuellen Bild.

Diese Methode kann nur während der Drehbuchaufzeichnung verwendet werden und führt dieselbe Funktion aus wie die `duplicateFrame()`-Methode.

Parameter

Keiner

Beispiel

Die folgende Prozedur erstellt ein Bild, dem der Übergangsdarsteller `Fog` im Übergangskanal zugeordnet ist, gefolgt von einem Satz leerer Bilder. Das Argument `numberOfFrames` bestimmt die Anzahl der Bilder.

```
-- Lingo syntax
on animBall(numberOfFrames)
    _movie.beginRecording()
    _movie.frameTransition = member("Fog").number
    _movie.go(_movie.frame + 1)
    repeat with i = 0 to numberOfFrames
        _movie.insertFrame()
    end repeat
    _movie.endRecording()
end animBall

// JavaScript syntax
function animBall(numberOfFrames) {
    _movie.beginRecording();
    _movie.frameTransition = member("Fog").number;
    _movie.go(_movie.frame + 1);
    for (var i = 0; i <= numberOfFrames; i++) {
        _movie.insertFrame();
    }
    _movie.endRecording();
}
```

Siehe auch

duplicateFrame(), Movie

insertOverlay

Syntax

```
sprite(whichSprite).camera{(index)}.insertOverlay(index, texture, locWithinSprite, rotation)
member(whichCastmember).camera(whichCamera).insertOverlay(index, texture, locWithinSprite,
rotation)
```

Beschreibung

Dieser 3D-Kamerabefehl fügt an einer angegebenen Position in der Liste eine Überlagerung zur Liste der Überlagerungen der Kamera hinzu.

Parameter

index Erforderlich. Gibt die Indexposition in der Hintergrundliste der Kamera an, an welcher der Hintergrund hinzugefügt wird.

texture Erforderlich. Gibt die Textur der hinzugefügten Überlagerung an.

locWithinSprite Erforderlich. Eine 2D-Position, an der die Überlagerung im 3D-Sprite angezeigt wird. Die Position wird von der linken oberen Ecke des Sprites an gemessen.

rotation Optional. Gibt die Drehung der hinzugefügten Überlagerung an.

Beispiel

In der ersten Zeile in diesem Beispiel wird eine Textur namens "Zeder" erstellt. In der zweiten Zeile wird diese Textur an der ersten Position in der Überlagerungsliste der Kamera von Sprite 5 eingefügt. Die Überlagerung wird am Punkt (300, 120) positioniert (gemessen von der oberen linken Ecke des Sprite) und um 45° gedreht.

```
-- Lingo
t1 = member("scene").texture("Cedar")
sprite(5).camera.insertOverlay(1, t1, point(300, 120), 45)

// Javascript
Var t1=member("scene").getProp("texture",1);
Sprite(5).getPropRef("camera",1).insertOverlay(1,t1,point(300,120),45);
```

Siehe auch

removeOverlay, overlay, bevelDepth

inside()

Syntax

```
point.inside(rectangle)
inside(point, rectangle)
```

Beschreibung

Diese Funktion bestimmt, ob ein angegebener Punkt innerhalb eines vorgegebenen Rechtecks (TRUE) oder außerhalb des Rechtecks liegt (FALSE).

Parameter
rectangle Erforderlich. Gibt das Rechteck an, das den zu testenden Punkt enthält.

Beispiel
Die folgende Anweisung gibt an, ob der Punkt "Mitte" innerhalb des Rechtecks "Zone" liegt, und zeigt das Ergebnis im Nachrichtenfenster an:

```
-- Lingo
put Center.inside(Zone)

// Javascript
trace(Center. inside(Zone));
```

Siehe auch
`map()`, `mouseH`, `mouseV`, `point()`

installMenu

Syntax
`installMenu whichCastMember`

Beschreibung
Playermethode, mit der das Menü, das im durch *whichCastMember* bezeichneten Felddarsteller definiert ist, installiert ist. Diese benutzerdefinierten Menüs erscheinen nur, während der Film abgespielt wird. Wenn Sie die benutzerdefinierten Menüs entfernen wollen, verwenden Sie den Befehl `installMenu` ohne Argument oder mit dem Argument 0. Bei hierarchischen Menüs funktioniert dieser Befehl nicht.

Erläuterungen zum Definieren von Menüeinträgen in einem Felddarsteller finden Sie unter dem Schlüsselwort `menu`.

Vermeiden Sie häufige Menüänderungen, weil sich dies negativ auf die Systemressourcen auswirkt.

Falls das Menü länger als der Bildschirm ist, erscheint in Windows nur ein Teil des Menüs, während auf dem Mac durch das Menü gerollt werden kann.

Hinweis: *Im Shockwave Player sind keine Menüs verfügbar.*

Parameter
fieldMemberObjRef Optional. Gibt den Felddarsteller an, in dem ein Menü installiert wird.

Beispiel
Die folgende Anweisung installiert das in Felddarsteller 37 definierte Menü:

```
installMenu 37
```

Die folgende Anwendung installiert das im Felddarsteller „Menüleiste“ definierte Menü:

```
installMenu member "Menubar"
```

Die folgende Anweisung deaktiviert Menüs, die mit dem Befehl `installMenu` installiert wurden:

```
installMenu 0
```

Siehe auch

menu

integer()

Syntax

```
(numericExpression).integer
integer(numericExpression)
```

Beschreibung

Diese Funktion (nur Lingo) rundet den Wert eines Ausdrucks auf die nächste Ganzzahl auf oder ab.

Sie können eine Ganzzahl mit der Funktion `string()` zu einem String machen.

In JavaScript-Syntax wird die `parseInt()`-Funktion verwendet.

Parameter

numericExpression Erforderlich. Die auf eine Ganzzahl zu rundende Zahl.

Beispiel

Die folgende Anweisung rundet die Zahl 3.75 auf die nächste Ganzzahl auf:

```
put integer(3.75)
-- 4
```

Die folgende Anweisung rundet den Wert in Klammern. Sie erhalten so einen verwendbaren Wert für die Sprite-Eigenschaft locH, für die ein ganzzahliger Wert erforderlichist:

```
sprite(1).locH = integer(0.333 * stageWidth)
```

Siehe auch

float(), string()

integerP()

Syntax

```
expression.integerP
(numericExpression).integerP
integerP(expression)
```

Beschreibung

Diese Funktion (nur Lingo) gibt an, ob ein angegebener Ausdruck zu einer Ganzzahl ausgewertet werden kann (1 oder `TRUE`) oder nicht (0 oder `FALSE`). Das *P* in `integerP` steht für *predicate* (Prädikat).

Parameter

expression Erforderlich. Der zu testende Ausdruck.

Beispiel

Die folgende Anweisung prüft, ob die Zahl 3 als Ganzzahl ausgewertet werden kann, und zeigt dann 1 (TRUE) im Nachrichtenfenster an:

```
put(3).integerP
-- 1
```

Die folgende Anweisung prüft, ob die Zahl 3 als Ganzzahl ausgewertet werden kann. Da die 3 in Anführungszeichen steht, kann sie nicht als Ganzzahl ausgewertet werden, daher erscheint 0 (FALSE) im Nachrichtenfenster:

```
put("3").integerP
-- 0
```

Die folgende Anweisung prüft, ob der numerische Wert des Strings im Felddarsteller „Eintrag“ eine Ganzzahl ist, und blendet einen Warnhinweis ein, wenn dies nicht der Fall ist:

```
if field("Entry").value.integerP = FALSE then alert "Please enter an integer."
```

Siehe auch

floatP(), integer(), ilk(), objectP(), stringP(), symbolP()

Interface()

Syntax

```
xtra("XtraName").Interface()
Interface(xtra "XtraName")
```

Beschreibung

Diese Funktion gibt einen durch einen Zeilenumbruch begrenzten String zurück, der das Xtra beschreibt und dessen Methoden aufführt. Diese Funktion ersetzt die jetzt veraltete Funktion `mMessageList`.

Parameter

Keiner

Beispiel

Die folgende Anweisung zeigt die Ausgabe der Funktion, die im QuickTime Asset-Xtra verwendet wurde, im Nachrichtenfenster an:

```
-- Lingo
put Xtra("QuickTimeSupport").Interface()

// Javascript
trace(xtra("QuickTimeSupport").Interface());
```

interpolate()

Syntax

```
transform1.interpolate(transform2,percentage)
```

Beschreibung

Diese 3D-`transform`-Methode gibt eine Kopie von *transform1* zurück. Zu ihrer Erstellung wird um den angegebenen Prozentsatz von der Position und Drehung von *transform1* auf die Position und Drehung von *transform2* interpoliert. Die ursprüngliche Version von *transform1* bleibt unverändert. Um *transform1* zu interpolieren, verwenden Sie `interpolateTo()`.

Um eine manuelle Interpolation durchzuführen, multiplizieren Sie die Differenz der zwei Zahlen mit dem Prozentwert. Eine Interpolation von 4 auf 8 um 50% ergibt beispielsweise 6.

Beispiel

In diesem Beispiel ist "tBox" die Transformation des Modells "Box" und "tKugel" die Transformation des Modells "Kugel". In der dritten Zeile dieses Beispiels wird eine Kopie der Transformation von "Box" 50% bis zur Transformation von "Kugel" interpoliert.

```
-- Lingo
tBox = member("3d world").model("Box").transform
tSphere = member("3d world").model("Sphere").transform
tNew = tBox.interpolate(tSphere, 50)

// Javascript
var tBox = member("3d world").getPropRef("model",a).transform;
// where a is the number index for the model "Box"
var tSphere = member("3d world").getPropRef("model", i).transform;
// where i is the number index for the model "Sphere"
var tNew = tBox.interpolate(tSphere, 50);
```

Siehe auch

`interpolateTo()`

interpolateTo()

Syntax

```
transform1.interpolateTo(transform2, percentage)
```

Beschreibung

Diese 3D-`transform`-Methode modifiziert *transform1* durch Interpolation von der Position und Drehung von*transform1* auf die Position und Drehung einer neuen Transformation um einen angegebenen Prozentsatz. Die ursprüngliche Version von *transform1* wird entsprechend geändert. Um eine Kopie von *transform1* zu interpolieren, verwenden Sie `interpolate()`.

Um eine manuelle Interpolation durchzuführen, multiplizieren Sie die Differenz der zwei Zahlen mit dem Prozentwert. Eine Interpolation von 4 auf 8 um 50% ergibt beispielsweise 6.

Parameter

transform2 Erforderlich. Gibt die Transformation an, auf die eine angegebene Transformation interpoliert wird.

percentage Erforderlich. Gibt den Prozentsatz der Drehung von *transform2* an.

Beispiel

In diesem Beispiel ist "tBox" die Transformation des Modells "Box" und "tKugel" die Transformation des Modells "Kugel". In der dritten Zeile dieses Beispiels wird die Transformation von „Box“ 50% bis zur Transformation von "Kugel" interpoliert.

```
-- Lingo
tBox = member("3d world").model("Box").transform
tSphere = member("3d world").model("Sphere").transform
tBox.interpolateTo(tSphere, 50)

// Javascript
var tBox = member("3d world").getPropRef("model", i).transform;
// where i the number index for the model "Box"
var tSphere = member("3d world").getPropRef("model",j).transform;
// where j is the number index for the model "Sphere"
tBox.interpolateTo(tSphere, 50);
```

Siehe auch

interpolate()

intersect()

Syntax

```
rectangle1. Intersect(rectangle2)
intersect(rectangle1, rectangle2)
```

Beschreibung

Diese Funktion bestimmt das Rechteck, das sich ergibt, wenn sich zwei Rechtecke schneiden.

Parameter

rectangle2 Erforderlich. Gibt das zweite am Schnittpunkttest beteiligte Rechteck an.

Beispiel

Die folgende Anweisung ordnet die Variable `newRectangle` dem Rechteck zu, das sich ergibt, wenn sich das Rechteck `toolKit` mit dem Rechteck `Ramp` schneidet:

```
-- Lingo
newRectangle = toolKit.intersect(Ramp)

// Javascript
newRectangle = toolKit.intersect(Ramp);
```

Siehe auch

map(), rect(), union()

inverse()

Syntax

```
member(whichCastmember).model(whichModel).transform.inverse()
member(whichCastmember).group(whichGroup).transform.inverse()
member(whichCastmember).camera(whichCamera).transform.inverse()
sprite(whichSprite).camera{(index)}.transform.inverse()
member(whichCastmember).light(whichLight).transform.inverse()
transformReference.inverse()
```

Beschreibung

Diese 3D-`transform`-Methode gibt eine Kopie der Transformation zurück, bei der die Positions- und Rotationseigenschaften umgekehrt sind.

Die ursprüngliche Transformation wird durch diese Methode nicht geändert. Um die ursprüngliche Transformation umzukehren, verwenden Sie `invert()`.

Parameter

Keiner

Beispiel

Die folgende Anweisung kehrt eine Kopie der Transformation des Modells „Stuhl“ um

```
-- Lingo
boxInv = member("3d world").model("Chair").transform.inverse()

// Javascript
var boxInv = member("3d world").getPropRef("model", a).transform.inverse();
// where a the number index for the model "Chair"
```

Siehe auch

`invert()`

invert()

Syntax

```
member(whichCastmember).model(whichModel).transform.invert()
member(whichCastmember).group(whichGroup).transform.invert()
member(whichCastmember).camera(whichCamera).transform.invert()
sprite(whichSprite).camera{(index)}.transform.invert()
member(whichCastmember).light(whichLight).transform.invert()
transformReference.invert()
```

Beschreibung

Diese 3D-`transform`-Methode kehrt die Positions- und Rotationseigenschaften der Transformation um.

Durch diese Methode wird die ursprüngliche Transformation geändert. Um eine Kopie der ursprünglichen Transformation umzukehren, verwenden Sie `inverse()`.

Parameter
Keiner

Beispiel

```
-- Lingo
member("3d world").model("Box").transform.invert()

// Javascript
member("3d world").getPropRef("model", a).transform.invert();
// where a the number index for the model "Box"
```

Siehe auch
inverse()

isBusy()

Syntax

```
-- Lingo syntax
soundChannelObjRef.isBusy()

// JavaScript syntax
soundChannelObjRef.isBusy();
```

Beschreibung
Diese Soundkanalmethode bestimmt, ob ein Sound in einem Soundkanal abgespielt wird (TRUE)oder nicht (FALSE).

Stellen Sie sicher, dass sich der Abspielkopf weiterbewegt hat, bevor Sie den Soundkanal mit isBusy() überprüfen. Sollte diese Funktion weiter FALSE zurückgeben, obwohl ein Sound jetzt abgespielt werden müsste, fügen Sie die updateStage()-Methode hinzu, damit der Sound abgespielt wird, bevor der Abspielkopf sich erneut weiterbewegt.

Diese Methode gilt für Soundkanäle mit echten Audiodarstellern. QuickTime, Flash und Shockwave Player-Audio verarbeiten Sound anders; daher kann diese Funktion nicht für diese Medientypen verwendet werden.

In diesem Fall sollten Sie die Eigenschaft status eines Soundkanals verwenden statt isBusy(). Die Eigenschaft Status ist in vielen Fällen genauer.

Parameter
Keiner

Beispiel
Die folgende Anweisung prüft, ob in Soundkanal 1 ein Sound abgespielt wird, und führt dann eine Schleife im Bild aus, wenn das der Fall ist. Auf diese Weise kann der Sound zu Ende gespielt werden, bevor sich der Abspielkopf zum nächsten Bild bewegt.

```
-- Lingo syntax
if (sound(1).isBusy()) then
    _movie.go(_movie.frame)
end if

// JavaScript syntax
if (sound(1).isBusy()) {
    _movie.go(_movie.frame);
}
```

Siehe auch

status, Sound Channel

isCharSetInstalled

Syntax

```
_system.isCharSetInstalled(strCharSet)
```

Beschreibung

Diese Zeichensatzmethode prüft, ob der angegebene Zeichensatz auf einem Computer installiert ist. Diese Methode gibt einen boole'schen Wert zurück.

Parameter

Parameter	Beschreibung	Erforderlich/Optional
strCharSet	Zeichensatz, auf den geprüft werden soll. Beispiel: „Windows-1252“.	Erforderlich

Beispiele

```
--Lingo syntax
put _system.isCharSetInstalled("windows-1252")

//JavaScript syntax
put(_system.isCharSetInstalled("windows-1252"));
```

isInWorld()

Syntax

```
member(whichCastmember).model(whichModel).isInWorld()
member(whichCastmember).camera(whichCamera).isInWorld()
member(whichCastmember).light(whichLight).isInWorld()
member(whichCastmember).group(whichGroup).isInWorld()
```

Beschreibung

Dieser 3D-Befehl gibt `TRUE` zurück, wenn die Parent-Hierarchie des Modells, der Kamera, des Lichts oder der Gruppe in der Welt endet. Wenn `isInWorld` den Wert `TRUE` aufweist, ist das Modell, die Kamera, das Licht oder die Gruppe in der 3D-Welt des Darstellers funktionsfähig.

Modelle, Kameras, Lichtquellen und Gruppen können in einem 3D-Darsteller gespeichert sein, ohne in der 3D-Welt des Darstellers zum Einsatz zu kommen. Mit den Befehlen `addToWorld` und `removeFromWorld` können Sie Modelle, Kameras, Lichtquellen und Gruppen zur 3D-Welt des Darstellers hinzufügen bzw. daraus entfernen.

Parameter

Keiner

Beispiel

Die folgende Anweisung zeigt, dass das Modell `Teapot` in der 3D-Welt des Darstellers `TableScene` vorhanden ist:

```
--Lingo
put member("TableScene").model("Teapot").isInWorld()

// Javascript
put member("TableScene").getPropRef("model", a).isInWorld() ;
// where a is the member index for the Teapot model.
-- 1
```

Siehe auch

`addToWorld`, `removeFromWorld`, `child (3D)`

isPastCuePoint()

Syntax

```
-- Lingo syntax
spriteObjRef.isPastCuePoint(cuePointID)

// JavaScript syntax
spriteObjRef.isPastCuePoint(cuePointID);
```

Beschreibung

Diese Funktion gibt an, ob ein Sprite- oder Soundkanal einen bestimmten Aufrufpunkt in den zugehörigen Medien passiert hat. Diese Funktion kann bei Sound- (WAV, AIFF, SND, SWA, AU), QuickTime- und Xtra-Darstellern benutzt werden, die Aufrufpunkte unterstützen.

Ersetzen Sie *spriteNum* oder *channelNum* durch einen Sprite- oder Soundkanal. Shockwave Audio (SWA)-Sounds können in Sprite-Kanälen als Sprites erscheinen, die Soundwiedergabe erfolgt jedoch in einem Soundkanal. Es wird empfohlen, mit der Nummer des Sprite-Kanals und nicht mit der Nummer des Soundkanals auf SWA-Sound-Sprites Bezug zu nehmen.

Ersetzen Sie *cuePointID* durch einen Bezug auf einen Aufrufpunkt:

- Wenn *cuePointID* eine Ganzzahl ist, gibt`isPastCuePoint` 1 zurück, wenn der Aufrufpunkt passiert wurde, und 0, wenn er nicht passiert wurde.
- Wenn *cuePointID* ein Name ist, gibt `isPastCuePoint` die Anzahl der passierten Aufrufpunkte mit diesem Namen zurück.

Wenn der für *cuePointID* angegebene Wert im Sprite oder Sound nicht existiert, gibt die Funktion 0 zurück.

Die von `isPastCuePoint` zurückgegebene Nummer basiert auf der absoluten Position des Sprites in seinen Medien. Wenn ein Sound zum Beispiel den Aufrufpunkt „Main“ passiert, dann eine Schleife durchläuft und "Main" erneut passiert, gibt `isPastCuePoint` 1 an Stelle von 2 zurück.

Wenn das Resultat von `isPastCuePoint` als boolescher Operator behandelt wird, gibt die Funktion `TRUE` zurück, falls durch *cuePointID* identifizierte Aufrufpunkte passiert wurden. Wenn nicht, wird `FALSE` zurückgegeben.

Parameter

cuePointID Erforderlich. Eine Zeichenfolge oder Ganzzahl, die den Namen oder die Nummer des angegebenen Aufrufpunkts angibt.

Beispiel

Die folgende Anweisung spielt einen Sound so lange ab, bis der Aufrufpunkt „Refrain-Ende“ zum dritten Mal passiert wurde:

```
-- Lingo syntax
if (sound(1).isPastCuePoint("Chorus End")=3) then
    sound(1).stop()
end if

// JavaScript syntax
var ce = sound(1).isPastCuePoint("Chorus End");
if (ce = 3) {
    sound(1).stop();
}
```

Die folgende Anweisung zeigt im Darsteller „Feld 2“ Informationen zu der in Soundkanal 1 abgespielten Musik an. Wenn die Musik den Aufrufpunkt „Höhepunkt“ noch nicht passiert hat, lautet der Text im Darsteller „Dies ist der Anfang des Stücks“. Andernfalls lautet der Text „Dies ist das Ende des Stücks.“

```
--- Lingo syntax
if not sound(1).isPastCuePoint("climax") then
    member("field 2").text = "This is the beginning of the piece."
else
    member("field 2").text = "This is the end of the piece."
end if

// JavaScript syntax
var cmx = sound(1).isPastCuePoint("climax");
if (cmx != 1) {
    member("field 2").text = "This is the beginning of the piece.";
} else {
    member("field 2").text = "This is the end of the piece.";
}
```

ItemUpdate()

Syntax

```
ItemUpdate(MUIObject, itemNumber, itemInputPropList)
```

Beschreibung

Dieser Befehl aktualisiert eine Komponente in einem allgemeinen Dialogfeld. Er eignet sich zum Aktualisieren eines Dialogfelds als Reaktion auf Benutzeraktionen, solange das Dialogfeld angezeigt wird.

- `itemNumber` ist die Anzahl der aktualisierten Elemente.
- `itemInputPropList` ist die Liste der neuen Eigenschaften des Elements.

Der Befehl `ItemUpdate` kann für viele Zwecke eingesetzt werden, z. B. zum Aktivieren bzw. Deaktivieren von Schaltflächen, Ändern des Bereichs eines Popupfensters bzw. der Position von Schiebereglern sowie zum Modifizieren bearbeitbarer Textelemente, wenn der Benutzer einen ungültigen Wert eingibt.

Sie können einzelne Elemente in einem Dialogfeld abhängig von der Benutzereingabe oder Benutzerinteraktionen aktualisieren oder zugrunde liegende Daten anzeigen. Wenn gleich Sie zumeist den Parameter `#value` eines Elements ändern, können Sie außer dem Typ alle Elementattribute ändern. Stellen Sie die Eigenschaften `height`,`width`, `locH` und `locV` auf „-1“ ein, um deren aktuelle Werte beizubehalten.

Beispiel

Folgende Anweisungen aktualisieren die Dialogfeld-Komponente mit der Nummer `itemNum`.

- Die erste Anweisung bezieht die Definition der Komponente aus der Liste aller Elementdefinitionen.
- Die zweite und dritte Anweisung ändert die Eigenschaften „Typ“ und „Attribut“ der Komponenten.
- In der letzten Anweisung werden mit dem Befehl `ItemUpdate` die Einstellungen der Komponente aktualisiert.

```
--Lingo
set baseItemList = getAt ( theItemList, itemNum )
set the type of baseItemList = #IntegerSliderH
set the attributes of baseItemList = [#valueRange :[#min:1, #max:8, #increment:1,
#jump:1, #acceleration:1]
ItemUpdate(MUIObj, itemNum, baseItemList)
on smileyUpdate
    -- declare globals
    global smileyIndex, gMuiSmileDialObj, itemNumSmile, itemNumSlide, smileItemList
-- validate dialog object
if ( objectP ( gMuiSmileDialObj ) ) then
    -- get a list to put in new/updated values
    set baseItemList = duplicate ( getAt ( smileItemList,itemNumSmile ) )
    -- metrics can be set to -1, this "keeps them the same"
    -- instead of updating.
    -- could also be set to a new value if you
    -- wanted to resize the item or relocate it.
    set the width of baseItemList = -1 -- keep previous
    set the height of baseItemList = -1 -- keep previous
    set the locH of baseItemList = -1 -- keep previous
    set the locV of baseItemList = -1 -- keep previous
    -- in this particular case, the value is
    -- the only thing that's changing
    set the value of baseItemList = string(smileyIndex)
    -- member name
    -- tell the dialog to update the item number
    -- with the new item list
    ItemUpdate(gMuiSmileDialObj, itemNumSmile, baseItemList)
    end if
end
```

keyPressed()

Syntax

```
-- Lingo syntax
_key.keyPressed({keyCodeOrCharacter})

// JavaScript syntax
_key.keyPressed({keyCodeOrCharacter});
```

Beschreibung

Diese Tastenmethode gibt das Zeichen als Zeichenfolge (string) zurück, das der zuletzt gedrückten Taste zugeordnet ist, bzw. optional ob eine vorgegebene Taste gedrückt wurde.

Wird der Parameter *keyCodeOrCharacter* nicht angegeben, gibt die Methode das Zeichen als Zeichenfolge zurück, das der zuletzt gedrückten Taste zugeordnet ist. Wenn keine Taste gedrückt wurde, gibt diese Methode eine leere Zeichenfolge zurück.

Wird mithilfe von *keyCodeOrCharacter* die gedrückte Taste angegeben, gibt diese Methode `TRUE` zurück, wenn die angegebene Taste gedrückt wurde, bzw. `FALSE`, wenn nicht.

Diese Methode wird aktualisiert, wenn der Benutzer Tasten drückt, während sich Lingo in einer `repeat`-Schleife bzw. JavaScript-Syntax in einer `for`-Schleife befindet. Dies ist ein Vorteil im Vergleich zur Eigenschaft `key`, die nicht aktualisiert wird, wenn sich der Skriptcode in einer `repeat`- bzw. `for`-Schleife befindet.

Mit dem Beispielfilm „Tastatur-Lingo“ können Sie prüfen, welche Zeichen den verschiedenen Tasten auf unterschiedlichen Tastaturen entsprechen.

Parameter

keyCodeOrCharacter Optional. Der Tastencode oder das ASCII-Zeichen als Zeichenfolge, der bzw. das getestet werden soll.

Beispiel

Die folgende Anweisung prüft, ob der Benutzer die Eingabetaste gedrückt hat, und führt in diesem Fall die Prozedur `updateData` aus:

```
-- Lingo syntax
if (_key.keyPressed(RETURN)) then
    updateData
end if

// JavaScript syntax
if (_key.keyPressed(36)) {
    updateData();
}
```

Die folgende Anweisung prüft mit dem Tastencode `keyCode` für die Taste *a*, ob sie gedrückt ist, und zeigt das Ergebnis im Nachrichtenfenster an:

```
-- Lingo syntax
if (_key.keyPressed(0)) then
    put("The key is down")
end if

// JavaScript syntax
if (_key.keyPressed(0)) {
    put("The key is down");
}
```

Die folgende Anweisung prüft anhand der ASCII-Strings, ob die Tasten *a* und *b* gedrückt sind, und zeigt das Ergebnis im Nachrichtenfenster an:

```
-- Lingo syntax
if (_key.keyPressed("a") and _key.keyPressed("b")) then
    put("Both keys are down")
end if

// JavaScript syntax
if (_key.keyPressed("a") && _key.keyPressed("b")) {
    put("Both keys are down");
}
```

Siehe auch

Taste, key, keyCode

label()

Syntax

```
-- Lingo syntax
_movie.label(stringMarkerName)

// JavaScript syntax
_movie.marker(stringMarkerName);
```

Beschreibung

Diese Filmmethode gibt das Bild an, das einer Markierungsbeschriftung zugeordnet ist.

Der Parameter *stringMarkerName* muss eine im aktuellen Film vorhandene Beschriftung sein, andernfalls gibt die Methode „0“ zurück.

Parameter

stringMarkerName Erforderlich. Eine Zeichenfolge, die den Namen der Markierungsbeschriftung angibt, die einem Bild zugeordnet ist.

Beispiel

Die folgende Anweisung schickt den Abspielkopf zum zehnten Bild nach dem Bild mit der Beschriftung „Start“:

```
-- Lingo syntax
_movie.go(_movie.label("Start") + 10)

// JavaScript syntax
_movie.go(_movie.marker("Start") + 10);
```

Siehe auch

frameLabel, go(), labelList, Movie

last()

Syntax

```
the last chunk of (chunkExpression)
the last chunk in (chunkExpression)
```

Beschreibung

Diese Funktion gibt den letzten Chunk in einem Chunk-Ausdruck an.

Chunk-Ausdrücke verweisen auf beliebige Zeichen, Wörter, Elemente oder Zeilen in einem Zeichencontainer. Die unterstützten Container sind Felddarsteller, Variablen, die Strings enthalten, sowie bestimmte Zeichen, Wörter, Elemente, Zeilen und Bereiche innerhalb von Containern.

Parameter

stchunkExpressionring Erforderlich. Gibt den Chunk-Ausdruck an, der den letzten Chunk enthält.

Beispiel

Die folgende Anweisung identifiziert das letzte Wort im String „Adobe, die Multimediafirma“ und zeigt das Ergebnis im Nachrichtenfenster an:

```
-- Lingo
put the last word of "Adobe, the multimedia company"
```

Das Ergebnis ist das Wort *Multimediafirma*.

Die folgende Anweisung identifiziert das letzte Zeichen im String „Adobe, die Multimediafirma“ und zeigt das Ergebnis im Nachrichtenfenster an:

```
put the last char of ("Adobe, the multimedia company")
```

Das Ergebnis ist der Buchstabe *a*.

Siehe auch

char...of, word...of

lastClick()

Syntax

```
the lastClick
```

Beschreibung
Diese Funktion gibt die in Ticks (1/60 Sekunde) gemessene Zeit zurück, die seit der letzten Betätigung der Maustaste vergangen ist.

Diese Funktion kann getestet, aber nicht eingestellt werden.

Parameter
Keiner

Beispiel
Die folgende Anweisung prüft, ob seit dem letzten Mausklick 10 Sekunden vergangen sind, und schickt in diesem Fall den Abspielkopf zur Markierung "Kein Klick":

```
if the lastClick > 10 * 60 then go to "No Click"
```

Siehe auch
`lastEvent()`, `lastKey`, `lastRoll`, `milliseconds`

lastEvent()

Syntax
```
the lastEvent
```

Beschreibung
Diese Funktion gibt die in Ticks (1/60 Sekunde) gemessene Zeit zurück, die seit dem letzten Mausklick, Rollover oder Tastenanschlag vergangen ist.

Parameter
Keiner

Beispiel
Die folgende Anweisung prüft, ob seit dem letzten Mausklick, Rollover oder Tastenanschlag 10 Sekunden vergangen sind, und schickt in diesem Fall den Abspielkopf zur Markierung "Hilfe":

```
if the lastEvent > 10 * 60 then go to "Help"
```

Siehe auch
`lastClick()`, `lastKey`, `lastRoll`, `milliseconds`

length()

Syntax
```
string.length
length(string)
```

Beschreibung

Diese Funktion gibt die Anzahl der Zeichen in dem durch string *angegebenen String, einschließlich Leerstellen und Steuerzeichen (z.*B. Tabulator und Zeilenumbruch), zurück.

Parameter

Keiner

Beispiel

Die folgende Anweisung zeigt die Anzahl der Zeichen im String „Macro“&„media“ an:

```
put ("Macro" & "media").length
-- 10
```

Die folgende Anweisung prüft, ob der Inhalt des Felddarstellers „Dateiname“ mehr als 31 Zeichen enthält, und zeigt einen Warnhinweis an, falls dies der Fall ist:

```
-- Lingo syntax
if member("Filename").text.length > 31 then
    alert "That filename is too long."
end if

// JavaScript syntax
if (member("Filename").text.length > 31) {
    _player.alert("That filename is too long.");
}
```

Siehe auch

chars(), offset() (Zeichenfolgenfunktion)

light()

Syntax

```
member(whichCastmember).light(whichLight)
member(whichCastmember).light[index]
member(whichCastmember).light(whichLight).whichLightProperty
member(whichCastmember).light[index].whichLightProperty
```

Beschreibung

Dieses 3D-Element ist ein Objekt an einer Vektorposition, von der Licht ausstrahlt.

Eine vollständige Liste von Lichteigenschaften und -befehlen finden Sie in den Themen des Director-Benutzerhandbuchs im Director-Hilfefenster.

Beispiel

In diesem Beispiel werden die zwei Möglichkeiten zur Bezugnahme auf ein Licht veranschaulicht. In der ersten Zeile wird ein String in runden Klammern und in der zweiten Zeile eine Zahl in eckigen Klammern verwendet. Der String ist der Name des Lichts und die Nummer die Position des Lichts in der Lichtquellenliste des Darstellers.

```
-- Lingo
thisLight = member("3D World").light("spot01")
thisLight = member("3D World").light[2]

// Javascript
thisLight = member("3D World").light[1];
```

Siehe auch

newLight, deleteLight

lineHeight()

Syntax

```
-- Lingo syntax
memberObjRef.lineHeight(lineNumber)

// JavaScript syntax
memberObjRef.lineHeight(lineNumber);
```

Beschreibung

Diese Funktion gibt die Höhe einer bestimmten Zeile im angegebenen Felddarsteller in Pixeln zurück.

Parameter

lineNumber Erforderlich. Eine Ganzzahl, die die zu messende Zeile angibt.

Beispiel

Die folgende Anweisung bestimmt die Höhe der ersten Zeile im Felddarsteller "Nachrichten" in Pixeln und ordnet das Ergebnis der Variablen headline zu:

```
--Lingo syntax
headline = member("Today's News").lineHeight(1)

// JavaScript syntax
var headline = member("Today's News").lineHeight(1);
```

linePosToLocV()

Syntax

```
-- Lingo syntax
memberObjRef.linePosToLocV(lineNumber)

// JavaScript syntax
memberObjRef.linePosToLocV(lineNumber);
```

Beschreibung

Diese Funktion gibt den Abstand einer bestimmten Zeile vom oberen Rand des Felddarstellers in Pixeln zurück.

Parameter
lineNumber Erforderlich. Eine Ganzzahl, die die zu messende Zeile angibt.

Beispiel
Die folgende Anweisung misst den Abstand zwischen der zweiten Zeile des Felddarstellers „Nachrichten“ und dem oberen Rand des Felddarstellers in Pixeln und ordnet das Ergebnis der Variablen `startOfString` zu:

```
--Lingo syntax
startOfString = member("Today's News").linePosToLocV(2)

// JavaScript syntax
var startOfString = member("Today's News").linePosToLocV(2);
```

linkAs()

Syntax
`castMember.linkAs()`

Beschreibung
Diese Skriptdarstellerfunktion öffnet das Dialogfeld „Speichern“, in dem Sie den Inhalt des Skripts in einer externen Datei speichern können. Anschließend wird der Skriptdarsteller mit dieser Datei verknüpft.

Verknüpfte Skripts werden in den Film importiert, wenn Sie ihn als Projektor oder Film mit Shockwave-Inhalt speichern. Damit unterscheiden sie sich von anderen verknüpften Medien, die extern gespeichert bleiben, bis Sie sie ausdrücklich importieren.

Parameter
Keiner

Beispiel
Die folgenden Anweisungen werden in das Nachrichtenfenster eingegeben und öffnen das Dialogfeld „Speichern“, um das Skript „Zufällige Bewegung“ als externe Datei zu speichern:

```
-- Lingo
member("Random Motion").linkAs()
importFileInto, linked

// Javascript
member("Random Motion").linkAs() ;
```

list()

Syntax

```
-- Lingo syntax
list()
[]
list(stringValue1, stringValue2, ...)
[stringValue1, stringValue2, ...]

// JavaScript syntax
list();
list(stringValue1, stringValue2, ...);
```

Beschreibung

Diese Top-Level-Funktion erstellt eine lineare Liste.

Beim Erstellen einer Liste mithilfe der Syntax `list()`, mit Parametern oder ohne, beginnt der Index der Listenwerte mit 1.

Beim Erstellen einer Liste mithilfe der Syntax `[]`, mit Parametern oder ohne, beginnt der Index der Listenwerte mit 1.

Die maximale Länge einer einzelnen Zeile mit ausführbarem Skriptcode ist auf 256 Zeichen beschränkt. Große Listen können nicht mithilfe von `list()` erstellt werden. Um eine Liste mit einer großen Datenmenge zu erstellen, schließen Sie die Daten in eckige Klammern ([]) ein, legen die Daten in einem Feld ab und weisen dieses Feld dann einer Variablen zu. Der Inhalt der Variablen ist eine Liste der Daten.

Parameter

strigValue1, stringValue2 ... Optional. Eine Liste von Zeichenfolgen, die die Anfangswerte in der Liste angeben.

Beispiel

Die folgende Anweisung stellt die Variable „designers“ so ein, dass sie einer linearen Liste entspricht, die die Namen „Gee“, „Kayne“ und „Ohashi“ enthält:

```
-- Lingo syntax
designers = list("Gee", "Kayne", "Ohashi") -- using list()
designers = ["Gee", "Kayne", "Ohashi"] -- using brackets

// JavaScript syntax
var designers = list("Gee", "Kayne", "Ohashi");
```

Siehe auch

`propList()`

listP()

Syntax

```
listP(item)
```

Beschreibung
Diese Funktion gibt an, ob es sich bei einem angegebenen Element um eine Liste, ein Rechteck oder einen Punkt (1 oder TRUE) oder nicht handelt (0 oder FALSE).

Parameter
item Erforderlich. Gibt das zu testende Element an.

Beispiel
Die folgende Anweisung prüft, ob die Liste in der Variablen designers eine Liste, ein Rechteck oder ein Punkt ist, und zeigt das Ergebnis im Nachrichtenfenster an:

```
put listP(designers)
```

Das Resultat ist 1, das numerische Gegenstück zu TRUE.

Siehe auch
ilk(), objectP()

loadFile()

Syntax
```
member(whichCastmember).loadFile(fileName {, overwrite, generateUniqueNames})
```

Beschreibung
Dieser 3D-Darstellerbefehl importiert die Elemente einer W3D-Datei in einen Darsteller.

Die Darstellereigenschaft `state` muss entweder -1 (Fehler) oder 4 (geladen) lauten, damit der Befehl `loadFile` verwendet werden kann.

Parameter
fileName Erforderlich. Gibt die W3D-Datei an, der die zu importierenden Elemente enthält.

overwrite Optional. Gibt an, ob die Elemente der W3D-Datei die Elemente des Darstellers ersetzen (TRUE) oder zu den Darstellerelementen hinzugefügt werden (FALSE). Der Standardwert der Eigenschaft *overwrite* lautet TRUE.

generateUniqueNames Optional. Wenn der Parameter auf TRUE gesetzt ist, wird jedes Element in der W3D-Datei, das genauso heißt wie das entsprechende Element im Darsteller, umbenannt. Ist der Parameter auf FALSE gesetzt, werden die Elemente im Darsteller mit den entsprechenden gleichnamigen Elementen in der W3D-Datei überschrieben. Der Standardwert der Eigenschaft *generateUniqueNames* lautet TRUE.

Beispiel
Die folgende Anweisung importiert den Inhalt der Datei „Truck.W3d“ in den Darsteller "Autobahn". Der Inhalt von „Truck.W3d“ wird zum Inhalt von „Autobahn“ hinzugefügt. Wenn es importierte Objekte gibt, die den gleichen Namen aufweisen wie bereits in „Autobahn“ vorhandene Objekte, werden sie von Director umbenannt.

```
-- Lingo
member("Roadway").loadFile("Truck.W3d", FALSE, TRUE)

// Javascript
member("Roadway").loadFile("Truck.W3d",false,true);
```

Die folgende Anweisung importiert den Inhalt der Datei „Chevy.W3d“ in den Darsteller „Autobahn“. „Chevy.W3d“ befindet sich in einem Ordner namens „Modelle“, der sich auf einer Ebene unterhalb des Filmordners befindet. Der Inhalt von „Autobahn“ wird durch den Inhalt von „Chevy.W3d“ ersetzt. Der dritte Parameter ist irrelevant, da der Wert des zweiten Parameters TRUE lautet.

```
-- Lingo
member("Roadway").loadFile(the moviePath & "Models\Chevy.W3d", TRUE, TRUE)

// Javascript
member("Roadway").loadFile(_movie.Path + "Models\Chevy.W3d",true,true);
```

Siehe auch

`state (3D)`

loadPolicyFile()

Syntax

```
_movie.loadPolicyFile(URL))
```

Beschreibung

Lädt eine domänenübergreifende Richtlinie von einem Speicherort, der durch den `URL`-Parameter angegeben wird.

Falls beispielsweise der Shockwave-Film `http://www.xyz.com/crossdomain.dcr` versucht, Daten von `http://www.Server1.com/data.txt` zu laden, bestimmt die Cross-Domain-Richtliniendatei auf `http://www.Server1.com`, ob `http://www.xyz.com` Daten von `Server1` abrufen darf.

Folgende Richtliniendatei auf `http://www.Server1.com` berechtigt die Shockwave-Filme auf `http://www.xyz.com`, Daten vom Server abrufen zu können:

```
<cross-domain-policy>
    <allow-access-from domain="http://www.foo.com" to-ports="*" />
  </cross-domain-policy>
```

Die Cross-Domain-Richtliniendatei wird in folgendem Format festgelegt.

```
<cross-domain-policy>
    <allow-access-from domain="*" secure="true" to-ports="*"  />
  </cross-domain-policy>
```

Die Domain-Attribute legen die Domänen fest, die Zugriff auf den Server haben. Der Wert `domain="*"` bewirkt, dass alle Domains Zugriff haben. Der Wert `domain="www.bar.com/xyz/*"` bewirkt, dass alle Domains unter `www.bar.com/xyz/` Zugriff haben.

Das Attribut `secure` bewirkt mit dem Wert „False“, dass eine HTTPS-Richtliniendatei einer HTTP-Anfrage Zugriff gewähren kann. Der Standardwert für das Attribut `secure` ist „True“. Es wird nicht empfohlen, das Attribut auf „False“ zu setzen.

Das Attribut `to-ports` erlaubt das Festlegen eines Portbereichs für den Zugriff der betreffenden Domains auf den Server. Falls `to-ports` = "*" oder `to-ports` im Tag `<allow-access-from>` nicht verwendet wird, sind alle Ports für den Zugriff der Domäne autorisiert. Im `<allow-access-from>`-Tag sind die Attribute `to-port` und `secure` optional.

Folgendes Beispiel zeigt eine Cross-Domain-Richtlinie:

```
<cross-domain-policy>
<allow-access-from domain="www.bar.com" to-ports="1222,8000-9000,9192" />
  </cross-domain-policy>
```

Weitere Informationen über die Cross-Domain-Richtlinie finden Sie unter dem Stichwort „Cross-Domain-Richtlinie“ im Benutzerhandbuch.

Weiterhin:

- Wenn ein Shockwave-Film ohne `loadPolicyFile()` versucht, Daten von einer anderen als seiner eigenen Domäne abzurufen, versucht der Shockwave-Player, die Richtliniendatei `crossdomain.xml` vom Stammverzeichnis des Servers abzurufen. Falls die Richtliniendatei im Stammverzeichnis nicht verfügbar ist, zeigt Director ein Sicherheitsdialogfeld an.
- `loadPolicyFile()` kann nur jeweils eine einzige Richtliniendatei von einem Server abrufen. Falls mehr als eine Instanz von `loadPolicyFile()` verwendet wird, wird nur die erste Datei in der Reihe geladen.

  Im folgenden Code-Abschnitt wird `pf2.xml` nicht geladen:

  ```
  on prepareMovie
  _movie.loadPolicyFile("http://www.foo.com/bar/pf1.xml")
  _movie.loadPolicyFile("http://www.foo.com/bar/pf2.xml")
  end
  ```

  Im folgenden Code-Abschnitt werden beide Richtliniendateien geladen, weil diese von unterschiedlichen Serverstandorten stammen.

  ```
  on prepareMovie
  _movie.loadPolicyFile("http://www.foo.com/pf1.xml")
  _movie.loadPolicyFile("http://www.foo.com/bar/pf2.xml")
  end
  ```

- Eine Richtliniendatei vom Serverstandort `http://www.xyz.com/bar/pf.xml` ermöglicht allen Unterdomains von `http://www.xyz.com/bar/*` den Zugriff. Falls eine weitere Richtliniendatei mit unterschiedlichen Berechtigungen aus einemUnterverzeichnis von `http://www.xyz.com/bar/` geladen wird, werden auf dieses Unterverzeichnis nur diese Berechtigungen angewendet.

  Betrachten wir beispielsweise folgende Richtliniendatei mit dem Namen *policyfile.xml file* auf `http://www.director.com`.

  ```
  <cross-domain-policy>
  <allow-access-from domain="www.bar.com" to-ports="1222,8000-9000,9192" />
    </cross-domain-policy>
  ```

  Ein Shockwave-Film auf der URL „http://www.swgame.com“ mit folgendem Code-Abschnitt wird nicht funktionieren:

  ```
  on prepareMovie
  _movie.loadPolicyFile(http://www.director.com/policyfile.xml)
  getNetText("http://www.Director.com/data.txt")
  getNetText("http://www.Director.com/foo/bar/data.txt")
  end
  ```

  Nehmen wir an, wir laden eine andere Richtliniendatei `policyfile2.xml` mit folgendem Inhalt:

  ```
  <cross-domain-policy>
  <allow-access-from domain="http://www.swgames.com" to-ports="*" />
    </cross-domain-policy>
  ```

  Wenn ein Shockwave-Film unter der URL „http://www.swgame.com“ folgenden Code-Abschnitt verwendet, ist der Download in folgender Situation erfolgreich.

```
on prepareMovie
_movie.loadPolicyFile(http://www.director.com/policyfile.xml)
_movie.loadPolicyFile(http://www.director.com/foo/bar/policyfile2.xml)
getNetText("http://www.Director.com/data.txt") -- download fails
getNetText("http://www.Director.com/foo/bar/data.txt") --download succeeds
end
```

- Shockwave zeigt in folgenden Fällen ein Sicherheitsdialogfeld an:
  - Die Syntax der Cross-Domain-Datei ist nicht korrekt.
  - `loadPolicyFile()` legt eine falsche URL fest.
  - `loadPolicyfile()` wird im Film nicht festgelegt und eine Standarddatei `crossdomain.xml` ist im Stammverzeichnis des Servers nicht vorhanden.

Parameter

Parameter	Beschreibung	Erforderlich/Optional
URL	Legt den Speicherort fest, von dem die Richtliniendatei geladen wird.	Erforderlich

Beispiel

```
--Lingo
_movie.loadPolicyFile("http://www.testServer.com/test/pf.xml")
//JavaScript
_movie.loadPolicyFile("http://www.example.com/dir/pf.xml");
```

locToCharPos()

Syntax

```
-- Lingo syntax
memberObjRef.locToCharPos(location)

// JavaScript syntax
memberObjRef.locToCharPos(location);
```

Beschreibung

Diese Funktion gibt eine Nummer zurück, die angibt, welches Zeichen in einem angegebenen Felddarsteller einem Punkt im bezeichneten Feld am nächsten liegt.

Der Wert 1 entspricht dem ersten Zeichen im String, Wert 2 dem zweiten usw.

Parameter

location Erforderlich. Ein Punkt innerhalb des Felddarstellers. Der Wert für *location* ist ein Punkt relativ zur linken oberen Ecke des Felddarstellers.

Beispiel

Die folgende Anweisung ermittelt, welches Zeichen dem Punkt, der sich 100 Pixel rechts neben und 100 Pixel unter der oberen linken Ecke des Felddarstellers "Nachrichten" befindet, am nächsten liegt. Anschließend ordnet die Anweisung das Ergebnis der Variablen `PageDesign` zu.

```
--Lingo syntax
pageDesign = member("Today's News").locToCharPos(point(100, 100))

// JavaScript syntax
var pageDesign = member("Today's News").locToCharPos(point(100, 100));
```

locVToLinePos()

Syntax

```
-- Lingo syntax
memberObjRef.locVToLinePos(locV)

// JavaScript syntax
memberObjRef.locVToLinePos(locV);
```

Beschreibung

Diese Funktion gibt die Nummer der Textzeile zurück, die an einer angegebenen vertikalen Position erscheint.

Parameter

locV Erforderlich. Gibt die vertikale Position der Textzeile an. Dieser Wert ist die Anzahl der Pixel vom oberen Rand des Felddarstellers gemessen, nicht vom Teil des Felddarstellers, der gegenwärtig auf der Bühne erscheint.

Beispiel

Die folgende Anweisung ermittelt, welche Textzeile 150 Pixel vom oberen Rand des Felddarstellers „Nachrichten“ entfernt erscheint, und ordnet das Ergebnis der Variablen `pageBreak` zu:

```
--Lingo syntax
pageBreak = member("Today's News").locVToLinePos(150)

// JavaScript syntax
var pageBreak = member("Today's News").locVToLinePos(150);
```

log()

Syntax

```
log(number)
```

Beschreibung

Diese mathematische Funktion (nur Lingo) berechnet den natürlichen Logarithmus einer angegebenen Zahl.

In JavaScript-Syntax wird die `log()`-Funktion des Math-Objekts verwendet.

Parameter

number Erforderlich. Eine Zahl, von der der natürliche Logarithmus berechnet wird. Diese Zahl muss eine Dezimalzahl größer Null sein.

Beispiel

Die folgende Anweisung ordnet der Variablen Antwort den natürlichen Logarithmus von 10.5 zu.

```
-- Lingo
Answer = log(10.5)

// Javascript
Answer = Math.log(10.5);
```

Beispiel
Die folgende Anweisung berechnet den natürlichen Logarithmus der Quadratwurzel des Wertes Zahl und ordnet das Ergebnis der Variablen Antwort zu:

```
-- Lingo
Answer = log(Number.sqrt)

// Javascript
Answer = Math.log(Math.sqrt(Number));
```

makeList()

Syntax

```
--Lingo syntax
parserObject.makeList()

// JavaScript syntax
parserObject.makeList();
```

Beschreibung
Diese Funktion gibt eine Eigenschaftsliste zurück, die auf dem mit `parseString()` oder `parseURL()` geparsten XML-Dokument basiert.

Parameter
Keiner

Beispiel
Die folgende Prozedur parst ein XML-Dokument und gibt die dabei entstehende Liste zurück:

```
-- Lingo syntax
on ConvertToList xmlString
    parserObject = new(xtra "xmlparser")
    errorCode = parserObject.parseString(xmlString)
    errorString = parserObject.getError()
    if voidP(errorString) then
        parsedList = parserObject.makeList()
    else
        alert "Sorry, there was an error" && errorString
        exit
    end if
    return parsedList
end

// JavaScript syntax
function ConvertToList(xmlString) {
    parserObject = new Xtra("xmlparser"); // check syntax
    errorCode = parserObject.parseString(xmlString);
    errorString = parserObject.getError();
    if (voidP(errorString)) {
        parsedList = parserObject.makeList();
    } else {
        alert("Sorry, there was an error" + errorString);
        return false;
    }
    return parsedList;
}
```

Siehe auch

`makeSubList()`

makeScriptedSprite()

Syntax

```
-- Lingo syntax
spriteChannelObjRef.makeScriptedSprite({memberObjRef, loc})

// JavaScript syntax
spriteChannelObjRef.makeScriptedSprite({memberObjRef, loc});
```

Beschreibung

Diese Sprite-Kanalmethode wechselt die Steuerung eines Sprite-Kanals von der Bühne an ein Skript und platziert optional ein Sprite aus einem angegebenen Darsteller an einer vorgegebenen Position auf der Bühne.

Um die Steuerung des Sprite-Kanals wieder zurück auf die Bühne zu wechseln, rufen Sie `removeScriptedSprite()` auf.

Parameter

memberObjRef Optional. Ein Verweis auf den Darsteller, aus dem ein mit Skript erzeugtes Sprite erstellt wird. Durch alleinige Angabe dieses Parameters wird das Sprite in der Mitte der Bühne platziert.

loc Optional. Ein Punkt, der die Position auf der Bühne angibt, an der das mit Skript erzeugte Sprite platziert wird.

Beispiel

Die folgende Anweisung erstellt ein mit Skript erzeugtes Sprite in Sprite-Kanal 5 aus dem Darsteller „kite“ und platziert es an einem bestimmten Punkt auf der Bühne:

```
--- Lingo syntax
channel(5).makeScriptedSprite(member("kite"), point(35, 70))

// JavaScript syntax
channel(5).makeScriptedSprite(member("kite"), point(35, 70));
```

Siehe auch

removeScriptedSprite(), Sprite Channel

makeSubList()

Syntax

```
XMLnode.makeSubList()
```

Beschreibung

Analog zur Funktion `makeList()`, die den Stamm eines XML-Dokuments im Listenformat zurückgibt, gibt diese Funktion eine Eigenschaftsliste aus einem Child-Node zurück.

Parameter

Keiner

Beispiel

Als Ausgangspunkt wird das folgende XML-Dokument verwendet:

```
<e1>
        <tagName attr1="val1" attr2="val2"/>
        <e2> element 2</e2>
        <e3>element 3</e3>
</e1>
```

Die folgende Anweisung gibt eine Eigenschaftsliste zurück, die aus dem Inhalt des ersten Child-Nodes des Tags „<e1>“ besteht:

```
put gparser.child[ 1 ].child[ 1 ].makeSubList()
-- ["tagName": ["!ATTRIBUTES": ["attr1": "val1", "attr2": "val2"]]]
```

Siehe auch

makeList()

map()

Syntax

```
map(targetRect, sourceRect, destinationRect)
map(targetPoint, sourceRect, destinationRect)
```

Beschreibung
Diese Funktion legt die Position und Größe eines Rechtecks oder Punkts ausgehend vom Verhältnis zwischen einem Quellrechteck und einem Zielrechteck fest.

Das Verhältnis zwischen *targetRect* und *sourceRect* bestimmt das Verhältnis zwischen dem Ergebnis der Funktion und dem *destinationRect*.

Parameter
targetRect Erforderlich. Das Zielrechteck in der Beziehung.

targetPoint Erforderlich. Der Zielpunkt in der Beziehung.

sourceRect Erforderlich. Das Quellrechteck in der Beziehung.

destinationRect Erforderlich. Das Zielrechteck.

Beispiel
In diesem Verhalten wurden alle Sprites bereits als ziehbare Sprites definiert. Sprite 2b enthält eine kleine Bitmap. Sprite 1s ist ein rechteckiges Form-Sprite, in das sich Sprite 2b problemlos einpassen lässt. Sprite 4b ist eine größere Version der Bitmap in Sprite 2b. Sprite 3s ist eine größere Version der Form in Sprite 1s. Beim Verschieben von Sprite 2b oder Sprite 1s wird Sprite 4b automatisch ebenfalls verschoben. Wenn Sie Sprite 2b ziehen, werden seine Bewegungen von Sprite 4b nachvollzogen. Wenn Sie Sprite 1s ziehen, bewegt sich Sprite 4b in die entgegengesetzte Richtung. Eine Veränderung der Größe von Sprite 2b oder Sprite 1s bewirkt ebenfalls ein interessantes Ergebnis.

```
on exitFrame
    sprite(4b).rect = map(sprite(2b).rect, sprite(1s).rect, sprite(3s).rect)
    go the frame
end
```

map (3D)

Syntax
```
member(whichCastmember).motion(whichMotion).map(whichOtherMotion {, boneName})
```

Beschreibung
Dieser 3D-Bewegungsbefehl ordnet eine angegebene Bewegung der aktuellen Bewegung zu und wendet sie auf einen Knochen und alle seine Child-Objekte an. Dieser Befehl ersetzt alle Bewegungen, die zuvor dem angegebenen Knochen und seinen Child-Objekten zugeordnet waren. Die Abspielliste eines Modells bleibt unverändert.

Parameter
whichOtherMotion Erforderlich. Eine Zeichenfolge, die den Namen der zuzuordnenden Bewegung angibt.

boneName Optional. Eine Zeichenfolge, die den Namen des Knochens angibt, auf den die zugeordnete Bewegung angewendet wird. Bei Auslassung dieses Parameters wird der Stammknochen verwendet.

Beispiel
Die folgende Anweisung ordnet die Bewegung `LookUp` der Bewegung `SitDown` zu, beginnend mit dem Knochen `Neck`. Das Modell setzt sich hin und blickt gleichzeitig hoch.

```
member("Restaurant").motion("SitDown").map("LookUp", "Neck")
```

Siehe auch

motion(), duration (3D), cloneMotionFromCastmember

mapMemberToStage()

Syntax

```
sprite(whichSpriteNumber). mapMemberToStage(whichPointInMember)
mapMemberToStage(sprite whichSpriteNumber, whichPointInMember)
```

Beschreibung

Diese Funktion verwendet das angegebene Sprite und den angegebenen Punkt, um einen gleichwertigen Punkt innerhalb der Dimensionen der Bühnezurückzugeben. Damit wird den aktuellen Transformationen des Sprites unter Verwendung von `quad`, oder des Rechtecks, wenn es nicht transformiert wird, Rechnung getragen.

Diese Eigenschaft ist nützlich, wenn festgestellt werden soll, ob auf einen bestimmten Bereich eines Darstellers geklickt wurde, selbst wenn größere Veränderungen am Sprite auf der Bühne vorgenommen wurden.

Falls sich der angegebene Punkt auf der Bühne nicht innerhalb des Sprites befindet, wird `VOID` zurückgegeben.

Parameter

whichPointInMember Erforderlich. Ein Punkt, von dem ein gleichwertiger Punkt zurückgegeben wird.

Beispiel

Diese Anweisung verwendet das angegebene Sprite (sprite(1)) und den angegebenen Punkt (point(10,10)), um einen gleichwertigen Punkt innerhalb der Abmessungen der Bühne zurückzugeben.

```
-- Lingo
sprite(1). mapMemberToStage(point(10,10))

// Javascript
sprite(1). mapMemberToStage(point(10,10)) ;
```

Siehe auch

map(), mapStageToMember()

mapStageToMember()

Syntax

```
sprite(whichSpriteNumber). mapStageToMember(whichPointOnStage)
mapStageToMember(sprite whichSpriteNumber, whichPointOnStage)
```

Beschreibung

Diese Funktion verwendet das angegebene Sprite und den angegebenen Punkt, um einen gleichwertigen Punkt innerhalb der Dimensionen des Darstellers zurückzugeben. Damit wird den aktuellen Transformationen des Sprites unter Verwendung von `quad`, oder des Rechtecks, wenn es nicht transformiert wird, Rechnung getragen.

Diese Eigenschaft ist nützlich, wenn festgestellt werden soll, ob auf einen bestimmten Bereich eines Darstellers geklickt wurde, selbst wenn größere Transformationen am Sprite auf der Bühne vorgenommen wurden.

Falls sich der angegebene Punkt auf der Bühne nicht innerhalb des Sprites befindet, wird `VOID` zurückgegeben.

Parameter
whichPointOnStage Erforderlich. Ein Punkt, von dem ein gleichwertiger Punkt zurückgegeben wird.

Siehe auch
`map()`, `mapMemberToStage()`

marker()

Syntax
```
-- Lingo syntax
_movie.marker(markerNameOrNum)

// JavaScript syntax
_movie.marker(markerNameOrNum);
```

Beschreibung
Diese Filmmethode gibt die Bildnummer von Markierungen vor oder nach dem aktuellen Bild zurück.

Sie dient zur Implementierung einer Weiter- oder Zurück-Schaltfläche oder zur Einrichtung einer Animationsschleife.

Wenn der Parameter *markerNameOrNum* eine Ganzzahl ist, kann er zu einer beliebigen positiven oder negativen Ganzzahl oder 0 ausgewertet werden. Beispiel:

- `marker(2)` – Gibt die Bildnummer der zweiten Markierung nach dem aktuellen Bild zurück.
- `marker(1)` – Gibt die Bildnummer der ersten Markierung nach dem aktuellen Bild zurück.
- `marker(0)` – Gibt die Bildnummer des aktuellen Bildes zurück, wenn das Bild eine Markierung aufweist, oder die Bildnummer der vorhergehenden Markierung, wenn das aktuelle Bild nicht markiert ist.
- `marker(-1)` – Gibt die Bildnummer der ersten Markierung vor marker(0) zurück.
- `marker(-2)` – Gibt die Bildnummer der zweiten Markierung vor marker(0) zurück.

Falls der Parameter *markerNameOrNum* eine Zeichenfolge ist, gibt `marker()` die Bildnummer des ersten Bildes zurück, dessen Markierungsbeschriftung mit der Zeichenfolge übereinstimmt.

Parameter
markerNameOrNum Erforderlich. Eine Zeichenfolge, die ein Markierungsbeschriftung angibt, oder eine Ganzzahl, die eine Markierungsnummer angibt.

Beispiel
Die folgende Anweisung schickt den Abspielkopf zum Beginn des aktuellen Bildes, wenn dieses eine Markierung aufweist. Andernfalls sendet sie den Abspielkopf zur vorhergehenden Markierung.

```
-- Lingo syntax
_movie.go(_movie.marker(0))

// JavaScript syntax
_movie.go(_movie.marker(0));
```

Die folgende Anweisung stellt die Variable `nextMarker` so ein, dass sie mit der nächsten Markierung im Drehbuch übereinstimmt:

```
-- Lingo syntax
nextMarker = _movie.marker(1)

// JavaScript syntax
nextMarker = _movie.marker(1);
```

Siehe auch

`frame`, `frameLabel`, `go()`, `label()`, `markerList`, `Movie`

matrixAddition()

Syntax

```
<Matrix> matrixAddition(matrix1, matrix2)
```

Beschreibung

Globale Funktion, die eine Matrixaddition zwischen den beiden Matrizen ausführt und das Ergebnis als Matrix zurückgibt. „matrix1“ und „matrix2“ sollten dieselbe Dimension haben.

Parameter

matrix1 Erforderlich. Erste Matrix.

matrix2 Erforderlich. Zweite Matrix.

Beispiel

Der folgende Codeauszug erstellt zwei Matrizen, die mithilfe von `matrixAddition()` addiert werden.

```
--Lingo
mat1 = newMatrix(2, 2, [1,2,3,4])
mat2 = newMatrix(2, 2, [5,6,7,8])
mat3 = matrixAddition(mat1, mat2)
put(mat3.getVal(2,2))
-- 12.0000
//Java Script
mat1 = newMatrix(2, 2, list(1,2,3,4));
mat2 = newMatrix(2, 2, list(5,6,7,8));
mat3 = matrixAddition(mat1, mat2);
put(mat3.getVal(2,2));
// 12.0000
```

Siehe auch

`getVal()`, `setVal()`, `numRows()`, `numColumns()`, `matrixMultiply()`, `matrixMultiplyScalar()`, `matrixTranspose()`, `newMatrix()`

matrixMultiply()

Syntax

```
<Matrix> matrixMultiply(matrix1, matrix2)
```

Beschreibung

Globale Funktion, die eine Multiplikation zwischen den beiden Matrizen ausführen und das Ergebnis als Matrix zurückgibt. Die Anzahl der Spalten von „matrix1“ muss der Anzahl der Zeilen von „matrix2“ entsprechen.

Parameter

matrix1 Erforderlich. Erste Matrix.

matrix2 Erforderlich. Zweite Matrix.

Beispiel

Der folgende Codeauszug erstellt zwei Matrizen, die mithilfe von `matrixMultiply()` multipliziert werden.

```
--Lingo
mat1 = newMatrix(2, 2, [1,2,3,4])
mat2 = newMatrix(2, 2, [5,6,7,8])
mat3 = matrixMultiply(mat1, mat2)
put(mat3.getVal(2,2))
-- 50.0000

//Java Script
mat1 = newMatrix(2, 2, list(1,2,3,4));
mat2 = newMatrix(2, 2, list(5,6,7,8));
mat3 = matrixMultiply(mat1, mat2);
put(mat3.getVal(2,2));
// 50.0000
```

Siehe auch

`getVal()`, `setVal()`, `numRows()`, `numColumns()`, `matrixAddition()`, `matrixMultiplyScalar()`, `matrixTranspose()`, `newMatrix()`

matrixMultiplyScalar()

Syntax

```
<Matrix> matrixMultiplyScalar(matrix, scalarMultiplier)
```

Beschreibung

Globale Funktion, die jedes Element der Matrix mit einen angegeben Skalar multipliziert und das Ergebnis als Matrix zurückgibt.

Parameter

matrix Erforderlich. An der Skalarmultiplikation beteiligte Matrix.

scalarMultiplier Erforderlich. Skalarwert, mit dem die Matrix multipliziert werden soll.

Beispiel

Der folgende Codeauszug erstellt zwei Matrizen, die mithilfe von`matrixMultiply ()` multipliziert werden.

```
--Lingo
mat1 = newMatrix(2, 2, [1,2,3,4])
mat2 = matrixMultiplyScalar(mat1, 10)
put(mat2.getVal(2,2))
-- 40.0000
//Java Script
mat1 = newMatrix(2, 2, list(1,2,3,4));
mat2 = matrixMultiplyScalar(mat1, 10);
put(mat2.getVal(2,2));
// 40.0000
```

Siehe auch

`getVal()`, `setVal()`, `numRows()`, `numColumns()`, `matrixAddition()`, `matrixMultiply()`, `matrixTranspose()`, `newMatrix()`

matrixTranspose()

Syntax

```
<Matrix> matrixTranspose(matrix)
```

Beschreibung

Globale Funktion, die eine Transponieroperation auf die angegebene Matrix anwendet und das Ergebnis in Form einer anderen Matrix zurückgibt.

Parameter

matrix Erforderlich. Zu transponierende Matrix.

Beispiel

Der folgende Codeauszug erstellt eine Matrix, die mithilfe von `matrixTranspose()` transponiert wird.

```
--Lingo
mat1 = newMatrix(2, 2, [1,2,3,4])
mat2 = matrixTranspose(mat1)
put(mat1.getVal(1,2))
put(mat2.getVal(1,2))
-- 2.0000
-- 3.0000
//Java Script
mat1 = newMatrix(2, 2, list(1,2,3,4));
mat2 = matrixTranspose(mat1);
put(mat1.getVal(1,2));
put(mat2.getVal(1,2));
// 2.0000
// 3.0000
```

Siehe auch

`getVal()`, `setVal()`, `numRows()`, `numColumns()`, `matrixAddition()`, `matrixMultiply()`, `matrixMultiplyScalar()`, `newMatrix()`

max()

Syntax

```
list.max()
max(list)
max(value1, value2, value3, ...)
```

Beschreibung

Diese Funktion (nur Lingo) gibt den höchsten Wert in der angegebenen Liste oder den höchsten Wert in einer bestimmten Reihe von Wertenzurück.

Die Funktion `max` kann auch mit ASCII-Zeichen verwendet werden, so wie `<-` und `>`-Operatoren mit Strings verwendet werden können.

Parameter

Wert1, Wert2, Wert3, ... Optional. Eine Liste von Werten, aus der der niedrigste Wert ausgewählt wird.

Beispiel

Die folgende Prozedur ordnet der Variablen Gewinner den höchsten Wert in der Liste Gebote zu, die aus [#Castle:600, #Schmitz:750, #Wang:230] besteht. Das Ergebnis wird dann in den Inhalt des Felddarstellers "Gratulation" eingefügt.

```
-- Lingo syntax
on findWinner Bids
    Winner = Bids.max()
    member("Congratulations").text =
    "You have won, with a bid of $" & Winner &"!"
end

// JavaScript syntax
function findWinner(Bids) {
    Winner = Bids.max();
    member("Congratulations").text = "You have won, with a bid of $" +
    Winner + "!");
}
```

maximize()

Syntax

```
-- Lingo syntax
windowObjRef.maximize()

// JavaScript syntax
windowObjRef.maximize();
```

Beschreibung

Diese Fenstermethode maximiert ein Fenster.

Verwenden Sie diese Methode beim Erstellen benutzerdefinierter Titelleisten.

Parameter
Keiner

Beispiel
Die folgenden Anweisungen maximieren das Fenster namens „Artists“, wenn es nicht bereits maximiert ist.

```
-- Lingo syntax
if (window("Artists").sizeState <> #maximized) then
    window("Artists").maximize()
end if

// JavaScript syntax
if (window("Artists").sizeState != symbol("maximized")) {
    window("Artists").maximize();
}
```

Siehe auch
`minimize()`, `Window`

mci

Syntax
```
mci "string"
```

Beschreibung
Dieser Befehl wird nur unter Windows verwendet und übergibt den in *string* angegebenen String an die Windows Media Control Interface (MCI) zur Steuerung von Multimedia-Erweiterungen.

Hinweis: *Microsoft bietet keine Unterstützung mehr für die Verwendung der 16-Bit-MCI-Schnittstelle. Sie sollten daher für diese Funktionalität stattdessen Xtras von Drittherstellern verwenden.*

Parameter
string Erforderlich. Eine Zeichenfolge, die an die MCI übergeben wird.

Beispiel
Die folgende Anweisung bewirkt, dass der Befehl play cdaudio from 200 to 600 track 7 nur ausgeführt wird, wenn der Film unter Windows abgespielt wird:

```
mci "play cdaudio from 200 to 600 track 7"
```

member()

Syntax
```
-- Lingo syntax
member(memberNameOrNum {, castNameOrNum})

// JavaScript syntax
member(memberNameOrNum {, castNameOrNum});
```

Beschreibung

Diese Top-Level-Funktion erstellt einen Verweis auf einen Darsteller und gibt optional die Besetzungsbibliothek an, in der der Darsteller enthalten ist.

Die Methode `member()` ist ein spezifischer Verweis auf eine Besetzungsbibliothek und einen darin enthaltenen Darsteller, wenn sie mit den beiden Parametern *memberNameOrNum* und *castNameOrNum* verwendet wird:

```
trace(sprite(1).member);
// (member 1 of castLib 1)
```

Diese Methode unterscheidet sich von der Eigenschaft `spriteNum` eines Sprites, die stets eine Ganzzahl ist und eine Position in einer Besetzungsbibliothek bezeichnet, die Besetzungsbibliothek selbst aber nicht angibt:

```
trace(sprite(2).spriteNum);
// 2
```

Die Nummer eines Darstellers ist ein absoluter Bezug auf einen bestimmten Darsteller in einer bestimmten Besetzungsbibliothek:

```
trace(sprite(3).member.number)
// 3
```

Parameter

memberNameOrNum Erforderlich. Eine Zeichenfolge, die den Namen des Darstellers angibt, auf den verwiesen werden soll, oder eine Ganzzahl, die die Indexposition des Darstellers angibt, auf den verwiesen werden soll.

castNameOrNum Optional. Eine Zeichenfolge, die den Namen der Besetzungsbibliothek angibt, zu der der Darsteller gehört, oder eine Ganzzahl, die die Indexposition der Besetzungsbibliothek angibt, zu der der Darsteller gehört. Bei Auslassung des Parameters durchsucht `member()` alle Besetzungsbibliotheken, bis eine Übereinstimmung gefunden wurde.

Beispiel

Die folgende Anweisung legt die Variable `memWings` auf den Darsteller namens "Planes" fest, der sich in der Besetzungsbibliothek `Transportation` befindet.

```
-- Lingo syntax
memWings = member("Planes", "Transportation")

// JavaScript syntax
var memWings = member("Planes", "Transportation");
```

Siehe auch

Member, Sprite, spriteNum

mergeDisplayTemplate()

Syntax

```
-- Lingo syntax
_movie.mergeDisplayTemplate(propList)

// JavaScript syntax
_movie.mergeDisplayTemplate(propList);
```

Beschreibung
Diese Filmmethode führt in einem Schritt eine beliebige Anzahl von Anzeigevorlageneigenschaften mit einem vorhandenen Satz von Anzeigevorlageneigenschaften zusammen.

Parameter
propList Erforderlich. Eine Eigenschaftsliste, die die mit dem vorhandenen Satz von Anzeigevorlageneigenschaften zusammenzuführenden Anzeigevorlageneigenschaften enthält. In Lingo kann *propList* entweder eine durch Kommas getrennte Liste aus Name/Wert-Paaren oder eine durch Kommas getrennte Liste von Symbol/Wert-Paaren sein. In JavaScript-Syntax kann *propList* nur eine durch Kommas getrennte Liste von Name/Wert-Paaren sein.

Beispiel
Die folgende Anweisung führt einen Wert für die Titeleigenschaft mit `displayTemplate` zusammen:

```
-- Lingo syntax
_movie.mergeDisplayTemplate(propList(#title, "Welcome!"))

// JavaScript syntax
_movie.mergeDisplayTemplate(propList("title", "Welcome!"))
```

Siehe auch
`appearanceOptions`, `displayTemplate`, `Movie`, `propList()`, `titlebarOptions`

mergeProps()

Syntax

```
-- Lingo syntax
windowObjRef.mergeProps(propList)

// JavaScript syntax
windowObjRef.mergeProps(propList);
```

Beschreibung
Diese Fenstermethode führt in einem Schritt eine beliebige Anzahl von Fenstereigenschaften mit dem vorhandenen Satz von Fenstereigenschaften zusammen.

Parameter
propList Erforderlich. Ein Satz von Fenstereigenschaften, der mit dem vorhandenen Satz von Fenstereigenschaften zusammengeführt werden soll. Die Eigenschaften werden von den Eigenschaften `appearanceOptions` und `titlebarOptions` angegeben.

- In Lingo kann *propList* entweder eine durch Kommas getrennte Liste aus Name/Wert-Paaren oder eine durch Kommas getrennte Liste von Symbol/Wert-Paaren sein.
- In JavaScript-Syntax kann *propList* nur eine durch Kommas getrennte Liste von Name/Wert-Paaren sein.

Beispiel
Die folgende Anweisung legt verschiedene Eigenschaften für das Fenster namens „Cars“ fest.

```
-- Lingo syntax
window("Cars").mergeProps([#title:"Car pictures", #resizable:FALSE,
#titlebarOptions:[#closebox:TRUE, #icon:member(2)], #appearanceOptions:[#border:#line,
#shadow:TRUE]])

// JavaScript syntax
window("Cars").mergeProps(propList("title","Car pictures",
"resizable",false,"titlebarOptions",propList("closebox",true, "icon",member(2)),
"appearanceOptions",propList("border","line", "shadow",true)));
```

Siehe auch

appearanceOptions, titlebarOptions, Window

mesh (Eigenschaft)

Syntax

```
member(whichCastmember).model(whichModel).meshdeform.mesh[index].meshProperty
```

Beschreibung

Dieser 3D-Befehl ermöglicht den Zugriff auf die Gitternetzeigenschaften von Modellen, an denen der Modifizierer `meshDeform` angebracht ist. Bei Verwendung als `mesh.count` gibt dieser Befehl die Gesamtzahl von Gitternetzen im referenzierten Modell zurück.

Auf die folgenden Gitternetzeigenschaften kann zugegriffen werden:

- Mit `colorList` können Sie die Liste der im angegebenen Gitternetz verwendeten Farben ermitteln und festlegen.
- Mit `vertexList` können Sie die Liste der im angegebenen Gitternetz verwendeten Scheitelpunkte ermitteln und festlegen.
- Mit `normalList` können Sie die Liste der im angegebenen Gitternetz verwendeten normierten Vektoren (Normalen) abrufen und einstellen.
- Mit `textureCoordinateList` können Sie die von der ersten Texturebene im angegebenen Gitternetz verwendeten Texturkoordinaten ermitteln und festlegen. Um die Texturkoordinaten einer anderen Texturebene im angegebenen Gitternetz abzurufen oder einzustellen, verwenden Sie `meshdeform.mesh[`*index*`].texturelayer[`*index*`].textureCoordinateList`.
- Mit `textureLayer[index]` können Sie den Zugriff auf die Eigenschaften der angegebenen Texturebene bestimmen und festlegen.
- Mit `face[index]` können Sie die von den Flächen des angegebenen Gitternetzes verwendeten Scheitelpunkte, Normalen, Texturkoordinaten, Farben und Shader abrufen und einstellen.
- Mit `face.count` können Sie die Gesamtzahl von Flächen im angegeben Gitternetz ermitteln.

Hinweis: *Weitere Informationen über diese Eigenschaften enthalten die einzelnen Einträge (siehe Abschnitt "Siehe auch").*

Parameter

Keiner

Beispiel

Der folgende Lingo-Code fügt den Modifizierer `#meshDeform` zum Modell „thing1“ hinzu und zeigt dann die Scheitelpunktliste `vertexList` für das erste Gitternetz im Modell `thing1` an:

```
-- Lingo
member("newAlien").model("thing1").addModifier(#meshDeform)
put member("newalien").model("thing1").meshDeform.mesh[1].vertexList

// javascript
member("newAlien").getProp("model",1).addModifier(symbol("meshDeform"));
put
(member("newalien").getProp("model",1).getPropRef("meshDeform").getPropRef("mesh",1).vertexL
ist ;
-- [vector(239.0, -1000.5, 27.4), vector(162.5, -1064.7, 29.3), vector(115.3, -1010.8, -40.6),

vector(239.0, -1000.5, 27.4), vector(115.3, -1010.8, -40.6),
vector(162.5, -1064.7, 29.3), vector(359.0, -828.5, -46.3),
vector(309.9, -914.5, -45.3)]
```

Die folgende Anweisung zeigt die Anzahl von Gitternetzen im Modell "Flugzeug" an:

```
-- Lingo
put member("world").model("Aircraft").meshDeform.mesh.count
-- 4
```

Siehe auch

`meshDeform (Modifizierer)`, `colorList`, `textureCoordinateList`, `textureLayer`, `normalList`, `vertexList (mesh deform)`, `face[ ]`

meshDeform (Modifizierer)

Syntax

```
member(whichCastmember).model(whichModel).meshDeform.propertyName
```

Beschreibung

Mit diesem 3D-Modifizierer können Sie die verschiedenen Aspekte der Gitternetzstruktur des referenzierten Modells steuern. Nachdem Sie den Modifizierer `#meshDeform` mit dem Befehl `addModifier` zu einem Modell hinzugefügt haben, können Sie auf die folgenden Eigenschaften des Modifizierers `#meshDeform` zugreifen:

Hinweis: *Weitere Informationen über die folgenden Eigenschaften enthalten die einzelnen Einträge (siehe Abschnitt "Siehe auch").*

- `face.count` gibt die Gesamtzahl von Seiten im referenzierten Modell zurück.
- `mesh.count` gibt die Anzahl von Gitternetzen im referenzierten Modell zurück.
- `mesh[index]` ermöglicht den Zugriff auf die Eigenschaften des angegebenen Gitternetzes.

Parameter

Keiner

Beispiel

Die folgende Anweisung zeigt die Anzahl von Flächen im Modell `gbFace` an:

```
-- Lingo
put member("3D World").model("gbFace").meshDeform.face.count
-- 432

// Javascript
member("3D World").getProp("model", a).getPropRef("meshDeform") ;
// where a refers to the number index of the model "gbFace"
```

Die folgende Anweisung zeigt die `vertexList` (Liste der Scheitelpunkte) von Gitternetzen im Modell `gbFace` an:

```
-- Lingo
put member("3D World").model("gbFace").meshDeform.mesh[1].vertexList
-- 2

// Javscript
member("3D World").getProp("model", a).getPropRef("meshDeform").getPropRef("mesh"
,1).vertexList ;
```

Die folgende Anweisung zeigt die Anzahl von Flächen im zweiten Gitternetz des Modells `gbFace` an:

```
put member("3D World").model("gbFace").meshDeform.mesh[2].face.count
-- 204
```

Siehe auch

mesh (Eigenschaft), addModifier

min

Syntax

```
list.min
min(list)
min (a1, a2, a3...)
```

Beschreibung

Diese Funktion (nur Lingo) gibt den Mindestwert in einer Liste an.

Parameter

a1, a2, a3, ... Optional. Eine Liste von Werten, aus der der niedrigste Wert ausgewählt wird.

Beispiel

Die folgende Prozedur weist der Variablen vLowest den Mindestwert in der Liste bids zu, die aus [#Castle:600, #Shields:750, #Wang:230] besteht. Das Ergebnis wird sodann in den Inhalt des Felddarstellers „Sorry“ eingefügt:

```
on findLowest bids
    vLowest = bids.min()
    member("Sorry").text = \
    "We're sorry, your bid of $" & vLowest && "is not a winner!"
end
```

Siehe auch

max()

minimize()

Syntax

```
-- Lingo syntax
windowObjRef.minimize()

// JavaScript syntax
windowObjRef.minimize();
```

Beschreibung

Diese Fenstermethode minimiert ein Fenster.

Verwenden Sie diese Methode beim Erstellen benutzerdefinierter Titelleisten.

Parameter

Keiner

Beispiel

Die folgenden Anweisungen minimieren das Fenster namens „Artists“, wenn es nicht bereits minimiert ist.

```
-- Lingo syntax
if (window("Artists").sizeState <> #minimized) then
    window("Artists").minimize()
end if

// JavaScript syntax
if (window("Artists").sizeState.toString() != symbol("minimized").toString())
{
    window("Artists").minimized();
}
```

Siehe auch

`maximize()`, `Window`

model (3D)

Syntax

```
member(whichCastmember).model(whichModel)
member(whichCastmember).model[index]
member(whichCastmember).model.count
member(whichCastmember).model(whichModel).propertyName
member(whichCastmember).model[index].propertyName
```

Beschreibung

Dieser 3D-Befehl gibt das im referenzierten Darsteller gefundene Modell zurück, das den im Parameter *whichModel* angegebenen Namen besitzt bzw. das sich an der durch *index* angegebenen Indexposition befindet. Wenn für den angegebenen Parameter kein Modell gefunden wird, gibt der Befehl `void` zurück. In der Form `model.count` gibt der Befehl die im referenzierten Darsteller gefundene Anzahl von Modellen zurück. Über diesen Befehl können Sie außerdem auf die Eigenschaften des jeweiligen Modells zugreifen.

Beim Vergleich von Modellnamen spielt die Groß-/Kleinschreibung keine Rolle. Die Indexposition eines bestimmten Modells kann sich ändern, wenn Objekte an niedrigeren Indexpositionen gelöscht werden.

Wenn kein Modell mit dem angegebenen Namen bzw. an der angegebenen Indexposition gefunden wird, gibt dieser Befehl „void“ zurück.

Parameter
whichModel Optional. Eine Zeichenfolge, die den Namen des zurückzugebenden Modells angibt.

Beispiel
Die folgende Anweisung speichert einen Verweis auf das Modell "Spieleravatar" in der Variablen diesesModell:

```
thismodel = member("3DWorld").model("Player Avatar")
```

Die folgende Anweisung speichert einen Bezug auf das achte Modell des Darstellers „3DWelt“ in der Variablen diesesModell:

```
thismodel = member("3DWorld").model[8]
```

Die folgende Anweisung zeigt, dass es im Darsteller von Sprite 1 vier Modelle gibt:

```
put sprite(1).member.model.count
-- 4
```

modelResource

Syntax

```
member(whichCastmember).modelResource(whichModelResource)
member(whichCastmember).modelResource[index]
member(whichCastmember).modelResource.count
member(whichCastmember).modelResource(whichModelResource).propertyName
member(whichCastmember).modelResource[index].propertyName
```

Beschreibung
Dieser 3D-Befehl gibt die im referenzierten Darsteller gefundene Modellressource zurück, die den im Parameter *whichModelResource* angegebenen Namen besitzt bzw. die sich an der durch *index* angegebenen Indexposition befindet. Wenn für den angegebenen Parameter keine Modellressource gefunden wird, gibt der Befehl `void` zurück. In der Form `modelResource.count` gibt der Befehl die im referenzierten Darsteller gefundene Anzahl von Modellressourcen zurück. Über diesen Befehl können Sie außerdem auf die Eigenschaften des jeweiligen Modells zugreifen.

Beim Vergleich von Modellressourcennamenstrings spielt die Groß-/Kleinschreibung keine Rolle. Die Indexposition einer bestimmten Modellressource kann sich ändern, wenn Objekte an niedrigeren Indexpositionen gelöscht werden.

Parameter
whichModelResource Optional. Eine Zeichenfolge, die den Namen der zurückzugebenden Modellressource angibt.

Beispiel
Die folgende Anweisung speichert einen Bezug auf die Modellressource `HouseA` in der Variablen `thismodelResource`:

```
thismodelResource = member("3DWorld").modelResource("HouseA")
```

Die folgende Anweisung speichert einen Bezug auf das vierzehnte Modell des Darstellers `3DWorld` in der Variablen `thismodelResource`:

```
thismodelResource = member("3DWorld").modelResource[14]
```

Die folgende Anweisung zeigt, dass es im Darsteller von Sprite 1 zehn Modellressourcen gibt:

```
put sprite(1).member.modelResource.count
--10
```

modelsUnderLoc

Syntax

```
member(whichCastmember).camera(whichCamera).modelsUnderLoc(pointWithinSprite, optionsList)
```

Beschreibung

Dieser 3D-Befehl gibt eine Liste der Modelle zurück, die anhand der referenzierten Kamera unter einem angegebenen Punkt im Rechteck eines Sprites zu finden sind. Die Liste der Modelle kann auch mit einem Satz optionaler Parameter verglichen werden, bevor sie zurückgegeben wird.

In der zurückgegebenen Liste ist der erste Eintrag das dem Betrachter am nächsten gelegene und der letzte Eintrag das vom Betrachter am weitesten entfernte Modell.

Pro Modell wird nur ein Schnittpunkt (der am nächsten gelegene) zurückgegeben.

Wenn unter dem angegebenen Punkt keine Modelle zu finden sind, wird eine leere Liste zurückgegeben.

Parameter

pointWithinSprite Erforderlich. Ein Punkt, unter dem eine Liste von Modellen zu finden ist. Dieser Punkt ist relativ auf die linke obere Ecke des Sprites in Pixel bezogen.

optionsList Optional. Eine Liste, die die Höchstzahl von zurückzugebenden Modellen, den Detailliertheitsgrad der Informationen, eine Liste von Modellen für die Besetzung sowie den Höchstabstand für das Zeichnen des Strahls angibt. Alle diese Eigenschaften sind optional.

maxNumberOfModels Optional. Eine Ganzzahl, die die maximale Länge der zurückzugebenden Liste angibt. Wird dieser Parameter ausgelassen, gibt der Befehl eine Liste mit Verweisen für alle Modelle zurück, die unter dem angegebenen Punkt gefunden wurden.

levelOfDetail Optional. Ein Symbol, das den Detailliertheitsgrad der zurückgegebenen Informationen angibt. Folgende Werte sind gültig:

- `#simple` gibt eine Liste mit Bezügen auf die unter dem Strahl gefundenen Modelle zurück (Standardeinstellung).
- `#detailed` gibt eine Liste mit Eigenschaftslisten zurück (eine für jedes geschnittene Modell). Jede Eigenschaftsliste enthält folgende Eigenschaften:
  - `#model` ist ein Bezug auf das geschnittene Modellobjekt.
  - `#distance` ist die Entfernung zwischen Kamera und Schnittpunkt mit dem Modell.
  - `#isectPosition` ist ein Vektor, der die Weltposition des Schnittpunkts angibt.
  - `#isectNormal` ist der normierte Weltvektor zum Gitternetz am Schnittpunkt.
  - `#meshID` ist die`meshID` des geschnittenen Gitternetzes, die als Index für die Gitternetzliste des Modifizierers `meshDeform` verwendet werden kann.

- `#faceID` ist die ID der geschnittenen Fläche, die als Index für die Flächenliste des Modifizierers `meshDeform` verwendet werden kann.
- `#vertices` ist eine aus drei Elementen bestehende Liste mit Vektoren, die die Weltpositionen der Scheitelpunkte auf der geschnittenen Fläche angeben.
- `#uvCoord` ist eine Eigenschaftsliste mit den Eigenschaften `#u` und `#v`, die die baryzentrischen u- und v-Koordinaten der Fläche angeben.

modelList Optional. Eine Liste von Modellverweisen, die eingeschlossen werden, wenn sie unter dem angegebenen Strahl gefunden werden. In dieser Liste nicht enthaltene Modellverweise werden ignoriert, auch wenn sie sich unter dem angegebenen Strahl befinden. Verwenden Sie die Modellverweise, nicht die Zeichenfolgennamen der Modelle. Geben Sie jedes aufzunehmende Modell an. Durch das Hinzufügen eines Parent-Modellverweises werden nicht automatisch dessen Child-Modellverweise aufgenommen.

Beispiel

Diese Anweisung erstellt eines Liste mit zehn Modellen:

```
tModelList = [member("3D").model("foo"),member("3D").model[10]]
```

Die folgende Anweisung erzeugt eine Liste mit Optionen, die maximal zehn Modelle und einfache Detailinformationen zurückgibt sowie die Ergebnisse aus `tModelList` zeichnet:

```
tOptionsList = [#maxNumberOfModels: 10, #levelOfDetail: #simple, #modelList: tModelList]
```

Nachdem die Optionenliste erzeugt wurde, wird in der ersten Zeile der folgenden Prozedur die Position des Cursors von einem Punkt auf der Bühne an einen Punkt in Sprite 5 verschoben. In der zweiten Zeile werden mit dem Befehl `modelsUnderLoc` die ersten drei unter diesem Punkt gefundenen Modelle ermittelt. In der dritten Zeile werden die zurückgegebenen Informationen über die Modelle im Nachrichtenfenster angezeigt.

```
-- Lingo syntax
on mouseUp
    pt = the mouseLoc - point(sprite(5).left, sprite(5).top)
    m = sprite(5).camera.modelsUnderLoc(pt, tOptionsList)
    put m
    end

// JavaScript syntax
function mouseUp() {
    pt = _mouse.mouseLoc - point(sprite(5).left, sprite(5).top);
    m = sprite(5).camera.modelsUnderLoc(pt, tOptionsList);
    put(m);
}
```

Siehe auch

`modelsUnderRay`, `modelUnderLoc`

modelsUnderRay

Syntax

```
member(whichCastmember).modelsUnderRay(locationVector, directionVector, optionsList)
```

Beschreibung

Dieser 3D-Befehl gibt eine Liste der Modelle unter einem Strahl zurück, der ausgehend von einer angegebenen Position gezeichnet wird und in eine angegebene Richtung zeigt. Beide Vektoren werden in weltbezogenen Koordinaten angegeben. Die Liste der Modelle kann auch mit einem Satz optionaler Parameter verglichen werden, bevor sie zurückgegeben wird.

In der zurückgegebenen Liste ist der erste Eintrag das der durch *locationVector* angegebenen Position am nächsten gelegene Modell und der letzte Eintrag das von dieser Position am weitesten entfernte Modell.

Pro Modell wird nur ein Schnittpunkt (der am nächsten gelegene) zurückgegeben.

Wenn unter dem angegebenen Strahl keine Modelle zu finden sind, wird eine leere Liste zurückgegeben.

Parameter

locationVector Erforderlich. Ein Vektor, von dem aus ein Strahl gezeichnet wird und unter dem eine Liste von Modellen zu finden ist.

directionVector Erforderlich. Ein Vektor, der die Richtung angibt, in die der Strahl zeigt.

optionsList Optional. Eine Liste, die die Höchstzahl von zurückzugebenden Modellen, den Detailliertheitsgrad der Informationen, eine Liste von Modellen für die Besetzung sowie den Höchstabstand für das Zeichnen des Strahls angibt. Alle diese Eigenschaften sind optional.

maxNumberOfModels Optional. Eine Ganzzahl, die die maximale Länge der zurückzugebenden Liste angibt. Wird dieser Parameter ausgelassen, gibt der Befehl eine Liste mit Verweisen für alle Modelle zurück, die unter dem angegebenen Strahl gefunden wurden.

levelOfDetail Optional. Ein Symbol, das den Detailliertheitsgrad der zurückgegebenen Informationen angibt. Folgende Werte sind gültig:

- `#simple` gibt eine Liste mit Bezügen auf die unter dem Strahl gefundenen Modelle zurück (Standardeinstellung).
- `#detailed` gibt eine Liste mit Eigenschaftslisten zurück (eine für jedes geschnittene Modell). Jede Eigenschaftsliste enthält folgende Eigenschaften:
  - `#model` ist ein Bezug auf das geschnittene Modellobjekt.
  - `#distance` ist die Entfernung zwischen der durch *locationVector* angegebenen Weltposition und dem Schnittpunkt mit dem Modell.
  - `#isectPosition` ist ein Vektor, der die Weltposition des Schnittpunkts angibt.
  - `#isectNormal` ist der normierte Weltvektor zum Gitternetz am Schnittpunkt.
  - `#meshID` ist die `meshID` des geschnittenen Gitternetzes, die als Index für die Gitternetzliste des Modifizierers `meshDeform` verwendet werden kann.
  - `#faceID` ist die ID der geschnittenen Fläche, die als Index für die Flächenliste des Modifizierers `meshDeform` verwendet werden kann.
  - `#vertices` ist eine aus drei Elementen bestehende Liste mit Vektoren, die die Weltpositionen der Scheitelpunkte auf der geschnittenen Fläche angeben.
  - `#uvCoord` ist eine Eigenschaftsliste mit den Eigenschaften `#u` und `#v`, die die baryzentrischen u- und v-Koordinaten der Fläche angeben.

modelList Optional. Eine Liste von Modellverweisen, die eingeschlossen werden, wenn sie unter dem angegebenen Strahl gefunden werden. In dieser Liste nicht enthaltene Modellverweise werden ignoriert, auch wenn sie sich unter dem angegebenen Strahl befinden. Verwenden Sie die Modellverweise, nicht die Zeichenfolgennamen der Modelle. Geben Sie jedes aufzunehmende Modell an. Durch das Hinzufügen eines Parent-Modellverweises werden nicht automatisch dessen Child-Modellverweise aufgenommen.

maxDistance Optional. Der Höchstabstand von der durch *locationVector* angegebenen Weltposition. Wenn sich die Begrenzungskugel eines Modells innerhalb des angegebenen Höchstabstandes befindet, wird es eingeschlossen. Liegt die Begrenzungskugel im Bereich, enthält diese möglicherweise Polygone, die ebenfalls innerhalb des Bereichs liegen und so geschnitten werden können.

Beispiel

Diese Anweisung erstellt eines Liste mit zehn Modellen:

```
tModelList = [member("3D").model("foo"),member("3D").model[10]]
```

Die folgende Anweisung erzeugt eine Liste mit Optionen, die maximal zehn Modelle und einfache Detailinformationen zurückgibt sowie die Ergebnisse aus `tModelList` zeichnet und einen maximalen Strahlabstand von 50 hat:

```
tOptionsList = [#maxNumberOfModels: 10, #levelOfDetail: #simple, #modelList: tModelList,
#maxDistance: 50]
```

Nachdem die Optionenliste erzeugt wurde, schließt die folgende Anweisung die Liste unter einem Strahl ein, der von der Position vector(0, 0, 300) ausgehend und auf der -Z-Achse nach unten verläuft:

```
put member("3d").modelsUnderRay(vector(0, 0, 300), vector(0, 0, -1), tOptionsList)
```

Siehe auch

`modelsUnderLoc`, `modelUnderLoc`

modelUnderLoc

Syntax

```
member(whichCastmember).camera(whichCamera).modelUnderLoc(pointWithinSprite)
```

Beschreibung

Dieser 3D-Befehl gibt einen Verweis auf das erste Modell zurück, das anhand der referenzierten Kamera unter einem angegebenen Punkt im Rechteck eines Sprites zu finden ist.

Wenn unter dem angegebenen Punkt kein Modell zu finden ist, gibt dieser Befehl `void` zurück.

Eine Liste aller unter einem bestimmten Punkt gefundenen Modelle und detaillierte Informationen dazu finden Sie unter: `modelsUnderLoc`.

Parameter

pointWithinSprite Erforderlich. Ein Punkt, unter dem das erste Modell zu finden ist. Die Position von *pointWithinSprite* wird relativ zur oberen linken Ecke des Sprite in Pixel gemessen.

Beispiel

In der ersten Zeile in der folgenden Prozedur wird die Position des Cursors von einem Punkt auf der Bühne an einen Punkt in Sprite 5 verschoben. In der zweiten Zeile wird das erste Modell unter diesem Punkt ermittelt. In der dritten Zeile wird das Ergebnis im Nachrichtenfenster angezeigt.

```
-- Lingo syntax
on mouseUp
    pt = the mouseLoc - point(sprite(5).left, sprite(5).top)
    m = sprite(5).camera.modelUnderLoc(pt)
    put m
end

// JavaScript syntax
function mouseUp() {
    pt = _mouse.mouseLoc - point(sprite(5).left, sprite(5).top);
    m = sprite(5).camera.modelUnderLoc(pt);
    put(m);
}
```

Siehe auch

modelsUnderLoc, modelsUnderRay

motion()

Syntax

```
member(whichCastmember).motion(whichMotion)
member(whichCastmember).motion[index]
member(whichCastmember).motion.count
```

Beschreibung

Dieser 3D-Befehl gibt die im referenzierten Darsteller gefundene Bewegung zurück, die den im Parameter *whichMotion* angegebenen Namen besitzt bzw. die sich an der durch *index* angegebenen Indexposition befindet. In der Form `motion.count` gibt diese Eigenschaft die Gesamtzahl von Bewegungen im Darsteller zurück.

Beim Vergleich von Objektnamenstrings spielt die Groß-/Kleinschreibung keine Rolle. Die Indexposition einer bestimmten Bewegung kann sich ändern, wenn Objekte an niedrigeren Indexpositionen gelöscht werden.

Wenn keine Bewegung mit dem angegebenen Namen bzw. an der angegebenen Indexposition gefunden wird, gibt dieser Befehl „void“ zurück.

Beispiel

```
thisMotion = member("3D World").motion("Wing Flap")
thisMotion = member("3D World").motion[7]
put member("scene").motion.count
-- 2
```

Siehe auch

duration (3D), map (3D)

move()

Syntax

```
-- Lingo syntax
memberObjRef.move({intPosn, castLibName})

// JavaScript syntax
memberObjRef.move({intPosn, castLibName});
```

Beschreibung

Diese Darstellermethode bewegt einen angegebenen Darsteller entweder an die erste freie Position in der Besetzung, in der er enthalten ist, oder an eine vorgegebene Position in einer angegebenen Besetzung.

Verwenden Sie diese Methode bei der Filmerstellung und nicht während der Laufzeit, da die Bewegung normalerweise gemeinsam mit der Datei gespeichert wird. Die eigentliche Position eines Darstellers hat in der Regel bei der Wiedergabe keine Auswirkung auf die Präsentation. Um den Inhalt eines Sprites auszutauschen oder die Anzeige zur Laufzeit zu ändern, stellen Sie den Darsteller des Sprites ein.

Parameter

intPosn Optional. Eine Ganzzahl, die die Position in der Besetzungsbibliothek *castLibName* angibt, an die der Darsteller bewegt wird.

castLibName Optional. Eine Zeichenfolge, die den Namen der Besetzungsbibliothek angibt, in die der Darsteller bewegt wird.

Beispiel

Die folgende Anweisung verschiebt den Darsteller "Schrein" an die erste leere Position im Besetzungsfenster:

```
-- Lingo syntax
member("shrine").move()

// JavaScript syntax
member("shrine").move();
```

Die folgende Anweisung verschiebt den Darsteller „Schrein“ an Position 20 im Besetzungsfenster „Bitmaps“:

```
-- Lingo syntax
member("shrine").move(member(20, "Bitmaps"))

// JavaScript syntax
member("shrine").move(member(20, "Bitmaps"));
```

Siehe auch

Member

moveTo

Syntax

```
soundObject.moveTo(#Mixer)
```

Beschreibung

Diese Soundobjektmethode verschiebt das Soundobjekt von einem Mixer zu einem anderen.

Beispiele

Das folgende Beispiel verschiebt das angegebene Soundobjekt zu `mixer2`.

```
--Lingo syntax
on mouseUp me
     soundObjRef.moveTo(mixer2)
end

//JavaScript syntax
function mouseUp(){
     soundObjRef.moveTo(mixer2);
}
```

moveToBack()

Syntax

```
-- Lingo syntax
windowObjRef.moveToBack()

// JavaScript syntax
windowObjRef.moveToBack();
```

Beschreibung

Diese Fenstermethode schließt ein hinter allen anderen Fenstern liegendes Fenster.

Parameter

Keiner

Beispiel

Die folgenden Anweisungen verschieben das erste Fenster in `windowList` hinter alle anderen Fenster:

```
-- Lingo syntax
myWindow = _player.windowList[1]
myWindow.moveToBack()

// JavaScript syntax
var myWindow = _player.windowList[1];
myWindow.moveToBack();
```

Wenn Sie den Namen des Fensters kennen, verwenden Sie die folgende Syntax:

```
-- Lingo syntax
window("Demo Window").moveToBack()

// JavaScript syntax
window("Demo Window").moveToBack();
```

Siehe auch

moveToFront(), Window

moveToFront()

Syntax

```
-- Lingo syntax
windowObjRef.moveToFront()

// JavaScript syntax
windowObjRef.moveToFront();
```

Beschreibung

Diese Fenstermethode verschiebt ein Fenster vor alle anderen Fenster.

Parameter

Keiner

Beispiel

Die folgenden Anweisungen verschieben das erste Fenster in `windowList` vor alle anderen Fenster:

```
-- Lingo syntax
myWindow = _player.windowList[1]
myWindow.moveToFront()

// JavaScript syntax
var myWindow = _player.windowList[1];
myWindow.moveToFront();
```

Wenn Sie den Namen des Fensters kennen, verwenden Sie die folgende Syntax:

```
-- Lingo syntax
window("Demo Window").moveToFront()

// JavaScript syntax
window("Demo Window").moveToFront();
```

Siehe auch

moveToBack(), Window

moveVertex()

Syntax

```
-- Lingo syntax
memberObjRef.moveVertex(vertexIndex, xChange, yChange)

// JavaScript syntax
memberObjRef.moveVertex(vertexIndex, xChange, yChange);
```

Beschreibung

Diese Funktion verschiebt den Scheitelpunkt eines Vektorformdarstellers an eine andere Position.

Die horizontalen und vertikalen Koordinaten für die Bewegung sind relativ zur aktuellen Position des Scheitelpunkts. Die Position des Scheitelpunkts ist relativ zum Ausgangspunkt des Vektorformdarstellers.

Eine Änderung der Position eines Scheitelpunkts hat die gleichen Auswirkungen auf die Form wie das Ziehen des Scheitelpunkts in einem Editor.

Parameter

vertexIndex Erforderlich. Gibt die Indexposition des zu verschiebenden Scheitelpunktes an.

xChange Erforderlich. Gibt das Ausmaß der horizontalen Verschiebung des Scheitelpunktes an.

xChange Erforderlich. Gibt das Ausmaß der vertikalen Verschiebung des Scheitelpunktes an.

Beispiel

Die folgende Anweisung verschiebt den ersten Scheitelpunkt der Vektorform "Archie" von der aktuellen Position aus um 25 Pixel nach rechts und um 10 Pixel nach unten:

```
-- Lingo syntax
member("Archie").moveVertex(1, 25, 10)

// JavaScript syntax
member("Archie").moveVertex(1, 25, 10);
```

Siehe auch

`addVertex()`, `deleteVertex()`, `moveVertexHandle()`, `originMode`, `vertexList`

moveVertexHandle()

Syntax

```
-- Lingo syntax
memberObjRef.moveVertexHandle(vertexIndex, handleIndex, xChange, yChange)

// JavaScript syntax
memberObjRef.moveVertexHandle(vertexIndex, handleIndex, xChange, yChange);
```

Beschreibung

Diese Funktion verschiebt den Scheitelpunktanfasser eines Vektorformdarstellers an eine andere Position.

Die horizontalen und vertikalen Koordinaten für die Bewegung sind relativ zur aktuellen Position des Scheitelpunktanfassers. Die Position des Scheitelpunktanfassers ist relativ zum Scheitelpunkt, den er steuert.

Eine Änderung der Position eines Anfassers hat die gleichen Auswirkungen auf die Form wie das Ziehen des Scheitelpunkts in einem Editor.

Parameter

vertexIndex Erforderlich. Gibt die Indexposition des Scheitelpunktes an, der den zu verschiebenden Anfasser enthält.

handleIndex Erforderlich. Gibt die Indexposition des zu verschiebenden Anfassers an.

xChange Erforderlich. Gibt das Ausmaß der horizontalen Verschiebung des Scheitelpunktes

an.

xChange Erforderlich. Gibt das Ausmaß der vertikalen Verschiebung des Scheitelpunktanfassers an.

Beispiel

Die folgende Anweisung verschiebt den ersten Anfasser des zweiten Scheitelpunkts in der Vektorform "Archie" um 15 Pixel nach rechts und um 5 Pixel nach oben:

```
-- Lingo syntax
moveVertexHandle(member("Archie"), 2, 1, 15, -5)

// JavaScript syntax
member("Archie").moveVertexHandle(2, 1, 15, -5);
```

Siehe auch

addVertex(), deleteVertex(), originMode, vertexList

multiply()

Syntax

```
transform.multiply(transform2)
```

Beschreibung

Dieser 3D-Befehl wendet die Positions-, Drehungs- und Skalierungseffekte von *transform2* auf die ursprüngliche Transformation an.

Parameter

transform2 Erforderlich. Gibt die Transformation an, die die auf eine andere Transformation anzuwendenden Effekte enthält.

Beispiel

Die folgende Anweisung wendet die Positions-, Drehungs- und Skalierungseffekte der Transformation des Modells "Mars" auf die Transformation des Modells "Pluto" an. Dadurch ergibt sich eine ähnliche Wirkung, wie wenn "Mars" für die Dauer eines Bildes zum Parent-Objekt von "Pluto" gemacht wird.

```
-- Lingo
member("scene").model("Pluto").transform.multiply(member("scene").model("Mars").transform)

// Javascript
member("scene").getProp("model" ,i ).transform.multiply(member("scene").getProp("model" ,
j).transform) ;
```

mute (Mixer)

Syntax

```
mixer.Mute()
```

Beschreibung

Diese Mixermethode stellt die Eigenschaft `volume` der Mixerausgabe auf Null ein.

Beispiele

```
-- Lingo syntax
on mouseUp me
     mixerRef.mute() --Mutes the mixer.
end

//JavaScript syntax
function mouseUp(){
mixerRef.mute(); //Mutes the mixer.
}
```

Siehe auch

unmute (Mixer), Mixer

mute (Soundobjekt)

Syntax

```
soundObject.mute()
```

Beschreibung

Mit dieser Soundobjektmethode wird die Eigenschaft `volume` des Soundobjekts auf Null eingestellt.

Beispiele

```
-- Lingo syntax
on mouseUp me
     soundObjRef.mute() -- Mutes the sound object associated with soundObjRef.
end

//JavaScript syntax
function mouseUp(){
soundObjRef.mute(); // Mutes the sound object associated with soundObjRef.
}
```

Siehe auch

unmute (Soundobjekt)

neighbor

Syntax

```
member(whichCastmember).model(whichModel).meshdeform.mesh[index].face[index].neighbor[index]
```

Beschreibung

Dieser 3D-`meshDeform`-Befehl gibt eine Liste mit Listen zurück, in denen die Nachbarn einer bestimmten Fläche eines Gitternetzes gegenüber der durch den Nachbarindex `(1,2,3)` angegebenen Flächenecke aufgeführt sind. Wenn die Liste keinen Eintrag enthält, weist die Fläche in dieser Richtung keine Nachbarn auf. Enthält die Liste hingegen mehrere Listen, gilt das Gitternetz als "non-manifold". Normalerweise enthält die Liste eine einzige Liste aus vier ganzzahligen Werten: `[meshIndex, faceIndex, vertexIndex, flipped]`.

Der Wert `meshIndex` ist der Index des Gitternetzes mit der Nachbarfläche. Der Wert `faceIndex` ist der Index der Nachbarfläche in diesem Gitternetz. Der Wert `vertexIndex` ist der Index der nicht gemeinsam benutzten Scheitelpunkte der Nachbarfläche. Der Wert `flipped` gibt an, ob die Flächenausrichtung die gleiche ist wie bei der ursprünglichen Fläche `(1)` oder das Gegenteil `(2)`.

Parameter
Keiner

Siehe auch
`meshDeform (Modifizierer)`

netAbort

Syntax
```
netAbort(URL)
netAbort(netID)
```

Beschreibung
Dieser Befehl bricht einen Netzwerkvorgang ab, ohne auf ein Ergebnis zu warten.

Am besten beenden Sie einen Netzwerkvorgang mithilfe einer Netzwerk-ID. Die ID wird zurückgegeben, wenn Sie eine Netzwerkfunktion wie `getNetText()` oder `postNetText()` verwenden.

Wenn keine Netzwerk-ID verfügbar ist, können Sie in bestimmten Situationen auch eine URL verwenden, um die Übertragung von Daten für die betreffende URL anzuhalten. Die URL muss mit der URL identisch sein, mit welcher der Netzwerkvorgang gestartet wurde. Ist die Datenübertragung abgeschlossen, hat dieser Befehl keine Wirkung.

Parameter
URL Erforderlich. Gibt die abzubrechende URL an.

netID Optional. Gibt die ID des abzubrechenden Netzwerkvorgangs an.

Beispiel
Die folgende Anweisung übergibt die Netzwerk-ID `network ID` an `netAbort`, um einen bestimmten Netzwerkvorgang abzubrechen:

```
-- Lingo syntax
on mouseUp
    netAbort(myNetID)
end

// JavaScript syntax
function mouseUp() {
    netAbort(myNetID);
}
```

Siehe auch
`getNetText()`, `postNetText`

netByteArrayResult

Syntax

```
netByteArrayResult(netID)
```

Beschreibung

Diese Net Lingo-Methode ruft das Ergebnis der Abfrage als Bytearray ab.

Parameter

Parameter	Beschreibung	Erforderlich/Optional
netID	Gibt die ID des Netzwerkvorgangs an, der den zurückzugebenden Text enthält.	Optional

Beispiele

```
--Lingo syntax
put netByteArrayResult(ID)

//JavaScript syntax
put(netByteArrayResult(ID);
```

netDone()

Syntax

```
netDone()
netDone(netID)
```

Beschreibung

Diese Funktion gibt an, ob ein im Hintergrund durchgeführter Ladevorgang (wie `getNetText, preloadNetThing, gotoNetMovie`, `gotoNetPage` oder `netTextResult`) erfolgreich abgeschlossen oder aufgrund eines Browserfehlers vorzeitig abgebrochen wurde (`TRUE`, Standard) oder nach wie vor abläuft (`FALSE`).

- Anhand von `netDone()` können Sie den letzten Netzwerkvorgang testen.
- Anhand von `netDone(`*netID*`)` können Sie den letzten durch `netID` identifizierten Netzwerkvorgang testen.

Die Funktion `netDone` gibt 0 zurück, wenn ein Ladevorgang im Hintergrund aktiv ist.

Parameter

netID Optional. Gibt die ID des zu testenden Netzwerkvorgangs an.

Beispiel

Die folgende Prozedur testet anhand der Funktion `netDone`, ob der letzte Netzwerkvorgang beendet wurde. Ist der Vorgang beendet, wird der von `netTextResult` zurückgegebene Text im Felddarsteller `Display Text` angezeigt.

```
-- Lingo syntax
on exitFrame
    if netDone() = 1 then
        member("Display Text").text = netTextResult()
    end if
end

// JavaScript syntax
function exitFrame() {
    if (netDone() == 1) {
        member("Display Text").text = netTextResult();
    }
}
```

Die folgende Prozedur verwendet eine spezifische Netzwerk-ID als Argument für `netDone`, um den Status eines bestimmten Netzwerkvorgangs zu überprüfen:

```
-- Lingo syntax
on exitFrame
    -- stay on this frame until the net operation is
    -- completed
    global mynetID
    if netDone(mynetID) = FALSE then
        go to the frame
    end if
end

// JavaScript syntax
function exitFrame() {
    // stay on this frame until the net operation is completed
    global mynetID;
    if (!(netDone(mynetID))) {
        _movie.go(_movie.frame);
    }
}
```

Siehe auch

`getNetText()`, `netTextResult()`, `gotoNetMovie`, `preloadNetThing()`

netError()

Syntax

```
netError()
netError(netID)
```

Beschreibung

Diese Funktion ermittelt, ob während eines Netzwerkvorgangs ein Fehler aufgetreten ist, und gibt in diesem Fall eine Fehlernummer zurück, die einer bestimmten Fehlermeldung entspricht. War der Vorgang erfolgreich, gibt diese Funktion einen Code zurück, der darauf hinweist, dass der Vorgang ordnungsgemäß verlaufen ist. Wenn noch kein Ladevorgang im Hintergrund gestartet wurde oder der Vorgang gerade ausgeführt wird, gibt diese Funktion einen leeren String zurück.

- Anhand von `netError()` können Sie den letzten Netzwerkvorgang testen.

- Anhand von `netError(`*netID*`)` können Sie den letzten von `netID` angegebenen Netzwerkvorgang testen.

Es können verschiedene Fehlercodes zurückgegeben werden:

0	Alles in Ordnung.
1	Tritt auf, wenn der lokale Pfad auf ein nicht vorhandenes Verzeichnis verweist.
4	MOA-Klasse ungültig. Die erforderlichen Netzwerk- oder Nicht-Netzwerk-Xtra-Erweiterungen wurden nicht ordnungsgemäß oder überhaupt nicht installiert.
5	MOA-Schnittstelle ungültig. Siehe 4.
6	URL oder MOA-Klasse ungültig. Die erforderlichen Netzwerk- oder Nicht-Netzwerk-Xtra-Erweiterungen wurden nicht ordnungsgemäß oder überhaupt nicht installiert.
900	Die Datei, in die geschrieben werden soll, ist schreibgeschützt.
903	Der Datenträger ist voll.
905	Ungültige Dateispezifikation.
2018	Ein `postNetText`-Fehler tritt gewöhnlich auf, wenn `postNetText` mit den Parametern in der URL verwendet wird. M. Kloss empfiehlt, diese Syntax zu vermeiden 2018: `pNetID = postNetText(myURL, myParamPropList)`
4144	Fehler bei Netzwerkvorgang.
4145	Fehler bei Netzwerkvorgang.
4146	Zum Remote-Host konnte keine Verbindung hergestellt werden.
4149	Vom Server bereitgestellte Daten weisen ein unerwartetes Format auf.
4150	Verbindung wurde unerwartet vorzeitig abgebrochen.
4154	Aufgrund einer Zeitüberschreitung konnte der Vorgang nicht zu Ende geführt werden.
4155	Nicht genügend Speicher verfügbar, um die Transaktion abzuschließen.
4156	Protokollantwort auf Anfrage weist auf einen Fehler in der Antwort hin.
4157	Transaktion konnte nicht authentifiziert werden.
4159	URL ungültig.
4164	Socket konnte nicht erstellt werden.
4165	Angefordertes Objekt konnte nicht gefunden werden (URL möglicherweise falsch).
4166	Allgemeiner Proxy-Fehler.
4167	Übertragung wurde absichtlich vom Client unterbrochen.
4242	Download angehalten von `netAbort(url)`.
4836	Download angehalten – Ursache unbekannt; möglicherweise Netzwerkfehler, oder Download wurde abgebrochen.
4153	Fehler bei Netzwerkvorgang.
4154	Aufgrund einer Zeitüberschreitung konnte der Vorgang nicht zu Ende geführt werden.
4155	Nicht genügend Speicher verfügbar, um die Transaktion abzuschließen.
4157	Transaktion konnte nicht authentifiziert werden.
4159	URL ungültig.

4160	Fehler bei Netzwerkvorgang.
4161	Fehler bei Netzwerkvorgang.
4162	Fehler bei Netzwerkvorgang.
4163	Fehler bei Netzwerkvorgang.
4164	Socket konnte nicht erstellt werden.
4165	Angefordertes Objekt konnte nicht gefunden werden (URL möglicherweise falsch).
4166	Allgemeiner Proxy-Fehler.
4167	Übertragung wurde absichtlich vom Client unterbrochen.
4168	Fehler bei Netzwerkvorgang.
4240	Die Netzwerk-Xtras wurden nicht ordnungsgemäß initialisiert. (Lewis Francis)
4242	Download angehalten von `netAbort(url)`
4836	Download angehalten – Ursache unbekannt; möglicherweise Netzwerkfehler, oder Download wurde abgebrochen.

Parameter
netID Optional. Gibt die ID des zu testenden Netzwerkvorgangs an.

Beispiel
Die folgende Anweisung übergibt eine Netzwerk-ID an `netError`, um den Fehlerstatus eines bestimmten Netzwerkvorgangs zu überprüfen:

```
--Lingo syntax
on exitFrame
    global mynetID
    if netError(mynetID)<>"OK" then beep
end

// JavaScript syntax
function exitFrame() {
    global mynetID;
    if (netError(mynetID) != "OK") {
        _sound.beep();
    }
}
```

netLastModDate()

Syntax
`netLastModDate()`

Beschreibung
Diese Funktion gibt für das angegebene Objekt das Datum der letzten Änderung aus dem HTTP-Header zurück. Der String ist im Universal-Time-Format (GMT): *Ddd, nn Mmm jjjj hh:mm:ss* `GMT` (Beispiel: Thu, 30 Jan 1997 12:00:00 AM GMT). Die Tages- und Monatsnamen sind entweder ganz oder teilweise ausgeschrieben. Der String ist stets auf Englisch.

Die Funktion `netLastModDate` kann erst aufgerufen werden, nachdem `netDone` und `netError` den erfolgreichen Abschluss des Vorgangs gemeldet haben. Nach dem Start des nächsten Vorgangs verwirft der Director-Film oder der Projektor die Ergebnisse des vorherigen Vorgangs, um nicht unnötig Speicherplatz zu belegen.

Der tatsächliche Datumsstring wird direkt aus dem HTTP-Header im vom Server bereitgestellten Format abgerufen. Dieser String wird jedoch nicht immer bereitgestellt – in diesem Fall gibt `netLastModDateEMPTY` zurück.

Parameter

Keiner

Beispiel

Die folgenden Anweisungen überprüfen das Datum einer aus dem Internet heruntergeladenen Datei:

```
-- Lingo syntax
if netDone() then
    theDate = netLastModDate()
    if theDate.char[6..11] <> "Jan 30" then
        alert "The file is outdated."
    end if
end if

// JavaScript syntax
if (netDone()) {
    theDate = netLastModDate();
    if (theDate.char[6..11] != "Jan 30") {
        alert("The file is outdated");
    }
}
```

Siehe auch

`netDone()`, `netError()`

netMIME()

Syntax

```
netMIME()
```

Beschreibung

Diese Funktion stellt den MIME-Typ der Internet-Datei bereit, die vom letzten Netzwerkvorgang zurückgegeben wurde (das zuletzt heruntergeladene HTTP- oder FTP-Objekt).

Die Funktion `netMIME` kann erst aufgerufen werden, nachdem `netDone` und `netError` den erfolgreichen Abschluss des Vorgangs gemeldet haben. Nach dem Start des nächsten Vorgangs verwirft der Director-Film oder der Projektor die Ergebnisse des vorherigen Vorgangs, um nicht unnötig Speicherplatz zu belegen.

Parameter

Keiner

Beispiel

Die folgende Prozedur überprüft den MIME-Typ eines aus dem Internet heruntergeladenen Objekts und reagiert entsprechend:

```
-- Lingo syntax
on checkNetOperation theURL
    if netDone (theURL) then
        set myMimeType = netMIME()
        case myMimeType of
            "image/jpeg": go frame "jpeg info"
            "image/gif": go frame "gif info"
            "application/x-director": goToNetMovie theURL
            "text/html": gotoNetPage theURL
            otherwise: alert "Please choose a different item."
        end case
    else
        go the frame
    end if
end

// JavaScript syntax
function checkNetOperation(theURL) {
    if (netDone(theURL)) {
        myMimeType = netMIME();
        switch (myMimeType) {
            case "image/jpeg":
                _movie.go("jpeg info");
                break;
            case "image/gif":
                _movie.go("gif info");
                break;
            case "application/x-director":
                goToNetMovie(theURL);
                break;
            case "text/html":
                gotoNetPage(theURL);
                break;
            default:
                alert("Please choose a different item.");
        }
    } else {
        _movie.go(_movie.frame);
    }
}
```

Siehe auch

netDone(), netError(), getNetText(), postNetText, preloadNetThing()

netStatus

Syntax

```
netStatus msgString
```

Beschreibung
Dieser Befehl zeigt den angegebenen String im Statusbereich des Browserfensters an.

Der Befehl `netStatus` funktioniert in Projektoren nicht.

Parameter
msgString Erforderlich. Gibt die anzuzeigende Zeichenfolge an.

Beispiel
Die folgende Anweisung platziert die Zeichenfolge „Dies ist ein Test“ im Statusbereich des Browsers, in dem der Film abgespielt wird:

```
-- Lingo syntax
on exitFrame
    netStatus "This is a test"
end

// JavaScript syntax
function exitFrame() {
    _movie.netStatus("This is a test");
}
```

netTextResult()

Syntax
```
netTextResult(netID)
netTextResult()
```

Beschreibung
Diese Funktion gibt den Text zurück, der beim angegebenen Netzwerkvorgang erfasst wurde. Ist keine Netzwerk-ID angegeben, gibt `netTextResult` das Ergebnis des letzten Netzwerkvorgangs zurück.

Wenn der angegebene Netzwerkvorgang `getNetText()` war, wird der Text der Datei im Netzwerk zurückgegeben.

Wenn der angegebene Netzwerkvorgang `postNetText` war, ist das Ergebnis die Serverantwort.

Nach dem Start des nächsten Vorgangs verwirft Director die Ergebnisse des vorherigen Vorgangs, um nicht unnötig Speicherplatz zu belegen.

Director hält die Ergebnisse für die letzten sechs Netzwerkvorgänge in allen Wiedergabeumgebungen vor. Nachdem die Daten mithilfe von`netTextResult()` abgerufen wurden, werden die Ergebnisse für diese Netzwerkanforderung verworfen.

Parameter
netID Optional. Gibt die ID des Netzwerkvorgangs an, der den zurückzugebenden Text enthält.

Beispiel
Die folgende Prozedur testet mit den Funktionen `netDone` und `netError`, ob der letzte Netzwerkvorgang erfolgreich abgeschlossen wurde. Ist der Vorgang beendet, wird der von `netTextResult`zurückgegebene Text im Felddarsteller `Display Text` angezeigt.

```
-- Lingo syntax
global gNetID
on exitFrame
    if (netDone(gNetID) = TRUE) and (netError(gNetID) = "OK") then
        member("Display Text").text = netTextResult()
    end if
end

// JavaScript syntax
global gNetID;
function exitFrame() {
    if (netDone(gNetID) && (netError(gNetID) == "OK")) {
        member("Display Text").text = netTextResult();
    }
}
```

Siehe auch

netDone(), netError(), postNetText

new()

Syntax

```
new(type)
new(type, castLib whichCast)
new(type, member whichCastMember of castLib whichCast)
variableName = new(parentScript arg1, arg2, ...)
new(script parentScriptName, value1, value2, ...)
timeout("name").new(timoutPeriod, #timeoutHandler, {, targetObject})
new(xtra "xtraName")
```

Beschreibung

Diese Funktion erstellt einen neuen Darsteller, ein neues Child-Objekt oder eine Xtra-Instanz. Außerdem können Child-Objekten damit individuelle Eigenschaftswerte zugeordnet werden.

Für Darsteller wird mit dem Parameter *type* der Darstellertyp festgelegt. Die möglichen vordefinierten Werte entsprechen den definierten Darstellertypen: `#bitmap`, `#field` usw. Mit der Funktion `new` können Sie auch Xtra-Darstellertypen erstellen, die anhand eines vom Autor gewählten Namens identifiziert werden können.

Außerdem kann mit dem Xtra "Benutzerdefinierter Cursor" schnell ein neuer Farbcursor-Darsteller erstellt werden. Verwenden Sie `new(#cursor)`, und legen Sie die Eigenschaften des daraus resultierenden Darstellers so fest, dass sie einsatzbereit sind.

Die optionalen Parameter *whichCastMember* und *whichCast* geben die Darstellerposition und das Besetzungsfenster an, in dem der neue Darsteller gespeichert wird. Ist keine Darstellerposition angegeben, wird die erste leere Position verwendet. Die Funktion `new` gibt die Darstellerposition zurück.

Wenn das Argument für die Funktion `new` ein Parent-Skript ist, erstellt die Funktion `new` ein Child-Objekt. Das Parent-Skript sollte eine `on new`-Prozedur enthalten, die den Anfangszustand oder die Eigenschaftswerte des Child-Objekts einstellt und den Bezug `me` an das Child-Objekt zurückgibt.

Das Child-Objekt enthält sämtliche Prozeduren des Parent-Skripts. Es weist außerdem die gleichen Eigenschaftsvariablennamen auf, die im Parent-Skript deklariert sind, aber jedes Child-Objekt hat für diese Eigenschaften seine eigenen Werte.

Da ein Child-Objekt ein Wert ist, kann es Variablen zugeordnet, in Listen gestellt und als Parameter übergeben werden.

Wie bei anderen Variablen können Sie auch hier den Befehl `put` zum Anzeigen von Informationen über ein Child-Objekt im Nachrichtenfenster verwenden.

Wenn mit `new()` ein Timeout-Objekt erstellt wird, bestimmt `timeoutPeriod`, wie viele Millisekunden zwischen den vom Timeout-Objekt gesendeten Timeout-Ereignissen verstreichen. *#timeoutHandler* ist ein Symbol zur Identifizierung der Prozedur, die beim Auftreten eines Timeout-Ereignisses aufgerufen wird. *TargetObject* gibt den Namen des Child-Objekts mit *#timeoutHandler* an. Wenn kein *targetObject* angegeben ist, wird davon ausgegangen, dass sich *#timeoutHandler* in einem Filmskript befindet. Die Syntax zum Erstellen eines Timeouts kann in Abhängigkeit von der Einstellung `scriptExecutionStyle` variieren.

```
-- Lingo syntax when scriptExecutionStyle is set to 9
x = timeout(name).new(period,handler,targetData)

-- Lingo syntax when scriptExecutionStyle is set to 10
x = timeout().new(name, period, handler, targetData)
y = new timeout(name, period, handler, targetData)

// JavaScript syntax
x = new timeout(name, period, function, targetData)
```

Bei der Erstellung eines Timeout-Objekts wird das zugehörige *targetObject* für die Systemereignisse `prepareMovie`, `startMovie`, `stopMovie`, `prepareFrame` und `exitFrame` aktiviert. Hierzu muss das *targetObject* Prozeduren für diese Ereignisse enthalten. Die Ereignisse brauchen nicht übergeben zu werden, damit der restliche Film auf sie zugreifen kann.

Hinweis: *Ein mit Lingo erstelltes Timeout-Objekt kann eine JavaScript-Syntax-Funktion aufrufen und umgekehrt.*

Ein Beispiel für `new()` in einem fertigen Film finden Sie in den Filmen „Parent Scripts“ und „Read and Write Text“ im Ordner „Learning\\Lingo_“ im Director-Anwendungsordner.

Beispiel

Mit der folgenden Syntax erstellen Sie einen neuen Bitmapdarsteller an der ersten verfügbaren Position:

```
set newMember = new(#bitmap)
```

Nach Ausführung der Zeile enthält `newMember` einen Bezug auf den gerade erstellten Darsteller:

```
put newMember
-- (member 1 of castLib 1)
```

Wenn Sie mithilfe von JavaScript-Syntax einen neuen Darsteller erstellen, verwenden Sie die `newMember()`-Methode des Movie-Objekts. Die folgende Anweisung erstellt einen neuen Bitmapdarsteller:

```
var tMem = _movie.newMember(symbol("bitmap"))
```

Das folgende `startMovie`-Skript erstellt unter Verwendung des Befehls new einen neuen Flash-Darsteller und stellt dann für diesen Darsteller die Eigenschaft linked ein, so dass die Elemente dieses Darstellers in einer externen Datei gespeichert werden. Anschließend setzt das Skript die Eigenschaft `pathName` dieses Darstellers auf die Adresse eines Flash-Films im Internet:

```
on startMovie
    flashCastMember = new(#flash)
    member(flashCastMember).pathName = "http://www.someURL.com/myFlash.swf"
end
```

Beim Start des Films erstellt diese Prozedur einen neuen animierten Farbcursor-Darsteller und speichert dessen Darstellernummer in einer Variablen namens `customCursor`. Diese Variable wird dazu verwendet, die Eigenschaft `castMemberList` des neu erstellten Cursors einzustellen und auf den neuen Cursor umzuschalten.

```
on startmovie
    customCursor = new(#cursor)
    member(customCursor).castMemberList = [member 1, member 2, member 3]
    cursor (member(customCursor))
end
```

Die folgenden Anweisungen aus einem Parent-Skript enthalten die Prozedur on new zum Erstellen eines Child-Objekts. Das Parent-Skript ist ein Skriptdarsteller namens "Bird", der diese Prozeduren enthält.

```
on new me, nameForBird
    return me
end

on fly me
    put "I am flying"
end
```

Durch die erste Anweisung im folgenden Beispiel wird ein Child-Objekt aus dem obigen Skript des vorhergehenden Beispiels erstellt und in eine Variable namens `myBird` gestellt. Die zweite Anweisung lässt den Vogel fliegen, indem sie die Prozedur „fly“ im Parent-Skript `Bird` aufruft:

```
myBird = script("Bird").new()
myBird.fly()
```

Die folgende Anweisung verwendet ein neues Parent-Skript „Bird“, das die Eigenschaftsvariable speed enthält:

```
property speed
on new me, initSpeed
    speed = initSpeed
    return me
end
on fly me
    put "I am flying at " & speed & "mph"
end
```

Die folgenden Anweisungen erstellen zwei Child-Objekte namens `myBird1` und `myBird2`. Den Objekten werden unterschiedliche Anfangsgeschwindigkeiten zugeteilt: 15 und 25. Wenn die Prozedur `fly` für jedes Child-Objekt aufgerufen wird, wird im Nachrichtenfenster die Geschwindigkeit des Objekts angezeigt.

```
myBird1 = script("Bird").new(15)
myBird2 = script("Bird").new(25)
myBird1.fly()
myBird2.fly()
```

Die folgende Meldung erscheint im Nachrichtenfenster:

```
-- "I am flying at 15 mph"
-- "I am flying at 25 mph"
```

Die folgende Anweisung erstellt ein neues Timeout-Objekt namens `intervalTimer`, das alle 60 Sekunden ein Timeout-Ereignis an die Prozedur `MinuteBeep` im Child-Objekt `playerOne` sendet:

```
timeout("intervalTimer").new(60000, #minuteBeep, playerOne)
```

Die folgende Anweisung erstellt eine JavaScript-Beispielfunktion:

```
function sampleTimeout () { trace("hello"); }
```

An anderer Stelle im Film können Sie mithilfe dieser Anweisung ein Timeout-Objekt erstellen, das die JavaScript-Syntax-Funktion aufruft:

```
-- Lingo syntax
gTO = timeout().new("test",50,"sampleTimeout",0)

// JavaScript syntax
_global.gTO = new timeout("test",50,"sampleTimeout",0)
```

Siehe auch

on stepFrame, actorList, ancestor, end, type (Darsteller), timeout()

newCamera

Syntax

```
member(whichCastmember).newCamera(newCameraName)
```

Beschreibung

Dieser 3D-Befehl erstellt in einem Darsteller eine neue Kamera.

Parameter

newCameraName Erforderlich. Gibt den Namen der neuen Kamera an. Der Name der neuen Kamera muss im Darsteller eindeutig sein.

Beispiel

Diese Anweisung erstellt die neue Kamera „Kamera_in_Auto“

```
--Lingo
member("3D World").newCamera("in-car camera")

// Javascript
member("3D World").newCamera("in-car camera") ;
```

newColorRatio

Syntax

```
<ColorRatio> newColorRatio(color, ratio, alpha)
```

Beschreibung

Globale Funktion, die ein neues `ColorRatio`-Objekt erstellt, das zur Angabe der Farbanteile dient, die bei Verwendung eines Verlaufslicht- oder Verlaufsabschrägungs-Filters auf den verschiedenen Verhältnisebenen verwendet werden sollen.

Parameter

Parameter	Beschreibung
Farbe	Erforderlich. Farbobjekt, das aus Rot-, Grün- und Blauanteilen der Farbe besteht.
ratio	Erforderlich. Dieser Wert zwischen 0 und 255 bestimmt den Anfangspunkt für die Farbverteilung für das aktuelle `ColorRatio`-Objekt in der Farbliste `ColorList` von Verlaufsfiltern.
alpha	Optional. Transparenz (0-255) der Farbe.

Beispiel

Der folgende Codeauszug erstellt einen Verlaufslicht-Filter (`GradientGlowFilter`) und weist ihm verschiedene Farbverhältnisse zu.

```
--Lingo
on mouseUp me
     fil = filter(#GradientGlowFilter, [#distance:0, #blurX:10, #blurY:10, #strength:1.3,
#inner:0])
     fil.colorList.append(newColorRatio(color(255,255,255),1,0))
     fil.colorList.append(newColorRatio(color(255,0,0),63,255))
     fil.colorList.append(newColorRatio(color(255,255,0),126,25))
     fil.colorList.append(newColorRatio(color(0,204,255),255,255))
     sprite(me.spriteNum).filterList.append(fil)
end
//Java Script
function mouseUp(me)
{
     fil = filter(symbol("GradientGlowFilter"), propList(symbol("distance"),0,
symbol("blurX"), 10, symbol("blurY"), 10, symbol("strength"), 1.3, symbol("inner"), 0));
     fil.colorList.append(newColorRatio(color(255,255,255),1,0));
     fil.colorList.append(newColorRatio(color(255,0,0),63,255));
     fil.colorList.append(newColorRatio(color(255,255,0),126,255));
     fil.colorList.append(newColorRatio(color(0,204,255),255,255));
     sprite(me.spriteNum).filterList.append(fil);
}
```

newCurve()

Syntax

```
-- Lingo syntax
memberObjRef.newCurve(positionInVertexList)

// JavaScript syntax
memberObjRef.newCurve(positionInVertexList);
```

Beschreibung

Diese Funktion fügt ein `#newCurve`-Symbol zur `vertexList` des *vectorCastMember* hinzu, der die Vektorform durch eine neue Form ergänzt. Sie können eine vorhandene Form teilen, indem Sie beim Aufruf von `newCurve()` eine Position angeben, die in der Mitte einer Scheitelpunktserie liegt.

Parameter

positionInVertexList Erforderlich. Gibt die Position in der Liste der Scheitelpunkte (`vertexList`) an, an der das `#newCurve`-Symbol hinzugefügt werden soll.

Beispiel

Die folgenden Anweisungen fügen eine neue Kurve zu Darsteller 2 an der dritten Position in der Scheitelpunktliste (`vertexList`) des Darstellers hinzu. In der zweiten Zeile des Beispiels wird der Inhalt von Kurve 2 durch den Inhalt von Kurve 3 ersetzt.

```
-- Lingo syntax
member(2).newCurve(3)
member(2).curve[2] = member(2).curve[3]

// JavaScript syntax
member(2).newCurve(3);
member(2).curve[2] = member(2).curve[3]
```

Siehe auch

`curve`, `vertexList`

newGroup

Syntax

```
member(whichCastmember).newGroup(newGroupName)
```

Beschreibung

Dieser 3D-Befehl erstellt eine neue Gruppe und fügt sie zur Gruppenpalette hinzu.

Parameter

newGroupName Erforderlich. Gibt den Namen der neuen Gruppe an. Der Name der neuen Gruppe muss in der Gruppenpalette eindeutig sein.

Beispiel

Die folgende Anweisung erstellt eine Gruppe namens `gbGroup2` im Darsteller `Scene` und speichert einen Verweis auf sie in der Variablen `ng`:

```
-- Lingo
ng = member("Scene").newGroup("gbGroup2")
// Javascript
var ng = member("Scene").newGroup("gbGroup2") ;
```

newLight

Syntax

```
member(whichCastmember).newLight(newLightName, #typeIndicator)
```

Beschreibung
Dieser 3D-Befehl erstellt ein neues Licht eines vorgegebenen Typs und fügt es zur Lichtpalette hinzu.

Parameter
newLightName Erforderlich. Gibt den Namen des neuen Lichts an. Der Name des neuen Lichts muss in der Lichtpalette eindeutig sein.

typeIndicator Erforderlich. Ein Symbol, das den Typ des neuen Lichts angibt. Folgende Werte sind gültig:

- `#ambient` ist ein allgemeines Licht in der 3D-Welt.
- `#directional` ist ein Licht aus einer bestimmten Richtung.
- `#point` ist eine Lichtquelle wie eine Lichtbirne.
- `#spot` ist ein Spotlichteffekt.

Beispiel
Die folgende Anweisung erstellt ein neues Licht im Darsteller "3D-Welt". Es handelt sich um ein Umgebungslicht namens "Umgebungsraumlicht".

```
-- Lingo
member("3D World").newLight("ambient room light", #ambient)

// Javascript
member("3D World").newLight("ambient room light", symbol("ambient"));
```

newMatrix()

Syntax
```
<Matrix> newMatrix(numRows, numColumns, {elementList})
```

Beschreibung
Globale Funktion, die ein neues Matrixobjekt mit der angegebenen Anzahl von Zeilen und Spalten erstellt. Die Indizes für Zeilen und Spalten beginnen bei 1.

Hinweis: *Um die Werte einer Matrix anzuzeigen, die mithilfe von „newMatrix“ im Debug-Modus erstellt wurde, fragen Sie die Werte mit der „getval(i,j)“-Methode ab.*

Parameter
numRows Erforderlich. Gibt die Anzahl der Zeilen in der Matrix an.

numColumns Erforderlich. Gibt die Anzahl der Spalten in der Matrix an.

elementList Erforderlich. Eine lineare Liste mit Fließkommazahlen.

Beispiel
Die folgende Anweisung erstellt eine Matrix mit 2 Zeilen und 3 Spalten, wobei die erste Zeile mit [1,2,3] und die zweite Zeile mit [4,5,6] angegeben wird.

```
-- Lingo
mat = newMatrix(2, 3, [1,2,3,4,5,6])

// Javascript
mat = newMatrix(2, 3, list(1,2,3,4,5,6));
```

Siehe auch

getVal(), setVal(), numRows(), numColumns(), matrixAddition(), matrixMultiply(), matrixMultiplyScalar(), matrixTranspose()

newMember()

Syntax

```
-- Lingo syntax
_movie.newMember(symbol)
_movie.newMember(stringMemberType)

// JavaScript syntax
_movie.newMember(stringMemberType);
```

Beschreibung

Diese Filmmethode erstellt einen neuen Darsteller und ermöglicht das Zuweisen individueller Eigenschaftswerte zu Child-Objekten.

Bei neuen Darstellern wird mit dem Parameter *symbol* oder *stringMemberType* der Darstellertyp festgelegt. Die möglichen vordefinierten Werte entsprechen den definierten Darstellertypen: `#bitmap`, `#field`, usw. Mit der `newMember()`-Methode können Sie auch Xtra-Darstellertypen erstellen, die anhand eines vom Autor gewählten Namens identifiziert werden können.

Außerdem kann mit dem Xtra "Benutzerdefinierter Cursor" schnell ein neuer Farbcursor-Darsteller erstellt werden. Verwenden Sie `newMember(#cursor)` und legen Sie die Eigenschaften des daraus resultierenden Darstellers so fest, dass sie einsatzbereit sind.

Nach dem Aufrufen von `newMember()` wird der neue Darsteller an der ersten freien Position in der Besetzungsbibliothek platziert.

Ein Beispiel für `newMember()` in einem fertigen Film finden Sie in den Filmen „Parent Scripts“ und „Read and Write Text“ im Ordner „Learning/Lingo“ im Director-Anwendungsordner.

Parameter

symbol (nur Lingo) Erforderlich. Ein Symbol, das den Typ des neuen Darstellers angibt.

stringMemberType Erforderlich. Eine Zeichenfolge, die den Typ des neuen Darstellers angibt.

Beispiel

Die folgenden Anweisungen erstellen einen neuen Bitmapdarsteller und weisen diesen der Variablen `newBitmap` zu.

```
-- Lingo syntax
newBitmap = _movie.newMember(#bitmap) -- using a symbol
newBitmap = _movie.newMember("bitmap") -- using a string

// JavaScript syntax
var newBitmap = _movie.newMember(symbol("bitmap")) ; //using a symbol
var newBitmap = _movie.newMember("bitmap") ; // using a string
```

Siehe auch

Movie, type (Darsteller)

newMesh

Syntax

```
member(whichCastmember).newMesh(name,numFaces, numVertices,
numNormals,numColors,numTextureCoordinates)
```

Beschreibung

Dieser 3D-Befehl erstellt eine neue Gitternetzmodellressource. Nach Erstellung eines Gitternetzes müssen Sie zumindest für die Eigenschaften `vertexList` und `face[`*index*`].vertices` des neuen Gitternetzes Werte festlegen und dann den Befehl `build()` aufrufen, um die Geometrie tatsächlich zu erzeugen.

Parameter

meshName Erforderlich. Gibt den Namen der neuen Gitternetzmodellressource an.

numFaces Erforderlich. Gibt ist die gewünschte Gesamtzahl von Dreiecken im Gitternetz an.

numVertices Erforderlich. Gibt die von allen (dreieckigen) Flächen verwendete Gesamtzahl von Scheitelpunkten an. Ein Scheitelpunkt kann von mehreren Flächen verwendet werden.

numNormals Optional. Gibt die Gesamtzahl der Normalenvektoren an. Ein Normalenvektor kann von mehreren Flächen verwendet werden. Der Normalenvektor für ein Polygon bestimmt, welche Seite "außen" darstellt. Dadurch wird bestimmt, wie diese Ecke durch Lichtquellen beleuchtet wird. Geben Sie 0 ein oder lassen Sie diesen Parameter aus, wenn Sie die Normalenvektoren mit dem Gitternetzbefehl `generateNormals()` erzeugen.

numColors Optional. Gibt die von allen Flächen verwendete Gesamtzahl von Farben an. Eine Farbe kann von mehreren Flächen verwendet werden. Sie können für jede Ecke einer Fläche eine Farbe angeben. Bei Angabe von Farben ergeben sich glattere Verlaufeffekte. Geben Sie 0 ein oder lassen Sie diesen Parameter aus, um für jede Flächenecke die Standardfarbe (Weiß) zu verwenden.

numTextureCoordinates Optional. Gibt die von allen Flächen verwendete Anzahl benutzerdefinierter Texturkoordinaten an. Geben Sie 0 ein oder lassen Sie diesen Parameter aus, um die per Planar Mapping generierten Standardtexturkoordinaten zu verwenden. (Weitere Einzelheiten finden Sie unter `#planar` im Eintrag zu `shader.textureWrapMode`.) Geben Sie die Texturkoordinaten an, wenn Sie genau steuern möchten, wie Texturen auf die Gitternetzflächen gemappt werden.

Beispiel

In diesem Beispiel werden eine Modellressource vom Typ `#mesh` erstellt, seine Eigenschaften festgelegt und dann ein neues Modell aus der Modellressource erstellt. Die genaue Verfahrensweise wird in der folgenden Erläuterung des Beispielcodes beschrieben:

In **Zeile 1** wird ein Gitternetz mit 6 Flächen erstellt, die aus 5 eindeutigen Scheitelpunkten und 3 eindeutigen Farben bestehen. Die Anzahl von Normalenvektoren und die Anzahl von Texturkoordinaten (`textureCoordinates`) ist nicht festgelegt. Die Normalenvektoren werden mit dem Befehl `generateNormals` erstellt.

In **Zeile 2** werden die fünf eindeutigen Scheitelpunkte definiert, die von den Gitternetzflächen verwendet werden.

In **Zeile 3** werden die drei eindeutigen Farben definiert, die von den Gitternetzflächen verwendet werden.

In **Zeile 4 bis 9** wird festgelegt, welche Scheitelpunkte als Ecken der einzelnen Flächen in der `Pyramid`e verwendet werden sollen. Die Scheitelpunkte werden im Uhrzeigersinn angegeben. Diese Reihenfolge ist für `GenerateNormals()` wichtig.

In **Zeile 10 bis 15** werden den Ecken der einzelnen Flächen Farben zugeordnet. Die Farben erstrecken sich über die Flächen in Form von Farbverläufen.

In **Zeile 16** werden die Normalenvektoren von `Triangle` durch Aufruf des Befehls `generateNormals()` erstellt.

In **Zeile 17** wird der Befehl build zur Erstellung des Gitternetzes aufgerufen.

```
nm = member("Shapes").newMesh("pyramid",6 , 5, 0, 3)
nm.vertexList = [ vector(0,0,0), vector(40,0,0), vector(40,0,40), vector(0,0,40),
vector(20,50,20) ]
nm.colorList = [ rgb(255,0,0), rgb(0,255,0), rgb(0,0,255) ]
nm.face[1].vertices = [ 4,1,2 ]
nm.face[2].vertices = [ 4,2,3 ]
nm.face[3].vertices = [ 5,2,1 ]
nm.face[4].vertices = [ 5,3,2 ]
nm.face[5].vertices = [ 5,4,3 ]
nm.face[6].vertices = [ 5,1,4 ]
nm.face[1].colors = [3,2,3]
nm.face[2].colors = [3,3,2]
nm.face[3].colors = [1,3,2]
nm.face[4].colors = [1,2,3]
nm.face[5].colors = [1,3,2]
nm.face[6].colors = [1,2,3]
nm.generateNormals(#flat)
nm.build()
nm = member("Shapes").newModel("Pyramid1", nm)
```

Siehe auch

`newModelResource`

newModel

Syntax

```
member( whichCastmember ).newModel( newModelName {, whichModelResource } )
```

Beschreibung

Dieser 3D-Befehl erstellt ein neues Modell im referenzierten Darsteller. Bei allen neuen Modellen ist die Eigenschaft `resource` standardmäßig auf `VOID` gesetzt.

Parameter

newModelName Required. Gibt den Namen des neuen Modells an. Der Name des neuen Modells muss eindeutig sein.

whichModelResource Optional. Gibt eine Modellressource an, aus der das Modell erstellt werden soll.

Beispiel
Die folgende Anweisung erstellt ein Modell namens "neuesHaus" im Darsteller "3D-Welt":

```
-- Lingo
member("3D World").newModel("New House")

// Javascript
member("3D World").newModel("New House") ;
```

Die Modellressource für das neue Modellobjekt kann auch mit dem optionalen Parameter `whichModelResource` festgelegt werden.

```
member("3D World").newModel("New House", member("3D World").modelResource("bigBox"))
```

newModelResource

Syntax

```
member(whichCastmember).newModelResource(newModelResourceName { ,#type, #facing })
```

Beschreibung
Dieser 3D-Befehl erstellt eine neue Modellressource mit optional angegebenem Typ und Flächigkeit und fügt sie zur Modellressourcenpalette hinzu.

Wenn Sie den Parameter *facing* auslassen und `#box`, `#sphere`, `#particle` oder `#cylinder` für den Parameter *type* angeben, werden nur die Vorderseiten generiert. Wenn Sie `#plane` angeben, werden sowohl die Vorder- als auch die Rückseiten generiert. Für Modellressourcen vom Typ `#plane` werden zwei Gitternetze generiert, eines pro Fläche. Diese Modellressourcen weisen deshalb in der `shaderList` zwei Shader auf.

Bei Angabe von `#both` werden doppelt so viele Gitternetze und somit doppelt so viele Shader-Einträge in der `shaderList` erzeugt: 2 für Ebenen und Kugeln (für die Innen- und Außenseite des Modells), 12 für Würfel (6 auf der Außen- und 6 auf der Innenseite) und 6 für Zylinder (Oberseite, Rumpf und Unterseite außen und Oberseite, Rumpf und Unterseite innen).

Parameter
newModelResourceName Erforderlich. Gibt den Namen der neuen Modellressource an.

type Optional. Gibt den Primitiventyp der neuen Modellressource an. Folgende Werte sind gültig:

- `#plane`
- `#box`
- `#sphere`
- `#cylinder`
- `#particle`

facing Optional. Gibt die Fläche der neuen Modellressource an. Folgende Werte sind gültig:

- `#front`
- `#back`
- `#both`

Beispiel

Durch die folgende Prozedur wird eine Box erstellt. In der ersten Zeile der Prozedur wird eine neue Modellressource namens `box10`. Ihr Typ lautet `#box` und sie ist so definiert, dass nur ihre Rückseite zu sehen ist. In den nächsten drei Zeilen werden die Abmessungen von `box10` festgelegt. In der letzten Zeile wird ein neues Modell erstellt, das `box10` als Modellressource verwendet.

```
on makeBox
    nmr = member("3D").newModelResource("box10", #box, #back)
    nmr.height = 50
    nmr.width = 50
    nmr.length = 50
    aa = member("3D").newModel("gb5", nmr)
end
```

Die folgende Anweisung erstellt eine boxförmige Modellressource namens „Hutschachtel4“:

```
member("Shelf").newModelResource("hatbox4", #box)
```

Siehe auch

`primitives`

newMotion()

Syntax

```
member(whichCastmember).newMotion(name)
```

Beschreibung

Dieser 3D-Befehl erstellt eine neue Bewegung in einem referenzierten Darsteller und gibt einen Verweis auf die neue Bewegung zurück. Über den Befehl `map()` lassen sich mehrere in der Bewegungsliste des Darstellers vorhandene Bewegungen zu einer einzigen neuen Bewegung zusammenfassen.

Parameter

name Erforderlich. Gibt den Namen der neuen Bewegung an. Der Name der neuen Bewegung muss in dem referenzierten Darsteller eindeutig sein.

Beispiel

Der folgende Lingo-Code erstellt eine neue Bewegung namens `runWithWave` in Darsteller "1", mit der die Lauf- und Wellenbewegungen in der Bewegungsliste des Darstellers kombiniert werden:

```
runWithWave = member(1).newMotion("runWithWave")
runWithWave.map("run", "pelvisBone")
runWithWave.map("wave", "shoulderBone")
```

newObject()

Syntax

```
-- Lingo syntax
spriteObjRef.newObject(objectType {, arg1, arg2 ....})

// JavaScript syntax
spriteObjRef.newObject(objectType {, arg1, arg2 ....});
```

Beschreibung

Diese Befehl für Flash-Sprites erstellt ein ActionScript-Objekt mit dem angegebenen Typ.

Die folgende Syntax dient zur Erstellung eines Objekts in einem Flash-Sprite:

```
flashSpriteReference.newObject("objectType" {, arg1, arg2 ....})
```

Die folgende Syntax dient zur Erstellung eines globalen Objekts:

```
newObject("objectType" {, arg1, arg2 ....})
```

Hinweis: *Wenn Sie keine Flash-Darsteller importiert haben, müssen Sie das Flash Asset-Xtra manuell zur Xtra-Liste des Films hinzufügen, damit globale Flash-Befehle im Shockwave Player und in Projektoren ordnungsgemäß funktionieren. Xtra-Erweiterungen werden durch Auswahl von „Modifizieren“ > „Film“ > „Xtras“ in die Xtra-Liste aufgenommen. Weitere Informationen zum Verwalten von Xtra-Erweiterungen für verteilte Filme finden Sie in den Themen des Director-Benutzerhandbuchs im Director-Hilfefenster.*

Parameter

objectType Erforderlich. Gibt den Typ des neu zu erstellenden Objekts an.

arg1, *arg2*, ... Optional. Gibt alle für das Objekt erforderlichen Initialisierungsargumente an. Die einzelnen Argumente müssen durch Kommas getrennt sein.

Beispiel

Der folgende Lingo-Code setzt die Variable `tLocalConObject` auf einen Bezug auf ein neues `LocalConnection`-Objekt im Flash-Film in Sprite 3:

```
-- Lingo syntax
tLocalConObject = sprite(3).newObject("LocalConnection")

// JavaScript syntax
var tLocalConObject = sprite(3).newObject("LocalConnection");
```

Der folgende Lingo-Code setzt die Variable `tArrayObject` auf einen Bezug auf ein neues Array-Objekt im Flash-Film in Sprite 3. Das Array enthält die drei Ganzzahlen 23, 34 und 19.

```
-- Lingo syntax
tArrayObject = sprite(3).newObject("Array",23,34,19)

// JavaScript syntax
var tArrayObject = sprite(3).newObject("Array",23,34,19);
```

Siehe auch

`setCallback()`, `clearAsObjects()`

newShader

Syntax

```
member(whichCastmember).newShader(newShaderName, #shaderType)
```

Beschreibung

Dieser 3D-Befehl erstellt einen neuen Shader mit einem angegebenen Shader-Typ in der Shader-Liste eines referenzierten Darstellers und gibt einen Verweis auf den neuen Shader zurück.

Jeder Shader-Typ besitzt eine bestimmte Gruppe von Eigenschaften, die nur für ihn gültig sind. Darüber hinaus können alle Shader-Typen auf die Eigenschaften des `#standard`-Shaders zugreifen. Obwohl Sie jedem Shader-Typ eine beliebige `#standard`-Shader-Eigenschaft zuordnen können, erzeugt diese u.U. keine visuellen Effekt. Dies ist immer dann der Fall, wenn die `#standard`-Eigenschaft nicht mit dem Shader-Typ verträglich ist. Ein Beispiel hierfür ist die Standard-Shader-Eigenschaft `diffuseLightMap`, die von Shadern des Typs `#engraver`, `#newsprint`, und `#painter` ignoriert wird.

Parameter

newShaderName Erforderlich. Gibt den Namen des neuen Shaders an. Der Name des neuen Shaders muss in der Shader-Liste eindeutig sein.

shaderType Erforderlich. Ein Symbol, das die Art angibt, auf die der Shader angewendet wird. Folgende Werte sind gültig:

- `#standard`-Shader sind fotorealistisch und haben folgende Eigenschaften: `ambient`, `blend`, `blendConstant`, `blendConstantList`, `blendFunction`, `blendFunctionList`, `blendSource`, `blendSourceList`, `diffuse`, `diffuseLightMap`, `emissive`, `flat`, `glossMap`, `ilk`, `name`, `region`, `renderStyle`, `silhouettes`, `specular`, `specularLightMap`, `texture`, `textureMode`, `textureModeList`, `textureRepeat`, `textureRepeatList`, `textureTransform`, `textureTransformList`, `transparent`, `useDiffuseWithTexture`, `wrapTransform` und `wrapTransformList`.
- `#painter`-Shader sind geglättet und das Gesamtbild gleicht einem Gemälde. Zusätzlich zu den `#standard`-Eigenschaften haben diese Shader folgende Eigenschaften: `colorSteps`, `hilightPercentage`, `hilightStrength`, `name`, `shadowPercentage`, `shadowStrength` und `style`.
- `#engraver`-Shader sind liniert und sehen wie ein Kupferstich aus. Zusätzlich zu allen `#standard`-Eigenschaften weisen sie folgende Eigenschaften auf: `brightness`, `density`, `name` und `rotation`.
- `#newsprint`-Shaders erscheinen mit simulierten Punkten und sehen wie eine Zeitungsreproduktion aus. Zusätzlich zu den `#standard`-Eigenschaften weisen sie folgende Eigenschaften auf: `brightness`, `density` und `name`.

Beispiel

Die folgende Anweisung erstellt einen `#painter`-Shader namens `newPainter`:

```
-- Lingo
newPainter = member("3D World").newShader("newPainter",#painter)

// Javascript
var newPainter = member("3D World").newShader("newPainter",symbol("painter")) ;
```

Siehe auch

`shadowPercentage`, `shader`, `shaderList`, `deleteShader`

newTexture

Syntax

```
member(whichCastmember).newTexture(newTextureName {,#typeIndicator, sourceObjectReference})
```

Beschreibung

Dieser 3D-Befehl erstellt eine neue Textur in der Texturpalette des referenzierten Darstellers und gibt einen Bezug auf die neue Textur zurück. Darstellertexturen funktionieren nur dann, wenn Sie den Darsteller im Konstruktor `newTexture` angeben.

Parameter

newTextureName Erforderlich. Gibt den Namen der neuen Textur an. Der Name der neuen Textur muss in der Texturpalette des referenzierten Darstellers eindeutig sein.

typeIndicator Optional. Gibt den Typ der neuen Textur an. Wird der Parameter ausgelassen, wird die neue Textur ohne spezifischen Typ erstellt. Folgende Werte sind gültig:

- `#fromCastMember` (einDarsteller)
- `#fromImageObject` (ein Lingo-Grafikobjekt)

sourceObjectReference Optional. Gibt einen Verweis auf den Quelldarsteller oder das Lingo-Grafikobjekt an. Wird der Parameter ausgelassen, wird die neue Textur aus keiner bestimmten Quelle erstellt. *sourceObjectReference* muss auf einen Darsteller verweisen, wenn *typeIndicator* den Wert `#fromCastMember` hat, bzw. auf ein Lingo-Grafikobjekt, wenn *typeIndicator* den Wert `#fromImageObject` hat.

Beispiel

In der ersten Zeile der folgenden Anweisung wird eine neue Textur namens "Gras 02" aus Darsteller 5 in Besetzung 1 erstellt, in der zweiten Zeile eine leere neue Textur namens "Leer".

```
-- Lingo
member("3D World").newTexture("Grass 02",#fromCastMember,member(5,1))
member("3D World").newTexture("Blank")

// Javascript
member("3D World").newTexture("Grass 02",symbol("fromCastMember"),member(5,1)) ;
member("3D World").newTexture("Blank") ;
```

Siehe auch

deleteTexture, texture

normalize

Syntax

```
normalize(vector)
vector.normalize()
```

Beschreibung

Dieser 3D-Befehl normiert einen Vektor durch Dividieren der *x*-, *y*- und *z*-Komponenten durch den Vektorbetrag. Normierte Vektoren weisen immer den Betrag 1 auf.

Parameter
Keiner

Beispiel
Die folgende Anweisung zeigt den Wert des Vektors `MyVec` vor und nach der Normierung:

```
-- Lingo
MyVec = vector(-209.9019, 1737.5126, 0.0000)
MyVec.normalize()
put MyVec
-- vector(-0.1199, 0.9928, 0.0000)
put MyVec.magnitude
-- 1.0000

// Javascript
MyVec = vector(-209.9019, 1737.5126, 0.0000);
MyVec.normalize();
trace(MyVec);
// vector(-0.1199, 0.9928, 0.0000)
trace(MyVec.magnitude);
// 1.0000
```

Die folgende Anweisung zeigt den Wert des Vektors `ThisVector` vor und nach der Normierung:

```
-- Lingo
ThisVector = vector(-50.0000, 0.0000, 0.0000)
normalize(ThisVector)
put ThisVector
-- vector(-1.0000, 0.0000, 0.0000)

// Javascript
ThisVector = vector(-50.0000, 0.0000, 0.0000);
normalize(ThisVector);
trace(ThisVector);
// vector(-1.0000, 0.0000, 0.0000)
```

Siehe auch
`getNormalized`, `randomVector()`, `magnitude`

nothing

Syntax
`nothing`

Beschreibung
Dieser Befehl führt keine Aktion aus. Er eignet sich dazu, die Logik einer `if...then`-Anweisung noch offensichtlicher zu machen. Eine verschachtelte `if...then...else`-Anweisung, die keinen ausdrücklichen Befehl für die Klausel `else` enthält, erfordert unter Umständen `else nothing`, damit Lingo die `else`-Klausel nicht als Teil der vorhergehenden `if`-Klausel interpretiert.

Parameter
Keiner

Beispiel

Die verschachtelte if...then...else-Anweisung in der folgenden Prozedur verwendet den Befehl nothing, um die else-Klausel der Anweisung zu erfüllen:

```
-- Lingo syntax
on mouseDown
    if the clickOn = 1 then
        if sprite(1).moveableSprite = TRUE then member("Notice").text = "Drag the ball"
        else nothing
    else member("Notice").text = "Click again"
    end if
end

// JavaScript syntax
function mouseDown() {
    if (_mouse.clickOn == 1) {
        if (sprite(1).moveableSprite) {
            member("Notice").text = "Drag the ball";
        } else {
            // do nothing
        }
    } else {
        member("Notice").text = "Click again";
    }
}
```

Die folgende Prozedur weist den Film an, keine Aktion einzuleiten, solange die Maustaste gedrückt ist:

```
-- Lingo syntax
on mouseDown
    repeat while the stillDown
        nothing
    end repeat
end mouseDown

// JavaScript syntax
function mouseDown() {
    do {
        // do nothing
    } while _mouse.stillDown;
}
```

Siehe auch

if

nudge()

Syntax

```
-- Lingo syntax
spriteObjRef.nudge(#direction)

// JavaScript syntax
spriteObjRef.nudge(#direction);
```

Beschreibung
Dieser QuickTime VR-Befehl verschiebt die Ansichtsperspektive des angegebenen QuickTime VR-Sprites in eine angegebene Richtung.

Wenn eine Verschiebung nach rechts erfolgt, wird die Sprite-Grafik nach links verschoben. Der Befehl `nudge` hat keinen Rückgabewert.

Parameter
direction Erforderlich. Gibt die Richtung für die Verschiebung der Ansichtsperspektive an. Folgende Werte sind gültig:

- `#down`
- `#downLeft`
- `#downRight`
- `#left`
- `#right`
- `#up`
- `#upLeft`
- `#upRight`

Beispiel
Die folgende Prozedur bewirkt, dass sich die Perspektive des QTVR-Sprites nach links verschiebt, solange die Maus auf dem Sprite gedrückt gehalten wird:

```
-- Lingo syntax
on mouseDown me
    repeat while the stillDown
        sprite(1).nudge(#left)
    end repeat
end

// JavaScript syntax
function mouseDown() {
    do {
        sprite(1).nudge(symbol("left"));
    } while _mouse.stillDown;
}
```

numColumns()

Syntax
```
<matrixObject>.numColumns
```

Beschreibung
Eigenschaft `Matrix` zum Abrufen der Anzahl der Spalten in der Matrix. `numColumns` kann abgerufen, jedoch nicht geändert werden.

Beispiel

Der folgende Codeauszug erstellt eine Matrix und gibt deren Anzahl an Zeilen `numRows` und Spalten `numColumns` aus.

```
--Lingo
mat1 = newMatrix(2, 3, [1,2,3,4,5,6])
put(mat1.numRows)
put(mat1.numColumns)
-- 2
-- 3
//Java Script
mat1 = newMatrix(2, 3, list(1,2,3,4,5,6));
put(mat1.numRows);
put(mat1.numColumns);
// 2
// 3
```

Siehe auch

`getVal()`, `setVal()`, `numRows()`, `matrixAddition()`, `matrixMultiply()`, `matrixMultiplyScalar()`, `matrixTranspose()`, `newMatrix()`

numRows()

Syntax

```
<matrixObject>.numRows
```

Beschreibung

Eigenschaft `Matrix` zum Abrufen der Anzahl der Zeilen in der Matrix. `numRows` kann abgerufen, jedoch nicht geändert werden.

Beispiel

Der folgende Codeauszug erstellt eine Matrix und gibt deren Anzahl an Zeilen `numRows` und Spalten `numColumns` aus.

```
-- Lingo
mat1 = newMatrix(2, 3, [1,2,3,4,5,6])
put(mat1.numRows)
put(mat1.numColumns)
-- 2
-- 3

// Javascript
mat1 = newMatrix(2, 3, list(1,2,3,4,5,6));
put(mat1.numRows);
put(mat1.numColumns);
// 2
// 3
```

Siehe auch

`getVal()`, `setVal()`, `numColumns()`, `matrixAddition()`, `matrixMultiply()`, `matrixMultiplyScalar()`, `matrixTranspose()`, `newMatrix()`

numToChar()

Syntax

```
numToChar(integerExpression)
```

Beschreibung

Diese Funktion zeigt eine Zeichenfolge an, die das einzelne Zeichen enthält, dessen ASCII-Codewert dem Wert eines angegebenen Ausdrucks entspricht. Mithilfe dieser Funktion können Sie Daten aus externen Quellen interpretieren, die nicht in Form von Zeichen, sondern in Form von Zahlen vorliegen.

Auf allen Computern sind standardmäßig alle ASCII-Codewerte bis 127 vorhanden. Codewerte von 128 aufwärts können sich bei verschiedenen Computern auf verschiedene Zeichen beziehen.

Parameter

integerExpression Erforderlich. Gibt den ASCII-Codewert an, dessen entsprechendes Zeichen zurückgegeben wird.

Beispiel

Die folgende Prozedur entfernt aus einem beliebigen String alle nicht-alphabetischen Zeichen und gibt nur Großbuchstaben zurück:

```
-- Lingo syntax
on ForceUppercase input
    output = EMPTY
    num = length(input)
    repeat with i = 1 to num
        theASCII = charToNum(input.char[i])
        if theASCII = min(max(96, theASCII), 123) then
            theASCII = theASCII - 32
            if theASCII = min(max(63, theASCII), 91) then
                put numToChar(theASCII) after output
            end if
        end if
    end repeat
    return output
end

// JavaScript syntax
function ForceUpperCase(input) {
    output = "";
    num = input.length;
    for (i=1;i<=num;i++) {
        theASCII = charToNum (input.getProp("char",i);
        if (theASCII == list(list(96, theASCII).max(), 123).min()) {
            theASCII = theASCII - 32;
            if (theASCII == list(list(63, theASCII).max(), 91).min()) {
                output = output + numToChar (theASCII);
            }
        }
    }
    return output;
}
```

Siehe auch

charToNum()

objectP()

Syntax

```
objectP(expression)
```

Beschreibung

Diese Funktion gibt an, ob ein angegebener Ausdruck ein durch ein Parent-Skript, Xtra oder Fenster erstelltes Objekt ist (TRUE) oder nicht (FALSE).

Das *P* in objectP bedeutet *predicate* (Prädikat).

Es ist zu empfehlen, mit objectP zu ermitteln, welche Objekte bereits in Benutzung sind, wenn Sie Objekte mithilfe von Parent-Skripts oder Xtra-Instanzen erstellen.

Ein Beispiel für objectP() in einem fertigen Film finden Sie im Film „Read and Write Text“ im Ordner „Learning\\Lingo_“ im Director-Anwendungsordner.

Parameter

expression Erforderlich. Gibt den zu testenden Ausdruck an.

Beispiel

Die folgende Anweisung prüft, ob der globalen Variablen gDataBase ein neues Objekt zugeordnet ist; wenn nicht, wird ein Objekt zugeordnet. Diese Prüfung wird häufig dazu verwendet, um am Anfang eines Films oder Abschnitts Initialisierungen auszuführen, die nicht wiederholt werden sollen.

```
-- Lingo syntax
if objectP(gDataBase) then
    nothing
else
    gDataBase = script("Database Controller").new()
end if

// JavaScript syntax
if (objectP(gDataBase)) {
    // do nothing
} else {
    gDataBase = script("Database Controller").new();
}
```

Siehe auch

floatP(), ilk(), integerP(), stringP(), symbolP()

offset() (Zeichenfolgenfunktion)

Syntax

```
offset(stringExpression1, stringExpression2)
```

Beschreibung

Diese Funktion gibt eine Ganzzahl zurück, die die Position des ersten Zeichens einer Zeichenfolge innerhalb einer anderen Zeichenfolge angibt. Diese Funktion gibt 0 zurück, wenn die erste Zeichenfolge nicht in der zweiten zu finden ist. Lingo zählt Leerzeichen in beiden Strings als Zeichen.

Auf dem Mac spielt es bei diesem Stringvergleich keine Rolle, ob die Wörter groß oder klein geschrieben sind oder diakritische Zeichen enthalten. Auf dem Mac interpretiert Lingo z.*B.* a *und* Å als dasselbe Zeichen.

Parameter

stringExpression1 Erforderlich. Gibt die in *stringExpression2* zu suchende Teilzeichenfolge an.

stringExpression2 Erforderlich. Gibt die Zeichenfolge an, die die Teilzeichenfolge *stringExpression1* enthält.

Beispiel

Die folgende Anweisung zeigt im Nachrichtenfenster die Anfangsposition des Strings „media“ im String „kleptomedia“ an:

```
put offset("media","kleptomedia")
```

Das Ergebnis ist 7.

Die folgende Anweisung zeigt im Nachrichtenfenster die Anfangsposition des Strings „Micro“ im String „Adobe“ an:

```
put offset("Micro", "Adobe")
```

Das Ergebnis ist 0, da „Adobe“ die Zeichenfolge „Micro“ nicht enthält.

Die folgende Prozedur sucht alle Instanzen der durch `stringToFind` dargestellten Zeichenfolge in der durch input dargestellten Zeichenfolge und ersetzt sie durch die durch `stringToInsert` dargestellten Zeichenfolge:

```
-- Lingo syntax
on SearchAndReplace input, stringToFind, stringToInsert
    output = ""
    findLen = stringToFind.length - 1
    repeat while input contains stringToFind
        currOffset = offset(stringToFind, input)
        output = output & input.char [1..currOffset]
        delete the last char of output
        output = output & stringToInsert
        delete input.char [1.. (currOffset + findLen)]
    end repeat
    set output = output & input
    return output
end

// JavaScript syntax
function SearchAndReplace(input, stringToFind, stringToInsert) {
    output = "";
    findLen = stringToFind.length - 1;
    do {
        currOffset = offset(stringToFind, input);
        output = output + input.char[0..currOffset];
        output = output.substr(0,output.length-2);
        output = output + stringToInsert;
        input = input.substr(currOffset+findLen,input.length);
    } while (input.indexOf(stringToFind) >= 0);
    output = output + input;
    return output;
}
```

Siehe auch

chars(), length(), contains, starts

offset() (Rechteckfunktion)

Syntax

```
rectangle.offset(horizontalChange, verticalChange)
offset (rectangle, horizontalChange, verticalChange)
```

Beschreibung

Diese Funktion ergibt ein Rechteck, das sich in einem bestimmten Abstand zu dem in *rectangle* angegebenen Rechteck befindet.

Parameter

horizontalChange Erforderlich. Gibt den horizontalen Versatz in Pixel an. Wenn *horizontalChange* größer als 0 ist, erfolgt auf der Bühne eine Versetzung nach rechts. Wenn *horizontalChange* dagegen kleiner als 0 ist, erfolgt auf der Bühne eine Versetzung nach links.

verticalChange Erforderlich. Gibt den vertikalen Versatz in Pixel an. Wenn *verticalChange* größer als 0 ist, erfolgt die Versetzung zur Oberseite der Bühne hin. Ist *verticalChange* dagegen kleiner als 0, erfolgt die Versetzung zur Unterseite der Bühne hin.

Beispiel

Die folgende Prozedur verschiebt Sprite 1 um fünf Pixel nach rechts und fünf Pixel nach unten:

```
-- Lingo syntax
on diagonalMove
    newRect=sprite(1).rect.offset(5, 5)
    sprite(1).rect=newRect
end

// JavaScript syntax
function diagonalMove() {
    newRect = sprite(1).rect.offset(5,5);
    sprite(1).rect = newRect;
}
```

open() (Player)

Syntax

```
-- Lingo syntax
_player.open({stringDocPath,} stringAppPath)

// JavaScript syntax
_player.open({stringDocPath,} stringAppPath);
```

Beschreibung

Diese Player-Methode öffnet eine angegebene Anwendung und öffnet dabei optional noch eine angegebene Datei.

Wenn sich *stringDocPath* oder *stringAppPath* in einem anderen Ordner als der aktuelle Film befinden, müssen Sie den vollständigen Pfadnamen für die Datei(en) angeben.

Der Computer muss über ausreichend Arbeitsspeicher verfügen, damit Director und andere Anwendungen gleichzeitig ausgeführt werden können.

Dies ist eine ganz einfache Methode, mit der Sie eine Anwendung oder ein Dokument in einer Anwendung öffnen können. Falls Sie mehr Steuerfunktionalität benötigen, sollten Sie die verfügbaren Optionen in Xtras von Drittanbietern in Betracht ziehen.

Parameter

stringDocPath Optional. Eine Zeichenfolge, die das Dokument angibt, das geöffnet werden soll, wenn die über *stringAppPath* angegebene Anwendung geöffnet wird.

stringAppPath Erforderlich. Eine Zeichenfolge, die den Pfad zu der zu öffnenden Anwendung angibt.

Beispiel

Die folgende Anweisung startet die Anwendung `TextEdit`, die sich im Ordner „Applications“ auf dem Festplattenlaufwerk „HD“ (Mac) befindet, und öffnet dann das Dokument „Storyboards“:

```
-- Lingo syntax
_player.open("Storyboards", "HD:Applications:TextEdit")

// JavaScript syntax
_player.open("Storyboards", "HD:Applications:TextEdit");
```

Siehe auch

`Player`

open() (Fenster)

Syntax

```
-- Lingo syntax
windowObjRef.open()

// JavaScript syntax
windowObjRef.open();
```

Beschreibung

Diese Fenstermethode öffnet ein Fenster und positioniert es vor allen anderen Fenstern.

Wenn dem Fenster, für das `open()` aufgerufen wird, kein Film zugeordnet ist, wird das Dialogfeld zum Öffnen einer Datei angezeigt.

Wenn der Verweis auf das Fensterobjekt *windowObjRef* durch den Namen eines Films ersetzt wird, verwendet das Fenster den Dateinamen als Fensternamen. Dem Fenster muss dann aber mithilfe der Fenstereigenschaft `fileName` ein Film zugewiesen werden.

Wenn der Verweis auf das Fensterobjekt *windowObjRef* durch einen Fensternamen ersetzt wird, verwendet das Fenster diesen Namen. Dem Fenster muss dann aber mithilfe der Fenstereigenschaft `fileName` ein Film zugewiesen werden.

Um ein Fenster zu öffnen, das einen Film unter einer URL verwendet, laden Sie die Filmdatei zuerst mithilfe von `downloadNetThing()` auf einen lokalen Datenträger herunter und verwenden sie dann von dort aus. Diese Vorgehensweise minimiert Wartezeiten beim Download des Films.

Bei Verwendung eines lokalen Films laden Sie mithilfe von `preloadMovie()` zumindest das erste Bild des Films voraus, bevor `open()` aufgerufen wird. Dieses Verfahren verringert die Möglichkeit von Verzögerungen beim Laden des Films.

Das Öffnen eines Films in einem Fenster wird derzeit beim Abspielen in einem Browser nicht unterstützt.

Parameter

Keiner

Beispiel

Die folgende Anweisung öffnet das Fenster „Steuerpult“ und bringt es in den Vordergrund.

```
-- Lingo syntax
window("Control Panel").open()

// JavaScript syntax
window("Control Panel").open();
```

Siehe auch

`close()`, `downloadNetThing`, `fileName (Fenster)`, `preLoadMovie()`, `Window`

openFile()

Syntax

```
-- Lingo syntax
fileioObjRef.openFile(stringFileName, intMode)

// JavaScript syntax
fileioObjRef.openFile(stringFileName, intMode)
```

Beschreibung

Diese Datei-E/A-Methode (Fileio) öffnet eine angegebene Datei in einem vorgegebenen Modus.

Parameter

stringFileName Erforderlich. Eine Zeichenfolge, die den vollständigen Pfad und Namen der zu öffnenden Datei angibt.

intMode Erforderlich. Eine Ganzzahl, die den Modus der Datei angibt. Folgende Werte sind gültig:

- 0: Schreiben/Lesen
- 1: Nur Lesen
- 2: Schreibbar

Beispiel

Die folgende Anweisung öffnet eine Datei „c:\xtra.txt“ mit Lese-/Schreibberechtigung.

```
-- Lingo
objFileio = new xtra("fileio")
objFileio.openFile("c:\xtra.txt",0)

// JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.openFile("c:\xtra.txt",0);
```

Siehe auch

`Fileio`

openXlib

Syntax

```
openXlib whichFile
```

Beschreibung

Dieser Befehl öffnet eine angegebene Xlibrary-Datei.

Es ist zu empfehlen, geöffnete Dateien wieder zu schließen, sobald sie nicht mehr benötigt werden. Der Befehl `openXlib` wirkt sich nicht auf bereits geöffnete Dateien aus.

Der Befehl `openXlib` unterstützt keine URLs als Dateibezüge.

Xlibrary-Dateien enthalten Xtra-Erweiterungen. Im Gegensatz zu `openResFile` werden diese Xtra-Erweiterungen durch `openXlib` Director zur Verfügung gestellt.

Beim Öffnen einer Xtra-Skripterweiterung mithilfe von `openXlib` muss diese mit `closeXlib` wieder geschlossen werden, wenn sie nicht mehr von Director verwendet wird.

Unter Windows ist die Erweiterung ".dll" optional.

Hinweis: *Dieser Befehl wird im Shockwave Player nicht unterstützt.*

Parameter
whichFile Erforderlich. Gibt die zu öffnende Xlibrary-Datei an. Falls sich die Datei nicht im selben Ordner wie der aktuelle Film befindet, muss *whichFile* den entsprechenden Pfadnamen enthalten.

Beispiel
Die folgende Anweisung öffnet die Xlibrary-Datei "Video Disc Xlibrary":

```
openXlib "Video Disc Xlibrary"
```

Die folgende Anweisung öffnet die Xlibrary-Datei „Transporter-Xtras“, die sich in einem anderen Ordner als der aktuelle Film befindet:

```
openXlib "My Drive:New Stuff:Transporter Xtras"
```

Siehe auch
`closeXlib`, `Interface()`

param()

Syntax
```
param(parameterPosition)
```

Beschreibung
Diese Funktion gibt den Wert eines Parameters an, der an eine Prozedur übergeben wurde.

Diese Funktion kann dazu benutzt werden, den Typ eines bestimmten Parameters zu bestimmen, um auf diese Weise Prozedurfehler zu vermeiden.

Parameter
parameterPosition Erforderlich. Gibt die Position des Parameters in den an die Prozedur übergebenen Argumenten an.

Beispiel
Die folgende Prozedur akzeptiert eine beliebige Anzahl von Argumenten und addiert alle als Parameter übergebenen Werte, um dann die Summe zurückzugeben:

```
--Lingo syntax
on AddNumbers
    sum = 0
    repeat with currentParamNum = 1 to the paramCount
        sum = sum + param(currentParamNum)
    end repeat
    return sum
end

// JavaScript syntax
function AddNumbers() {
    sum = 0;
    for (currentParamNum=0;currentParamNum<arguments.length;currentParamNum++){
        sum = sum + arguments[currentParamNum];
    }
    return sum;
}
```

Sie verwenden diese Funktion, indem Sie die zu addierenden Werte wie folgt eingeben:

```
put AddNumbers(3, 4, 5, 6)
-- 18
put AddNumbers(5, 5)
-- 10
```

Siehe auch

getAt, paramCount(), return (Schlüsselwort)

paramCount()

Syntax

```
the paramCount
```

Beschreibung

Diese Funktion gibt die Anzahl der Parameter an, die an die aktuelle Prozedur übergeben wurden.

Parameter

Keiner

Beispiel

Die folgende Anweisung setzt die Variable counter auf die Anzahl der Parameter, die an die aktuelle Prozedur gesendet wurden:

```
set counter = the paramCount
```

parseByteArray

Syntax

```
xmlParser.parseByteArray(byteArrayXML)
```

Beschreibung
Die Kodierung einer XML-Datei wird anhand des XML-Deklarationstags korrekt erkannt.

Parameter

Parameter	Beschreibung	Erforderlich/Optional
byteArrayXML	Das Byte-Array, das analysiert werden soll.	Erforderlich

Beispiele

```
--Lingo syntax
gXMLParser = new xtra("xmlparser")
gFileIO = new xtra("fileIO")
gFileIO.openFile("test.xml",1)
gFileIO.setPosition(0)
bArray = gFileIO.readByteArray(gFileIO.getLength())
gXMLParser.parseByteArray(bArray)

//JavaScript syntax
gXMLParser = new xtra("xmlparser");
gFileIO = new xtra("fileIO");
gFileIO.openFile("test.xml",1);
gFileIO.setPosition(0);
bArray = gFileIO.readByteArray(gFileIO.getLength());
gXMLParser.parseByteArray(bArray);
```

parseString()

Syntax

```
parserObject.parseString(stringToParse)
```

Beschreibung
Diese Funktion dient zum Parsen eines XML-Dokuments, das dem Director-Film bereits vollständig zur Verfügung steht. Der erste Parameter ist die Variable, die das Parserobjekt enthält. Der Rückgabewert ist `<VOID>`, wenn die Operation erfolgreich ist, bzw. eine Fehlercodenummer, wenn die Operation fehlschlägt. Fehler sind in der Regel auf Probleme mit der XML-Syntax oder -Struktur zurückzuführen. Nach Abschluss der Operation enthält das Parser-Objekt die geparsten XML-Daten.

Um ein XML-Dokument zu parsen, das unter einer bestimmten URL zu finden ist, verwenden Sie `parseURL()`.

Parameter
stringToParse Erforderlich. Gibt die Zeichenfolge der zu parsenden XML-Daten an.

Beispiel
Die folgende Anweisung parst die XML-Daten im Textdarsteller `XMLtext`. Nach Abschluss der Operation enthält die Variable `gParserObject` die geparsten XML-Daten.

```
-- Lingo
errorCode = gParserObject.parseString(member("XMLtext").text)

// Javascript
errorCode = gParserObject.parseString(member("XMLtext").text);
```

Siehe auch

getError() (XML), parseURL()

parseString (XML Xtra)

Syntax

```
xmlParser.parseString(strXML)
```

Beschreibung

Diese XML Xtra-Methode stellt UTF-8 als Standardtyp für den Eingabestring ein.

Parameter

Parameter	Beschreibung	Erforderlich/Optional
strXML	String, der analysiert werden soll.	Erforderlich

Beispiele

```
--Lingo syntax
gXMLParser = new xtra("xmlparser")
gFileIO = new xtra("fileIO")
gFileIO.openFile("test.xml",1)
gFileIO.setPosition(0)
b = gFileIO.readFile()
gXMLParser.parseString(b)

//JavaScript syntax
gXMLParser = new xtra("xmlparser");
gFileIO = new xtra("fileIO");
gFileIO.openFile("test.xml",1);
gFileIO.setPosition(0);
b = gFileIO.readFile();
gXMLParser.parseString(b);
```

parseURL()

Syntax

```
parserObject.parseURL(URLstring {,#handlerToCallOnCompletion} {, objectContainingHandler})
```

Beschreibung

Diese Funktion parst ein XML-Dokument, die sich an einer externen Internetadresse befindet. Der erste Parameter ist das Parserobjekt, das eine Instanz des XML-Parser-Xtras enthält.

Diese Funktion sendet sofort einen Rückgabewert; es kann also vorkommen, dass noch nicht die gesamte URL geparst ist. Um ermitteln zu können, wann die Parsing-Operation abgeschlossen ist, müssen Sie die Funktion `doneParsing()` zusammen mit `parseURL()` verwenden.

Da diese Operation asynchron – und somit potenziell zeitaufwendig – ist, können Sie mithilfe optionaler Parameter nach Abschluss der Operation eine spezifische Routine aufrufen.

Der Rückgabewert ist „VOID“, wenn die Operation erfolgreich ist, bzw. eine Fehlercodenummer, wenn die Operation fehlschlägt.

Zum Parsen eines lokalen XML-Dokuments wird `parseString()` verwendet.

Parameter

URLstring Erforderlich. Gibt die eigentliche URL an, unter der die XML-Daten zu finden sind.

handlerToCallOnCompletion Optional. Gibt den Namen der Prozedur an, die im Anschluss an das vollständige Parsen der URL ausgeführt werden soll.

objectContainingHandler Optional. Gibt den Namen des Skriptobjekts an, das die Prozedur *handlerToCallOnCompletion* enthält. Wird der Parameter ausgelassen, wird standardmäßig angenommen, dass es sich um eine Filmprozedur handelt.

Beispiel

Die folgende Anweisung parst die Datei "beispiel.xml" an der URL "meinefirma.com". Verwenden Sie `doneParsing()`, um zu ermitteln, wann die Parsing-Operation abgeschlossen ist.

```
--Lingo syntax
errorCode = gParserObject.parseURL("http://www.MyCompany.com/sample.xml")

// JavaScript syntax
errorCode = _global.gParserObject.parseURL("http://- www.MyCompany.com/sample.xml");
```

Hinweis: *Das folgende Beispiel setzt voraus, dass eine Instanz des Xtra bereits erstellt und ein Verweis darauf in der globalen Variablen `gParserObject` gespeichert wurde.*

Die folgende Lingo-Anweisung parst die Datei "beispiel.xml" und ruft die Prozedur on `ParseDone` auf. Da in der Funktion `doneParsing()` kein Skriptobjekt angegeben ist, wird davon ausgegangen, dass die Prozedur on `parseDone` sich in einem Filmskript befindet.

```
errorCode = gParserObject.parseURL("http://www.MyCompany.com/sample.xml", #parseDone)
```

Das Filmskript enthält die Prozedur on `parseDone`:

```
on parseDone
    global gParserObject
    if voidP(gParserObject.getError()) then
        put "Successful parse"
    else
        put "Parse error:"
        put " " & gParserObject.getError()
    end if
end
```

Die folgende JavaScript-Syntax parst die Datei „beispiel.xml“ und ruft die Funktion `parseDone` auf. Da in der Funktion `doneParsing()` kein Skriptobjekt angegeben ist, wird davon ausgegangen, dass sich die parseDone-Funktion in einem Filmskript befindet.

```
errorCode = _global.gParserObject.parseURL("http://- www.MyCompany.com/sample.xml",
symbol("parseDone"));
```

Hinweis: *Das folgende Beispiel setzt voraus, dass eine Instanz des Xtra bereits erstellt und ein Verweis darauf in der globalen Variablen* `gParserObject` *gespeichert wurde.*

Das Filmskript enthält die Prozedur on `parseDone`:

```
// JavaScript syntax
function parseDone () {
    if (_global.gParserObject.getError() == undefined) {
        trace("successful parse");
    } else {
        trace("Parse error:");
        trace(" " + _global.gParserObject.getError());
    }
}
```

Die folgende Lingo-Anweisung parst das Dokument „beispiel.xml“ an der URL „meinefirma.com“ und ruft die Prozedur on `ParseDone` im Skriptobjekt `testObject` auf, bei dem es sich um ein Child des Parent-Skripts `TestScript` handelt:

```
parserObject = new(xtra "XMLParser")
testObject = new(script "TestScript", parserObject)
errorCode = gParserObject.parseURL("http://www.MyCompany.com/sample.xml", #parseDone,
testObject)
```

Nachfolgend das Parent-Skript `TestScript`:

```
property myParserObject

on new me, parserObject
    myParserObject = parserObject
end

on parseDone me
    if voidP(myParserObject.getError()) then
        put "Successful parse"
    else
        put "Parse error:"
        put " " & myParserObject.getError()
    end if
end
```

Die folgende JavaScript-Syntax parst das Dokument „beispiel.xml“ unter „meinefirma.com“ und ruft die Funktion `parseDone` im Objekt `testObject` auf, bei dem es sich um eine Instanz der definierten `TestScript`-Klasse handelt:

```
parserObject = new xtra("XMLParser");
testObject = new TestScript(parserObject);
errorCode = parserObject .parseURL("http://www.MyCompany.com/sam- ple.xml",
symbol("parseDone"), testObject)
```

Nachfolgend die Definition der `TestScript`-Klasse:

```
TestScript = function (aParser) {
    this.myParserObject = aParser;
}
TestScript.prototype.parseDone = function () {
    if (this.myParserObject.getError() == undefined) {
        trace("successful parse");
    } else {
        trace("Parse error:");
        trace(" " + this.myParserObject.getError());
    }
}
```

Siehe auch

getError() (XML), parseString()

parseURL (XML Xtra)

Syntax

```
xmlParser.parseURL(strXML)
```

Beschreibung

Diese XML Xtra-Methode erkennt die Kodierung einer XML-Datei anhand des XML-Deklarationstags. Diese Methode ähnelt `xmlParser.parseByteArray(byteArrayXML)`.

Parameter

Parameter	Beschreibung	Erforderlich/Optional
strXML	String, der analysiert werden soll.	Erforderlich

Beispiele

```
--Lingo syntax
gXMLParser = new xtra("xmlparser")
gFileIO = new xtra("fileIO")
gXMLParser.parseURL("http://www.bytearray.com/byte.xml")

//JavaScript syntax
gXMLParser = new xtra("xmlparser");
gFileIO = new xtra("fileIO");
gXMLParser.parseURL("http://www.bytearray.com/byte.xml");
```

pass

Syntax

```
pass
```

Beschreibung

Dieser Befehl übergibt der nächsten Position in der Nachrichtenhierarchie eine Ereignisnachricht und ermöglicht die Ausführung von mehreren Prozeduren für das jeweilige Ereignis.

Der Befehl `pass` geht bei der Ausführung sofort zur nächsten Position über. Alle Lingo-Anweisungen, die in der Prozedur auf den Befehl `pass` folgen, werden dann nicht ausgeführt.

Eine Ereignisnachricht hält standardmäßig an der ersten Position an, die eine Prozedur für das Ereignis enthält, in der Regel auf der Sprite-Ebene.

Befindet sich der Befehl `pass` in einer Prozedur, wird das Ereignis an andere Objekte in der Hierarchie übergeben, obwohl die Prozedur das Ereignis normalerweise abfangen würde.

Parameter

Keiner

Beispiel

Die folgende Prozedur prüft, welche Tasten gedrückt werden, und gibt diese Informationen dann an das bearbeitbare Text-Sprite weiter, sofern es sich um gültige Zeichen handelt:

```
-- Lingo syntax
on keyDown me
    legalCharacters = "1234567890"
    if legalCharacters contains the key then
        pass
    else
        beep
    end if
end

// JavaScript syntax
function keyDown() {
    legalCharacters = "1234567890";
    if (legalCharacters.indexOf(_key.key) >= 0) {
        pass();
    } else {
        _sound.beep();
    }
}
```

Siehe auch

`stopEvent()`

pasteClipBoardInto()

Syntax

```
-- Lingo syntax
memberObjRef.pasteClipBoardInto()

// JavaScript syntax
memberObjRef.pasteClipBoardInto();
```

Beschreibung
Diese Darstellermethode fügt den Inhalt der Zwischenablage in einen angegebenen Darsteller ein und löscht den bisherigen Darsteller.

Sie können beliebige Elemente einfügen, sofern sie sich in einem Format befinden, das Director als Darsteller verwenden kann.

Wenn Sie Strings aus einer anderen Anwendung kopieren, geht die Formatierung des Strings verloren.

Diese Methode eignet sich insbesondere dazu, Objekte aus anderen Filmen und anderen Anwendungen ins Besetzungsfenster zu kopieren. Da kopierte Darsteller im RAM-Speicher abgelegt werden müssen, sollten Sie diesen Befehl möglichst nicht während des Abspielens bei niedriger Speicherkapazität verwenden.

Wenn Sie diese Methode im Shockwave Player oder in der Erstellungsumgebung bzw. in Projektoren verwenden, in denen die Eigenschaft `safePlayer` auf `TRUE` gesetzt ist, erscheint ein Warndialog, damit der Benutzer den Vorgang abbrechen kann.

Parameter
Keiner

Beispiel
Die folgende Anweisung fügt den Inhalt der Zwischenablage in den Bitmapdarsteller "Schrein" ein:

```
-- Lingo syntax
member("shrine").pasteClipBoardInto()

// JavaScript syntax
member("shrine").pasteClipBoardInto();
```

Siehe auch
`Member`, `safePlayer`

pause() (DVD)

Syntax

```
-- Lingo syntax
dvdObjRef.pause()

// JavaScript syntax
dvdObjRef.pause();
```

Beschreibung
Diese DVD-Methode unterbricht die Wiedergabe.

Parameter
Keiner

Beispiel
Die folgende Anweisung unterbricht die Wiedergabe:

```
-- Lingo syntax
member(1).pause()

// JavaScript syntax
member(1).pause();
```

Siehe auch

DVD

pause() (Mixer)

Syntax

```
Mixer.pause([soundobjectlist])
```

Beschreibung

Diese Mixermethode pausiert den Mixer, wenn kein Soundobjekt übergeben wird, oder pausiert die Liste der Soundobjekte, die in der Liste vorhanden sind.

Wirkung auf den Mixer	Funktion
Pause()	Pausiert den Mixer.
pause([so1, so2])	Pausiert die Soundobjekte des Mixers mit den Referenzen `so1 and so2`.
pause(["so1", "so2"])	Pausiert die Soundobjekte des Mixers mir den Namen `so1` und `so2`.

Beispiele

```
-- Lingo syntax
on mouseUp me
     mixer1.pause() --Pauses mixer1.
end

// JavaScript syntax
function mouseup()
{
mixer1.pause(); //Pauses mixer1.

}
```

Siehe auch

play() (Mixer), Mixer

pause (MP4Media/FLV)

Syntax

```
sprite(1).pause()
member(1).pause()
```

Beschreibung

Diese MP4Media/FLV-Sprite-Methode pausiert die Wiedergabe des Media-Streams. Der Wert `mediaStatus` ändert sich in `#paused`.

Der Aufruf dieser Methode während der Wiedergabe des MP4Media/FLV-Streams ändert nicht die Eigenschaft `currentTime`. Daher wird die Wiedergabe nach einem `Play`-Befehl ohne erneutes Puffern fortgesetzt.

Beispiele

Im folgenden Beispiel wird Sprite 2 des MP4Media/FLV-Darstellers pausiert.

```
-- Lingo syntax
sprite(2).pause()
member("MP4Media/FLV").pause()

// JavaScript syntax
sprite(2).pause();
member("MP4Media/FLV").pause();
```

Siehe auch

`play() (MP4Media/FLV)`, `isPlayable (MP4Media/FLV)`, `Mixer`

pause() (3D)

Syntax

```
member(whichCastmember).model(whichModel).bonesPlayer.pause()
member(whichCastmember).model(whichModel).keyframePlayer.pause()
```

Beschreibung

Dieser 3D-Befehl für die Modifizierer `#keyframePlayer` und `#bonesPlayer` hält die Bewegung an, die gegenwärtig vom Modell ausgeführt wird. Mit dem Befehl `play()` können Sie die Bewegung fortsetzen.

Wenn die Bewegung eines Modells mit diesem Befehl angehalten wurde, wird die Modelleigenschaft `bonesPlayer.playing` auf `FALSE` gesetzt.

Parameter

Keiner

Beispiel

Die folgende Anweisung hält die aktuelle Animation des Modells "Ameise3" an:

```
member("PicnicScene").model("Ant3").bonesplayer.pause()
```

Siehe auch

`play() (3D)`, `playing (3D)`, `playlist`

pause() (RealMedia, SWA, Windows Media)

Syntax

```
-- Lingo syntax
memberOrSpriteObjRef.pause()

// JavaScript syntax
memberOrSpriteObjRef.pause();
```

Beschreibung

Diese Methode für RealMedia- und Windows Media-Sprites und -Darsteller unterbricht die Wiedergabe des Medienstreams.

Der Wert von `mediaStatus` ändert sich in `#paused`.

Bei Aufruf dieser Methode während der Wiedergabe des RealMedia- oder Windows Media-Streams wird weder die Eigenschaft `currentTime` geändert noch der Medienpuffer geleert. Auf diese Weise kann die Wiedergabe mit späteren `play`-Befehlen fortgesetzt werden, ohne dass der Stream neu gepuffert werden muss.

Parameter

Keiner

Beispiel

In den folgenden Beispielen wird die Wiedergabe von Sprite 2 bzw. des Darstellers "Real" unterbrochen:

```
-- Lingo syntax
sprite(2).pause()
member("Real").pause()

// JavaScript syntax
sprite(2).pause();
member("Real").pause();
```

Siehe auch

`mediaStatus (RealMedia, Windows Media)`, `play() (RealMedia, SWA, Windows Media)`, `seek()`, `stop() (RealMedia, SWA, Windows Media)`

pause() (Soundkanal)

Syntax

```
-- Lingo syntax
soundChannelObjRef.pause()

// JavaScript syntax
soundChannelObjRef.pause();
```

Beschreibung

Diese Soundkanalmethode unterbricht die Wiedergabe des aktuellen Sounds in einem Soundkanal.

Die Wiedergabe wird durch eine spätere `play()`-Methode fortgesetzt.

Parameter
Keiner

Beispiel
Die folgende Anweisung unterbricht die Wiedergabe des Sounddarstellers, der in Soundkanal 1 abgespielt wird:

```
-- Lingo syntax
sound(1).pause()

// JavaScript syntax
sound(1).pause();
```

Siehe auch
breakLoop(), play() (Soundkanal), playNext() (Soundkanal), queue(), rewind() (Soundkanal)SoundChannel, stop() (Soundkanal)

pause (Soundobjekt)

Syntax
```
soundObject.pause()
```

Beschreibung
Diese Soundobjektmethode pausiert die Wiedergabe der verknüpften Audiodatei mit dem Soundobjekt.

Beispiele
```
--Lingo syntax
on mouseUp me
     soundObjRef.pause() --Pauses the sound object associated with soundObjRef.
end


// JavaScript syntax
function mouseUp(){
soundObjRef.pause(); //Pauses the sound object associated with soundObjRef.
}
```

Siehe auch
play (Soundobjekt)

perlinNoise()

Syntax
```
-- Lingo syntax
member(membername).Image.perlinNoise(baseX, baseY, numOctaves, seed, bStitch, bFractalNoise,
[#channelOptions:value, #grayscale:bolean, #offsets:listOfPoints])
// JavaScript syntax
```

Beschreibung

Der Algorithmus zur Erzeugung von Perlin Noise interpoliert und kombiniert einzelne Zufallsrauschenfunktionen (als Oktaven bezeichnet) in einer einzigen Funktion, die ein natürlicher wirkendes weißes Rauschen erzeugt.

Parameter

Eigenschaft	Beschreibung	Wertebereich	Standard
baseX:Zahl	Bestimmt den x-Wert (Größe) der erzeugten Muster.		
baseY:Zahl	Bestimmt den y-Wert (Größe) der erzeugten Muster.		
numOctaves:Zahl	Anzahl der Oktaven oder einzelnen Rauschenfunktionen, die zum Erzeugen dieses Rauschens kombiniert werden sollen. Je mehr Oktaven verwendet werden, desto detaillierter sind die erstellen Grafiken. Diese benötigen dann aber auch mehr Verarbeitungszeit.		
randomSeed:Zahl	Der Ausgangswert für den Zufallsgenerator.		
bStitch:Boole'scher Wert	Wenn dieser Parameter auf „true“ festgelegt ist, versucht die Methode, die Übergangskanten der Grafik zu glätten („stitch“), um übergangslose Texturen für das Kacheln als Bitmapfüllung zu erzeugen.	true/false	
bFractalNoise:Boole'scher Wert	Dieser Parameter bezieht sich auf die Kanten der Verläufe, die von der Methode erzeugt werden. Wenn dieser Parameter auf „true“ festgelegt ist, erzeugt die Methode fraktales Rauschen, das die Kanten des Effekts glättet. Wird der Parameter auf „false“ festgelegt, werden Turbulenzen erzeugt. Eine Grafik mit Turbulenzen beinhaltet sichtbare Diskontinuitäten in Verläufen, wodurch schärfer abgegrenzte visuelle Effekte besser ausgearbeitet werden können, wie z. B. Flammen oder Meereswellen.	true/false	

Eigenschaft	Beschreibung	Wertebereich	Standard
channelOptions : Zahl (optional)	Gibt an, auf welchen Farbkanal (der Bitmap) das Rauschenmuster angewendet wird. Die Zahl kann eine beliebige Kombination aus den vier Farbkanälen RGBA(1, 2, 4 und 8) sein. Der Standardwert ist 7.		0
grayScale:Boole'scher Wert (optional)	Wenn dieser Parameter auf „true“ festgelegt wird, wird der Zufallswert `randomSeed` auf die Bitmappixel angewendet, wodurch jegliche Farbe aus dem Bild „herausgewaschen“ wird. Der Standardwert ist false.	0 oder 1	0
Offsets:Liste oder Punkt (Optional)	Kann eine Liste von Punkten oder ein einzelner Punkt sein, die für jede Oktave den x- und y-Versatzwerten entsprechen. Durch Manipulation der Versatzwerte können die Ebenen der Grafik gleichmäßig gerollt werden. Jeder Punkt in der Versatzliste wirkt sich auf eine spezifische Oktavrauschenfunktion aus. Der Standardwert ist 0. Wenn Sie einen Punkt angeben, werden für alle Oktavrauschenfunktionen dieselben Werte verwendet.		Point (0,0)

Beispiel

In den folgenden Beispielen wird die Wiedergabe von Sprite 2 bzw. des Darstellers "Real" unterbrochen:

```
-- Lingo syntax
myList = [point(0,1), point(5,5)] --List of Points
member("myMember").Image.perlinNoise(300, 300, 2, 2, true, true, [#channelOptions:7,
#grayscale:false, #offsets:myList])
// JavaScript syntax
myList = list(point(0,1), point(5,5)) //List of Points
member("myMember").image.perlinNoise(300, 300, 2, 2, true,
true, propList(symbol("channelOptions"),7, symbol("grayscale"),false,
symbol("offsets"),myList));
```

perpendicularTo

Syntax

```
vector1.perpendicularTo(vector2)
```

Beschreibung

Dieser 3D-Vektorbefehl gibt einen Vektor zurück, der senkrecht zum Originalvektor und zu einem zweiten Vektor verläuft. Dieser Befehl entspricht dem Vektorbefehl `crossProduct`.

Parameter

vector2 Erforderlich. Gibt den zweiten Vektor an.

Beispiel

Im folgenden Beispiel ist `pos1` ein Vektor auf der x-Achse und `pos2` ein Vektor auf der y-Achse. Der von `pos1.perpendicularTo(pos2)` zurückgegebene Wert ist vector(0.0000, 0.0000, 1.00000e4). Die letzten zwei Zeilen in diesem Beispiel zeigen den Vektor, der im 90° Winkel zu `pos1` und`pos2` verläuft.

```
-- Lingo
pos1 = vector(100, 0, 0)
pos2 = vector(0, 100, 0)
put pos1.perpendicularTo(pos2)
-- vector( 0.0000, 0.0000, 1.00000e4 )

// Javascript
pos1 = vector(100, 0, 0);
pos2 = vector(0, 100, 0);
trace(pos1.perpendicularTo(pos2));
// vector( 0.0000, 0.0000, 1.00000e4 )
```

Siehe auch

`crossProduct()`, `cross`

pictureP()

Syntax

```
-- Lingo syntax
pictureP(pictureValue)

// JavaScript syntax
pictureP(pictureValue);
```

Beschreibung

Diese Funktion meldet, ob der Zustand der Darstellereigenschaft `picture` für den angegebenen Darsteller `TRUE` (1) oder `FALSE` (0) ist.

Da `pictureP` nicht direkt prüft, ob ein Bild mit einem Darsteller verbunden ist, müssen Sie dies testen, indem Sie die Eigenschaft `picture` des Darstellers prüfen.

Parameter

pictureValue Erforderlich. Gibt einen Verweis auf das Bild eines Darstellers an.

Beispiel

Die erste Anweisung ordnet der Variablen `pictureValue` den Wert der Darstellereigenschaft „picture“ des Darstellers „Schrein“ zu, bei dem es sich um eine Bitmap handelt. Die zweite Anweisung prüft, ob `Shrine` ein Bild ist; hierzu wird der der Variablen `pictureValue` zugeordnete Wert überprüft.

```
-- Lingo syntax
pictureValue = member("Shrine").picture
put pictureP(pictureValue)

// JavaScript syntax
var pictureValue = member("Shrine").picture;
put(pictureP(pictureValue));
```

Das Resultat ist 1, das numerische Gegenstück zu TRUE.

play() (3D)

Syntax

```
member(whichCastmember).model(whichModel).bonesPlayer.play()
member(whichCastmember).model(whichModel).keyframePlayer.play()
member(whichCastmember).model(whichModel).bonesPlayer.play(motionName {, looped, startTime,
endTime, scale, offset})
member(whichCastmember).model(whichModel).keyframePlayer.play(motionName {, looped,
startTime, endTime, scale, offset})
```

Beschreibung

Dieser 3D-Befehl für die Modifizierer `#keyframePlayer` und `#bonesPlayer` leitet die Ausführung einer Bewegung ein bzw. setzt sie fort.

Wenn die Bewegung eines Modells mit diesem Befehl eingeleitet oder fortgesetzt wurde, wird die Modelleigenschaft `bonesPlayer.playing` auf `TRUE` gesetzt.

Verwenden Sie den Befehl `play()` ohne Parameter, um die Ausführung einer mit dem Befehl `pause()` angehaltenen Bewegung fortzusetzen.

Wenn Sie mit dem Befehl `play()` eine Bewegung einleiten, wird sie am Anfang der Abspielliste des Modifizierers eingefügt. Wenn dadurch die Wiedergabe einer anderen Bewegung unterbrochen wird, bleibt die unterbrochene Bewegung in der Abspielliste an der nächsten Position nach der neu eingeleiteten Bewegung vorhanden. Sobald die neu eingeleitete Bewegung endet (falls sie nicht in Schleife läuft) bzw. der Befehl `playNext()` aufgerufen wird, wird die Wiedergabe der unterbrochenen Bewegung an der Stelle fortgesetzt, an der sie unterbrochen wurde.

Parameter

motionName Erforderlich. Gibt den Namen der auszuführenden Bewegung an. Wenn nur der Parameter *motionName* an `play()` übergeben wird, führt das Modell die Bewegung einmal von Anfang bis Ende aus, und zwar mit der durch die Modifiziereigenschaft `playRate` festgelegten Geschwindigkeit.

looped Optional. Gibt an, ob die Bewegung einmal (`FALSE`) oder kontinuierlich (`TRUE`) abgespielt wird.

startTime Optional. Wird in Millisekunden ab Bewegungsbeginn gemessen. Wenn *looped* auf `TRUE` gesetzt ist, beginnt die erste Iteration der Schleife bei *offset* und endet bei *endTime*. Alle nachfolgenden Wiederholungen dieser Bewegung beginnen bei *startTime* und enden bei *endTime*.

endTime Optional. Wird in Millisekunden ab Bewegungsbeginn gemessen. Wenn *looped* auf `FALSE` gesetzt ist, beginnt die Bewegung bei *offset* und endet bei *endTime*. Wenn *looped* auf `TRUE` gesetzt ist, beginnt die erste Iteration der Schleife bei *offset* und endet bei *endTime*. Alle nachfolgenden Wiederholungen beginnen bei *startTime* und enden bei *endTime*. Setzen Sie *endTime* auf -1, wenn die Bewegung bis zum Ende abgespielt werden soll.

playRate Optional. Gibt die tatsächliche Wiedergabegeschwindigkeit der Bewegung an. *playRate* wird mit der Eigenschaft `playRate` des Modifizierers `#keyframePlayer` bzw. `#bonesPlayer` des Models multipliziert, um die tatsächliche Wiedergabegeschwindigkeit zu ermitteln.

offset Optional. Wird in Millisekunden ab Bewegungsbeginn gemessen. Wenn *looped* auf `FALSE` gesetzt ist, beginnt die Bewegung bei *offset* und endet bei *endTime*. Wenn *looped* auf `TRUE` gesetzt ist, beginnt die erste Iteration der Schleife bei *offset* und endet bei *endTime*. Alle nachfolgenden Wiederholungen beginnen bei *startTime* und enden bei *cropEnd*. Sie können für den Parameter *offset* auch den Wert `#synchronized` angeben, damit die Bewegung an der gleichen relativen Position beginnt wie die gegenwärtig laufende Animation.

Beispiel

Der folgende Befehl bewirkt, dass das Modell "Geher" die Wiedergabe der Bewegung "Fallen" beginnt. Nach Abschluss dieser Bewegung setzt das Modell die Wiedergabe der zuvor laufenden Bewegung fort.

```
sprite(1).member.model("Walker").bonesPlayer.play("Fall", 0, 0, -1, 1, 0)
```

Der folgende Befehl bewirkt, dass das Modell `Walker` die Wiedergabe der Bewegung `Kick` beginnt. Wenn `Walker` gegenwärtig eine Bewegung ausführt, wird diese durch `Kick` unterbrochen und ein Abschnitt von `Kick` wird in einer kontinuierlichen Schleife abgespielt. Die erste Iteration der Schleife beginnt 2000 Millisekunden nach Bewegungsanfang. Alle nachfolgenden Iterationen der Schleife beginnen 1000 Millisekunden nach Anfang von `Kick` und enden 5000 Millisekunden nach Anfang von `Kick`. Die Wiedergabegeschwindigkeit ist das Dreifache der Eigenschaft `playRate` des Modellmodifizierers `bonesPlayer`.

```
sprite(1).member.model("Walker").bonesPlayer.play("Kick", 1, 1000, 5000, 3, 2000)
```

Siehe auch

`queue() (3D)`, `playNext() (3D)`, `playRate (3D)`, `playlist`, `pause() (3D)`, `removeLast()`, `playing (3D)`

play() (DVD)

Syntax

```
-- Lingo syntax
dvdObjRef.play()
dvdObjRef.play(beginTitle, beginChapter, endTitle, endChapter)
dvdObjRef.play(beginTimeList, endTimeList)

// JavaScript syntax
dvdObjRef.play();
dvdObjRef.play(beginTitle, beginChapter, endTitle, endChapter);
dvdObjRef.play(beginTimeList, beginTimeList);
```

Beschreibung

Diese DVD-Methode startet die Wiedergabe oder setzt sie fort.

Ohne Angabe von Parametern setzt diese Methode die Wiedergabe fort, wenn sie angehalten ist, bzw. startet sie bei beendeter Wiedergabe am Anfang einer Disc bzw. ab dem über die Eigenschaft `startTimeList` angegebenen Wert. Die Wiedergabe wird bis zum optional durch die Eigenschaft `stopTimeList` festgelegten Wert fortgesetzt.

Bei Angabe der Parameter *beginTitle* und *beginChapter* sowie *endTitle* und *endChapter* beginnt diese Methode die Wiedergabe bei einem vorgegebenen Titel bzw. Kapitel. Die Wiedergabe wird bis zu den optional festgelegten Parameterwerten *endTitle* und *endChapter* fortgesetzt.

Bei Angabe der Parameter *beginTimeList* und *endTimeList* erfolgt die Wiedergabe durch diese Methode ab dem durch den Parameter `beginTimeList` festgelegten Wert bis zu dem durch den Parameter `endTimeList` festgelegten Wert.

Folgende Listenformate werden für die Parameter `beginTimeList` und `endTimeList` verwendet:

```
[#title:1, #chapter:1, #hours:0, #minutes:1, #seconds:1]
```

oder

```
[#title:1, #hours:0, #minutes:1, #seconds:1]
```

Bei Erfolg gibt diese Methode 0 zurück.

Parameter

beginTitle Erforderlich, wenn die Wiedergabe bei einem vorgegebenen Titel oder Kapitel beginnt. Eine Zahl, die den Titel angibt, der das wiederzugebende Kapitel enthält. Dieser Parameter setzt die Darstellereigenschaft `startTimeList` außer Kraft.

beginChapter Erforderlich, wenn die Wiedergabe bei einem vorgegebenen Titel oder Kapitel beginnt. Eine Zahl, die das wiederzugebende Kapitel angibt. Dieser Parameter setzt die Darstellereigenschaft `startTimeList` außer Kraft.

endTitle Erforderlich, wenn die Wiedergabe bei einem vorgegebenen Titel oder Kapitel endet. Eine Zahl, die den Titel angibt, bei dem die Wiedergabe enden soll. Dieser Parameter setzt die Darstellereigenschaft `stopTimeList` außer Kraft.

endChapter Erforderlich, wenn die Wiedergabe bei einem vorgegebenen Titel oder Kapitel endet. Eine Zahl, die das wiederzugebende Kapitel angibt. Dieser Parameter setzt die Darstellereigenschaft `stopTimeList` außer Kraft.

beginTimeList Erforderlich, wenn die Wiedergabe zu einer vorgegebenen Startzeit beginnt. Eine Eigenschaftsliste, die die Zeit angibt, zu der die Wiedergabe starten soll. Dieser Parameter setzt die Darstellereigenschaft `startTimeList` außer Kraft.

endTimeList Erforderlich, wenn die Wiedergabe zu einer vorgegebenen Startzeit beginnt. Eine Eigenschaftsliste, die die Zeit angibt, zu der die Wiedergabe enden soll. Dieser Parameter setzt die Darstellereigenschaft `stopTimeList` außer Kraft.

Beispiel

Die folgende Anweisung setzt die Wiedergabe eines angehaltenen Sprites fort:

```
-- Lingo syntax
member(12).play()

// JavaScript syntax
member(12).play();
```

Die folgenden Anweisungen beginnen die Wiedergabe mit Kapitel 2 von Titel 1 und beenden die Wiedergabe mit Kapitel 4:

```
member(15).play([#title:1, #chapter:2], [#title:1, #chapter:4])
```

oder

```
member(15).play(1,2,1,4)
```

Die folgenden Anweisungen beginnen die Wiedergabe ab der 10. Sekunde von Kapitel 2 und beenden die Wiedergabe nach 17 Sekunden:

```
member(15).play([#title:2, #seconds:10], [#title:2, #seconds:17])
```

Siehe auch

DVD, startTimeList, stopTimeList

play() (Mixer)

Syntax

```
Mixer.play([soundobjectlist])
```

Beschreibung

Diese Mixermethode startet alle Soundobjekte im Mixer, falls keine Soundobjekte übergeben werden, oder spielt die in der Liste vorhandenen Soundobjekte ab.

Wirkung auf den Mixer	Funktion
Abspielen()	Spielt den Mixer mit den Soundobjekten ab, die vor dem Aufruf der Methode `play()` aufgerufen wurden.
Abspielen([])	Spielt nur den Mixer ab. Soundobjekte werden nicht abgespielt.
Play([so1, so2])	Spielt den Mixer mit den angegebenen Soundobjekten mit den Referenzen `so1` und `so2` ab.
Play(["so1", "so2"])	Spielt den Mixer mit den angegebenen Soundobjekten mit den Namen `so1` and `so2`.

Beispiele

```
-- Lingo syntax
on mouseUp me
     mixer1.play() --Starts playing mixer1.
end

//JavaScript syntax
function mouseup()
{
mixer1.play(); //Starts playing mixer1.

}
```

Siehe auch

`pause() (Mixer)`, `Mixer`

play() (MP4Media/FLV)

Syntax

```
sprite(1).play()
member(1).play()
```

Beschreibung

Diese MP4Media/FLV-Darsteller oder -Sprite-Methode startet den Abspielvorgang des Videos, wenn dieses pausiert oder gestoppt wurde. Der Wert `mediaStatus` ändert sich zu `#playing`.

Beispiele

Das folgende Beispiel startet die Wiedergabe von „sprite 2“, einem MP4Media-Sprite.

```
-- Lingo syntax
sprite(2).play()
member("MP4Media").play()

// JavaScript syntax
sprite(2).play();
member("MP4Media").play();
```

Siehe auch

pause (MP4Media/FLV)

play() (RealMedia, SWA, Windows Media)

Syntax

```
-- Lingo syntax
windowsMediaObjRef.play()
realMediaObjRef.play()

// JavaScript syntax
windowsMediaObjRef.play();
realMediaObjRef.play();
```

Beschreibung

Diese Methode für Windows Media- und RealMedia-Darsteller und -Sprites gibt den Windows Media- oder RealMedia-Darsteller bzw. das -Sprite wieder, der/das sich auf der Bühne befindet.

Bei Darstellern wird Audio nur gerendert, wenn es im Film vorkommt. Wird der Darsteller bereits wiedergegeben, hat das Aufrufen dieser Methode keine Wirkung.

Parameter

Keiner

Beispiel

In den folgenden Beispielen wird das Streaming für den Stream in „Sprite 2“ und im Darsteller "Real" eingeleitet:

```
-- Lingo syntax
sprite(2).play()
member("Real").play()

// JavaScript syntax
sprite(2).play();
member("Real").play();
```

Siehe auch

RealMedia, Windows Media

play() (Soundkanal)

Syntax

```
-- Lingo syntax
soundChannelObjRef.play()
soundChannelObjRef.play(memberObjRef)
soundChannelObjRef.play(propList)

// JavaScript syntax
soundChannelObjRef.play();
soundChannelObjRef.play(memberObjRef);
soundChannelObjRef.play(propList);
```

Beschreibung

Diese Soundkanalmethode spielt Sounds ab, die sich in einem Soundkanal in der Warteschlange befinden, bzw. stellt einen angegebenen Darsteller in die Warteschlange und beginnt, ihn abzuspielen.

Es dauert einige Zeit, bis Sounddarsteller in den Arbeitsspeicher geladen und abspielbereit sind. Es empfiehlt sich, Sounds mit `queue()` in die Warteschlange zu stellen, bevor die Wiedergabe beginnen soll, und dann das erste Format dieser Funktion zu verwenden. In den zweiten Formaten wird die Vorausladefunktionalität des Befehls `queue()` nicht genutzt.

Anhand einer optionalen Eigenschaftsliste können Sie genaue Abspieleinstellungen für einen Sound angeben.

Ein Beispiel für `play()` in einem fertigen Film finden Sie im Film „Sound Control“ im Ordner „Learning/Lingo“ im Director-Anwendungsordner.

Parameter

memberObjRef Erforderlich bei der Wiedergabe eines bestimmten Darstellers. Ein Verweis auf das Darstellerobjekt, das in die Warteschlange gestellt und wiedergegeben werden soll.

propList Erforderlich beim Angeben von Wiedergabeeinstellungen für einen Sound. Eine Eigenschaftsliste, die die genauen Wiedergabeeinstellungen für den Sound angibt. Die folgenden Eigenschaften können bei Bedarf eingestellt werden:

Eigenschaft	Beschreibung
`#member`	Der in die Warteschlange einzureihende Sounddarsteller. Diese Eigenschaft ist erforderlich; alle anderen sind optional.
`#startTime`	Der Zeitpunkt im Sound, zu dem die Wiedergabe beginnt, in Millisekunden. Standardmäßig ist dies der Anfangspunkt des Sounds. Weitere Informationen finden Sie unter `startTime`.
`#endTime`	Der Zeitpunkt im Sound, zu dem die Wiedergabe endet, in Millisekunden. Standardmäßig ist dies das Ende des Sounds. Weitere Informationen finden Sie unter`endTime`.
`#loopCount`	Angabe, wie oft eine Schleife abgespielt werden soll, die in `#loopStartTime` und `#loopEndTime` definiert ist. DerStandardwert ist1. Weitere Informationen finden Sie unter `loopCount`.
`#loopStartTime`	Der Zeitpunkt im Sound, zu dem die Schleifenwiedergabe beginnt, in Millisekunden. Weitere Informationen finden Sie unter `loopStartTime`.
`#loopEndTime`	Der Zeitpunkt im Sound, zu dem die Schleifenwiedergabe endet, in Millisekunden. Weitere Informationen finden Sie unter `loopEndTime`.
`#preloadTime`	Angabe, wie viel Sound vor Abspielbeginn in den Zwischenspeicher geladen werden soll, in Millisekunden. Weitere Informationen finden Sie unter `preloadTime`.

Beispiel

Die folgende Anweisung spielt den Darsteller `introMusic` in Soundkanal 1 ab:

```
-- Lingo syntax
sound(1).play(member("introMusic"))

// JavaScript syntax
sound(1).play(member("introMusic"));
```

Die folgende Anweisung spielt den Darsteller Nachspannmusik (`creditsMusic`) in Soundkanal 2 ab. Die Wiedergabe beginnt 4 Sekunden nach Anfang des Sounds und endet 15 Sekunden nach Anfang des Sounds. Der Abschnitt zwischen 10,5 und 14 Sekunden wird sechsmal wiederholt.

```
-- Lingo syntax
sound(2).play([#member:member("creditsMusic"), #startTime:4000, #endTime:15000, #loopCount:6,
#loopStartTime:10500, #loopEndTime:14000])

// JavaScript syntax
sound(2).play(propList("member",member("creditsMusic"), "startTime",4000,"endTime",15000,
"loopCount",6, "loopStartTime",10500, "loopEndTime",14000));
```

Siehe auch

endTime (Soundkanal), loopCount, loopEndTime (Soundkanal), loopStartTime, pause() (Soundkanal), preLoadTime, queue(), Sound Channel, startTime (Sound Channel), stop() (Soundkanal)

play (Soundobjekt)

Syntax

```
soundObject.play()
```

Beschreibung

Diese Soundobjektmethode startet die Wiedergabe der mit dem Soundobjekt verknüpften Audiodatei.

Beispiele

```
--Lingo syntax
on mouseUp me
     soundObjRef.play() --Starts playing the sound object associated with soundObjRef.
end

// JavaScript syntax
function mouseUp(){
soundObjRef.play(); //Starts playing the sound object associated with soundObjRef.
}
```

Siehe auch

pause (Soundobjekt)

playFile()

Syntax

```
-- Lingo syntax
soundChannelObjRef.playFile(stringFilePath)

// JavaScript syntax
soundChannelObjRef.playFile(stringFilePath);
```

Beschreibung

Diese Soundkanalmethode gibt den Sound vom Typ AIFF, SWA, AU oder WAV in einem Soundkanal wieder.

Damit der Sound richtig abgespielt wird, muss dem Film das richtige MIX-Xtra zur Verfügung stehen (in der Regel im Xtras-Ordner der Anwendung zu finden).

Wenn sich die Sounddatei in einem anderen Ordner befindet als der Film, muss *whichFile* den vollständigen Pfad zur Datei enthalten.

Zur Wiedergabe von Sounds, die von einer URL heruntergeladen wurden, ist es normalerweise ratsam, zunächst die Datei mit `downloadNetThing()` oder `preloadNetThing()` auf einen lokalen Datenträger zu laden. Auf diese Weise können Sie eventuell auftretende Probleme beim Warten auf den Abschluss des Dateiladevorgangs weitgehend vermeiden.

Die `playFile()`-Methode streamt Dateien von einem Datenträger, statt sie aus dem Arbeitsspeicher abzuspielen. Daher kann der Befehl `playFile()` beim Abspielen von Digitalvideos oder beim Laden von Darstellern in den Speicher zu Konflikten führen, wenn der Computer versucht, den Datenträger an zwei Stellen gleichzeitig zu lesen.

Parameter

stringFilePath Erforderlich. Eine Zeichenfolge, die den Namen der wiederzugebenden Datei angibt. Wenn sich die Sounddatei in einem anderen Ordner befindet als der zurzeit wiedergegebene Film, muss *stringFilePath* außerdem den vollständigen Pfad zur Datei enthalten.

Beispiel

Die folgende Anweisung spielt die Datei "Donner" in Kanal 1 ab:

```
-- Lingo syntax
sound(1).playFile("Thunder.wav")

// JavaScript syntax
sound(1).playFile("Thunder.wav");
```

Die folgende Anweisung spielt die Datei "Donner" in Kanal 3 ab:

```
-- Lingo syntax
sound(3).playFile(_movie.path & "Thunder.wav")

// JavaScript syntax
sound(3).playFile(_movie.path + "Thunder.wav");
```

Siehe auch

`play() (Soundkanal)`, `Sound Channel`, `stop() (Soundkanal)`

playNext() (Soundkanal)

Syntax

```
-- Lingo syntax
soundChannelObjRef.playNext()

// JavaScript syntax
soundChannelObjRef.playNext();
```

Beschreibung

Diese Soundkanalmethode unterbricht sofort die Wiedergabe des aktuellen Sounds, der in einem Soundkanal abgespielt wird, und beginnt, den nächsten Sound in der Warteschlange abzuspielen.

Wenn im jeweiligen Kanal keine weiteren Sounds auf die Wiedergabe warten, wird einfach der aktuelle Sound angehalten.

Parameter

Keiner

Beispiel

Die folgende Anweisung spielt den nächsten Sound in Soundkanal 2 ab:

```
-- Lingo syntax
sound(2).playNext()

// JavaScript syntax
sound(2).playNext();
```

Siehe auch

`pause() (Soundkanal)`, `play() (Soundkanal)`, `Sound Channel`,`stop() (Soundkanal)`

playNext() (3D)

Syntax

```
member(whichMember).model(whichModel).bonesPlayer.playNext()
member(whichMember).model(whichModel).keyframePlayer.playNext()
```

Beschreibung

Dieser 3D-Befehl für die Modifizierer `#keyframePlayer` und `#bonesPlayer` leitet die Wiedergabe der nächsten Bewegung in der Abspielliste des am Modell angebrachten Modifizierers `#keyframePlayer` bzw. `#bonesPlayer` ein. Die aktuelle Bewegung, bei der es sich um den ersten Eintrag in der Abspielliste handelt, wird unterbrochen und aus der Abspielliste entfernt.

Wenn die Überblendung aktiviert ist und die Abspielliste zwei oder mehr Bewegungen enthält, erfolgt beim Aufruf von `playNext()` eine Überblendung von der aktuellen Bewegung auf die nächste Bewegung in der Abspielliste.

Beispiel

Die folgende Anweisung unterbricht die Bewegung, die gegenwärtig von Modell 1 ausgeführt wird, und leitet die Wiedergabe der nächsten Bewegung in der Abspielliste ein:

```
-- Lingo
member("scene").model[1].bonesPlayer.playnext()

// Javascript
member("scene").getProp("model",1).bonesPlayer.playNext();
```

Siehe auch

blend (3D), playlist

playerParentalLevel()

Syntax

```
-- Lingo syntax
dvdObjRef.playerParentalLevel()

// JavaScript syntax
dvdObjRef.playerParentalLevel();
```

Beschreibung

Diese DVD-Methode gibt die übergeordnete Ebene des Players zurück.

Mögliche übergeordnete Ebenen sind 1 bis 8.

Parameter

Keiner

Siehe auch

DVD

point()

Syntax

```
-- Lingo syntax
point(intH, intV)

// JavaScript syntax
point(intH, intV);
```

Beschreibung

Diese Top-Level-Funktion und dieser Datentyp geben einen Punkt zurück, der über vorgegebene horizontale und vertikaleKoordinaten verfügt.

Ein Punkt, der über die beiden Eigenschaften `locH` und `locV` verfügt.

Die Punktkoordinaten können nur durch arithmetische Operationen mithilfe von Lingo geändert werden. Beispielsweise können die folgenden zwei Punkte mithilfe von Lingo addiert werden, doch `NaN` wird unter Verwendung von JavaScript-Syntax zurückgegeben:

```
-- Lingo
pointA = point(10,10)
pointB = point(5,5)
put(pointA + pointB)
-- point(15,15)

// JavaScript syntax
var pointA = point(10,10);
var pointB = point(5,5);
trace(pointA + pointB);
// NaN
```

Beispiele für `point()` in einem fertigen Film finden Sie in den Filmen „Imaging“ und „Vector Shapes“ im Ordner „Learning\\Lingo_“ im Director-Anwendungsordner.

Parameter

intH Erforderlich. Eine Ganzzahl, die die horizontale Koordinate des Punkts angibt.

intV Erforderlich. Eine Ganzzahl, die die vertikale Koordinate des Punkts angibt.

Beispiel

Die folgende Anweisung setzt die Variable `lastLocation` auf den Punkt (250, 400):

```
-- Lingo syntax
lastLocation = point(250, 400)

// JavaScript syntax
var lastLocation = point(250, 400);
```

Die folgende Anweisung fügt 5 Pixel zur horizontalen Koordinaten des Punktes hinzu, der der Variablen `myPoint` zugeordnet ist:

```
-- Lingo syntax
myPoint.locH = myPoint.locH + 5

// JavaScript syntax
myPoint.locH = myPoint.locH + 5;
```

Nur in Lingo setzen die folgenden Anweisungen die Bühnenkoordinaten eines Sprites auf mouseH und mouseV plus 10 Pixel. Die beiden Anweisungen sind gleichbedeutend.

```
-- Lingo syntax
sprite(_mouse.clickOn).loc = point(_mouse.mouseH, _mouse.mouseV) + point(10, 10)
sprite(_mouse.clickOn).loc = _mouse.mouseLoc + 10
```

Siehe auch

locH, locV

pointAt

Syntax

```
member(whichCastmember).model(whichModel).pointAt(vectorPosition{, vectorUp})
member(whichCastmember).camera(whichCamera).pointAt(vectorPosition{, vectorUp})
member(whichCastmember).light(whichLight).pointAt(vectorPosition{, vectorUp})
member(whichCastmember).group(whichGroup).pointAt(vectorPosition{, vectorUp})
```

Beschreibung

Dieser 3D-Befehl dreht das referenzierte Objekt zuerst so, dass sein Vorwärtsvektor auf eine angegebene weltbezogene Position zeigt, und dann so, dass sein Aufwärtsvektor in die Richtung zeigt, die von einem angegebenen weltbezogenen Vektor vorgegeben wird.

Die Vorwärts- und Aufwärtsvektoren des Objekts werden durch die Objekteigenschaft `pointAtOrientation` definiert.

Parameter

vectorPosition Erforderlich. Gibt die weltbezogene Position an. Dieser Wert kann auch eine Knotenreferenz sein.

vectorUp Optional. Gibt einen weltbezogenen Vektor an, der vorgibt, wohin der Aufwärtsvektor des Objekts zeigen soll. Wenn dieser Parameter nicht angegeben wird, verwendet `pointAt` standardmäßig die y-Achse der Welt als Aufwärtshinweisvektor. Wenn Sie versuchen, das Objekt auf eine Position zu richten, an der sein Vorwärtsvektor parallel zur y-Achse der Welt liegt, wird die x-Achse der Welt als Aufwärtshinweisvektor verwendet. Die Richtung, in die Vorwärtsrichtung des Objekts zeigen soll, und die durch *vectorUp* angegebene Richtung brauchen nicht senkrecht zueinander zu verlaufen, da dieser Befehl nur den Parameter *vectorUp* als Hinweisvektor verwendet.

Beispiel

Das Beispiel richtet drei Objekte auf das Modell `Mars`: die Kamera `MarsCam`, die Lichtquelle `BrightSpot` und das Modell `BigGun`:

```
thisWorldPosn = member("Scene").model("Mars").worldPosition
member("Scene").camera("MarsCam").pointAt(thisWorldPosn)
member("Scene").light("BrightSpot").pointAt(thisWorldPosn)
member("Scene").model("BigGun").pointAt(thisWorldPosn, vector(0,0,45))
```

Wenn Sie eine uneinheitliche Skalierung und einen benutzerdefinierten `pointAtOrientation`-Wert auf den gleiche Node wie etwa ein Modell anwenden, bewirkt der Aufruf von `pointAt` höchstwahrscheinlich eine unvorhergesehene uneinheitliche Skalierung. Dies liegt an der Reihenfolge, in der die uneinheitliche Skalierung und die Drehung angewendet werden, um den Node korrekt auszurichten. Um dieses Problem zu verhindern, führen Sie einen der folgenden Schritte aus:

- Vermeiden Sie die gleichzeitige Verwendung einer uneinheitlichen Skalierung und eines nicht-standardmäßigen `pointAtOrientation`-Wertes für ein und denselben Node.
- Entfernen Sie vor dem Aufruf von `pointAt`den Skalierungswert und wenden Sie ihn anschließend wieder an.

  Beispiel:

  ```
  scale = node.transform.scale
  node.scale = vector( 1, 1, 1 )
  node.pointAt(vector(0, 0, 0)) -- non-default pointAtOrientation
  node.transform.scale = scale
  ```

Siehe auch

pointAtOrientation

pointInHyperlink()

Syntax

```
-- Lingo syntax
spriteObjRef.pointInHyperlink(point)

// JavaScript syntax
spriteObjRef.pointInHyperlink(point);
```

Beschreibung

Diese Text-Sprite-Funktion gibt den Wert TRUE oder FALSE zurück, je nachdem, ob der angegebene Punkt sich im Text-Sprite innerhalb eines Hyperlinks befindet oder nicht. In der Regel ist der verwendete Punkt die Cursorposition. Diese Funktion ist sehr praktisch, wenn ein benutzerdefinierter Cursor festgelegt werden soll.

Parameter

point Erforderlich. Gibt den zu testenden Punkt an.

Beispiel

Die folgende Anweisung prüft, ob sich die Maus über einen Hyperlink in sprite(1) bewegt.

```
-- Lingo
Put Sprite(1).pointInHyperLink(_mouse.mouseLoc)

// Javascript
trace(sprite(1).pointInHyperLink(_mouse.mouseLoc));
```

Siehe auch

cursor(), mouseLoc

pointToChar()

Syntax

```
-- Lingo syntax
spriteObjRef.pointToChar(pointToTranslate)

// JavaScript syntax
spriteObjRef.pointToChar(pointToTranslate);
```

Beschreibung

Diese Funktion gibt eine Ganzzahl zurück, die der Zeichenposition innerhalb des Text- oder Feld-Sprites an einer angegebenen Bildschirmkoordinate entspricht. Wenn sich der Punkt nicht innerhalb des Textes befindet, wird der Wert -1 zurückgegeben.

Mit dieser Funktion kann das Zeichen ermittelt werden, auf dem sich der Cursor befindet.

Parameter

pointToTranslate Erforderlich. Gibt die zu testende Bildschirmkoordinate an.

Beispiel

Die folgenden Anweisungen zeigen im Nachrichtenfenster die Nummer des angeklickten Zeichens sowie den Buchstaben an:

```
--Lingo syntax
property spriteNum

on mouseDown me
    pointClicked = _mouse.mouseLoc
    currentMember = sprite(spriteNum).member
    charNum = sprite(spriteNum).pointToChar(pointClicked)
    actualChar = currentMember.char[charNum]
    put("Clicked character" && charNum & ", the letter" && actualChar)
end

// JavaScript syntax
function mouseDown() {
    var pointClicked = _mouse.mouseLoc;
    var currentMember = sprite(this.spriteNum).member;
    var charNum = sprite(this.spriteNum).pointToChar(pointClicked);
    var actualChar = currentMember.getProp("char", charNum);
    put("Clicked character " + charNum +", the letter " + actualChar);
}
```

Siehe auch

mouseLoc, pointToWord(), pointToItem(), pointToLine(), pointToParagraph()

pointToItem()

Syntax

```
-- Lingo syntax
spriteObjRef.pointToItem(pointToTranslate)

// JavaScript syntax
spriteObjRef.pointToItem(pointToTranslate);
```

Beschreibung

Diese Funktion gibt eine Ganzzahl zurück, die der Elementposition innerhalb des Text- oder Feld-Sprites an einer angegebenen Bildschirmkoordinate entspricht. Wenn sich der Punkt nicht innerhalb des Textes befindet, wird der Wert -1 zurückgegeben. Elemente werden durch die Eigenschaft `itemDelimiter` getrennt, die standardmäßig auf ein Komma eingestellt ist.

Durch diese Funktion kann ermittelt werden, auf welchem Element sich der Cursor befindet.

Parameter

pointToTranslate Erforderlich. Gibt die zu testende Bildschirmkoordinate an.

Beispiel

Die folgenden Anweisungen zeigen im Nachrichtenfenster die Nummer des angeklickten Elements sowie den Text des Elements an:

```
--Lingo syntax
property spriteNum

on mouseDown me
    pointClicked = _mouse.mouseLoc
    currentMember = sprite(spriteNum).member
    itemNum = sprite(spriteNum).pointToItem(pointClicked)
    itemText = currentMember.item[itemNum]
    put("Clicked item" && itemNum & ", the text" && itemText)
end

// JavaScript syntax
function mouseDown() {
    var pointClicked = _mouse.mouseLoc;
    var currentMember = sprite(this.spriteNum).member;
    var itemNum = sprite(this.spriteNum).pointToItem(pointClicked);
    var itemText = currentMember.getProp("item",itemNum);
    trace( "Clicked item " + itemNum + ", the text " + itemText);
}
```

Siehe auch

itemDelimiter, mouseLoc, pointToChar(), pointToWord(), pointToLine(), pointToParagraph()

pointToLine()

Syntax

```
-- Lingo syntax
spriteObjRef.pointToLine(pointToTranslate)

// JavaScript syntax
spriteObjRef.pointToLine(pointToTranslate);
```

Beschreibung

Diese Funktion gibt eine Ganzzahl zurück, die der Elementposition innerhalb des Text- oder Feld-Sprites an einer angegebenen Bildschirmkoordinate entspricht. Wenn sich der Punkt nicht innerhalb des Textes befindet, wird der Wert -1 zurückgegeben. Zeilen werden im Text- oder Felddarsteller durch Zeilenschaltungen getrennt.

Mit dieser Funktion kann ermittelt werden, in welcher Zeile sich der Cursor befindet.

Parameter

pointToTranslate Erforderlich. Gibt die zu testende Bildschirmkoordinate an.

Beispiel

Die folgenden Anweisungen zeigen im Nachrichtenfenster die Nummer der angeklickten Zeile sowie den Text der Zeile an:

```
-- Lingo syntax
property spriteNum

on mouseDown me
    pointClicked = _mouse.mouseLoc
    currentMember = sprite(spriteNum).member
    lineNum = sprite(spriteNum).pointToLine(pointClicked)
    lineText = currentMember.line[lineNum]
    put("Clicked line" && lineNum & ", the text" && lineText)
end

// JavaScript syntax
functionmouseDown() {
    var pointClicked = _mouse.mouseLoc;
    var currentMember = sprite(this.spriteNum).member;
    var lineNum = sprite(this.spriteNum).pointToLine(pointClicked);
    var lineText = currentMember.getProp("line", lineNum);
    put("Clicked line " + lineNum + ", the text " + lineText);
}
```

Siehe auch

itemDelimiter, mouseLoc, pointToChar(), pointToWord(), pointToItem(), pointToParagraph()

pointToParagraph()

Syntax

```
-- Lingo syntax
spriteObjRef.pointToParagraph(pointToTranslate)

// JavaScript syntax
spriteObjRef.pointToParagraph(pointToTranslate);
```

Beschreibung

Diese Funktion gibt eine Ganzzahl zurück, die der Absatznummer innerhalb des Text- oder Feld-Sprites an einer angegebenen Bildschirmkoordinate entspricht. Wenn sich der Punkt nicht innerhalb des Textes befindet, wird der Wert -1 zurückgegeben. Absätze werden in einem Textblock durch Zeilenschaltungen getrennt.

Mit dieser Funktion kann ermittelt werden, in welchem Absatz sich der Cursor befindet.

Parameter

pointToTranslate Erforderlich. Gibt die zu testende Bildschirmkoordinate an.

Beispiel

Die folgenden Anweisungen zeigen im Nachrichtenfenster die Nummer des angeklickten Absatzes sowie den Text des Absatzes an:

```
-- Lingo syntax
property spriteNum

on mouseDown me
    pointClicked = _mouse.mouseLoc
    currentMember = sprite(spriteNum).member
    paragraphNum = sprite(spriteNum).pointToParagraph(pointClicked)
    paragraphText = currentMember.paragraph[paragraphNum]
    put("Clicked paragraph" && paragraphNum & ", the text" && paragraphText)
end

// JavaScript syntax
function mouseDown() {
    var pointClicked = _mouse.mouseLoc;
    var currentMember = sprite(this.spriteNum).member;
    var paragraphNum = sprite(this.spriteNum).pointToParagraph(pointClicked);
    var paragraphText = currentMember.getProp("paragraph", paragraphNum);
    trace("Clicked paragraph " + paragraphNum + ", the text " + paragraphText);
}
```

Siehe auch

itemDelimiter, mouseLoc, pointToChar(), pointToWord(), pointToItem(), pointToLine()

pointToWord()

Syntax

```
-- Lingo syntax
spriteObjRef.pointToWord(pointToTranslate)

// JavaScript syntax
spriteObjRef.pointToWord(pointToTranslate);
```

Beschreibung

Diese Funktion gibt eine Ganzzahl zurück, die der Zahl eines Wortes innerhalb des Text- oder Feld-Sprites an einer angegebenen Bildschirmkoordinate entspricht. Wenn sich der Punkt nicht innerhalb des Textes befindet, wird der Wert -1 zurückgegeben. Wörter werden in einem Textblock durch Leerzeichen getrennt.

Mit dieser Funktion kann ermittelt werden, auf welchem Wort sich der Cursor befindet.

Parameter

pointToTranslate Erforderlich. Gibt die zu testende Bildschirmkoordinate an.

Beispiel

Die folgenden Anweisungen zeigen im Nachrichtenfenster die Nummer des angeklickten Wortes sowie den Text des Wortes an:

```
-- Lingo syntax
property spriteNum

on mouseDown me
    pointClicked = _mouse.mouseLoc
    currentMember = sprite(spriteNum).member
    wordNum = sprite(spriteNum).pointToWord(pointClicked)
    wordText = currentMember.word[wordNum]
    put("Clicked word" && wordNum & ", the text" && wordText)
end

// JavaScript syntax
function mouseDown(me) {
    var pointClicked = _mouse.mouseLoc;
    var currentMember = sprite(this.spriteNum).member;
    var wordNum = sprite(this.spriteNum).pointToWord(pointClicked);
    var wordText = currentMember.getProp("word", wordNum);
    trace("Clicked word " + wordNum + ", the text " + wordText);
}
```

Siehe auch

itemDelimiter, mouseLoc, pointToChar(), pointToItem(), pointToLine(), pointToParagraph()

postNetByteArray

Syntax

```
postNetByteArray(URL, byteArray, [propertyList])
postNetByteArray(URL, [propertyList which contains bytearray]))
```

Beschreibung

Net Lingo-Methode; sendet das Bytearray („post“) Diese Methode gleicht in etwa postNetText. Sie verwendet den Inhaltstyp als „Content-Type: multipart/form-data“. Diese Methode gibt ein Bytearray aus. Verwenden Sie die Methode „netByteArrayResult“, um die Ergebnisse dieser Methode zu erhalten.

Parameter

Parameter	Beschreibung	Erforderlich/Optional
URL	Die URL zu der Datei, die das abzurufende byteArray enthält.	Erforderlich
byteArray	Byte-Array, das versendet werden soll.	Erforderlich. Hinweis: Optional, wenn „propertyList“ verwendet wird.
propertyList	Gibt eine Eigenschaftsliste an, die für CGI-Abfragen verwendet wird. Hier kann auch die Liste der Bytearrays festgelegt werden.	Optional

Beispiele

```
--Lingo syntax
id = postNetByteArray("http://www.bytearray.com/byte.asp",bArray)

//JavaScript syntax
id = postNetByteArray("http://www.bytearray.com/byte.asp",[#name:"Director",#bArray:bArray])
```

postNetText

Syntax

```
postNetText(url, propertyList {,serverOSString} {,serverCharSetString})
postNetText(url, postText {,serverOSString} {,serverCharSetString})
```

Beschreibung

Dieser Befehl sendet eine `POST`-Anforderung an eine URL, bei der es sich um eine HTTP-Adresse mit angegebenen Daten handelt.

Dieser Befehl ist `getNetText()` ähnlich. Wie bei `getNetText()` wird die Antwort des Servers von `netTextResult(`*netID*`)` zurückgegeben, sobald `netDone(`*netID*`)` zu 1 wird und falls `netError(`*netID*`)` 0 ist (d. h. kein Fehler vorliegt).

Die optionalen Parameter können ungeachtet ihrer Position ohne Weiteres ausgelassen werden.

Dieser Befehl hat einen zusätzlichen Vorteil vor `getNetText()`: eine `postNetText()`-Abfrage kann beliebig lang sein, wohingegen die `getNetText()`-Abfrage auf die Länge einer URL (1K oder 4K, je nach Browser) beschränkt ist.

Hinweis: *Wenn Sie `postNetText` dazu benutzen, Daten an eine andere Domäne zu übertragen als die, von der aus der Film abgespielt wird, erscheint beim Abspielen im Shockwave Player ein Sicherheitswarnhinweis.*

Ein Beispiel für `postNetText` in einem fertigen Film finden Sie im Film „Forms and Post“ im Ordner Learning\\Lingo_ im Director-Anwendungsordner.

Parameter

url Erforderlich. Gibt die URL an, an die die `POST`-Anforderung gesendet werden soll.

propertyList oder *postText* Erforderlich. Gibt die Daten an, an die zusammen mit der Anforderung gesendet werden sollen. Wenn eine Eigenschaftsliste anstelle eines Strings verwendet wird, werden die Informationen genauso übertragen wie ein HTML-Formular durch einen Browser, nämlich über `METHOD=POST`. Dadurch wird innerhalb eines Director-Titels die Erstellung und Übertragung von Formulardaten erleichtert. Eigenschaftsnamen entsprechen den HTML-Formularfeldnamen und Eigenschaftswerte den Feldwerten.

In der Eigenschaftsliste können entweder Strings oder Symbole als Eigenschaftsnamen verwendet werden. Bei Verwendung eines Symbols wird dieses automatisch in einen String konvertiert, aber ohne vorangestellte Raute (#). Analog dazu werden numerische Werte bei Verwendung als Eigenschaftswerte in Strings konvertiert.

Hinweis: *Wenn ein Programm das Alternativformat verwendet – d. h. einen String anstelle einer Eigenschaftsliste –, wird der String `postText` als HTTP-POST-Anforderung an den Server gesendet, und zwar unter Verwendung des MIME-Typs "text/plain". Dies kann zwar für einige Anwendungen sehr praktisch sein, ist aber nicht kompatibel mit der Übertragung von HTML-Formularen. PHP-Skripts sollten beispielsweise immer eine Eigenschaftsliste verwenden.*

serverOSString Optional. Der Standardwert lautet `UNIX`, kann jedoch auf `Windows` oder `Mac` gesetzt werden. Er bewirkt, dass alle Zeilenschaltungen im Argument *postText* in die auf dem Server verwendeten Steuerzeichen umgewandelt werden. Bei den meisten Anwendungen ist diese Einstellung nicht erforderlich, da in Formularantworten normalerweise keine Zeilenumbrüche verwendet werden.

serverCharSetString Optional. Ist nur gültig, wenn der Benutzer ein japanisches Shift-JIS-System verwendet. Mögliche Einstellungen: `"JIS"`, `"EUC"`, `"ASCII"` und `"AUTO"`. Formulardaten werden von Shift-JIS in den angegebenen Zeichensatz umgewandelt. Rückgabedaten werden genauso behandelt wie bei `getNetText()` (d. h. aus dem angegebenen Zeichensatz in Shift-JIS umgewandelt). Bei Verwendung von `"AUTO"` werden die Formulardaten nicht aus dem lokalen Zeichensatz umgewandelt. Die vom Server zurückgesendeten Ergebnisse werden wie bei `getNetText()` konvertiert. `"ASCII"` ist die Standardeinstellung, wenn *serverCharSetString* nicht vorhanden ist. Bei `"ASCII"` werden Formulardaten oder Ergebnisse nicht umgewandelt.

Beispiel

In der folgenden Anweisung wird der Parameter *serverCharSetString* ausgelassen:

```
-- Lingo
netID = postNetText("www.mydomain.com\database.cgi", "Bill Jones", "Win")

// Javascript
netID = postNetText("www.mydomain.com\database.cgi", "Bill Jones", "Win");
```

Das folgende Beispiel zeigt, wie ein Formular aus Eingabefeldern für den Vor- und Nachnamen zusammen mit einem Drehbuch erstellt wird. Wie Sie sehen, werden sowohl `serverOSString` als auch `serverCharSetString` weggelassen:

```
-- LingonetID = postNetText("www.mydomain.com/userbase.cgi", infoList);
lastName = member("Last Name").text
firstName = member("First Name").text
totalScore = member("Current Score").text
infoList = ["FName":firstName, "LName":lastName, "Score":totalScore]
netID = postNetText("www.mydomain.com/userbase.cgi", infoList);

// Javascript
lastName = member("Last Name").text;
firstName = member("First Name").text;
totalScore = member("Current Score").text;
infoList = propList("FName",firstName, "LName",lastName, "Score",totalScore);
netID = postNetText("www.mydomain.com/userbase.cgi", infoList);
```

Siehe auch

`getNetText()`, `netTextResult()`, `netDone()`, `netError()`

power()

Syntax

```
power(base, exponent)
```

Beschreibung

Diese mathematische Funktion berechnet den Wert einer angegebenen Zahl zu einem vorgegebenen Exponenten.

Parameter
base Erforderlich. Gibt die Basis an.

exponent Erforderlich. Gibt den Wert des Exponenten an.

Beispiel
Die folgende Anweisung setzt die Variable vResult auf den Wert 4 hoch 3:

```
-- Lingo
set vResult = power(4,3)

// Javascript
Var vResult = Math.pow(4,3);
```

preLoad() (Darsteller)

Syntax

```
-- Lingo syntax
memberObjRef.preLoad({toMemberObjRef})

// JavaScript syntax
memberObjRef.preLoad({toMemberObjRef});
```

Beschreibung
Diese Darstellermethode lädt einen Darsteller bzw. einen Darstellerbereich im Voraus in den Speicher und beendet den Vorausladevorgang, sobald der Speicher voll ist oder alle angegebenen Darsteller vorausgeladen wurden.

Ohne den Parameter *toMemberObjRef* werden von `preLoad()` alle verwendeten Darsteller vorausgeladen – angefangen vom aktuellen bis hin zum letzten Bild eines Films.

Parameter
toMemberObjRef Optional. Ein Verweis auf den letzten Darsteller im Darstellerbereich, der in den Speicher geladen wird. Der erste Darsteller im Bereich wird durch *memberObjRef* angegeben.

Beispiel
Die folgende Anweisung zeigt im Nachrichtenfenster an, ob der QuickTime-Film "Drehstuhl" im Voraus in den Speicher geladen werden kann:

```
-- Lingo syntax
put(member("Rotating Chair").preload())

// JavaScript syntax
put(member("Rotating Chair").preload());
```

Die folgende `startMovie`-Prozedur konfiguriert einen Flash-Filmdarsteller für das Streaming und stellt dann für ihn die Eigenschaft `bufferSize` ein:

```
-- Lingo syntax
on startMovie
    member("Flash Demo").preload = FALSE
    member("Flash Demo").bufferSize = 65536
end

// JavaScript syntax
function startMovie() {
    member("Flash Demo").preload = false;
    member("Flash Demo").bufferSize = 65536;
}
```

Siehe auch

Member

preLoad() (Film)

Syntax

```
-- Lingo syntax
_movie.preLoad({frameNameOrNum})
_movie.preLoad(fromFrameNameOrNum, toFrameNameOrNum)

// JavaScript syntax
_movie.preLoad({frameNameorNum});
_movie.preLoad(fromFrameNameOrNum, toFrameNameOrNum);
```

Beschreibung

Diese Methode lädt die Darsteller des angegebenen Bildes oder Bildbereichs im Voraus in den Speicher. Der Vorausladevorgang wird abgebrochen, sobald der Speicher voll ist oder alle angegebenen Darsteller vorausgeladen sind.

- Wenn dieser Befehl ohne Argumente benutzt wird, werden alle verwendeten Darsteller vorausgeladen – angefangen vom aktuellen bis hin zum letzten Bild des Films.
- Wenn dieser Befehl nur mit dem Argument *frameNameOrNum* benutzt wird, werden alle im Bildbereich verwendeten Darsteller vom aktuellen Bild bis hin zum Bild *frameNameOrNum* vorausgeladen, d. h. genauso, wie es durch die Bildnummer oder Beschriftung angegeben ist.
- Wenn dieser Befehl mit den beiden Argumenten *fromFrameNameOrNum* und *toFrameNameOrNum* benutzt wird, werden alle im Bildbereich verwendeten Darsteller vom Bild *fromFrameNameOrNum* bis hin zum Bild *toFrameNameOrNum* vorausgeladen, d. h. genauso, wie es durch die Bildnummer oder Beschriftung angegeben ist.

Die Methode `preLoad()` gibt außerdem die Nummer des zuletzt erfolgreich geladenen Bildes zurück. Sie erhalten diesen Wert mithilfe der `result()`-Methode.

Parameter

frameNameOrNum Optional. Eine Zeichenfolge, die das spezifische vorauszuladende Bild angibt, oder eine Ganzzahl, die die Nummer des spezifisch vorauszuladenden Bildes angibt.

fromFrameNameOrNum Erforderlich beim Vorausladen eines Bildbereichs. Eine Zeichenfolge, die den Namen oder die Beschriftung des ersten Bildes im vorauszuladenden Bildbereich angibt, oder eine Ganzzahl, die die Nummer des ersten Bildes im vorauszuladenden Bildbereich angibt.

toFrameNameOrNum Erforderlich beim Vorausladen eines Bildbereichs. Eine Zeichenfolge, die den Namen oder die Beschriftung des letzten Bildes im vorauszuladenden Bildbereich angibt, oder eine Ganzzahl, die die Nummer des letzten Bildes im vorauszuladenden Bildbereich angibt.

Beispiel

Die folgende Anweisung lädt die verwendeten Darsteller vom aktuellen bis hin zum Bild mit der nächsten Markierung voraus:

```
-- Lingo syntax
_movie.preLoad(_movie.marker(1))

// JavaScript syntax
_movie.preLoad(_movie.marker(1));
```

Die folgende Anweisung lädt die verwendeten Darsteller von Bild 10 bis Bild 50 voraus:

```
-- Lingo syntax
_movie.preLoad(10, 50)

// JavaScript syntax
_movie.preLoad(10, 50);
```

Siehe auch

Movie, result

preLoadBuffer()

Syntax

```
-- Lingo syntax
memberObjRef.preLoadBuffer()

// JavaScript syntax
memberObjRef.preLoadBuffer();
```

Beschreibung

Dieser Befehl lädt einen Teil einer bestimmten Shockwave Audio (SWA)-Datei im Voraus in den Speicher. Die vorausgeladene Menge wird durch die Eigenschaft `preLoadTime` bestimmt. Dieser Befehl funktioniert nur, wenn der SWA-Darsteller gestoppt ist.

Nach erfolgreicher Ausführung des Befehls `preLoadBuffer` hat die Darstellereigenschaft `state` den Wert 2.

Die meisten SWA-Darstellereigenschaften können erst getestet werden, nachdem der Befehl `preLoadBuffer` erfolgreich ausgeführt ist. These properties include: `cuePointNames`, `cuePointTimes`, `currentTime`, `duration`, `percentPlayed`, `percentStreamed`, `bitRate`, `sampleRate` und `numChannels`.

Parameter

Keiner

Beispiel

Die folgende Anweisung lädt den Darsteller "Mel Torme" in den Speicher:

```
-- Lingo syntax
member("Mel Torme").preLoadBuffer()

// JavaScript syntax
member("Mel Torme").preLoadBuffer();
```

Siehe auch

preLoadTime

preLoadMember()

Syntax

```
-- Lingo syntax
_movie.preLoadMember({memberObjRef})
_movie.preLoadMember(fromMemNameOrNum, toMemNameOrNum)

// JavaScript syntax
_movie.preLoadMember({memberObjRef});
_movie.preLoadMember(fromMemNameOrNum, toMemNameOrNum);
```

Beschreibung

Diese Filmmethode lädt Darsteller voraus und stoppt den Vorgang, wenn der Speicher voll ist oder alle angegebenen Darsteller vorausgeladen sind.

Diese Methode gibt die Darstellernummer des zuletzt erfolgreich geladenen Darstellers zurück. Sie erhalten diesen Wert mithilfe der `result()`-Methode.

Wenn Sie `preLoadMember()` ohne Argumente verwenden, lädt die Methode alle Darsteller im Film.

Bei Verwendung mit dem Argument *memberObjRef* werden nur die entsprechenden Darsteller von `preLoadMember()` vorausgeladen. Wenn es sich bei *memberObjRef* um eine Ganzzahl handelt, wird nur auf die erste Besetzungsbibliothek verwiesen. Falls *memberObjRef* eine Zeichenfolge ist, wird der erste Darsteller verwendet, dessen Name die Zeichenfolge ist.

Bei Verwendung mit den Argumenten *fromMemNameOrNum* und *toMemNameOrNum* werden alle Darsteller von `preLoadMember()` vorausgeladen, die sich in dem durch die Darstellernummern oder -namen angegebenen Bereich befinden.

Parameter

memberObjRef Optional. Ein Verweis auf den vorauszuladenden Darsteller.

fromMenNameOrNum Erforderlich beim Vorausladen eines Darstellerbereichs. Eine Zeichenfolge oder Ganzzahl, die den ersten Darsteller im vorauszuladenden Darstellerbereich angibt.

toMemNameOrNum Erforderlich beim Vorausladen eines Darstellerbereichs. Eine Zeichenfolge oder Ganzzahl, die den ersten Darsteller im vorauszuladenden Darstellerbereich angibt.

Beispiel

Die folgende Anweisung lädt den Darsteller „SWF“ in den Film.

```
-- Lingo
_movie.preLoadMember(member("SWF"))

// Javascript
_movie.preLoadMember(member("SWF")) ;
```

Siehe auch

Movie, preLoad() (Darsteller), result

preLoadMovie()

Syntax

```
-- Lingo syntax
_movie.preLoadMovie(stringMovieName)

// JavaScript syntax
_movie.preLoadMovie(stringMovieName);
```

Beschreibung

Diese Filmmethode lädt die Daten und Darsteller voraus, die mit dem ersten Bild des angegebenen Films verbunden sind. Das Vorausladen eines Films bewirkt bei Verwendung der Methoden `go()` oder `play()` einen schnelleren Start des Films.

Zum Vorausladen von Darstellern aus einer URL verwenden Sie am besten `preloadNetThing()`, um die Darsteller direkt in den Cachespeicher zu laden, oder `downLoadNetThing()`, um den Film auf einen lokalen Datenträger herunterzuladen, von dem der Film dann in den Speicher geladen wird, was eine Reduzierung der benötigten Ladezeit ermöglicht.

Parameter

stringMovieName Erforderlich. Eine Zeichenfolge, die den Namen des vorauszuladenden Films angibt.

Beispiel

Die folgende Anweisung lädt den Film "Einführung", der sich im gleichen Ordner wie der aktuelle Film befindet, voraus:

```
-- Lingo syntax
_movie.preLoadMovie("Introduction")

// JavaScript syntax
_movie.preLoadMovie("Introduction");
```

Siehe auch

downloadNetThing, go(), Movie, preloadNetThing()

preloadNetThing()

Syntax

```
preloadNetThing (url)
```

Beschreibung

Diese Funktion bewirkt, dass eine Datei aus dem Internet in den lokalen Cachespeicher vorausgeladen wird, sodass sie später verwendet werden kann, ohne erst heruntergeladen werden zu müssen. Der Rückgabewert ist eine Netzwerk-ID, über die Sie den Verlauf des Vorgangs überwachen können.

Die Funktion `preloadNetThing()` lädt die Datei herunter, während der aktuelle Film weiterläuft. Verwenden Sie `netDone()`, um herauszufinden, ob der Downloadvorgang beendet ist.

Nach dem Herunterladen einer Datei kann diese sofort angezeigt werden, da sie aus dem lokalen Cachespeicher und nicht aus dem Netzwerk übernommen wird.

Obwohl viele Netzwerkvorgänge parallel aktiv sein können, ist bei mehr als vier gleichzeitigen Vorgängen die Leistung in der Regel nicht mehr zufriedenstellend.

Das Verhalten der Funktion `preloadNetThing` wird weder von der Cachegröße noch von der Browseroption „Dokumente überprüfen“ (unter „Voreinstellungen“) beeinflusst.

Die Links einer Director-Datei werden durch die Funktion `preloadNetThing()` nicht geparst. Selbst wenn eine Director-Datei mit Besetzungen und Grafikdateien verknüpft ist, lädt `preloadNetThing()` nur die Director-Datei herunter. Die anderen verknüpften Objekte müssen separat heruntergeladen werden.

Parameter

url (erforderlich). Gibt den Namen einer gültigen Internetdatei an, wie z. B. einen Director-Film, eine Grafik oder eine FTP-Serveradresse.

Beispiel

Die folgende Anweisung verwendet `preloadNetThing()` und gibt die Netzwerk-ID für den Vorgang zurück:

```
-- Lingo
set mynetid = preloadNetThing("http://www.yourserver.com/menupage/mymovie.dir")

// Javascript
set mynetid = preloadNetThing("http://www.yourserver.com/menupage/mymovie.dir");
```

Sobald Sie mit dem Herunterladen fertig sind, können Sie mit Hilfe derselben URL zum Film navigieren. Der Film wird dann aus dem Cachespeicher und nicht über die URL abgespielt, da er sich bereits im Cache befindet.

Siehe auch

`netDone()`

preMultiply

Syntax

```
transform1.preMultiply(transform2)
```

Beschreibung

Dieser 3D-Transformationsbefehl ändert eine Transformation durch vorheriges Anwenden der Positions-, Drehungs- und Skalierungseffekte einer anderen Transformation.

Wenn *transform2* eine Drehung von 90° um die X-Achse und *transform1* eine Translation von 100 Einheiten auf der Y-Achse beschreibt, ändert `transform1.multiply(`*transform2*`)` diese Transformation so, dass sie eine Translation gefolgt von einer Drehung beschreibt. Die Anweisung `transform1.preMultiply(`*transform2*`)` ändert diese Transformation dahingehend, dass sie eine Drehung gefolgt von einer Translation beschreibt, d. h. die Operationen werden in umgekehrter Reihenfolge durchgeführt.

Parameter
transform2 Erforderlich. Gibt die Transformation an, von der Effekte auf eine andere Transformation zuvor angewendet werden.

Beispiel
Die folgende Anweisung führt eine Berechnung durch, bei der die Transformation des Modells "Mars" auf die Transformation des Modells "Pluto" angewendet wird:

```
-- Lingo
member("scene").model("Pluto").transform.preMultiply(member("scene").model("Mars").transform
)

// Javascript
member("scene").getPropRef("model" ,
i).transform.preMultiply(member("scene").getPropRef("model",j).transform) ;
// where i and j are the number index of the models "Pluto" and "Mars" respectively.
```

preRotate

Syntax

```
transformReference.preRotate( xAngle, yAngle, zAngle )
transformReference.preRotate( vector )
transformReference.preRotate( positionVector, directionVector, angle )
member( whichCastmember ).node.transform.preRotate( xAngle, yAngle, zAngle )
member( whichCastmember ).node.transform.preRotate( vector )
member( whichCastmember ).node.transform.preRotate( positionVector, directionVector, angle )
```

Beschreibung
Dieser 3D-Transformationsbefehl führt vor den aktuellen Positions-, Drehungs- und Skalierungsoffsets des referenzierten Transformationsobjekts eine Drehung durch. Die Drehung kann in Form von drei Winkelwerten angegeben werden, von denen jeder einen Drehwinkel um eine der drei Achsen definiert. Diese Winkel können explizit im Format `xAngle`, `yAngle` und `zAngle` oder anhand eines Vektors angegeben werden, bei dem die x-Komponente des Vektors der Drehung um die *x*-Achse, die y-Komponente der Drehung um die *y*-Achse und die z-Komponente der Drehung um die *z*-Achse entspricht.

Die Drehung kann auch als Drehung um eine beliebige Achse angegeben werden, die im Raum durch *positionVector* und *directionVector* definiert wird. Der Drehwinkel um diese Achse wird durch *angle* angegeben.

Node bezieht sich auf ein Modell, eine Gruppe, ein Licht oder eine Kamera.

Parameter
xAngle Erforderlich beim Anwenden einer Drehung mithilfe von *x*-, *y*- und *z*-Achsen. Gibt den Winkel der Drehung um die *X*-Achse an.

yAngle Erforderlich beim Anwenden einer Drehung mithilfe von *x*-, *y*- und *z*-Achsen. Gibt den Winkel der Drehung um die *Y*-Achse an.

zAngle Erforderlich beim Anwenden einer Drehung mithilfe von *x*-, *y*- und *z*-Achsen. Gibt den Winkel der Drehung um die *Z*-Achse an.

vector Erforderlich beim Anwenden einer Drehung mithilfe eines Vektors. Gibt den Vektor an, dessen Winkel für die Drehung verwendet werden.

positionVector Erforderlich beim Anwenden einer Drehung um eine beliebige Achse. Gibt den Positionsversatzan.

directionVector Erforderlich beim Anwenden einer Drehung um eine beliebige Achse. Gibt den Richtungsversatzan.

angle Erforderlich beim Anwenden einer Drehung um eine beliebige Achse. Gibt den Drehwinkel um eine zufällige Achse an.

Beispiel

Die folgende Zeile führt eine Drehung um 20° um alle drei Achsen durch. Da die Modelleigenschaft transform die Positions-, Drehungs- und Skalierungsoffsets des Modells relativ zu seinem Parent-Node angibt und `preRotate` die Ausrichtungsänderung vor allen vorhandenen Effekten der transform dieses Modells anwendet, wird dadurch das Modell an der Stelle gedreht, anstatt eine Umlaufbahn um seinen Parent-Node zu beschreiben.

```
-- Lingo
member("scene").model("bip01").transform.preRotate(20, 20, 20)

// Javascript
member("scene").getPropRef("model", i).transform.preRotate(20, 20, 20) ;
// where i is the number index of the model "bip01"
```

Diese Anweisung hat die gleiche Wirkung wie die folgende:

```
member("scene").model("bip01").rotate(20,20,20).
// javascript
member("scene").getPropRef("model", i).rotate(20,20,20) ;
// where i is the number index of the model "bip01"
```

Im Allgemeinen ist `preRotate()` nur bei Verwendung von Transformationsvariablen nützlich. Die folgende Zeile bewirkt, dass die Kamera um den Punkt (100, 0, 0) kreist, und zwar um 180° um die x-Achse:

```
-- Lingo
t = transform()
t.position = member("scene").camera[1].transform.position
t.preRotate(vector(100, 0, 0), vector(0, 1, 0), 180)
member("scene").camera[1].transform = t

// javascript
var t = transform() ;
t.position = member("scene").getPropRef("camera", 1).transform.position ;
t.preRotate(vector(100, 0, 0), vector(0, 1, 0), 180) ;
member("scene").getPropRef("camera",1).transform = t ;
```

Siehe auch

rotate

preScale()

Syntax

```
transformReference.preScale( xScale, yScale, zScale )
transformReference.preScale( vector )
member( whichCastmember ).node.transform.preScale( xScale, yScale, zScale )
member( whichCastmember ).node.transform.preScale( vector )
```

Beschreibung

Dieser 3D-Befehl führt vor den vorhandenen Positions-, Drehungs- und Skalierungseffekte der jeweiligen Transformation eine Skalierung durch.

Node kann ein Bezug auf ein Modell, eine Gruppe, ein Licht oder eine Kamera sein.

Parameter

xScale Erforderlich beim Anwenden einer Skalierung mithilfe von *x*-, *y*- und *z*-Achsen. Gibt die Skalierung um die *X*-Achse an.

yScale Erforderlich beim Anwenden einer Skalierung mithilfe von *x*-, *y*- und *z*-Achsen. Gibt die Skalierung um die *Y*-Achse an.

zScale Erforderlich beim Anwenden einer Skalierung mithilfe von *x*-, *y*- und *z*-Achsen. Gibt die Skalierung um die *Z*-Achse an.

vector Erforderlich beim Anwenden einer Skalierung mithilfe eines Vektors. Gibt den Vektor an, der die anzuwendende Winkel enthält.

Beispiel

In **Zeile 1** des folgenden Lingo-Codes wird ein Duplikat der Transformation von "Mond1" erstellt. Der Zugriff auf die Transformationseigenschaft eines Modells erfolgt durch Verweis.

In **Zeile 2** wird vor allen vorhandenen Positions- oder Drehungseffekten dieser Transformation eine Skalierung auf diese Transformation angewandt. Angenommen, die Transformation beschreibt den Positionsoffset und die Umlaufbahn von `Moon1` in Bezug auf seinen übergeordneten (Parent-) Planeten. Ferner sei angenommen, dass `Moon2` denselben Parent-Planeten hat wie `Moon1`. Wenn Sie in diesem Fall `scale()` anstelle von `preScale()` verwenden, wird `Moon2` doppelt so weit nach außen geschoben und um doppelt so viel um den Planeten gedreht werden wie `Moon1`. da die Skalierung auf die vorhandenen Positions- und Drehungsoffsets der Transformation angewendet wird. `preScale()` führt die Größenänderung ohne Auswirkung auf die vorhandenen Positions- und Drehungsoffsets durch.

In **Zeile 3** wird eine zusätzliche Drehung um 180° um die x-Achse des Planeten durchgeführt. `Moon2` befindet sich dann auf der gegenüberliegenden Seite der Umlaufbahn von `Moon1`. Bei Verwendung von `preRotate()` hätte sich `Moon2` an der gleichen Stelle befunden wie `Moon1`, d.h. um 180° um seine eigene x-Achse gedreht.

In **Zeile 4** wird diese neue Transformation Mond2 zugeordnet.

```
-- Lingo
t = member("scene").model("Moon1").transform.duplicate()
t.preScale(2,2,2)
t.rotate(180,0,0)
member("scene").model("Moon2").transform = t

// Javascript
var t = member("scene").getPropRef("model", i).transform.duplicate() ;
t.preScale(2,2,2) ;
t.rotate(180,0,0) ;
member("scene").getPropRef("model", i).transform = t ;
// where i the number index of model " Moon2".
```

preTranslate()

Syntax

```
transformReference.preTranslate( xIncrement, yIncrement, zIncrement )
transformReference.preTranslate( vector )
member( whichCastmember ).node.transform.preTranslate(xIncrement, yIncrement, zIncrement)
member( whichCastmember ).node.transform.preTranslate( vector )

// Javascript
member( whichCastmember ).getProp("model",a).transform.preTranslate(xIncrement, yIncrement,
zIncrement) ;
```

Beschreibung

Dieser 3D-Transformationsbefehl führt vor den aktuellen Positions-, Drehungs- und Skalierungsoffsets des referenzierten Transformationsobjekts eine Translation durch. Die Translation kann in Form von drei Inkrementwerten entlang der drei Achsen angegeben werden. Diese Inkrementwerte können explizit im Format *xIncrement*, *yIncrement* und *zIncrement* oder anhand eines Vektors angegeben werden, bei dem die x-Komponente des Vektors der Translation um die x-Achse, die y-Komponente der Translation um die y-Achse und die z-Komponente der Translation um die z-Achse entspricht.

Nach Durchführung einer Reihe von Transformationen in der angegebenen Reihenfolge befindet sich der lokale Ursprung des Modells an (0, 0, -100), sofern der Parent des Modells die Welt ist:

```
model.transform.identity()
model.transform.rotate(0, 90, 0)
model.transform.preTranslate(100, 0, 0)
```

Hätten Sie `translate()` anstelle von `preTranslate()` verwendet, würde sich der lokale Ursprung des Modells an der Position (100, 0, 0) befinden und das Modell wäre um 90° um die eigene y-Achse gedreht. Die Anweisung `model.transform.pretranslate(x, y, z)` hat die gleiche Funktion wie `model.translate(x, y, z)`. Im Allgemeinen ist `preTranslate()` nur bei Verwendung von Transformationsvariablen statt `model.transform`-Bezügen nützlich.

Parameter

xIncrement Erforderlich beim Anwenden einer Translation mithilfe von *x*-, *y*- und *z*-Achsen. Gibt die Translation um die *X*-Achse an.

yIncrement Erforderlich beim Anwenden einer Translation mithilfe von *x*-, *y*- und *z*-Achsen. Gibt die Translation um die *Y*-Achse an.

zIncrement Erforderlich beim Anwenden einer Translation mithilfe von *x*-, *y*- und *z*-Achsen. Gibt die Translation um die *Z*-Achse an.

vector Erforderlich beim Anwenden einer Translation mithilfe eines Vektors. Legt den für die Transformation vorgesehenen Vektor fest.

Beispiel

```
-- Lingo
    t = transform()
t.transform.identity()
t.transform.rotate(0, 90, 0)
t.transform.preTranslate(100, 0, 0)
gbModel = member("scene").model("mars")
gbModel.transform = t
put gbModel.transform.position
-- vector(0.0000, 0.0000, -100.0000)

// Javascript

gbModel = member("scene").getProp("model" , a) ;
// where a is the number index for the mars model.
gbModel.transform.preTranslate(xIncrement , yIncrement, zIncrement) ;
member("scene").getProp("model" , a).transform.preTranslate(xIncrement , yIncrement,
zIncrement) ;
```

print()

Syntax

```
-- Lingo syntax
spriteObjRef.print({targetName, #printingBounds})

// JavaScript syntax
spriteObjRef.print({targetName, #printingBounds});
```

Beschreibung

Dieser Befehl ruft den entsprechenden ActionScript-Befehl `print` auf (neu in Flash 5). Alle mit `#p` beschrifteten Bilder des Flash-Films werden gedruckt. Falls keine Einzelbilder beschriftet wurden, wird der gesamte Film gedruckt.

Da es sich bei der Druckausgabe eines Flash-Films um einen recht komplizierten Vorgang handelt, sollten Sie den Abschnitt über das Drucken in der Flash-5-Dokumentation aufmerksam durchlesen, bevor Sie diese Sprite-Funktion verwenden.

Parameter

targetName Optional. Gibt den Namen des Zielfilms oder Movieclips an, der gedruckt werden soll. Wird der Parameter nicht angegeben (entspricht einem Zielwert von 0), wird der Flash-Hauptfilm gedruckt.

printingBounds Optional. Gibt die Optionen für die Druckbegrenzungen an. Wird der Parameter nicht angegeben, werden die Begrenzungen des Zielfilms übernommen. Bei Verwendung des Parameters muss *printingBounds* einen der folgenden Werte enthalten:

- `#bframe`. Bei Angabe des Wertes werden die Druckbegrenzungen für die einzelnen Seiten an das jeweils zu druckende Bild angepasst.

- `#bmax`. Bei Verwendung des Parameters wird als Druckbereich ein virtuelles Rechteck verwendet, das groß genug ist für alle zu druckenden Bilder.

Beispiel
Die folgende Anweisung druckt den vorhandenen Flash-Film als Darsteller „SWF“.

```
-- Lingo
member("SWF").print()
// javascript
member("SWF").print() ;
```

printAsBitmap()

Syntax

```
-- Lingo syntax
spriteObjRef.printAsBitmap({targetName, #printingBounds})

// JavaScript syntax
spriteObjRef.printAsBitmap({targetName, #printingBounds});
```

Beschreibung
Dieser Flash-Sprite-Befehl funktioniert wie der Befehl `print`, aber nur mit Flash-Sprites. Mit dem Befehl `printAsBitmap` lassen sich jedoch auch Objekte drucken, die Alphakanaldaten enthalten.

printFrom()

Syntax

```
-- Lingo syntax
_movie.printFrom(startFrameNameOrNum {, endFrameNameOrNum, redux})

// JavaScript syntax
_movie.printFrom(startFrameNameOrNum {, endFrameNameOrNum, redux});
```

Beschreibung
Diese Filmmethode druckt alles, was in den Bildern auf der Bühne dargestellt wird, ganz gleich, ob das Bild ausgewählt ist oder nicht, und zwar angefangen mit dem durch *startFrame* angegebenen Bild. Optional können Sie auch *endFrame* und einen Skalierungsfaktor (*redux*; 100%, 50% oder 25%) angeben.

Das zu druckende Bild braucht dabei derzeit nicht angezeigt zu sein. Dieser Befehl druckt stets alle auf dem Bildschirm angezeigten Daten mit 72 Punkten pro Zoll (dpi) und im Hochformat als Bitmap aus; der Text erscheint dadurch mitunter nicht sehr gleichmäßig. Die Druckeinrichtungseinstellungen werden dabei ignoriert. Wenn Sie mehr Flexibilität beim Drucken in Director benötigen, sollten Sie sich das Xtra "PrintOMatic Lite" auf dem Installationsdatenträger ansehen.

Parameter
startFrameNameOrNum Erforderlich. Eine Zeichenfolge oder Ganzzahl, die den Namen oder die Nummer des ersten zu druckenden Bildes angibt.

endFrameNameOrNum Optional. Eine Zeichenfolge oder Ganzzahl, die den Namen oder die Nummer des letzten zu druckenden Bildes angibt.

redux Optional. Eine Ganzzahl, die den Skalierungsfaktor angibt. Gültige Werte sind 100, 50 und 25.

Beispiel

Die folgende Anweisung druckt alles, was sich in Bild 1 auf der Bühne befindet:

```
-- Lingo syntax
_movie.printFrom(1)

// JavaScript syntax
_movie.printFrom(1);
```

Die folgende Anweisung druckt alles, was in den einzelnen Bildern von Bild 10 bis Bild 25 auf der Bühne erscheint.

```
-- Lingo syntax
_movie.printFrom(10, 25, 50)

// JavaScript syntax
_movie.printFrom(10, 25, 50);
```

Siehe auch

Movie

propList()

Syntax

```
-- Lingo syntax
propList()
[:]
propList(string1, value1, string2, value2, ...)
propList(#symbol1, value1, #symbol2, value2, ...)
[#symbol1:value1, #symbol2:value2, ...]

// JavaScript syntax
propList();
propList(string1, value1, string2, value2, ...);
```

Beschreibung

Diese Top-Level-Funktion erstellt eine Eigenschaftsliste, in der jedes Element aus einem Name/Wert-Paar besteht.

Beim Erstellen einer Eigenschaftsliste mithilfe der Syntax `propList()` oder `[:]` (nur Lingo), mit Parametern oder ohne, beginnt der Index der Listenwerte mit 1.

Die maximale Länge einer einzelnen Zeile mit ausführbarem Skriptcode ist auf 256 Zeichen beschränkt. Große Eigenschaftslisten können nicht mithilfe von `propList()` erstellt werden. Um eine Eigenschaftsliste mit einer großen Datenmenge zu erstellen, schließen Sie die Daten in eckige Klammern (`[]`) ein, legen die Daten in einem Feld ab und weisen dieses Feld dann einer Variablen zu. Der Inhalt der Variablen ist eine Liste der Daten.

Parameter

string1, *string2*, ... (optional). Werte, die den Wertteil der Elemente in der Liste angeben.

value1, *value2*, ... Optional. Werte, die den Wertteil der Elemente in der Liste angeben.

#symbol1, *#symbol2*, ... (Nur Lingo, optional). Symbole, die den Namensteil der Elemente in der Liste angeben.

Beispiel

Die folgende Anweisung erstellt eine Eigenschaftsliste mit verschiedenen Eigenschaften und Werten und zeigt die verschiedenen Eigenschaftswerte anschließend im Nachrichtenfenster an:

```
-- Lingo syntax
-- using propList()
colorList = propList(#top,"red", #sides,"blue", #bottom,"green")
-- using brackets
colorList = [#top:"red", #sides:"blue", #bottom:"green"]
put(colorList.top) -- "red"
put(colorList.sides) -- "blue"
put(colorList.bottom) -- "green"

// JavaScript syntax
var colorList = propList("top","red", "sides","blue", "bottom","green");
put(colorList.top); // red
put(colorList.sides); // blue
put(colorList.bottom); // green
```

Siehe auch

`list()`

proxyServer

Syntax

```
proxyServer serverType, "ipAddress", portNum
proxyServer()
```

Beschreibung

Dieser Befehl legt die Werte für einen FTP- oder HTTP-Proxyserver fest.

Ohne Parameter gibt `proxyServer()` die Einstellungen eines FTP- oder HTTP-Proxyservers zurück.

Parameter

serverType Optional. Ein Symbol, das den Typ des Proxyservers angibt. Der Wert kann entweder `#ftp` oder `#http` sein.

ipAddress Optional. Eine Zeichenfolge, die die IP-Adresse angibt.

portNum Optional. Eine Ganzzahl, die die Anschlussnummer angibt.

Beispiel

Die folgende Anweisung konfiguriert einen HTTP-Proxyserver mit der IP-Adresse 197.65.208.157, und zwar unter Verwendung von Anschluss 5:

```
-- Lingo
proxyServer #http,"197.65.208.157",5
// Javascript
proxyServer (symbol("http"),"197.65.208.157",5) ;
```

Die folgende Anweisung gibt die Anschlussnummer eines HTTP-Proxyservers zurück:

```
-- Lingo
put proxyServer(#http,#port)

// Javascript
put (proxyServer(symbol("http"),symbol("port"))) ;
```

Wenn kein Servertyp angegeben ist, gibt diese Funktion den Wert 1 zurück.

Die folgende Anweisung gibt die IP-Adresse eines HTTP-Proxyservers als String zurück:

```
-- Lingo
put proxyServer(#http)

// Javascript
put (proxyServer(symbol("http")) ;
```

Die folgende Anweisung deaktiviert einen FTP-Proxyserver:

```
proxyServer #ftp,#stop
// Javascript
proxyServer(symbol("ftp"),symbol("stop")) ;
```

ptToHotSpotID()

Syntax

```
-- Lingo syntax
spriteObjRef.ptToHotSpotID(point)

// JavaScript syntax
spriteObjRef.ptToHotSpotID(point);
```

Beschreibung

Diese QuickTime VR-Funktion gibt gegebenenfalls die ID des am angegebenen Punkt befindlichen Hotspot zurück. Falls kein Hotspot vorhanden ist, gibt die Funktion den Wert 0 zurück.

Parameter

point Erforderlich. Gibt den zu testenden Punkt an.

puppetPalette()

Syntax

```
-- Lingo syntax
_movie.puppetPalette(palette {, speed} {, frames})

// JavaScript syntax
_movie.puppetPalette(palette {, speed} {, frames});
```

Beschreibung

Diese Filmmethode bewirkt, dass sich der Palettenkanal wie ein Puppet verhält. Dadurch kann Skriptcode die Paletteneinstellungen im Palettenkanal des Drehbuchs außer Kraft setzen und dem Film Paletten zuordnen.

Die `puppetPalette()`-Methode stellt die aktuelle Palette auf den Palettendarsteller ein, der in *palette* angegeben ist. Falls *palette* zu einer Zeichenfolge ausgewertet wird, gibt diese den Namen der Besetzungsbibliothek der Palette an. Wenn *Palette* dagegen zu einer Ganzzahl ausgewertet wird, gibt diese die Darstellernummer der Palette an.

Sie erhalten in der Regel die besten Ergebnisse, wenn Sie die `puppetPalette()`-Methode verwenden, bevor Sie zu dem Bild navigieren, auf das sich der Befehl auswirken soll. Auf diese Weise hat Director Zeit, die gewünschte Palette zuzuordnen, bevor das nächste Bild gezeichnet wird.

Sie können die Palette auch einblenden, indem Sie *speed* durch eine Ganzzahl ersetzen. Sie können hierbei einen Wert zwischen 1 (am langsamsten) und 60 (am schnellsten) wählen. Es ist ebenfalls möglich, die Palette über mehrere Bilder hinweg einzublenden, indem Sie *Frames* durch eine Ganzzahl ersetzen, die die Anzahl der Bilder angibt.

Eine Puppet-Palette bleibt so lange aktiv, bis Sie sie mithilfe der Syntax `_movie.puppetPalette(0)` deaktivieren. Sobald die Puppet-Palette aktiv ist, werden keine weiteren Palettenänderungen im Drehbuch mehr vorgenommen.

Hinweis: *Der Browser steuert die Palette für die gesamte Webseite. Somit verwendet der Shockwave Player stets die Browserpalette.*

Parameter

palette Erforderlich. Eine Zeichenfolge oder Ganzzahl, die den Namen oder die Nummer der neuen Palette angibt.

speed Optional. Eine Ganzzahl, die die Überblendgeschwindigkeit angibt. Gültige Werte: 1 bis 60.

frames Optional. Eine Ganzzahl, die die Anzahl der Bilder angibt, innerhalb derer eine Überblendung durchgeführt wird.

Beispiel

Die folgende Anweisung macht "Regenbogen" zur Palette des Films:

```
-- Lingo syntax
_movie.puppetPalette("Rainbow")

// JavaScript syntax
_movie.puppetPalette("Rainbow");
```

Die folgende Anweisung macht „Rainbow“ zur Palette des Films. Der Übergang zur Palette „Rainbow“ erfolgt über eine Zeiteinstellung von 15 sowie innerhalb von 20 Bildern.

```
-- Lingo syntax
_movie.puppetPalette("Rainbow", 15, 20)

// JavaScript syntax
_movie.puppetPalette("Rainbow", 15, 20);
```

Siehe auch

Movie

puppetSprite()

Syntax

```
-- Lingo syntax
_movie.puppetSprite(intSpriteNum, bool)

// JavaScript syntax
_movie.puppetSprite(intSpriteNum, bool);
```

Beschreibung

Diese Filmmethode bestimmt, ob es sich bei einem Sprite-Kanal um ein skriptgesteuertes Puppet (`TRUE`) oder um kein Puppet (`FALSE`) handelt. Im letzteren Fall wird der Kanal durch das Drehbuch gesteuert.

Während sich der Abspielkopf im Sprite befindet, können Sie die Puppets des Sprite-Kanals mit der Syntax `puppetSprite(`*intSpriteNum*`, FALSE)` ausschalten. Dadurch werden die Sprite-Eigenschaften auf die im Drehbuch angegebenen zurückgesetzt.

Die Anfangseigenschaften des Sprite-Kanals entsprechen den Kanaleinstellungen zum Zeitpunkt der Ausführung der `puppetSprite()`-Methode. Die Sprite-Eigenschaften können jedoch mit Skriptcode wie folgt geändert werden:

- Wenn der Sprite-Kanal ein Puppet ist, bleiben alle durch Skriptcode an den Eigenschaften des Sprite-Kanals vorgenommenen Änderungen weiterhin aktiv, nachdem der Abspielkopf das Sprite verlässt.
- Wenn der Sprite-Kanal jedoch kein Puppet ist, sind alle durch Skriptcode am Sprite vorgenommenen Änderungen nur für die Dauer des aktuellen Sprites gültig.

Sie können die `puppetSprite()`-Methode nur dann verwenden, wenn der Kanal auch ein Sprite enthält.

Wenn Sie den Sprite-Kanal als Puppet verwenden, können Sie viele Sprite-Eigenschaften, wie z. B. `member`, `locH` und `width`, mit Skriptcode steuern, sobald der Abspielkopf das Sprite verlässt.

Sie sollten die Steuerung mit der Syntax `puppetSprite(`*intSpriteNum*`, FALSE)` sofort an das Drehbuch zurückgeben, wenn der Sprite-Kanal nicht mehr durch Skriptcode gesteuert wird. Andernfalls kann es zu unerwarteten Ergebnissen kommen, wenn sich der Abspielkopf in Bildern befindet, die nicht als Puppets gedacht sind.

Hinweis: *In Version 6 von Director wurde die automatische Puppet-Generierung eingeführt, wodurch es in den meisten Fällen nicht mehr notwendig ist, ein Sprite explizit zum Puppet zu machen. Die explizite Steuerung ist aber weiterhin von Vorteil, wenn Sie auch nach dem Abspielen eines Sprite-Einschlusses noch vollständige Kontrolle über den Kanalinhalt behalten wollen.*

Parameter

intSpriteNum Erforderlich. Eine Ganzzahl, die den zu testenden Sprite-Kanal angibt.

bool Erforderlich. Ein Boolescher Wert, der angibt, ob ein Sprite-Kanal skriptgesteuert (`TRUE`) oder durch das Drehbuch gesteuert (`FALSE`) ist.

Beispiel

Die folgende Anweisung macht aus dem Sprite in Kanal 15 ein Puppet:

```
-- Lingo syntax
_movie.puppetSprite(15, TRUE)

// JavaScript syntax
_movie.puppetSprite(15, true);
```

Die folgende Anweisung deaktiviert die Puppet-Bedingung des Sprites in Kanal i + 1:

```
-- Lingo syntax
_movie.puppetSprite(i + 1, FALSE)

// JavaScript syntax
_movie.puppetSprite(i + 1, false);
```

Siehe auch

`makeScriptedSprite()`, `Movie`, `Sprite Channel`

puppetTempo()

Syntax

```
-- Lingo syntax
_movie.puppetTempo(intTempo)

// JavaScript syntax
_movie.puppetTempo(intTempo);
```

Beschreibung

Diese Filmmethode bewirkt, dass sich der Tempokanal wie ein Puppet verhält und das Tempo auf eine angegebene Anzahl von Bildern eingestellt wird.

Wenn es sich beim Tempokanal um ein Puppet handelt, kann Skriptcode die Tempoeinstellung im Drehbuch außer Kraft setzen und die dem Film zugeordnete Geschwindigkeit ändern.

Sie brauchen die Puppet-Tempobedingung nicht extra auszuschalten, um nachfolgende Tempoänderungen im Drehbuch in Kraft treten zu lassen.

Hinweis: *Theoretisch wäre es zwar möglich, mit der* `puppetTempo()` *-Methode Bildraten bis zu 30.000 Bilder pro Sekunde (BpS) zu erzielen, doch dies ist nur bei geringer Animation und einem extrem leistungsfähigen Computer realisierbar.*

Parameter

intTempo Erforderlich. Eine Ganzzahl, die die Geschwindigkeit angibt.

Beispiel

Die folgende Anweisung stellt die Filmgeschwindigkeit auf 30 Bilder pro Sekunde ein:

```
-- Lingo syntax
_movie.puppetTempo(30)

// JavaScript syntax
_movie.puppetTempo(30);
```

Die folgende Anweisung erhöht die bisherige Filmgeschwindigkeit um 10 Bilder pro Sekunde:

```
-- Lingo syntax
_movie.puppetTempo(oldTempo + 10)

// JavaScript syntax
_movie.puppetTempo(oldTempo + 10);
```

Siehe auch

Movie

puppetTransition()

Syntax

```
-- Lingo syntax
_movie.puppetTransition(memberObjRef)
_movie.puppetTransition(int {, time} {, size} {, area})

// JavaScript syntax
_movie.puppetTransition(memberObjRef);
_movie.puppetTransition(int {, time} {, size} {, area});
```

Beschreibung

Diese Filmmethode führt den angegebenen Übergang vom aktuellen zum nächsten Bild aus.

Um einen Xtra-Übergangsdarsteller zu verwenden, setzen Sie die Syntax `puppetTransition(`*memberObjRef*`)` ein.

Sie können auch einen integrierten Director-Übergang verwenden, indem Sie *int* durch einen Wert aus der folgenden Tabelle ersetzen. Ersetzen Sie *time* durch die Anzahl von Viertelsekunden, in denen der Übergang vorgenommen werden soll. Der Mindestwert ist 0, der Höchstwert 120 (30 Sekunden). Ersetzen Sie *size* durch die Anzahl der Pixel, die in jedem Übergangs-Chunk vorhanden sein sollen. Der Mindestwert ist 1, der Höchstwert 128. Kleinere Chunk-Größen ergeben weichere Übergänge, laufen jedoch langsamer ab.

Code	Übergang	Code	Übergang
01	Nach rechts wischen	27	Reihen, nach Zufallsprinzip
02	Nach links wischen	28	Spalten, nach Zufallsprinzip
03	Nach unten wischen	29	Nach unten überdecken
04	Nach oben wischen	30	Nach links unten überdecken
05	Von Bildmitte auswärts, horizontal	31	Nach rechts unten überdecken
06	Bildränder einwärts, horizontal	32	Nach links überdecken
07	Von Bildmitte auswärts, vertikal	33	Nach rechts überdecken
08	Bildränder einwärts, vertikal	34	Nach oben überdecken
09	Von Bildmitte auswärts, quadratisch	35	Nach links oben überdecken
10	Bildränder einwärts, quadratisch	36	Nach rechts oben überdecken
11	Nach links schieben	37	Jalousie
12	Nach rechts schieben	38	Schachbrett
13	Nach unten schieben	39	Streifen unten, Zuwachs links
14	Nach oben schieben	40	Streifen unten, Zuwachs rechts
15	Nach oben aufdecken	41	Streifen links, Zuwachs unten
16	Nach rechts oben aufdecken	42	Streifen links, Zuwachs oben

Code	Übergang	Code	Übergang
17	Nach rechts aufdecken	43	Streifen rechts, Zuwachs unten
18	Nach rechts unten aufdecken	44	Streifen rechts, Zuwachs oben
19	Nach unten aufdecken	45	Streifen oben, Zuwachs links
20	Nach links unten aufdecken	46	Streifen oben, Zuwachs rechts
21	Nach links aufdecken	47	Zoom-Öffnen
22	Nach links oben aufdecken	48	Zoom-Schließen
23	Pixelweise auflösen, schnell*	49	Vertikale Jalousie
24	Auflösen, Rechteck	50	Bitweise auflösen, schnell*
25	Auflösen, Quadrat	51	Pixelweise auflösen*
26	Auflösen, Muster	52	Bitweise auflösen*

Die mit einem Stern (*) markierten Übergänge funktionieren nicht bei Monitoren, die auf 32 Bit eingestellt sind.

Es besteht kein direkter Zusammenhang zwischen einem niedrigen Zeitwert und einem schnellen Übergang. Die tatsächliche Übergangsgeschwindigkeit hängt vielmehr vom Verhältnis zwischen *size* und *time* ab. Bei einer *size* von einem Pixel dauert der Übergang z. B. länger, ganz gleich wie klein der Zeitwert ist, da der Computer intensive Berechnungen durchführen muss. Um die Übergänge zu beschleunigen, sollten Sie daher mit größeren Chunks anstatt kürzeren Zeiten arbeiten.

Ersetzen Sie *area* durch einen Wert, der bestimmt, ob der Übergang nur im Änderungsbereich (`TRUE`) oder auf der ganzen Bühne (`FALSE`, Standard) erfolgen soll. Bei der Variablen *area* handelt es sich um einen Bereich, in dem sich die Sprites geändert haben.

Parameter

memberObjRef Erforderlich bei Verwendung eines Xtra-Übergangsdarstellers. Ein Verweis auf den als Übergang zu verwendenden Xtra-Darsteller.

int Erforderlich bei Verwendung eines integrierten Director-Übergangs. Eine Ganzzahl, die die Anzahl der zu verwendenden Übergänge angibt.

time Optional. Eine Ganzzahl, die die Anzahl von Viertelsekunden angibt, in denen der Übergang vorgenommen werden soll. Gültige Werte: 0 bis 120.

size Optional. Eine Ganzzahl, die die Anzahl der Pixel angibt, die in jedem Übergangs-Chunk vorhanden sein sollen. Gültige Werte: 1 bis 128.

area Optional. Ein Boolescher Wert, der bestimmt, ob der Übergang nur im Änderungsbereich (`TRUE`) oder auf der ganzen Bühne (`FALSE`) erfolgen soll.

Beispiel

Die folgende Anweisung führt den Übergang "Nach rechts wischen" aus. Da für *area* kein Wert angegeben ist, erfolgt der Übergang auf der gesamten Bühne (Standardeinstellung).

```
-- Lingo syntax
_movie.puppetTransition(1)

// JavaScript syntax
_movie.puppetTransition(1);
```

Die folgende Anweisung führt den Übergang „Nach links wischen“ aus, der 1 Sekunde dauert, die Chunk-Größe 20 verwendet und sich über die ganze Bühne erstreckt:

```
-- Lingo syntax
_movie.puppetTransition(2, 4, 20, FALSE)

// JavaScript syntax
_movie.puppetTransition(2, 4, 20, false);
```

Siehe auch

Movie

put()

Syntax

```
-- Lingo syntax
put(value)

// JavaScript syntax
put(value);
```

Beschreibung

Diese Top-Level-Funktion wertet einen Ausdruck aus und zeigt das Ergebnis im Nachrichtenfenster an.

Die Funktionalität dieser Methode ist mit der der Top-Level-Methode `trace()`, die in Lingo als auch in JavaScript-Syntax verfügbar ist, identisch.

Diese Methode kann auch als Debugging-Werkzeug verwendet werden, um bei der Wiedergabe des Films die Werte von Variablen zu verfolgen.

Parameter

value Erforderlich. Der auszuwertende Ausdruck.

Beispiel

Die folgende Anweisung zeigt im Nachrichtenfenster die Zeit an:

```
-- Lingo syntax
put(_system.time())

// JavaScript syntax
put(_system.time());
```

Die folgende Anweisung zeigt im Nachrichtenfenster den der Variablen bid zugeordneten Wert an:

```
-- Lingo syntax
bid = "Johnson"
put(bid) -- "Johnson"

// JavaScript syntax
var bid = "Johnson";
put(bid); // Johnson
```

Siehe auch

trace()

qtRegisterAccessKey()

Syntax

```
-- Lingo syntax
qtRegisterAccessKey(categoryString, keyString)

// JavaScript syntax
qtRegisterAccessKey(categoryString, keyString);
```

Beschreibung

Dieser Befehl ermöglicht die Registrierung eines Schlüssels für verschlüsselte QuickTime-Medien.

Es handelt sich hier um einen Schlüssel der Anwendungs- und nicht der Systemebene. Sobald die Anwendung die Registrierung des Schlüssels aufhebt oder beendet wird, kann auf die Medien nicht mehr zugegriffen werden.

Hinweis: *Aus Sicherheitsgründen kann keine Auflistung der registrierten Schlüssel angezeigt werden.*

Siehe auch

qtUnRegisterAccessKey()

qtUnRegisterAccessKey()

Syntax

```
-- Lingo syntax
qtUnRegisterAccessKey(categoryString, keyString)

// JavaScript syntax
qtUnRegisterAccessKey(categoryString, keyString);
```

Beschreibung

Mit diesem Befehl kann die Registrierung des Schlüssels für verschlüsselte QuickTime-Medien aufgehoben werden.

Es handelt sich hier um einen Schlüssel der Anwendungs- und nicht der Systemebene. Nach Aufhebung der Schlüsselregistrierung werden nur mit diesem Schlüssel verschlüsselte Filme weiter abgespielt. Auf andere Medien kann nicht mehr zugegriffen werden.

Siehe auch

qtRegisterAccessKey()

queue()

Syntax

```
-- Lingo syntax
soundChannelObjRef.queue(memberObjRef)
soundChannelObjRef.queue(propList)

// JavaScript syntax
soundChannelObjRef.queue(memberObjRef);
soundChannelObjRef.queue(propList);
```

Beschreibung

Diese Soundkanalmethode fügt einen Sounddarsteller zur Warteschlange eines Soundkanals hinzu.

Nachdem ein Sound in die Warteschlange eingereiht ist, kann er sofort mit der `play()`-Methode abgespielt werden, da Director einen bestimmten Teil eines in der Warteschlange befindlichen Sounds vorauslädt, um Verzögerungen zwischen der `play()`-Methode und dem Wiedergabebeginn zu verhindern. Standardmäßig werden 1500 Millisekunden eines Sounds vorausgeladen. Zur Änderung dieses Parameters können Sie mit der `queue()`-Methode eine Eigenschaftsliste mit einem oder mehreren Parametern übergeben. Diese Parameter können auch mit der `setPlayList()`-Methode übergeben werden.

Ein Beispiel für `queue()` in einem fertigen Film finden Sie im Film „Sound Control“ im Ordner „Learning\\Lingo_“ im Director-Anwendungsordner.

Parameter

memberObjRef Erforderlich bei Angabe eines Sounddarstellers. Ein Verweis auf den in die Warteschlange zu stellenden Sounddarsteller.

propList Erforderlich beim Übergeben einer Eigenschaftsliste als Parameter. Eine Eigenschaftsliste, die für den in die Warteschlange zu stellenden Sounddarsteller gültig ist. Dies ist bei den folgenden Eigenschaften der Fall:

Eigenschaft	Beschreibung
`#member`	Der in die Warteschlange einzureihende Sounddarsteller. Diese Eigenschaft ist erforderlich; alle anderen sind optional.
`#startTime`	Der Zeitpunkt im Sound, zu dem die Wiedergabe beginnt, in Millisekunden. Standardmäßig ist dies der Anfangspunkt des Sounds. Weitere Informationen finden Sie unter `startTime`.
`#endTime`	Der Zeitpunkt im Sound, zu dem die Wiedergabe endet, in Millisekunden. Standardmäßig ist dies das Ende des Sounds. Weitere Informationen finden Sie unter`endTime`.
`#loopCount`	Angabe, wie oft eine Schleife abgespielt werden soll, die in `#loopStartTime` und `#loopEndTime` definiert ist. DerStandardwert ist1. Weitere Informationen finden Sie unter `loopCount`.
`#loopStartTime`	Der Zeitpunkt im Sound, zu dem die Schleifenwiedergabe beginnt, in Millisekunden. Weitere Informationen finden Sie unter `loopStartTime`.
`#loopEndTime`	Der Zeitpunkt im Sound, zu dem die Schleifenwiedergabe endet, in Millisekunden. Weitere Informationen finden Sie unter `loopEndTime`.
`#preloadTime`	Angabe, wie viel Sound vor Abspielbeginn in den Zwischenspeicher geladen werden soll, in Millisekunden. Weitere Informationen finden Sie unter `preloadTime`.

Beispiel

Durch die folgende Prozedur werden zwei Sounds in die Warteschlange gestellt und abgespielt. Der erste Sound (Darsteller `Chimes`) wird ganz abgespielt. Der zweite Sound (Darsteller `introMusic`) wird wie folgt abgespielt: Die Wiedergabe beginnt 3 Sekunden nach Anfang des Sounds, der Abschnitt zwischen 8 und 8,9 Sekunden wird 5 Mal wiederholt, und die Wiedergabe endet 10 Sekunden nach Anfang des Sounds.

```
-- Lingo syntax
on playMusic
    sound(2).queue(member("Chimes"))
    sound(2).queue([#member:member("introMusic"), #startTime:3000, #endTime:10000,
#loopCount:5, #loopStartTime:8000, #loopEndTime:8900])
    sound(2).play()
end playMusic

// JavaScript syntax
function playMusic() {
    sound(2).queue(member("Chimes"))
    sound(2).queue(propList("member",member("introMusic"), "startTime",3000, "endTime",10000,
"loopCount",5, "loopStartTime",8000, "loopEndTime",8900));
    sound(2).play();
}
```

Siehe auch

endTime (Soundkanal), loopCount, loopEndTime (Soundkanal), loopStartTime, pause() (Soundkanal), play() (Soundkanal)preLoadTime, setPlayList(), Sound Channel, startTime (Sound Channel), stop() (Soundkanal)

queue() (3D)

Syntax

```
member(whichCastmember).model(whichModel).bonesPlayer.queue(motionName {, looped, startTime,
endTime, scale, offset})
member(whichCastmember).model(whichModel).keyframePlayer.queue(motionName {, looped,
startTime, endTime, scale, offset})
```

Beschreibung

Dieser 3D-Befehl für die Modifizierer `keyframePlayer` und `bonesPlayer` fügt eine angegebene Bewegung am Ende der Eigenschaft `playList` des Modifizierers ein. Die Bewegung wird vom Modell ausgeführt, wenn die Wiedergabe aller Bewegungen, die in der Abspielliste vor ihr aufgeführt sind, beendet ist.

Parameter

motionName Erforderlich. Gibt den Namen der hinzuzufügenden Bewegung an.

looped Optional. Gibt an, ob die Bewegung einmal (`FALSE`) oder kontinuierlich (`TRUE`) abgespielt wird.

startTime Optional. Wird in Millisekunden ab Bewegungsbeginn gemessen. Wenn *looped* auf `FALSE` gesetzt ist, beginnt die Bewegung bei *offset* und endet bei *endTime*. Wenn *looped* auf `TRUE` gesetzt ist, beginnt die erste Iteration der Schleife bei *offset* und endet bei *endTime*. Alle nachfolgenden Wiederholungen beginnen bei *startTime* und enden bei *endTime*.

endTime Optional. Wird in Millisekunden ab Bewegungsbeginn gemessen. Wenn *looped* auf `FALSE` gesetzt ist, beginnt die Bewegung bei *offset* und endet bei *endTime*. Wenn *looped* auf `TRUE` gesetzt ist, beginnt die erste Iteration der Schleife bei *offset* und endet bei *endTime*. Alle nachfolgenden Wiederholungen beginnen bei *cropStart* und enden bei *endTime*. Setzen Sie *endTime* auf -1, wenn die Bewegung bis zum Ende abgespielt werden soll.

scale Optional. Gibt die tatsächliche Wiedergabegeschwindigkeit der Bewegung an. *scale* wird mit der Eigenschaft `playRate` des Modifizierers `#keyframePlayer` bzw. `#bonesPlayer` des Models multipliziert, um die tatsächliche Wiedergabegeschwindigkeit der Bewegung zu ermitteln.

offset Optional. Wird in Millisekunden ab Bewegungsbeginn gemessen. Wenn *looped* auf `FALSE` gesetzt ist, beginnt die Bewegung bei *offset* und endet bei *endTime*. Wenn *looped* auf `TRUE` gesetzt ist, beginnt die erste Iteration der Schleife bei *offset* und endet bei *endTime*. Alle nachfolgenden Wiederholungen beginnen bei *startTime* und enden bei *endTime*.

Beispiel

Der folgende Befehl fügt die Bewegung "Fallen" am Ende der `bonesPlayer`-Abspielliste des Modells `Walker` hinzu. Wenn alle in der Liste vor `Fall` aufgeführten Bewegungen ausgeführt sind, wird `Fall` einmal von Anfang bis Ende abgespielt.

```
sprite(1).member.model("Walker").bonesPlayer.queue("Fall", 0, 0, -1, 1, 0)
```

Der folgende Befehl fügt die Bewegung `Kick` am Ende der `bonesPlayer`-Abspielliste des Modells `Walker` hinzu. Wenn alle in der Liste vor `Kick`aufgeführten Bewegungen ausgeführt sind, wird ein Abschnitt von `Kick` in einer kontinuierlichen Schleife abgespielt. Die erste Iteration der Schleife beginnt 2000 Millisekunden nach Bewegungsanfang. Alle nachfolgenden Iterationen der Schleife beginnen 1000 Millisekunden nach Anfang von `Kick` und enden 5000 Millisekunden nach Anfang von `Kick`. Die Wiedergabegeschwindigkeit ist das Dreifache der Eigenschaft `playRate` des Modellmodifizierers `bonesPlayer`.

```
sprite(1).member.model("Walker").bonesPlayer.queue("Kick", 1, 1000, 5000, 3, 2000)
```

Siehe auch

`play() (3D)`, `playNext() (3D)`, `playRate (3D)`

QuickTimeVersion()

Syntax

```
-- Lingo syntax
QuickTimeVersion()

// JavaScript syntax
QuickTimeVersion();
```

Beschreibung

Diese Funktion gibt einen Fließkommawert zurück, der die derzeit installierte QuickTime-Version angibt. Diese Funktion ersetzt die aktuelle Funktion `QuickTimePresent`.

Wenn unter Windows mehrere Versionen von QuickTime 3.0 oder später installiert sind, gibt `QuickTimeVersion()` die neueste Versionsnummer zurück. Wenn dagegen eine QuickTime-Version vor 3.0 installiert ist, gibt `QuickTimeVersion()` automatisch die Versionsnummer 2.1.2 zurück, ganz gleich, welche Version installiert ist.

Parameter
Keiner

Beispiel
Die folgende Anweisung verwendet `QuickTimeVersion()`, um die gegenwärtig installierte Version von QuickTime im Nachrichtenfenster anzuzeigen:

```
-- Lingo syntax
put(QuickTimeVersion())

// JavaScript syntax
put(QuickTimeVersion());
```

quit()

Syntax

```
-- Lingo syntax
_player.quit()

// JavaScript syntax
_player.quit();
```

Beschreibung
Diese Player-Methode beendet Director oder einen Projektor und kehrt zum Windows-Desktop bzw. zum Mac-Finder zurück.

Parameter
Keiner

Beispiel
Bei der folgenden Anweisung beendet der Computer Director und kehrt zum Windows-Desktop bzw. zum Mac-Finder zurück, sobald der Benutzer Strg+Q (Windows) bzw. Befehl+Q (Mac) drückt:

```
-- Lingo syntax
if (_key.key = "q" and _key.commandDown) then
    _player.quit()
end if

// JavaScript syntax
if (_key.key == "q" && _key.commandDown) {
    _player.quit();
}
```

Siehe auch
Player

ramNeeded()

Syntax

```
-- Lingo syntax
_movie.ramNeeded(intFromFrame, intToFrame)

// JavaScript syntax
_movie.ramNeeded(intFromFrame, intToFrame);
```

Beschreibung

Diese Filmmethode bestimmt den zur Anzeige eines Bildbereichs benötigten Speicherplatz in Bytes. Beispielsweise lässt sich die Größe von Bildern mit 32-Bit-Motiven prüfen: wenn `ramNeeded()` größer ist als `freeBytes()`, sollten Sie zu Bildern mit 8-Bit-Motiven übergehen und die erhaltene Größe durch 1024 teilen, um Byte in Kilobyte (K) zu konvertieren.

Parameter

intFromFrame Erforderlich. Eine Ganzzahl, die die Nummer des ersten Bildes im Bereich angibt.

intToFrame Erforderlich. Eine Ganzzahl, die die Nummer des letzten Bildes im Bereich angibt.

Beispiel

Die folgende Anweisung setzt die Variable `frameSize` auf die Anzahl von Bytes, die zur Anzeige der Bilder 100 bis 125 benötigt werden:

```
-- Lingo syntax
frameSize = _movie.ramNeeded(100, 125)

// JavaScript syntax
var frameSize = _movie.ramNeeded(100, 125);
```

Die folgende Anweisung prüft, ob der zur Anzeige der Bilder 100 bis 125 benötigte Speicherplatz den verfügbaren Speicherplatz übersteigt. Wenn ja, geht die Anweisung zu einem Abschnitt mit Darstellern über, die eine geringere Farbtiefe aufweisen.

```
-- Lingo syntax
if (_movie.ramNeeded(100, 125) > _system.freeBytes) then
    _movie.go("8-bit")
end if

// JavaScript syntax
if (_movie.ramNeeded(100, 125) > _system.freeBytes) {
    _movie.go("8-bit");
}
```

Siehe auch

freeBytes(), Movie

random()

Syntax

```
-- Lingo syntax
random(integerExpression)

// JavaScript syntax
random(integerExpression);
```

Beschreibung

Diese Top-Level-Funktion gibt eine zufällig ausgewählte Ganzzahl zurück, die zwischen 1 und einem angegebenen Wert liegt. Mit dieser Funktion können Werte in einem Film geändert werden, z. B. um den Pfad in einem Spiel zu variieren, Zufallszahlen zuzuordnen oder die Farbe und Position von Sprites zu ändern.

Falls Sie eine Reihe möglicher Zufallszahlen festlegen möchten, die nicht mit 1 beginnen sollen, müssen Sie die entsprechende Summe von der Funktion `random()` subtrahieren. Der Ausdruck `random(n + 1) - 1` verwendet beispielsweise den Bereich 0 bis `n`.

Parameter

integerExpression Erforderlich. Gibt den Höchstwert der Zufallszahl an.

Beispiel

Die folgende Anweisung ordnet der Variablen `diceRoll` Zufallswerte zu:

```
-- Lingo syntax
diceRoll = (random(6) + random(6))

// JavaScript syntax
var diceRoll = (random(6) + random(6));
```

Die folgende Anweisung ändert die Vordergrundfarbe von Sprite 10 nach dem Zufallsprinzip:

```
-- Lingo syntax
sprite(10).foreColor = (random(256) - 1)

// JavaScript syntax
sprite(10).foreColor = (random(256) - 1);
```

Die folgende Prozedur bestimmt nach dem Zufallsprinzip, welcher von zwei Filmabschnitten abgespielt wird:

```
-- Lingo syntax
on SelectScene
    if (random(2) = 2) then
        _movie.go("11a")
    else
        _movie.go("11b")
    end if
end

// JavaScript syntax
function SelectScene() {
    if (random(2) == 1) {
        _movie.go("11a");
    } else {
        _movie.go("11b");
    }
}
```

Die folgende Anweisung generiert nach dem Zufallsprinzip ein Vielfaches von 5 im Bereich 5 bis 100:

```
-- Lingo syntax
theScore = (5 * random(20))

// JavaScript syntax
var theScore = (5 * random(20));
```

randomVector()

Syntax

```
-- Lingo syntax
randomVector()

// JavaScript syntax
randomVector();
```

Beschreibung

Diese Top-Level-Funktion gibt einen Einheitsvektor zurück, der einen zufällig gewählten Punkt auf der Oberfläche einer Einheitskugel beschreibt.

Diese Funktion unterscheidet sich von `vector(random(10)/10.0, random(10)/10.0, random(10)/10.0,)` darin, dass der resultierende Vektor, der `randomVector()` verwendet, garantiert ein Einheitsvektor ist.

Ein Einheitsvektor weist eine Länge von 1 auf.

Parameter

Keiner

Beispiel

Die folgenden Anweisungen erstellen zwei willkürlich definierte Einheitsvektoren und zeigen sie im Nachrichtenfenster an:

```
-- Lingo syntax
vec1 = randomVector()
vec2 = randomVector()
put(vec1 & RETURN & vec2)

// JavaScript syntax
var vec1 = randomVector();
var vec2 = randomVector();
put(vec1 + "\n" + vec2);
```

Siehe auch

vector()

randomVector

Syntax

```
randomVector()
```

Beschreibung

Dieser 3D-Befehl gibt einen Einheitsvektor zurück, der einen willkürlich gewählten Punkt auf der Oberfläche einer Einheitskugel beschreibt. Diese Methode unterscheidet sich von `vector( random(10)/10.0,random(10)/10.0,random(10)/10.0)` insofern, als der entstehende Vektor garantiert ein Einheitsvektor ist.

Parameter

Keiner

Beispiel

Die folgenden Anweisungen erstellen zwei willkürlich definierte Einheitsvektoren und zeigen sie im Nachrichtenfenster an:

```
vec = randomVector()
put vec
-- vector(-0.1155, 0.9833, -0.1408)
vec2 = randomVector()
put vec2
-- vector(0.0042, 0.8767, 0.4810)
```

Siehe auch

getNormalized, generateNormals(), normalize

rawNew()

Syntax

```
parentScript.rawNew()
rawNew(parentScript)
```

Beschreibung
Diese Funktion erstellt ein Child-Objekt aus einem Parent-Skript, ohne die zugehörige `on new`-Prozedur aufzurufen. Auf diese Weise kann ein Film Child-Objekte erstellen, ohne ihre Eigenschaften initialisieren zu müssen. Dies ist besonders dann nützlich, wenn Sie eine große Anzahl von Child-Objekten zur späteren Verwendung erstellen möchten. Um die Eigenschaften eines dieser Child-Objekte zu initialisieren, rufen Sie die zugehörige `on new`-Prozedur auf.

Parameter
Keiner

Beispiel
Die folgende Anweisung erstellt ein Child-Objekt namens `RedCar` aus dem Parent-Skript `CarParentScript`, ohne dessen Eigenschaften zu initialisieren:

```
RedCar = script("CarParentScript").rawNew()
```

Die folgende Anweisung initialisiert die Eigenschaften des Child-Objekts `RedCar`:

```
RedCar.new()
```

Siehe auch
`new()`, `script()`

readBoolean

Syntax
```
byteArrayObject.readBoolean()
```

Beschreibung
Diese Bytearray-Methode liest einen Boole'schen Wert aus einem Bytearray. Boole'sche Werte belegen jeweils ein Byte. Die Methode gibt „False“ an, wenn der Bytewert Null ist; ansonsten „True“.

Beispiele
```
--Lingo syntax
put bArray.readBoolean()

//JavaScript syntax
put(bArray.readBoolean());
```

readByteArray

Syntax
```
readByteArray(intSize, [ByteArray], [intOffset])
```

Beschreibung
Diese Bytearray-Methode kopiert ein Bytearray teilweise in ein anderes Bytearray. Falls ein bestehendes Byte-Array als Eingabequelle übergeben wird, wird dieses ab Position `intOffset` an das Zielarray angehängt.

Parameter

Parameter	Beschreibung	Erforderlich/Optional
intSize	Länge des Byte-Array.	Erforderlich
ByteArray	Bestehendes Byte-Array, dass dem Zielarray angehängt wird.	Optional
intOffset	Anfangsposition.	Optional

Beispiele

```
--Lingo syntax
bArray.readByteArray(10)

//JavaScript syntax
bArray.readByteArray(10);
```

readByteArray (FileIO Xtra)

Syntax

```
FileIO.readByteArray(intSize,[byteArray],[intOffset])
```

Beschreibung

Diese FileIO Xtra-Methode liest die Anzahl der durch `intBytes` festgelegten Bytes. Falls bereits ein Bytearray vorhanden ist, werden die festgelegten `intBytes` ab Position `intOffset` an das Bytearray angehängt.

Parameter

Parameter	Beschreibung	Erforderlich/Optional
intSize	Länge des Byte-Array.	Erforderlich
ByteArray	Bestehendes Byte-Array, aus dem die Bytes ausgelesen werden.	Optional
intOffset	Anfangsposition.	Optional

Beispiele

```
--Lingo syntax
FileIO.readByteArray(10,bArray,1)

//JavaScript syntax
FileIO.readByteArray(10,bArray,1);
```

readChar()

Syntax

```
-- Lingo syntax
fileioObjRef.readChar()

// JavaScript syntax
fileioObjRef.readChar();
```

Beschreibung

Diese Datei-E/A-Methode (Fileio) liest das nächste Zeichen einer Datei und gibt es zurück.

Sie müssen zuerst durch Aufrufen von `openFile()` eine Datei öffnen, bevor Sie mithilfe von `readChar()` ein Zeichen lesen können. Beim Lesen von Unicode-Dateien gibt `readchar()` nur das nächste Byte und nicht das Unicode-Zeichen zurück. Verwenden Sie zum Lesen von Unicode-Zeichen die Methode `readFile()`.

Parameter

Keiner

Beispiel

Die folgende Anweisung öffnet eine Datei „c:\xtra.txt“ mit Lese-/Schreibberechtigung und liest alle Zeichen in der Datei bis das Zeichen „e“ gefunden wird.

```
-- Lingo
objFileio = new xtra("fileio")
objFileio.openFile("c:\xtra.txt",0)
repeat while(numToChar(objFileio.readChar()) <> 'e')
 put numToChar(objFileio.readChar())
end repeat

// JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.openFile("c:\xtra.txt",0);
while(numToChar(objFileio.readChar()) != 'e')
{
trace(numToChar(objFileio.readChar()));
}
```

Siehe auch

Fileio, openFile()

readFile()

Syntax

```
-- Lingo syntax
fileioObjRef.readFile()

// JavaScript syntax
fileioObjRef.readFile();
```

Beschreibung
Diese Datei-E/A-Methode (Fileio) liest ab der aktuellen Position bis zum Ende einer angegebenen Datei und gibt das Ergebnis als Zeichenfolge zurück.

Sie müssen zuerst durch Aufrufen von `openFile()` eine Datei öffnen, bevor Sie mithilfe von `readFile()` eine Datei lesen können.

Beim Lesen von UTF-8- oder UTF-16-Dateien mit „readFile()“ wird der Anfang der Datei möglicherweise nicht korrekt eingelesen. Um einen korrekten Lesevorgang zu gewährleisten, müssen Sie das Zeichen „BOM“ im Skript abfangen. Für UTF-8-Dateien ist das BOM-Zeichen „EF BB BF“. Für UTF-16-Dateien ist das BOM-Zeichen „FF FE“.

Parameter
Keiner

Beispiel
Die folgende Anweisung öffnet eine Datei „c:\xtra.txt“ mit Lese-/Schreibberechtigung und liest den Inhalt der Datei.

```
-- Lingo
objFileio = new xtra("fileio")
objFileio.openFile("c:\xtra.txt",0)
contents =objFileio.readFile()
put contents

// JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.openFile("c:\xtra.txt",0);
var contents = objFileio.readFile();
trace(contents);
```

Siehe auch
Fileio, openFile()

readFloat32

Syntax
```
byteArrayObject.readFloat32()
```

Beschreibung
Diese Bytearray-Methode liest einen 32-Bit-Fließkommawert aus einem Bytearray.

Beispiele
```
--Lingo syntax
put bArray.readFloat32()

//JavaScript syntax
put(bArray.readFloat32());
```

readFloat64

Syntax

```
byteArrayObject.readFloat64()
```

Beschreibung

Diese Bytearray-Methode liest einen 64-Bit-Fließkommawert aus einem Bytearray.

Beispiele

```
--Lingo syntax
put bArray.readFloat64()

//JavaScript syntax
put(bArray.readFloat64());
```

readInt8

Syntax

```
byteArrayObject.readInt8()
```

Beschreibung

Diese Bytearray-Methode liest einen vorzeichenbehafteten 8-Bit-Integerwert aus einem Bytearray.

Beispiele

```
--Lingo syntax
put bArray.readInt8()

//JavaScript syntax
put(bArray.readInt8());
```

readInt16

Syntax

```
byteArrayObject.readInt16()
```

Beschreibung

Diese Bytearray-Methode liest einen vorzeichenbehafteten 16-Bit-Integerwert aus einem Bytearray.

Beispiele

```
--Lingo syntax
put bArray.readInt16()

//JavaScript syntax
put(bArray.readInt16());
```

readInt32

Syntax

```
byteArrayObject.readInt32()
```

Beschreibung

Diese Bytearray-Methode liest einen vorzeichenbehafteten 32-Bit-Integerwert aus einem Bytearray.

Beispiele

```
--Lingo syntax
put bArray.readInt32()

//JavaScript syntax
put(bArray.readInt32());
```

readLine()

Syntax

```
-- Lingo syntax
fileioObjRef.readLine()

// JavaScript syntax
fileioObjRef.readLine();
```

Beschreibung

Diese Datei-E/A-Methode (Fileio) liest die nächste Zeile einer Datei, einschließlich des nächsten Zeilenwechsels (RETURN), und gibt diese als Zeichenfolge zurück.

Sie müssen zuerst durch Aufrufen von `openFile()` eine Datei öffnen, bevor Sie mithilfe von `readLine()` eine Zeile lesen können.

Weitere Informationen zu Dateien im UNIX-Format finden Sie unter „setNewLineConversion()“.

Parameter

Keiner

Beispiel

Die folgende Anweisung öffnet eine Datei „c:\xtra.txt“ mit Lese-/Schreibberechtigung und liest die erste Zeile der Datei.

```
-- Lingo
objFileio = new xtra("fileio")
objFileio.openFile("c:\xtra.txt",0)
contents = objFileio.readLine()
put contents

// JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.openFile("c:\xtra.txt",0);
contents = objFileio.readLine();
trace(contents);
```

Mit folgendem Code lässt sich ermitteln, ob zusätzliche Zeilen zum Lesen vorliegen:

```
-- Lingo:
if objFileio.getPosition() < objFileio.getLength() then -- There is at least one more
-- line available.
end if

// JavaScript syntax
if (objFileio.getPosition() < objFileio.getLength()) { // There is at least one more line
// available.
}
```

Siehe auch

`Fileio, openFile()`, setNewLineConversion()

readRawString

Syntax

```
readRawString(intLen, [strCharSet])
```

Beschreibung

Diese Bytearray-Methode liest eine feste Anzahl von Bytes als String.

Parameter

Parameter	Beschreibung	Standardwert
intLen	Länge des Strings, der vom Byte-Array gelesen werden soll.	Erforderlich
strCharSet	Legt die erforderliche Encoding-Übersetzung fest.	UTF-8

Beispiele

```
--Lingo syntax
put bArray.readRawString(10)

//JavaScript syntax
put(bArray.readRawString(10));
```

readString

Syntax

```
readString([strCharSet])
```

Beschreibung

Diese Bytearray-Methode liest ein Bytearray als String ein. Die ersten vier Bytes des String-Blob enthalten die Länge des Strings.

Parameter

Parameter	Beschreibung	Standardwert
strCharSet	Legt die erforderliche Encoding-Übersetzung fest.	UTF-8

Beispiele

```
--Lingo syntax
put bArray.readString()

//JavaScript syntax
put(bArray.readString());
```

readToken()

Syntax

```
-- Lingo syntax
fileioObjRef.readToken(stringSkip, stringBreak)

// JavaScript syntax
fileioObjRef.readToken(stringSkip, stringBreak);
```

Beschreibung

Diese Datei-E/A-Methode (Fileio) liest das nächste Token und gibt es als Zeichenfolge zurück.

Sie müssen zuerst durch Aufrufen von `openFile()` eine Datei öffnen, bevor Sie mithilfe von `readToken()` ein Token lesen können.

Parameter

stringSkip Erforderlich. Eine Zeichenfolge, die den Satz von Zeichen angibt, nach dem das Token beginnt. Die Zeichenfolge *stringSkip* ist in der zurückgegebenen Zeichenfolge nicht enthalten.

stringBreak Erforderlich. Eine Zeichenfolge, die den Satz von Zeichen angibt, vor dem das Token endet. Die Zeichenfolge *stringBreak* ist in der zurückgegebenen Zeichenfolge nicht enthalten.

Beispiel

Die folgende Anweisung öffnet eine Datei „c:\xtra.txt“ mit Lese-/Schreibberechtigung und liest das Token zwischen „Director ist gut“. Die zu überspringende Zeichenfolge ist “D” und `stringBreak` ist bei „g“.

```
-- Lingo
objFileio = new xtra("fileio")
objFileio.openFile("c:\xtra.txt",0)
contents = objFileio.readToken("D","g")
put contents

// JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.openFile("c:\xtra.txt",0);
contents = objFileio.readToken("D","g");
trace(contents);
```

Die Ausgabe ist „irector is“.

Siehe auch

`Fileio`, `openFile()`

readWord()

Syntax

```
-- Lingo syntax
fileioObjRef.readWord()

// JavaScript syntax
fileioObjRef.readWord();
```

Beschreibung

Diese Datei-E/A-Methode (Fileio) liest das nächste Wort einer Datei und gibt es als Zeichenfolge zurück.

Sie müssen zuerst durch Aufrufen von `openFile()` eine Datei öffnen, bevor Sie mithilfe von `readWord()` ein Wort lesen können.

Parameter

Keiner

Beispiel

Die folgende Anweisung öffnet eine Datei „c:\xtra.txt“ mit Lese-/Schreibberechtigung und liest das erste Wort der Datei.

```
-- Lingo
objFileio = new xtra("fileio")
objFileio.openFile("c:\xtra.txt",0)
contents =objFileio.readWord()
put contents

// JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.openFile("c:\xtra.txt",0);
contents = objFileio.readWord();
trace(contents);
```

Siehe auch

Fileio, openFile()

realPlayerNativeAudio()

Syntax

```
-- Lingo syntax
realPlayerNativeAudio()

// JavaScript syntax
realPlayerNativeAudio();
```

Beschreibung

Mit dieser RealMedia-Funktion können Sie ein globales Flag abrufen oder setzen, das bestimmt, ob der Audioabschnitt des RealMedia-Darstellers von RealPlayer verarbeitet wird (TRUE) oder von Director (FALSE). Diese Funktion gibt den vorherigen Wert des Flags zurück.

Dieses Flag hat nur dann eine Wirkung, wenn es vor dem Laden von RealPlayer gesetzt wird, d. h. bevor der erste RealMedia-Darsteller im Drehbuch bzw. der erste Lingo-Bezug auf einen RealMedia-Darsteller gefunden wird. Flag-Änderungen nach dem Laden von RealPlayer werden ignoriert. Dieses Flag sollte in einer prepareMovie-Ereignisprozedur in einem Filmskript ausgeführt werden. Dieses Flag wird für die gesamte Sitzung gesetzt (d. h. den Zeitraum zwischen Start und Schließen/Neustart des Shockwave Players), nicht nur für den aktuellen Film.

Standardmäßig wird dieses Flag auf FALSE gesetzt und Audio von Director verarbeitet, damit Sie den Audiostream eines RealMedia-Sprites durch Einstellen der Eigenschaft soundChannel und Verwenden der Standard-Lingo-Soundmethoden und -eigenschaften manipulieren können (wie z. B. RealAudio® mit anderen Director-Audiodaten mischen). Ist dieses Flag auf TRUE gesetzt, wird die Lingo-Steuerung des Soundkanals ignoriert und der Sound von RealPlayer verarbeitet.

Parameter

Keiner

Beispiel

Der folgende Code zeigt, dass die Funktion realPlayerNativeAudio() auf FALSE gesetzt ist, d.h. dass der Audioabschnitt des RealMedia-Darstellers von Director verarbeitet wird:

```
-- Lingo syntax
put(realPlayerNativeAudio())
-- 0

// JavaScript syntax
trace(realPlayerNativeAudio());
// 0
```

Der folgende Code setzt die Funktion realPlayerNativeAudio() auf TRUE, d.h. der Audioabschnitt des RealMedia-Streams wird von RealPlayer verarbeitet und die Lingo-Steuerung des Soundkanals ignoriert.

```
-- Lingo syntax
realPlayerNativeAudio(TRUE)

// JavaScript syntax
realPlayerNativeAudio(1);
```

Siehe auch

soundChannel (RealMedia)

realPlayerPromptToInstall()

Syntax

```
-- Lingo syntax
realPlayerPromptToInstall()

// JavaScript syntax
realPlayerPromptToInstall();
```

Beschreibung

Mit dieser RealMedia-Funktion können Sie ein globales Flag abrufen oder setzen, das bestimmt, ob die automatische Erkennung von RealPlayer 8 und ggf. die Anzeige einer entsprechenden Installationsaufforderung aktiviert ist (TRUE) oder nicht (FALSE).

Standardmäßig ist diese Funktion auf TRUE gesetzt. Wenn ein Benutzer, auf dessen Computer RealPlayer 8 nicht installiert ist, versucht, einen Film mit RealMedia zu laden, wird er automatisch gefragt, ob er RealPlayer von der RealNetworks®-Website herunterladen und installieren möchte. Sie können dieses Flag auf FALSE setzen, wenn Sie mit der Funktion realPlayerVersion() und eigenem Code selbst einen Erkennungs- und Aufforderungsmechanismus einrichten möchten. Wenn dieses Flag auf FALSE gesetzt und kein anderer Erkennungs- und Aufforderungsmechanismus für RealPlayer 8 vorhanden ist, können Benutzer ohne RealPlayer zwar Filme mit RealMedia-Darstellern laden, aber die RealMedia-Sprites erscheinen nicht.

Diese Funktion liest die Build-Nummer der auf dem Benutzersystem vorhandenen RealPlayer-Software, um festzustellen, ob RealPlayer 8 installiert ist. Auf Windows-Systemen bedeutet die Build-Nummer 6.0.8.132 oder höher, dass RealPlayer 8 installiert ist. Auf Mac-Systemen bedeutet die Build-Nummer 6.0.7.1001 oder höher (für die RealPlayer-Core-Komponente), dass RealPlayer 8 installiert ist.

Dieses Flag sollte in einer prepareMovie-Ereignisprozedur in einem Filmskript ausgeführt werden.

Diese Funktion gibt den vorherigen Wert des Flags zurück.

Parameter

Keiner

Beispiel

Der folgende Code zeigt, dass die Funktion realPlayerPromptToInstall() auf TRUE gesetzt ist, d.h. dass Benutzer, auf deren Computer die RealPlayer-Software noch nicht installiert ist, aufgefordert werden, sie herunterzuladen.

```
-- Lingo syntax
put(realPlayerPromptToInstall()) -- 1

// JavaScript syntax
trace(realPlayerPromptToInstall()); // 1
```

Der folgende Code setzt die Funktion realPlayerPromptToInstall() auf FALSE, d. h. Benutzer ohne RealPlayer-Software werden nur dann zur Installation aufgefordert, wenn Sie einen eigenen Erkennungs- und Aufforderungsmechanismus eingerichtet haben.

```
-- Lingo syntax
realPlayerPromptToInstall(FALSE)

// JavaScript syntax
realPlayerPromptToInstall(0);
```

realPlayerVersion()

Syntax

```
-- Lingo syntax
realPlayerVersion()

// JavaScript syntax
realPlayerVersion();
```

Beschreibung

Diese RealMedia-Funktion gibt einen String mit der Build-Nummer der auf dem Benutzersystem installierten RealPlayer-Software zurück bzw. einen leeren String, wenn RealPlayer nicht installiert ist. Zur Anzeige von Director-Filmen mit RealMedia-Inhalten ist RealPlayer 8 oder neuer erforderlich. Auf Windows-Systemen bedeutet die Build-Nummer 6.0.8.132 oder höher, dass RealPlayer 8 installiert ist. Auf Mac-Systemen bedeutet die Build-Nummer 6.0.7.1001 oder höher (für die RealPlayer-Core-Komponente), dass RealPlayer 8 installiert ist.

realPlayerPromptToInstall().

Sie müssen die folgenden Schritte durchführen, um mit der Funktion `realPlayerVersion()` einen eigenen Erkennungs- und Aufforderungsmechanismus einzurichten:

- Rufen Sie `realPlayerPromptToInstall(FALSE)` auf (diese Funktion ist standardmäßig auf `TRUE` gesetzt), bevor ein RealMedia-Darsteller in Lingo referenziert wird oder im Drehbuch erscheint. Diese Funktion sollte in einer `prepareMovie`-Ereignisprozedur in einem Filmskript eingestellt werden.
- Stellen Sie anhand der Systemeigenschaft `xtraList` sicher, dass das Xtra für RealMedia (RealMedia Asset.x32) im Dialogfeld „Film-Xtras“ enthalten ist. Die Funktion `realPlayerVersion()` kann nur verwendet werden, wenn das Xtra für RealMedia vorhanden ist.

Die von dieser Funktion zurückgegebene Build-Nummer ist die gleiche, die auch in RealPlayer angezeigt werden kann.

So zeigen Sie die RealPlayer-Build-Nummer unter Windows an:

1. Starten Sie RealPlayer.
2. Wählen Sie „Info über RealPlayer“ im Menü „Hilfe“.

Im daraufhin angezeigten Fenster erscheint die Build-Nummer in der zweiten Zeile oben auf dem Bildschirm.

So zeigen Sie die RealPlayer-Build-Nummer auf dem Mac an:

1. Starten Sie RealPlayer.
2. Wählen Sie „Info über RealPlayer“ im Apple®-Menü.

   Das Dialogfeld „Info“ über RealPlayer erscheint. Ignorieren Sie die Build-Nummer in der zweiten Zeile oben auf dem Bildschirm; sie ist falsch.
3. Klicken Sie auf das Feld „Versionsinfo“.

   Das Dialogfeld „RealPlayer Versionsinformationen“ erscheint.

4 Wählen Sie „RealPlayer Core“ in der Liste der angezeigten Komponenten aus.

Die Build-Nummer für die RealPlayer-Core-Komponente (z. B. 6.0.8.1649) ist identisch mit der von der Funktion `realPlayerVersion()` zurückgegebenen Build-Nummer.

Parameter
Keiner

Beispiel
Der folgende Code zeigt, dass die Build-Nummer der auf dem System installierten RealPlayer®-Software 6.0.9.357 lautet:

```
-- Lingo syntax
put(realPlayerVersion())

// JavaScript syntax
put(realPlayerVersion());
```

recordFont

Syntax
`recordFont(whichCastMember, font {[,face]} {,[bitmapSizes]} {,characterSubset} {, userFontName})`

Beschreibung
Dieser Befehl bettet eine TrueType- oder Type 1-Schrift als Darsteller ein. Nach Einbettung stehen diese Schriften dem Autor genauso wie alle anderen im System installierten Schriften zur Verfügung.

Sie müssen mit dem Befehl `new()` einen leeren Schriftdarsteller erstellen, bevor Sie den Befehl `recordFont` verwenden können.

Dieser Befehl erstellt einen Shock Font in *whichCastMember*, und zwar unter Verwendung der im Parameter *font* angegebenen Schrift. Aus dem durch diesen Befehl zurückgegebenen Wert ist ersichtlich, ob der Vorgang erfolgreich war oder nicht. Der Wert 0 bedeutet einen erfolgreichen Abschluss.

Parameter
font Erforderlich. Gibt den Namen der aufzuzeichnenden Originalschrift an.

face Optional. Gibt eine Liste mit Symbolen an, die das Schriftbild der Originalschrift anzeigen. Mögliche Werte sind`#plain`, `#bold` und `#italic`. Wenn Sie keinen Wert für diesen Parameter angeben, wird `#plain`verwendet. Falls die Werte `[# bold, #italic]` gemeinsam eingegeben werden, wird daraus eine neue Schriftart mit fett-kursiver Formatierung generiert.

bitmapSizes Optional. Gibt eine Liste von Ganzzahlen an, durch die die Größen für die aufzuzeichnenden Bitmaps angegeben werden. Dieser Parameter kann leer sein. Wenn Sie diesen Parameter auslassen, werden keine Bitmaps generiert. Bitmaps ergeben in der Regel bei kleineren Punktgrößen (d. h. unter 14 Punkt) ein besseres Schriftbild, belegen aber auch mehr Speicherplatz.

characterSubset Optional. Gibt eine Zeichenfolge zu kodierender Zeichen an. Nur die angegebenen Zeichen sind in der Schrift verfügbar. Wenn Sie diesen Parameter auslassen, werden alle Zeichen kodiert. Falls nur bestimmte Zeichen kodiert sind, aber ein nicht kodiertes Zeichen verwendet wird, erscheint dieses Zeichen als leeres Feld.

userFontName(optional). – Ein String, der als Name für den neu aufgezeichneten Schriftdarsteller verwendet werden soll.

Beispiel

Die folgende Anweisung erstellt einen einfachen Shock Font unter Verwendung der zwei Argumente für den Darsteller und die aufzuzeichnende Schrift:

```
-- Lingo
myNewFontMember = new(#font)
recordFont(myNewFontMember, "Lunar Lander")

// Javascript
var myNewFontMember = new(symbol("font")) ;
myNewFontMember.recordFont( "Lunar Lander") ;
```

Die folgende Anweisung gibt die zu erstellenden Bitmapgrößen und die Zeichen an, für die Schriftdaten erstellt werden sollen:

```
-- Lingo
myNewFontMember = new(#font)
recordfont(mynewmember,"lunar lander",[],[14, 18, 45], "Lunar Lander Game High Score First
Last Name")

// Javascript
var myNewFontMember = new(symbol("font")) ;
recordfont(mynewmember,"lunar lander",[],[14, 18, 45], "Lunar Lander Game High Score First
Last Name") ;
```

Hinweis: *Da die Schriftdaten durch* `recordFont` *nur neu synthetisiert, aber nicht direkt verwendet werden, kann der Shock Font ohne rechtliche Beschränkungen verteilt werden.*

Siehe auch

`newMember()`

rect()

Syntax

```
-- Lingo syntax
rect(intLeft, intTop, intRight, intBottom)

// JavaScript syntax
rect(intLeft, intTop, intRight, intBottom);
```

Beschreibung

Diese Top-Level-Funktion definiert ein Rechteck.

Sie können an Rechtecken sowohl mit Lingo als auch mit JavaScript-Syntax arithmetische Operationen ausführen. Wenn Sie einem Rechteck z. B. einen Einzelwert hinzufügen, wird dieser Wert durch Lingo automatisch zu jedem Element im Rechteck hinzugefügt.

Sie können die Rechteckkomponenten über die Listen- oder Eigenschaftssyntax referenzieren. Die folgenden Zuweisungen legen beispielsweise sowohl `myRectWidth1` als auch `myRectWidth2` auf 50 fest:

```
// JavaScript syntax
var myRect = rect(40,30,90,70);
var myRectWidth1 = myRect.right - myRect.left; // 50
var myRectWidth2 = myRect[3] - myRect[1]; // 50
```

Ein Beispiel für `rect()` in einem fertigen Film finden Sie im Film „Imaging“ im Ordner „Learning\\Lingo_“ im Director-Anwendungsordner.

Parameter

intLeft Erforderlich. Eine Ganzzahl, die die Anzahl von Pixeln angibt, die die linke Seite des Rechtecks vom linken Rand der Bühne entfernt ist.

intTop Erforderlich. Eine Ganzzahl, die die Anzahl von Pixeln angibt, die die obere Seite des Rechtecks vom oberen Rand der Bühne entfernt ist.

intRight Erforderlich. Eine Ganzzahl, die die Anzahl von Pixeln angibt, die die rechte Seite des Rechtecks vom linken Rand der Bühne entfernt ist.

intBottom Erforderlich. Eine Ganzzahl, die die Anzahl von Pixeln angibt, die die untere Seite des Rechtecks vom oberen Rand der Bühne entfernt ist.

Beispiel

Die folgende Anweisung setzt die Variable `newArea` auf ein Rechteck, dessen linke Seite bei 100, Oberkante bei 150, rechte Seite bei 300 und Unterkante bei 400 Pixel liegt:

```
-- Lingo syntax
newArea = rect(100, 150, 300, 400)

// JavaScript syntax
var newArea = rect(100, 150, 300, 400);
```

Nur in Lingo stellt die folgende Anweisung die Variable `newArea` auf ein Rechteck ein, das durch die Punkte `firstPoint` und `secondPoint` definiert ist.

```
-- Lingo syntax
firstPoint = point(100, 150)
secondPoint = point(300, 400)
newArea = rect(firstPoint, secondPoint)
```

Nur in Lingo addieren und subtrahieren die folgenden Anweisungen Werte für Rechtecke:

```
-- Lingo syntax
put(rect(0, 0, 100, 100) + rect(30, 55, 120, 95)) -- rect(30, 55, 220, 195)
put(rect(0, 0, 100, 100) -rect(30, 55, 120, 95)) -- rect(-30, -55, -20, 5)
```

Nur in Lingo addiert die folgende Anweisung den Wert 80 zu jeder Koordinate eines Rechtecks:

```
-- Lingo syntax
put(rect(60, 40, 120, 200) + 80) -- rect(140, 120, 200, 280)
```

Nur in Lingo dividiert die folgende Anweisung jede Koordinate in einem Rechteck durch 3:

```
-- Lingo syntax
put(rect(60, 40, 120, 200) / 3) -- rect(20, 13, 40, 66)
```

Siehe auch

`point()`, `quad`

registerByteArrayCallback

Syntax

```
soundObject.registerByteArrayCallback(position, #callbackFunction, [castMemRef], [read-write/read-only flag])
```

Beschreibung

Diese Bytearray-Methode ruft die Callback-Methode auf, die unter `#callbackFunction` festgelegt wird. Das Bytearray wird mit Audiosamples des Soundobjekts gefüllt.

Parameter

Parameter	Beschreibung	Standardwert
position	Die Werte für „#preFilter“ und „#postFilter“ lassen sich mit diesem Parameter festlegen. `#preFilter` legt die Audio-PCM-Samples vor dem Anwenden der Soundobjektfilter fest. `#postFilter` legt die Audio-PCM-Samples nach dem Anwenden der Soundobjektfilter fest.	Erforderlich
symCallback	Registers `callbackFunction` als Callbackmethode für die Audiodaten im Soundobjekt.	Erforderlich
castMemRef	Legt eine Darstellerreferenz fest, wenn die Callbackmethode ein Darsteller des Parent-Skripts ist. Legen Sie den Parameter nicht fest, wenn die Callbackmethode in „movieScript“ ist.	Optional
Schreib-Lese/Schreibschutz-Flag	Legt fest, ob die Callbackfunktion die Eigenschaften `#readOnly` oder `#readWrite` verwendet. #readOnly - Die Bytearray-Audiodaten sind in der Callbackmethode schreibgeschützt. #readWrite - Die Bytearray-Audiodaten können in der Callbackmethode gelesen und geschrieben werden. Dadurch können die Daten im Callback geändert werden.	#readOnly

Die Implementierung der `callbackFunction` ist wie folgt:

```
On callbackFunction byteArray, bitDepth, sampleRate, channelCount
    Put  byteArray --Audio output implementation
End
```

Hinweis: *Wenn innerhalb der Callbackprozedur ein Skript Laufzeitfehler verursacht (weil z. B. eine Eigenschaft nicht gefunden wird), werden die Fehlermeldungen nicht angezeigt und die Prozedurausführung wird an dem Punkt abgebrochen, an welchem der Fehler auftrat. Alle nachfolgenden Anweisungen werden nicht ausgeführt.*

Beispiele

```
-- BUTTON BEHAVIOR --
global gSound -- soundObject
global gMixer -- mixer object
global counter -- Counter to fill the silence
global fillSilence -- Flag to fill the silence
on mouseUp(me)
-------------------------------------------------------
-- ACTION: Creates a soundObject from the Cast member.
-- The callback handler HAS to be in a Parent Script.
-- See the Movie Script for callback details.
-------------------------------------------------------------------
gMixer = new(#mixer)
gSound = gMixer.createSoundObject("1", member(1))
--Create the sound object from cast member
sPre = new script("Output") -- Output is a parent script
sPost = new script("Output")
sPre.countLimit = 50 -- Setting the Property value for filling the silence
sPost.countLimit = 10
fl = gSound.filterlist
fl.append(audioFilter(#reverbFilter)) -- Adding Reverb Filter
gSound.registerByteArraycallback(#preFilter, #audioOutput,sPre,#readWrite)
-- Registering the callback with preFilter, so that the bytes will be available
--before applying filters
-- Specifying the script reference as sPre, can specify it as movie script if it is
--empty
-- Specifying readWrite, so that the bytes can be modified in the callback method
gSound.registerByteArrayCallback(#postFilter, #audioOutput,sPost,#readWrite)
-- Registering the callback with postFilter, so that the bytes will be available
--after applying filters
gMixer.bufferSize = 200 -- Setting the bufferSize of mixer to 200
gMixer.play()   -- Play the sound
end
```

Callback-Methode (Parentskript)

```
-- PARENT SCRIPT named OUTPUT --
global gSoundIOInstance -- instance of FileIO for reading sound file
global gSound -- soundObject
global counter -- Counter to  fill the  silence
global fillSilence -- Flag to fill the silence
property countLimit
--Callback Method
on audioOutput me, aByteArray, bitDepth, sampleFreq, numChannel
 ------------ --------- ---------------- -------
-- SOURCE: Called back from gSound whenever gSound has finished
--  playing its current contents.
-- INPUT: <aByteArray> will be the byteArray with audio data which has to be
-- modified and the byteArray to send back.
-- ACTION: Transfers the (remaining) contents of the file to
-- aByteArray
-- OUTPUT: Returns TRUE if there was any data to pas to aByteArray
-- FALSE if not.
-----------------------------------------------------------------
if fillSilence = 1 then -- Checking the fillSilence flag
repeat with i = 1 to aByteArray.length -- Looping thru the bytes of audio data
aByteArray[i] = 0 -- Setting the byte to silence
--Modify the bytes before playing, here it is set to 0 means silence
end repeat
end if
put aByteArray
counter = counter + 1 -- Increasing the counter to keep filling the silence
if(counter > countLimit)then -- If the countLimit is reached stop filling the silence
counter = 0
fillSilence = 1 - fillSilence -- Toggling between silence & play
end if
end
```

Siehe auch

unregisterByteArrayCallback

registerCuePointCallback

Syntax

```
soundObject.registerCuePointCallback(#symCallback, [castMemRef])
```

Beschreibung

Diese Soundobjektmethode registriert `symCallback` als Callback für die Aufrufpunkte des Soundstreams für die Wiedergabe des aktuellen Soundobjekts.

Optional kann auch eine Darstellerreferenz übergeben werden, wenn die Callback-Funktion zum übergeordneten Skript gehört.

Parameter

Parameter	Beschreibung	Erforderlich/Optional
symcallback	Ruft das Symbol jedesmal auf, wenn der Cue-Punkt erreicht ist.	Erforderlich
castMemRef	Übergibt eine Darstellerreferenz, wenn die Callbackmethode ein Darsteller des Parentskripts ist.	Optional

Beispiele

```
--Lingo syntax
on mouseUp me
     soundObjRef.registerCuePointCallback(#callMe) -- Calls the callMe function when the
-- cue points of the sound object associated with soundObjRef are hit.
end

// JavaScript syntax
function mouseUp(){
soundObjRef.registerCuePointCallback(#callMe); // Calls the callMe function when the cue
// points of the sound object associated with soundObjRef are hit.
}
```

Siehe auch

unregisterCuePointCallback

registerEndOfSpoolCallback()

Syntax

```
soundObject.registerEndOfSpoolCallback(#callbackFun, #parentScriptRef)
```

Beschreibung

Registriert „callbackFun“ als die Callbackfunktion, wenn das Auslesen des Soundobjekts abgeschlossen ist. Dieses Callback gibt das Mitglied aus, das als nächstes wiedergegeben wird. Optional können Sie außerdem einen Verweis zum Parent-Skript übergeben, wenn die Callbackfunktion ein Bestandteil eines Parent-Skripts ist.

Parameter

Parameter	Beschreibung	Erforderlich/Optional
callbackFun	Ruft das Symbol aus, wenn das Auslesen des Soundobjekts abgeschlossen ist.	Erforderlich
parentScriptRef	Übergibt eine Darstellerreferenz, wenn die Callbackmethode ein Darsteller des Parentskripts ist.	Optional

Beispiel

```
--Lingo syntax
on mous eUp mesoundObj Ref.r egist erEndOfSpoolCallback( #callMe)
-- Calls the callMe method when the sound objec t read is complete.
-- callme method should return the reference of next sound cast member
end
--The callme method is implemented as
On callme
bgcount = bgcount +1
return(member(bgcount))
End
// JavaScript syntax
function mouseUp(){
soundObjRef.registerEndOfSpoolCallback(#callMe); // Calls the callMe method when the
// sound object read is finished. callme method should return the reference of next sound cast member
}
```

Siehe auch

unregisterEndOfSpoolCallback()

registerForEvent()

Syntax

```
member(whichCastmember).registerForEvent(eventName, handlerName, scriptObject {, begin,
period, repetitions})
```

Beschreibung

Dieser 3D-Befehl gibt an, dass die angegebene Prozedur aufgerufen werden soll, wenn das angegebene Ereignis im angegebenen Darsteller auftritt.

Die folgenden Parameterbeschreibungen gelten für die Befehle `registerForEvent()` und `registerScript()`.

Hinweis: *Mit dem Befehl `registerScript()` können Sie die Registrierung eines Skripts mit einem bestimmten Node (statt mit einem Darsteller) verknüpfen.*

Parameter

eventName Erforderlich. Gibt den Namen des Ereignisses an. Das Ereignis kann eines der folgenden vordefinierten Ereignisse oder ein benutzerdefiniertes Ereignis sein:

- `#collideAny` ist ein beliebiges Kollisionsereignis.
- `#collideWith` ist ein Kollisionsereignis, an dem dieses bestimmte Modell beteiligt ist. Der Befehl `setCollisionCallback()` ist eine Kurzform des Befehls `registerScript()` für das Ereignis `#collideWith`.
- `#animationStarted` und `#animationEnded` sind Benachrichtigungsereignisse, die auftreten, wenn die Wiedergabe einer Knochen- oder Schlüsselbildanimation beginnt oder endet. Die Prozedur erhält drei Argumente: *eventName*, *motion* und *time*. Das Argument *eventName* lautet entweder `#animationStarted` oder `#animationEnded`. Das Argument *motion* ist der Name der Bewegung, deren Wiedergabe begonnen hat oder abgeschlossen ist, und *time* ist die aktuelle Zeit der Bewegung.
- Bei Animationen, die in Schleife ablaufen, tritt das Ereignis `#animationStarted` nur bei der ersten Schleife auf. Wenn zwei Animationen gemischt werden, wird dieses Ereignis zu Beginn des Mischungsprozesses gesendet.

- Wenn sich mehrere Animationen für das Modell in einer Warteschlange befinden und die Animationseigenschaft `autoBlend` auf `TRUE` gesetzt ist, kann das Ereignis `#animationEnded` vor dem sichtbaren Ende einer bestimmten Bewegung auftreten, da die Eigenschaft `autoBlend` den Anschein einer fortgesetzten Bewegung erwecken kann, selbst wenn die Animation ihr definiertes Ende bereits erreicht hat.
- `#timeMS` ist ein Zeitereignis. Das erste `#timeMS`-Ereignis tritt auf, wenn die im Parameter *begin* angegebene Anzahl von Millisekunden nach Aufruf von `registerForEvent` verstrichen ist. Der Parameter *period* bestimmt die Anzahl von Millisekunden zwischen `#timeMS`-Ereignissen, wenn der Wert von *repetitions* größer als 0 ist. Wenn *repetitions* 0 ist, tritt das `#timeMS`-Ereignis unbegrenzt lang auf.

handlerName Erforderlich. Gibt den Namen der Prozedur an, die aufgerufen werden soll, wenn das Ereignis *eventName* eintritt. Diese Prozedur ist in dem durch *scriptObject* bezeichneten Skriptobjekt zu finden. An die Prozedur werden die folgenden Argumente gesendet:

- `type` ist immer 0.
- `delta` ist die Zeit in Millisekunden, die seit dem letzten `#timeMS`-Ereignis vergangen ist.
- `time` ist die Zeit in Millisekunden, die seit dem ersten `#timeMS`-Ereignis vergangen ist. Wenn es beispielsweise drei Iterationen mit einer Dauer von 500 ms gibt, ist der „time“-Wert bei der ersten Iteration 0, bei der zweiten 500 und bei der dritten 1000.
- `duration` ist die Gesamtzeit in Millisekunden, die zwischen dem Aufruf von `registerForEvent` und dem letzten `#timeMS`-Ereignis verstreicht. Wenn es beispielsweise fünf Iterationen mit einer Dauer von jeweils 500 ms gibt, beträgt der "duration"-Wert 2500 ms. Bei Aufgaben mit unbegrenzten Iterationen ist der "duration"-Wert 0.
- `systemTime` ist die absolute Zeit in Millisekunden seit Beginn des Director-Films.

scriptObject Erforderlich. Gibt das Skriptobjekt an, das die Prozedur *handlerName* enthält. Wenn Sie für *scriptObject* "0" angeben, wird die erste Ereignisprozedur mit dem angegebenen Namen aufgerufen, die im Filmskript gefunden wird.

begin Optional. Gibt die Anzahl der Millisekunden an, nach denen seit dem Aufrufen von `registerForEvent()` das erste `#timeMS`-Ereignis eintritt.

period Optional. Gibt die Anzahl der Millisekunden zwischen `#timeMS`-Ereignissen an, wenn der Wert von *repetitions* größer als 0 ist.

repetitions Optional. Gibt die Anzahl der Wiederholungen für das `#timeMS`-Ereignis an. Wenn *repetitions* auf 0 festgelegt ist, tritt das `#timeMS`-Ereignis auf unbestimmte Zeit ein.

Beispiel

Die folgende Anweisung registriert die in einem Filmskript gefundene Ereignisprozedur `promptUser`, damit sie zweimal im Abstand von 5 Sekunden aufgerufen wird:

```
member("Scene").registerForEvent(#timeMS, #promptUser, 0, 5000, 5000, 2)
```

Die folgende Anweisung registriert die in einem Filmskript gefundene Ereignisprozedur `promptUser`, damit sie bei jeder Kollision im Darsteller `Scene` aufgerufen wird:

```
member("Scene").registerForEvent(#collideAny, #promptUser, 0)
```

Die folgende Anweisung deklariert die Prozedur `promptUser` im selben Skript wie den Befehl `registerForEvent`, damit sie bei jeder Kollision mit dem Modell `Pluto` im Darsteller `Scene` aufgerufen wird:

```
member("Scene").registerForEvent(#collideWith, #promptUser, me,
member("Scene").model("Pluto"))
```

Siehe auch

`setCollisionCallback()`, `registerScript()`, `play() (3D)`, `playNext() (3D)`, `autoblend`, `blendTime`, `sendEvent`, `unregisterAllEvents`

registerScript()

Syntax

```
member(whichCastmember).model(whichModel).registerScript(eventName, handlerName,
scriptObject {, begin, period, repetitions})
member(whichCastmember).camera(whichCamera).registerScript(eventName, handlerName,
scriptObject {, begin, period, repetitions})
member(whichCastmember).light(whichLight).registerScript(eventName, handlerName,
scriptObject {, begin, period, repetitions})
member(whichCastmember).group(whichGroup).registerScript(eventName, handlerName,
scriptObject {, begin, period, repetitions})
```

Beschreibung

Dieser 3D-Befehl registriert die angegebene Prozedur, damit sie aufgerufen wird, wenn für den referenzierten Node das angegebene Ereignis auftritt.

Die folgenden Parameterbeschreibungen gelten für die Befehle `registerForEvent()` und `registerScript()`.

Parameter

eventName Erforderlich. Gibt den Namen des Ereignisses an. Das Ereignis kann eines der folgenden vordefinierten Ereignisse oder ein benutzerdefiniertes Ereignis sein:

- `#collideAny` ist ein beliebiges Kollisionsereignis.
- `#collideWith` ist ein Kollisionsereignis, an dem dieses bestimmte Modell beteiligt ist. Der Befehl `setCollisionCallback()` ist eine Kurzform des Befehls `registerScript()` für das Ereignis `#collideWith`.
- `#animationStarted` und `#animationEnded` sind Benachrichtigungsereignisse, die auftreten, wenn die Wiedergabe einer Knochen- oder Schlüsselbildanimation beginnt oder endet. Die Prozedur erhält drei Argumente: *eventName*, *motion* und *time*. Das Argument *eventName* lautet entweder `#animationStarted` oder `#animationEnded`. Das Argument *motion* ist der Name der Bewegung, deren Wiedergabe begonnen hat oder abgeschlossen ist, und *time* ist die aktuelle Zeit der Bewegung.

  Bei Animationen, die in Schleife ablaufen, tritt das Ereignis `#animationStarted` nur bei der ersten Schleife auf. Wenn zwei Animationen gemischt werden, wird dieses Ereignis zu Beginn des Mischungsprozesses gesendet.

  Wenn sich mehrere Animationen für das Modell in einer Warteschlange befinden und die Animationseigenschaft `autoBlend` auf `TRUE` gesetzt ist, kann das Ereignis `#animationEnded` vor dem sichtbaren Ende einer bestimmten Bewegung auftreten, da die Eigenschaft `autoBlend` den Anschein einer fortgesetzten Bewegung erwecken kann, selbst wenn die Animation ihr definiertes Ende bereits erreicht hat.
- `#timeMS` ist ein Zeitereignis. Das erste `#timeMS`-Ereignis tritt auf, wenn die im Parameter *begin* angegebene Anzahl von Millisekunden nach Aufruf von `registerForEvent` verstrichen ist. Der Parameter *period* bestimmt die Anzahl von Millisekunden zwischen `#timeMS`-Ereignissen, wenn der Wert von *repetitions* größer als 0 ist. Wenn *repetitions* 0 ist, tritt das `#timeMS`-Ereignis unbegrenzt lang auf.

handlerName Erforderlich. Gibt den Namen der Prozedur an, die aufgerufen werden soll, wenn das Ereignis *eventName* eintritt. Diese Prozedur ist in dem durch *scriptObject* bezeichneten Skriptobjekt zu finden. An die Prozedur werden die folgenden Argumente gesendet:

- `type` ist immer 0.
- `delta` ist die Zeit in Millisekunden, die seit dem letzten `#timeMS`-Ereignis vergangen ist.
- `time` ist die Zeit in Millisekunden, die seit dem ersten `#timeMS`-Ereignis vergangen ist. Wenn es beispielsweise drei Iterationen mit einer Dauer von 500 ms gibt, ist der „time“-Wert bei der ersten Iteration 0, bei der zweiten 500 und bei der dritten 1000.
- `duration` ist die Gesamtzeit in Millisekunden, die zwischen dem Aufruf von `registerForEvent` und dem letzten `#timeMS`-Ereignis verstreicht. Wenn es beispielsweise fünf Iterationen mit einer Dauer von jeweils 500 ms gibt, beträgt der "duration"-Wert 2500 ms. Bei Aufgaben mit unbegrenzten Iterationen ist der "duration"-Wert 0.
- `systemTime` ist die absolute Zeit in Millisekunden seit Beginn des Director-Films.

scriptObject Erforderlich. Gibt das Skriptobjekt an, das die Prozedur *handlerName* enthält. Wenn Sie für *scriptObject* "0" angeben, wird die erste Ereignisprozedur mit dem angegebenen Namen aufgerufen, die im Filmskript gefunden wird.

begin Optional. Gibt die Anzahl der Millisekunden an, nach denen seit dem Aufrufen von `registerForEvent()` das erste `#timeMS`-Ereignis eintritt.

period Optional. Gibt die Anzahl der Millisekunden zwischen `#timeMS`-Ereignissen an, wenn der Wert von *repetitions* größer als 0 ist.

repetitions Optional. Gibt die Anzahl der Wiederholungen für das `#timeMS`-Ereignis an. Wenn *repetitions* auf 0 festgelegt ist, tritt das `#timeMS`-Ereignis auf unbestimmte Zeit ein.

Beispiel

Die folgende Anweisung registriert die in einem Filmskript enthaltene Ereignisprozedur `messageReceived`, damit sie aufgerufen wird, wenn das Modell "Player" das benutzerdefinierte Ereignis `#message` empfängt:

```
member("Scene").model("Player").registerScript(#message, #messageReceived, 0)
```

Die folgende Anweisung registriert die Ereignisprozedur `collisionResponder`, die im selben Skript zu finden ist wie der Befehl `registerScript`, damit sie jedesmal aufgerufen wird, wenn es zu einem Zustammenstoß zwischen dem Modell `Player` und einem anderen Modell kommt, an dem der Modifizierer `#collision` angebracht ist:

```
member("Scene").model("Player").registerScript(#collideWith, #collisionResponder, me)
```

Siehe auch

`registerForEvent()`, `sendEvent`, `setCollisionCallback()`

removeBackdrop

Syntax

```
member(whichCastmember).camera(whichCamera).removeBackdrop(index)
```

Beschreibung

Dieser 3D-Befehl entfernt den Hintergrund an einer angegebenen Position aus der Hintergrundliste der Kamera.

Parameter
index Erforderlich. Gibt die Indexposition des Hintergrunds in der Hintergrundliste an.

Beispiel
Die folgende Anweisung entfernt den dritten Hintergrund aus der Hintergrundliste von Kamera 1 im Darsteller "Szene". Der Hintergrund verschwindet auch von der Bühne, wenn diese Kamera gegenwärtig von Sprites verwendet wird.

```
-- Lingo
member("Scene").camera[1].removeBackdrop(3)

// Javascript
member("Scene").getProp("camera", 1).removeBackdrop(3) ;
```

Siehe auch
`insertBackdrop`, `overlay`, `backdrop`

removeFromWorld

Syntax
```
member(whichCastmember).model(whichModel).removeFromWorld()
member(whichCastmember).light(whichLight).removeFromWorld()
member(whichCastmember).camera(whichCamera).removeFromWorld()
member(whichCastmember).group(whichGroup).removeFromWorld()
```

Beschreibung
Dieser 3D-Befehl setzt das Parent-Objekt von Modellen, Lichtquellen, Kameras oder Gruppen, deren Parent-Hierarchie im Objekt "World" endet, auf „void“ und entfernt es aus der Welt.

Bei Objekten, deren Parent-Hierarchie nicht in der Welt endet, ist dieser Befehl wirkungslos.

Parameter
Keiner

Beispiel
Die folgende Anweisung entfernt das Modell `gbCyl` aus der 3D-Welt des Darstellers `Scene`:

```
-- Lingo
member("Scene").model("gbCyl").removeFromWorld()

// Javascript
member("Scene").getPropRef("model" , a).removeFromWorld() ;
// where a is the number index for gbCyl model.
```

removeLast()

Syntax
```
member(whichCastmember).model(whichModel).bonesPlayer.removeLast()
member(whichCastmember).model(whichModel).keyframePlayer.removeLast()
```

Beschreibung
Dieser 3D-Befehl für die Modifizierer `keyframePlayer` und `bonesPlayer` entfernt die letzte Bewegung aus der Abspielliste (`playlist`) des Modifizierers.

Parameter
Keiner

Beispiel
Die folgende Anweisung entfernt die letzte Bewegung aus der Abspielliste des Modifizierers `bonesPlayer` für das Modell `Walker`:

```
member("MyWorld").model("Walker").bonesPlayer.removelast()
```

removeModifier

Syntax
```
member(whichCastmember).model(whichModel).removeModifier.(#whichModifier)
```

Beschreibung
Dieser 3D-Befehl entfernt den angegebenen Modifizierer aus dem vorgegebenen Modell.

Dieser Befehl gibt `TRUE` zurück, wenn die Ausführung erfolgreich ist, und `FALSE`, wenn *#whichModifier* kein gültiger Modifizierer ist bzw. der angegebene Modifizierer nicht am Modell angebracht war.

Parameter
whichModifier Erforderlich. Gibt den zu entfernenden Modifizierer an.

Beispiel
Die folgende Anweisung entfernt den Modifizierer `#toon` aus dem Modell `Box`:

```
-- Lingo
member("shapes").model("Box").removeModifier(#toon)

// JavaScript syntax
member("shapes").getPropRef("model" , i).removeModifier(symbol("toon"));
// where "i" is the number index.
```

Siehe auch
`addModifier`, `modifier`, `modifier[]`, `modifiers`

removeOverlay

Syntax
```
member(whichCastmember).camera(whichCamera).removeOverlay(index)
```

Beschreibung
Dieser 3D-Befehl entfernt die Überlagerung an einer angegebenen Position aus der Überlagerungsliste der Kamera.

Parameter
index Erforderlich. Gibt die Indexposition der Überlagerung in der Liste der Überlagerungen an.

Beispiel
Die folgende Anweisung entfernt die erste Überlagerung aus der Überlagerungsliste der von Sprite 5 verwendeten Kamera. Die Überlagerung verschwindet von der Bühne.

```
-- Lingo
sprite(5).camera.removeOverlay(1)

// Javascript
sprite(5).camera.removeOverlay(1) ;
```

Siehe auch
`overlay`, `addOverlay`

removeScriptedSprite()

Syntax
```
-- Lingo syntax
spriteChannelObjRef.removeScriptedSprite()

// JavaScript syntax
spriteChannelObjRef.removeScriptedSprite();
```

Beschreibung
Diese Sprite-Kanalmethode wechselt die Steuerung eines Sprite-Kanals vom Skript zurück zur Bühne.

Parameter
Keiner

Beispiel
Die folgende Anweisung entfernt das mit Skript erzeugte Sprite aus Sprite-Kanal 5:

```
-- Lingo syntax
channel(5).removeScriptedSprite()

// JavaScript syntax
channel(5).removeScriptedSprite();
```

Siehe auch
`makeScriptedSprite()`, `Sprite Channel`

replaceMember

Syntax
```
soundObject.replaceMember(<member>, [startTime, endTime, loopCount, loopStartTime,
loopEndTime, preLoadTime])
```

Beschreibung

Mit dieser Soundobjektmethode werden die mit einem Soundobjekt verknüpften Audiodarsteller dynamisch ersetzt. Diese Ersetzung erfolgt unverzüglich.

Wenn die Formate der neuen und alten Darsteller unterschiedlich sind, wird der neue Darsteller vor der Wiedergabe zum Format des alten Darstellers konvertiert. Die Wiedergabe des Soundobjekts erfolgt daher immer in dem Format des Darstellers, das bei dessen Erstellung verwendet wurde.

Wenn ein Soundobjekt angehalten und wiedergegeben wird, gibt das Soundobjekt den Darsteller wieder, der bei dessen Erstellung verwendet wurde.

Hinweis: *Die Methode* `replaceMember()` *funktioniert nicht für Bytearray-Audiodarsteller und die Audiospur von Videodarstellern.*

Parameter

Parameter	Beschreibung
member	Der Audiodarsteller, der den aktuell wiedergegebenen Sound ersetzt.

Der Rest der Parameter für die Methode `replaceMember` ist ähnlich den Parametern für die Methode `createSoundObject`. Siehe createSoundObject.

Beispiele

Die folgenden Beispiele ersetzen das Soundobjektmitglied `gun` mit dem Mitglied `laser`, das in einer unendlichen Schleife wiedergegeben wird:

```
-- Lingo syntax
on mouseUp me
gun.replaceMember(member("laser"),[#loopcount:0])
end

// JavaScript syntax
function mouseUp(){
gun.replacemember(member("laser");
}
```

Siehe auch

createSoundObject

reset (Mixer)

Syntax

```
mixer.reset()
```

Beschreibung

Versetzt den Mixer zurück in den Zustand vor dem letzten Speichervorgang. Falls der Mixer erst nach dem letzten Speichervorgang erstellt wurde, wird der Mixer (mit den

Rufen Sie `reset`nur auf, wenn sich der Mixer im Zustand „Gestoppt“ befindet. Diese Methode gibt bei Erfolg den Wert 1 zurück und im Fehlerfall 0. Falls die Methode beispielsweise aufgerufen wird, während die Wiedergabe des Mixers läuft, wird 0 zurückgeliefert.

Beispiele

```
-- Lingo syntax
on stopMovie
     mixer1.reset()
end

// JavaScript syntax
function stopMovie() {
     mixer1.reset();

}
```

Siehe auch

Mixer

resetWorld

Syntax

```
member(whichCastmember).resetWorld()
member(whichTextCastmember).resetWorld()
```

Beschreibung

Dieser 3D-Befehl setzt die Darstellereigenschaften des referenzierten 3D-Darstellers auf die Werte zurück, die gespeichert wurden, als der Darsteller anfänglich in den Speicher geladen wurde. Die Eigenschaft `state` des Darstellers muss entweder 0 (nicht geladen), 4 (Media geladen) oder -1 (Fehler) sein, bevor dieser Befehl ausgeführt werden kann. Andernfalls tritt ein Skriptfehler auf.

Dieser Befehl unterscheidet sich von `revertToWorldDefaults` darin, dass die Werte aus dem Zustand übernommen werden, in dem sich der Darsteller beim anfänglichen Laden in den Speicher befand, und nicht aus dem Erstellungszustand des Darstellers.

Parameter

Keiner

Beispiel

Die folgende Anweisung setzt die Eigenschaften des Darstellers "Szene" auf die Werte zurück, die er besaß, als er in den Speicher geladen wurde:

```
-- Lingo
member("Scene").resetWorld()

// Javascript
member("Scene").resetWorld() ;
```

Siehe auch

revertToWorldDefaults

resolveA

Syntax

```
collisionData.resolveA(bResolve)
```

Beschreibung

Diese 3D-Kollisionsmethode überschreibt das durch die Eigenschaft `collision.resolve` für `collisionData.modelA` festgelegte Verhalten. Rufen Sie diese Funktion nur dann auf, wenn Sie das mit `collision.resolve` für "modelA" festgelegte Verhalten ändern möchten.

Parameter

bResolve Erforderlich. Gibt an, ob Kollisionen für „modelA“ aufgelöst werden. Ist *bResolve* auf `TRUE` gesetzt, wird die Kollision für „modelA“ aufgelöst. Ist *bResolve* auf `FALSE` gesetzt, wird die Kollision für „modelA“ nicht aufgelöst.

Siehe auch

`collisionData`, `registerScript()`, `resolve`, `modelA`, `setCollisionCallback()`

resolveB

Syntax

```
collisionData.resolveB(bResolve)
```

Beschreibung

Diese 3D-Kollisionsmethode überschreibt das durch die Eigenschaft `collision.resolve` für `collisionData.modelB` festgelegte Verhalten. Rufen Sie diese Funktion nur dann auf, wenn Sie das mit `collision.resolve` für "modelB" festgelegte Verhalten ändern möchten.

Parameter

bResolve Erforderlich. Gibt an, ob Kollisionen für „modelB“ aufgelöst werden. Ist *bResolve* auf `TRUE` gesetzt, wird die Kollision für „modelB“ aufgelöst. Ist *bResolve* auf `FALSE` gesetzt, wird die Kollision für „modelB“ nicht aufgelöst.

Siehe auch

`collisionData`, `resolve`, `registerScript()`, `modelB`, `setCollisionCallback()`

restart()

Syntax

```
-- Lingo syntax
_system.restart()

// JavaScript syntax
_system.restart();
```

Beschreibung
Diese Systemmethode schließt alle offenen Anwendungen und startet den Computer neu.

Parameter
Keiner

Beispiel
Die folgende Anweisung startet den Computer neu, wenn der Benutzer Befehl+R (Mac) bzw. Strg+R (Windows) drückt:

```
-- Lingo syntax
if (_key.key = "r" and _key.commandDown) then
    _system.restart()
end if

// JavaScript syntax
if (_key.key = "r" && _key.commandDown) {
    _system.restart();
}
```

Siehe auch
System

restore()

Syntax

```
-- Lingo syntax
windowObjRef.restore()

// JavaScript syntax
windowObjRef.restore();
```

Beschreibung
Diese Fenstermethode stellt ein Fenster wieder her, nachdem es maximiert wurde.

Verwenden Sie diese Methode beim Erstellen benutzerdefinierter Titelleisten für Filme in einem Fenster (MIAWs).

Parameter
Keiner

Beispiel
Die folgende Anweisung stellt das maximierte Fenster namens „Steuerpult“ wieder her:

```
-- Lingo syntax
window("Control Panel").restore()

// JavaScript syntax
window("Control Panel").restore();
```

Siehe auch

`maximize()`, `Window`

result

Syntax

```
the result
```

Beschreibung

Diese Funktion zeigt den Wert des Rückgabeausdrucks der zuletzt ausgeführten Prozedur an.

Die Funktion `result` eignet sich besonders dazu, Werte aus den in Fenstern abspielenden Filmen abzurufen und den Verlauf von zu verfolgen. Hierzu werden die Prozedurergebnisse im Nachrichtenfenster angezeigt, während der Film abläuft.

Sie können ein Prozedurergebnis zurückgeben, indem Sie das Ergebnis einer Variablen zuordnen und dann den Wert dieser Variablen prüfen.Sie können z. B. die Anweisung `set myVariable =` *function()* verwenden, wobei *function()* der Name einer bestimmten Funktion ist.

Parameter

Keiner

Beispiel

Die folgende Prozedur gibt den Zufallswert für zwei Würfel zurück:

```
on diceRoll
    return random(6) + random(6)
end
```

Im folgenden Beispiel sind die zwei Anweisungen

```
diceRoll
roll = the result
```

gleichbedeutend mit der Anweisung

```
set roll = diceRoll()
```

Die Anweisung `roll = diceRoll` ruft die Prozedur nicht auf, da nach `diceRoll` keine Klammern stehen; `diceRoll` wird hier als Variablenverweis betrachtet.

Siehe auch

`return (Schlüsselwort)`

resume()

Syntax

```
-- Lingo syntax
animGifSpriteRef.resume()

// JavaScript syntax
animGifSpriteRef.resume();
```

Beschreibung

Diese Methode für animierte GIFs sorgt dafür, dass die Wiedergabe des unterbrochenen Sprites im Bild nach dem aktuellen Bild fortgesetzt wird. Dieser Befehl ist wirkungslos, wenn das animierte GIF-Sprite nicht unterbrochen wurde.

Parameter

Keiner

Siehe auch

rewind() (Animiertes GIF, Flash)

returnToTitle()

Syntax

```
-- Lingo syntax
dvdObjRef.returnToTitle()

// JavaScript syntax
dvdObjRef.returnToTitle();
```

Beschreibung

Diese DVD-Methode setzt die Wiedergabe fort, nachdem ein Menü angezeigt wurde.

Parameter

Keiner

Beispiel

Die folgende Anweisung setzt die Wiedergabe fort, nachdem ein Menü angezeigt wurde:

```
-- Lingo syntax
member(1).returnToTitle()

// JavaScript syntax
member(1).returnToTitle()
```

Siehe auch

DVD

revertToWorldDefaults

Syntax

```
member(whichCastmember).revertToWorldDefaults()
```

Beschreibung

Dieser 3D-Befehl setzt die Eigenschaften des angegebenen 3D-Darstellers auf die Werte zurück, die bei der Erstellung des Darstellers gespeichert wurden. Die Eigenschaft `state` des Darstellers muss entweder4 (geladen) oder -1 (Fehler) sein, bevor dieser Befehl ausgeführt werden kann. Andernfalls tritt ein Skriptfehler auf.

Dieser Befehl unterscheidet sich von `resetWorld` insofern, als die Werte aus dem Erstellungszustand des Darstellers übernommen werden und nicht aus dem Zustand, in dem sich der Darsteller beim anfänglichen Laden in den Speicher befand.

Parameter

Keiner

Beispiel

Die folgende Anweisung setzt die Eigenschaften des Darstellers "Szene" auf die Werte zurück, die bei der Erstellung des Darstellers gespeichert wurden:

```
-- Lingo
member("Scene").revertToWorldDefaults()

// Javascript
member("Scene").revertToWorldDefaults() ;
```

Siehe auch

`resetWorld`

rewind() (MP4Media/FLV)

Syntax

```
sprite(1).rewind()
```

Beschreibung

Diese MP4Media/FLV-Sprite-Methode ruft den Anfang des Videos auf. Der Aufruf dieser Methode hat keinen Einfluss auf `mediaStatus`.

Beispiele

Das folgende Beispiel spult das MP4-Video-Sprite an den Anfang zurück.

```
-- Lingo syntax
sprite(2).rewind()

// JavaScript syntax
sprite(2).rewind();
```

rewind() (Soundkanal)

Syntax

```
-- Lingo syntax
soundChannelObjRef.rewind()

// JavaScript syntax
soundChannelObjRef.rewind();
```

Beschreibung

Diese Soundkanalmethode unterbricht die Wiedergabe des aktuellen Sounds in einem Soundkanal und beginnt dann erneut bei `startTime`.

Wenn der Sound bereits unterbrochen ist, bleibt er auch weiterhin unterbrochen, und `currentTime` wird auf `startTime` gesetzt.

Parameter

Keiner

Beispiel

Die folgende Anweisung bewirkt, dass die Wiedergabe des Sounddarstellers, der in Soundkanal 1 abgespielt wird, erneut am Anfang beginnt:

```
-- Lingo syntax
sound(1).rewind()

// JavaScript syntax
sound(1).rewind();
```

Siehe auch

`Sound Channel`, `startTime (Sound Channel)`

rewind() (Windows Media)

Syntax

```
-- Lingo syntax
windowsMediaObjRef.rewind()

// JavaScript syntax
windowsMediaObjRef.rewind();
```

Beschreibung

Diese Methode für Windows Media-Darsteller und -Sprites spult zum ersten Bild eines Windows Media-Darstellers oder -Sprites zurück.

Das Aufrufen dieser Methode hat keine Auswirkungen auf `mediaStatus`.

Parameter

Keiner

Siehe auch

mediaStatus (RealMedia, Windows Media), Windows Media

rewind() (Animiertes GIF, Flash)

Syntax

```
-- Lingo syntax
animGifSpriteRef.rewind()

// JavaScript syntax
animGifSpriteRef.rewind();
```

Beschreibung

Dieser Befehl sendet ein Flash- oder animiertes GIF-Film-Sprite auf Bild 1 zurück, wenn das Sprite angehalten ist oder wenn es abgespielt wird.

Parameter

Keiner

Beispiel

Das folgende Bildskript prüft, ob das Flash-Film-Sprite, das sich in dem Sprite befindet, in dem das Verhalten platziert wurde, auch tatsächlich abgespielt wird. Ist dies der Fall, wird die Schleife im aktuellen Bild fortgesetzt. Sobald der Film beendet ist, spult das Sprite den Film zurück, sodass das erste Bild des Films auf der Bühne erscheint. Der Abspielkopf geht dann auf das nächste Bild über.

```
-- Lingo syntax
property spriteNum

    on exitFrame
    if sprite(spriteNum).playing then
        _movie.go(_movie.frame)
    else
        sprite(spriteNum).rewind()
        _movie.updatestage()
    end if
end

// JavaScript syntax
function exitFrame(me) {
    var plg = sprite(this.spriteNum).playing;
    if (plg == 1) {
        _movie.go(_movie.frame);
    } else {
        sprite(this.spriteNum).rewind();
        _movie.updateStage();
    }
}
```

rollOver()

Syntax

```
-- Lingo syntax
_movie.rollOver({intSpriteNum})

// JavaScript syntax
_movie.rollOver({intSpriteNum});
```

Beschreibung

Diese Filmmethode gibt an, ob sich der Cursor (Mauszeiger) derzeit über dem Begrenzungsrechteck eines angegebenen Sprites befindet (`TRUE` oder 1) oder nicht (`FALSE` oder 0).

Die `rollOver()`-Methode wird in der Regel in Bildskripten verwendet. Sie eignet sich zum Erstellen von Prozeduren, die eine Aktion ausführen, sobald der Benutzer den Zeiger auf ein bestimmtes Sprite setzt.

Falls der Benutzer die Maus weiterbewegt, kann sich der Wert von `rollOver()` ändern, während ein Skript eine Prozedur ausführt, was zu unvorhergesehenem Verhalten führen kann. Um sicherzustellen, dass die Prozedur immer den gleichen Rollover-Wert verwendet, sollten Sie bei Beginn der Prozedur die Funktion `rollOver()` einer Variablen zuordnen.

Wenn sich der Mauszeiger über einem Bühnenbereich befindet, in dem zuvor ein Sprite angezeigt wurde, wird `rollOver()` weiterhin aufgerufen und meldet, dass das Sprite immer noch vorhanden ist. Sie können dieses Verhalten vermeiden, indem Sie einfach keine Rollovers über diesen Positionen ausführen oder aber das Sprite über die Menüleiste verschieben, bevor Sie es löschen.

Parameter

intSpriteNum Optional. Eine Ganzzahl, die die Sprite-Nummer angibt.

Beispiel

Die folgende Anweisung ändert den Inhalt des Felddarstellers „Nachricht“ in „Hier sind Sie richtig“, sobald sich der Zeiger über Sprite 6 befindet:

```
-- Lingo syntax
if (_movie.rollOver(6)) then
    member("Message").text = "This is the place."
end if

// JavaScript syntax
if (_movie.rollOver(6)) {
    member("Message").text = "This is the place.";
}
```

Die folgende Prozedur verschiebt den Abspielkopf auf andere Bilder, wenn sich der Zeiger auf der Bühne über bestimmten Sprites befindet. Dabei wird der `rollOver`-Wert zuerst einer Variablen zugeordnet. Dadurch ist die Prozedur in der Lage, stets den `rollOver`-Wert zu verwenden, der bei Beginn des Rollovers aktiv war, ganz gleich, ob der Benutzer die Maus weiterbewegt oder nicht:

```
-- Lingo syntax
on exitFrame
    currentSprite = _movie.rollOver()
    case currentSprite of
        1: _movie.go("Left")
        2: _movie.go("Middle")
        3: _movie.go("Right")
    end case
end exitFrame

// JavaScript syntax
function exitFrame() {
    var currentSprite = _movie.rollOver();
    switch (currentSprite) {
        case 1: _movie.go("Left");
            break;
        case 2: _movie.go("Middle");
            break;
        case 3: _movie.go("Right");
            break;
    }
}
```

Siehe auch

Movie

rootMenu()

Syntax

```
-- Lingo syntax
dvdObjRef.rootMenu()

// JavaScript syntax
dvdObjRef.rootMenu();
```

Beschreibung

Diese DVD-Methode zeigt das Stammmenü an.

Parameter

Keiner

Beispiel

Die folgende Anweisung zeigt das Stammmenü an:

```
-- Lingo syntax
member(1).rootMenu()

// JavaScript syntax
member(1).rootMenu();
```

Siehe auch

DVD

rotate

Syntax

```
member(whichCastmember).node(whichNode).rotate(xAngle, yAngle, zAngle {, relativeTo})
member(whichCastmember).node(whichNode).rotate(rotationVector {, relativeTo})
member(whichCastmember).node(whichNode).rotate(position, axis, angle {, relativeTo})
transform.rotate(xAngle, yAngle, zAngle {, relativeTo})
transform.rotate(rotationVector {, relativeTo})
transform.rotate(position, axis, angle {, relativeTo})
```

Beschreibung

Dieser 3D-Befehl führt nach den aktuellen Positions-, Drehungs- und Skalierungsoffsets des Transformationsobjekts des Nodes bzw. des direkt referenzierten Transformationsobjekts eine Drehung durch. Die Drehung kann in Form von drei Winkelwerten angegeben werden, von denen jeder einen Drehwinkel um eine der drei Achsen definiert. Diese Winkel können explizit im Format *xAngle*, *yAngle* und *zAngle* oder anhand eines Vektors (*rotationVector*) angegeben werden, bei dem die *x*-Komponente des Vektors der Drehung um die x-Achse, die *y*-Komponente der Drehung um die y-Achse und die *z*-Komponente der Drehung um die z-Achse entspricht. Die Drehung kann auch als Drehung um eine beliebige Achse durch einen Punkt im Raum angegeben werden.

Parameter

xAngle Erforderlich beim Anwenden einer Drehung mithilfe von *x*-, *y*- und *z*-Achsen. Gibt den Winkel der Drehung um die *X*-Achse an.

yAngle Erforderlich beim Anwenden einer Drehung mithilfe von *x*-, *y*- und *z*-Achsen. Gibt den Winkel der Drehung um die *Y*-Achse an.

zAngle Erforderlich beim Anwenden einer Drehung mithilfe von *x*-, *y*- und *z*-Achsen. Gibt den Winkel der Drehung um die *Z*-Achse an.

rotationVector Erforderlich beim Anwenden einer Drehung mithilfe eines Vektors. Gibt den Vektor an, der die anzuwendende Winkel enthält.

position Erforderlich beim Anwenden einer Drehung um eine zufällige Achse, die durch einen Punkt im Raum verläuft. Gibt die Position im Raum an.

axis Erforderlich beim Anwenden einer Drehung um eine zufällige Achse, die durch einen Punkt im Raum verläuft. Gibt die Achse an, die durch die mittels *position* angegebene Position verläuft.

angle Erforderlich beim Anwenden einer Drehung um eine zufällige Achse, die durch einen Punkt im Raum verläuft. Gibt den Drehwinkel um die Achse *axis* an.

relativeTo Optional. Gibt an, welche Koordinatensystemachsen zur Durchführung der gewünschten Drehungsänderungen verwendet werden. Der Parameter *relativeTo* kann folgende Werte aufweisen:

- `#self` wendet die Inkrementwerte bezogen auf das lokale Koordinatensystem des Nodes (die für das Modell beim Authoring angegebenen x-, y- und z-Achsen) an. Dieser Wert wird als Standardwert verwendet, wenn Sie den Befehl `rotate` zusammen mit einem Node-Bezug benutzen und den Parameter *relativeTo* nicht angeben.

- `#parent` wendet die Inkrementwerte bezogen auf das Koordinatensystem des Parent-Objekts des Nodes an. Dieser Wert wird als Standardwert verwendet, wenn Sie den Befehl `rotate` zusammen mit einem Transformationsbezug benutzen und den Parameter *relativeTo* nicht angeben.
- `#world` wendet die Inkrementwerte bezogen auf das Koordinatensystem der Welt an. Wenn das Parent-Objekt eines Modells die Welt ist, entspricht dies dem Wert `#parent`.
- Mit `nodeReference` können Sie einen Node angeben, auf dem die Drehung basieren soll. Der Befehl wendet die Inkrementwerte dann bezogen auf das Koordinatensystem des angegebenen Node an.

Beispiel

Im folgenden Beispiel dreht sich das Modell "Mond" zuerst um die eigene z-Achse (d. h. auf der Stelle) und dann um seinen Parent-Node, das Modell "Erde" (Umlaufbahn des Mondes um die Erde):

```
member("Scene").model("Moon").rotate(0,0,15)
member("Scene").model("Moon").rotate(vector(0, 0, 5), member("Scene").model("Moon"))
```

Im folgenden Beispiel dreht sich das Modell „Ball“ um die vom Modell „Stange“ belegte Position im Raum. Der Effekt besteht darin, dass das Modell „Ball“ eine Umlaufbahn um den „Pol“ entlang der x-y-Ebene beschreibt.

```
polePos = member("3d Scene").model("Pole").worldPosition
member("3d Scene").model("Ball").rotate(polePos, vector(0,0,1), 5, #world)
```

Siehe auch

pointAt, preRotate, rotation (Transformation), rotation (engraver shader), rotation (backdrop and overlay)preScale(), transform (Eigenschaft)

run

Syntax

```
run(MUIObject)
```

Beschreibung

Dieser Befehl zeigt ein allgemeines modales Dialogfeld an, das aus einer Instanz des Xtras MUI erzeugt wurde.

Damit Director das Dialogfeld öffnen kann, definieren Sie es zuvor über den Befehl „Initialize“.

Um ein nicht modales Dialogfeld zu öffnen, verwenden Sie den Befehl `WindowOperation` mit der Option `#show`. Dieser Befehl ermöglicht die Ausführung weiteren Lingo-Codes in einem Film, während das nicht modale Dialogfeld geöffnet ist.

Beispiel

Diese Prozedur prüft, ob das Objekt „MUIObject“ vorhanden ist und zeigt, falls ja, ein auf „UIObject“ basierendes allgemeines Dialogfeld an:

```
--Lingo syntax
on runDialog
    global MUIObject
    if objectP(MUIObject) then
        run(MUIObject)
    end if
end
```

runMode

Syntax

```
_system.environmentPropList.runMode
```

Beschreibung

Diese Funktion gibt einen String zurück, aus dem der Modus hervorgeht, in dem der Film abgespielt wird. Mögliche Werte:

- `Author` – Der Film wird in Director abgespielt.
- `Projector` – Der Film wird als Projektor abgespielt.
- `BrowserPlugin` – Der Film wird als Shockwave Player-Plug-In oder in einer anderen Skriptumgebung (z. B. LiveConnect oder ActiveX) abgespielt.

Mit dem Operator `contains` lässt sich auf sichere Weise ermitteln, ob bestimmte Werte in dieser Eigenschaft enthalten sind. Dadurch werden Fehler vermieden und auch teilweise Entsprechungen ermöglicht.

Parameter

Keiner

Beispiel

Die folgende Anweisung bestimmt, ob externe Parameter verfügbar sind. Wenn ja, werden sie abgerufen.

```
--Lingo syntax
if _system.environmentPropList.runMode contains "Plugin" then
    -- decode the embed parameter
    if externalParamName(swURL) = swURL then
        put externalParamValue(swURL) into myVariable
    end if
end if

// JavaScript syntax
if (_system.environmentPropList.runMode.indexOf("Plugin") >=0) {
    // decode the embed parameter
    if (_player.externalParamName(swURL) == swURL) {
        myVariable = _player.externalParamValue(swURL);
    }
}
```

Siehe auch

`environmentPropList`, `platform`

save (Mixer)

Syntax

```
Mixer.save(filepath)
```

Beschreibung

Mit dieser Mixermethode wird der gemischte Sound in dem festgelegten Dateipfad gespeichert. Die Datei wird im Format WAV oder MP4 gespeichert.

Wenn diese Datei gespeichert wird, funktionieren andere Operationen des Mixers nicht, wie z. B. `play` und `pause`. Mit `mixer.stop()` lässt sich der Speichervorgang vorzeitig beenden.


Mit der Eigenschaft `mixer.isSaving` lässt sich der Status des Speichervorgangs prüfen.

Beispiele

```
-- Lingo
on mouseUp me
mixerRef.save("C:\audio.wav") -- Saves the mixer output at the given filepath as a WAV file.
end

// Javascript
function mouseUp(){
mixerRef.save("C:\audio.wav"); //Saves the mixer output at the given filepath as a WAV file.
}
```

Siehe auch

isSaving (Mixer), Mixer

Save (Soundobjekt)

Syntax

```
soundObject.save(filePath)
```

Beschreibung

Diese Soundobjektmethode speichert den Soundstream als Soundobjekt am angegebenen Speicherort. Die Datei wird im Format WAV oder MP4 gespeichert.

Während diese Datei gespeichert wird, funktionieren andere Operationen des Soundobjekts nicht, wie z. B. `play` und `pause` nicht. Mit der Funktion `soundObject.stop()` lässt sich der Speichervorgang stoppen.

Verwenden Sie die Eigenschaft `isSaving` des Soundobjekts, um zu testen, ob das Soundobjekt derzeit gespeichert wird.

Beispiele

```
--Lingo syntax
on mouseUp me
     soundObjRef.save("C:\audio.wav") -- Saves the sound object associated with
-- soundObjRef at the given filepath in the .wav format.
end

// JavaScript syntax
function mouseUp(){
soundObjRef.save("C:\audio.wav"); // Saves the sound object associated with
// soundObjRef at the given path in the .wav format.
}
```

Siehe auch

isSaving (Soundobjekt)

save castLib

Syntax

```
castLib(whichCast).save()
save castLib whichCast {,pathName&newFileName}
```

Beschreibung

Dieser Befehl speichert die vorgenommenen Besetzungsänderungen in der Originaldatei der Besetzung oder in einer neuen Datei. Für weitere Vorgänge oder für Besetzungsbezüge wird dann der gespeicherte Darsteller verwendet.

Dieser Befehl kann nicht bei komprimierten Dateien verwendet werden.

Der Befehl `save CastLib` unterstützt keine URLs als Dateiverweise.

Parameter

pathName&newFileName Optional. Gibt den Pfad und Dateinamen an, unter denen gespeichert werden soll. Wird der Parameter ausgelassen, muss die Originalbesetzung verknüpft sein.

Beispiel

Die folgende Anweisung bewirkt, dass Director die geänderte Version der Besetzung `Buttons` im gleichen Ordner in der neuen Datei `UpdatedButtons` speichert:

```
-- Lingo
castLib("Buttons").save(the moviePath & "UpdatedButtons.cst")

// Javascript
castLib("Buttons").save(_movie.path & "UpdatedButtons.cst") ;
```

Siehe auch

@ (Pfadname)

saveMovie()

Syntax

```
-- Lingo syntax
_movie.saveMovie({stringFilePath})

// JavaScript syntax
_movie.saveMovie({stringFilePath});
```

Beschreibung

Diese Filmmethode speichert den aktuellen Film.

Sofern der optionale Parameter *stringFilePath* angegeben ist, wird der Film in der angegebenen Datei gespeichert. Diese Methode kann nicht bei komprimierten Dateien verwendet werden. Der angegebene Dateiname muss die Erweiterung „.dir“ aufweisen.

Die `saveMovie()`-Methode unterstützt keine URLs als Dateiverweise.

Parameter

stringFilePath Optional. Eine Zeichenfolge, die den Pfad und Namen der Datei angibt, in der der Film gespeichert wird.

Beispiel

Die folgende Anweisung speichert den aktuellen Film in der Datei "Update":

```
-- Lingo syntax
_movie.saveMovie(_movie.path & "Update.dir")

// JavaScript syntax
_movie.saveMovie(_movie.path + "Update.dir");
```

Siehe auch

`Movie`

scale (Befehl)

Syntax

```
member(whichCastmember).node(whichNode).scale(xScale, yScale, zScale)
member(whichCastmember).node(whichNode).scale(uniformScale)
transform.scale(xScale, yScale, zScale)
transform.scale(uniformScale)
```

Beschreibung

Dieser 3D-Transformationsbefehl führt nach den aktuellen Positions-, Drehungs- und Skalierungsoffsets des Transformationsobjekts des referenzierten Node bzw. des direkt referenzierten Transformationsobjekts eine Skalierung durch. Die Skalierung muss als Gruppe von drei Skalierungen entlang der entsprechenden Achsen oder als einzelne Skalierung, die entlang aller Achsen gleichmäßig angewendet wird, angegeben werden. Sie können entweder die drei Einzelskalierungen mithilfe der Parameter *xScale*, *yScale* und *zScale* oder den einheitlichen Skalierungswert mithilfe des Parameters *uniformScale* angeben.

Ein Node kann ein Kamera-, Gruppen-, Licht- oder Modellobjekt sein. Der Befehl `scale` ändert zwar die Eigenschaft `transform.scale` des referenzierten Nodes, hat aber keine sichtbare Auswirkung auf Lichtquellen oder Kameras, da diese keine Geometrie enthalten.

Die angegebenen Skalierungswerte müssen größer als 0 sein.

Parameter

xScale Erforderlich beim Angeben von drei Skalierungen. Gibt die Skalierung entlang der *X*-Achse an.

yScale Erforderlich beim Angeben von drei Skalierungen. Gibt die Skalierung entlang der *Y*-Achse an.

zScale Erforderlich beim Angeben von drei Skalierungen. Gibt die Skalierung entlang der *Z*-Achse an.

uniformScale Erforderlich bei Angabe einer einzelnen, einheitlichen Skalierung. Gibt die einheitliche Skalierung an.

Beispiel

Im folgenden Beispiel wird zuerst die Eigenschaft transform.scale des Modells "Mond" angezeigt, dann das Modell mit dem Befehl scale skaliert und abschließend der daraus resultierende transform.scale-Wert angezeigt:

```
-- Lingo
put member("Scene").model("Moon").transform.scale

// Javascript
put member("Scene").getProp("model", i).transform.scale ;
-- vector( 1.0000, 1.0000, 1.0000)
```

Die folgende Anweisung skaliert das Modell „Pluto“ einheitlich entlang der drei Achsen um 0.5, so dass das Modell anschließend nur noch halb so groß ist:

```
-- Lingo
member("Scene").model("Pluto").scale(0.5)

// Javascript
member("Scene").getPropRef("model", a).scale(0.5);
// where a is the number index of the model "Pluto"
```

Die folgende Anweisung skaliert das Modell „Oval“ uneinheitlich – es erfolgt eine Skalierung entlang der *z*-Achse, nicht jedoch entlang der *x*- oder *y*-Achse.

```
-- Lingo
member("Scene").model("Pluto").scale(0.0, 0.0, 0.5)

// Javascript
member("Scene").getPropRef("model", a).scale(0.0, 0.0, 0.5);
// where a is the number index of the model "Pluto"
```

Siehe auch

transform (Eigenschaft), preScale(), scale (Transformation)

script()

Syntax

```
-- Lingo syntax
script(memberNameOrNum {, castNameOrNum})

// JavaScript syntax
script(memberNameOrNum {, castNameOrNum});
```

Beschreibung

Diese Top-Level-Funktion erstellt einen Verweis auf einen vorgegebenen Darsteller, der ein Skript enthält, und gibt optional die Besetzungsbibliothek an, in der der Darsteller enthalten ist.

Wenn der vorgegebene Darsteller kein Skript enthält oder nicht vorhanden ist, wird ein Fehler zurückgegeben.

Parameter

memberNameOrNum (erforderlich). Eine Zeichenfolge, die den Namen des Darstellers angibt, der ein Skript enthält, oder eine Ganzzahl, die die Indexposition des Darstellers angibt, der ein Skript enthält.

castNameOrNum (optional). Eine Zeichenfolge, die den Namen der Besetzungsbibliothek angibt, die den Darsteller *memberNameOrNum* enthält, oder eine Ganzzahl, die die Indexposition der Besetzungsbibliothek angibt, die den Darsteller *memberNameOrNum* enthält. Wird der Parameter ausgelassen, durchsucht `script()` die erste Besetzungsbibliothek.

Beispiel

Nur in Lingo prüfen die folgenden Anweisungen, ob ein Child-Objekt eine Instanz des Parent-Skripts „Kriegerameise“ ist:

```
-- Lingo syntax
if (bugObject.script = script("Warrior Ant")) then
    bugObject.attack()
end if
```

Die folgende Anweisung setzt die Variable `actionMember` actionMember auf den Skriptdarsteller `Actions`:

```
-- Lingo syntax
actionMember = script("Actions")

// JavaScript syntax
var actionMember = script("Actions");
```

scrollByLine()

Syntax

```
-- Lingo syntax
memberObjRef.scrollByLine(amount)

// JavaScript syntax
memberObjRef.scrollByLine(amount);
```

Beschreibung

Dieser Befehl rollt den angegebenen Feld- oder Textdarsteller um eine angegebene Anzahl von Zeilen nach oben oder unten. Zeilen werden durch Zeilenschaltungen getrennt und nicht durch automatischen Zeilenumbruch.

Parameter

amount Erforderlich. Gibt die Anzahl der zu rollenden Zeilen an. Falls *amount* eine positive Zahl ist, rollt das Feld nach unten. Falls *amount* eine negative Zahl ist, roll das Feld nach oben.

Beispiel

Die folgende Anweisung rollt den Felddarsteller "Aktuelle Nachrichten" um fünf Zeilen nach unten:

```
--Lingo syntax
member("Today's News").scrollbyline(5)

// JavaScript syntax
member("Today's News").scrollbyline(5);
```

Die folgende Anweisung rollt den Felddarsteller „Aktuelle Nachrichten“ um fünf Zeilen nach oben:

```
--Lingo syntax
member("Today's News").scrollByLine(-5)

// JavaScript syntax
member("Today's News").scrollByLine(-5);
```

scrollByPage()

Syntax

```
-- Lingo syntax
memberObjRef.scrollByPage(amount)

// JavaScript syntax
memberObjRef.scrollByPage(amount);
```

Beschreibung

Dieser Befehl rollt den angegebenen Feld- oder Textdarsteller um eine angegebene Anzahl von Seiten nach oben oder unten. Eine Seite entspricht der Anzahl an Textzeilen, die auf dem Bildschirm zu sehen ist.

Parameter

amount Erforderlich. Gibt die Anzahl der zu rollenden Seiten an. Falls *amount* eine positive Zahl ist, rollt das Feld nach unten. Falls *amount* eine negative Zahl ist, roll das Feld nach oben.

Beispiel

Die folgende Anweisung rollt den Felddarsteller "Aktuelle Nachrichten" um eine Seite nach unten:

```
--Lingo syntax
member("Today's News").scrollbypage(1)

// JavaScript syntax
member("Today's News").scrollbypage(1);
```

Die folgende Anweisung rollt den Felddarsteller „Aktuelle Nachrichten“ um eine Seite nach oben:

```
--Lingo syntax
member("Today's News").scrollbypage(-1)

// JavaScript syntax
member("Today's News").scrollbypage(-1);
```

Siehe auch

scrollTop

seek()

Syntax

```
-- Lingo syntax
memberOrSpriteObjRef.seek(milliseconds)

// JavaScript syntax
memberOrSpriteObjRef.seek(milliseconds);
```

Beschreibung

Diese Methode für RealMedia-Sprites und -Darsteller setzt die Wiedergabeposition des Medienstreams auf die angegebene Anzahl von Millisekunden ab Streambeginn.Der Wert von `mediaStatus` ändert sich normalerweise in `#seeking` und dann in `#buffering`.

Mit dieser Methode können Sie die Wiedergabe an einer anderen Stelle im RealMedia-Stream starten oder im Stream vorwärts und rückwärts springen. Die im Parameter *milliseconds* angegebene Ganzzahl ist die Anzahl von Millisekunden ab Beginn des Streams. Um rückwärts zu springen, geben Sie einen niedrigeren Wert an, aber keine negative Zahl.

Wird der Befehl `seek` aufgerufen, wenn der `mediaStatus`-Wert `#paused` lautet, wird der Stream neu gepuffert und an der durch `seek` angegebenen neuen Stelle erneut auf `#paused` gesetzt. Wird `seek` aufgerufen, wenn der `mediaStatus`-Wert `#playing` lautet, wird der Stream neu gepuffert, und die Wiedergabe beginnt automatisch an der neuen Position im Stream. Wird `seek` aufgerufen, wenn der `mediaStatus`-Wert `#closed` lautet, passiert gar nichts.

Wenn Sie einen Millisekundenwert eingeben, der höher ist als der Wert der Eigenschaft `duration`, wird das ganzzahlige Argument auf den Bereich zwischen 0 und der Streamdauer zugeschnitten. Bei einem RealMedia-Sprite mit streamendem Live-Inhalt kann der Befehl „seek“ nicht verwendet werden.

Die Anweisung `x.seek(n)` ist mit `x.currentTime = n` identisch; beide Aufrufe führen zu einer Neupufferung des Streams.

Parameter

milliseconds Erforderlich. Eine Ganzzahl, die die Anzahl der Millisekunden ab Streambeginn angibt.

Beispiel

Die folgenden Beispiele setzen die aktuelle Wiedergabeposition des Streams auf 10.000 Millisekunden (10 Sekunden):

```
-- Lingo syntax
sprite(2).seek(10000)
member("Real").seek(10000)

// JavaScript syntax
sprite(2).seek(10000);
member("Real").seek(10000);
```

Siehe auch

duration (RealMedia, SWA), currentTime (RealMedia), play() (RealMedia, SWA, Windows Media), pause() (RealMedia, SWA, Windows Media), stop() (RealMedia, SWA, Windows Media), mediaStatus (RealMedia, Windows Media)

seek(mSec) (MP4Media/FLV)

Syntax

```
sprite(1).seek(3000) -- Seeks to the third second in the video.
member(1).seek(3000) -- Seeks to the third second of the video.
```

Beschreibung

Diese MP4Media/FLV-Sprite-Methode ruft eine bestimmte Stelle des Videos auf. Die aufzusuchende Stelle wird in Millisekunden angegeben.

Parameter

Parameter	Beschreibung
mSec	Erforderlich. Dieser Parameter ist ein ganzzahliger Wert, der die gesuchte Stelle in Millisekunden ab Anfang des MP4-Videostreams beschreibt.

Beispiele

Die folgenden Beispiele setzen die aktuelle Wiedergabeposition des Streams auf 10.000 Millisekunden (10 Sekunden):

```
-- Lingo syntax
sprite(2).seek(10000)
member("MP4Media/FLV").seek(10000)

// JavaScript syntax
sprite(2).seek(10000);
member("MP4Media/FLV").seek(10000);
```

seek (Soundobjekt)

Syntax

```
so.seek(<milliseconds>)
```

Beschreibung

Mit dieser Soundobjektmethode wird eine bestimmte Position (in Millisekunden) innerhalb der Audiodatei gesucht.

Beispiele

```
-- Lingo syntax
on mouseUp me
soundObjectRef.seek = 7000  --  Seeks the seventh second within the audio.
end

// JavaScript syntax
function mouseUp(){
soundobjectref.seek= 7000  // Seeks the seventh second within the audio.
}
```

selectAtLoc()

Syntax

```
-- Lingo syntax
dvdObjRef.selectAtLoc(point(x, y))

// JavaScript syntax
dvdObjRef.selectAtLoc(point(x, y));
```

Beschreibung

Diese DVD-Methode verschiebt den Fokus auf die Schaltfläche unter einem bestimmten Punkt.

Diese Methode hat dieselbe Funktionalität wie ein über der Schaltfläche verharrender Mauszeiger.

Parameter

point(x, y) Erforderlich. Ein Punkt in Bühnenkoordinaten, der die Position angibt, an der eine Schaltfläche den Fokus erhält.

Beispiel

Die folgende Anweisung verschiebt den Fokus auf die Schaltfläche unter einem bestimmten Punkt.

```
-- Lingo syntax
member(10).selectAtLoc(point(50, 75))

// JavaScript syntax
member(10).selectAtLoc(point(50, 75));
```

Siehe auch

DVD

selectButton()

Syntax

```
-- Lingo syntax
dvdObjRef.selectButton(intButton)

// JavaScript syntax
dvdObjRef.selectButton(intButton);
```

Beschreibung

Diese DVD-Methode aktiviert eine angegebene Schaltfläche.

Bei Erfolg gibt diese Methode `0` zurück.

Hinweis: *Diese Methode wird auf Mac®-Intel® nicht unterstützt.*

Parameter

intButton Erforderlich. Eine Ganzzahl, die die Schaltfläche angibt, die den Fokus erhält.

Beispiel

Die folgende Anweisung aktiviert Schaltfläche 5:

```
-- Lingo syntax
sprite(11).selectButton(5)

// JavaScript syntax
sprite(11).selectButton(5);
```

Siehe auch

DVD

selectButtonRelative()

Syntax

```
-- Lingo syntax
dvdObjRef.selectButtonRelative(direction)

// JavaScript syntax
dvdObjRef.selectButtonRelative(direction);
```

Beschreibung

Diese DVD-Methode aktiviert eine Schaltfläche relativ zur aktuellen Schaltflächenposition im Menü.

Parameter

direction Erforderlich. Ein Symbol (Lingo) oder eine Zeichenfolge (JavaScript-Syntax), die die Richtung der Bewegung von der aktuellen Schaltflächenposition aus angibt. Gültige Werte sind `left` oder `right`.

Hinweis: *Diese Methode wird auf einem Macintosh®-Intel® nicht unterstützt.*

Beispiel

Die folgende Anweisung gibt die Schaltfläche links neben der aktuellen Schaltfläche an:

```
-- Lingo syntax
member(12).member.selectButtonRelative(#left)

// JavaScript syntax
member(12).member.selectButtonRelative("left");
```

Siehe auch

DVD

selection() (Funktion)

Syntax

```
the selection
```

Beschreibung
Diese Funktion gibt einen String zurück, der den hervorgehobenen Teil des aktuellen bearbeitbaren Feldes enthält. Diese Funktion ist nützlich, wenn Sie testen möchten, was ein Benutzer in einem Feld ausgewählt hat.

Die Funktion `selection` gibt lediglich an, welcher String ausgewählt ist. Sie können `selection` nicht zur Auswahl des Strings selbst verwenden.

Parameter
Keiner

Beispiel
Die folgende Anweisung prüft, ob Zeichen ausgewählt wurden, und zeigt die Hinweismeldung „Bitte wählen Sie ein Wort aus“ an, falls dies nicht der Fall ist::

```
if the selection = EMPTY then alert "Please select a word."
```

Siehe auch
`selStart`, `selEnd`

sendAllSprites()

Syntax

```
-- Lingo syntax
_movie.sendAllSprites(stringEventMessage {, args})

// JavaScript syntax
_movie.sendAllSprites(stringEventMessage {, args});
```

Beschreibung
Diese Filmmethode sendet eine bestimmte Nachricht an alle Sprites, nicht nur das Sprite, das an dem Ereignis beteiligt war. Wie bei jeder anderen Nachricht wird auch hier die Nachricht an alle am Sprite angebrachten Skripts gesendet, es sei denn, Sie verwenden die `stopEvent()`-Methode.

Das beste Resultat erhalten Sie, wenn Sie die Nachricht mit der `sendSprite()`-Methode nur an die Sprites senden, die die Nachricht korrekt verarbeiten können. Es tritt zwar kein Fehler auf, wenn die Nachricht an alle Sprites gesendet wird, doch die Leistung wird dadurch eventuell beeinträchtigt. Es kann auch zu Problemen kommen, wenn unterschiedliche Sprites die gleiche Prozedur in einem Verhalten aufweisen. Vermeiden Sie also Konflikte, indem Sie Nachrichten, die übertragen werden, eindeutige Namen geben.

Nachdem die Nachricht an alle Verhalten gesendet wurde, folgt das Ereignis der regulären Nachrichtenhierarchie: Darstellerskripts, Bildskripts, gefolgt von Filmskripts.

Wenn Sie die `sendAllSprites()`-Methode verwenden, beachten Sie Folgendes:

- Ersetzen Sie *stringEventMessage* durch die Nachricht.
- Ersetzen Sie *args* durch beliebige Argumente, die mit der Nachricht zusammen gesendet werden sollen.

Wenn kein Sprite ein angefügtes Verhalten mit der betreffenden Prozedur aufweist, gibt `sendAllSprites()FALSE` zurück.

Parameter

stringEventMessage Erforderlich. Eine Zeichenfolge, die die an alle Sprites zu sendende Nachricht angibt.

args Optional. Ein Argument bzw. Argumente, die zusammen mit der Nachricht gesendet werden sollen.

Beispiel

Die folgende Prozedur sendet die benutzerdefinierte Nachricht `allSpritesShouldBumpCounter` und das Argument "2" an alle Sprites, wenn der Benutzer mit der Maus klickt:

```
-- Lingo syntax
on mouseDown me
    _movie.sendAllSprites(#allspritesShouldBumpCounter, 2)
end

// JavaScript syntax
function mouseDown() {
    _movie.sendAllSprites("allspritesShouldBumpCounter", 2);
}
```

Siehe auch

`Movie`, `sendSprite()`, `stopEvent()`

sendEvent

Syntax

```
member(whichCastmember).sendEvent(#eventName, arg1, arg2,...)
```

Beschreibung

Dieser 3D-Befehl sendet ein Ereignis und eine beliebige Anzahl von Argumenten an alle Skripts, die zum Empfang des Ereignisses registriert sind. Verwenden Sie `registerForEvent()` oder `setCollisionCallback()`, um Skripts für Ereignisse zu registrieren.

Parameter

eventName Erforderlich. Gibt den Namen des zu sendenden Ereignisses an.

arg1, arg2, ... (erforderlich). Eins oder mehrere Argumente, die zusammen mit dem Ereignis *eventName* gesendet werden.

Beispiel

Die erste Zeile in diesem Beispiel erstellt eine Instanz eines Parent-Skripts namens tester. Die zweite Zeile bewirkt, dass die Prozedur der Skriptinstanz, `jumpPluto`, aufgerufen wird, wenn das Ereignis `#jump` gesendet wird. Die dritte Zeile registriert eine Filmskriptprozedur namens `jumpMars` als weitere Prozedur, die aufgerufen wird, wenn das Ereignis `#jump` gesendet wird. Die vierte Zeile sendet das Ereignis `#jump`. Die Prozedur `#jumpMars`, die sich in einem Filmskript befindet, und die Prozedur `#jumpPluto` werden zusammen mit allen anderen für das Ereignis `#jump` registrierten Prozeduren aufgerufen. Der Skriptinstanzenwert 0 bedeutet, dass Sie eine Prozedur eines Filmskripts registrieren, und keine Prozedur einer Verhaltensinstanz oder eines Child-Objekts eines Parent-Skripts.

```
t = new (script "tester")
member("scene").registerForEvent(#jump, #jumpPluto, t)
member("scene").registerForEvent(#jump, #jumpMars, 0)
member("scene").sendEvent(#jump)
```

Siehe auch

registerScript(), registerForEvent(), setCollisionCallback()

sendSprite()

Syntax

```
-- Lingo syntax
_movie.sendSprite(spriteNameOrNum, event {, args})

// JavaScript syntax
_movie.sendSprite(spriteNameOrNum, event {, args});
```

Beschreibung

Diese Filmmethode sendet eine Nachricht an alle an einem bestimmten Sprite angebrachten Skripts.

Nachrichten, die mit `sendSprite()` gesendet werden, gehen an sämtliche an einem Sprite angebrachten Skripts. Die Nachricht folgt dann der regulären Nachrichtenhierarchie: Darstellerskripts, Bildskripts, gefolgt von Filmskripts.

Wenn an das betreffende Sprite kein Verhalten mit der jeweiligen Prozedur angebracht ist, gibt `sendSprite()FALSE` zurück.

Parameter

spriteNameOrNum Erforderlich. Eine Zeichenfolge oder Ganzzahl, die den Namen oder die Nummer des Sprites angibt, das das Ereignis empfangen soll.

event Erforderlich. Ein Symbol oder eine Zeichenfolge, das bzw. die das an das angegebene Sprite zu sendende Ereignis angibt.

args Optional. Ein Argument bzw. Argumente, die zusammen mit der Nachricht gesendet werden sollen.

Beispiel

Die folgende Prozedur sendet die benutzerdefinierte Nachricht `bumpCounter` und das Argument "2" an Sprite 1, wenn der Benutzer klickt:

```
-- Lingo syntax
on mouseDown me
    _movie.sendSprite(1, #bumpCounter, 2)
end

// JavaScript syntax
function mouseDown() {
    _movie.sendSprite(1, "bumpCounter", 2);
}
```

Siehe auch

Movie

setAlpha()

Syntax

```
imageObject.setAlpha(alphaLevel)
imageObject.setAlpha(alphaImageObject)
```

Beschreibung

Diese Funktion setzt den Alphakanal eines Grafikobjekts auf eine flache Alphaebene (`alphaLevel`) oder ein vorhandenes *alphaImageObject*. *alphaLevel* muss eine Zahl zwischen 0 und 255 sein. Niedrigere Werte bewirken, dass die Grafik transparenter erscheint. Höhere Werte bewirken, dass die Grafik eher opak erscheint. Der Wert 255 hat die gleiche Wirkung wie der Wert Null. Damit *alphaLevel* eine Wirkung haben kann, muss `useAlpha()` des Grafikobjekts auf `TRUE` gesetzt sein.

Das Grafikobjekt muss eine Farbtiefe von 32 Bit aufweisen. Wenn Sie ein Alphagrafikobjekt angeben, muss dieses eine Farbtiefe von 8 Bit besitzen. Die Grafiken müssen die gleichen Abmessungen aufweisen. Wenn diese Bedingungen nicht erfüllt sind, hat `setAlpha()` keine Wirkung und gibt „FALSE“ zurück. Wenn die Funktion erfolgreich ist, gibt sie „TRUE“ zurück.

Beispiel

Die folgende Lingo-Anweisung macht die Grafik des Bitmapdarstellers "Vordergrund" opak und deaktiviert den Alphakanal. Mit dieser Methode lässt sich die Alphaebene einer Grafik mühelos entfernen:

```
member("Foreground").image.setAlpha(255)
member("Foreground").image.useAlpha = FALSE
```

Der folgende Lingo-Code ruft die Alphaebene aus dem Darsteller „Sonnenaufgang“ ab und stellt ihn in die Alphaebene des Darstellers „Sonnenuntergang“:

```
tempAlpha = member("Sunrise").image.extractAlpha()
member("Sunset").image.setAlpha(tempAlpha)
```

Siehe auch

`useAlpha`, `extractAlpha()`

setaProp

Syntax

```
setaProp list, listProperty, newValue
setaProp (childObject, listProperty, newValue)
list.listProperty = newValue
list[listProperty] = newValue
childObject.listProperty = newValue
```

Beschreibung

Dieser Befehl ersetzt den der Eigenschaft *listProperty* zugeordneten Wert durch den in *newValue* angegebenen Wert. Der `setaProp`-Befehl funktioniert für Eigenschaftslisten und Child-Objekte. Wenn Sie `setaProp` bei einer linearen Liste verwenden, erhalten Sie einen Skriptfehler.

- In Eigenschaftslisten ersetzt `setaProp` eine Eigenschaft aus der in *list* angegebenen Liste. Ist die Eigenschaft noch nicht in der Liste enthalten, fügt Lingo die neue Eigenschaft und den Wert hinzu.

- Bei Child-Objekten ersetzt `setaProp` die Eigenschaft des Child-Objekts. Ist die Eigenschaft noch nicht in der Liste enthalten, fügt Lingo die neue Eigenschaft und den Wert hinzu.
- Der Befehl `setaProp` kann auch `ancestor`-Eigenschaften einstellen.

Parameter

listProperty Erforderlich. Ein Symbol (nur Lingo) oder eine Zeichenfolge, das bzw. die den Namen der Eigenschaft angibt, deren Wert geändert wird.

newValue Erforderlich. Der neue Wert für die Eigenschaft *listProperty*.

Beispiel

Die folgenden Anweisungen erstellen eine Eigenschaftsliste und fügen das Element `#c:10` der Liste hinzu:

```
newList = [#a:1, #b:5]
put newList
-- [#a:1, #b:5]
setaProp newList, #c, 10
put newList
```

Mit Hilfe des Punktoperators können Sie den Eigenschaftswert einer bereits in der Liste enthaltenen Eigenschaft ohne Verwendung von `setaProp` modifizieren:

```
newList = [#a:1, #b:5]
put newList
-- [#a:1, #b:5]
newList.b = 99
put newList
-- [#a:1, #b:99]
```

Hinweis: *Um mit Hilfe des Punktoperators eine Eigenschaft zu manipulieren, muss sich die Eigenschaft bereits in der Liste, dem Child-Objekt oder dem Verhalten befinden.*

Siehe auch

`ancestor`, `property`, `. (Punkt-Operator)`

setAt

Syntax

```
setAt list, orderNumber, value
list[orderNumber] = value
```

Beschreibung

Dieser Befehl ersetzt das in *orderNumber* angegebene Element durch den in *value* angegebenen Wert aus der in *list* angegebenen Liste. Wenn *orderNumber* größer ist als die Anzahl der Elemente einer Eigenschaftsliste, gibt der Befehl `setAt` einen Skriptfehler zurück. Ist *orderNumber* größer als die Anzahl der Elemente in einer linearen Liste, erweitert Director die Leereinträge in der Liste, um die in *orderNumber* angegebene Anzahl von Positionen bereitzustellen.

Beispiel

Die folgende Prozedur ordnet der Liste [12, 34, 6, 7, 45] einen Namen zu, ersetzt das vierte Element in der Liste durch den Wert 10 und zeigt dann das Ergebnis im Nachrichtenfenster an:

```
--Lingo
on enterFrame
    set vNumbers = [12, 34, 6, 7, 45]
    setAt vnumbers, 4, 10
    put vNumbers
end enterFrame

// Javascript
function enterFrame
{
    vNumbers =list(12, 34, 6, 7, 45) ;
    vNumbers.setAt(4, 10) ;
    put (vNumbers);
}
```

Wenn die Prozedur ausgeführt wird, erscheint im Nachrichtenfenster Folgendes:

```
[12, 34, 6, 10, 45]
```

Der gleiche Vorgang kann mit Listenzugriff auf folgende Weise durchgeführt werden:

```
--Lingo
on enterFrame
    set vNumbers = [12, 34, 6, 7, 45]
    vnumbers[4] = 10
    put vNumbers
end enterFrame

// Javascript
function enterFrame
{
    vNumbers = list(12, 34, 6, 7, 45);
    vnumbers[4] = 10 ;
    put (vNumbers);
}
```

Wenn die Prozedur ausgeführt wird, erscheint im Nachrichtenfenster Folgendes:

```
[12, 34, 6, 10, 45]
```

Siehe auch

[ ] (Klammernzugriff)

setCallback()

Syntax

```
-- Lingo syntax
spriteObjRef.setCallback(actionScriptObject, ASEventName, #LingoHandlerName,
lingoScriptObject)

// JavaScript syntax
spriteObjRef.setCallback(actionScriptObject, ASEventName, #LingoHandlerName,
lingoScriptObject);
```

Beschreibung

Dieser Flash-Befehl kann als Sprite- oder globale Methode zum Definieren einer Lingo-Rückruffunktion für ein vom angegebenen Objekt generiertes Ereignis verwendet werden. Wenn das Ereignis im Objekt durch ActionScript ausgelöst wird, wird es zusammen mit allen übergebenen Argumenten an die angegebene Lingo-Prozedur weitergeleitet.

Wurde das ActionScript-Objekt ursprünglich in einem Flash-Sprite erstellt, verwenden Sie die Syntax *flashSpriteReference*. Wurde das Objekt ursprünglich global erstellt, verwenden Sie die globale Syntax.

Hinweis: *Wenn Sie keine Flash-Darsteller importiert haben, müssen Sie das Flash Asset-Xtra manuell zur Xtra-Liste des Films hinzufügen, damit globale Flash-Befehle ordnungsgemäß funktionieren. Xtra-Erweiterungen werden durch Auswahl von „Modifizieren" > „Film" > „Xtras" in die Xtra-Liste aufgenommen. Weitere Informationen zum Verwalten von Xtra-Erweiterungen für verteilte Filme finden Sie in den Themen des Director-Benutzerhandbuchs im Director-Hilfefenster.*

Parameter

actionScriptObject Erforderlich. Gibt das ActionScript-Objekt an, das das Ereignis *ASEventName* enthält.

ASEventName Erforderlich. Gibt das ActionScript-Ereignis an, das eintritt.

lingoHandlerName Erforderlich. Gibt die Lingo-Prozedur an, die das Ereignis *ASEventName* verarbeitet.

lingoScriptObject Erforderlich. Gibt das Lingo-Skriptobjekt an, das die Prozedur *lingoHandlerName* enthält.

Beispiel

Die folgende Anweisung legt eine Lingo-Prozedur namens `myOnStatus` im Lingo-Skriptobjekt "me" fest, die aufgerufen wird, wenn im Flash-Film in Sprite 3 ein `onStatus`-Ereignis durch das ActionScript-Objekt `tLocalConObject` generiert wird:

```
Lingo syntax
sprite(3).setCallback(tLocalConObject, "onStatus", #myOnStatus, me)

// JavaScript syntax
sprite(3).setCallback(tLocalConObject, "onStatus", symbol("myOnStatus"), me);
```

Die folgenden Anweisungen erstellen ein neues globales XML-Objekt sowie eine Rückrufprozedur zum Parsen der eingehenden XML-Daten. In der dritten Zeile wird eine XML-Datei geladen. Die Rückrufprozedur wird ebenfalls angegeben.

```
-- Lingo syntax
gXMLCB = newObject("XML")
setCallback( gXMLCB, "onData", #dataFound, 0 )
gXMLCB.load( "myfile.xml")

-- Callback handler invoked when xml data arrives
on dataFound me, obj, source
    obj.parseXML(source)
    obj.loaded = 1
    obj.onload(TRUE)
end dataFound

// JavaScript syntax
gXMLCB = newObject("XML");
setCallback( gXMLCB, "onData", symbol("dataFound"), 0 );
gXMLCB.load( "myfile.xml" );

// Callback handler invoked when xml data arrives
function dataFound(me, obj, source) {
    obj.parseXML(source);
    obj.loaded = 1;
    obj.onload(1);
}
```

Siehe auch

newObject(), clearAsObjects()

setCharSet

Syntax

```
FileIO.setCharSet(strCharSet)
```

Beschreibung

Diese FileIO Xtra-Methode weist FileIO an, den angebenen Zeichensatz bei allen Schreib-/Lesefunktionen für Strings zu verwenden.

Parameter

Parameter	Beschreibung	Standardwert
strCharSet	Legt die erforderliche Encoding-Übersetzung fest.	UTF-8

Beispiele

```
--Lingo syntax
FileIO.setCharSet("UTF-16")

//JavaScript syntax
FileIO.setCharSet("UTF-16");
```

Siehe auch

readChar(), readFile(), readLine(), readToken(), readWord(), writeChar(), writeString()

setCollisionCallback()

Syntax

```
member(whichCastmember).model(whichModel).collision.setCollisionCallback (#handlerName,
scriptInstance)
```

Beschreibung

Dieser 3D-Befehl für den Modifizierer „collision“ registriert eine bestimmte Prozedur in einer angegebenen Skriptinstanz, die aufgerufen wird, wenn das Modell *whichModel* an einer Kollision beteiligt ist.

Dieser Befehl funktioniert nur, wenn die Eigenschaft `collision.enabled` des Modells auf `TRUE` gesetzt ist. Das Standardverhalten richtet sich nach dem Wert von `collision.resolve`, kann aber mit dem Befehl `collision.resolveA` und/oder `collision.resolveB` überschrieben werden. Verwenden Sie nicht den Befehl `updateStage` in der angegebenen Prozedur.

Dieser Befehl ist für Kollisionen eine kürzere Alternative zum Befehl `registerScript`; das Ergebnis ist aber jeweils gleich. Dieser Befehl führt eine kleine Untermenge der Funktionalität des Befehls `registerScript` aus.

Parameter

handlerName Erforderlich. Gibt die Prozedur an, die aufgerufen wird, wenn ein Modell an einer Kollision beteiligt ist.

scriptInstance Erforderlich. Gibt die Skriptinstanz an, die die durch *handlerName* angegebene Prozedur enthält.

Beispiel

Die folgende Anweisung bewirkt, dass die Prozedur `#bounce` im Darsteller `colScript` aufgerufen wird, wenn das Modell `Sphere` mit einem anderen Modell zusammenstößt:

```
member("3d world").model("Sphere").collision.setCollisionCallback(#bounce,
member("colScript"))
```

Siehe auch

`collisionData`, `collision (modifier)`, `resolve`, `resolveA`, `resolveB`, `registerForEvent()`, `registerScript()`, `sendEvent`

setFilterMask()

Syntax

```
-- Lingo syntax
fileioObjRef.setFilterMask(stringMask)

// JavaScript syntax
fileioObjRef.setFilterMask(stringMask);
```

Beschreibung

Diese Datei-E/A-Methode (Fileio) legt die Filtermaske für das Feld *Dateityp* eines Dialogfelds so fest, dass der Dateityp bei Anzeige des Dialogfelds angegeben wird.

Parameter

stringMask Erforderlich. Eine Zeichenfolge, die die Filtermaske angibt.

Beispiel

```
-- Lingo syntax
objFileio = new xtra("fileio")
objFileio.openFile(stringFileName, intMode)
objFileio.setFilterMask(stringMask)

    // JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.openFile(stringFileName, intMode) ;
objFileio.setFilterMask(stringMask) ;
```

Siehe auch

Fileio

setFinderInfo()

Syntax

```
-- Lingo syntax
fileioObjRef.setFinderInfo(stringAttrs)

// JavaScript syntax
fileioObjRef.setFinderInfo(stringAttrs)
```

Beschreibung

Diese Datei-E/A-Methode (Fileio; nur Mac) legt die Finder-Informationen für eine offene Datei fest.

Parameter

stringAttrs Erforderlich. Eine Zeichenfolge, die die Finder-Informationen angibt.

Beispiel

```
-- Lingo syntax
objFileio = new xtra("fileio")
objFileio.openFile(stringFileName, intMode)
objFileio.setFinderInfo(stringAttrs)

// JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.openFile(stringFileName, intMode) ;
objFileio.setFinderInfo(stringAttrs) ;
```

Siehe auch

Fileio

setFlashProperty()

Syntax

```
-- Lingo syntax
spriteObjRef.setFlashProperty(targetName, #property, newValue)

// JavaScript syntax
spriteObjRef.setFlashProperty(targetName, #property, newValue);
```

Beschreibung

Diese Funktion ermöglicht Lingo den Aufruf der Flash-ActionScript-Funktion `setProperty()` für das angegebene Flash-Sprite. Mit der Funktion `setFlashProperty()` werden die Eigenschaften von Movieclips oder Stufen in einem Flash-Film festgelegt. Dies ist der Einstellung von Sprite-Eigenschaften in Director ähnlich.

Um eine globale Eigenschaft des Flash-Sprites festzulegen, übergeben Sie als *targetName* einen leeren String. Folgende globale Flash-Eigenschaften lassen sich einstellen: `#focusRect` und `#spriteSoundBufferTime`.

Eine Beschreibung dieser Eigenschaften finden Sie in der Flash-Dokumentation.

Parameter

targetName Erforderlich. Gibt den Namen des Movieclips oder der Stufe an, dessen/deren Eigenschaft Sie im angegebenen Flash-Sprite festlegen möchten.

property Erforderlich. Gibt den Namen der festzulegenden Eigenschaft an. Folgende Filmclip-Eigenschaften lassen sich festlegen: `#posX`, `#posY`, `#scaleX`, `#scaleY`, `#visible`, `#rotate`, `#alpha` und `#name`.

newValue Erforderlich. Gibt den neuen Wert an.

Beispiel

Die folgende Anweisung setzt den Wert der Eigenschaft `#rotate` des Movieclips `Star` im Flash-Darsteller von Sprite 3 auf 180.

```
-- Lingo syntax
sprite(3).setFlashProperty("Star", #rotate, 180)

// JavaScript syntax
sprite(3).setFlashProperty("Star", symbol("rotate"), 180);
```

Siehe auch

`getFlashProperty()`

setNewLineConversion()

Syntax

```
-- Lingo syntax
fileioObjRef.setNewLineConversion(intOnOff)

// JavaScript syntax
fileioObjRef.setNewLineConversion(intOnOff)
```

Beschreibung
Diese Datei-E/A-Methode (Fileio; nur Mac) gibt an, ob die automatische Umwandlung von Zeilenvorschubzeichen aktiviert oder deaktiviert ist. Diese Methode ist beim Lesen von Dateien mit Zeilenenden im UNIX-Format nützlich, ist aber beim Lesen von Dateien mit Zeilenenden in Windows- und Macintosh-Classic-Formaten nicht erforderlich.

Parameter
intOnOff Erforderlich. Eine Ganzzahl, die angibt, ob die automatische Umwandlung aktiviert oder deaktiviert ist. Gültige Werte sind 0 (aus) und 1 (ein). Der Standardwert ist 0 (aus).

Beispiel
```
-- Lingo syntax
objFileio = new xtra("fileio")
objFileio.openFile(stringFileName, intMode)
objFileio.setNewLineConversion(intOnOff)

    // JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.openFile(stringFileName, intMode) ;
objFileio.setNewLineConversion(intOnOff) ;
```

Siehe auch
Fileio

setPixel()

Syntax
```
-- Lingo syntax
imageObjRef.setPixel(x, y, colorObjOrIntValue)
imageObjRef.setPixel(point(x, y), colorObjOrIntValue)

// JavaScript syntax
imageObjRef.setPixel(x, y, colorObjOrIntValue);
imageObjRef.setPixel(point(x, y), colorObjOrIntValue);
```

Beschreibung
Diese Grafikmethode legt den Farbwert des Pixels an der angegebenen Position in einer vorgegebenen Grafik fest.

Wenn Sie mithilfe von `getPixel()` zahlreiche Pixel auf die Farbe eines anderen Pixels einstellen, sollten Sie Ganzzahlen verwenden, da dieses Verfahren schneller geht.

Zur Erzielung der optimalen Leistung für Farbobjekte sollte bei Grafiken mit einer Farbtiefe von 8 Bit oder niedriger ein indiziertes Farbobjekt und bei Grafiken mit einer Farbtiefe von 16 Bit oder höher ein RGB-Farbobjekt verwendet werden.

Diese Methode gibt „FALSE“ zurück, falls das angegebene Pixel außerhalb der betreffenden Grafik liegt.

Ein Beispiel für diese Methode in einem fertigen Film finden Sie im Film „Imaging“ im Ordner „Learning/Lingo“ im Director-Anwendungsordner.

Parameter

x Erforderlich beim Angeben eines Pixels mithilfe von `x`- und `y`-Koordinaten. Eine Ganzzahl, die die `x`-Koordinate des Pixels angibt.

Y Erforderlich beim Angeben eines Pixels mithilfe von `x`- und `y`-Koordinaten. Eine Ganzzahl, die die `y`-Koordinate des Pixels angibt.

`point`*(x, y)* Erforderlich beim Angeben eines Pixels mithilfe eines Positionspunkts. Ein Punkt, der das Pixel angibt.

colorObjOrIntValue Erforderlich beim Festlegen der Farbe auf ein Farbobjekt oder einen Ganzzahlwert. Ein Verweis auf ein Farbobjekt, das die Farbe des Pixels angibt, oder eine Ganzzahl, die den Farbwert des Pixels angibt.

Beispiel

Die folgende Anweisung legt die Farbe des Pixels an der Position (20, 20) im Darsteller „Grafik“ auf der Bühne auf „Rot“ fest.

```
-- Lingo
objImage = _movie.stage.image
objImage.setPixel(20, 20 , rgb(255,0,0))
put (objImage)

-- Javascript
var objImage = _movie.stage.image ;
objImage.setPixel(20, 20 , color(255,0,0)) ;
put (objImage) ;
```

Siehe auch

`color()`, `draw()`, `fill()`, `getPixel()`, `image()`

setPixels()

Syntax

```
Image.setPixels(bytearray, #symbolImageFormat)
Possible values form symImageFormat are:
#bgra8888 => 32 bit BGRA
#argb8888=> 32 bit ARGB
#rgba8888=> 32 bit RGBA
#rgb888=> 24bit RGB
#bgr888=> 24bit BGR
```

Beschreibung

Diese Bytearray-Methode kopiert den Inhalt eines Byte-Arrays in die Bilddaten.

Hinweis: *Es werden nur 32-Bit Lingo-Bildobjekte unterstützt.*

Beispiel

Mit dem folgenden Code-Abschnitt wird ein Bild mit roter Farbe gefüllt und der Alphawert auf 50 eingestellt.

```
--Lingo syntax
on mouseUp me
i32=image(128,128,32) -- Creating a 32 bit image
i32.useAlpha=1 -- Setting the Alpha to 1
ba=i32.getPixels(#RGBA8888) -- Getting the pixels of the image using 32 bit RGBA format
ba2=byteArray(128*128*4,0) -- Creating the bytearray
i=0
repeat with r=1 to 128
repeat with c=1 to 128
ba2[i+1]=255 -- Value for RED
ba2[i+2]=0 -- Value for GREEN
ba2[i+3]=0 -- Value for BLUE
ba2[i+4]=50 -- Value for Alpha
i=i+4
end repeat
end repeat
i32.setPixels(ba2,#RGBA8888) -- Setting the pixels of the image using 32 bit RGBA format
member("test").image=i32
end
//JavaScript Syntax
function mouseUp(me)
{
var i=0;
i32=image(128,128,32);
i32.useAlpha=1;
ba=i32.getPixels(symbol("RGBA8888"));
ba2=byteArray(128*128*4,0);
i=0;
for(r=1;r<=128;r++)
{
for(c=1;c<=128;c++)
{
ba2[i+1]=255;
ba2[i+2]=0;
ba2[i+3]=0;
ba2[i+4]=50;
i=i+4;
}
}
i32.setPixels(ba2,symbol("RGBA8888"));
member("test").image=i32;
}
```

Siehe auch

getPixels()

setPlayList()

Syntax

```
-- Lingo syntax
soundChannelObjRef.setPlayList(linearListOfPropLists)

// JavaScript syntax
soundChannelObjRef.setPlayList(linearListOfPropLists);
```

Beschreibung

Diese Soundkanalfunktion legt die Abspielliste eines Soundkanals fest oder setzt sie zurück.

Diese Methode eignet sich zum gleichzeitigen Einreihen mehrerer Sounds in eine Warteschlange.

Ein Beispiel für `setPlaylist()` in einem fertigen Film finden Sie im Film „Sound Control“ im Ordner „Learning\\Lingo_“ im Director-Anwendungsordner.

Parameter

linearListOfPropLists Erforderlich. Eine lineare Liste von Eigenschaftslisten, die Parameter einer Abspielliste angibt. Sie können diese Parameter für jeden Sound angeben, den Sie in eine Warteschleife einreihen möchten:

Eigenschaft	Beschreibung
`#member`	Der in die Warteschlange einzureihende Sounddarsteller. Diese Eigenschaft ist erforderlich; alle anderen sind optional.
`#startTime`	Der Zeitpunkt im Sound, zu dem die Wiedergabe beginnt, in Millisekunden. Standardmäßig ist dies der Anfangspunkt des Sounds. Weitere Informationen finden Sie unter `startTime`.
`#endTime`	Der Zeitpunkt im Sound, zu dem die Wiedergabe endet, in Millisekunden. Standardmäßig ist dies das Ende des Sounds. Weitere Informationen finden Sie unter`endTime`.
`#loopCount`	Angabe, wie oft eine Schleife abgespielt werden soll, die in `#loopStartTime` und `#loopEndTime` definiert ist. DerStandardwert ist1. Weitere Informationen finden Sie unter `loopCount`.
`#loopStartTime`	Der Zeitpunkt im Sound, zu dem die Schleifenwiedergabe beginnt, in Millisekunden. Weitere Informationen finden Sie unter `loopStartTime`.
`#loopEndTime`	Der Zeitpunkt im Sound, zu dem die Schleifenwiedergabe endet, in Millisekunden. Weitere Informationen finden Sie unter `loopEndTime`.
`#preloadTime`	Angabe, wie viel Sound vor Abspielbeginn in den Zwischenspeicher geladen werden soll, in Millisekunden. Weitere Informationen finden Sie unter `preloadTime`.

Beispiel

Die folgende Prozedur reiht den Darsteller `introMusic` in eine Warteschlange ein, spielt ihn beginnend bei der 3-Sekunden-Marke in einer Schleife ab, die fünf Mal wiederholt wird (von der 8-Sekunden- bis zur 8,9-Sekunden-Marke), und hört an der 10-Sekunden-Marke auf.

```
-- Lingo syntax
on play Music
sound(2).setPlayList([]) -- empty the playlist sound(2).queue([#member:member("introMusic"),
#startTime:3000, #endTime:10000, #loopCount:5, #loopStartTime:8000, #loopEndTime:8900])
    sound(2).play()
end playMusic

// JavaScript syntax
function playMusic() {
sound(2).setPlayList(propList());
sound(2).queue(propList("member",member("introMusic"),"startTime",3000, "endTime",10000,
"loopCount",5, "loopStartTime",8000,"loopEndTime",8900)); sound(2).play();
}
```

Siehe auch

endTime (Soundkanal), getPlayList(), loopCount, loopEndTime (Soundkanal), loopStartTime, Member, member, preLoadTime, queue(), Sound Channel, startTime (Sound Channel)

setPosition()

Syntax

```
-- Lingo syntax
fileioObjRef.setPosition(intPosition)

// JavaScript syntax
fileioObjRef.setPosition(intPosition);
```

Beschreibung

Diese Datei-E/A-Methode (Fileio) legt die Position einer Datei fest.

Parameter

intPosition Erforderlich. Eine Ganzzahl, die die neue Position der Datei angibt.

Beispiel

```
-- Lingo syntax
objFileio = new xtra("fileio")
objFileio.openFile(stringFileName, intMode)
objFileio.setPosition(intPosition)

    // JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.openFile(stringFileName, intMode) ;
objFileio.setPosition(intPosition) ;
```

Siehe auch

Fileio

setPref()

Syntax

```
-- Lingo syntax
_player.setPref(stringPrefName, prefString)

// JavaScript syntax
_player.setPref(stringPrefName, prefString);
```

Beschreibung

Diese Player-Methode schreibt die in *prefString* angegebene Zeichenfolge in die in *stringPrefName* angegebene Datei auf dem lokalen Datenträger des Computers.

Das Argument *stringPrefName* muss ein gültiger Dateiname sein. Um sicherzustellen, dass der Dateiname auf allen Plattformen gültig ist, sollten Sie ihn auf acht alphanumerische Zeichen beschränken.

Nachdem die Methode `setPref()` ausgeführt wurde, wird, wenn der Film in einem Browser abgespielt wird, ein Ordner namens „Prefs“ im Supportordner für Plug-Ins erstellt. Die Methode `setPref()` schreibt nur in diesen Ordner.

Wird der Film hingegen in einem Projektor oder in Director abgespielt, wird der Ordner im selben Ordner erstellt, in dem sich auch die Anwendung befindet. Der Ordner erhält den Namen *Prefs*.

Diese Methode darf nicht verwendet werden, um auf schreibgeschützte Medien zu schreiben. Je nach Plattform und Version des Betriebssystems können Fehler oder andere Probleme auftreten.

Diese Methode führt keine komplexe Manipulation der Stringdaten oder deren Formatierung durch. Formatierungen und andere Bearbeitungsvorgänge sind in Zusammenhang mit `getPref()` durchzuführen; Sie können die Daten im Speicher manipulieren und die alte Datei mit `setPref()` überschreiben.

Von `setPref()` geschriebene Daten sind in einem Browser nicht privat; jeder Film mit Shockwave-Inhalt kann diese Informationen lesen und auf einen Server hochladen. Speichern Sie mit `setPref()` also keine vertraulichen Informationen.

Unter Windows schlägt die Ausführung der Methode `setPref()` fehl, wenn der Benutzer nur eingeschränkte Zugriffsrechte besitzt.

Ein Beispiel für `setPref()` in einem fertigen Film finden Sie im Film „Read and Write Text“ im Ordner „Learning\\Lingo_“ im Director-Anwendungsordner.

Parameter

stringPrefName Erforderlich. Eine Zeichenfolge, die den Namen der Datei angibt, in die die Zeichenfolge *prefString* geschrieben wird. Die Datei ist eine Standardtextdatei.

prefString Erforderlich. Die in die durch *stringPrefName* angegebene Datei zu schreibende Zeichenfolge.

Beispiel

Die folgende Prozedur speichert den Inhalt des Felddarstellers `Text Entry` in der Datei `DayWare`:

```
-- Lingo syntax
on mouseUp me
    _player.setPref("CurPrefs", member("Text Entry").text)
end

// JavaScript syntax
function mouseUp() {
    _player.setPref("CurPrefs", member("Text Entry").text);
}
```

Siehe auch

`getPref() (Player)`, `Player`

setPref() (Player)

Syntax

```
-- Lingo syntax
_player.setPref(stringPrefName, prefString)

// JavaScript syntax
_player.setPref(stringPrefName, prefString);
```

Beschreibung

Diese Player-Methode schreibt eine angegebene Zeichenfolge in eine vorgegebene Datei auf dem lokalen Datenträger des Computers. Die Datei ist eine Standardtextdatei.

Nachdem `setPref()` ausgeführt wurde, wird, wenn der Film in einem Browser abgespielt wird, ein Ordner namens „Prefs“ im Supportordner für Plug-Ins erstellt. Die Methode `setPref()` schreibt nur in diesen Ordner.

Dies sind die Speicherorte, an denen die Datei erstellt wird.

Windows Projector/Director ..\Documents and Settings\<username>\Application Data\Adobe\Director<version number>\Prefs\

Windows Shockwave ..\Dokumente und Einstellungen\Benutzername\Anwendungsdaten\Adobe\Shockwave Player Versionsnummer\Prefs\

MAC Projector ~/Library/Application Support/Adobe/Director Versionsnummer/Prefs

MAC Shockwave ~/Library/Application Support/Adobe/Shockwave Player Versionsnummer/Prefs

Diese Methode darf nicht verwendet werden, um auf schreibgeschützte Medien zu schreiben. Je nach Plattform und Version des Betriebssystems können Fehler oder andere Probleme auftreten.

Von `setPref()` geschriebene Daten sind in einem Browser nicht privat; jeder Film mit Shockwave-Inhalt kann diese Informationen lesen und auf einen Server hochladen. Speichern Sie mit `setPref()` also keine vertraulichen Informationen.

Unter Windows schlägt die Ausführung von `setPref()` fehl, wenn der Benutzer nur eingeschränkte Zugriffsrechte besitzt.

Ein Beispiel für `setPref()` in einem fertigen Film finden Sie im Film „Read and Write Text“ im Ordner „Learning\\Lingo_“ im Director-Anwendungsordner.

Parameter

prefName Erforderlich. Eine Zeichenfolge, die die Datei angibt, in die geschrieben werden soll. Der Parameter *prefName* muss ein gültiger Dateiname sein. Um sicherzustellen, dass der Dateiname auf allen Plattformen gültig ist, sollten Sie ihn auf acht alphanumerische Zeichen beschränken.

prefValue Erforderlich. Eine Zeichenfolge, die den Text angibt, der in die Datei *prefName* geschrieben werden soll.

Beispiel

Die folgende Prozedur speichert den Inhalt des Felddarstellers `Text Entry` in der Datei `DayWare`:

```
-- Lingo syntax
on mouseUp me
    _player.setPref("DayWare", member("Text Entry").text)
end

// JavaScript syntax
function mouseUp() {
    _player.setPref("DawWare", member("Text Entry").text);
}
```

Siehe auch

`getPref() (Player)`, `Player`

setProp

Syntax

```
setProp list, property, newValue
list.listProperty = newValue
list[listProperty] = newValue
```

Beschreibung

Dieser Befehl ersetzt in einer Liste den einer angegebenen Eigenschaft zugewiesenen Wert durch einen neuen Wert. Enthält die Liste die angegebene Eigenschaft nicht, gibt `setProp` einen Skriptfehler zurück.

Der Befehl `setProp` funktioniert nur bei Eigenschaftslisten. Wenn Sie `setProp` bei einer linearen Liste verwenden, erhalten Sie einen Skriptfehler.

Dieser Befehl ist vergleichbar mit dem Befehl `setaProp`; allerdings gibt `setProp` einen Fehler zurück, wenn die Eigenschaft nicht bereits in der Liste enthalten ist.

Parameter

property Erforderlich. Ein Symbol (nur Lingo) oder eine Zeichenfolge, das bzw. die die Eigenschaft angibt, deren Wert durch *newValue* ersetzt wird.

newValue Erforderlich. Der neue Wert für die durch *property* angegebene Eigenschaft.

Beispiel

Die folgende Anweisung ändert den Wert, der der Eigenschaft age in der Eigenschaftsliste x zugewiesen wurde, in 11:

```
--Lingo
setProp x, #age, 11

// Javascript
x[age] = 11
```

Mit Hilfe des Punktoperators können Sie den Eigenschaftswert einer bereits in der Liste enthaltenen Eigenschaft genau wie oben angegeben modifizieren:

```
x.age = 11
```

Siehe auch

setaProp

setScriptList()

Syntax

```
spriteReference.setScriptList(scriptList)
sprite(whichSprite).setScriptList(scriptList)
```

Beschreibung

Dieser Befehl stellt die `scriptList` des betreffenden Sprites ein. Die `scriptList` gibt an, welche Skripts am Sprite angebracht sind und wie die Einstellungen der einzelnen Skripteigenschaften lauten. Durch Einstellen dieser Liste können Sie die an einem Sprite angebrachten Verhalten oder die Verhaltenseigenschaften ändern.

Die Liste setzt sich folgendermaßen zusammen:

```
[ [ (whichBehaviorMember), " [ #property1: value, #property2: value, . . . ] ",
[(whichBehaviorMember), " [ #property1: value, #property2: value, . . . ] " ] ]
```

Dieser Befehl kann nicht während der Drehbuchaufzeichnung verwendet werden. Verwenden Sie `setScriptList()` für Sprites, die bei der Drehbuchaufzeichnung hinzugefügt werden, nachdem die Drehbuchaufzeichnung beendet ist.

Parameter

scriptList Erforderlich. Gibt die Skriptliste für ein vorgegebenes Sprite an.

Siehe auch

scriptList, value(), string()

settingsPanel()

Syntax

```
-- Lingo syntax
spriteObjRef.settingsPanel({integerPanelIndex})

// JavaScript syntax
spriteObjRef.settingsPanel({integerPanelIndex});
```

Beschreibung

Dieser Flash-Sprite-Befehl ruft das Dialogfeld mit den Flash Player-Einstellungen auf und blendet die angegebene Registerkarte ein. Hierbei handelt es sich um das gleiche Dialogfeld, das erscheint, wenn der Benutzer mit der rechten Maustaste (Windows) bzw. bei gedrückter Ctrl-Taste (Mac) auf einen Flash-Film im Browser klickt.

Das Dialogfeld „Einstellungen“ wird nicht angezeigt, wenn das Rechteck des Flash-Sprites zu klein ist.

Zum Emulieren des Flash Players durch Aufruf des Einstellungsbedienfelds, wenn der Benutzer mit der rechten Maustaste (Windows) bzw. bei gedrückter Ctrl-Taste (Mac) klickt, können Sie diesen Befehl in einer `mouseDown`-Prozedur verwenden, die das Vorhandensein der Eigenschaft `rightMouseDown` oder `controlDown` überprüft.

Zum Emulieren des Flash Players durch Aktivieren des Einstellungsbedienfelds in einem in einem Browser ablaufenden Director-Film müssen Sie zuerst das Shockwave Player-Kontextmenü deaktivieren, das erscheint, wenn der Benutzer mit der rechten Maustaste (Windows) bzw. bei gedrückter Ctrl-Taste (Mac) auf einen Film mit Shockwave-Inhalt im Browser klickt. Weitere Informationen zum Deaktivieren dieses Menüs finden Sie in den Themen des Director-Benutzerhandbuchs im Director-Hilfefenster.

Parameter

integerPanelIndex Optional. Gibt an, welche Registerkarte beim Öffnen des Dialogfelds aktiviert werden soll. Gültige Werte sind 0, 1, 2 und 3.Beim Wert 0 wird die Registerkarte "Zugriffsschutz" eingeblendet, beim Wert 1 die Registerkarte "Lokaler Speicher", beim Wert 2 die Registerkarte "Mikrofon" und beim Wert 3 die Registerkarte "Kamera". Der Standardwert für diesen Parameter ist 0.

Beispiel

Die folgende Anweisung öffnet das Flash-Einstellungsbedienfeld und zeigt die Registerkarte „Lokaler Speicher“ an:

```
-- Lingo syntax
sprite(3).settingsPanel(1)

// JavaScript syntax
sprite(3).settingsPanel(1);
```

Siehe auch

`on mouseDown (Ereignisprozedur)`, `rightMouseDown`, `controlDown`

setTrackEnabled()

Syntax

```
-- Lingo syntax
spriteObjRef.setTrackEnabled(whichTrack, trueOrFalse)

// JavaScript syntax
spriteObjRef.setTrackEnabled(whichTrack, trueOrFalse);
```

Beschreibung

Dieser Befehl bestimmt, ob die angegebene Spur eines Digitalvideos zum Abspielen aktiviert ist.

- Wenn `setTrackEnabled` den Wert `TRUE` aufweist, ist die angegebene Spur aktiviert und wird abgespielt.
- Wenn `setTrackEnabled` den Wert `FALSE` aufweist, ist die angegebene Spur deaktiviert und ausgeschaltet. Für Videospuren bedeutet dies, dass sie nicht mehr auf dem Bildschirm aktualisiert werden.

Um zu testen, ob eine Spur bereits aktiviert wurde, testen Sie die Sprite-Eigenschaft `trackEnabled`.

Parameter
whichTrack Erforderlich. Gibt die zu testende Spur an.

trueOrFalse Erforderlich. Gibt an, ob die Spur im Digitalvideo zum Abspielen aktiviert ist (`TRUE`) oder nicht (`FALSE`).

Beispiel
Die folgende Anweisung aktiviert Spur 3 des dem Sprite-Kanal 8 zugeordneten Digitalvideos:

```
-- Lingo syntax
sprite(8).setTrackEnabled(3, TRUE)

// JavaScript syntax
sprite(8).setTrackEnabled(3, 1);
```

Siehe auch
`trackEnabled`

setVal()

Syntax
```
<Void> Matrix.setVal(whichRow, whichColumn, valueToSet)
```

Beschreibung
Matrixmethode. Legt den Wert des angegebenen Elements in der angegebenen Matrix fest.

Parameter
whichRow Erforderlich. Zeilennummer des Elements, dessen Wert festgelegt wird.

`whichColumn` Erforderlich. Spaltennummer des Elements, dessen Wert festgelegt wird.

valueToSet (erforderlich). Float. Erforderlicher Fließkommawert zum Festlegen der Zeile „Rowth“ und der Spalte „Columnth“ in der Matrix.

Beispiel
Die folgende Funktion erstellt eine Matrix mit 4 Zeilen und 5 Spalten und legt den Wert jedes Elements in der Matrix beliebig fest.

```
--Lingo
on randomMatrix()
     rows = 4
     cols = 5
     mat = newMatrix(rows,cols)
     repeat with i = 1 to rows
       repeat with j = 1 to cols
         mat.setVal(i,j,random(255))
       end repeat
     end repeat
     return mat
end
//Java Script
function randomMatrix()
{
     rows = 4;
     cols = 5;
     mat = newMatrix(rows,cols);
     for(i = 1 ; i <= rows;i++)
     for(j = 1 ; j <= rows;j++)
         mat.setVal(i,j,random(255));
     return mat;
}
```

Siehe auch

getVal(), numRows(), numColumns(), matrixAddition(), matrixMultiply(), matrixMultiplyScalar(), matrixTranspose(), newMatrix()

setVariable()

Syntax

```
-- Lingo syntax
spriteObjRef.setVariable(variableName, newValue)

// JavaScript syntax
spriteObjRef.setVariable(variableName, newValue);
```

Beschreibung

Diese Funktion stellt den Wert der angegebenen Variablen im angegebenen Flash-Sprite ein. Flash-Variablen wurden in Version 4 von Flash eingeführt.

Parameter

variableName Erforderlich. Gibt den Namen der Variablen an.

newValue Erforderlich. Gibt den neuen Wert der Variablen an.

Beispiel

Die folgende Anweisung legt den Wert der Variablen `currentURL` im Darsteller `Flash` in Sprite 3 fest. Der neue Wert von `currentURL` ist "http://www.adobe.com/software/flash/".

```
-- Lingo syntax
sprite(3).setVariable("currentURL", "http://www.adobe.com/software/flash/")

// JavaScript syntax
sprite(3).setVariable("currentURL", "http://www.adobe.com/software/flash/");
```

Siehe auch

`hitTest()`, `getVariable()`

showLocals()

Syntax

```
-- Lingo syntax
showLocals()
```

Beschreibung

Diese Top-Level-Funktion (nur Lingo) zeigt alle lokalen Variablen im Nachrichtenfenster an. Dieser Befehl kann nur innerhalb von Prozeduren oder Parent-Skripts verwendet werden, die lokale Variablen zur Anzeige enthalten. Alle im Nachrichtenfenster verwendeten Variablen sind automatisch global.

Lokale Variablen in einer Prozedur sind nach Ausführung der Prozedur nicht mehr verfügbar. Durch das Einfügen der Anweisung `showLocals()` in eine Prozedur werden alle lokalen Variablen in dieser Prozedur im Nachrichtenfenster angezeigt.

Dieser Befehl ist für das Debugging von Skripts sehr nützlich.

Parameter

Keiner

Siehe auch

`clearGlobals()`, `global`, `showGlobals()`

showProps()

Syntax

```
-- Lingo syntax
memberOrSpriteObjRef.showProps()

// JavaScript syntax
memberOrSpriteObjRef.showProps();
```

Beschreibung

Dieser Befehl zeigt eine Liste der aktuellen Eigenschaftseinstellungen eines Flash-Films, eines Vektordarstellers oder eines gerade abgespielten Sounds im Nachrichtenfenster an. Setzen Sie diesen Befehl nur beim Authoring ein; er funktioniert nicht in Projektoren oder in Filmen mit Shockwave-Inhalt.

Parameter
Keiner

Beispiel
Die folgende Prozedur akzeptiert den Namen einer Besetzung als Parameter, durchsucht diese Besetzung auf Flash-Filmdarsteller und zeigt den Darstellernamen, die Darstellernummer und die Darstellereigenschaften im Nachrichtenfenster an:

```
-- Lingo syntax
on ShowCastProperties(whichCast)
    repeat with i = 1 to castLib(whichCast).member.count
        castType = member(i, whichCast).type
        if (castType = #flash) OR (castType = #vectorShape) then
            put castType&&"cast member" && i & ":" && member(i, whichCast).name
            put RETURN
            member(i ,whichCast).showProps()
        end if
    end repeat
end

// JavaScript syntax
function ShowCastProperties(whichCast) {
    i = 1;
    while( i < (castLib(whichCast).member.count) +1 ) {
        castType = member(i, whichCast).type;
        if ((castType = "flash") || (castType = "vectorShape")) {
            trace (castType + " cast member " + i + ": " + member(i, whichCast).name) + \n;
            member(i ,whichCast).showProps();
i++;
        }
    }
}
```

Siehe auch
queue(), setPlayList()

showGlobals()

Syntax

```
-- Lingo syntax
_global.showGlobals()

// JavaScript syntax
_global.showGlobals();
```

Parameter
Keiner

Beschreibung
Diese globale Methode zeigt alle globalen Variablen im Nachrichtenfenster an.

Sie ist für das Debugging von Skripts sehr nützlich.

Beispiel

Die folgende Anweisung zeigt alle globalen Variablen im Nachrichtenfenster an:

```
-- Lingo syntax
on mouseDown
    _global.showGlobals()
end

// JavaScript syntax
function mouseDown() {
    _global.showGlobals();
}
```

Siehe auch

Global

shutDown()

Syntax

```
-- Lingo syntax
_system.shutDown()

// JavaScript syntax
_system.shutDown();
```

Beschreibung

Diese Systemmethode schließt alle offenen Anwendungen und schaltet den Computer aus.

Parameter

Keiner

Beispiel

Die folgende Anweisung prüft, ob der Benutzer Strg+S (Windows) bzw. Befehl+S (Macintosh) gedrückt hat, und fährt den Computer herunter, wenn dies der Fall ist:

Siehe auch

System

sin()

Syntax

```
sin(angle)
```

Beschreibung
Diese mathematische Funktion (nur Lingo) berechnet den Sinus des angegebenen Winkels. Der Winkel muss im Bogenmaß als Fließkommazahl ausgedrückt werden.

In JavaScript-Syntax wird die `sin()`-Funktion des Math-Objekts verwendet.

Parameter
angle Erforderlich. Gibt den Winkel an.

Beispiel
Die folgende Anweisung berechnet den Sinus von pi/2:

```
put sin (PI/2.0)
-- 1
```

Siehe auch
`PI`

sort

Syntax
```
list.sort()
sort list
```

Beschreibung
Dieser Befehl sortiert Elemente in einer Liste alphanumerisch.

- Handelt es sich bei der Liste um eine lineare Liste, werden die Werte in der Liste geordnet.
- Handelt es sich dabei um eine Eigenschaftsliste, werden die Eigenschaften alphabetisch geordnet.

Nachdem eine Liste einmal sortiert ist, wird diese Sortierreihenfolge auch dann beibehalten, wenn Sie mit dem Befehl `add` neue Variablen hinzufügen.

Parameter
Keiner

Beispiel
Die folgende Anweisung sortiert die Listenwerte (`Values`) [`#a: 1, #d: 2, #c: 3`] in alphanumerischer Reihenfolge. Das Ergebnis wird unterhalb der Anweisung angezeigt.

```
put values
-- [#a: 1, #d: 2, #c: 3]
values.sort()
put values
--[#a: 1, #c: 3, #d: 2]
```

sound()

Syntax

```
-- Lingo syntax
sound(intSoundChannel)

// JavaScript syntax
sound(intSoundChannel);
```

Beschreibung

Diese Top-Level-Funktion gibt einen Verweis auf einen angegebenen Soundkanal zurück.

Die Funktionalität dieser Methode ist mit der der Sound-Objektmethode `channel()` identisch.

Parameter

intSoundChannel Erforderlich. Eine Ganzzahl, die den Soundkanal angibt, auf den verwiesen werden soll.

Beispiel

Das folgende Beispiel weist den Soundkanal 1 einer Variablen music zu und spielt einen Sound ab.

```
-- Lingo syntax
music = sound(1)
music.play(member("waltz1"))

// JavaScript syntax
var music = sound(1);
music.play(member("waltz1"));
```

Siehe auch

`channel() (Sound)`, `Sound Channel`

sprite()

Syntax

```
-- Lingo syntax
sprite(nameOrNum)

// JavaScript syntax
sprite(nameOrNum);
```

Beschreibung

Diese Top-Level-Funktion gibt einen Verweis auf ein angegebenes Sprite im Drehbuch zurück.

Wenn die Filmeigenschaft `scriptExecutionStyle` auf den Wert 9 festgelegt ist, gibt der Aufruf von `sprite("foo")`, da kein Sprite mit diesem Namen vorhanden ist, einen Verweis auf Sprite 1 zurück. Wenn die Filmeigenschaft `scriptExecutionStyle` auf den Wert 10 festgelegt ist, gibt der Aufruf von `sprite("foo")`, da kein Sprite mit diesem Namen vorhanden ist, bei Verwendung von Lingo den Wert `VOID` und bei Verwendung von JavaScript-Syntax den Wert `undefined` zurück.

Parameter

nameOrNum Erforderlich. Eine Zeichenfolge oder Ganzzahl, die den Namen oder die Indexposition des Sprites angibt.

Beispiel

Die folgende Anweisung setzt die Variable `thisSprite` auf das Sprite `Cave`:

```
-- Lingo syntax
thisSprite = sprite("Cave")

// JavaScript syntax
var thisSprite = sprite("Cave");
```

Siehe auch

`Sprite Channel`

spriteSpaceToWorldSpace

Syntax

```
sprite(whichSprite).camera.spriteSpaceToWorldSpace(loc)
sprite(whichSprite).camera(index).spriteSpaceToWorldSpace(loc)
```

Beschreibung

Dieser 3D-Befehl gibt eine Weltposition auf der Projektionsebene der angegebenen Kamera zurück, die einer Stelle im referenzierten Sprite entspricht.

Die Projektionsebene wird durch die x- und y-Achse der Kamera definiert und befindet sich so weit vor der Kamera, dass ein Pixel einer Welteinheit entspricht. Die Projektionsebene wird zur Sprite-Anzeige auf der Bühne verwendet.

Die Syntax `camera.spriteSpaceToWorldSpace()` ist eine Kurzform für `camera(1).spriteSpaceToWorldSpace()`.

Alle vom referenzierten Sprite benutzten Kameras sprechen so auf den Befehl `spriteSpaceToWorldSpace` an, als ob ihr Anzeigerechteck genau so groß sei wie das Sprite.

Parameter

loc Erforderlich. Gibt die Position im referenzierten Sprite an. Diese Position muss ein Punkt relativ zur oberen linken Ecke des Sprites sein.

Beispiel

Die folgende Anweisung zeigt, dass der Punkt (50, 50) in Sprite 5 dem Vector (-1993.6699, 52.0773, 2263.7446) auf der Projektionsebene der Kamera von Sprite 5 entspricht:

```
-- Lingo
put sprite(5).camera.spriteSpaceToWorldSpace(point(50, 50))
-- vector(-1993.6699, 52.0773, 2263.7446)

// Javascript
put (sprite(5).camera.spriteSpaceToWorldSpace(point(50, 50)) );
//<vector(-1993.6699, 52.0773, 2263.7446)>
```

Siehe auch

worldSpaceToSpriteSpace, rect (camera), camera

sqrt()

Syntax

```
sqrt(number)
the sqrt of number
```

Beschreibung

Diese mathematische Funktion (nur Lingo) gibt die Quadratwurzel einer angegebenen Zahl zurück.

Der Wert muss eine Dezimalzahl über 0 sein. Negativwerte geben 0 zurück.

In JavaScript-Syntax wird die `sqrt()`-Funktion des Math-Objekts verwendet.

Parameter

number Erforderlich. Gibt die Zahl an. Diese Zahl ist entweder ein Fließkommawert oder ein auf die nächste Ganzzahl auf- oder abgerundeter Ganzzahlwert.

Beispiel

Die folgende Anweisung zeigt die Quadratwurzel von 3.0 im Nachrichtenfenster an:

```
put sqrt(3.0)
-- 1.7321
```

Die folgende Anweisung zeigt die Quadratwurzel von 3 im Nachrichtenfenster an:

```
put sqrt(3)
-- 2
```

Siehe auch

floatPrecision

stageBottom

Syntax

```
the stageBottom
```

Beschreibung

In Verbindung mit `stageLeft`, `stageRight` und `stageTop` gibt diese Funktion an, wo sich die Bühne auf dem Desktop befindet. Sie gibt die untere vertikale Koordinate der Bühne relativ zur linken oberen Ecke des Hauptbildschirms an. Die Höhe der Bühne in Pixel wird durch `the stageBottom - the stageTop` berechnet.

Wenn der Film als Applet abgespielt wird, ist die Eigenschaft `stageBottom` gleich der Höhe des Applets in Pixel.

Diese Funktion kann getestet, aber nicht eingestellt werden.

Parameter
Keiner

Beispiel
Die folgenden Anweisungen positionieren Sprite 3 in einem Abstand von 50 Pixeln vom unteren Bühnenrand:

```
stageHeight = the stageBottom - the stageTop
sprite(3).locV = stageHeight - 50
```

Sprite-Koordinaten werden relativ zur linken oberen Ecke der Bühne ausgedrückt. Weitere Informationen finden Sie in den Themen des Director-Benutzerhandbuchs im Director-Hilfefenster.

```
// Javascript
var stageHeight =_movie.stage.rect.Bottom -_movie.stage.rect.Top ;
sprite(3).locV = stageHeight - 50 ;
```

Siehe auch
stageLeft, stageRight, stageTop, locH, locV

stageLeft

Syntax
```
the stageLeft
```

Beschreibung
In Verbindung mit `stageRight`, `stageTop` und `stageBottom` gibt diese Funktion an, wo sich die Bühne auf dem Desktop befindet. Sie gibt die linke horizontale Koordinate der Bühne relativ zu der linken oberen Ecke des Hauptbildschirms zurück. Wenn der Bühnenrand mit dem linken Rand des Hauptbildschirms übereinstimmt, ist diese Koordinate gleich 0.

Wird der Film als Applet abgespielt, so ist die Eigenschaft `stageLeft` 0, was der Position des linken Randes des Applets entspricht.

Diese Eigenschaft kann getestet, aber nicht eingestellt werden.

Sprite-Koordinaten werden relativ zur linken oberen Ecke der Bühne ausgedrückt.

Parameter
Keiner

Beispiel
Die folgende Anweisung prüft, ob der linke Rand der Bühne über den linken Rand des Bildschirms hinausragt, und ruft die Prozedur `leftMonitorProcedure` auf, wenn dies der Fall ist:

```
if the stageLeft < 0 then leftMonitorProcedure
```

Siehe auch
stageBottom, stageRight, stageTop, locH, locV

stageRight

Syntax

```
the stageRight
```

Beschreibung

In Verbindung mit `stageLeft`, `stageTop` und `stageBottom` gibt diese Funktion an, wo sich die Bühne auf dem Desktop befindet. Sie gibt die rechte horizontale Koordinate der Bühne relativ zur linken oberen Ecke des Hauptbildschirm-Desktops zurück. Die Breite der Bühne in Pixel wird durch `the stageRight -the stageLeft` berechnet.

Wird der Film als Applet abgespielt, so ist die Eigenschaft `stageRight` gleich der Breite des Applets in Pixel.

Diese Funktion kann getestet, aber nicht eingestellt werden.

Sprite-Koordinaten werden relativ zur linken oberen Ecke der Bühne ausgedrückt.

Parameter

Keiner

Beispiel

Die folgenden zwei Anweisungen positionieren Sprite 3 in einem Abstand von 50 Pixeln vom rechten Bühnenrand:

```
stageWidth = the stageRight - the stageLeft
    sprite(3).locH = stageWidth - 50
```

Siehe auch

`stageLeft`, `stageBottom`, `stageTop`, `locH`, `locV`

stageToFlash()

Syntax

```
-- Lingo syntax
spriteObjRef.stageToFlash(pointOnDirectorStage)

// JavaScript syntax
spriteObjRef.stageToFlash(pointOnDirectorStage);
```

Beschreibung

Diese Funktion gibt die Koordinate eines Flash-Film-Sprites zurück, die mit einer spezifischen Koordinate auf der Director-Bühne übereinstimmt. Die Funktion akzeptiert die Director-Bühnenkoordinate und gibt die Flash-Filmkoordinate in Director-Punktwerten zurück. Beispiel: (point 300,300).

Flash-Filmkoordinaten werden in Flash-Filmpixeln gemessen, die bei der Erstellung des Films in Flash anhand seiner Originalgröße festgelegt wurden. „Point(0,0)“ eines Flash-Films ist immer die linke obere Ecke. (Die Darstellereigenschaft `originPoint` wird nicht zur Berechnung von Filmkoordinaten verwendet, sondern nur zur Drehung und Skalierung eingesetzt.)

Mit der Funktion `stageToFlash()` und der entsprechenden `flashToStage()`-Funktion kann bestimmt werden, welche Flash-Filmkoordinate sich direkt über einer Director-Bühnenkoordinate befindet. Sowohl bei Flash als auch bei Director ist „point(0,0)“ die linke obere Ecke der Bühne. Diese Koordinaten stimmen möglicherweise nicht mit der Director-Bühne überein, wenn ein Flash-Sprite gestreckt, skaliert oder gedreht wurde.

Parameter

pointOnDirectorStage Erforderlich. Gibt den Punkt auf der Director-Bühne an.

Beispiel

Die folgende Prozedur prüft, ob sich die Maus (deren Position in Director-Bühnenkoordinaten verfolgt wird) über der Koordinate „point(130,10)“ in einem Flash-Film-Sprite in Kanal 5 befindet. Wenn sich die Maus über dieser Flash-Filmkoordinate befindet, wird der Flash-Film angehalten.

```
-- Lingo syntax
on checkFlashRollover
    if sprite(5).stageToFlash(point(_mouse.mouseH,_mouse.mouseV)) = point(130,10) then
        sprite(5).stop()
    end if
end

// JavaScript syntax
function checkFlashRollover() {
    var stf = sprite(5).stageToFlash(point(_mouse.mouseH,_mouse.mouseV));
    if (stf = point(130,10)) {
        sprite(5).stop();
    }
}
```

Siehe auch

`flashToStage()`

stageTop

Syntax

```
the stageTop
```

Beschreibung

In Verbindung mit `stageBottom`, `stageLeft` und `stageRight` gibt diese Funktion an, wo sich die Bühne auf dem Desktop befindet. Sie gibt die obere vertikale Koordinate der Bühne relativ zur linken oberen Ecke des Hauptbildschirm-Desktops zurück. Befindet sich die Bühne in der linken oberen Ecke des Hauptbildschirms, so ist diese Koordinate gleich 0.

Wird der Film als Applet abgespielt, so ist die Eigenschaft `stageTop` immer gleich 0, was der Position des linken Rands des Applets entspricht.

Diese Funktion kann getestet, aber nicht eingestellt werden.

Sprite-Koordinaten werden relativ zur linken oberen Ecke der Bühne ausgedrückt.

Parameter

Keiner

Beispiel

Die folgende Anweisung prüft, ob der obere Bühnenrand über den Rand des Bildschirms hinausragt, und ruft die Prozedur `upperMonitorProcedure` auf, wenn dies der Fall ist:

```
if the stageTop < 0 then upperMonitorProcedure
```

Siehe auch

`stageLeft`, `stageRight`, `stageBottom`, `locH`, `locV`

startSave (Mixer)

Syntax

```
mixer.startSave(bitdepth,samplingFreq,NumChannels,filePath)
```

Beschreibung

Diese Mixermethode speichert die Mixerausgabe in der angegebenen Datei. Das Soundobjekt kann als WAV- oder MP4-Datei gespeichert werden. Verwenden Sie `startSave` nur während der Wiedergabe eines Sounds. Folgende Anweisung speichert einen Sound während der Wiedergabe und speichert ihn in der angegebenen Datei.

Wenn die Parameter `BitDepth`, `samplingFreq`, und `NumChannels` nicht angegeben werden, verwendet Director zum Speichern der Datei die Parameter des ursprünglichen Soundobjekts.

Für die WAV-Ausgabe werden Bit-Tiefen von 8, 16 und 24 unterstützt. Für die MP4-Ausgabe wird jedoch nur die 16-Bit-Tiefe unterstützt.

Hinweis: *Mit der Methode* `stopsave()` *wird der mit* `startSave` *begonnene Speichervorgang beendet.*

Beispiele

```
-- Lingo
on mouseUp me
mixerRef.startSave(16,44100,2,"C:\audio.wav")
end
// Javascript
function mouseUp(){
mixerRef.startSave(16,44100,2,"C:\audio.wav");
}
```

Siehe auch

`stopSave (Mixer)`, `Mixer`

startSave (Soundobjekt)

Syntax

```
soundObject.startSave(bitdepth,samplingFreq,NumChannels,filename)
```

Beschreibung

Diese Funktion speichert das aktuell wiedergegebene Soundobjekt in der angegebenen Datei. Das Soundobjekt kann als WAV- oder MP4-Datei gespeichert werden. Verwenden Sie `startSave` nur während der Wiedergabe eines Sounds. Die Funktion kopiert den Sound während der Wiedergabe und speichert sie in der angegebenen Datei.

Wenn die Parameter `BitDepth`, `samplingFreq`, und `NumChannels` nicht angegeben werden, verwendet Director zum Speichern der Datei die Parameter des ursprünglichen Soundobjekts.

Für die WAV-Ausgabe werden Bit-Tiefen von 8, 16 und 24 unterstützt. Für die MP4-Ausgabe wird jedoch nur die 16-Bit-Tiefe unterstützt.

Hinweis: *Mit der Methode* `stopsave()` *wird der mit* `startSave` *begonnene Speichervorgang beendet.*

Beispiele

In folgenden Beispielen wird das aktuelle Soundobjekt unter `C:\test.wav` mit einer Samplingfrequenz von 48 kHz und einer Bittiefe von 16 gespeichert.

```
--Lingo syntax
on mouseUp me
     soundObjRef.startSave(16,48000,2,"C:\audio.wav")
end.

// JavaScript syntax
function mouseUp(){
soundObjRef.startSave(16,48000,2,"C:\audio.wav");
}
```

Siehe auch

`stopSave (Soundobjekt)`

status()

Syntax

```
-- Lingo syntax
fileioObjRef.status()

// JavaScript syntax
fileioObjRef.status();
```

Beschreibung

Diese Datei-E/A-Methode (Fileio) gibt den Fehlercode der zuletzt aufgerufenen Methode zurück.

Parameter

Keiner

Beispiel

Die folgende Anweisung zeigt den aktuellen Status von Soundkanal 2 im Nachrichtenfenster an:

```
-- Lingo syntax
put(sound(2).status)

// JavaScript syntax
put(sound(2).status);
```

Siehe auch

Fileio

stop() (DVD)

Syntax

```
-- Lingo syntax
dvdObjRef.stop()

// JavaScript syntax
dvdObjRef.stop();
```

Beschreibung

Diese DVD-Methode beendet die Wiedergabe.

Bei Erfolg gibt diese Methode TRUE (1) zurück.

Parameter

Keiner

Beispiel

Die folgende Anweisung beendet die Wiedergabe:

```
-- Lingo syntax
member(1).stop()

// JavaScript syntax
member(1).stop();
```

Siehe auch

DVD

stop() (Flash)

Syntax

```
-- Lingo syntax
spriteObjRef.stop()

// JavaScript syntax
spriteObjRef.stop();
```

Beschreibung

Dieser Flash-Befehl stoppt ein Flash-Film-Sprite, das im aktuellen Bild abgespielt wird.

Parameter

Keiner

Beispiel

Das folgende Bildskript stoppt die Flash-Film-Sprites, die in den Kanälen 5 bis 10 abgespielt werden:

```
-- Lingo syntax
on enterFrame
    repeat with i = 5 to 10
        sprite(i).stop()
    end repeat
end

// JavaScript syntax
function enterFrame() {
    var i = 5;
    while (i < 11) {
        sprite(i).stop();
        i++;
    }
}
```

Siehe auch

`hold()`

stop() (Mixer)

Syntax

```
Mixer.stop([soundobjectlist])
```

Beschreibung

Diese Mixermethode stoppt die Wiedergabe aller Soundobjekte im Mixer, falls kein Soundobjekt übergeben wurde, oder stoppt die in der Liste vorhandenen Soundobjekt.

Wirkung auf den Mixer	Funktion
stop()	Stoppt den Mixer.
stop([so1, so2])	Stoppt die angegebenen Soundobjekte im Mixer mit den Referenzen `so1` und `so2`.
stop(["so1", "so2"])	Stoppt die angegebenen Soundobjekte im Mixer mit den Namen `so1` und `so2`.

Beispiele

```
-- Lingo syntax
on mouseUp me
     mixer1.stop() --Stops mixer1.
end

// JavaScript syntax
function mouseUp()
{
mixer1.stop(); //Stops mixer1.

}
```

Siehe auch

Mixer

stop() (MP4Media/FLV)

Syntax

```
sprite(1).stop()
member("MP4Media/FLV").stop()
```

Beschreibung

Diese MP4Media/FLV-Sprite-Methode stoppt die Wiedergabe eines MP4Media-Sprites. Der Wert von `mediaStatus` wird auf `#stopped` gesetzt.

Beispiele

In folgendem Beispiel wird die Wiedergabe von Sprite 2 angehalten:

```
-- Lingo syntax
sprite(2).stop()
member("MP4Media/FLV").stop()

// JavaScript syntax
sprite(2).stop();
member("MP4Media/FLV").stop();
```

stop() (RealMedia, SWA, Windows Media)

Syntax

```
-- Lingo syntax
windowsMediaObjRef.stop()
realMediaObjRef.stop()

// JavaScript syntax
windowsMediaObjRef.stop();
realMediaObjRef.stop();
```

Beschreibung
Diese Methode für Windows Media- oder RealMedia-Darsteller und -Sprites beendet die Wiedergabe eines Windows Media- oder RealMedia-Darstellers oder -Sprites.

Parameter
Keiner

Beispiel
In den folgenden Beispielen wird die Wiedergabe von Sprite 2 bzw. des Darstellers "Real" gestoppt:

```
-- Lingo syntax
sprite(2).stop()
member("Real").stop()

// JavaScript syntax
sprite(2).stop();
member("Real").stop();
```

Siehe auch
RealMedia, Windows Media

stop() (Soundkanal)

Syntax

```
-- Lingo syntax
soundChannelObjRef.stop()

// JavaScript syntax
soundChannelObjRef.stop();
```

Beschreibung
Diese Soundkanalmethode beendet den Sound, der gerade in einem Soundkanal abgespielt wird.

Mit dem Befehl `play()` kann der erste der Sounds abgespielt werden, die sich noch in der Warteschlange für den jeweiligen Soundkanal befinden.

Ein Beispiel für `stop()` in einem fertigen Film finden Sie im Film „Sound Control“ im Ordner „Learning/Lingo“ im Director-Anwendungsordner.

Parameter
Keiner

Beispiel
Die folgende Anweisung stoppt die Wiedergabe des Sounddarstellers, der gegenwärtig in Soundkanal 1 abgespielt wird:

```
-- Lingo syntax
sound(1).stop()

// JavaScript syntax
sound(1).stop();
```

Siehe auch

`getPlayList()`, `pause() (Soundkanal)`, `play() (Soundkanal)`, `playNext() (Soundkanal)`, `rewind() (Soundkanal)`, `Sound Channel`

stop (Soundobjekt)

Syntax

```
soundObject.stop()
```

Beschreibung

Diese Soundobjektmethode stoppt die Wiedergabe der mit dem Soundobjekt verknüpften Audiodatei.

Beispiele

```
-- Lingo syntax
on mouseUp me
     soundObjRef.stop() -- Stops the playback of the sound object associated with soundObjRef.
end

// JavaScript syntax
function mouseUp(){
soundObjRef.stop();  //Stops the playback of the sound object associated with soundObjRef.
}
```

stop

Syntax

```
stop(MUIObject, stopItem)
```

Beschreibung

Diese Funktion beendet ein allgemeines aus einer Instanz des Xtras MUI erstelltes Dialogfeld. Nach Aufruf der Funktion gibt Lingo die Ergebnisse als Anzahl der `stopItem`-Parameter zurück.

Der Parameter „stopItem“ wird vom Abruf von `run(MUIOject)` zurückgegeben. Übergeben Sie damit einen Parameter zurück, der angibt, wie das Dialogfeld beendet wurde. Sie können beispielsweise „1“ zurückgeben, wenn der Benutzer auf `OK` geklickt hat, und „0“ zurückgeben, wenn der Benutzer auf `Cancel` geklickt hat.

Hinweis: *Um ein nicht modales Dialogfeld zu schließen, verwenden Sie den Befehl* `WindowOperation` *mit der Option* `#hide`.

Beispiel

Diese Prozedur beendet das aus „MUIObject“ erstellte allgemeine Dialogfeld. Der zweite Parameter des Befehls „stop“ ist Null, wodurch die Anforderung nach einem Wert erfüllt ist, der jedoch keinen anderen Zweck hat:

```
--Lingo syntax
on stopDialog
    global MUIObject
    if ( objectP (MUIObject)) then
        stop(MUIObject, 0)
    end if
end stopDialog
```

stopEvent()

Syntax

```
-- Lingo syntax
_movie.stopEvent()

// JavaScript syntax
_movie.stopEvent();
```

Beschreibung

Diese Filmmethode hindert Skriptcode daran, eine Ereignisnachricht an nachfolgende Positionen in der Nachrichtenhierarchie zu übergeben.

Diese Methode kann auch auf Sprite-Skripts angewendet werden.

Verwenden Sie die `stopEvent()`-Methode, um die Nachricht in einer primären Ereignisprozedur oder einem Sprite-Skript zu stoppen, sodass die Nachricht auch für nachfolgende Sprite-Skripts nicht mehr verfügbar ist.

Standardmäßig stehen Nachrichten zuerst einer primären Ereignisprozedur zur Verfügung (sofern eine existiert) und erst dann allen anderen Skripts, die an einem an diesem Ereignis beteiligten Sprite angebracht sind. Wenn mehrere Skripts am Sprite angebracht sind, ist die Nachricht für jedes Skript des Sprites verfügbar. Wenn kein Sprite-Skript auf die Nachricht antwortet, wird die Nachricht an ein Darstellerskript, ein Bildskript und schließlich an ein Filmskript weitergegeben.

Die `stopEvent()`-Methode betrifft nur das gegenwärtig in Verarbeitung befindliche Ereignis. Er hat keinen Einfluss auf zukünftige Ereignisse. Die `stopEvent()`-Methode ist nur in primären Ereignisprozeduren, Prozeduren, die von primären Ereignisprozeduren aufgerufen werden, oder in mehreren Sprite-Skripts gültig. Er hat an anderen Stellen keine Wirkung.

Parameter

Keiner

Beispiel

Die folgende Anweisung zeigt, wie das Ereignis `mouseUp` in einem Verhalten gestoppt wird, wenn die globale Variable `grandTotal` gleich 500 ist:

```
-- Lingo syntax
global grandTotal
on mouseUp me
    if (grandTotal = 500) then
        _movie.stopEvent()
    end if
end

// JavaScript syntax
_global.grandTotal;
function mouseUp() {
    if (_global.grandTotal == 500) {
        _movie.stopEvent();
    }
}
```

Weder nachfolgende Skripts noch andere Verhalten in diesem Sprite erhalten dieses Ereignis, wenn es auf diese Weise gestoppt wird.

Siehe auch

Movie

stopSave (Mixer)

Syntax

```
mixer.stopSave()
```

Beschreibung

Diese Mixermethode beendet den Speichervorgang, der durch `startSave` begonnen wurde.

Beispiele

```
--Lingo syntax
on mouseUp me
     mixerRef.stopSave() --Ends the save operation.
end

//JavaScript syntax
function mouseUp(){
     mixerRef.stopSave(); //Ends the save operation.
}
```

Siehe auch

startSave (Mixer), Mixer

stopSave (Soundobjekt)

Syntax

```
soundObject.stopSave()
```

Beschreibung

Diese Soundobjektmethode beendet den Speichervorgang, der durch `startSave` begonnen wurde.

Beispiele

```
--Lingo syntax
on mouseUp me
     soundObjRef.stopSave() --Ends the save operation.
end

//JavaScript syntax
function mouseUp(){
     soundObjRef.stopSave(); //Ends the save operation.
}
```

Siehe auch

`startSave (Soundobjekt)`

stream()

Syntax

```
-- Lingo syntax
memberObjRef.stream(numberOfBytes)

// JavaScript syntax
memberObjRef.stream(numberOfBytes);
```

Beschreibung

Dieser Befehl streamt einen Teil des angegebenen Flash-Filmdarstellers manuell in den Speicher.

Der Befehl `stream` gibt die Anzahl von Bytes zurück, die bereits gestreamt wurden. Aufgrund verschiedener Umstände (z. B. der Netzwerkgeschwindigkeit oder der Verfügbarkeit der angeforderten Daten) ist die Anzahl von Bytes, die bis zu diesem Zeitpunkt gestreamt wurden, möglicherweise geringer als die angeforderte Anzahl.

Sie können den Befehl `stream` immer für einen Darsteller verwenden, unabhängig von der Eigenschaft `streamMode` des Darstellers.

Parameter

numberOfBytes Optional. Eine Ganzzahl, die die Anzahl der zu streamenden Bytes angibt. Wenn Sie den Parameter *numberOfBytes* auslassen, versucht Director, so viele Bytes zu streamen, wie in der Darstellereigenschaft `bufferSize` angegeben sind.

Beispiel

Das folgende Bildskript prüft, ob ein verknüpfter Flash-Filmdarsteller in den Speicher gestreamt wurde. Hierzu wird die Eigenschaft `percentStreamed` überprüft. Ist der Darsteller noch nicht ganz in den Speicher geladen, versucht das Skript, 32.000 Bytes des Films in den Speicher zu streamen.

Das Skript speichert auch die tatsächliche Anzahl von gestreamten Bytes in der Variablen `bytesReceived`. Wenn die tatsächlich gestreamte Anzahl von Bytes nicht mit der angeforderten Anzahl von Bytes übereinstimmt, aktualisiert das Skript einen Textdarsteller, der dann die tatsächlich empfangene Anzahl von Bytes anzeigt. Das Skript sorgt dafür, dass der Abspielkopf im aktuellen Bild eine Schleife durchläuft, bis der Darsteller in den Speicher geladen ist.

```
-- Lingo syntax
on exitFrame
    if member(10).percentStreamed < 100 then
        bytesReceived = member(10).stream(32000)
        if bytesReceived < 32000 then
            member("Message Line").text = "Received only" && bytesReceived && "of 32,000 bytes
requested."
            _movie.updateStage()
        else
            member("Message Line").text = "Received all 32,000 bytes."
        end if
        _movie.go(_movie.frame)
    end if
end

// JavaScript syntax
function exitFrame() {
    var pctStm = member(10).percentStreamed;
    if (pctStm < 100) {
        var bytesReceived = member(10).stream(32000);
        if (bytesReceived < 32000) {
            member("Message Line").text = "Received only " + bytesReceived + " of32,000 bytes
requested.";
            _movie.updateStage();
        } else {
            member("Message Line").text = "Received all 32,000 bytes.";
        }
        _movie.go(_movie.frame);
    }
}
```

string()

Syntax

```
string(expression)
```

Beschreibung

Diese Funktion konvertiert Ganzzahlen, Fließkommazahlen, Objektbezüge, Listen, Symbole oder andere nicht als Strings klassifizierte Ausdrücke in Strings.

Parameter

expression Erforderlich. Der in eine Zeichenfolge zu konvertierende Ausdruck.

Beispiel

Die folgende Anweisung addiert 2.0 + 2.5 und fügt das Ergebnis in den Felddarsteller "Summe" ein:

```
--Lingo
member("total").text = string(2.0 + 2.5)

// Javascript
member("total").text = (2.0 + 2.5).toString() ;
```

Die folgende Anweisung konvertiert das Symbol `#red` in einen String und fügt diesen in den Felddarsteller `Color` ein:

```
--Lingo
member("Color").text = string(#red)

// Javascript
member("Color").text = symbol("red").toString();
```

Siehe auch

`value()`, `stringP()`, `float()`, `integer()`, `symbol()`

stringP()

Syntax

```
stringP(expression)
```

Beschreibung

Diese Funktion bestimmt, ob ein Ausdruck ein String ist (`TRUE`) oder nicht (`FALSE`).

Das *P* in `stringP` steht für *predicate* (Prädikat).

Parameter

expression Erforderlich. Der zu testende Ausdruck.

Beispiel

Die folgende Anweisung prüft, ob *3* ein String ist:

```
put stringP("3")
```

Das Ergebnis ist 1, das numerische Gegenstück zu TRUE.

Die folgende Anweisung prüft, ob die Fließkommazahl 3.0 ein String ist:

```
put stringP(3.0)
```

Da 3.0 eine Fließkommazahl und keine Zeichenfolge ist, ist das Ergebnis 0, das numerische Gegenstück zu FALSE.

Siehe auch

`floatP()`, `ilk()`, `integerP()`, `objectP()`, `symbolP()`

subPictureType()

Syntax

```
-- Lingo syntax
dvdObjRef.subPictureType(intStream)

// JavaScript syntax
dvdObjRef.subPictureType(intStream);
```

Beschreibung
Diese DVD-Methode gibt den Typ eines angegebenen Untergrafikstreams an.

Sie kann folgende Werte zurückgeben:

Symbol	Beschreibung
`#unknown`	Der Untergrafiktyp ist unbekannt.
#Language	Die Untergrafik enthält sprachbezogenen Inhalt, wie z. B. Filmuntertitel oder anderen Text.
#Other	Die Untergrafik enthält nicht sprachbezogenen Inhalt, wie z. B. einen springenden Ball in Karaoketiteln.

Parameter
intStream Erforderlich. Eine Ganzzahl, die den zu testenden Stream angibt.

Beispiel
Die folgende Anweisung gibt den Untergrafiktyp in Stream 2 zurück:

```
-- Lingo syntax
member(12).member.subPictureType(2)

// JavaScript syntax
member(12).member.subPictureType(2);
```

Siehe auch
DVD

substituteFont

Syntax
```
TextMemberRef.substituteFont(originalFont, newFont)
substituteFont(textMemberRef, originalFont, newFont)
```

Beschreibung
Dieser Textdarstellerbefehl ersetzt alle Instanzen einer Schriftart in einem Textdarsteller durch eine andere Schriftart.

Parameter
originalFont Erforderlich. Die zu ersetzende Schrift.

newFont Erforderlich. Die neue Schrift, durch die die in *originalFont* angegebene Schrift ersetzt wird.

Beispiel
Das folgende Skript prüft, ob die Schriftart "Bonneville" in einem Textdarsteller verfügbar ist, und ersetzt sie durch "Arial", wenn dies nicht der Fall ist:

```
-- Lingo syntax
property spriteNum

on beginSprite me
    currMember = sprite(spriteNum).member
    if currMember.missingFonts contains "Bonneville" then
        currMember.substituteFont("Bonneville", "Arial")
    end if
end

// JavaScript syntax
function beginSprite() {
    currMember = sprite(spriteNum).member;
    if (currMember.missingFonts contains "Bonneville") { //check syntax
        currMember.substituteFont("Bonneville", "Arial");
    }
}
```

Siehe auch

missingFonts

swing()

Syntax

```
-- Lingo syntax
spriteObjRef.swing(pan, tilt, fieldOfView, speedToSwing)

// JavaScript syntax
spriteObjRef.swing(pan, tilt, fieldOfView, speedToSwing);
```

Beschreibung

Diese QuickTime VR-Sprite-Funktion schwenkt ein QuickTime 3-Sprite mit einem VR-Panorama auf die neuen Ansichtseinstellungen. Das Schwenken bewirkt einen weichen „Kamerawagen“-Effekt.

whichQTVRSprite ist die Sprite-Nummer des Sprites mit dem QuickTime VR-Darsteller.

Die Funktion gibt umgehend einen Wert zurück, aber das Sprite verändert fortlaufend die Ansicht, bis es die endgültige Ansichtsposition erreicht hat. Der erforderliche Zeitraum bis zur endgültigen Einstellung ist vom Computertyp, von der Größe des Sprite-Rechtecks, der Farbtiefe des Bildschirms und anderen Parametern abhängig, die sich auf die Leistung auswirken.

Um zu prüfen, ob ein Schwenkvorgang beendet ist, testen Sie, ob die Sprite-Eigenschaft `pan` ihren Endwert erreicht hat.

Parameter

pan Erforderlich. Gibt die neue Schwenkposition in Grad an.

tilt Erforderlich. Gibt die neue Neigung in Grad an.

fieldOfView Erforderlich. Gibt den neuen Blickwinkel in Grad an.

speedToSwing Erforderlich. Gibt die Geschwindigkeit an, mit der der Schwenk durchgeführt werden soll. Gültige Werte sind 1 (langsam) bis 10 (schnell).

Beispiel

Die folgende Anweisung stellt QTVR-Sprite 1 auf eine Schwenkposition von 300°, eine Neigung von -15° und einen Blickwinkel von 40° ein:

```
-- Lingo syntax
sprite(1).swing(300, -15, 40, 1)

// JavaScript syntax
sprite(1).swing(300, -15, 40, 1);
```

Siehe auch

pan (QTVR-Eigenschaft)

symbol()

Syntax

```
-- Lingo syntax
symbol(stringValue)

// JavaScript syntax
symbol(stringValue);
```

Beschreibung

Diese Top-Level-Funktion gibt eine eingegebene Zeichenfolge als Symbol zurück.

Parameter

stringValue Erforderlich. Die in ein Symbol zu konvertierende Zeichenfolge.

Beispiel

Die folgende Anweisung zeigt das Symbol `#hello` an:

```
--Lingo syntax
put(symbol("hello"))

// JavaScript syntax
put(symbol("hello"));
```

Die folgende Anweisung zeigt das Symbol `#goodbye` an:

```
--Lingo syntax
x = "goodbye"
put(symbol(x))

// JavaScript syntax
var x = "goodbye";
put(symbol(x));
```

Siehe auch

value(), string()

symbolP()

Syntax

```
Expression.symbolP
symbolP(expression)
```

Beschreibung

Diese Funktion bestimmt, ob ein angegebener Ausdruck ein Symbol ist (`TRUE`) oder nicht (`FALSE`).

Das *P* in `symbolP` steht für "predicate" (Prädikat).

Parameter

expression Erforderlich. Gibt den zu testenden Ausdruck an.

Beispiel

Die folgende Anweisung prüft, ob die Variable `myVariable` ein Symbol ist:

```
put myVariable.symbolP
```

Siehe auch

`ilk()`

tan()

Syntax

```
tan(angle)
```

Beschreibung

Diese mathematische Funktion ergibt den Tangenswert des angegebenen Winkels, der im Bogenmaß als Fließkommazahl ausgedrückt ist.

In JavaScript-Syntax wird die `tan()`-Funktion des Math-Objekts verwendet.

Parameter

angle Erforderlich. Gibt den Winkel an, von dem der Tangens zurückgegeben wird.

Beispiel

Die folgende Funktion ergibt den Tangenswert von pi/4:

```
tan (PI/4.0) = 1
```

Das Symbol p kann nicht in einem Lingo-Ausdruck verwendet werden.

Siehe auch

`PI`

tellStreamStatus()

Syntax

```
tellStreamStatus(onOrOffBoolean)
```

Beschreibung

Diese Funktion schaltet die Streamstatus-Prozedur ein (`TRUE`) oder aus (`FALSE`).

`tellStreamStatus()` ermittelt den Status der Prozedur.

Wenn `streamStatusHandlerTRUE`ist, wird beim Streamen aus dem Internet periodisch das Filmskript aufgerufen und dadurch die Prozedur `streamStatusHandler` ausgelöst. Die Prozedur wird ausgeführt, und Director setzt automatisch Informationen über den Fortschritt des Downloadvorgangs in die Parameter ein.

Parameter

onOrOffBoolean Optional. Gibt den Status der Prozedur an.

Beispiel

Die folgende `on prepareMovie`-Prozedur schaltet die `on streamStatus`-Prozedur ein, wenn der Film beginnt:

```
-- Lingo syntax
on prepareMovie
    tellStreamStatus(TRUE)
end

// JavaScript syntax
function prepareMovie() {
    tellStreamStatus(TRUE);
}
```

Die folgende Anweisung bestimmt den Status der Streamstatus-Prozedur:

```
-- Lingo syntax
on mouseDown
    put tellStreamStatus()
end

// JavaScript syntax
function mouseDown() {
    put(tellStreamStatus());
}
```

Siehe auch

`on streamStatus`

tellTarget()

Syntax

```
-- Lingo syntax
spriteObjRef.tellTarget(targetName)

// JavaScript syntax
spriteObjRef.tellTarget(targetName);
```

Beschreibung

Dieser Befehl entspricht den Flash-Methoden `beginTellTarget` und `endTellTarget`. Der Befehl `tellTarget()` ermöglicht es dem Benutzer, eine Zielzeitlinie festzulegen, auf der nachfolgende Sprite-Befehle agieren. Wenn als Ziel ein Flash-Movieclip oder eine Stufe angegeben wird, die einen geladenen Flash-Film enthält, beziehen sich bestimmte Befehle nicht auf die Hauptzeitleiste, sondern vielmehr auf die Zielkomponenten. In diesem Fall können Sie durch Aufruf von `endTellTarget()` zur Hauptzeitleiste zurückkehren.

Das einzige zulässige Argument für `tellTarget` ist der Name des Ziels. Der Befehl `endTellTarget` akzeptiert keine Argumente.

`tellTarget` wirkt sich auf die folgenden Flash-Sprite-Funktionen aus: `stop`, `play`, `getProperty`, `setProperty`, `gotoFrame`, `call(frame)` und `find(label)`. Darüber hinaus wird die Sprite-Eigenschaft `frame` (die die Nummer des aktuellen Bildes zurückgibt) von `tellTarget` beeinflusst.

Parameter

targetName Erforderlich. Gibt den Zielnamen an.

Beispiel

Der folgende Befehl stellt den Movieclip als Ziel ein:

```
-- Lingo syntax
sprite(1).tellTarget("myMovieClip")

// JavaScript syntax
sprite(1).tellTarget("myMovieClip");
```

Der folgende Befehl hält den Movieclip an:

```
-- Lingo syntax
sprite(1).stop()

// JavaScript syntax
sprite(1).stop();
```

Der folgende Befehl startet die Wiedergabe des Movieclips:

```
-- Lingo syntax
sprite(1).play()

// JavaScript syntax
sprite(1).play();
```

Der folgende Befehl setzt den Fokus auf die Hauptzeitleiste zurück:

```
-- Lingo syntax
sprite(1).endTellTarget()

// JavaScript syntax
sprite(1).endTellTarget();
```

Der folgende Befehl hält den Hauptfilm an:

```
-- Lingo syntax
sprite(1).stop()

// JavaScript syntax
sprite(1).stop();
```

time() (System)

Syntax

```
-- Lingo syntax
_system.time()

// JavaScript syntax
_system.time();
```

Beschreibung

Diese Systemmethode gibt die aktuelle Uhrzeit der Systemuhr als Zeichenfolge zurück. Die drei Zeitformate sind je nach Zeiteinstellungen des Computers unterschiedlich.

Parameter

Keiner

Beispiel

Die folgende Prozedur gibt die aktuelle Uhrzeit in einem Textfeld aus.

```
-- Lingo syntax
on exitFrame
    member("clock").text = _system.time()
end

// JavaScript syntax
function exitFrame() {
    member("clock").text = _system.time();
}
```

Siehe auch

date() (System), System

timeout()

Syntax

```
-- Lingo syntax
timeout(timeoutObjName)

// JavaScript syntax
timeout(timeoutObjName);
```

Beschreibung

Diese Top-Level-Funktion gibt ein vorgegebenes Timeout-Objekt zurück.

Mithilfe der `new()`-Methode können Sie ein neues Timeout-Objekt erstellen und der `timeoutList` hinzufügen.

Parameter

timeoutObjName Erforderlich. Eine Zeichenfolge, die den Namen des zurückzugebenden Timeout-Objekts angibt.

Beispiel

Die folgende Prozedur löscht das Timeout-Objekt "Blitz":

```
-- Lingo syntax
on exitFrame
    timeout("Random Lightning").forget()
end

// JavaScript syntax
function exitFrame() {
    timeout("Random Lightning").forget();
}
```

Siehe auch

`new()`, `timeoutList`, `timeoutHandler`, `time (Timeout-Objekt)`, `name (timeout)`, `period`, `persistent`, `target`

titleMenu()

Syntax

```
-- Lingo syntax
dvdObjRef.titleMenu()

// JavaScript syntax
dvdObjRef.titleMenu();
```

Beschreibung

Diese DVD-Methode zeigt das Titelmenü an.

Parameter

Keiner

Beispiel

Die folgende Anweisung zeigt das Titelmenü an:

```
-- Lingo syntax
member(1).titleMenu()

// JavaScript syntax
member(1).titleMenu();
```

Siehe auch

DVD

toHexString

Syntax

```
toHexString(intOffset, intLength)
```

Beschreibung

Diese Bytearray-Methode gibt den Inhalt des Bytearrays als String zurück. Falls der Wert (`offset+length`) die tatsächliche Länge des Bytearrays überschreitet, wird das Bytearray auf (`actualLength-offset`) gekürzt.

Verwenden Sie diese Methode nur in Nachrichtenfenstern, nicht im Debugger. Der String wird im Debugger-Watch-Fenster abgeschnitten.

Parameter

Parameter	Beschreibung	Erforderlich/Optional
intOffset	Anfangsposition.	Erforderlich
intLength	Länge des Byte-Arrays.	Erforderlich

Beispiele

```
--Lingo syntax
put bArray.toHexString(1,50)

//JavaScript syntax
put(bArray.toHexString(1,50));
```

top (3D)

Syntax

```
modelResourceObjectReference.top
```

Beschreibung

Mit diesem 3D-Befehl können Sie bei einer Modellressource vom Typ `#box` die Eigenschaft `top` der Modellressource ermitteln und festlegen.

Die Eigenschaft `top` bestimmt, ob die Oberseite der Box geschlossen (`TRUE`) oder offen ist (`FALSE`). Der Standardwert ist `TRUE`.

Parameter
Keiner

Beispiel
Die folgende Anweisung prüft, ob der obere Rand von Sprite 3 über den oberen Rand der Bühne hinausragt, und ruft die Prozedur `offTopEdge` auf, wenn dies der Fall ist:

```
-- Lingo syntax
if (sprite(3).top < 0) then
    offTopEdge()
end if

// JavaScript syntax
if (sprite(3).top < 0) {
    offTopEdge();
}
```

Siehe auch
back, bottom (3D), front

topCap

Syntax
```
modelResourceObjectReference.topCap
```

Beschreibung
Mit diesem 3D-Befehl können Sie bei einer Modellressource vom Typ `#cylinder` die Eigenschaft `topCap` der Modellressource ermitteln und festlegen.

Die Eigenschaft `topCap` bestimmt, ob die Oberseite des Zylinders geschlossen (`TRUE`) oder offen ist (`FALSE`). Der Standardwert dieser Eigenschaft lautet `FALSE`.

Parameter
Keiner

Beispiel
Die folgende Anweisung setzt die Eigenschaft `topCap` der Modellressource `Tube` auf `FALSE`, d. h. die Oberseite des Zylinders ist offen:

```
-- Lingo syntax
member("3D World").modelResource("Tube").topCap = FALSE

// JavaScript syntax
member("3D World").getPropRef("modelResource", 10).topCap = false;
```

topRadius

Syntax

```
modelResourceObjectReference.topRadius
```

Beschreibung

Mit diesem 3D-Befehl können Sie bei einer Modellressource vom Typ `#cylinder` die Eigenschaft `topRadius` der Modellressource ermitteln und festlegen (Fließkommawert).

Die Eigenschaft `topRadius` bestimmt den Radius der Oberseite des Zylinders. Der Wert muss immer größer als 0.0 sein. Der Standardwert ist 25,0. Das Einstellen von `topRadius` to 0.0 produces a cone.

Parameter

Keiner

Beispiel

Die folgende Anweisung stellt die Eigenschaft `topRadius` der Modellressource `Tube` auf 0.0. Falls der untere Radius einen Wert größer 0 hat, sind Modelle, die `Tube` verwenden, konisch.

```
-- Lingo syntax
member("3D World").modelResource("Tube").topRadius = 0.0

// JavaScript syntax
member("3D World").getPropRef("modelResource", 10).topRadius = 0.0;
```

trace()

Syntax

```
-- Lingo syntax
trace(value)

// JavaScript syntax
trace(value);
```

Beschreibung

Diese Top-Level-Funktion wertet einen Ausdruck aus und zeigt das Ergebnis im Nachrichtenfenster an.

Die Funktionalität dieser Methode ist mit der der Top-Level-Methode `put()`, die in Lingo als auch in JavaScript-Syntax verfügbar ist, identisch.

Diese Methode kann auch als Debugging-Werkzeug verwendet werden, um bei der Wiedergabe des Films die Werte von Variablen zu verfolgen.

Parameter

value Erforderlich. Der auszuwertende Ausdruck.

Beispiel

Die folgende Anweisung gibt den Wert der Variablen counter im Nachrichtenfenster aus:

```
-- Lingo syntax
counter = (_system.milliseconds / 1000)
trace(counter)

// JavaScript syntax
var counter = (_system.milliseconds / 1000);
trace(counter);
```

Siehe auch

put()

transform (Befehl)

Syntax

```
transform()
```

Beschreibung

Dieser 3D-Befehl erstellt ein Transformationsobjekt, das gleich der Identitätstransformation ist. Die Positions- und Drehungskomponente der Einheitsmatrix lautet jeweils „vector(0,0,0)“, die Skalierungskomponente „vector(1,1,1)“.

Wenn Sie Transformationsinformationen speichern und dann neu erzeugen müssen, speichern Sie zuerst die Transformationseigenschaften (Position, Drehung und Skalierung) und erzeugen dann die Transformation neu, indem Sie eine Identitätstransformation durchführen und anschließend Position, Drehung und Skalierung mithilfe der gespeicherten Daten einstellen.

Parameter

Keiner

Beispiel

Die folgende Anweisung erstellt eine Einheitsmatrix und speichert sie in der Variablen „tTransform“:

```
-- Lingo syntax
tTransform = transform()

// JavaScript syntax
tTransform = transform();
```

Siehe auch

transform (Eigenschaft), preRotate, preTranslate(), preScale(), rotate, translate, scale (Befehl)

translate

Syntax

```
member(whichCastmember).node(whichNode).translate(xIncrement, yIncrement, zIncrement {,
relativeTo})
member(whichCastmember).node(whichNode).translate(translateVector {, relativeTo})
transform.translate(xIncrement, yIncrement, zIncrement {, relativeTo})
transform.translate(translateVector {, relativeTo})
```

Beschreibung

Dieser 3D-Befehl führt nach den aktuellen Positions-, Drehungs- und Skalierungsoffsets des Transformationsobjekts des referenzierten Nodes bzw. des direkt referenzierten Transformationsobjekts eine Translation durch. Die Translation muss in Form von drei Inkrementwerten entlang den drei Achsen angegeben werden. Diese Inkrementwerte können explizit im Format *xIncrement*, *yIncrement* und *zIncrement* oder anhand eines *translateVector* angegeben werden, bei dem die x-Komponente des Vektors der Translation um die x-Achse, die y-Komponente der Translation um die y-Achse und die z-Komponente der Translation um die z-Achse entspricht.

Ein Node kann ein Kamera-, Modell, Licht- oder Gruppenobjekt sein.

Parameter

xIncrement Erforderlich beim Angeben eines Satzes von drei Inkrementen. Gibt das Inkrement für die *X*-Achse an.

yIncrement Erforderlich beim Angeben eines Satzes von drei Inkrementen. Gibt das Inkrement für die *Y*-Achse an.

zIncrement Erforderlich beim Angeben eines Satzes von drei Inkrementen. Gibt das Inkrement für die *Z*-Achse an.

translateVector Erforderlich beim Angeben eines Vektors. Gibt den Vektor an, der die *x*-, *y*- und *z*-Komponenten enthält.

relativeTo Optional. Bestimmt, welche Koordinatensystemachsen zur Durchführung der gewünschten Translationsänderungen verwendet werden. Der Parameter *relativeTo* kann folgende Werte aufweisen:

- `#self` wendet die Inkrementwerte bezogen auf das lokale Koordinatensystem des Nodes (die für das Modell beim Authoring angegebenen x-, y- und z-Achsen) an. Dieser Wert wird als Standardwert verwendet, wenn Sie den Befehl `translate` zusammen mit einem Node-Bezug benutzen und den Parameter *relativeTo* nicht angeben.
- `#parent` wendet die Inkrementwerte bezogen auf das Koordinatensystem des Parent-Objekts des Nodes an. Dieser Wert wird als Standardwert verwendet, wenn Sie den Befehl `translate` zusammen mit einem Transformationsbezug benutzen und den Parameter *relativeTo* nicht angeben.
- `#world` wendet die Inkrementwerte bezogen auf das Koordinatensystem der Welt an. Wenn das Parent-Objekt eines Modells die Welt ist, entspricht dies dem Wert `#parent`.
- Mit `nodeReference` können Sie einen Node angeben, auf dem die Translation basieren soll. Der Befehl wendet die Translationen dann bezogen auf das Koordinatensystem des angegebenen Nodes an.

Beispiel

Das folgende Beispiel erstellt eine Transformation mithilfe des Befehls „transform“, initialisiert die räumliche Position und Ausrichtung der Transformation, ordnet sie dann dem Modell "Mars" zu und zeigt die daraus resultierende Position des Modells an:

```
t =transform()
t.transform.identity()
t.transform.rotate(0, 90, 0)
t.transform.translate(100, 0, 0)
gbModel = member("scene").model("mars")
gbModel.transform = t
put gbModel.transform.position
-- vector(100.0000, 0.0000, 0.0000)
```

Der folgende Lingo-Code verschiebt das Modell „Bip“ um 20 Einheiten auf der x-Achse des zugehörigen Parent-Nodes:

```
put member("Scene").model("Bip").position
-- vector( -38.5000, 21.2500, 2.0000)
member("Scene").model("Bip").translate(20, 10, -0.5)
put member("Scene").model("Bip").position
-- vector( -18.5000, 31.2500, 1.5000)
```

Siehe auch

transform (Eigenschaft), preTranslate(), scale (Befehl), rotate

uncompress()

Syntax

```
byteArrayObject.uncompress()
```

Beschreibung

Diese Bytearray-Methode entpackt den Inhalt des Bytearrays mit dem herkömmlichen Zlib-Verfahren. Nach dem Entpacken wird die Position auf 1 gesetzt.

Beispiele

```
--Lingo syntax
bArray.uncompress()

//JavaScript syntax
bArray.uncompress();
```

union()

Syntax

```
rect(1).union(rect(2))
union (rect1, rect2)
```

Beschreibung

Diese Funktion gibt das kleinste Rechteck zurück, das zwei Rechtecke einschließt.

Parameter

rect2 Erforderlich. Gibt das zweite Rechteck an.

Beispiel

Die folgende Anweisung gibt das Rechteck zurück, das die angegebenen Rechtecke einschließt:

```
-- Lingo syntax
put union (rect (0, 0, 10, 10), rect (15, 15, 20, 20))
-- rect (0, 0, 20, 20)
```

oder

```
put rect(0, 0, 10, 10).union(rect(15, 15, 20, 20))
--rect (0, 0, 20, 20)

// JavaScript syntax
put ( rect (0, 0, 10, 10).union( rect (15, 15, 20, 20) ) );
// <rect(0, 0, 20, 20)>
```

Siehe auch

`map()`, `rect()`

unLoad() (Darsteller)

Syntax

```
-- Lingo syntax
memberObjRef.unLoad({toMemberObjRef})

// JavaScript syntax
memberObjRef.unLoad({toMemberObjRef});
```

Beschreibung

Diese Darstellermethode zwingt Director, die angegebenen Darsteller aus dem Speicher zu löschen.

Director entlädt automatisch die am längsten unbenutzten Darsteller, um Platz für die `preLoad()`-Methode oder das normale Laden der Besetzungsbibliothek zu schaffen.

- Bei Verwendung ohne Parameter löscht `unLoad()` den aktuellen Darsteller aus dem Speicher.
- Bei Verwendung mit dem Parameter *toMemberObjRef* werden alle im angegebenen Bereich enthaltenen Darsteller von `unLoad()` aus dem Speicher gelöscht.

  Wird die Methode in einem neuen Film verwendet, in dem keine Darsteller geladen sind, erscheint eine Fehlermeldung.

Darsteller, die bei der Erstellung oder durch Einstellung von `picture`, `pasteClipBoadInto()` usw. geändert wurden, können nicht entladen werden.

Parameter

toMemberObjRef Optional. Ein Verweis auf den letzten Darsteller im aus dem Speicher zu löschenden Bereich.

Beispiel

Die folgende Anweisung löscht den Darsteller „Ships“ aus dem Speicher:

```
-- Lingo syntax
member("Ships").unLoad()

// JavaScript syntax
member("Ships").unLoad();
```

Die folgende Anweisung löscht die Darsteller 10 bis 15 aus dem Speicher:

```
-- Lingo syntax
member(10).unLoad(15)

// JavaScript syntax
member(10).unLoad(15);
```

Siehe auch

Member, unLoad() (Film), unLoadMember(), unLoadMovie()

unLoad() (Film)

Syntax

```
-- Lingo syntax
_movie.unLoad({intFromFrameNum} {, intToFrameNum})

// JavaScript syntax
_movie.unLoad({intFromFrameNum} {, intToFrameNum});
```

Beschreibung

Diese Filmmethode entfernt den angegebenen Bildbereich des Films aus dem Speicher.

Dieser Befehl ist nützlich, um bei knappem Speicherplatz das Entladen von Filmen zu erzwingen.

Sie können eine URL als Dateibezug verwenden.

Wenn der Film nicht bereits im RAM-Speicher vorhanden ist, ist das Ergebnis -1.

Parameter

intFromFrameNum Optional. Eine Ganzzahl, die die Nummer des ersten Bildes im aus dem Speicher zu entladenden Bereich angibt.

intToFrameNum Optional. Eine Ganzzahl, die die Nummer des letzten Bildes im aus dem Speicher zu entladenden Bereich angibt.

Beispiel

Die folgende Anweisung entlädt die Bilder 10 bis 25 aus dem Speicher.

```
-- Lingo syntax
_movie.unLoad(10, 25)

// JavaScript syntax
_movie.unLoad(10, 25);
```

Siehe auch

Movie, unLoad() (Darsteller), unLoadMember(), unLoadMovie()

unLoadMember()

Syntax

```
-- Lingo syntax
_movie.unLoadMember({memberObjRef})
_movie.unLoadMember(fromMemberNameOrNum, toMemberNameOrNum)

// JavaScript syntax
_movie.unLoadMember({memberObjRef});
_movie.unLoadMember(fromMemberNameOrNum, toMemberNameOrNum);
```

Beschreibung

Diese Filmmethode zwingt Director, einen bestimmten Darsteller oder Darstellerbereich aus dem Speicher zu löschen. Director entlädt automatisch die am längsten unbenutzten Darsteller, um Platz für die `preLoad()`-Methode oder das normale Laden der Besetzungsbibliothek zu schaffen.

- Wenn die `unLoadMember()`-Methode ohne Argumente verwendet wird, werden die Darsteller in allen Bildern des Films aus dem Speicher gelöscht.
- Bei Verwendung mit nur einem Argument, *memberObjRef*, löscht die Methode `unLoadMember()` den angegebenen Darsteller aus dem Speicher.
- Bei Verwendung mit zwei Argumenten, *fromMemberNameOrNum* und *toMemberNameOrNum*, entlädt die Methode `unLoadMember()` alle Darsteller im angegebenen Bereich. Sie können anhand von Darstellernummern oder -namen einen Darstellerbereich definieren.

Parameter

memberObjRef Optional. Ein Verweis auf den aus dem Speicher zu entladenden Darsteller.

fromMemberNameOrNum Erforderlich beim Löschen eines Darstellerbereichs. Eine Zeichenfolge oder Ganzzahl, die den Namen oder die Nummer des ersten Darstellers in einem aus dem Speicher zu entladenden Bereich angibt.

toMemberNameOrNum Erforderlich beim Löschen eines Darstellerbereichs. Eine Zeichenfolge oder Ganzzahl, die den Namen oder die Nummer des letzten Darstellers in einem aus dem Speicher zu entladenden Bereich angibt.

Beispiel

Die folgende Anweisung löscht den Darsteller „Screen1“ aus dem Speicher.

```
-- Lingo syntax
_movie.unLoadMember(member("Screen1"))

// JavaScript syntax
_movie.unLoadMember(member("Screen1"));
```

Die folgende Anweisung löscht alle Darsteller von Darsteller 1 bis Darsteller „Monumentalfilm“ aus dem Speicher:

```
-- Lingo syntax
_movie.unLoadMember(member(1), member("Big Movie"))

// JavaScript syntax
_movie.unLoadMember(member(1), member("Big Movie"));
```

Siehe auch

Movie, unLoad() (Darsteller), unLoad() (Film), unLoadMovie()

unLoadMovie()

Syntax

```
-- Lingo syntax
_movie.unLoadMovie(stringMovieName)

// JavaScript syntax
_movie.unLoadMovie(stringMovieName);
```

Beschreibung

Diese Filmmethode entfernt den angegebenen vorausgeladenen Film aus dem Speicher.

Dieser Befehl ist nützlich, um bei knappem Speicherplatz das Entladen von Filmen zu erzwingen.

Sie können eine URL als Dateibezug verwenden.

Wenn der Film nicht bereits im RAM-Speicher vorhanden ist, ist das Ergebnis -1.

Parameter

stringMovieName Erforderlich. Eine Zeichenfolge, die den Namen des aus dem Speicher zu entladenden Films angibt.

Beispiel

Die folgende Anweisung prüft, ob der größte zusammenhängende Block an freiem Speicherplatz kleiner als 100KB ist und entlädt den Film "Parsifal", wenn dies der Fall ist:

```
-- Lingo syntax
if (_system.freeBlock < (100*1024)) then
    _movie.unLoadMovie("Parsifal")
end if

// JavaScript syntax
if (_system.freeBlock < (100*1024)) {
    _movie.unLoadMovie("Parsifal");
}
```

Die folgende Anweisung entlädt den Film an der URL http://www.cbDemille.com/SunsetBlvd.dir:

```
-- Lingo syntax
_movie.unLoadMovie("http://www.cbDemille.com/SunsetBlvd.dir")

// JavaScript syntax
_movie.unLoadMovie("http://www.cbDemille.com/SunsetBlvd.dir");
```

Siehe auch

Movie, unLoad() (Darsteller), unLoad() (Film), unLoadMember()

unmute (Mixer)

Syntax

```
mixer.unmute()
```

Beschreibung

Mit dieser Mixermethode wird die Eigenschaft `volume` der Mixerausgabe wieder auf den Wert eingestellt, den sie vor dem Aufruf der Funktion `mute()` hatte.

Beispiele

```
--Lingo syntax
on mouseUp me
     mixerRef.unmute() -- Unmutes the mixer.
end

// JavaScript syntax
function mouseUp(){
    mixerRef.unMute(); // Unmutes the mixer.
}
```

Siehe auch

mute (Mixer), Mixer

unmute (Soundobjekt)

Syntax

```
soundObject.Unmute()
```

Beschreibung

Mit dieser Soundobjektmethode wird die Eigenschaft `volume` wieder auf den Wert eingestellt, den sie vor dem Aufruf der Funktion `mute()` hatte.

Beispiele

```
--Lingo syntax
on mouseUp me
     soundObjRef.unmute() -- Unmutes the sound object associated with soundObjRef.
end

// JavaScript syntax
function mouseUp(){
soundObjRef.unmute(); // Unmutes the sound object associated with soundObjRef.
}
```

Siehe auch

mute (Soundobjekt)

unregisterAllEvents

Syntax

```
-- Lingo syntax
member(whichMember).unregisterAllEvents()

// JavaScript syntax
member(whichMember).unregisterAllEvents();
```

Beschreibung

Dieser 3D-Befehl hebt die Registrierung des referenzierten Darstellers für alle Ereignisbenachrichtigungen auf. Dies bedeutet, dass alle Prozeduren, die mit dem Befehl `registerForEvent` registriert worden waren und deshalb auf Ereignisse reagierten, beim Auftreten dieser Ereignisse jetzt nicht mehr ausgelöst werden.

Parameter

Keiner

Beispiel

Die folgende Anweisung hebt die Registrierung des Darstellers "Szene" für alle Ereignisbenachrichtigungen auf:

```
-- Lingo syntax
member("Scene").unregisterAllEvents()

// JavaScript syntax
member("Scene").unregisterAllEvents();
```

Siehe auch

`registerForEvent()`

unregisterByteArrayCallback

Syntax

```
soundObject.unRegisterByteArrayCallback(position)
```

Parameter

Parameter	Beschreibung	Erforderlich/Optional
position	Deregistriert alle Callbacks, falls keine Symbole übergeben werden. Legen Sie die Symbole `#preFilter` oder `#postFilter` in diesem Parameter fest.	Optional.

Beschreibung

Diese Bytearraymethode deregistriert die Bytearray-Callbackfunktion.

Beispiel

```
--Lingo
so.unRegisterByteArrayCallback(#preFilter)
//JavaScript
so.unRegisterByteArrayCallback(#preFilter);
```

Siehe auch

registerByteArrayCallback

unregisterCuePointCallback

Syntax

```
soundObject.unregisterCuePointCallback()
```

Beschreibung

Diese Soundobjektmethode deregistriert die Cuepunkt-Callbackfunktion.

Beispiele

```
--Lingo syntax
on mouseUp me
     soundObjRef.unregisterCuePointCallback() -- Unregisters callback.
end

//JavaScript syntax
function mouseUp(){
soundObjRef.unregisterCuePointCallback(); // Unregisters callback.
}
```

Siehe auch

registerCuePointCallback

unregisterEndOfSpoolCallback()

Syntax

```
soundObject.unregisterEndOfSpoolCallback()
```

Beschreibung

Mit dieser Soundobjektmethode wird die Registrierung des Callbacks „endofSpool“ aufgehoben

Beispiele

```
--Lingo syntax
on mouseUp me
     soundObjRef.unregisterEndOfSpoolCallback() -- Unregisters callback.
end

//JavaScript syntax
function mouseUp(){
soundObjRef.unregisterEndOfSpoolCallback(); // Unregisters callback.
}
```

Siehe auch

registerEndOfSpoolCallback()

update

Syntax

```
-- Lingo syntax
member(whichCastmember).model(whichModel).update

// JavaScript syntax
member(whichCastMember).model(whichModel).update();
```

Beschreibung

Dieser 3D-Befehl aktualisiert Animationen im Modell, ohne sie zu rendern. Mithilfe dieses Befehls können Sie die genaue Position eines Animationsmodells in Lingo ermitteln.

Parameter

Keiner

Beispiel

```
-- Lingo syntax
member(whichCastmember).model(whichModel).update

// JavaScript syntax
member(whichCastMember).getPropRef("model",1).update();
```

updateFrame()

Syntax

```
-- Lingo syntax
_movie.updateFrame()

// JavaScript syntax
_movie.updateFrame();
```

Beschreibung

Diese Filmmethode registriert nur während der Drehbucherstellung die bei der Aufzeichnung des Drehbuchs vorgenommenen Änderungen am aktuellen Bild und geht zum nächsten Bild über. Alle bei Beginn der Aktualisierung bereits im Bild vorhandenen Objekte bleiben im Bild erhalten. Sie müssen die `updateFrame()`-Methode für jedes Bild verwenden, das aktualisiert werden soll.

Parameter

Keiner

Beispiel

Bei Verwendung in der folgenden Prozedur gibt der Befehl `updateFrame` die im aktuellen Bild vorgenommenen Änderungen ein und geht zum nächsten Bild über, wenn Lingo das Ende der Wiederholungsschleife erreicht. Die Anzahl der Bilder wird durch das Argument `numberOfFrames` festgelegt.

```
-- Lingo syntax
on animBall(numberOfFrames)
    _movie.beginRecording()
    horizontal = 0
    vertical = 100
    repeat with i = 1 to numberOfFrames
        _movie.go(i)
        sprite(20).member = member("Ball").number
        sprite(20).locH = horizontal
        sprite(20).locV = vertical
        sprite(20).foreColor = 255
        horizontal = horizontal + 3
        vertical = vertical + 2
        _movie.updateFrame()
    end repeat
    _movie.endRecording()
end animBall

// JavaScript syntax
function animBall(numberOfFrames) {
    _movie.beginRecording();
    var horizontal = 0;
    var vertical = 100;
    for (var i = 1; i <= numberOfFrames; i++) {
        _movie.go(1);
        sprite(20).member = member("Ball");
        sprite(20).locH = horizontal;
        sprite(20).locV = vertical;
        sprite(20).foreColor = 255;
        horizontal = horizontal + 3;
        vertical = vertical + 2;
        _movie.updateFrame();
    }
    _movie.endRecording();
}
```

Siehe auch

beginRecording(), endRecording(), Movie, scriptNum, tweened

updateStage()

Syntax

```
-- Lingo syntax
_movie.updateStage()

// JavaScript syntax
_movie.updateStage();
```

Beschreibung

Diese Filmmethode zeichnet die Bühne sofort neu, nicht nur bei einem Bildwechsel.

Die `updateStage()`-Methode zeichnet Sprites neu, führt Übergänge aus, spielt Sounds ab, sendet eine `prepareFrame`-Nachricht (die sich auf Film- und Verhaltensskripten auswirkt) und sendet eine `stepFrame`-Nachricht (die sich auf `actorList` auswirkt).

Parameter

Keiner

Beispiel

Die folgende Prozedur ändert die horizontale und vertikale Position des Sprites und zeichnet die Bühne neu, sodass das Sprite an der neuen Position erscheint, ohne auf die Bewegung des Abspielkopfs warten zu müssen:

```
-- Lingo syntax
on moveRight(whichSprite, howFar)
    sprite(whichSprite).locH = sprite(whichSprite).locH + howFar
    _movie.updateStage()
end moveRight

// JavaScript syntax
function moveRight(whichSprite, howFar) {
    sprite(whichSprite).locH = sprite(whichSprite).locH + howFar;
    _movie.updateStage();
}
```

Siehe auch

`actorList`, `Movie`, `on prepareFrame`, `on stepFrame`

URLEncode

Syntax

```
URLEncode(proplist_or_string {, serverOSString} {, characterSet})
```

Beschreibung

Diese Funktion gibt den URL-kodierten String für das erste Argument zurück und ermöglicht die Verwendung von CGI-Parametern in anderen Befehlen. Es erfolgt die gleiche Umsetzung wie bei den Funktionen `postNetText` und `getNetText()`, wenn diesen eine Eigenschaftsliste übergeben wird.

Parameter

propListOrString Erforderlich. Gibt die Eigenschaftsliste oder Zeichenfolge an, die URL-kodiert werden soll.

serverOSString Optional. Kodiert alle Return-Zeichen (Zeilenumbrüche) in *propListOrString*. Der Wert lautet standardmäßig `"Unix"`, kann aber auf `"Win"` oder `"Mac"` umgestellt werden und übersetzt in *propListOrString* enthaltene Zeilenschaltungen in Zeichen, die auf dem Server verwendet werden können. Bei den meisten Anwendungen ist diese Einstellung nicht erforderlich, da in Formularantworten normalerweise keine Zeilenumbrüche verwendet werden.

characterSet Optional. Ist nur gültig, wenn der Benutzer ein japanisches Shift-JIS-System verwendet. Mögliche Einstellungen: `"JIS"`, `"EUC"`, `"ASCII"` und `"AUTO"`. Abgerufene Daten werden von Shift-JIS in den angegebenen Zeichensatz umgewandelt. Rückgabedaten werden genauso behandelt wie bei `getNetText()` (d. h. aus dem angegebenen Zeichensatz in Shift-JIS umgewandelt). Bei Verwendung von `"AUTO"` werden die Formulardaten nicht aus dem lokalen Zeichensatz umgewandelt. Die vom Server zurückgesendeten Ergebnisse werden wie bei `getNetText()` konvertiert. `"ASCII"` ist die Standardeinstellung, wenn *characterSet* weggelassen wird. Bei `"ASCII"` werden Formulardaten oder Ergebnisse nicht umgewandelt.

Beispiel

Im folgenden Beispiel übergibt URLEncode den URL-kodierten String an eine CGI-Abfrage an der angegebenen Adresse:

```
URL = "http://aserver/cgi-bin/echoquery.cgi"
gotonetpage URL & "?" & URLEncode( [#name: "Ken", #hobby: "What?"] )
```

Siehe auch

`getNetText()`, `postNetText`

value()

Syntax

```
value(stringExpression)
```

Beschreibung

Diese Funktion gibt den Wert eines Strings zurück. Wenn `value()` aufgerufen wird, parst Lingo den angegebenen Ausdruck (*stringExpression*) und gibt dessen logischen Wert zurück.

Jeder Lingo-Ausdruck, der im Nachrichtenfenster angezeigt (`put`) oder als Wert einer Variablen festgelegt werden kann, kann außerdem mit `value()` verwendet werden.

Die folgenden beiden Lingo-Anweisungen sind gleichbedeutend:

```
put sprite(2).member.duration * 5
put value("sprite(2).member.duration * 5")
```

Die folgenden beiden Lingo-Anweisungen sind ebenfalls gleichbedeutend:

```
x = (the mouseH - 10) / (the mouseV + 10)
x = value("(the mouseH - 10) / (the mouseV + 10)")
```

Ausdrücke, die Lingo nicht parsen kann, führen zwar zu unerwarteten Ergebnissen, aber nicht zu Lingo-Fehlern. Das Ergebnis ist der Wert des anfänglichen Abschnitts des Ausdrucks bis hin zum ersten im String gefundenen Syntaxfehler.

Mit der Funktion `value()` lassen sich Ausdrücke parsen, die von Endbenutzern in Textfelder eingegeben wurden, von Xtras an Lingo übergebene Stringausdrücke sowie alle anderen Ausdrücke, die aus einem String in einen Lingo-Wert umgewandelt werden müssen.

Beachten Sie jedoch, dass die Verwendung von `value()` zusammen mit Benutzereingaben in manchen Fällen gefährlich sein kann, z. B. wenn der Benutzer den Namen einer benutzerdefinierten Prozedur in das Feld eingibt. Die Prozedur wird bei Übergabe an `value()` ausgeführt.

Die Aktionen der Funktion „value“ dürfen nicht mit den Funktionen `integer()` und `float()` verwechselt werden.

Parameter

stringExpression Erforderlich. Gibt die Zeichenfolge an, aus der ein Wert zurückgegeben wird. Der String kann ein beliebiger Ausdruck sein, den Lingo versteht.

Beispiel

Die folgende Anweisung zeigt den numerischen Wert der Zeichenfolge "the sqrt of" && "2.0" an:

```
put value("the sqrt of" && "2.0")
```

Das Ergebnis ist 1.4142.

Die folgende Anweisung zeigt den numerischen Wert der Zeichenfolge „penny“ an:

```
put value("penny")
```

Das im Nachrichtenfenster angezeigte Ergebnis ist VOID, da das Wort *penny* keinen numerischen Wert hat.

Sie können einen String, der als Liste formatiert ist, unter Verwendung der folgenden Syntax in eine echte Liste konvertieren:

```
myString = "[" & QUOTE & "cat" & QUOTE & ", " & QUOTE & "dog" & QUOTE & "]"
myList = value(myString)
put myList
-- ["cat", "dog"]
```

Auf diese Weise kann eine Liste in einen Feld- oder Textdarsteller gestellt und dann extrahiert und erneut als Liste formatiert werden.

Die folgende Anweisung parst den String "3 5" und gibt den Wert des Abschnitts zurück, den Lingo versteht:

```
put value("3 5")
-- 3
```

Siehe auch

`string()`, `integer()`, `float()`

vector()

Syntax

```
-- Lingo syntax
vector()
vector(intX, intY, intZ)

// JavaScript syntax
vector();
vector(intX, intY, intZ);
```

Beschreibung

Diese Top-Level-Funktion und dieser Datentyp beschreiben einen Punkt im 3D-Raum gemäß drei Parametern, bei denen es sich um die jeweils spezifischen Abstände zum Bezugspunkt entlang der *x*-, *y*- und *z*-Achse handelt.

Wenn sich der Vektor im Weltraum befindet, ist der Bezugspunkt der Weltursprung, d. h. `vector(0, 0, 0)`. Wenn sich der Vektor im Objektraum befindet, ist der Bezugspunkt die Position und Ausrichtung des Objekts.

Diese Methode gibt ein Vektorobjekt zurück.

Vektorwerte können mit den Operatoren `+`, `-`, `*` und `/` manipuliert werden. Weitere Informationen finden Sie in den Einträgen zu den einzelnen Operatoren.

Parameter

intX Optional. Eine Ganzzahl, die den Punkt auf der *x*-Achse angibt.

intY Optional. Eine Ganzzahl, die den Punkt auf der *y*-Achse angibt.

intZ Optional. Eine Ganzzahl, die den Punkt auf der *z*-Achse angibt.

Beispiel

Die folgende Anweisung erstellt einen Vektor und weist ihn der Variablen `myVector` zu:

```
-- Lingo syntax
myVector = vector(10.0, -5.0, 0.0)

// JavaScript syntax
var myVector = vector(10.0, -5.0, 0.0);
```

Nur in Lingo addiert die folgende Anweisung zwei Vektoren und weist den Ergebniswert der Variablen `thisVector` zu:

```
-- Lingo syntax
thisVector = vector(1.0, 0.0, 0.0) + vector(0.0, -12.5, 2.0)
```

version()

Syntax

```
-- Lingo syntax
fileioObjRef.version()

// JavaScript syntax
fileioObjRef.version();
```

Beschreibung
Diese Datei-E/A-Methode (Fileio) zeigt die Fileio-Versions- und -Buildinformationen im Nachrichtenfenster an.

Parameter
Keiner

Beispiel
Die folgende Anweisung zeigt die Version von Director im Nachrichtenfenster an:

```
-- Lingo
put(_player.productVersion)

// Javascript
trace(_player.productVersion)
```

Siehe auch
`Fileio`

voiceCount()

Syntax
`voiceCount()`

Beschreibung
Gibt die Anzahl installierter Stimmen zurück, die der Text-to-Speech-Engine zur Verfügung stehen. Der Rückgabewert ist eine Ganzzahl. Dieser Wert kann mit `voiceSet()` und `voiceGet()` zur Aktivierung einer bestimmten Stimme verwendet werden.

Parameter
Keiner

Beispiel
Die folgende Anweisung setzt die Variable `numVoices` auf die Anzahl der verfügbaren Stimmen für die Umwandlung von Text in Sprache:

```
-- Lingo
numVoices = voiceCount()

// Javascript
Var numVoices = voiceCount();
```

Siehe auch
`voiceInitialize()`, `voiceSet()`, `voiceGet()`

voiceGet()

Syntax

```
voiceGet()
```

Beschreibung

Diese Funktion gibt eine Eigenschaftsliste zurück, die die aktuelle Stimme für die Umwandlung von Text in Sprache beschreibt. Die Liste enthält die folgenden Eigenschaften:

- `#name`: Name der installierten Stimme.
- `#age`: Alter der Stimme. Der Wert ist ein String. Mögliche Werte: "Teen", "Adult", "Toddler" und "Senior" sowie numerische Werte wie "35". Die tatsächlichen Werte hängen vom Betriebssystem, der Sprachsoftwareversion und den installierten Stimmen ab.
- `#gender`: Geschlecht der Stimme. Der Wert ist ein String.
- `#index`: Position der Stimme in der Liste mit installierten Stimmen. Bei Verwendung des Befehls `voiceSet()` können Sie über den Index auf eine Stimme Bezug nehmen.

Mithilfe von `voiceCount()` lässt sich die Anzahl verfügbarer Stimmen ermitteln.

Parameter

Keiner

Beispiel

Die folgende Anweisung setzt die Variable `oldVoice` auf die Eigenschaftsliste, die die aktuelle Stimme für die Umwandlung von Text in Sprache beschreibt:

```
-- Lingo
oldVoice = voiceGet()

// Javascript
Var oldVoice = voiceGet();
```

Die folgende Anweisung zeigt die Eigenschaftsliste für die aktuelle Stimme an, die bei der Umwandlung von Text in Sprache verwendet wird:

```
-- Lingo
put voiceGet()
-- [#name: "Mary", #age: "teen", #gender: "female", #index: 5]

// Javascript
trace(voiceget())
// <[#name: "Mary", #age: "teen", #gender: "female", #index: 5]>
```

Siehe auch

`voiceInitialize()`, `voiceCount()`, `voiceSet()`

voiceGetAll()

Syntax

```
voiceGetAll()
```

Beschreibung

Diese Funktion gibt eine Liste der auf dem Computer installierten verfügbaren Stimmen zurück. Die Liste besteht aus Eigenschaftslisten (eine pro verfügbare Stimme).

Jede Liste enthält die folgenden Eigenschaften:

- `#name`: Name der installierten Stimme.
- `#age`: Alter der Stimme. Der Wert ist ein String. Mögliche Werte: "Teen", "Adult", "Toddler" und "Senior" sowie numerische Werte wie "35". Die tatsächlichen Werte hängen vom Betriebssystem, der Sprachsoftwareversion und den installierten Stimmen ab.
- `#gender`: Geschlecht der Stimme.
- `#index`: Position der Stimme in der Liste mit installierten Stimmen. Bei Verwendung des Befehls `voiceSet()` können Sie über den Index auf eine Stimme Bezug nehmen.

Mithilfe von `voiceCount()` lässt sich die Anzahl verfügbarer Stimmen ermitteln.

Parameter

Keiner

Beispiel

Die folgende Anweisung setzt die Variable `currentVoices` auf die Liste der auf dem Benutzercomputer installierten Stimmen:

```
-- Lingo
currentVoices = voiceGetAll()

// Javascript
Var currentVoices = voiceGetAll();
```

Die folgende Anweisung zeigt die Eigenschaftsliste an, in der alle gegenwärtig installierten Text-to-Speech-Stimmen beschrieben werden:

```
-- Lingo
put voiceGetAll()
-- [[#name: "Mary", #age: "teen", #gender: "female", #index: 1], [#name: "Joe", #age: "adult",
#gender: "male", #index: 2]]

// Javascript
trace(voiceGetAll());
// <[[#name: "Mary", #age: "teen", #gender: "female", #index: 1], [#name: "Joe", #age: "adult",
#gender: "male", #index: 2]]>
```

Siehe auch

`voiceInitialize()`, `voiceCount()`, `voiceSet()`, `voiceGet()`

voiceGetPitch()

Syntax

```
voiceGetPitch()
```

Beschreibung
Diese Funktion gibt die aktuelle Tonhöhe der aktuellen Stimme als Ganzzahl zurück. Welche Werte gültig sind, hängt von der Betriebssystemplattform und der Sprachsoftware ab.

Parameter
Keiner

Beispiel
Die folgenden Anweisungen prüfen, ob die Tonhöhe der aktuellen Stimme über 10 liegt. Wenn ja, wird die Tonhöhe auf 10 herabgesetzt.

```
-- Lingo syntax
if voiceGetPitch() > 10 then
    voiceSetPitch(10)
end if

// JavaScript syntax
if (voiceGetPitch() > 10) {
    voiceSetPitch(10);
}
```

Siehe auch
voiceSpeak(), voicePause(), voiceResume(), voiceStop(), voiceGetRate(), voiceSetRate(), voiceSetPitch(), voiceGetVolume(), voiceSetVolume(), voiceState(), voiceWordPos()

voiceGetRate()

Syntax
```
voiceGetRate()
```

Beschreibung
Diese Funktion gibt die aktuelle Wiedergabegeschwindigkeit der Text-to-Speech-Engine zurück. Der Rückgabewert ist eine Ganzzahl. Welche Werte gültig sind, hängt von der Betriebssystemplattform und der Sprachsoftware ab. Im Allgemeinen sind Werte zwischen -10 und 10 zu erwarten.

Parameter
Keiner

Beispiel
Die folgenden Anweisungen prüfen, ob die Sprachgeschwindigkeit unter 50 liegt. Wenn ja, wird sie auf 50 gesetzt.

```
-- Lingo syntax
if voiceGetRate() < 50 then
    voiceSetRate(50)
end if

// JavaScript syntax
if (voiceGetRate() < 50) {
    voiceSetRate(50);
}
```

Siehe auch

`voiceSpeak()`, `voicePause()`, `voiceResume()`, `voiceStop()`, `voiceSetRate()`, `voiceGetPitch()`, `voiceSetPitch()`, `voiceGetVolume()`, `voiceSetVolume()`, `voiceState()`, `voiceWordPos()`

voiceGetVolume()

Syntax

```
voiceGetVolume()
```

Beschreibung

Funktion: Gibt die aktuelle Lautstärke der Sprachsynthese zurück. Bei diesem Wert handelt es sich um eine Ganzzahl. Welche Werte gültig sind, hängt von der Betriebssystemplattform ab.

Parameter

Keiner

Beispiel

Die folgenden Anweisungen prüfen, ob die Sprachsyntheselautstärke mindestens 55 beträgt. Wenn sie niedriger ist, wird sie auf 55 gesetzt.

```
-- Lingo syntax
if voiceGetVolume() < 55 then
    voiceSetVolume(55)
end if

// JavaScript syntax
if (voiceGetVolume() < 55) {
    voiceSetVolume(55);
}
```

Siehe auch

`voiceSpeak()`, `voicePause()`, `voiceResume()`, `voiceStop()`, `voiceGetRate()`, `voiceSetRate()`, `voiceGetPitch()`, `voiceSetPitch()`, `voiceSetVolume()`, `voiceState()`, `voiceWordPos()`

voiceInitialize()

Syntax

```
voiceInitialize()
```

Beschreibung

Dieser Befehl lädt die Text-to-Speech-Engine des Computers. Wenn der Befehl `voiceInitialize()` 0 zurückgibt, ist keine Sprachsoftware installiert bzw. sie konnte nicht geladen werden.

Der Befehl gibt 1 zurück, wenn er erfolgreich ausgeführt werden konnte, andernfalls 0.

Parameter

Keiner

Beispiel

Die folgenden Anweisungen laden die Text-to-Speech-Engine des Computers und prüfen dann, ob diese vollständig geladen ist. Anschließend wird mit dem Befehl `voiceSpeak()` die Nachricht „Willkommen bei Shockwave“ gesprochen:

```
-- Lingo syntax
err = voiceInitialize()
if err = 1 then
    voiceSpeak("Welcome to Shockwave")
else
    alert "Text-to-speech software failed to load."
end if

// JavaScript syntax
err = voiceInitialize();
if (err == 1) {
    voiceSpeak("Welcome to Shockwave");
} else {
    alert("Text-to-speech software failed to load.");
}
```

Siehe auch

`voiceCount()`, `voiceSet()`, `voiceGet()`

voicePause()

Syntax

```
voicePause()
```

Beschreibung

Dieser Befehl hält die Sprachausgabe für die Text-to-Speech-Engine an. Der Befehl gibt 1 zurück, wenn er erfolgreich ausgeführt werden konnte, andernfalls 0.

Parameter

Keiner

Beispiel

Die folgenden Anweisungen halten die Text-to-Speech-Engine an, wenn der Benutzer mit der Maus klickt:

```
-- Lingo syntax
on mouseUp
    voicePause()
end mouseUp

// JavaScript syntax
function mouseUp() {
    voicePause();
}
```

Siehe auch

voiceSpeak(), voiceResume(), voiceStop(), voiceGetRate(), voiceSetRate(), voiceGetPitch(), voiceSetPitch(), voiceGetVolume(), voiceSetVolume(), voiceState(), voiceWordPos()

voiceResume()

Syntax

```
voiceResume()
```

Beschreibung

Dieser Befehl setzt die Sprachausgabe für die Text-to-Speech-Engine fort. Der Befehl gibt 1 zurück, wenn er erfolgreich ausgeführt werden konnte, andernfalls 0.

Parameter

Keiner

Beispiel

Die folgenden Anweisungen setzen die Sprachausgabe fort, wenn der Abspielkopf das nächste Bild im Drehbuch erreicht:

```
-- Lingo syntax
on exitFrame
    voiceResume()
end exitFrame

// JavaScript syntax
function exitFrame() {
    voiceResume();
}
```

Siehe auch

voiceSpeak(), voicePause(), voiceStop(), voiceGetRate(), voiceSetRate(), voiceGetPitch(), voiceSetPitch(), voiceGetVolume(), voiceSetVolume(), voiceState(), voiceWordPos()

voiceSet()

Syntax

```
voiceSet(integer)
```

Beschreibung

Dieser Befehl legt die aktuelle Stimme für die Sprachsynthese zurück. Bei erfolgreicher Ausführung gibt der Befehl den eingestellten Wert zurück. Mithilfe von `voiceCount()` lässt sich die Anzahl verfügbarer Stimmen ermitteln.

Parameter
integer Erforderlich. Eine Ganzzahl, die die Nummer der zu verwendenden Text-to-Speech-Stimme angibt. Welche Werte gültig sind, hängt davon ab, wie viele Stimmen auf dem Computer des Benutzers installiert sind. Bei Eingabe eines ungültigen Werts wird die Stimme mit dem nächstgelegenen Wert verwendet.

Beispiel
Die folgende Anweisung setzt die für die Umwandlung von Text in Sprache zu verwendende Stimme auf die dritte auf dem Computer des Benutzers installierte Stimme:

```
voiceSet(3)
```

Siehe auch
`voiceInitialize()`, `voiceCount()`, `voiceGet()`

voiceSetPitch()

Syntax
```
voiceSetPitch(integer)
```

Beschreibung
Dieser Befehl setzt die Tonhöhe, die von der aktuellen Stimme für die Text-to-Speech-Engine verwendet wird, auf den angegebenen Wert. Der Rückgabewert ist der neue Tonhöhenwert, der soeben eingestellt wurde.

Parameter
integer Erforderlich. Eine Ganzzahl, die die Tonhöhe der Text-to-Speech-Stimme angibt. Welche Werte gültig sind, hängt von der Betriebssystemplattform und der Sprachsoftware ab.

Beispiel
Die folgende Anweisung setzt die Tonhöhe für die aktuelle Stimme auf 75:

```
-- Lingo
voiceSetPitch(75)

// Javascript
voiceSetPitch(75);
```

Siehe auch
`voiceSpeak()`, `voicePause()`, `voiceResume()`, `voiceStop()`, `voiceGetRate()`, `voiceSetRate()`, `voiceGetPitch()`, `voiceGetVolume()`, `voiceSetVolume()`, `voiceState()`, `voiceWordPos()`

voiceSetRate()

Syntax
```
voiceSetRate(integer)
```

Beschreibung
Dieser Befehl setzt die Wiedergabegeschwindigkeit der Text-to-Speech-Engine auf den angegebenen ganzzahligen Wert. Der Befehl gibt den neuen Wert zurück, der soeben eingestellt wurde.

Parameter
integer Erforderlich. Eine Ganzzahl, die die Wiedergabegeschwindigkeit angibt, die von der Text-to-Speech-Engine verwendet werden soll. Welche Werte gültig sind, hängt von der Betriebssystemplattform ab. Im Allgemeinen sind Werte zwischen -10 und 10 für die meisten Speech-Softwareprogramme geeignet. Bei Eingabe eines ungültigen Werts wird die Geschwindigkeit auf den nächstgelegenen gültigen Wert gesetzt.

Beispiel
Die folgende Anweisung setzt die Wiedergabegeschwindigkeit der Text-to-Speech-Engine auf 7:

```
-- Lingo
voiceSetRate(7)

// Javascript
voiceSetRate(7);
```

Siehe auch
voiceSpeak(), voicePause(), voiceResume(), voiceStop(), voiceGetRate(), voiceGetPitch(), voiceSetPitch(), voiceGetVolume(), voiceSetVolume(), voiceState(), voiceWordPos()

voiceSetVolume()

Syntax
```
voiceSetVolume(integer)
```

Beschreibung
Dieser Befehl legt die Lautstärke für die Sprachsynthese fest.

Parameter
integer Erforderlich. Eine Ganzzahl, die die Lautstärke der Text-to-Speech-Synthese angibt. Welche Werte gültig sind, hängt von der Betriebssystemplattform ab. Bei erfolgreicher Ausführung gibt der Befehl den eingestellten Wert zurück. Bei Eingabe eines ungültigen Werts wird die Lautstärke auf den nächstgelegenen gültigen Wert gesetzt.

Beispiel
Die folgende Anweisung setzt die Lautstärke für die Sprachsynthese auf 55:

```
-- Lingo
voiceSetVolume(55)

// Javascript
voiceSetVolume(55);
```

Siehe auch
voiceSpeak(), voicePause(), voiceResume(), voiceStop(), voiceGetRate(), voiceSetRate(), voiceGetPitch(), voiceSetPitch(), voiceGetVolume(), voiceState(), voiceWordPos()

voiceSpeak()

Syntax

```
-- Lingo syntax
voiceSpeak("string")

// JavaScript syntax
voiceSpeak("string"); // documentation n/a
```

Beschreibung

Dieser Befehl bewirkt, dass der angegebene String von der Text-to-Speech-Engine gesprochen wird. Wenn bereits eine Sprachausgabe im Gange ist, wird diese durch den neuen String unterbrochen.

Parameter

string Erforderlich. Die von der Text-to-Speech-Engine zu sprechende Zeichenfolge.

Beispiel

Die folgende Anweisung bewirkt, dass die Text-to-Speech-Engine die Zeichenfolge „Willkommen bei Shockwave“ spricht:

```
voiceSpeak("Welcome to Shockwave")
```

Siehe auch

`voicePause()`, `voiceResume()`, `voiceStop()`, `voiceGetRate()`, `voiceSetRate()`, `voiceGetPitch()`, `voiceSetPitch()`, `voiceGetVolume()`, `voiceSetVolume()`, `voiceState()`, `voiceWordPos()`

voiceState()

Syntax

```
-- Lingo syntax
voiceState()

// JavaScript syntax
voiceState(); // documentation n/a
```

Beschreibung

Diese Funktion gibt den aktuellen Status der Stimme als Symbol zurück. Mögliche Rückgabewerte: `#playing`, `#paused` und `#stopped`.

Parameter

Keiner

Beispiel

Die folgenden Anweisungen prüfen, ob die Text-to-Speech-Engine gegenwärtig aktiv ist, und setzt die Stimme auf 1, wenn dies der Fall ist:

```
--Lingo syntax
if voiceState() <> #playing then
    voiceSet(1)
end if

// JavaScript syntax
if (voiceState() != symbol("playing")) {
    voiceSet(1);
}
```

Siehe auch

voiceSpeak(), voicePause(), voiceResume(), voiceStop(), voiceGetRate(), voiceSetRate(), voiceGetPitch(), voiceSetPitch(), voiceGetVolume(), voiceSetVolume(), voiceWordPos(), voiceSpeak()

voiceStop()

Syntax

```
-- Lingo syntax
voiceStop()

// JavaScript syntax
voiceStop(); // documentation n/a
```

Beschreibung

Dieser Befehl stoppt die Sprachausgabe für die Text-to-Speech-Engine und leert den Sprachausgabepuffer. Der Befehl gibt 1 zurück, wenn er erfolgreich ausgeführt werden konnte, andernfalls 0.

Parameter

Keiner

Beispiel

Die folgenden Anweisungen stoppen die Sprachausgabe fort, wenn der Abspielkopf das nächste Bild im Drehbuch erreicht:

```
-- Lingo syntax
on exitFrame
    voiceStop()
end exitFrame

// JavaScript syntax
function exitFrame() {
    voiceStop();
}
```

Siehe auch

voiceSpeak(), voicePause(), voiceResume(), voiceGetRate(), voiceSetRate(), voiceGetPitch(), voiceSetPitch(), voiceGetVolume(), voiceSetVolume(), voiceState(), voiceWordPos(), voiceSpeak()

voiceWordPos()

Syntax

```
-- Lingo syntax
voiceWordPos()

// JavaScript syntax
voiceWordPos(); // documentation n/a
```

Beschreibung

Diese Funktion gibt eine Ganzzahl zurück, die die Position des gegenwärtig gesprochenen Worts in dem String angibt, in dem es enthalten ist. Wenn beispielsweise gegenwärtig die Sprachausgabe eines Darstellers mit 15 Wörtern erfolgt und bei Verwendung dieser Funktion das fünfte Wort im Darsteller gesprochen wird, lautet der Rückgabewert 5.

Parameter

Keiner

Beispiel

Die folgenden Anweisungen bewirken, dass der Satz „Hallo, wie geht es Ihnen?“ gesprochen und die aktuelle Wortposition im Nachrichtenfenster angezeigt wird. Da die Funktion `voiceWordPos()` unmittelbar nach dem Befehl `voiceSpeak()` aufgerufen wird, lautet der Rückgabewert 1.

```
-- Lingo syntax
voiceSpeak("Hello, how are you?")
put voiceWordPos()
-- 1

// JavaScript syntax
voiceSpeak("Hello, how are you?");
put(voiceWordPos());
// 1
```

Siehe auch

`voiceSpeak()`, `voicePause()`, `voiceResume()`, `voiceStop()`, `voiceGetRate()`, `voiceSetRate()`, `voiceGetPitch()`, `voiceSetPitch()`, `voiceGetVolume()`, `voiceSetVolume()`, `voiceState()`, `voiceSpeak()`

voidP()

Syntax

```
-- Lingo syntax
voidP(variableName)

// JavaScript syntax
variableName == null
```

Beschreibung
Diese Funktion bestimmt, ob eine angegebene Variable einen Wert hat. Wenn die Variable keinen Wert hat oder `VOID` ist, gibt diese Funktion `TRUE` zurück. Wenn die Variable einen anderen Wert als `VOID` hat, gibt diese Funktion `FALSE` zurück.

Parameter
variableName Erforderlich. Gibt die zu testende Variable an.

Beispiel
Die folgende Anweisung prüft, ob die Variable answer einen Anfangswert hat:

```
-- Lingo syntax
put voidP(answer)

// JavaScript syntax
put(answer == null));
```

Siehe auch
`ilk()`, `VOID`

window()

Syntax

```
-- Lingo syntax
window(stringWindowName)

// JavaScript syntax
window(stringWindowName);
```

Beschreibung
Diese Top-Level-Funktion gibt einen Verweis auf ein angegebenes Fenster zurück.

Das angegebene Fenster muss einen Director-Film enthalten.

Fenster, in denen Filme ablaufen, sind nützlich zur Erstellung von schwebenden Paletten, separaten Steuerpulten und Fenstern mit unterschiedlicher Form. Anhand von Fenstern, in denen Filme ablaufen, können Sie mehrere Filme auf einmal öffnen und interaktiv kommunizieren lassen.

Parameter
stringWindowName Erforderlich. Eine Zeichenfolge, die den Namen des zu referenzierenden Fensters angibt.

Beispiel
Die folgende Anweisung legt die Variable `myWindow` auf das Fenster namens `Collections` fest:

```
-- Lingo syntax
myWindow = window("Collections")

// JavaScript syntax
var myWindow = window("Collections");
```

Siehe auch

Window

WindowOperation

Syntax

```
WindowOperation(MUIObject, operation)
```

Beschreibung

Dieser Befehl steuert das Fenster eines allgemeinen Dialogfelds.

Ersetzen Sie den Parameter „operation“ durch einen Wert, der den Zweck des Fensters bestimmt. Es folgen mögliche Werte und ihr Ergebnis:

Mögliche Werte	Ergebnis
#show	Zeigt nur ein nicht modales Dialogfeld an. (Öffnen Sie über den Befehl „Run“ ein modales Dialogfeld.)
#hide	Blendet ein nicht modales Dialogfeld aus. (Schließen Sie über den Befehl „Stop“ ein modales Dialogfeld.)
#center	Zentriert das Fenster auf dem Bildschirm.
#zoom	Sendet die Nachricht, dass der Benutzer auf das Zoomfeld des Fensters geklickt hat. Die Rückrufprozedur muss die Größe des Dialogfelds ändern, wenn sich die Größe des Fensters ändern soll, nachdem der Benutzer auf das Zoomfeld geklickt hat.
#tipsOn	Aktiviert QuickInfos. (Für künftige Versionen des Xtras MUI reserviert.)
#tipsOff	Deaktiviert QuickInfos. (Für künftige Versionen des Xtras MUI reserviert.)

Beispiel

Diese Prozedur prüft, ob „MUIObject“ vorhanden ist, und zeigt, falls ja, das Dialogfeld an:

```
--Lingo syntax
on showDialog
    global MUIObject
    if objectP( MUIObject ) then
        WindowOperation( MUIObject, #show )
    end if
end showDialog
```

Diese Anweisung blendet das aus „MUIObject“ erstellte Dialogfeld aus:

```
WindowOperation(MUIObject, #hide)
```

windowPresent()

Syntax

```
-- Lingo syntax
_player.windowPresent(stringWindowName)

// JavaScript syntax
_player.windowPresent(stringWindowName);
```

Beschreibung
Diese Player-Methode zeigt an, ob das in *stringWindowName* angegebene Objekt als Film in einem Fenster ausgeführt wird (`TRUE`) oder nicht (`FALSE`).

Wenn ein Fenster geöffnet war, bleibt `windowPresent()` für das Fenster so lange `TRUE`, bis das Fenster aus der Eigenschaft `windowList` entfernt wird.

Das Argument *stringWindowName* muss mit dem Fensternamen identisch sein, der in der Eigenschaft `windowList` erscheint.

Parameter
stringWindowName Erforderlich. Eine Zeichenfolge, die den Namen des zu testenden Fensters angibt.

Beispiel
Die folgende Anweisung testet, ob das Objekt `myWindow` ein Film in einem Fenster (MIAW) ist, und zeigt das Ergebnis im Nachrichtenfenster an:

```
-- Lingo syntax
put(_player.windowPresent(myWindow))

// JavaScript syntax
put(_player.windowPresent(myWindow));
```

Siehe auch
`Player`, `windowList`

worldSpaceToSpriteSpace

Syntax
```
-- Lingo syntax
member(whichCastmember).camera(whichCamera).worldSpaceToSpriteSpace(vector)

// JavaScript syntax
member(whichCastmember).camera(whichCamera).worldSpaceToSpriteSpace(vector);
```

Beschreibung
Dieser 3D-Befehl gibt den Punkt im Rechteck der Kamera zurück, an dem eine angegebene weltbezogene Position erscheint. Die von diesem Befehl zurückgegebene Position ist relativ zur oberen linken Ecke des Kamerarechtecks.

Wenn sich die angegebene Position außerhalb des Sichtfelds der Kamera befindet, gibt dieser Befehl `void` zurück.

Parameter
vector Erforderlich. Gibt die weltbezogene Position an, die erscheint.

Beispiel
Die folgende Anweisung zeigt, dass der durch „vector(0, 0, 0)“ gezeichnete Weltursprung an „point(250,281)“ im Rechteck der Kamera erscheint:

```
-- Lingo syntax
put sprite(5).camera.worldSpaceToSpriteSpace(vector(0, 0, 0))
-- point(250, 281)

// JavaScript syntax
put(sprite(5).camera.worldSpaceToSpriteSpace(vector(0,0,0)));
```

Siehe auch

spriteSpaceToWorldSpace, rect (camera)

writeBoolean

Syntax

```
byteArrayObject.writeBoolean(boolVal)
```

Beschreibung

Diese Bytearraymethode schreibt einen Boole'schen Wert in das Bytearray. Boole'sche Werte belegen jeweils ein Byte. Schreibt eine 1, falls das Bytearray einen von 0 verschiedenen Wert hat; schreibt ansonsten eine 0.

Parameter

Parameter	Beschreibung	Erforderlich/Optional
boolVal	Boole'scher Wert, der in das Byte-Array geschrieben wird.	Erforderlich

Beispiele

```
--Lingo syntax
put bArray.writeBoolean(1)

//JavaScript syntax
put(bArray.writeBoolean(1));
```

writeByteArray

Syntax

```
writeByteArray(byteArray, [intOffset],[ intLen])
```

Beschreibung

Diese Bytearraymethode schreibt ein Bytearray ganz oder teilweise in ein anderes Bytearray.

Parameter

Parameter	Beschreibung	Erforderlich/Optional
byteArray	Bestehendes Byte-Array.	Erforderlich
intOffset	Anfangsposition.	Optional
intLen	Länge des Byte-Arrays.	Optional

Beispiele

```
--Lingo syntax
bArray.writeByteArray(gByteArray, 1, 10)

//JavaScript syntax
bArray.writeByteArray(gByteArray, 1, 10);
```

writeByteArray (FileIO Xtra)

Syntax

```
FileIO.writeByteArray(byteArray, [intOffset],[intLen])
```

Beschreibung

Diese Bytearraymethode schreibt ein Bytearray ganz oder teilweise in eine Datei.

Parameter

Parameter	Beschreibung	Erforderlich/Optional
byteArray	Bestehendes Byte-Array.	Erforderlich
intOffset	Anfangsposition.	Optional
intLen	Länge des Byte-Arrays.	Optional

Beispiele

```
--Lingo syntax
FileIO.writeByteArray(gByteArray, 1, 10)

//JavaScript syntax
FileIO.writeByteArray(gByteArray, 1, 10);
```

writeChar()

Syntax

```
-- Lingo syntax
fileioObjRef.writeChar(stringChar)

// JavaScript syntax
fileioObjRef.writeChar(stringChar)
```

Beschreibung

Diese Datei-E/A-Methode (Fileio) schreibt ein einzelnes, angegebenes ASCII-Zeichen in eine Datei.

Sie müssen zuerst durch Aufrufen von `openFile()` eine Datei öffnen, bevor Sie mithilfe von `writeChar()` ein Zeichen schreiben können.

Parameter

stringChar Erforderlich. Gibt das in die Datei zu schreibende ASCII-Zeichen an.

Beispiel

Die folgende Anweisung öffnet eine Datei „c:\xtra.txt“ mit Lese-/Schreibberechtigung, schreibt das Zeichen „d“ in die Datei und schließt sie.

```
-- Lingo
objFileio = new xtra("fileio")
objFileio.openFile("c:\xtra.txt",0)
objFileio.writeChar("d")
objFileio.closeFile()

// JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.openFile("c:\xtra.txt",0);
objFileio.writeChar("d");
objFileio.closeFile();
```

Siehe auch

`Fileio`

writeFloat32

Syntax

```
byteArrayObject.writeFloat32(floatVal)
```

Beschreibung

Diese Bytearraymethode schreibt einen 32-Bit-Fließkommawert in ein Byte-Array.

Parameter

Parameter	Beschreibung	Erforderlich/Optional
floatVal	Fließkommawert, der in das Byte-Array geschrieben wird.	Erforderlich

Beispiele

```
--Lingo syntax
put bArray.writeFloat32(3.3)

//JavaScript syntax
put(bArray.writeFloat32(3.3));
```

writeFloat64

Syntax

```
byteArrayObject.writeFloat64(floatVal)
```

Beschreibung

Diese Bytearraymethode schreibt einen 64-Bit-Fließkommawert in ein Byte-Array.

Parameter

Parameter	Beschreibung	Erforderlich/Optional
floatVal	Fließkommawert, der in das Byte-Array geschrieben wird.	Erforderlich

Beispiele

```
--Lingo syntax
put bArray.writeFloat64(36666666.366666)

//JavaScript syntax
put(bArray.writeFloat64(36666666.366666));
```

writeInt8

Syntax

```
byteArrayObject.writeInt8(intVal)
```

Beschreibung

Diese Bytearraymethode schreibt eine vorzeichenbehaftete 8-Bit-Ganzzahl in ein Byte-Array. Falls der Eingabewert nicht mit acht Bits dargestellt werden kann, berücksichtigt die Methode nur das niederwertige Byte.

Parameter

Parameter	Beschreibung	Erforderlich/Optional
intVal	Integer-Wert, der in das Byte-Array geschrieben wird.	Erforderlich

Beispiele

```
--Lingo syntax
put bArray.writeInt8(16)

//JavaScript syntax
put(bArray.writeInt8(16));
```

writeInt16

Syntax

```
byteArrayObject.writeInt16(intVal)
```

Beschreibung
Diese Bytearraymethode schreibt eine vorzeichenbehaftete 16-Bit-Ganzzahl in ein Byte-Array. Falls der Eingabewert nicht mit 16 Bits dargestellt werden kann, berücksichtigt die Methode nur die beiden niederwertigen Bytes.

Parameter

Parameter	Beschreibung	Erforderlich/Optional
intVal	Integer-Wert, der in das Byte-Array geschrieben wird.	Erforderlich

Beispiele

```
--Lingo syntax
put bArray.writeInt16(1234)

//JavaScript syntax
put(bArray.writeInt16(1234));
```

writeInt32

Syntax

```
byteArrayObject.writeInt32(intVal)
```

Beschreibung
Diese Bytearraymethode schreibt eine vorzeichenbehaftete 32-Bit-Ganzzahl in ein Byte-Array.

Parameter

Parameter	Beschreibung	Erforderlich/Optional
intVal	Integer-Wert, der in das Byte-Array geschrieben wird.	Erforderlich

Beispiele

```
--Lingo syntax
put bArray.writeInt32(56781234)

//JavaScript syntax
put(bArray.writeInt32(56781234));
```

writeRawString

Syntax

```
writeRawString(strValue, intLen,[strCharSet])
```

Beschreibung
Diese Bytearraymethode schreibt eine feste Anzahl von Bytes in ein Bytearray. Der Text wird falls nötig mit NULL-Zeichen aufgefüllt.

Parameter

Parameter	Beschreibung	Standardwert
strValue	String, der in das Byte-Array geschrieben wird.	Erforderlich
intLen	Länge des Strings, der vom Byte-Array gelesen werden soll.	Erforderlich
strCharSet	Legt die erforderliche Encoding-Übersetzung fest.	UTF-8

Beispiele

```
--Lingo syntax
bArray.writeRawString("Director",13)

//JavaScript syntax
bArray.writeRawString("Director",13);
```

writeReturn()

Syntax

```
-- Lingo syntax
fileioObjRef.writeReturn(symbol(""))

// JavaScript syntax
fileioObjRef.writeReturn(symbol(""));
```

Beschreibung

Diese Datei-E/A-Methode (Fileio) fügt einen Zeilenumbruch in eine Datei ein.

Parameter

Keiner

Beispiel

Die folgende Anweisung öffnet eine Datei „c:\xtra.txt“ mit Lese-/Schreibberechtigung, schreibt die Zeile „First Line“ in die Datei, gefolgt von einem Zeilenumbruch und der zweiten Zeile „Second Line“, und schließt die Datei.

```
-- Lingo
objFileio = new xtra("fileio")
objFileio.openFile("c:\xtra.txt",0)
objFileio.writeString("First Line")
objFileio.writeReturn(#windows)
objFileio.writeString("Second Line")
objFileio.closeFile()

// JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.openFile("c:\xtra.txt",0);
objFileio.writeString("First Line");
objFileio.writeReturn(Symbol("windows"));
objFileio.writeString("Second Line");
objFileio.closeFile();
```

Siehe auch

Fileio

writeString()

Syntax

```
-- Lingo syntax
fileioObjRef.writeString(string)

// JavaScript syntax
fileioObjRef.writeString(string)
```

Beschreibung

Diese Datei-E/A-Methode (Fileio) schreibt eine auf Null endende Zeichenfolge in eine Datei.

Parameter

string Erforderlich. Die in eine Datei zu schreibende Zeichenfolge.

Beispiel

Die folgende Anweisung öffnet eine Datei „c:\xtra.txt“ mit Lese-/Schreibberechtigung, schreibt die Zeile „First Line“ sowie die zweite Zeile „Second Line“ in die Datei und schließt sie.

```
-- Lingo
objFileio = new xtra("fileio")
objFileio.openFile("c:\xtra.txt",0)
objFileio.writeString("First Line")
objFileio.writeString("Second Line")
objFileio.closeFile()

// JavaScript syntax
var objFileio = new xtra("fileio");
objFileio.openFile("c:\xtra.txt",0);
objFileio.writeString("First Line");
objFileio.writeString("Second Line");
objFileio.closeFile();
```

Siehe auch

Fileio

writeString (byte array)

Syntax

```
writeString(strValue, [strCharSet])
```

Beschreibung

Diese Bytearraymethode schreibt einen String in ein Bytearray. Die ersten vier Bytes des String-Blob enthalten die Länge des Strings.

Parameter

Parameter	Beschreibung	Standardwert
strValue	String, der in das Byte-Array geschrieben wird.	Erforderlich
strCharSet	Legt die erforderliche Encoding-Übersetzung fest.	UTF-8

Beispiele

```
--Lingo syntax
bArray.writeString("Director")

//JavaScript syntax
bArray.writeString("Director");
```

xtra()

Syntax

```
-- Lingo syntax
xtra(xtraNameOrNum)

// JavaScript syntax
xtra(xtraNameOrNum);
```

Beschreibung

Diese Top-Level-Funktion gibt eine Instanz eines angegebenen Xtras zurück.

Wenn das angegebene Xtra nicht gefunden wird, wird ein Verweis auf ein leeres Objekt zurückgegeben.

Ein Beispiel für `xtra` in einem fertigen Film finden Sie im Film „Read and Write Text“ im Ordner „Learning\\Lingo_“ im Director-Anwendungsordner.

Parameter

xtraNameOrNum Erforderlich. Eine Zeichenfolge, die den Namen des zurückzugebenden Xtras angibt, oder eine Ganzzahl, die die Indexposition des zurückzugebenden Xtras angibt. Bei Zeichenfolgennamen wird nicht zwischen Groß- und Kleinschreibung unterschieden.

Beispiel

Die folgende Anweisung legt die Variable `myNetLingo` auf die Xtra-NetLingo-Erweiterung fest:

```
-- Lingo syntax
myNetLingo = xtra("netlingo")

// JavaScript syntax
var myNetLingo = xtra("netlingo");
```

zoomBox

Syntax

```
-- Lingo syntax
zoomBox startSprite, endSprite {,delayTicks}

// JavaScript syntax
zoomBox(startSprite, endSprite {,delayTicks}); // not yet documented
```

Beschreibung

Dieser Befehl erstellt einen Zoomeffekt ähnlich den erweiterbaren Fenstern im Mac-Finder. Der Zoomeffekt beginnt am Begrenzungsrechteck eines angegebenen Ausgangs-Sprites und endet am Begrenzungsrechteck eines angegebenen End-Sprites. Der Befehl `zoomBox` verwendet bei der Ausführung folgende Logik:

1 Nach *endSprite* im aktuellen Bild suchen; sonst,

2 Nach *endSprite* im nächsten Bild suchen.

Beachten Sie jedoch, dass der Befehl `zoomBox` nicht funktioniert, wenn sich *endSprite* im selben Kanal befindet wie *startSprite*.

Parameter

startSprite Erforderlich. Gibt das Ausgangs-Sprite an.

endSprite Erforderlich. Gibt das End-Sprite an.

delayTicks Optional. Gibt die Verzögerung in Ticks zwischen den einzelnen Bewegungen der Zoomrechtecke an. Wenn *delayTicks* nicht angegeben wird, ist die Verzögerung gleich 1.

Beispiel

Die folgende Anweisung erstellt einen Zoomeffekt zwischen Sprites 7 und 3:

```
-- Lingo syntax
zoomBox 7, 3

// JavaScript syntax
zoomBox(7, 3); // not yet documented
```

Kapitel 13: Operatoren

In diesem Abschnitt finden Sie eine alphabetische Liste aller in Director ® verfügbaren Operatoren.

Die Mehrheit dieser Operatoren ist nur in Lingo gültig. JavaScript-Syntax enthält einige Operatoren, die den aufgeführten Lingo-Operatoren ähnlich oder mit ihnen identisch sind. Daher werden an gegebener Stelle die Syntax und Beispiele für JavaScript-Syntax bereitgestellt, um Ihnen bei der Zuordnung der Funktionalität von Lingo-Operatoren zu ihren passendsten JavaScript-Syntax-Entsprechungen zu helfen. Weitere Informationen über die Syntax von JavaScript-Operatoren finden Sie unter „Grundlagen der Skripterstellung in Director“ auf Seite 4.

(symbol)

Syntax

```
--Lingo syntax
#symbolName

// JavaScript syntax
symbol("symbolName");
```

Beschreibung

Dieser Symboloperator beschreibt ein Symbol, eine selbständige Einheit, die eine Bedingung oder ein Attribut darstellen kann. Der Wert *symbolName* beginnt mit einem alphabetischen Zeichen, auf das eine beliebige Anzahl alphabetischer oder numerischer Zeichen folgen können.

Mit einem Symbol können Sie folgende Aufgaben ausführen:

- einer Variablen einen Wert zuweisen
- Strings, Ganzzahlen, Rechtecke und Punkte miteinander vergleichen
- einen Parameter an eine Prozedur oder eine Methode übergeben
- einen Wert von einer Prozedur oder einer Methode zurückgeben

Ein Symbol nimmt weniger Platz ein als ein String und kann manipuliert werden. Es besteht jedoch nicht wie ein String aus einzelnen Zeichen. Mit der Funktion `string` können Sie ein Symbol zu Anzeigezwecken in einen String konvertieren.

Es folgen einige wichtige Punkte zur Symbolsyntax:

- Symbole unterscheiden nicht zwischen Groß- und Kleinschreibung.
- Symbole dürfen nicht mit einer Zahl beginnen.
- Es dürfen keine Leerzeichen verwendet werden. Leerzeichen können jedoch mit Unterstrichzeichen simuliert werden.
- Symbole benutzen die 128 ASCII-Zeichen. Buchstaben mit diakritischen Zeichen oder Akzenten werden wie der entsprechende Grundbuchstabe behandelt.
- Punkte dürfen in Symbolen nicht verwendet werden.

Alle Symbole, globalen Variablen und Parameternamen, die an globale Variablen übergeben werden, sind in einer gemeinsamen Nachschlagetabelle gespeichert.

Beispiel

Die folgende Anweisung stellt die Variable „state“ auf das Symbol `#Playing` ein:

```
-- Lingo syntax
state = #Playing

// JavaScript syntax
var state = symbol("Playing");
```

Siehe auch

`ilk()`, `string()`, `symbol()`, `symbolP()`

. (Punkt-Operator)

Syntax

```
-- Lingo syntax
objectReference.objectProperty
textExpression.objectProperty
object.commandOrFunction()

// JavaScript syntax
objectReference.objectProperty;
textExpression.objectProperty;
object.commandOrFunction();
```

Beschreibung

Dieser Operator wird verwendet, um Objekteigenschaften zu testen oder einzustellen, einen Befehl zu erteilen oder eine Funktion des Objekts auszuführen. Das Objekt kann ein Darsteller, ein Sprite, eine Eigenschaftsliste, ein Child-Objekt eines Parent-Skripts oder ein Verhalten sein.

Beispiel

Die folgende Anweisung gibt den aktuellen Darsteller an, der im Sprite in Kanal 10 enthalten ist:

```
-- Lingo syntax
put(sprite(10).member)

// JavaScript syntax
put(sprite(10).member);
```

Um mit der alternativen Syntax eine Funktion aufzurufen, verwenden Sie folgendes Format:

```
-- Lingo syntax
myColorObject = color(124, 22, 233)
put(myColorObject.ilk())
-- #color

// JavaScript syntax
var myColorObject = color(124, 22, 233);
put(myColorObject.ilk());
// #color
```

- (Minuszeichen)

Syntax

```
-- Lingo syntax
(Negation): -expression
(Subtraction): expression1 - expression2

// JavaScript syntax
(Negation): -expression
(Subtraction): expression1 - expression2
```

Beschreibung

Der mathematische Operator „-“ (Minus) kehrt das Vorzeichen des Werts von *expression* um, wenn er zur Negation verwendet wird. Wird er zur Subtraktion verwendet, führt „-“ (Minus) eine arithmetische Subtraktion zwischen zwei numerischen Ausdrücken durch, wobei *expression2* von *expression1* subtrahiert wird.

„-“ (Minus) hat bei Verwendung in Negationen als arithmetischer Operator die Prioritätsstufe 5.

Bei der Subtraktion sind beide Ausdrücke Ganzzahlen, deren Differenz ebenfalls eine Ganzzahl ist. Wenn einer oder beide Ausdrücke Fließkommazahlen sind, ist die Differenz eine Fließkommazahl. Der Operator „-“ (Minus) ist ein arithmetischer Operator mit der Prioritätsstufe 3.

Beispiel

(Die folgende Anweisung kehrt das Vorzeichen des Ausdrucks 2 + 3 um:

```
-- Lingo syntax
put(-(2 + 3))

// JavaScript syntax
put(-(2 + 3));
```

Das Ergebnis ist -5.

(Subtraktion) : Die folgende Anweisung subtrahiert die Ganzzahl 2 von der Ganzzahl 5 und zeigt das Ergebnis im Nachrichtenfenster an:

```
-- Lingo syntax
put(5 - 2)

// JavaScript syntax
put(5 - 2);
```

Das Ergebnis ist die Ganzzahl 3.

(Subtraktion) : Die folgende Anweisung subtrahiert die Fließkommazahl 1.5 von der Fließkommazahl 3.25 und zeigt das Ergebnis im Nachrichtenfenster an:

```
-- Lingo syntax
put(3.25 - 1.5)

// JavaScript syntax
put(3.25 - 1.5);
```

Das Ergebnis ist die Fließkommazahl 1.75.

-- (comment)

Syntax

```
-- Lingo syntax
-- comment

// JavaScript syntax
// comment
```

Beschreibung

Dieser Kommentarbegrenzer zeigt den Anfang eines Skriptkommentars an. Was in einer beliebigen Zeile zwischen dem Kommentarbegrenzer (zwei Bindestrichen) und dem Zeilenumbruch am Zeilenende steht, wird als Kommentar und nicht als Lingo-Anweisung interpretiert.

Beispiel

Die folgende Prozedur verwendet zwei Bindestriche, um die zweite, vierte und sechste Zeile zu einem Kommentar zu machen:

```
-- Lingo syntax
on resetColors
    -- This handler resets the sprite's colors.
    sprite(1).forecolor = 35
    -- bright red
    sprite(1).backcolor = 36
    -- light blue
end

// JavaScript syntax
function resetColors() {
    // this handler resets the sprite's colors
    sprite(1).forecolor = 35;
    // bright red
    sprite(1).backcolor = 36;
    // light blue
}
```

&, + (Verkettungsoperator)

Syntax

```
-- Lingo syntax
expression1 & expression2

// JavaScript syntax
expression1 + expression2
```

Beschreibung

Dieser String-Operator verkettet zwei Ausdrücke. Wenn entweder *expression1* oder *expression2* eine Zahl ist, wird sie zuerst in einen String konvertiert. Der resultierende Ausdruck ist ein String.

Dieser String-Operator hat die Prioritätsstufe 2.

In Lingo können einige Befehle und Funktionen, die nur ein Argument akzeptieren, ohne Klammern um das Argument verwendet werden. Wenn ein Argumentausdruck einen Operator enthält, interpretiert Lingo nur das erste Argument als Teil der Funktion, was Probleme verursachen kann.

Sie können dieses Problem vermeiden, indem Sie den gesamten Ausdruck, der einen Operator einschließt, in Klammern setzen. Die Klammern verhindern ein Missverständnis, da hierdurch die Priorität, nach der Lingo den Operator behandelt, geändert wird. Lingo behandelt die zwei Teile des Arguments jetzt als ganzes Argument.

Beispiel

Die folgende Anweisung verkettet die Strings "Abra" und "kadabra" und zeigt das Ergebnis im Nachrichtenfenster an:

```
-- Lingo syntax
put("abra" & "cadabra")

// JavaScript syntax
put("abra" + "cadabra");
```

Das Ergebnis ist der String „Abrakadabra“.

Die folgende Anweisung verkettet den String "$" mit dem Inhalt der Variablen `price` und ordnet den verketteten String dem Felddarsteller "Preis" zu.

```
-- Lingo syntax
member("Price").text = "$" & price

// JavaScript syntax
member("Price").text = "$" + price;
```

Siehe auch

&&, + (Verkettungsoperator)

&&, + (Verkettungsoperator)

Syntax

```
-- Lingo syntax
expression1 && expression2

// JavaScript syntax
expression1 + expression2
```

Beschreibung

Dieser String-Operator verkettet zwei Ausdrücke und fügt zwischen die ursprünglichen Strings ein Leerzeichen ein. Wenn entweder *expression1* oder *expression2* eine Zahl ist, wird sie zuerst in einen String konvertiert. Der resultierende Ausdruck ist ein String.

Dieser String-Operator hat die Prioritätsstufe 2.

Beispiel

Die folgende Anweisung verkettet die Strings "Abra" und "kadabra" und fügt ein Leerzeichen ein:

```
-- Lingo syntax
put("abra" && "cadabra")

// JavaScript syntax
put("abra " + "cadabra");
```

Das Ergebnis ist der String „Abra kadabra“.

Die folgende Anweisung verkettet die Strings "Heute ist" und das heutige Datum im langen Format und fügt zwischen beide ein Leerzeichen ein:

```
-- Lingo syntax
put("Today is" && date())

// JavaScript syntax
put("Today is " + Date());
```

Siehe auch

&, + (Verkettungsoperator)

() (Klammern)

Syntax

```
-- Lingo syntax
(expression)

// JavaScript syntax
(expression)
```

Beschreibung

Dieser Gruppierungsoperator führt eine Gruppierungsoperation für einen Ausdruck aus, um die Reihenfolge zu steuern, in der die Operatoren im Ausdruck ausgeführt werden. Dieser Operator hebt die automatische Prioritätsordnung auf, damit der Ausdruck in der Klammer zuerst ausgewertet wird. Bei verschachtelten Klammern wird der Inhalt der inneren Klammer vor dem Inhalt der äußeren ausgewertet.

Dieser Gruppierungsoperator hat die Prioritätsstufe 5.

Beachten Sie, dass in Lingo einige Befehle und Funktionen, die nur ein Argument akzeptieren, ohne Klammern um das Argument verwendet werden können. Wenn ein Argumentausdruck einen Operator enthält, interpretiert Lingo nur das erste Argument als Teil der Funktion, was Probleme verursachen kann.

Zum Beispiel lässt der Befehl `open window` ein Argument zu, das angibt, welches Fenster geöffnet wird. Wenn Sie den Operator `&` verwenden, um einen Pfad- und einen Dateinamen zu definieren, interpretiert Director nur den String vor dem `&`-Operator als Dateinamen. Lingo interpretiert beispielsweise die Anweisung `open window the applicationPath & "theMovie"` als `(open window the applicationPath) & ("theMovie")`. Sie können dieses Problem vermeiden, indem Sie den gesamten Ausdruck, der einen Operator einschließt, in Klammern setzen, wie das folgende Beispiel zeigt:

```
-- Lingo syntax
open window (the applicationPath & "theMovie")

// JavaScript syntax
window(the applicationPath + "theMovie").open();
```

Beispiel

Die folgenden Anweisungen verwenden den Gruppierungsoperator, um die Reihenfolge, in der die Operationen ausgeführt werden, zu ändern (das Ergebnis wird unterhalb der einzelnen Anweisungen angezeigt):

```
-- Lingo syntax
put((2 + 3) * (4 + 5))
-- 45
put(2 + (3 * (4 + 5)))
-- 29
put(2 + 3 * 4 + 5)
-- 19

// JavaScript syntax
put((2 + 3) * (4 + 5));
// 45
put(2 + (3 * (4 + 5)));
// 29
put(2 + 3 * 4 + 5);
// 19
```

* (Multiplikation)

Syntax

```
-- Lingo syntax
expression1 * expression2

// JavaScript syntax
expression1 * expression2
```

Beschreibung

Dieser mathematische Operator führt eine arithmetische Multiplikation zweier numerischer Ausdrücke durch. Wenn beide Ausdrücke Ganzzahlen sind, ist das Produkt eine Ganzzahl. Wenn einer oder beide Ausdrücke Fließkommazahlen sind, ist das Ergebnis eine Fließkommazahl.

Dieser arithmetische Operator hat die Prioritätsstufe 4.

Beispiel

Die folgende Anweisung multipliziert die Ganzzahlen 2 und 3 und zeigt das Ergebnis im Nachrichtenfenster an:

```
-- Lingo syntax
put(2 * 3)

// JavaScript syntax
put(2 * 3);
```

Das Ergebnis ist die Ganzzahl 6.

Die folgende Anweisung multipliziert die Fließkommazahlen 2.0 und 3.1414 und zeigt das Ergebnis im Nachrichtenfenster an:

```
-- Lingo syntax
put(2.0 * 3.1416)

// JavaScript syntax
put(2.0 * 3.1416);
```

Das Ergebnis ist die Fließkommazahl 6.2832.

+ (Addition)

Syntax

```
-- Lingo syntax
expression1 + expression2

// JavaScript syntax
expression1 + expression2
```

Beschreibung

Dieser mathematische Operator addiert zwei numerische Ausdrücke zu einer arithmetischen Summe. Wenn beide Ausdrücke Ganzzahlen sind, ist die Summe eine Ganzzahl. Wenn einer oder beide Ausdrücke Fließkommazahlen sind, ist die Summe eine Fließkommazahl.

Dieser arithmetische Operator hat die Prioritätsstufe 4.

Beispiel

Die folgende Anweisung addiert die Ganzzahlen 2 und 3 und zeigt das Ergebnis, die Ganzzahl 5, im Nachrichtenfenster an:

```
-- Lingo syntax
put(2 + 3)
// JavaScript syntax
put(2 + 3);
```

Die folgende Anweisung addiert die Fließkommazahlen 2.5 und 3.25 und zeigt das Ergebnis, die Fließkommazahl 5.7500, im Nachrichtenfenster an:

```
-- Lingo syntax
put(2.5 + 3.25)
// JavaScript syntax
put(2.5 + 3.25);
```

+ (Addition) (3D)

Syntax

```
-- Lingo syntax
vector1 + vector2
vector + scalar
```

Beschreibung

Dieser 3D-Vektoroperator addiert die Komponenten von zwei Vektoren bzw. den Skalarwert zu jeder Vektorkomponente und gibt einen neuen Vektor zurück.

vector1 + *vector2* addiert die Komponenten von *vector1* zu den entsprechenden Komponenten von *vector2* und gibt einen neuen Vektor zurück.

vector + *scalar* addiert den Skalarwert zu jeder Vektorkomponente und gibt einen neuen Vektor zurück.

- (Minus) (3D)

Syntax

```
-- Lingo syntax
vector1 - vector2
vector - scalar
```

Beschreibung

Dieser 3D-Vektoroperator subtrahiert die Komponenten von *vector2* von den entsprechenden Komponenten von *vector1* bzw. den Skalarwert von jeder Komponente und gibt einen neuen Vektor zurück.

vector1 - *vector2* subtrahiert die Werte von *vector2* von den entsprechenden Komponenten in *vector1* und gibt einen neuen Vektor zurück.

Vektor - *Skalar* subtrahiert den Skalarwert von jeder Vektorkomponente und gibt einen neuen Vektor zurück.

* (Multiplikation (3D))

Syntax

```
-- Lingo syntax
vector1 * vector2
vector * scalar
transform * vector
```

Beschreibung

Dieser 3D-Vektoroperator multipliziert die Komponenten von *vector1* mit den entsprechenden Komponenten in *vector2* und gibt das Punktprodukt zurück bzw. multipliziert jede Vektorkomponente mit dem Skalarwert und gibt einen neuen Vektor zurück.

vector1 * *vector2* gibt das Punktprodukt der zwei Vektoren zurück (kein neuer Vektor). Diese Operation ist mit *vector1*.`dotproduct`.*vector2* identisch.

vector * *scalar* multipliziert jede Vektorkomponente mit dem Skalarprodukt und gibt einen neuen Vektor zurück.

transform * *vector* multipliziert die Transformation *transform* mit dem Vektor *vector* und gibt einen neuen Vektor zurück. Der neue Vektor entsteht durch Anwendung der durch die Transformation *transform* definierten Positions- und Drehungsänderungen auf den Vektor *vector*. Hinweis: *vector* * *transform* wird nicht unterstützt.

Siehe auch

`dotProduct()`

/ (Division)

Syntax

```
-- Lingo syntax
expression1 / expression2

// JavaScript syntax
expression1 / expression2
```

Beschreibung

Dieser mathematische Operator führt eine arithmetische Division zweier numerischer Ausdrücke aus, indem er *expression1* durch *expression2* dividiert. Wenn beide Ausdrücke Ganzzahlen sind, ist der Quotient eine Ganzzahl. Wenn einer oder beide Ausdrücke Fließkommazahlen sind, ist der Quotient eine Fließkommazahl.

Dieser arithmetische Operator hat die Prioritätsstufe 4.

Beispiel

Die folgende Anweisung dividiert die Ganzzahl 22 durch 7 und zeigt das Ergebnis im Nachrichtenfenster an:

```
-- Lingo syntax
put(22 / 7)

// JavaScript syntax
put(22 / 7);
```

Das Ergebnis ist 3. Da beide Zahlen in der Division Ganzzahlen sind, rundet Lingo das Ergebnis auf die nächstliegende Ganzzahl.

Die folgende Anweisung dividiert die Fließkommazahl 22.0 durch 7.0 und zeigt das Ergebnis im Nachrichtenfenster an:

```
-- Lingo syntax
put(22.0 / 7.0)

// JavaScript syntax
put(22.0 / 7.0);
```

Das Ergebnis ist die Fließkommazahl 3.1429.

/ (Division) (3D)

Syntax

```
-- Lingo syntax
vector / scalar
```

Beschreibung

Dieser 3D-Vektoroperator dividiert jede Vektorkomponente durch den Skalarwert und gibt einen neuen Vektor zurück.

< (kleiner als)

Syntax

```
-- Lingo syntax
expression1 < expression2

// JavaScript syntax
expression1 < expression2
```

Beschreibung

Dieser Vergleichsoperator vergleicht zwei Ausdrücke und bestimmt, ob *expression1* kleiner als *expression2* ist (`TRUE`) bzw. ob *expression1* größer oder gleich *expression2* (`FALSE`) ist.

Dieser Operator kann Strings, Ganzzahlen, Fließkommazahlen, Rechtecke und Punkte miteinander vergleichen. Beachten Sie, dass Rechtecke und Punkte so miteinander verglichen werden, als ob sie Listen seien. Dabei wird jedes Element der ersten Liste mit dem entsprechenden Element der zweiten Liste verglichen.

Dieser Vergleichsoperator hat die Prioritätsstufe 1.

<= (kleiner als oder gleich)

Syntax

```
-- Lingo syntax
expression1 <= expression2

// JavaScript syntax
expression1 <= expression2
```

Beschreibung

Dieser Vergleichsoperator vergleicht zwei Ausdrücke und bestimmt, ob *expression1* kleiner oder gleich *expression2* ist (`TRUE`) bzw. ob *expression1* größer als *expression2* (`FALSE`) ist.

Dieser Operator kann Strings, Ganzzahlen, Fließkommazahlen, Rechtecke und Punkte miteinander vergleichen. Beachten Sie, dass Rechtecke und Punkte so miteinander verglichen werden, als ob sie Listen seien. Dabei wird jedes Element der ersten Liste mit dem entsprechenden Element der zweiten Liste verglichen.

Dieser Vergleichsoperator hat die Prioritätsstufe 1.

<> (nicht gleich)

Syntax

```
-- Lingo syntax
expression1 <> expression2

// JavaScript syntax
expression1 != expression2
```

Beschreibung
Dieser Vergleichsoperator vergleicht zwei Ausdrücke, Symbole oder Operatoren und bestimmt, ob *expression1* ungleich *expression2* ist (`TRUE`) bzw. ob *expression1* gleich *expression2* (`FALSE`) ist.

Dieser Operator kann Strings, Ganzzahlen, Fließkommazahlen, Rechtecke und Punkte miteinander vergleichen. Beachten Sie, dass Rechtecke und Punkte so miteinander verglichen werden, als ob sie Listen seien. Dabei wird jedes Element der ersten Liste mit dem entsprechenden Element der zweiten Liste verglichen.

Dieser Vergleichsoperator hat die Prioritätsstufe 1.

= (ist gleich)

Syntax

```
-- Lingo syntax
expression1 = expression2

// JavaScript syntax
expression1 = expression2
```

Beschreibung
Dieser Vergleichsoperator vergleicht zwei Ausdrücke, Symbole oder Objekte und bestimmt, ob *expression1* gleich *expression2* ist (`TRUE`) bzw. ob *expression1* ungleich *expression2* (`FALSE`) ist.

Dieser Operator kann Strings, Ganzzahlen, Fließkommazahlen, Rechtecke, Listen und Punkte miteinander vergleichen.

Listen werden anhand der Anzahl der in ihnen enthaltenen Elemente verglichen. Eine Liste mit mehr Elementen ist größer als eine Liste mit weniger Elementen.

Dieser Vergleichsoperator hat die Prioritätsstufe 1.

> (größer als)

Syntax

```
-- Lingo syntax
expression1 > expression2

// JavaScript syntax
expression1 > expression2
```

Beschreibung
Dieser Vergleichsoperator vergleicht zwei Ausdrücke und bestimmt, ob *expression1* größer als *expression2* ist (`TRUE`) bzw. ob *expression1* kleiner oder gleich *expression2* (`FALSE`) ist.

Dieser Operator kann Strings, Ganzzahlen, Fließkommazahlen, Rechtecke und Punkte miteinander vergleichen. Beachten Sie, dass Rechtecke und Punkte so miteinander verglichen werden, als ob sie Listen seien. Dabei wird jedes Element der ersten Liste mit dem entsprechenden Element der zweiten Liste verglichen.

Dieser Vergleichsoperator hat die Prioritätsstufe 1.

>= (größer als oder gleich)

Syntax

```
-- Lingo syntax
expression1 >= expression2

// JavaScript syntax
expression1 >= expression2
```

Beschreibung

Dieser Vergleichsoperator vergleicht zwei Ausdrücke und bestimmt, ob *expression1* größer oder gleich *expression2* ist (`TRUE`) bzw. ob *expression1* kleiner als *expression2* (`FALSE`) ist.

Dieser Operator kann Strings, Ganzzahlen, Fließkommazahlen, Rechtecke und Punkte miteinander vergleichen. Beachten Sie, dass Rechtecke und Punkte so miteinander verglichen werden, als ob sie Listen seien. Dabei wird jedes Element der einen Liste mit dem entsprechenden Element der zweiten Liste verglichen.

Dieser Vergleichsoperator hat die Prioritätsstufe 1.

[ ] (Klammernzugriff)

Syntax

```
-- Lingo syntax
textExpression[chunkNumberBeingAddressed]
textExpression[firstChunk..lastChunk]
```

Beschreibung

Mit diesem Operator kann ein Chunk-Ausdruck mit einem numerischen Wert angesprochen werden. Dies ist nützlich, um den *n*-ten Chunk in einem Ausdruck zu finden. Der Chunk kann ein Wort, eine Zeile, ein Zeichen, ein Absatz oder ein anderer Testdarsteller-Chunk sein.

Beispiel

Die folgende Anweisung gibt das erste Wort der dritten Zeile im Textdarsteller "Vornamen" aus:

```
-- Lingo syntax
put(member("First Names").text.line[3].word[1])

// JavaScript syntax
put(member("First Names").getPropRef("line", 1).getProp("word", 1));
```

[ ] (list)

Syntax

```
[entry1, entry2, entry3, ...]
```

Beschreibung

Dieser Listenoperator gibt an, dass die Einträge innerhalb der eckigen Klammern zu den folgenden vier Listentypen gehören:

- nicht geordnete lineare Listen
- geordnete lineare Listen
- nicht geordnete Eigenschaftslisten
- geordnete Eigenschaftslisten

Jeder Eintrag in einer linearen Liste ist ein einzelner Wert, der nicht mit einer Eigenschaft verbunden ist. Jeder Eintrag in einer Eigenschaftsliste besteht aus einer Eigenschaft und einem Wert. Die Eigenschaft steht vor dem Wert und ist durch einen Doppelpunkt von ihm getrennt. Sie können eine Eigenschaft nicht in einer linearen Liste speichern. Wenn Sie Strings als Einträge in einer Liste verwenden, müssen Sie den String zwischen Anführungszeichen setzen.

[6, 3, 8] ist zum Beispiel eine lineare Liste. Die Zahlen sind nicht mit Eigenschaften verbunden. Die Liste [`#gears:6`, `#balls:3`,`#ramps:8`] ist allerdings eine Eigenschaftsliste. Jede Zahl ist mit einer Eigenschaft – in diesem Fall mit einem Maschinenteil – verbunden. Diese Eigenschaftsliste ist hilfreich, wenn Sie feststellen möchten, wie viele Maschinenteile jedes Typs in einer mechanischen Simulation gegenwärtig auf der Bühne sind. Eigenschaften können in einer Eigenschaftsliste mehrfach aufgeführt sein.

Listen können in alphanumerischer Reihenfolge geordnet werden. Eine geordnete lineare Liste ist nach den Werten in der Liste geordnet. Eine geordnete Eigenschaftsliste ist nach den Eigenschaften in der Liste geordnet. Zum Ordnen einer Liste verwenden Sie den entsprechenden Befehl für eine lineare Liste oder Eigenschaftsliste.

- In linearen Listen wird bei Symbolen und Strings zwischen Groß- und Kleinschreibung unterschieden.
- In Eigenschaftslisten wird nur bei Strings – nicht aber bei Symbolen – zwischen Groß- und Kleinschreibung unterschieden.

Eine lineare Liste oder Eigenschaftsliste kann überhaupt keine Werte enthalten. Eine leere Liste besteht aus zwei eckigen Klammern ([ ]). Um eine lineare Liste zu erstellen oder zu löschen, stellen Sie die Liste auf `[ ]` ein. Um eine Eigenschaftsliste zu erstellen oder zu löschen, stellen Sie die Liste auf [:] ein.

Sie können Einträge in einer Liste modifizieren, testen oder lesen.

Lingo behandelt eine Instanz einer Liste als Verweis auf die Liste. Dies bedeutet, dass es sich bei jeder Instanz um die gleichen Daten handelt. Wenn sie geändert werden, ändert sich auch das Original. Verwenden Sie den Befehl `duplicate`, um Kopien der Liste zu erstellen.

Listen werden automatisch entfernt, wenn sich keine Variable mehr auf sie bezieht. Wenn eine Liste in einer globalen Variable enthalten ist, bleibt sie von Film zu Film erhalten.

Sie können eine Liste in der Prozedur `on prepareMovie` initialisieren oder die Liste als Felddarsteller erstellen, der Liste einer Variablen zuordnen und dann mit der Liste arbeiten, indem Sie die Variable verwenden.

Nicht alle PC-Tastaturen haben eckige Klammern. Wenn eckige Klammern nicht verfügbar sind, verwenden Sie die Funktion `list`, um eine lineare Liste zu erstellen.

Erstellen Sie für eine Eigenschaftsliste die einzelnen Teile der Liste als String, bevor Sie diesen in eine brauchbare Liste konvertieren.

```
myListString = numToChar(91) & ":" & numToChar(93)
put myListString
-- "[:]"
myList = myListString.value
put myList
-- [:]
put myList.listP
-- 1
myList[#name] = "Brynn"
put myList
-- [#name: "Brynn"]
```

Beispiel

Die folgende Anweisung definiert eine Liste, indem sie die Variable `machinery` mit der Liste gleichsetzt:

```
-- Lingo syntax
machinery = [#gears:6, #balls:3, #ramps:8]

// JavaScript syntax
var machinery = propList("gears",6, "balls",3, "ramps",8);
```

Die folgende Prozedur ordnet die Liste „aList“ und zeigt das Resultat im Nachrichtenfenster an:

```
-- Lingo syntax
on sortList aList
    alist.sort()
    put(aList)
end sortList

// JavaScript syntax
function sortList(aList) {
    aList.sort();
    put(aList);
}
```

Wenn der Film die Anweisung `sortList machinery` ausgibt, in der `machinery`der Liste im vorhergehenden Beispiel entspricht, ist das Ergebnis [`#balls:3`, `#gears:6`, `#ramps:8`].

Die folgenden Anweisungen erstellen eine leere lineare Liste:

```
-- Lingo syntax
x = [ ]
x = list()

// JavaScript syntax
var x = list();
```

Die folgenden Anweisungen erstellen eine leere Eigenschaftsliste:

```
-- Lingo syntax
x = [:]
x = propList()

// JavaScript syntax
var x = propList();
```

Siehe auch

add, addVertex(), append, count(), deleteAt, duplicate() (Listenfunktion), findPos, findPosNear, getProp(), getAt, getLast(), getPos(), ilk(), list(), max(), min, setAt, setaProp, sort

@ (Pfadname)

Syntax

```
@pathReference
```

Beschreibung

Dieser Pfadnamenoperator definiert den Pfad zum Ordner des aktuellen Films und ist unter Windows® wie auch auf dem Macintosh® gültig.

Kennzeichnen Sie den Ordner des aktuellen Film mit dem Symbol @, gefolgt von einem der folgenden Pfadnamentrennzeichen:

- / (Schrägstrich)
- \ (umgekehrter Schrägstrich)
- : (Doppelpunkt)

Wenn ein Film nach seiner Position abgefragt wird, enthält der zurückgegebene String das Symbol @.

Achten Sie unbedingt darauf, dass Sie das Symbol @ nur dann verwenden, wenn Sie zwischen Director-Filmen navigieren oder die Quelle eines verknüpften Mediendarstellers ändern. Das @-Symbol funktioniert nicht, wenn die FileIOXtra-Erweiterung oder andere Funktionen, die nicht innerhalb von Director verfügbar sind, verwendet werden.

Sie können auf diesem Pfadnamen aufbauen, um Ordner festzulegen, die sich eine oder mehrere Ebenen über oder unter dem Ordner des aktuellen Films befinden. Denken Sie daran, dass der @-Abschnitt die Position des aktuellen Films und nicht unbedingt die Position des Projektors darstellt.

- Fügen Sie direkt nach dem @-Symbol ein weiteres Pfadnamentrennzeichen hinzu, um einen Ordner auf der nächsthöheren Ebene in der Hierarchie anzugeben.
- Zum Festlegen von Unterordnern und Dateien innerhalb von Ordnern können Sie Ordner und Dateinamen (durch /, \ oder : getrennt) hinzufügen.

Verwenden Sie relative Pfadnamen in Lingo, um die Position einer verknüpften Datei anzugeben, die sich in einem anderen Ordner als der Film befindet.

Beispiel

Die folgenden gleichbedeutenden Ausdrücke bezeichnen den Unterordner bigFolder, der sich im Ordner des aktuellen Films befindet:

```
@/bigFolder
@:bigFolder
@\bigFolder
```

Die folgenden gleichbedeutenden Ausdrücke bezeichnen die Datei linkedFile im Unterordner bigFolder, der sich im Ordner des aktuellen Films befindet:

```
@:bigFolder:linkedFile
@\bigFolder\linkedFile
@/bigFolder/linkedFile
```

Der folgende Ausdruck bezeichnet die Datei `linkedFile`, die sich eine Ebene über dem Ordner des aktuellen Films befindet:

```
@//linkedFile
```

Der folgende Ausdruck bezeichnet die Datei `linkedFile`, die sich zwei Ebenen über dem Ordner des aktuellen Films befindet:

```
@:::linkedFile
```

Die folgenden gleichbedeutenden Ausdrücke bezeichnen die Datei `linkedFile`, die sich im Ordner `otherFolder` befindet. Der Ordner `otherFolder` befindet sich im Ordner eine Ebene über dem Ordner des aktuellen Films.

```
@::otherFolder:linkedFile
@\\otherFolder\linkedFile
@//otherFolder/linkedFile
```

Siehe auch

`searchPathList`, `fileName (Besetzung)`, `fileName (Darsteller)`, `fileName (Fenster)`

and

Syntax

```
-- Lingo syntax
logicalExpression1 and logicalExpression2

// JavaScript syntax
logicalExpression1 && logicalExpression2
```

Beschreibung

Dieser logische Operator bestimmt, ob *logicalExpression1* und *logicalExpression2* beide `TRUE` (1) oder ob ein oder beide Ausdrücke `FALSE` (0) sind.

Dieser logische Operator hat die Prioritätsstufe 4.

Beispiel

Die folgende Anweisung bestimmt, ob beide logischen Ausdrücke `TRUE` sind, und zeigt das Ergebnis im Nachrichtenfenster an:

```
-- Lingo syntax
put(1 < 2 and 2 < 3)

// JavaScript syntax
put((1 < 2) && (2 < 3));
```

Das Resultat ist 1, das numerische Gegenstück zu `TRUE`.

Der erste logische Ausdruck in der folgenden Anweisung ist `TRUE`, und der zweite logische Ausdruck ist `FALSE`. Da beide logischen Ausdrücke nicht `TRUE` sind, zeigt der logische Operator das Ergebnis 0 an, was das numerische Gegenstück zu `FALSE` ist.

```
-- Lingo syntax
put(1 < 2 and 2 < 1)
-- 0

// JavaScript syntax
put((1 < 2) && (2 < 1));
// 0
```

Siehe auch

not, oder

contains

Syntax

```
-- Lingo syntax
stringExpression1 contains stringExpression2

// JavaScript syntax
stringExpression1.indexOf(stringExpression2);
```

Beschreibung

Dieser Operator vergleicht zwei Strings und prüft, ob *stringExpression1stringExpression2* enthält (`TRUE`) oder nicht (`FALSE`).

Der Vergleichsoperator `contains` hat die Prioritätsstufe 1.

Der Vergleichsoperator `contains` ist nützlich, wenn Sie überprüfen möchten, ob der Benutzer bestimmte Zeichen oder Strings eingibt. Sie können mit dem `contains`-Operator auch ein oder mehrere Felder nach bestimmten Strings durchsuchen.

Beispiel

Dieses Beispiel stellt fest, ob das übergebene Zeichen eine Zahl ist:

```
-- Lingo syntax
on isNumber aLetter
    digits = "1234567890"
    if digits contains aLetter then
        return TRUE
    else
        return FALSE
    end if
end

// JavaScript syntax
function isNumber(aLetter) {
    var digits = "1234567890"
    if (digits.indexOf(aLetter) >= 0) {
        return true;
    } else {
        return false;
    }
}
```

Hinweis: *Beim String-Vergleich spielen Groß- und Kleinschreibung sowie diakritische Zeichen keine Rolle; „a“ und „Ä“ werden beispielsweise gleich behandelt.*

Siehe auch

`offset() (Zeichenfolgenfunktion)`, `starts`

mod

Syntax

```
-- Lingo syntax
integerExpression1 mod integerExpression2

// JavaScript syntax
integerExpression1 % integerExpression2
```

Beschreibung

Dieser mathematische Operator führt die arithmetische Modulo-Operation mit zwei ganzzahligen Ausdrücken durch. Bei dieser Operation wird *integerExpression1* durch *integerExpression2* dividiert.

Das Ergebnis des gesamten Ausdrucks ist der ganzzahlige Rest der Division. Er hat immer das Vorzeichen von *integerExpression1*.

Dieser arithmetische Operator hat die Prioritätsstufe 4.

Beispiel

Die folgende Anweisung dividiert 7 durch 4 und zeigt den Restwert im Nachrichtenfensteran:

```
-- Lingo syntax
put(7 mod 4)

// JavaScript syntax
put(7 % 4);
```

Das Ergebnis ist 3.

Die folgende Prozedur stellt den Farbeffekt aller Sprites mit einer ungeraden Zahl auf `copy` ein, welches der durch die Ziffer 0 angegebene Farbeffekt ist. Die Prozedur überprüft zuerst, ob das Sprite in der Variablen `mySprite` ein ungerades Sprite ist, indem es die Sprite-Nummer durch 2 teilt und prüft, ob als Rest 1 bleibt. Ist dies der Fall, erkennt die Prozedur, dass es sich um eine ungerade Zahl handelt, und setzt den Farbeffekt auf `copy`.

```
-- Lingo syntax
on setInk
    repeat with mySprite = 1 to _movie.lastChannel
        if (mySprite mod 2) = 1 then
            sprite(mySprite).ink = 0
        else
            sprite(mySprite).ink = 8
        end if
    end repeat
end setInk

// JavaScript syntax
function setInk() {
    for (mySprite=1; mySprite<=_movie.lastChannel; mySprite++) {
        if ((mySprite % 2) == 1) {
            sprite(mySprite).ink = 0;
        } else {
            sprite(mySprite).ink = 8;
        }
    }
}
```

Die folgende Prozedur bewirkt, dass Darsteller eines Sprites regelmäßig eine Reihe von Bitmaps durchläuft:

```
-- Lingo syntax
on exitFrame
    global gCounter
    -- These are sample values for bitmap cast member numbers
    theBitmaps = [2,3,4,5,6,7]
    -- Specify which sprite channel is affected
    theChannel = 1
    -- This cycles through the list
    gCounter = 1 + (gCounter mod theBitmaps.count)
    sprite(theChannel).memberNum = theBitmaps[gCounter]
    _movie.go(_movie.frame)
end

// JavaScript syntax
function exitFrame() {
    // these are sample values for bitmap cast member numbers
    theBitmaps = new Array(2,3,4,5,6,7);
    // specify which sprite channel is affected
    theChannel = 1;
    // this cycles through the list
    _global.gCounter = 1 + (_global.gCounter % theBitmaps.length);
    sprite(theChannel).memberNum = theBitmaps[_global.gCounter];
    _movie.go(_movie.frame);
}
```

not

Syntax

```
-- Lingo syntax
not logicalExpression

// JavaScript syntax
! logicalExpression
```

Beschreibung

Dieser Operator führt eine logische Negation eines logischen Ausdrucks durch. Dies entspricht der Änderung eines `TRUE`-Wertes in `FALSE` bzw. eines `FALSE`-Wertes in `TRUE`. Hierdurch lässt sich feststellen, ob eine bestimmte bekannte Bedingung unzutreffend ist.

Dieser logische Operator hat die Prioritätsstufe 5.

Beispiel

Die folgende Anweisung stellt fest, ob 1 nicht kleiner als 2 ist:

```
-- Lingo syntax
put(not (1 < 2))

// JavaScript syntax
put(!(1 < 2));
```

Da 1 kleiner als 2 ist, ist das Resultat 0, was bedeutet, dass der Ausdruck `FALSE` ist.

Die folgende Anweisung stellt fest, ob 1 nicht größer als 2 ist:

```
-- Lingo syntax
put(not (1 > 2))

// JavaScript syntax
put(!(1 > 2));
```

Da 1 nicht größer als 2 ist, ist das Resultat 1, was bedeutet, dass der Ausdruck `TRUE` ist.

Die folgende Prozedur kehrt die Menüelementeigenschaft `the checkMark` für „Fett“ im Menü „Stil“ in das Gegenteil ihrer aktuellen Einstellung um:

```
-- Lingo syntax
on resetMenuItem
    menu("Style").menuItem("Bold").checkMark =
    not (menu("Style").menuItem("Bold").checkMark)
end resetMenuItem

// JavaScript syntax
function resetMenuItem() {
    menu("Style").menuItem("Bold").checkMark =
    !(menu("Style").menuItem("Bold").checkMark)
}
```

Siehe auch

and, oder

oder

Syntax

```
-- Lingo syntax
logicalExpression1 or logicalExpression2

// JavaScript syntax
logicalExpression1 || logicalExpression2
```

Beschreibung

Dieser Operator führt für zwei oder mehr logische Ausdrücke eine logische OR-Operation aus, um festzustellen, ob einer dieser Ausdrücke TRUE ist.

Dieser logische Operator hat die Prioritätsstufe 4.

Beispiel

Die folgende Anweisung zeigt im Nachrichtenfenster an, ob zumindest einer der Ausdrücke „1 < 2“ und „1 > 2“ TRUE ist:

```
-- Lingo syntax
put((1 < 2) or (1 > 2))

// JavaScript syntax
put((1 < 2) || (1 > 2));
```

Da der erste Ausdruck TRUE ist, erhalten Sie das Ergebnis 1, was dem numerischen Wert für TRUE entspricht.

Die folgende Anweisung prüft, ob der Inhalt des Felddarstellers "Bundesstaat" entweder AK oder HI ist, und zeigt in diesem Fall eine Hinweismeldung an:

```
-- Lingo syntax
if member("State").text = "AK" or member("State").text = "HI" then
    _player.alert("You're off the map!")
end if

// JavaScript syntax
if (member("State").text == "AK" || member("State").text == "HI") {
    _player.alert("You're off the map!");
}
```

Siehe auch

and, not

Operator für wahlfreien Zugriff.

Syntax

```
byteArrayObject[index]
```

Beschreibung

Dieser Bytearray-Operator liefert den Wert eines nicht signierten Bytes des Arrays oder schreibt einen Wert in das Array. Bei Werten über 255 wird nur das niederwertige Byte berücksichtigt.

Beispiele

```
--Lingo syntax
bArray = byteArray("Sample ByteArray")
put bArray[1]

//JavaScript syntax
bArray = byteArray("Sample ByteArray");
put(bArray[1]);
```

starts

Syntax

```
-- Lingo syntax
string1 starts string2

// JavaScript syntax
string1.indexOf(string2) == 0;
```

Beschreibung

Dieser Vergleichsoperator bestimmt, ob *string1* mit *string2* beginnt (`TRUE` oder 1) oder nicht (`FALSE` oder 0).

Beim Stringvergleich wird nicht zwischen Groß- und Kleinschreibung oder diakritischen Zeichen unterschieden: Zum Beispiel sind *a* und *Å* gleichbedeutend.

Dieser Vergleichsoperator hat die Prioritätsstufe 1.

Beispiel

Die folgende Anweisung meldet im Nachrichtenfenster, ob das Wort *Macrostuff* mit dem String „Macro“ beginnt:

```
-- Lingo syntax
put("Macrostuff" starts "Macro")

// JavaScript syntax
var string1 = "Macrostuff";
put(string1.indexOf("Macro") == 0);
```

Das Resultat ist 1, das numerische Gegenstück zu `TRUE`.

Siehe auch

contains

Zeichenfolge

Syntax

```
String(byteArrayObject)
```

Beschreibung

Dieser Bytearray-Operator konvertiert das Bytearray in ein lesbares Hexadezimalformat. Diese Methode empfiehlt sich für das Schreiben des Bytearrays in ein Nachrichtenfenster, einen Objektinspektor oder in einen Debugger.

Beispiele

```
--Lingo syntax
bArray = byteArray("Sample ByteArray")
put String(bArray)

//JavaScript syntax
bArray = byteArray("Sample ByteArray");
put(String(bArray));
```

Kapitel 14: Eigenschaften

In diesem Abschnitt finden Sie eine alphabetische Liste aller in Director® verfügbaren Eigenschaften.

_global

Syntax

```
-- Lingo syntax
_global

// JavaScript syntax
_global;
```

Beschreibung

Diese Top-Level-Eigenschaft stellt einen Verweis auf das Global-Objekt bereit, in dem alle globalen Variablen gespeichert werden. Nur Lesen.

Auf alle globalen Variablen kann sowohl mit Lingo als auch mit JavaScript-Syntax zugegriffen werden.

Beispiel

Die folgende Anweisung setzt die Variable `objGlobal` auf die Eigenschaft `_global` fest:

```
-- Lingo syntax
objGlobal = _global

// JavaScript syntax
var objGlobal = _global;
```

Die folgende Anweisung verwendet die Eigenschaft `_global` direkt zum Löschen aller globalen Variablen:

```
-- Lingo syntax
_global.clearGlobals()

// JavaScript syntax
_global.clearGlobals();
```

Siehe auch

`Global`

_key

Syntax

```
-- Lingo syntax
_key

// JavaScript syntax
_key;
```

Beschreibung
Diese Top-Level-Eigenschaft stellt einen Verweis auf das Key-Objekt bereit, mit dessen Hilfe die Tastaturaktivitäten von Benutzern überwacht werden. Nur Lesen.

Beispiel
Die folgende Anweisung setzt die Variable `objKey` auf die Eigenschaft `_key` fest:

```
-- Lingo syntax
objKey = _key

// JavaScript syntax
var objKey = _key;
```

Die folgende Anweisung verwendet die Eigenschaft `_key` direkt für den Zugriff auf den Wert der Eigenschaft `key`:

```
-- Lingo syntax
theKey = _key.key

// JavaScript syntax
var theKey = _key.key;
```

Siehe auch
Taste

_mouse

Syntax

```
-- Lingo syntax
_mouse

// JavaScript syntax
_mouse;
```

Beschreibung
Diese Top-Level-Eigenschaft stellt einen Verweis auf das Mouse-Objekt bereit, das den Zugriff auf die Mausaktivitäten von Benutzern bereitstellt, einschließlich Mausbewegungen und Drücken von Maustasten. Nur Lesen.

Beispiel
Die folgende Anweisung setzt die Variable `objMouse` auf die Eigenschaft `_mouse` fest:

```
-- Lingo syntax
objMouse = _mouse

// JavaScript syntax
var objMouse = _mouse;
```

Die folgende Anweisung verwendet die Eigenschaft `_mouse` direkt für den Zugriff auf den Wert der Eigenschaft `mouseH`:

```
-- Lingo syntax
theMouseH = _mouse.mouseH

// JavaScript syntax
var theMouseH = _mouse.mouseH;
```

Siehe auch

Mouse

_movie

Syntax

```
-- Lingo syntax
_movie

// JavaScript syntax
_movie;
```

Beschreibung

Diese Top-Level-Eigenschaft stellt einen Verweis auf das Movie-Objekt bereit, das den aktuell aktiven Film im Director-Player darstellt, und bietet Zugriff auf Eigenschaften und Methoden, die auf Filmebene verfügbar sind. Nur Lesen.

Beispiel

Die folgende Anweisung setzt die Variable `objMovie` auf die Eigenschaft `_movie` fest:

```
-- Lingo syntax
objMovie = _movie

// JavaScript syntax
var objMovie = _movie;
```

Die folgende Anweisung verwendet die Eigenschaft `_movie` direkt für den Zugriff auf den Wert der Eigenschaft `displayTemplate`:

```
-- Lingo syntax
theTemplate = _movie.displayTemplate

// JavaScript syntax
var theTemplate = _movie.displayTemplate;
```

Siehe auch

Movie

_player

Syntax

```
-- Lingo syntax
_player

// JavaScript syntax
_player;
```

Beschreibung

Diese Top-Level-Eigenschaft stellt einen Verweis auf das Player-Objekt bereit, das alle Filme verwaltet und ausführt, einschließlich Filmen in einem Fenster (MIAWs). Nur Lesen.

Beispiel

Die folgende Anweisung setzt die Variable `objPlayer` auf die Eigenschaft `_player` fest:

```
-- Lingo syntax
objPlayer = _player

// JavaScript syntax
var objPlayer = _player;
```

Die folgende Anweisung verwendet die Eigenschaft `_player` direkt für den Zugriff auf den Wert der Eigenschaft `xtraList`:

```
-- Lingo syntax
theXtras = _player.xtraList

// JavaScript syntax
var theXtras = _player.xtraList;
```

Siehe auch

Player

_sound

Syntax

```
-- Lingo syntax
_sound

// JavaScript syntax
_sound;
```

Beschreibung

Diese Top-Level-Eigenschaft stellt einen Verweis auf das Sound-Objekt bereit, das die Audiowiedergabe in allen acht verfügbaren Soundkanälen steuert. Nur Lesen.

Beispiel

Die folgende Anweisung setzt die Variable `objSound` auf die Eigenschaft `_sound` fest:

```
-- Lingo syntax
objSound = _sound

// JavaScript syntax
var objSound = _sound;
```

Die folgende Anweisung verwendet die Eigenschaft `_sound` direkt für den Zugriff auf die Eigenschaft `soundLevel`:

```
-- Lingo syntax
theLevel = _sound.soundLevel

// JavaScript syntax
var theLevel = _sound.soundLevel;
```

Siehe auch

Sound

_system

Syntax

```
-- Lingo syntax
_system

// JavaScript syntax
_system;
```

Beschreibung

Diese Top-Level-Eigenschaft stellt einen Verweis auf das System-Objekt bereit, das den Zugriff auf System- und Umgebungsinformationen bereitstellt, einschließlich Methoden auf Systemebene. Nur Lesen.

Beispiel

Die folgende Anweisung setzt die Variable `objSystem` auf die Eigenschaft `_system` fest:

```
-- Lingo syntax
objSystem = _system

// JavaScript syntax
var objSystem = _system;
```

Diese Anweisung verwendet direkt die Eigenschaft `_system`, um auf die Eigenschaft `freeBytes` zuzugreifen:

```
-- Lingo syntax
theBytes = _system.freeBytes

// JavaScript syntax
var theBytes = _system.freeBytes;
```

Siehe auch

System

aboutInfo

Syntax

```
-- Lingo syntax
_movie.aboutInfo

// JavaScript syntax
_movie.aboutInfo;
```

Beschreibung

Diese Filmeigenschaft ist ein String, der bei der Filmerstellung im Dialogfeld „Filmeigenschaften“ eingegeben wird. Nur Lesen.

Beispiel

Die folgende Anweisung zeigt im Nachrichtenfenster Filminformationen an.

```
-- Lingo syntax
trace(_movie.aboutInfo)

// JavaScript syntax
trace(_movie.aboutInfo);
```

Siehe auch

copyrightInfo (Film), Movie

actionsEnabled

Syntax

```
-- Lingo syntax
memberOrSpriteObjRef.actionsEnabled

// JavaScript syntax
memberOrSpriteObjRef.actionsEnabled;
```

Beschreibung

Diese Darsteller- und Sprite-Eigenschaft legt fest, ob die Aktionen in Adobe® Flash®-Inhalt aktiviert (`TRUE`, Standard) oder deaktiviert (`FALSE`) sind.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Die folgende Prozedur akzeptiert einen Sprite-Bezug als Parameter und schaltet dann die Eigenschaft `actionsEnabled` des Sprites ein oder aus.

```
-- Lingo syntax
on ToggleActions(whichSprite)
     sprite(whichSprite).actionsEnabled = not(sprite(whichSprite).actionsEnabled)
end

// JavaScript syntax
function ToggleActions(whichSprite) {
     switch(sprite(whichSprite).actionsEnabled){
        case 0:
             sprite(whichSprite).actionsEnabled = 1;
            break;
        case 1:
            sprite(whichSprite).actionsEnabled = 0;
            break;
    }
}
```

active3dRenderer

Syntax

```
-- Lingo syntax
_movie.active3dRenderer

// JavaScript syntax
_movie.active3dRenderer;
```

Beschreibung

Diese Filmeigenschaft gibt den Renderer an, der gegenwärtig vom Film zum Zeichnen von 3D-Sprites verwendet wird. Diese Eigenschaft entspricht der Eigenschaft `getRendererServices().renderer`. Nur Lesen.

Mögliche Werte für die Eigenschaft `active3DRenderer`: `#openGL`, `#directX7_0`, `#directx9`, `#directX5_2` und `#software`. Die Werte `#openGL`, `#directX7_0`, `#directx9`, und `#directX5_2`, sind Hardware-Renderer und führen zu einer erheblich besseren Leistung als der Wert `#software` – ein Software-Renderer, der verwendet wird, wenn keine der ersten drei Optionen verfügbar ist.

Verwenden Sie `getRendererServices().renderer` zum Einstellen dieser Eigenschaft.

Beispiel

In den folgenden Beispiele lernen Sie zwei Möglichkeiten kennen, um herauszufinden, welcher Renderer gegenwärtig verwendet wird.

```
-- Lingo syntax
put(_movie.active3dRenderer)
put(getRendererServices().renderer)

// JavaScript syntax
put(_movie.active3dRenderer);
put(getRendererServices().renderer);
```

Siehe auch

Movie, renderer

activeCastLib

Syntax

```
-- Lingo syntax
_movie.activeCastLib

// JavaScript syntax
_movie.activeCastLib;
```

Beschreibung

Diese Filmeigenschaft zeigt an, welche Besetzungsbibliothek zuletzt aktiviert wurde. Nur Lesen.

Der Wert der Eigenschaft `activeCastLib` ist die Nummer der Besetzungsbibliothek.

Die Eigenschaft `activeCastLib` ist hilfreich, wenn Sie mit der Eigenschaft `selection` des Cast-Objekts arbeiten. Anhand dieser Eigenschaft können Sie feststellen, auf welche Besetzungsbibliothek die Auswahl verweist.

Beispiel

Die folgenden Anweisungen ordnen die ausgewählten Darsteller in der zuletzt ausgewählten Besetzung der Variablen `selectedMembers` zu:

```
-- Lingo syntax
castLibOfInterest = _movie.activeCastLib
selectedMembers = castLib(castLibOfInterest).selection

// JavaScript syntax
var castLibOfInterest = _movie.activeCastLib;
var selectedMembers = castLib(castLibOfInterest).selection;
```

Siehe auch

`Player`, `selection`

activeWindow

Syntax

```
-- Lingo syntax
_player.activeWindow

// JavaScript syntax
_player.activeWindow;
```

Beschreibung

Diese Player-Eigenschaft gibt an, welches Filmfenster gegenwärtig aktiv ist. Nur Lesen.

Beim Hauptfilm ist `activeWindow` die Bühne. Bei einem Film in einem Fenster (MIAW) ist `activeWindow` das Fenster, in dem der Film abgespielt wird.

Beispiel

In diesem Beispiel wird das Wort "Aktiv" in die Titelleiste des angeklickten Fensters eingefügt und das Wort "Inaktiv" in die Titelleiste aller anderen offenen Fenster gesetzt:

```
-- Lingo syntax
on activateWindow
    clickedWindow = _player.windowList.getPos(_player.activeWindow)
    windowCount = _player.windowList.count
    repeat with x = 1 to windowCount
        if (x = clickedWindow) then
            _player.window[clickedWindow].title = "Active"
        else
            _player.windowList[x].title = "Inactive"
        end if
    end repeat
end activateWindow

// JavaScript syntax
function activateWindow() {
    var clickedWindow = _player.windowList.getPos(_player.activeWindow);
    var windowCount = _player.windowList.count;
    for (var x = 1; x <= windowCount; x++) {
        if (x == clickedWindow) {
            _player.window[clickedWindow].title = "Active"
        }
        else {
            _player.windowList[x].title = "Inactive"
        }
    }
}
```

Siehe auch

Player

actorList

Syntax

```
-- Lingo syntax
_movie.actorList

// JavaScript syntax
_movie.actorList;
```

Beschreibung

Diese Filmeigenschaft ist eine Liste der Child-Objekte, die ausdrücklich dieser Liste hinzugefügt wurden. Lesen/Schreiben.

Objekte in `actorList` erhalten jedes Mal eine `stepFrame`-Nachricht, wenn der Abspielkopf in ein Bild eintritt.

Mithilfe von `_movie.actorList.append(`*newScriptObjRef*`)` können Sie der `actorList` ein Objekt hinzufügen. Die Prozedur `stepFrame` des Objekts in seinem Parent- oder Ancestor-Skript wird dann jedes Mal automatisch aufgerufen, wenn der Abspielkopf in ein Bild eintritt.

Um Objekte aus `actorList` zu löschen, stellen Sie `actorList` auf `[ ]` und damit auf eine leere Liste ein.

Der Inhalt von `actorList` wird beim Verzweigen zu einem anderen Film nicht gelöscht, was in dem neuen Film zu unvorhersehbaren Verhaltensweisen führen kann. Um zu verhindern, dass Child-Objekte im aktuellen Film in den neuen Film übernommen werden, fügen Sie die Anweisung `actorList = [ ]` in die `prepareMovie`-Prozedur des neuen Films ein.

Beispiel

Die folgende Anweisung fügt ein Child-Objekt hinzu, das von dem Parent-Skript „Ball“ erstellt wurde. Alle drei Werte sind Parameter, die im Skript erforderlich sind.

Die folgende Anweisung zeigt den Inhalt von `actorList` im Nachrichtenfenster an:

```
-- Lingo syntax
put(_movie.actorList)

// JavaScript syntax
put(_movie.actorList);
```

Die folgende Anweisung löscht alle Objekte in `actorList`.

```
-- Lingo syntax
_movie.actorList = [] -- using brackets
_movie.actorList = list() -- using list()

// JavaScript syntax
_movie.actorList = list();
```

Siehe auch

`Movie`, `on prepareMovie`, `on stepFrame`

alertHook

Syntax

```
-- Lingo syntax
_player.alertHook

// JavaScript syntax
_player.alertHook;
```

Beschreibung

Diese Player-Eigenschaft gibt ein Parent-Skript an, das die Prozedur `alertHook` enthält. Lesen/Schreiben.

Verwenden Sie `alertHook`, um die Anzeige von Warnhinweisen über Dateifehler oder Skriptfehler zu steuern. Wenn ein Fehler auftritt und `alertHook` ein Parent-Skript zugeordnet ist, führt Director die Prozedur `alertHook` im Parent-Skript aus.

Obwohl Sie `alertHook`-Prozeduren in Filmskripts stellen können, sollten Sie eine `alertHook`-Prozedur unbedingt in ein Verhalten oder ein Parent-Skript platzieren, um zu verhindern, dass die Prozedur von allen möglichen Stellen her aufgerufen wird und es unklar ist, wo sich der Fehler ereignet hat.

Da die Prozedur `alertHook` bei einem Fehlerzustand ausgeführt wird, sollten Sie die Prozedur `alertHook` nicht für Skripts ausführen, die nicht der Behandlung von Fehlern dienen. Die Prozedur `alertHook` eignet sich zum Beispiel nicht für eine `go()`-Anweisung.

Der Prozedur `alertHook` wird Folgendes übergeben: ein Instanzargument, zwei String-Argumente mit einer Fehlerbeschreibung sowie ein optionales Argument, das ein zusätzliches Ereignis angibt, welches die Prozedur aufruft.

Das vierte Argument kann einen der folgenden vier Werte aufweisen:

- `#alert` bewirkt, dass die Prozedur von der `alert()`-Methode ausgelöst wird.
- `#movie` bewirkt, dass die Prozedur ausgelöst wird, wenn bei Verarbeitung des Befehls `go()` eine Datei nicht gefunden wird.
- `#script` bewirkt, dass die Prozedur durch einen Skriptfehler ausgelöst wird.
- `#safeplayer` bewirkt, dass die Prozedur bei Überprüfung der Eigenschaft `safePlayer` ausgelöst wird.

  Je nach dem Skriptcode, der sich innerhalb der Prozedur `alertHook` befindet, kann diese den Fehler entweder ignorieren oder ihn auf andere Weise melden.

Beispiel

Die folgende Anweisung bezeichnet das Parent-Skript "Warnhinweis" als das Skript, das bestimmt, ob ein Warnhinweis angezeigt werden soll, wenn ein Fehler auftritt. Tritt ein Fehler auf, ordnet das Skript die Fehler- und Nachrichtenstrings dem Felddarsteller „Ausgabe“ zu und gibt den Wert 1 zurück.

```
-- Lingo syntax
on prepareMovie
    _player.alertHook = script("Alert")
end

-- "Alert" script
on alertHook me, err, msg
    member("Output").text = err && msg
    return 1
end

// JavaScript syntax
function prepareMovie() {
     _player.alertHook = "alert("Error type", "Error message");"
}

// alert handler
function alert(err, msg) {
    member("Output").text = err + " " + msg; return 1;
}
```

Siehe auch

Player, safePlayer

alignment

Syntax

```
-- Lingo syntax
memberObjRef.alignment

// JavaScript syntax
memberObjRef.alignment;
```

Beschreibung

Diese Darstellereigenschaft bestimmt die Ausrichtung, mit der Zeichen innerhalb des angegebenen Darstellers angezeigt werden. Diese Eigenschaft kann nur für Feld- und Textdarsteller verwendet werden, die Zeichen enthalten, auch wenn das Zeichen nur ein Leerzeichen ist.

Für Felddarsteller ist der Wert der Eigenschaft ein String, der aus einer der folgenden Angaben besteht: „left“, „center“, „right“.

Für Textdarsteller ist der Wert der Eigenschaft ein Symbol, das aus einem der folgenden Bezeichnungen besteht: `#left`, `#center`, `#right` oder `#full`.

Der Parameter *whichCastMember* kann entweder ein Darstellername oder eine Darstellernummer sein.

Diese Eigenschaft kann getestet und eingestellt werden. Bei Textdarstellern kann die Eigenschaft für jeden Absatz neu festgelegt werden.

Beispiel

Die folgende Anweisung stellt die Variable `characterAlign` auf die aktuelle alignment-Einstellung für den Felddarsteller "Rokujo spricht" ein:

```
--Lingo syntax
characterAlign = member("Rokujo Speaks").alignment

// JavaScript syntax
var characterAlign = member("Rokujo Speaks").alignment;
```

Siehe auch

`text`, `font`, `lineHeight`, `fontSize`, `fontStyle`, `&`, `+ (Verkettungsoperator)`, `&&`, `+ (Verkettungsoperator)`

allowCustomCaching

Syntax

```
-- Lingo syntax
_movie.allowCustomCaching

// JavaScript syntax
_movie.allowCustomCaching;
```

Beschreibung

Diese Filmeigenschaft wird in zukünftigen Director-Versionen Informationen über einen privaten Cache enthalten. Lesen/Schreiben.

Diese Eigenschaft ist standardmäßig auf `TRUE` gesetzt.

Siehe auch

`allowGraphicMenu`, `allowSaveLocal`, `allowTransportControl`, `allowVolumeControl`, `allowZooming`, `Movie`

allowGraphicMenu

Syntax

```
-- Lingo syntax
_movie.allowGraphicMenu

// JavaScript syntax
_movie.allowGraphicMenu;
```

Beschreibung

Diese Filmeigenschaft steuert die Verfügbarkeit der grafischen Steuerelemente im Kontextmenü, wenn der Film in einer Adobe Shockwave®-Umgebung abgespielt wird. Lesen/Schreiben.

Setzen Sie diese Eigenschaft auf FALSE, wenn statt des grafischen Kontextmenüs ein Textmenü angezeigt werden soll.

Diese Eigenschaft ist standardmäßig auf TRUE gesetzt.

Siehe auch

allowCustomCaching, allowSaveLocal, allowTransportControl, allowVolumeControl, allowZooming, Movie

allowSaveLocal

Syntax

```
-- Lingo syntax
_movie.allowSaveLocal

// JavaScript syntax
_movie.allowSaveLocal;
```

Beschreibung

Diese Filmeigenschaft steuert die Verfügbarkeit des Steuerelements „Speichern“ im Kontextmenü, wenn der Film in einer Shockwave® Player-Umgebung abgespielt wird. Lesen/Schreiben.

Sie dient möglichen Funktionsverbesserungen in zukünftigen Versionen von ShockwavePlayer.

Diese Eigenschaft ist standardmäßig auf TRUE gesetzt.

Siehe auch

allowCustomCaching, allowGraphicMenu, allowTransportControl, allowVolumeControl, allowZooming, Movie

allowTransportControl

Syntax

```
-- Lingo syntax
_movie.allowTransportControl

// JavaScript syntax
_movie.allowTransportControl;
```

Beschreibung

Diese Filmeigenschaft dient möglichen Funktionsverbesserungen in zukünftigen Versionen vonShockwave Player. Dies entspricht der Einstellung des Kontrollkästchens „Transportsteuerung“ in den Einstellungen zur Veröffentlichung für Shockwave. Lesen/Schreiben.

Diese Eigenschaft ist standardmäßig auf TRUE gesetzt.

Siehe auch

allowCustomCaching, allowGraphicMenu, allowSaveLocal, allowVolumeControl, allowZooming, Movie

allowVolumeControl

Syntax

```
-- Lingo syntax
_movie.allowVolumeControl

// JavaScript syntax
_movie.allowVolumeControl;
```

Beschreibung

Diese Filmeigenschaft bestimmt die Verfügbarkeit des Lautstärkereglers im Kontextmenü, wenn der Film in einer Shockwave Player-Umgebung abgespielt wird. Lesen/Schreiben.

Wenn die Eigenschaft auf TRUE eingestellt ist, ist einer der Lautstärkeregler aktiv. Er wird deaktiviert, wenn die Eigenschaft auf FALSE eingestellt wird.

Diese Eigenschaft ist standardmäßig auf TRUE gesetzt.

Siehe auch

allowCustomCaching, allowGraphicMenu, allowSaveLocal, allowTransportControl, allowZooming, Movie

allowZooming

Syntax

```
-- Lingo syntax
_movie.allowZooming

// JavaScript syntax
_movie.allowZooming;
```

Beschreibung

Diese Filmeigenschaft bestimmt, ob der Benutzer den Film während der Wiedergabe in Shockwave Player und ShockMachine strecken oder zoomen kann. [Lesen/Schreiben.

Wenn Sie verhindern möchten, dass die Größe des Films im Browser geändert wird, setzen Sie diese Eigenschaft auf `FALSE`.

Die Eigenschaft ist standardmäßig auf `TRUE` gesetzt.

Siehe auch

`allowCustomCaching`, `allowGraphicMenu`, `allowSaveLocal`, `allowTransportControl`, `allowVolumeControl`, `Movie`

alphaThreshold

Syntax

```
-- Lingo syntax
memberObjRef.alphaThreshold

// JavaScript syntax
memberObjRef.alphaThreshold;
```

Beschreibung

Diese Bitmapdarstellereigenschaft bestimmt, welchen Einfluss der Alphakanal der Bitmap auf die Registrierung von Hits (Treffern) hat. Diese Eigenschaft hat einen Wert von 0 bis 255, was genau den Alphawerten im Alphakanal einer 32-Bit-Bitmapgrafik entspricht.

Bei einer bestimmten `alphaThreshold`-Einstellung registriert Director einen Mausklick, wenn der Pixelwert der Alpha-Map an jenem Punkt größer oder gleich dem Schwellenwert ist. Wenn `alphaThreshold` auf 0 eingestellt ist, werden alle Pixel für die Registrierung von Hits opak, ungeachtet des Inhalts des Alphakanals.

Siehe auch

`useAlpha`

ambient

Syntax

```
member(whichCastmember).shader(whichShader).ambient
member(whichCastmember).model(whichModel).shader.ambient
member(whichCastmember).model(whichModel).shaderList{[index]}.ambient
```

Beschreibung

Diese 3D-Eigenschaft für den Shader `#standard` gibt an, welcher Anteil der einzelnen Farbkomponenten des Umgebungslichts im Darsteller vom Shader reflektiert wird.

Wenn die Farbe des Umgebungslichts beispielsweise `rgb(255, 255, 255)` und der Wert der Shader-Eigenschaft `ambientrgb(255, 0, 0)` ist, reflektiert der Shader den gesamten Rotanteil des Lichts, den die Farben des Shaders widerspiegeln können. Die Blau- und Grünanteile des Lichts werden jedoch unabhängig von den Farben des Shaders nicht reflektiert. Wenn es in der Szene keine anderen Lichtquellen gibt, reflektieren die Blau- und Grünfarben des Shaders kein Licht und erscheinen schwarz.

Der Standardwert dieser Eigenschaft ist `rgb(63,63,63)`.

Beispiel

Die folgende Anweisung setzt die Eigenschaft ambient des Modells "Stuhl" auf rgb(255, 255, 0). Das Modell „Stuhl“ reflektiert die Rot- und Grünanteile des Umgebungslichts in der Szene; der Blauanteil wird völlig ignoriert.

```
-- Lingo syntax
member("Room").model("Chair").shader.ambient = rgb(255, 0, 0)

// JavaScript syntax
member("Room").getPropRef("model",1).shader.ambient = color(255,0,0);
```

Siehe auch

`ambientColor`, `newLight`, `type (light)`, `diffuse`, `specular (shader)`

ambientColor

Syntax

```
member(whichCastmember).ambientColor
```

Beschreibung

Diese 3D-Darstellereigenschaft gibt die RGB-Farbe des Standardumgebungslichts des Darstellers an.

Der Standardwert dieser Eigenschaft lautet rgb(0, 0, 0). dadurch wird kein Licht zur Szene hinzugefügt.

Beispiel

Die folgende Anweisung setzt die Eigenschaft `ambientColor` des Darstellers "Raum" auf "rgb(255, 0, 0)". Das Standardumgebungslicht des Darstellers ist rot. Diese Eigenschaft kann auch im Eigenschafteninspektor eingestellt werden.

```
-- Lingo syntax
member("Room").ambientColor = rgb(255, 0, 0)

// JavaScript syntax
member("Room").ambientColor = color(255,0,0);
```

Siehe auch

`directionalColor`, `directionalPreset`, `ambient`

ancestor

Syntax

```
property {optionalProperties} ancestor
```

Beschreibung

Diese Objekteigenschaft ermöglicht Child-Objekten und Verhalten, Prozeduren und Eigenschaften zu verwenden, die nicht im Parent-Skript oder Verhalten enthalten sind.

Die Eigenschaft `ancestor` wird normalerweise mit zwei oder mehreren Parent-Skripts verwendet. Sie können diese Eigenschaft verwenden, wenn Child-Objekte und Verhalten bestimmte Verhalten, die sie von einem Ancestor-Skript übernommen haben, beibehalten und gleichzeitig von anderen Verhalten, die von den Parent-Skripts übernommen wurden, abweichen sollen.

Bei Child-Objekten wird die Eigenschaft `ancestor` gewöhnlich in der Prozedur `on new` innerhalb des Parent-Skripts zugeordnet. Wenn Sie eine Nachricht an ein Child-Objekt senden, das keine definierte Prozedur enthält, wird die Nachricht an das durch die Eigenschaft `ancestor` definierte Skript weitergeleitet.

Wenn ein Verhalten ein Ancestor-Skript hat, erhält dieses Mausereignisse wie z. B. `mouseDown` und `mouseWithin`.

Mit der Eigenschaft `ancestor` können Sie Verhalten und Eigenschaften für eine große Anzahl von Objekten mit einem einzigen Befehl ändern.

Das Ancestor-Skript kann unabhängige Eigenschaftsvariablen enthalten, auf die von Child-Objekten zugegriffen werden kann. Um auf Eigenschaftsvariablen im Ancestor-Skript zu verweisen, müssen Sie die folgende Syntax verwenden:

`me`.*EigenschaftsVariable* = *Wert*

Die folgende Anweisung setzt zum Beispiel die Eigenschaftsvariable `legCount` in einem Ancestor-Skript auf 4:

```
me.legCount = 4
```

Verwenden Sie die Syntax `the`*variableName* `of` *scriptName*, um auf Eigenschaftsvariablen zuzugreifen, die nicht im aktuellen Objekt enthalten sind. Die folgende Anweisung ermöglicht der Variablen `myLegCount` im Child-Objekt den Zugriff auf die Eigenschaftsvariable `legCount` im Ancestor-Skript:

```
set myLegCount to the legCount of me
```

Beispiel

Jedes der nachstehenden Skripts ist ein Darsteller. Das Ancestor-Skript "Tier“ sowie die Parent-Skripts "Hund“ und "Mensch“ interagieren miteinander und definieren so Objekte.

Das erste Skript (Hund) setzt die Eigenschaftsvariable "breed" (Rasse) auf Promenadenmischung, das Ancestor-Skript von "Hund" auf das Skript "Tier" und die im Ancestor-Skript gespeicherte Variable `legCount` (Beinzahl) auf 4:

```
property breed, ancestor

on new me
    set breed = "Mutt"
    set the ancestor of me to new(script "Animal")
    set the legCount of me to 4
    return me
end
```

Das zweite Skript (Mensch) setzt die Eigenschaftsvariable "race" (Rasse) auf „Weiß“, das Ancestor-Skript von „Mensch“ auf das Skript „Tier“ und die im Ancestor-Skript gespeicherte Variable `legCount` auf 2:

```
property race, ancestor
on new me
    set race to "Caucasian"
    set the ancestor of me to new(script "Animal")
    set the legCount of me to 2
    return me
end
```

Siehe auch

new(), menu, property

angle (3D)

Syntax

```
member(whichCastmember).modelResource(whichModelResource).emitter.angle
```

Beschreibung

Diese 3D-Emittereigenschaft beschreibt den Bereich, in den die Partikel eines Partikelsystems ausgestrahlt werden. Ein Partikelsystem ist eine Modellressource vom Typ `#particle`.

Die Hauptrichtung der Partikelemission ist der durch die Emittereigenschaft `direction` festgelegte Vektor. Die Emissionsrichtung eines bestimmten Partikels weicht jedoch um einen willkürlich gewählten Winkel zwischen 0 und dem Wert der Emittereigenschaft `angle` von diesem Vektor ab.

Der Gültigkeitsbereich dieser Eigenschaft liegt zwischen 0.0 und 180.0; der Standardwert ist 180.0.

Beispiel

Die folgende Anweisung setzt den Emissionswinkel der Modellressource `mrFount` auf 1, so dass die ausgestrahlten Partikel eine dünne Linie bilden.

```
member("fountain").modelResource("mrFount").emitter.angle = 1
```

Siehe auch

emitter, direction

angle (DVD)

Syntax

```
-- Lingo syntax
dvdObjRef.angle

// JavaScript syntax
dvdObjRef.angle;
```

Beschreibung

Diese DVD-Eigenschaft gibt die Nummer des aktuellen Kamerawinkels zurück. Lesen/Schreiben.

Der Rückgabewert ist eine Ganzzahl.

Beispiel

Die folgende Anweisung gibt die Nummer des aktuellen Kamerawinkels zurück:

```
-- Lingo syntax
put(member(1).angle)-- 1

// JavaScript syntax
put(member(1).angle);// 1
```

Siehe auch

DVD

angleCount

Syntax

```
-- Lingo syntax
dvdObjRef.angleCount

// JavaScript syntax
dvdObjRef.angleCount;
```

Beschreibung

Diese DVD-Eigenschaft gibt die Anzahl der verfügbaren Kamerawinkel im aktuellen Titel zurück. Nur Lesen.

Der Rückgabewert ist eine Ganzzahl zwischen 1 und 9.

Beispiel

Die folgende Anweisung gibt die Anzahl der verfügbaren Kamerawinkel zurück:

```
-- Lingo syntax
put(member(12).angleCount)-- 2

// JavaScript syntax
put(member(12).angleCount);// 2
```

Siehe auch

DVD

animationEnabled

Syntax

```
member(whichCastmember).animationEnabled
```

Beschreibung

Diese 3D-Darstellereigenschaft gibt an, ob Animationen ausgeführt werden (TRUE) oder nicht (FALSE). Diese Eigenschaft kann auch im Eigenschafteninspektor eingestellt werden.

Der Standardwert dieser Eigenschaft lautet TRUE.

Beispiel

Die folgende Anweisung deaktiviert die Animation für den Darsteller "Szene“:

```
member("Scene").animationEnabled = FALSE
```

antiAlias

Syntax

```
-- Lingo syntax
memberObjRef.antiAlias
// JavaScript syntax
memberObjRef.antiAlias;
```

Beschreibung

Diese Darstellereigenschaft legt fest, ob ein Vektorform- oder Flash® -Darsteller mit Anti-Aliasing (geglättet) wiedergegeben wird, wodurch die Qualität der Darstellung verbessert, aber möglicherweise das Abspielen des Films verlangsamt wird. Die Eigenschaft `antiAlias` ist standardmäßig auf `TRUE` eingestellt.

Bei Vektorformen ist `TRUE` gleichbedeutend mit der Qualitätseinstellung `#high` (hoch) für ein Flash-Element und `FALSE` ist gleichbedeutend mit `#low` (niedrig).

Die Eigenschaft `antiAlias` kann auch als Sprite-Eigenschaft verwendet werden, allerdings nur für Vektorform-Sprites.

Diese Eigenschaft kann getestet und eingestellt werden.

Hinweis: *Die `antiAlias`-Eigenschaft wird nicht für Textdarsteller in Director 11 unterstützt. Verwenden Sie für diese Darsteller die `antiAliasType`-Eigenschaft.*

Beispiel

Das folgende Verhalten prüft die Farbtiefe des Computers, auf dem der Film abgespielt wird. Wenn die Farbtiefe auf 8 Bit oder weniger (256 Farben) eingestellt ist, setzt das Skript die `antiAlias`-Eigenschaft des Sprites auf FALSE.

```
--Lingo syntax
property spriteNum

on beginsprite me
    if _system.colorDepth <= 8 then
        member("text").antiAlias = FALSE
    end if
end

// JavaScript syntax
function beginsprite() {
    var cd = _system.colorDepth;
    if (cd <= 8 ) {
        member("text").antiAlias = 0;
    }
}
```

Siehe auch

antiAliasThreshold, quality

antiAliasingEnabled

Syntax

```
sprite(whichSprite).antiAliasingEnabled
```

Beschreibung

Diese 3D-Sprite-Eigenschaft gibt an, ob für die 3D-Welt im Sprite *whichSprite* ein Anti-Aliasing erfolgt. Sie kann getestet und eingestellt werden. Der Standardwert lautet `FALSE` (Anti-Aliasing deaktiviert). Wenn die Eigenschaft `antiAliasingEnabled` auf `TRUE` gesetzt ist und Sie auf einen 3D-Renderer umschalten, der Anti-Aliasing nicht unterstützt, wird die Eigenschaft automatisch auf `FALSE` eingestellt. Der Wert dieser Eigenschaft wird nicht zusammen mit dem Film gespeichert.

Sprites mit Anti-Aliasing beanspruchen mehr Prozessorleistung und Arbeitsspeicher als Sprites ohne Anti-Aliasing. Um die Leistung von Animationen und Benutzerinteraktionen zu verbessern, sollten Sie das Anti-Aliasing während Operationen dieser Art vorübergehend deaktivieren.

Beispiel

Mit dem folgenden Lingo-Code wird geprüft, ob der aktuelle 3D-Renderer für Sprite 2 das Anti-Aliasing unterstützt (Eigenschaft `antiAliasingSupported`). Wenn ja, wird in der zweiten Anweisung das Anti-Aliasing für das Sprite mithilfe der Eigenschaft `antiAliasingEnabled` aktiviert.

```
if sprite(2).antiAliasingSupported = TRUE then
    sprite(2).antiAliasingEnabled = TRUE
end if
```

Siehe auch

`antiAliasingSupported`, `renderer`, `rendererDeviceList`

antiAliasingSupported

Syntax

```
sprite(whichSprite).antiAliasingSupported
```

Beschreibung

Diese 3D-Sprite-Eigenschaft gibt an, ob das Anti-Aliasing vom aktuellen 3D-Renderer unterstützt wird. Diese Eigenschaft kann getestet, aber nicht eingestellt werden. Sie gibt entweder `TRUE` oder `FALSE` zurück.

Beispiel

Mit dem folgenden Lingo-Code wird geprüft, ob der aktuelle 3D-Renderer für Sprite 3 das Anti-Aliasing unterstützt. Wenn ja, wird in der zweiten Anweisung das Anti-Aliasing für das Sprite mithilfe der Eigenschaft `antiAliasingEnabled` aktiviert.

```
if sprite(3).antiAliasingSupported = TRUE then
    sprite(3).antiAliasingEnabled = TRUE
end if
```

Siehe auch

`antiAliasingEnabled`, `renderer`, `rendererDeviceList`

antiAliasThreshold

Syntax

```
-- Lingo syntax
memberObjRef.antiAliasThreshold

// JavaScript syntax
memberObjRef.antiAliasThreshold;
```

Beschreibung

Diese Textdarsteller-Eigenschaft steuert die Punktgröße, bei der ein automatisches Anti-Aliasing in einem Textdarsteller erfolgt. Dies hat nur dann eine Wirkung, wenn die Eigenschaft `antiAlias` des Textdarstellers auf `TRUE` eingestellt ist.

Die Einstellung selbst ist eine Ganzzahl, die den Schriftgrad angibt, bei dem ein Anti-Aliasing stattfindet.

Der Standardwert dieser Eigenschaft lautet 14 Punkte.

Siehe auch

`antiAliasType`

antiAliasType

Syntax

```
-- Lingo syntax
memberObjRef.antiAliasType
// JavaScript syntax
memberObjRef.antiAliasType;
```

Beschreibung

Diese Darstellereigenschaft legt fest, ob ein Textdarsteller mit Anti-Aliasing (geglättet) wiedergegeben wird, wodurch die Qualität der Darstellung verbessert. Die Eigenschaft `antiAliasType` ist standardmäßig auf `#autoAlias` eingestellt.

Wenn Sie eine frühere Version von Director aktualisieren, ist die Option "Automatisch" auf "Graustufe·größer als" zugeordnet, wobei `antiAliasThreshold` ·auf Null gesetzt ist Die Option „Größer als“ ist mit dem entsprechenden Schwellenwert auf "Graustufe größer als" zugeordnet.

Verwenden Sie die folgenden Symbole, um die entsprechende Anti-Aliasing-Option festzulegen:

Auto #AutoAlias (Uses the font file information for anti-aliasing.)–

Grayscale All - #GrayScaleAllAlias (Aktiviert das Graustufen-Anti-Aliasing für alle Textdarsteller.)

Subpixel All - #SubpixelAllAlias (Aktiviert das Sub-Pixel-Anti-Aliasing für alle Textdarsteller.)

Grayscale Larger Than - #GrayscaleLargerThanAlias (Aktiviert das Graustufen-Anti-Aliasing mit einem Schriftgrad größer als dem angegebenen Schwellenwert.)

None - #NoneAlias (keine bewirkt, dass für den aktuellen Darsteller kein Anti-Aliasing durchgeführt wird.)

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Das folgende Verhalten prüft die Farbtiefe des Computers, auf dem der Film abgespielt wird. Wenn die Farbtiefe auf 8 Bit oder weniger (256 Farben) eingestellt ist, setzt das Skript die `antiAlias`-Eigenschaft des Sprites auf FALSE.

```
--Lingo syntax
property spriteNum

on beginsprite me
    if _system.colorDepth <= 8 then
        member("text").antiAliasType = #noneAlias
    end if
end

// JavaScript syntax
function beginsprite() {
    var cd = _system.colorDepth;
    if (cd <= 8 ) {
        member("text").antiAliasType = symbol("noneAlias");
    }
}
```

Siehe auch

antiAliasThreshold

appearanceOptions

Syntax

```
-- Lingo syntax
windowObjRef.appearanceOptions

// JavaScript syntax
windowObjRef.appearanceOptions;
```

Beschreibung

Diese Fenstereigenschaft gibt eine Liste von Eigenschaften an, in der die Darstellungsoptionen eines Fensters gespeichert sind. Lesen/Schreiben.

Die Eigenschaftsliste enthält die folgenden Eigenschaften.

Eigenschaft	Beschreibung
`#mask`	Gibt den als Maske für das Fenster zu verwendenden 1-Bit-Darsteller an.
`#border`	Gibt den Rahmentyp für das Fenster an. Diese Eigenschaft kann einen der folgenden drei Werte aufweisen: • `#none`. Kein Rahmen ums Fenster. • `#line`. Ein schwarzer 1-Pixel-Rahmen ums Fenster. Die Eigenschaften `#none` und `#line` wirken sich nur aus, wenn die Eigenschaft `titlebarOptions.visible` auf `FALSE` eingestellt ist.
`#metal`	(Nur Mac) Gibt an, ob das Fenster wie Metall aussieht (`TRUE`). Bei Festlegung auf `FALSE` sieht das Fenster wie Eis aus.

Eigenschaft	Beschreibung
`#dragRegionMask`	Gibt den als Maske für einen Bereich des Fensters zu verwendenden 1-Bit-Darsteller an.
`#shadow`	(Nur Mac) Gibt an, ob das Fenster einen Schatten hat. Mac-Fenster haben normalerweise einen Schatten.
`#liveresize`	(Nur Mac) Gibt an, ob die Größe des Fenster während der Änderung angepasst wird („live"). Bei `TRUE` ist diese Funktion aktiviert, bei `False` ist sie deaktiviert.

Auf diese Eigenschaften kann ebenfalls unter Verwendung der Eigenschaft `displayTemplate` des Movie-Objekts zugegriffen werden.

Beispiel

Die folgende Anweisung zeigt alle aktuellen Darstellungsoptionen für das Fenster namens „Control Panel" im Nachrichtenfenster an:

```
-- Lingo syntax
put(window("Control Panel").appearanceOptions)

// JavaScript syntax
put(window("Control Panel").appearanceOptions);
```

Die folgende Anweisung legt die Eigenschaft border auf die Anzeige eines 1-Pixel-Rahmens um das Fenster „Control Panel" an:

```
-- Lingo syntax
window("Control Panel").appearanceOptions.border = #line

// JavaScript syntax
window("Control Panel").appearanceOptions.border = symbol("line");
```

Siehe auch

`displayTemplate`, `titlebarOptions`, `visible`, `Window`

applicationName

Syntax

```
-- Lingo syntax
_player.applicationName

// JavaScript syntax
_player.applicationName;
```

Beschreibung

Diese Player-Eigenschaft gibt den Namen der Kopie der Director-Anwendung an, die bei der Erstellung ausgeführt wird, oder den Namen einer Projektordatei während der Laufzeit. Nur Lesen.

Der Eigenschaftswert ist ein String.

Der Shockwave Player bietet keine Unterstützung für diese Eigenschaft.

Beispiel

Die folgende Anweisung zeigt den Namen der Director-Anwendung „Director.exe" an.

```
-- Lingo syntax
put(_player.applicationName)

// JavaScript syntax
put(_player.applicationName);
```

Siehe auch

applicationPath, Player

applicationPath

Syntax

```
-- Lingo syntax
_player.applicationPath

// JavaScript syntax
_player.applicationPath;
```

Beschreibung

Diese Player-Eigenschaft bestimmt den Pfad oder die Position des Ordners, der die laufende Kopie der Director-Anwendung bei der Erstellung enthält, oder den Ordner, der den Projektor während der Laufzeit enthält. Nur Lesen.

Der Eigenschaftswert ist ein String.

Wenn Sie `applicationPath`, gefolgt von `&` und dem Pfad zu einem Unterordner verwenden, setzen Sie den gesamten Ausdruck in Klammern, damit Skriptcode den Ausdruck als einen ganzen Ausdruck versteht.

Der Shockwave Player bietet keine Unterstützung für diese Eigenschaft.

Beispiel

Die folgende Anweisung zeigt den Pfadnamen für den Ordner an, in dem sich die Director-Anwendung befindet.

```
-- Lingo syntax
put(_player.applicationPath)

// JavaScript syntax
put(_player.applicationPath);
```

Die folgende Anweisung öffnet den Film „Sunset Boulevard“ in einem Fenster (auf einem Windows-System):

```
-- Lingo syntax
window(_player.applicationPath & "Film Noir\Sunset Boulevard").open()

// JavaScript syntax
window(_player.applicationPath + "Film Noir\Sunset Boulevard").open();
```

Siehe auch

applicationName, Player

aspectRatio

Syntax

```
-- Lingo syntax
dvdObjRef.aspectRatio

// JavaScript syntax
dvdObjRef.aspectRatio;
```

Beschreibung

Diese DVD-Eigenschaft gibt eine Eigenschaftsliste zurück, die die Breite und Höhe des DVD-Darstellers angibt. Nur Lesen.

Die Rückgabewerte für Breite und Höhe sind Ganzzahlen.

Beispiel

Die folgende Anweisung gibt das Bildgrößenverhältnis `aspectRatio` von Darsteller 1 zurück:

```
-- Lingo syntax
trace(member(1).aspectRatio) -- [#width: 16, #height:9]

// JavaScript syntax
trace(member(1).aspectRatio); // ["width": 16, "height":9];
```

Siehe auch

DVD

attenuation

Syntax

```
member(whichCastMember).light(whichLight).attenuation
```

Beschreibung

Diese 3D-Lichteigenschaft gibt den konstanten, linearen und quadratischen Dämpfungsfaktor für Spot- und Punktlichter an.

Der Standardwert dieser Eigenschaft lautet vector(1.0, 0.0, 0.0).

Beispiel

Die folgende Anweisung setzt die Dämpfungseigenschaft des Lichts `HouseLight` auf "vector(.5, 0, 0)", wodurch es leicht abgedunkelt wird.

```
-- Lingo syntax
member("3d world").light("HouseLight").attenuation = vector(.5, 0, 0)

// JavaScript syntax
member("3d world").getProp("light",1).attenuation = vector(.5, 0, 0);
```

Siehe auch

color (light)

attributeName

Syntax

```
XMLnode.attributeName[ attributeNumber ]
```

Beschreibung

Diese XML-Eigenschaft gibt den Namen des angegebenen Child-Nodes eines geparsten XML-Dokuments zurück.

Beispiel

Als Ausgangspunkt wird das folgende XML-Dokument verwendet:

```
<?xml version="1.0"?>
    <e1>
        <tagName attr1="val1" attr2="val2"/>
        <e2> element 2</e2>
        <e3>element 3</e3>
        here is some text
    </e1>
```

Die folgende Lingo-Anweisung gibt den Namen des ersten Attributs des Tags `tagName` zurück:

```
put gParserObject.child[1].child[1].attributeName[1]
-- "attr1"
```

Siehe auch

attributeValue

attributeValue

Syntax

```
XMLnode.attributeValue[ attributeNameOrNumber ]
```

Beschreibung

Diese XML-Eigenschaft gibt den Wert des angegebenen Child-Nodes eines geparsten XML-Dokuments zurück.

Beispiel

Als Ausgangspunkt wird das folgende XML-Dokument verwendet:

```
<?xml version="1.0"?>
    <e1>
        <tagName attr1="val1" attr2="val2">
        <e2> element 2</e2>
        <e3>element 3</e3>
        here is some text
    </e1>
```

Die folgende Lingo-Anweisung gibt den Wert des ersten Attributs des Tags `tagName` zurück:

```
put gParserObject.child[1].child[1].attributeValue[1]
-- "val1"
```

Siehe auch

attributeName

audio (DVD)

Syntax

```
-- Lingo syntax
dvdObjRef.audio

// JavaScript syntax
dvdObjRef.audio;
```

Beschreibung

Diese DVD-Eigenschaft bestimmt, ob Audio aktiviert ist (TRUE, Standard) oder nicht (FALSE). Lesen/Schreiben.

Beispiel

Die folgende Anweisung deaktiviert Audio:

```
-- Lingo syntax
member(14).audio = 0

// JavaScript syntax
member(14).audio = 0;
```

Siehe auch

DVD

audio (MP4Media/FLV)

Syntax

```
member(1).audio = true
Sprite(1).audio = true
```

Beschreibung

MP4Media/FLV-Eigenschaft, die festlegt, ob Audio aktiviert ist (TRUE, Standard) oder nicht (FALSE). Diese Eigenschaft kann ausgelesen und beschrieben werden.

Beispiele

In folgendem Beispiel wird die Audioeigenschaft für „Sprite 2“ und des Darstellers „MP4Media/FLV“ auf „True“ gesetzt. Der Audioanteil des MP4Media/FLV-Streams wird beim Abspielen des Films wiedergegeben.

```
-- Lingo syntax
put(sprite(2).audio) -- 1
put(member("MP4Media/FLV").audio) -- 1

// JavaScript syntax
put(sprite(2).audio); // 1
put(member("MP4Media/FLV").audio); // 1
```

In folgendem Beispielen wird die Audioeigenschaft für Sprite 2 und des Darstellers „MP4Media/FLV“ auf „False“ gesetzt. Der Audioanteil des MP4Media/FLV-Streams wird beim Abspielen des Films nicht wiedergegeben.

```
-- Lingo syntax
sprite(2).audio = FALSE
member("MP4Media/FLV").audio = FALSE

// JavaScript syntax
sprite(2).audio = 0;
member("MP4Media/FLV").audio = 0;
```

audio (RealMedia)

Syntax

```
-- Lingo syntax
memberOrSpriteObjRef.audio

// JavaScript syntax
memberOrSpriteObjRef.audio;
```

Beschreibung

Mit dieser Eigenschaft für `RealMedia`-Sprites und -Darsteller können Sie das Audiosegment im `RealMedia`-Stream abspielen (`TRUE`) oder ausschalten (`FALSE`). Die Standardeinstellung dieser Eigenschaft lautet `TRUE` (1). Andere Ganzzahlen als 1 oder 0 werden als `TRUE` (1) interpretiert. Das Festlegen dieser Eigenschaft hat keine Auswirkungen, wenn die `realPlayerNativeAudio()`-Methode auf `TRUE` festgelegt ist.

Selbst wenn die Eigenschaft `audio` zu Beginn der Wiedergabe eines `RealMedia`-Darstellers auf `FALSE` gesetzt ist, wird ein Soundkanal zugeordnet, damit der Sound beim Abspielen ein- und ausgeschaltet werden kann.

Beim Einstellen dieser Eigenschaft tritt u. U. eine gewisse Latenz auf, sodass es zu einer kurzen Verzögerung kommen kann, bevor der Sound ein- oder ausgeschaltet wird.

Beispiel

In den folgenden Beispielen wird die Eigenschaft „audio“ für Sprite 2 und den Darsteller „Real“ auf TRUE gesetzt, d. h. der Audioabschnitt des `RealMedia`-Streams wird abgespielt.

```
-- Lingo syntax
put(sprite(2).audio) -- 1
put(member("Real").audio) -- 1

// JavaScript syntax
put(sprite(2).audio); // 1
put(member("Real").audio); // 1
```

Der folgende Lingo-Code legt die Eigenschaft „audio“ für Sprite 2 und den Darsteller „Real“ auf FALSE fest, d. h. der Audioabschnitt des `RealMedia`-Streams wird beim Abspielen des Films nicht wiedergegeben.

```
-- Lingo syntax
sprite(2).audio = FALSE
member("Real").audio = FALSE

// JavaScript syntax
sprite(2).audio = 0;
member("Real").audio = 0;
```

Siehe auch

soundChannel (RealMedia), video (RealMedia, Windows Media), sound (Player)

audio (Windows Media)

Syntax

```
-- Lingo syntax
windowsMediaObjRef.audio

// JavaScript syntax
windowsMediaObjRef.audio;
```

Beschreibung

Diese Windows Media-Eigenschaft bestimmt, ob Audio aktiviert ist (TRUE, Standard) oder nicht (FALSE). Lesen/Schreiben.

Beispiel

Die folgende Anweisung zeigt im Nachrichtenfenster an, ob Audio für Darsteller 5 aktiviert ist:

```
-- Lingo syntax
trace(member(5).audio)

// JavaScript syntax
trace(member(5).audio);
```

Siehe auch

Windows Media

audioChannelCount

Syntax

```
-- Lingo syntax
dvdObjRef.audioChannelCount

// JavaScript syntax
dvdObjRef.audioChannelCount;
```

Beschreibung

Diese DVD-Eigenschaft gibt die Anzahl der Audiokanäle zurück. Nur Lesen.

Beispiel

Die folgende Anweisung gibt die Anzahl der Audiokanäle zurück:

```
-- Lingo syntax
member(1).audioChannelCount

// JavaScript syntax
member(1).audioChannelCount;
```

Siehe auch

DVD

audioExtension

Syntax

```
-- Lingo syntax
dvdObjRef.audioExtension

// JavaScript syntax
dvdObjRef.audioExtension;
```

Beschreibung

Diese DVD-Eigenschaft gibt ein Symbol zurück, das, falls vorhanden, die Audioerweiterungen eines Audiostreams angibt. Nur Lesen.

Mögliche Rückgabewerte sind:

Symbol	Beschreibung
`#caption`	Der Audiostream enthält Untertitel.
`#lowvision`	Der Audiostream enthält Inhalte für Personen mit eingeschränktem Sehvermögen.
`#directorcomments1`	Der Audiostream enthält „Regieanweisungen 1“.
`#directorcomments2`	Der Audiostream enthält „Regieanweisungen 2“.
`#none`	Die DVD gibt keine Audioerweiterung für diesen Audiostream an bzw. es konnte keine ermittelt werden.

Siehe auch

DVD

audioFormat

Syntax

```
-- Lingo syntax
dvdObjRef.audioFormat

// JavaScript syntax
dvdObjRef.audioFormat;
```

Beschreibung

Diese DVD-Eigenschaft gibt ein Symbol zurück, das das Format (Kodiermodus) eines Audiostreams angibt. Nur Lesen.

Mögliche Rückgabewerte sind:

Symbol	Beschreibung
#AC3	Das Audioformat ist Dolby AC-3.
#MPEG1	Das Audioformat ist MPEG-1.
#MPEG1DRC	Das Audioformat ist MPEG-1 mit Dynamikbereichssteuerung.
#MPEG2	Das Audioformat ist MPEG-2.
#MPEG2DRC	Das Audioformat ist MPEG-2 mit Dynamikbereichssteuerung.
#LPCM	Das Audioformat ist LPCM (Linear Pulse Code Modulated).
#DTS	Das Audioformat ist DTS (Digital Theater Systems).
#SDDS	Das Audioformat ist SDDS (Sony Dynamic Digital Sound).

Siehe auch

DVD

audioSampleRate

Syntax

```
-- Lingo syntax
dvdObjRef.audioSampleRate

// JavaScript syntax
dvdObjRef.audioSampleRate;
```

Beschreibung

Diese DVD-Eigenschaft gibt die Frequenz eines Audiostreams in Hertz zurück. Nur Lesen.

Siehe auch

DVD

audioStream

Syntax

```
-- Lingo syntax
dvdObjRef.audioStream

// JavaScript syntax
dvdObjRef.audioStream;
```

Beschreibung

Diese DVD-Eigenschaft gibt den aktuell aktiven Audiostream zurück. Lesen/Schreiben.

Gültige Werte: 1 bis 8.

Siehe auch

DVD

audioStreamCount

Syntax

```
-- Lingo syntax
dvdObjRef.audioStreamCount

// JavaScript syntax
dvdObjRef.audioStreamCount;
```

Beschreibung

Diese DVD-Eigenschaft gibt die Anzahl der verfügbaren Audiostreams im aktuellen Titel zurück. Nur Lesen.

Die Anzahl der verfügbaren Audiostreams liegt zwischen 1 und 8.

Siehe auch

DVD

auto

Syntax

```
member(whichCastmember).model(whichModel).lod.auto
```

Beschreibung

Diese 3D-Eigenschaft für den Modifizierer `lod` ermöglicht es dem Modifizierer, die Abnahme von Details im Modell zu verwalten, wenn sich der Abstand zwischen Modell und Kamera ändert.

Die Einstellung der Modifiziereigenschaft `bias` bestimmt, wie aggressiv der Modifizierer Details aus dem Modell entfernt, wenn die Eigenschaft `auto` auf `TRUE` gesetzt ist.

Der Modifizierer aktualisiert die Eigenschaft `level`, wenn er den Detailliertheitsgrad anpasst. Die Einstellung der Eigenschaft `level` ist wirkungslos, es sei denn, die Eigenschaft `auto` ist auf `FALSE` gesetzt.

Der Modifizierer `#lod` kann nur zu Modellen hinzugefügt werden, die außerhalb von Director in einem 3D-Modellierungsprogramm erstellt wurden. Der Wert der Eigenschaft `type` der von diesen Modellen verwendeten Modellressourcen lautet `#fromFile`. Der Modifizierer kann nicht zu in Director erstellten Primitiven hinzugefügt werden.

Beispiel

Die folgende Anweisung setzt die Eigenschaft auto des Modifizierers "lod" des Modells "Raumschiff" auf TRUE. Der Modifizierer legt automatisch den Detailliertheitsgrad des Modells fest.

```
-- Lingo syntax
member("3D World").model("Spaceship").lod.auto = TRUE

// Java Script
member("3D World").getPropRef("model", 1).getPropRef("lod", 1).auto = true;
```

Siehe auch

lod (Modifizierer), bias, level

autoblend

Syntax

```
member(whichCastmember).model(whichModel).keyframePlayer.autoblend
member(whichCastmember).model(whichModel).bonesPlayer.autoblend
```

Beschreibung

Diese 3D-Eigenschaft für die Modifizierer `keyframePlayer` und `bonesPlayer` gibt an, ob der Modifizierer einen linearen Übergang zur aktuell wiedergegebenen Bewegung von der vorhergehenden Bewegung erzeugt (`TRUE`) oder nicht (`FALSE`). Wenn `autoBlend` auf `TRUE` gesetzt ist, wird die Länge des Übergangs durch die Modifiziereigenschaft `blendTime` festgelegt. Wenn `autoBlend` auf `FALSE` gesetzt ist, wird der Übergang durch die Modifiziereigenschaft `blendFactor` gesteuert und `blendTime` wird ignoriert.

Die Bewegungsüberblendung wird ganz deaktiviert, wenn `blendTime` auf 0 und `autoBlend` auf `TRUE` gesetzt ist.

Der Standardwert dieser Eigenschaft lautet `TRUE`.

Beispiel

Die folgende Anweisung deaktiviert autoblend für das Modell „Kreatur3“. Um aufeinanderfolgende Bewegungen in der Abspielliste zu mischen, wird die `blendFactor`-Einstellung des Modells verwendet.

```
-- Lingo syntax
member("newaliens").model("Alien3").addModifier(#keyframeplayer)
member("newaliens").model("Alien3").keyframePlayer.autoblend = FALSE

// JavaScript syntax
member("newaliens").getPropRef("model",1).addModifier(symbol("keyframeplayer"));
member("newaliens").getPropRef("model",1).getPropRef("keyframeplayer",1).autoBlend = false;
```

Siehe auch

blendFactor, blendTime

autoCameraPosition

Syntax

```
member(whichTextCastmember).autoCameraPosition
```

Beschreibung

Diese 3D-Kameraeigenschaft gibt an, ob die Kamera des 3D-Textdarstellers automatisch so positioniert wird, dass der gesamte Text zu sehen ist (`TRUE`) oder nicht (`FALSE`). Dies ist besonders dann nützlich, wenn Sie den Text, die Schriftart, die Schriftgröße und andere Eigenschaften des Darstellers ändern.

Diese Eigenschaft ist bei anderen Arten von 3D-Darstellern nicht gültig.

Beispiel

Die folgende Anweisung setzt die Eigenschaft `autoCameraPosition` des Darstellers "Ueberschrift" auf FALSE. Bei Anzeige des Darstellers im 3D-Modus wird die Kamera nicht automatisch positioniert.

```
-- Lingo syntax
member("Headline").autoCameraPosition = FALSE

// JavaScript syntax
member("Headline").autoCameraPosition = false;
```

Siehe auch

displayMode

autoMask

Syntax

```
member(whichCursorCastMember).autoMask
the autoMask of member whichCastMember
```

Beschreibung

Diese Darstellereigenschaft gibt an, ob die weißen Pixel im animierten Farbcursor-Darsteller *whichCursorCastMember* transparent sind, sodass der Hintergrund durchscheint (`TRUE`, Standard), oder ob sie opak sind (`FALSE`).

Beispiel

Wenn in diesem Skript der benutzerdefinierte animierte Cursor, der im Darsteller 5 gespeichert ist, in das Sprite tritt, wird die Auto-Maske eingeschaltet. Hierdurch wird der Hintergrund des Sprites durch die weißen Pixel hindurch sichtbar. Wenn der Cursor das Sprite wieder verlässt, wird die Auto-Maske ausgeschaltet.

```
-- Lingo syntax
on mouseEnter
    member 5.autoMask = TRUE
end

on mouseLeave
    member 5.autoMask = FALSE
end
```

In der herkömmlichen Lingo-Syntax wird das Skript folgendermaßen geschrieben:

```
on mouseEnter
    set the autoMask of member 5 = TRUE
end

on mouseLeave
    set the autoMask of member 5 = FALSE
end
```

autoTab

Syntax

```
-- Lingo syntax
memberObjRef.autoTab

// JavaScript syntax
memberObjRef.autoTab;
```

Beschreibung

Diese Darstellereigenschaft bestimmt, welche Wirkung die Betätigung der Tabulatortaste auf den bearbeitbaren Feld- oder Textdarsteller hat, der in *whichCastMember* angegeben ist. Die Eigenschaft kann aktiviert (`TRUE`) oder deaktiviert (`FALSE`) werden. Die Tabulatorreihenfolge basiert auf der Reihenfolge der Sprite-Nummern, nicht deren Positionen auf der Bühne.

Beispiel

Die folgende Anweisung bewirkt, dass der Darsteller "Kommentare“ die Cursorposition automatisch zum nächsten bearbeitbaren Feld- oder Text-Sprite vorrückt, wenn der Benutzer die Tabulatortaste drückt.

```
--Lingo syntax
member ("Comments").autotab = TRUE

// JavaScript syntax
member ("Comments").autotab = true;
```

axisAngle

Syntax

```
member(whichCastmember).model(whichModel).transform.axisAngle
member(whichCastmember).camera(whichCamera).transform.axisAngle
member(whichCastmember).light(whichLight).transform.axisAngle
member(whichCastmember).group(whichGroup).transform.axisAngle
transformReference.axisAngle
```

Beschreibung

Diese 3D-Transformationseigenschaft beschreibt die Drehung der Transformation als Achsen-/Winkelpaar.

Die Eigenschaft `axisAngle` ist eine lineare Liste mit einem Vektor (Achse) und einer Fließkommazahl (Winkel). Der Vektor ist die Achse, um die die Transformation gedreht wird. Die Fließkommazahl ist der Drehwinkel in Grad.

Der Standardwert dieser Eigenschaft lautet `[vector( 1.0000, 0.0000, 0.0000 ), 0.0000]`.

Beispiel

Die folgende Anweisung zeigt die Drehung des Modells „Briefkasten“ als `axisAngle` an. Das Modell wird um 145,5° gegen den Uhrzeigersinn um die y-Achse gedreht.

```
-- Lingo syntax
put member("Yard").model("Mailbox").transform.axisAngle
-- [vector( 0.0000, 1.0000, 0.0000 ), -145.5000]

// JavaScript syntax
put(member("Yard").getProp("model",1).transform.axisAngle);
// <[vector( 0.0000, 1.0000, 0.0000 ), -145.5000]>
```

Siehe auch

rotation (Transformation)

back

Syntax

```
member(whichCastmember).modelResource(whichModelResource).back
```

Beschreibung

Diese 3D-Eigenschaft für die Modellressource `#box` gibt an, ob die Seite der Box, die von der +Z-Achse geschnitten wird, geschlossen (`TRUE`) oder offen ist (`FALSE`).

Der Standardwert dieser Eigenschaft lautet `TRUE`.

Beispiel

Die folgende Anweisung setzt die Eigenschaft back der Modellressource "Kiste" auf FALSE, d. h. die Rückseite der Box ist offen.

```
-- Lingo syntax
nmr = member("3D World").newModelResource("Crate", symbol("box"))
member("3D World").modelResource("Crate").back = FALSE

// JavaScript syntax
nmr = member("3D World").newModelResource("Crate", symbol("box"));
member("3D World").getProp("modelresource",10).back = false;
```

Siehe auch

bottom (3D), front, top (3D), left (3D), right (3D)

backColor

Syntax

```
-- Lingo syntax
spriteObjRef.backColor

// JavaScript syntax
spriteObjRef.backColor;
```

Beschreibung

Diese Sprite-Eigenschaft stellt die Hintergrundfarbe eines angegebenen Sprites auf den zugeordneten Farbwert ein. Lesen/Schreiben.

Das Einstellen der Eigenschaft `backColor` eines Sprites hat dieselbe Wirkung wie die Auswahl der Hintergrundfarbe in der Werkzeugpalette, wenn das Sprite auf der Bühne ausgewählt ist. Damit der mit Skriptcode festgesetzte Wert über das aktuelle Sprite hinweg gültig ist, muss das Sprite ein mit Skript erstelltes Sprite sein. Die Hintergrundfarbe gilt zusätzlich zu Feld-, Schaltflächen-, Kontrollkästchen- und Radiodarstellern für alle Bitmapdarsteller.

Die `backColor`-Werte liegen zwischen 0 und 255 für 8-Bit-Farben und zwischen 0 und 15 für 4-Bit-Farben. Die Zahlen entsprechen der Indexnummer der Hintergrundfarbe in der aktuellen Palette. (Die Indexnummer einer Farbe wird in der linken unteren Ecke der Farbpalette angezeigt, wenn Sie auf die Farbe klicken.)

Wenn diese Eigenschaft auf Bitmapdarsteller festgelegt wird, die eine höhere Tiefe als 1-Bit haben, ist die `backColor` möglicherweise nicht sichtbar, wenn der Hintergrund des Bitmaps nicht angezeigt wird.

Wenn die Mischung eines Sprites unter 100, aber größer als 0 ist, mischt sich die `backColor` mit den transparenten Farben.

Hinweis: *Anstelle der Eigenschaft `backColor` sollte die neuere Eigenschaft `bgColor` verwendet werden.*

Beispiel

Die folgende Anweisung stellt die Variable `oldColor` auf die Hintergrundfarbe von Sprite 5:

```
-- Lingo syntax
oldColor = sprite(5).backColor

// JavaScript syntax
var oldColor = sprite(5).backColor;
```

Die folgende Anweisung ändert die Hintergrundfarbe eines nach dem Zufallsprinzip ausgewählten Sprites zwischen Sprite 11 und Sprite 13 in Farbe 36:

```
-- Lingo syntax
sprite(10 + random(3)).backColor = 36

// JavaScript syntax
sprite(10 + random(3)).backColor = 36;
```

Siehe auch

Sprite

backdrop

Syntax

```
sprite(whichSprite).camera{(index)}.backdrop[index].loc
member(whichCastmember).camera(whichCamera).backdrop[index].loc
sprite(whichSprite).camera{(index)}.backdrop[index].source
member(whichCastmember).camera(whichCamera).backdrop[index].source
sprite(whichSprite).camera{(index)}.backdrop[index].scale
member(whichCastmember).camera(whichCamera).backdrop[index].scale
sprite(whichSprite).camera{(index)}.backdrop[index].rotation
member(whichCastmember).camera(whichCamera).backdrop[index].rotation
sprite(whichSprite).camera{(index)}.backdrop[index].regPoint
member(whichCastmember).camera(whichCamera).backdrop[index].regPoint
sprite(whichSprite).camera{(index)}.backdrop[index].blend
member(whichCastmember).camera(whichCamera).backdrop[index].blend
sprite(whichSprite).camera{(index)}.backdrop.count
member(whichCastmember).camera(whichCamera).backdrop.count
```

Beschreibung

Diese 3D-Kameraeigenschaft gibt ein 2D-Bild an, das auf der Projektionsebene der Kamera wiedergegeben wird. Alle Modelle in der Kameraansicht erscheinen vor dem Hintergrund.

Ein Hintergrund hat folgende Eigenschaften:

Hinweis: *Mithilfe dieser Eigenschaften lassen sich auch Überlagerungen ermitteln, festlegen und manipulieren. Weitere Informationen finden Sie in den Einträgen zu den einzelnen Eigenschaften.*

loc (backdrop and overlay)gibt die 2D-Position des Hintergrunds bzw. der Überlagerung von der oberen linken Ecke des Sprites gemessen an.

sourcegibt die für den Hintergrund verwendete Textur an.

scale (backdrop and overlay) ist eine Zahl, um die die Höhe und Breite der Textur zur Bestimmung der Hintergrundabmessungen multipliziert werden.

rotation (backdrop and overlay) ist der Betrag, um den der Hintergrund um das Registrierungskreuz `regPoint` gedreht ist.

regPoint (3D)ist das Registrierungskreuz des Hintergrundes.

blend (3D) ist die Opazität des Hintergrundes.

count (3D) gibt die Anzahl von Einträgen in der Hintergrundliste der Kamera an.

Die folgenden Befehle dienen zum Erstellen und Entfernen von Hintergründen:

addBackdrop erstellt einen Hintergrund aus einer Textur und fügt ihn ans Ende der Hintergrundliste der Kamera an.

insertBackdrop erstellt einen Hintergrund aus einer Textur und fügt ihn an der angegebenen Indexposition in die Hintergrundliste der Kamera ein.

removeBackdrop entfernt den Hintergrund.

Beispiel

```
-- Lingo syntax
put sprite(2).camera.backdrop[1].scale
//JavaScript syntax
put (sprite(2).camera.getPropRef("backdrop",1).scale);
```

Siehe auch

overlay

insertBackdrop

removeBackdrop

backgroundColor

Syntax

```
-- Lingo syntax
memberObjRef.backgroundColor

// JavaScript syntax
memberObjRef.backgroundColor;
```

Beschreibung

Diese Vektorform-Darstellereigenschaft stellt die Hintergrundfarbe des angegebenen Darstellers oder Sprites auf den zugeordneten RGB-Farbwert ein.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

```
-- Lingo syntax
member("Archie").backgroundColor= color(255,255,255)
// JavaScript syntax
member("Archie").backgroundColor= color(255,255,255);
```

Siehe auch

bgColor (Fenster)

beepOn

Syntax

```
-- Lingo syntax
_movie.beepOn

// JavaScript syntax
_movie.beepOn;
```

Beschreibung

Diese Filmeigenschaft bestimmt, ob der Computer automatisch einen Warnton erklingen lässt, wenn der Benutzer außerhalb eines aktiven Sprites klickt (`TRUE`) oder nicht (`FALSE`, Standard). Lesen/Schreiben.

Skripts, die die Eigenschaft `beepOn` einstellen, sollten in Bild- oder Filmskripten gestellt werden.

Beispiel

Die folgende Anweisung setzt die Eigenschaft `beepOn` auf TRUE:

```
-- Lingo syntax
_movie.beepOn = TRUE

// JavaScript syntax
_movie.beepOn = true;
```

Die folgende Anweisung kehrt die aktuelle Einstellung von `beepOn` um:

```
-- Lingo syntax
_movie.beepOn = not(_movie.beepOn)

// JavaScript syntax
_movie.beepOn = !(_movie.beepOn);
```

Siehe auch

Movie

bevelDepth

Syntax

```
member(whichTextCastmember).bevelDepth
member(which3DCastmember).modelResource(whichModelResource).bevelDepth
```

Beschreibung

Diese 3D-Texteigenschaft gibt den Abschrägungsgrad des 3D-Texts an.

Bei Textdarstellern hat diese Eigenschaft keinerlei Auswirkung, es sei denn, die Darstellereigenschaft `displayMode` ist auf `#mode3D` und die Eigenschaft `bevelType` auf `#miter` oder `#round` gesetzt.

Bei extrudiertem Text in einem 3D-Darsteller hat diese Eigenschaft keinerlei Auswirkung, es sei denn, die Modellressourceneigenschaft `bevelType` ist auf `#miter` oder `#round` gesetzt.

Der Gültigkeitsbereich dieser Eigenschaft liegt zwischen 0.0 und 10.0; die Standardeinstellung ist 10.0.

Beispiel

In diesem Beispiel ist der Darsteller "Logo“ ein Textdarsteller. Die folgende Anweisung setzt die Eigenschaft `bevelDepth` von „Logo“ auf 5.5. Wenn "Logo" im 3D-Modus angezeigt wird und die Eigenschaft `bevelType` auf `#miter` oder `#round` gesetzt ist, weisen die Kanten der Buchstaben eine dramatische Abschrägung auf.

```
-- Lingo syntax
member("Logo").bevelDepth = 5.5

// JavaScript syntax
member("Logo").bevelDepth = 5.5;
```

Im folgenden Beispiel ist die Modellressource des Modells "Slogan" extrudierter Text. Die folgende Anweisung setzt die Eigenschaft `bevelDepth` der Modellressource von „Slogan“ auf 5. Wenn die Eigenschaft `bevelType` von „Slogan“ auf `#miter` oder `#round` gesetzt ist, weisen die Kanten der Buchstaben eine dramatische Abschrägung auf.

```
-- Lingo syntax
member("scene").model("Slogan").resource.bevelDepth = 5
```

Siehe auch

`bevelType`, `extrude3D`, `displayMode`

bevelType

Syntax

```
member(whichTextCastmember).bevelType
member(which3DCastmember).modelResource(whichModelResource).bevelType
```

Beschreibung

Diese 3D-Texteigenschaft gibt die Art der Abschrägung für den 3D-Text an.

Bei Textdarstellern handelt es sich hierbei um eine Darstellereigenschaft. Bei extrudiertem Text in einem 3D-Darsteller handelt es sich um eine Modellressourceneigenschaft.

Für die Eigenschaft `bevelType` gibt es folgende gültige Werte:

- `#none`
- `#miter` (Standard)
- `#round`

Beispiel

In diesem Beispiel ist der Darsteller "Logo“ ein Textdarsteller. Die folgende Anweisung setzt die Eigenschaft `bevelType` für den Darsteller „Logo“ auf `#round`.

```
member("logo").beveltype = #round
```

Im folgenden Beispiel ist die Modellressource des Modells "Slogan" extrudierter Text. Die folgende Anweisung setzt die Eigenschaft `bevelType` der Modellressource von „Slogan“ auf `#miter`.

```
member("scene").model("Slogan").resource.bevelType = #miter
```

Siehe auch

`bevelDepth`, `extrude3D`, `displayMode`

bgColor (Fenster)

Syntax

```
-- Lingo syntax
windowObjRef.bgColor

// JavaScript syntax
windowObjRef.bgColor;
```

Beschreibung

Diese Fenstereigenschaft bestimmt die Hintergrundfarbe eines Fensters. Lesen/Schreiben.

Das Einstellen der Eigenschaft `bgColor` hat die gleiche Wirkung wie das Einstellen der Farbe im Dialogfeld „Filmeigenschaften“.

Beispiel

Das folgende Beispiel stellt die Farbe des Fensters „Animals“ auf einen RGB-Wert ein.

```
-- Lingo syntax
window("Animals").bgColor = color(255, 153, 0)

// JavaScript syntax
window("Animals").bgColor = color(255, 153, 0);
```

Siehe auch

`Window`

bgColor (Sprite, 3D-Darsteller)

Syntax

```
sprite(whichSpriteNumber).bgColor
the bgColor of sprite whichSpriteNumber
the bgColor of the stage
(the stage).bgColor
member(which3dMember).bgcolor
```

Beschreibung

Diese Sprite-, System- und 3D-Darstellereigenschaft bestimmt die Hintergrundfarbe des in *whichSprite* angegebenen Sprites, die Farbe der Bühne bzw. die Hintergrundfarbe des 3D-Darstellers. Das Einstellen der Sprite-Eigenschaft `bgColor` hat die gleiche Wirkung wie das Auswählen der Hintergrundfarbe im Werkzeugfenster, wenn das Sprite auf der Bühne ausgewählt ist. Das Einstellen der Eigenschaft `bgColor` für die Bühne hat die gleiche Wirkung wie das Einstellen der Farbe im Dialogfeld „Filmeigenschaften“.

Diese Sprite-Eigenschaft hat die gleiche Funktion wie die Sprite-Eigenschaft `backColor`. Der zurückgegebene Farbwert ist jedoch ein Farbobjekt des Typs, der für das betreffende Sprite eingestellt wurde.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Das folgende Beispiel stellt die Bühnenfarbe auf einen RGB-Wert ein.

Punktsyntax:

```
(the stage).bgColor = rgb(255, 153, 0)
```

Ausführliche Lingo-Syntax:

```
set the bgColor of the stage = rgb(255, 153, 0)
```

Siehe auch

color(), backColor, backgroundColor

bias

Syntax

```
member(whichCastmember).model(whichModel).lod.bias
```

Beschreibung

Diese 3D-Eigenschaft für den Modifizierer `lod` bestimmt, wie aggressiv der Modifizierer Details aus dem Modell entfernt, wenn die Eigenschaft `auto` auf `TRUE` gesetzt ist. Diese Eigenschaft ist wirkungslos, wenn die Modifiziereigenschaft `auto` auf `FALSE` gesetzt ist.

Der Gültigkeitsbereich dieser Eigenschaft liegt zwischen 0.0 (alle Polygone entfernen) und 100.0 (keine Polygone entfernen). Der Standardwert ist 100.0.

Der Modifizierer `#lod` kann nur zu Modellen hinzugefügt werden, die außerhalb von Director in einem 3D-Modellierungsprogramm erstellt wurden. Der Wert der Eigenschaft `type` der von diesen Modellen verwendeten Modellressourcen lautet `#fromFile`. Der Modifizierer kann nicht zu in Director erstellten Primitiven hinzugefügt werden.

Beispiel

Die folgende Anweisung setzt die Eigenschaft bias des Modifizierers lod des Modells „Raumschiff“ auf 10. Wenn die Eigenschaft auto des Modifizierers lod auf TRUE gesetzt ist, reduziert der Modifizierer die im Modell enthaltenen Details sehr aggressiv, wenn sich das Raumschiff von der Kamera weg bewegt.

```
-- Lingo syntax
member("3D World").model("Spaceship").lod.bias = 10

// Java Script
member("3D World").getPropRef("model", 1).getPropRef("lod", 1).bias = 10;
```

Siehe auch

lod (Modifizierer), auto, level

bitDepth (Mixer)

Syntax

```
mixer.bitDepth (Read-write)
```

Beschreibung
Diese Mixereigenschaft gibt die Bit-Tiefe eines Mixers aus bzw. legt sie fest.

Beispiel
```
-- Lingo syntax
on mouseUp me
mixerRef.bitdepth = 16 -- Sets the bit depth of the mixer to 16.
end

// JavaScript syntax
function mouseUp(){
mixerRef.bitdepth = 16; // Sets the bit depth of the mixer to 16.
}
```

Siehe auch
Mixer

bitDepth (Soundobject)

Syntax
```
SoundObjectRef.bitDepth (Read-only)
```

Beschreibung
Diese Soundobjekteigenschaft gibt die Bit-Tiefe des Soundobjekts aus.

Beispiel
```
--Lingo syntax
on mouseUp me
put SoundObjectRef.bitdepth   -- Returns the bit depth of the sound object.
end

// JavaScript syntax
function mouseUp(){
put (SoundObjectRef.bitdepth); // Returns the bit depth of the sound object.
}
```

bitmapSizes

Syntax
```
-- Lingo syntax
memberObjRef.bitmapSizes

// JavaScript syntax
memberObjRef.bitmapSizes;
```

Beschreibung
Diese Schriftart-Darstellereigenschaft gibt eine Liste der Bitmap-Punktgrößen zurück, die beim Erstellen des Schriftartdarstellers eingebunden wurden.

Beispiel

Die folgende Anweisung zeigt die Bitmap-Punktgrößen an, die beim Erstellen von Darsteller 11 eingebunden wurden:

```
-- Lingo syntax
put(member(11).bitmapSizes)

// JavaScript syntax
put(member(11).bitmapSizes);
```

Siehe auch

recordFont, characterSet, originalFont

bitRate

Syntax

```
-- Lingo syntax
memberObjRef.bitRate

// JavaScript syntax
memberObjRef.bitRate;
```

Beschreibung

Diese Shockwave Audio (SWA)-Darstellereigenschaft gibt die Bitrate des angegebenen SWA-Darstellers, der vom Server vorausgeladen wurde, in Kilobit pro Sekunde (Kbit/s) zurück.

Die Darstellereigenschaft `bitRate` gibt so lange 0 an, bis das Streaming beginnt.

Beispiel

Das folgende Verhalten meldet die Bitrate eines SWA-Darstellers, wenn das Sprite zum ersten Mal angetroffen wird.

```
-- Lingo syntax
property spriteNum

on beginSprite (me)
    memName = sprite(spriteNum).member.name
    put("The bitRate of member"&&memName&&"is"&&member(memName).bitRate)
end

// JavaScript syntax
function beginSprite() {
    var memName = sprite(spriteNum).member.name;
    put("The bitRate of member " + memName +" is " + member(memName).bitRate);
}
```

bitsPerSample

Syntax

```
-- Lingo syntax
memberObjRef.bitsPerSample

// JavaScript syntax
memberObjRef.bitsPerSample;
```

Beschreibung

Diese Shockwave Audio (SWA)-Darstellereigenschaft gibt die Bittiefe der Originaldatei an, die für Shockwave Audio (SWA) kodiert wurde. Diese Eigenschaft ist erst verfügbar, nachdem die Wiedergabe des SWA-Sounds begonnen hat oder nachdem die Datei mithilfe des Befehls `preLoadBuffer` vorausgeladen wurde.

Diese Eigenschaft kann getestet, aber nicht eingestellt werden.

Beispiel

Die folgende Anweisung ordnet die Originalbitrate der Datei, die im SWA-Streaming-Darsteller "Paul Robeson“ verwendet wird, dem Felddarsteller "Tiefe“ zu.

```
-- Lingo syntax
member("How Deep").text = member("Paul Robeson").bitsPerSample

// JavaScript syntax
member("How Deep").text = member("Paul Robeson").bitsPerSample;
```

blend (3D)

Syntax

```
sprite(whichSprite).camera{(index)}.backdrop[index].blend
member(whichCastmember).camera(whichCamera).backdrop[index].blend
sprite(whichSprite).camera{(index)}.overlay[index].blend
member(whichCastmember).camera(whichCamera).overlay[index].blend
member(whichCastmember).shader(whichShader).blend
member(whichCastmember).model(whichModel).shader.blend
member(whichCastmember).model(whichModel).shaderList{[index]}.blend
```

Beschreibung

Diese für Hintergründe, Überlagerungen und den Shader `#standard` verwendete 3D-Eigenschaft gibt die Hintergrund-, Überlagerungs- bzw. Shader-Opazität an.

Bei einem Shader hat die Eigenschaft `blend` keinerlei Auswirkung, es sei denn, die Shader-Eigenschaft `transparent` ist auf `TRUE` gesetzt.

Der Gültigkeitsbereich dieser Eigenschaft liegt zwischen 100 und 100; die Standardeinstellung ist 0.

Beispiel

Die folgende Anweisung setzt die Eigenschaft blend des Shaders für das Modell „Fenster“ auf 80. Wenn die Shader-Eigenschaft transparent von „Fenster“ > „TRUE“ lautet, ist das Modell leicht durchsichtig.

```
-- Lingo syntax
member("House").model("Window").shader.blend = 80

// Java Script
member("House").getPropRef("model", 1).shaderList[1].blend = 80;
```

Siehe auch

bevelDepth, overlay, shadowPercentage, transparent

blend (Sprite)

Syntax

```
-- Lingo syntax
spriteObjRef.blend

// JavaScript syntax
spriteObjRef.blend;
```

Beschreibung

Diese Sprite-Eigenschaft gibt den Mischwert eines Sprites von 0 bis 100 zurück oder legt ihn fest, und zwar entsprechend den Mischwerten im Dialogfeld „Sprite-Eigenschaften“. Lesen/Schreiben.

Die möglichen Farben hängen von den in der Palette verfügbaren Farben ab, wobei die Farbtiefe des Bildschirms keine Rolle spielt.

Die besten Resultate erhalten Sie, wenn Sie den Farbeffekt „Mischen“ nur auf Grafiken anwenden, deren Farbtiefe größer als 8 Bit ist.

Beispiel

Die folgende Anweisung stellt den Mischwert für Sprite 3 auf 40 Prozent ein.

```
-- Lingo syntax
sprite(3).blend = 40

// JavaScript syntax
sprite(3).blend = 40;
```

Die folgende Anweisung zeigt den Mischwert von Sprite 3 im Nachrichtenfenster an:

```
-- Lingo syntax
put(sprite(3).blend)

// JavaScript syntax
put(sprite(3).blend);
```

Siehe auch

blendLevel, Sprite

blendConstant

Syntax

```
member(whichCastmember).shader(whichShader).blendConstant
member(whichCastmember).model(whichModel).shader.blendConstant
member(whichCastmember).model(whichModel).shaderList{[index]}.blendConstant
```

Beschreibung

Diese 3D-Eigenschaft für den Shader `#standard` gibt das Mischungsverhältnis für die erste Texturebene des Shaders an.

Wenn die Shader-Eigenschaft `useDiffuseWithTexture` auf `TRUE` gesetzt ist, vermischt sich die Textur mit der durch die Shader-Eigenschaft `diffuse` festgelegten Farbe. Wenn `useDiffuseWithTextureFALSE` lautet, wird zum Mischen Weiß verwendet.

Jede weitere Texturebene wird mit der darunter liegenden Texturebene vermischt. Mit der Eigenschaft `blendConstantList` lässt sich die Mischung in diesen Texturebenen steuern.

Die Eigenschaft `blendConstant` funktioniert nur dann, wenn die Shader-Eigenschaft `blendSource` auf `#constant` gesetzt ist. Weitere Informationen hierzu finden Sie unter `blendSource` und `blendSourceList`.

Der Gültigkeitsbereich dieser Eigenschaft liegt zwischen 0 und 100; die Standardeinstellung ist 50.

Beispiel

In diesem Beispiel enthält die Shader-Liste des Modells `MysteryBox` sechs Shader. Die folgende Anweisung setzt die Eigenschaft `blendConstant` des zweiten Shaders auf 20. Diese Eigenschaft wird durch die Einstellungen für die Eigenschaften `blendFunction`, `blendFunctionList`, `blendSource` und `blendSourceList` beeinflusst.

```
member("Level2").model("MysteryBox").shaderList[2].blendConstant = 20
```

Siehe auch

`blendConstantList`, `blendFunction`, `blendFunctionList`, `blendSource`, `blendSourceList`, `useDiffuseWithTexture`, `diffuse`, `diffuseColor`

blendConstantList

Syntax

```
member(whichCastmember).shader(whichShader).blendConstantList
member(whichCastmember).model(whichModel).shader.blendConstantList{[index]}
member(whichCastmember).model(whichModel).shaderList{[index]}.blendConstantList{[index]}
```

Beschreibung

Diese 3D-Eigenschaft für den Shader `#standard` gibt das Verhältnis zum Vermischen einer Texturebene des Shaders mit der darunter liegenden Texturebene an.

Die Texturliste des Shaders und die Mischungskonstantenliste weisen jeweils acht Indexpositionen auf. Jede Indexposition in der Mischungskonstantenliste steuert die Mischung für die Textur, die sich in der Texturliste an der gleichen Indexposition befindet. Um für alle Indexpositionen in der Liste den gleichen Wert zu verwenden, lassen Sie den optionalen Parameter *index* weg. Mit dem Parameter *index* können Sie jede Indexposition in der Liste einzeln definieren.

Die Eigenschaft `blendConstantList` funktioniert nur dann, wenn die Eigenschaft `blendSource` der entsprechenden Texturebene auf `#constant` gesetzt ist.

Der Gültigkeitsbereich dieser Eigenschaft liegt zwischen 0 und 100; die Standardeinstellung ist 50.

Beispiel

In diesem Beispiel enthält die Shader-Liste des Modells `MysteryBox` sechs Shader. Die folgende Anweisung zeigt die Eigenschaft `blendConstant` aller vom zweiten Shader verwendeten Texturen an. Diese Eigenschaft wird durch die Einstellungen für die Eigenschaften `blendFunction`, `blendFunctionList`, `blendSource` und `blendSourceList` beeinflusst.

```
-- Lingo syntax
put member("Level2").model("MysteryBox").shaderList[2].blendConstantList
-- [20.0000, 50.0000, 50.0000, 50.0000, 20.0000, 50.0000, 50.0000, 50.0000]

// JavaScript syntax
put(member("Level2").getPropRef("model",3).shaderList[1].blendConstantList);
// <[20.0000, 50.0000, 50.0000, 50.0000, 20.0000, 50.0000, 50.0000, 50.0000]>
put member("Level2").model("MysteryBox").shaderList[2].blendConstantList
-- [20.0000, 50.0000, 50.0000, 50.0000, 20.0000, 50.0000, 50.0000, 50.0000]
```

Siehe auch

`blendConstant`, `blendFunction`, `blendFunctionList`, `blendSource`, `blendSourceList`, `useDiffuseWithTexture`, `diffuse`, `diffuseColor`

blendFactor

Syntax

```
member(whichCastmember).model(whichModel).keyframePlayer.blendFactor
member(whichCastmember).model(whichModel).bonesPlayer.blendFactor
```

Beschreibung

Diese 3D-Eigenschaft für die Modifizierer `keyframePlayer` und `bonesPlayer` bestimmt, um welchen Faktor eine Bewegung mit der vorausgehenden Bewegung kombiniert wird.

Der Gültigkeitsbereich dieser Eigenschaft liegt zwischen 0 und 100; die Standardeinstellung ist 0.

`BlendFactor` wird nur verwendet, wenn die Modifiziereigenschaft `autoblend` auf `FALSE` gesetzt ist. Wenn der Wert der Eigenschaft `blendFactor` 100 lautet, weist die aktuelle Bewegung keine Merkmale der vorausgehenden Bewegung auf. Wenn der Wert der Eigenschaft `blendFactor` 0 lautet, weist die aktuelle Bewegung alle Merkmale der vorausgehenden Bewegung und keine eigenen auf. Wenn der Wert der Eigenschaft `blendFactor` 50 lautet, setzt sich die aktuelle Bewegung in gleichem Maße aus eigenen Merkmalen und aus den Merkmalen der vorausgehenden Bewegung zusammen. Der Wert von `blendFactor` kann über eine bestimmte Zeit hinweg variiert werden, um Übergänge zu erstellen, die ganz anders sind als der lineare Übergang, der entsteht, wenn die Modifiziereigenschaft `autoblend` auf `TRUE` gesetzt wird.

Beispiel

Die folgende Anweisung setzt die Eigenschaft `blendFactor` des Modells „Kreatur3“ auf 50. Wenn die Modifiziereigenschaft autoblend FALSE lautet, setzt sich jede Bewegung in der Abspielliste des `keyframePlayer` für „Kreatur3“ im gleichen Verhältnis aus sich selbst und der vorhergehenden Bewegung zusammen.

```
member("newaliens").model("Alien3").keyframePlayer.blendFactor = 50
```

Siehe auch

autoblend, keyframePlayer (Modifizierer)

blendFunction

Syntax

```
member(whichCastmember).shader(whichShader).blendFunction
member(whichCastmember).model(whichModel).shader.blendFunction
member(whichCastmember).model(whichModel).shaderList{[index]}.blendFunction
```

Beschreibung

Diese 3D-Eigenschaft für den Shader `#standard` gibt die Mischungsart für die erste Texturebene des Shaders an.

Wenn die Shader-Eigenschaft `useDiffuseWithTexture` auf `TRUE` gesetzt ist, vermischt sich die Textur mit der durch die Shader-Eigenschaft `diffuse` festgelegten Farbe. Wenn `useDiffuseWithTextureFALSE` lautet, wird zum Mischen Weiß verwendet.

Jede weitere Texturebene wird mit der darunter liegenden Texturebene vermischt. Mit der Eigenschaft `blendFunctionList` lässt sich die Mischung in diesen Texturebenen steuern.

Die Eigenschaft `blendFunction` kann folgende Werte aufweisen:

`#multiply` multipliziert die RGB-Werte der Texturebene mit der zum Mischen verwendeten Farbe (siehe oben).

`#add` fügt die RGB-Werte der Texturebene zu der zum Mischen verwendeten Farbe hinzu (Höchstwert: 255).

`#replace` verhindert, dass die Textur mit der durch die Shader-Eigenschaft `diffuse` festgelegten Farbe vermischt wird.

`#blend` kombiniert die Farben der Texturebene mit der zum Mischen verwendeten Farbe in dem durch die Eigenschaft `blendConstant` bestimmten Verhältnis.

Der Standardwert dieser Eigenschaft lautet `#multiply`.

Beispiel

In diesem Beispiel enthält die Shader-Liste des Modells `MysteryBox` sechs Shader. Die folgende Anweisung setzt die Eigenschaft `blendFunction` des zweiten Shaders auf `#blend`. Dadurch werden die Einstellungen für die Eigenschaften `blendSource`,`blendSourceList`, `blendConstant` und `blendConstantList` aktiviert.

```
member("Level2").model("MysteryBox").shaderList[2].blendFunction = #blend
```

Siehe auch

blendConstant, blendConstantList, blendFunctionList, blendSource, blendSourceList, useDiffuseWithTexture, diffuse, diffuseColor

blendFunctionList

Syntax

```
member(whichCastmember).shader(whichShader).blendFunctionList{[index]}
member(whichCastmember).model(whichModel).shader.blendFunctionList{[index]}
member(whichCastmember).model(whichModel).shaderList{[index]}.blendFunctionList{[index]}
```

Beschreibung

Diese 3D-Eigenschaft für den Shader `#standard` gibt eine lineare Liste an, aus der hervorgeht, wie jede Texturebene mit der darunter liegenden Texturebene vermischt wird.

Die Texturliste des Shaders und die Mischungsfunktionsliste weisen jeweils acht Indexpositionen auf. Jede Indexposition in der Mischungsfunktionsliste steuert die Mischung für die Textur, die sich in der Texturliste an der gleichen Indexposition befindet. Um für alle Indexpositionen in der Liste den gleichen Wert zu verwenden, lassen Sie den optionalen Parameter *index* weg. Mit dem Parameter *index* können Sie jede Indexposition in der Liste einzeln definieren.

Jede Indexposition in der Mischungsfunktionsliste kann einen der folgenden Werte aufweisen:

`#multiply` multipliziert die RGB-Werte der Texturebene mit den RGB-Werten der darunter liegenden Texturebene.

`#add` fügt die RGB-Werte der Texturebene zu den RGB-Werten der darunter liegenden Texturebene hinzu (Höchstwert: 255).

`#replace` bewirkt, dass die Textur die darunter liegende Textur bedeckt. Eine Mischung findet nicht statt.

`#blend` bewirkt, dass die Mischung durch den Wert der Eigenschaft `blendSource`, die eine Alphamischung zulässt, gesteuert wird.

Der Standardwert dieser Eigenschaft lautet `#multiply`.

Beispiel

In diesem Beispiel enthält die Eigenschaft `shaderList` des Modells `MysteryBox` sechs Shader. Die folgende Anweisung zeigt, dass der Wert der vierten Indexposition der Eigenschaft `blendFunctionList` des zweiten Shaders auf `#blend` gesetzt ist. Die Mischung der vierten Texturebene des zweiten Shaders des Modells wird durch die Einstellungen der Eigenschaften `blendSource`, `blendSourceList`, `blendConstant`, `blendConstantList`, diffuse, `diffuseColor` und `useDiffuseWithTexture` gesteuert.

```
put member("Level2").model("MysteryBox").shaderList[2].blendFunctionList[4]
-- #blend
```

Siehe auch

`blendConstant`, `blendConstantList`, `blendFunction`, `blendSource`, `blendSourceList`, `diffuse`, `diffuseColor`, `useDiffuseWithTexture`

blendLevel

Syntax

```
sprite(whichSpriteNumber).blendLevel
the blendLevel of sprite whichSpriteNumber
```

Beschreibung

Mit dieser Sprite-Eigenschaft kann der aktuelle Mischungsgrad eines Sprites eingestellt oder aufgerufen werden. Die möglichen Werte liegen zwischen 0 und 255. Damit unterscheidet sich diese Eigenschaft vom Sprite-Inspektor, der Werte im Bereich von 0 bis 100 anzeigt. Das Ergebnis ist jedoch das Gleiche, nur die Skalierung ist anders.

Diese Eigenschaft ist mit der Sprite-Eigenschaft `blend` identisch.

Beispiel

```
sprite(3).blendlevel = 99
```

Siehe auch

`blend (Sprite)`

blendRange

Syntax

```
member(whichCastmember).modelResource(whichModelResource).blendRange.start
modelResourceObjectReference.blendRange.end
member(whichCastmember).modelResource(whichModelResource).blendRange.start
modelResourceObjectReference.blendRange.end
```

Beschreibung

Mit dieser 3D-Eigenschaft können Sie bei einer Modellressource vom Typ `#particle` den Anfang und das Ende des Mischungsbereichs der Modellressource ermitteln und festlegen.

Die Opazität der Partikel im System wird über die Lebensdauer jedes Partikels hinweg linear zwischen `blendRange.start` und `blendRange.end` interpoliert.

Der Wert dieser Eigenschaft muss zwischen (einschließlich) 0.0 und 100.0 liegen. Der Standardwert dieser Eigenschaft ist 100.0.

Beispiel

Die folgende Anweisung legt die `blendRange`-Eigenschaften der Modellressource `ThermoSystem` vom Typ `#particle` fest.

Die erste Zeile setzt den Startwert auf 100, die zweite Zeile den Endwert auf 0. Diese Anweisung bewirkt, dass die Partikel der Ressource `ThermoSystem` anfänglich völlig undurchsichtig sind und dann über ihre Lebensdauer hinweg allmählich durchsichtig werden.

```
member("Heater").modelResource("ThermoSystem").blendRange.start = 100.0
member("Heater").modelResource("ThermoSystem").blendRange.end = 0.0
```

blendSource

Syntax

```
member(whichCastmember).shader(whichShader).blendSource
member(whichCastmember).model(whichModel).shader.blendSource
member(whichCastmember).model(whichModel).shaderList{[index]}.blendSource
```

Beschreibung

Diese 3D-Eigenschaft für den Shader `#standard` gibt an, ob die Mischung der ersten Texturebene in der Texturliste des Shaders auf den Alphainformationen der Textur oder auf einem konstanten Verhältnis basiert.

Wenn die Shader-Eigenschaft `useDiffuseWithTexture` auf `TRUE` gesetzt ist, vermischt sich die Textur mit der durch die Shader-Eigenschaft `diffuse` festgelegten Farbe. Wenn `useDiffuseWithTextureFALSE` lautet, wird zum Mischen Weiß verwendet.

Jede weitere Texturebene wird mit der darunter liegenden Texturebene vermischt. Mit der Eigenschaft `blendSourceList` lässt sich die Mischung in diesen Texturebenen steuern.

Die Eigenschaft `blendSource` funktioniert nur dann, wenn die Shader-Eigenschaft `blendFunction` auf `#blend` gesetzt ist.

Mögliche Werte für diese Eigenschaft:

`#alpha` bewirkt, dass sich das Verhältnis zur Mischung der einzelnen Pixel in der Textur mit der zum Mischen verwendeten Farbe nach den Alphainformationen in der Textur richtet (siehe oben).

`#constant` bewirkt, dass als Mischungsverhältnis für alle Pixel der Textur der Wert der Shader-Eigenschaft `blendConstant` verwendet wird.

Der Standardwert dieser Eigenschaft lautet `#constant`.

Beispiel

In diesem Beispiel enthält die Shader-Liste des Modells `MysteryBox` sechs Shader. Die folgende Anweisung setzt die Eigenschaft `blendSource` der ersten vom zweiten Shader verwendeten Textur auf `#constant`. Dadurch werden die Einstellungen für die Eigenschaften `blendConstant` und `blendConstantList` aktiviert.

```
-- Lingo syntax
member("Level2").model("MysteryBox").shaderList[2].blendSource = #constant

// JavaScript syntax
member("Level2").getPropRef("model",3).shaderList[2].blendSource = symbol("constant");
```

Siehe auch

`blendSourceList`, `blendFunction`, `blendFunctionList`, `blendConstant`, `blendConstantList`, `useDiffuseWithTexture`, `diffuse`, `diffuseColor`

blendSourceList

Syntax

```
member(whichCastmember).shader(whichShader).blendSourceList[index]
member(whichCastmember).model(whichModel).shader.blendSourceList{[index]}
member(whichCastmember).model(whichModel).shaderList{[index]}.blendSourceList{[index]}
```

Beschreibung

Diese 3D-Eigenschaft für den Shader `#standard` gibt an, ob die Mischung einer Texturebene mit den darunter liegenden Texturebenen auf den Alphainformationen der Textur oder auf einem konstanten Verhältnis basiert.

Die Texturliste des Shaders und die Mischungsquellenliste weisen jeweils acht Indexpositionen auf. Jede Indexposition in der Mischungsquellenliste steuert die Mischung für die Textur, die sich in der Texturliste an der gleichen Indexposition befindet. Um für alle Indexpositionen in der Liste den gleichen Wert zu verwenden, lassen Sie den optionalen Parameter *index* weg. Mit dem Parameter *index* können Sie jede Indexposition in der Liste einzeln definieren.

Die Eigenschaft `blendSourceList` funktioniert nur dann, wenn die Eigenschaft `blendFunction` der entsprechenden Texturebene auf `#blend` gesetzt ist. Weitere Informationen finden Sie unter `blendFunction` und `blendFunctionList`.

Mögliche Werte für diese Eigenschaft:

`#alpha` bewirkt, dass sich das Verhältnis zur Mischung der einzelnen Pixel der Texturebene mit der darunter liegenden Ebene nach den Alphainformationen in der Textur richtet.

`#constant` bewirkt, dass als Mischungsverhältnis für alle Pixel der Texturebene der Wert der Eigenschaft `blendConstant` der entsprechenden Texturebene verwendet wird. Weitere Informationen finden Sie unter `blendConstant` und `blendConstantList`.

Der Standardwert dieser Eigenschaft lautet `#constant`.

Beispiel

In diesem Beispiel enthält die Shader-Liste des Modells `MysteryBox` sechs Shader. Jeder Shader verfügt über eine Texturliste mit bis zu acht Texturen. Die folgende Anweisung zeigt, dass die Eigenschaft `blendSource` der vierten vom zweiten Shader verwendeten Textur auf `#constant` gesetzt ist. Dadurch werden die Einstellungen für die Eigenschaften `blendConstant`, `blendConstantList` und `useDiffuseWithTexture` aktiviert.

```
member("Level2").model("MysteryBox").shaderList[2].blendSourceList[4] = #constant
```

Siehe auch

`blendSource`, `blendFunction`, `blendFunctionList`, `blendConstant`, `blendConstantList`, `useDiffuseWithTexture`, `diffuse`, `diffuseColor`

blendTime

Syntax

```
member(whichCastmember).model(whichModel).keyframePlayer.blendTime
member(whichCastmember).model(whichModel).bonesPlayer.blendTime
```

Beschreibung

Diese 3D-Eigenschaft für die Modifizierer `keyframePlayer` und `bonesPlayer` gibt die Dauer des Übergangs zwischen Bewegungen in der Abspielliste des für das Modell verwendeten Modifizierers in Millisekunden an.

Die Eigenschaft `blendTime` funktioniert in Verbindung mit der Modifiziereigenschaft `autoBlend`. Wenn `autoBlend` auf `TRUE` gesetzt ist, erzeugt der Modifizierer einen linearen Übergang auf die aktuelle Bewegung des Modells von der vorhergehenden Bewegung. Der Wert der Eigenschaft `blendTime` entspricht der Länge dieses Übergangs. Die Eigenschaft `blendTime` wird ignoriert, wenn `autoBlend` auf `FALSE` gesetzt ist.

Der Standardwert dieser Eigenschaft lautet 500.

Beispiel

Die folgende Anweisung stellt die Dauer des Übergangs zwischen Bewegungen in der Abspielliste des Modifizierers für das Modell "Kreatur5" auf 1200 Millisekunden ein.

```
member("newaliens").model("Alien5").keyframePlayer.blendTime = 1200
```

Siehe auch

autoblend, blendFactor

bone

Syntax

```
-- Lingo Usage
member(whichCastmember).modelResource(whichModelResource).bone.count
member(whichCastmember).model(whichModel).bonesPlayer.bone[index].transform
member(whichCastmember).model(whichModel).bonesPlayer.bone[index].worldTransform

// JavaScript Usage
member(whichCastmember).getProp("model",
whichModelIndex).getPropRef("bonesplayer").getPropRef("bone", whichBoneIndex).transform
member(whichCastmember).getProp("model",
whichModelIndex).getPropRef("bonesplayer").getPropRef("bone", whichBoneIndex).worldTransform
```

Beschreibung

3D-Element; ein Knochen ist ein Strukturelement einer in einem 3D-Modellierungsprogramm erstellten Modellressource. Knochen können in Director weder erstellt noch gelöscht oder anders angeordnet werden.

`#bones`-Bewegungen, die ebenfalls in einem 3D-Modellierungsprogramm geskriptet werden müssen, wirken sich auf die Knochenstruktur einer Modellressource aus und werden in Director mit dem Modifizierer `bonesPlayer` verwaltet.

Siehe auch

count (3D), bonesPlayer (Modifizierer), transform (Eigenschaft), worldTransform

bonesPlayer (Modifizierer)

Syntax

```
member(whichCastmember).model(whichModel).bonesPlayer.whichBonesPlayerProperty
```

Beschreibung

Dieser 3D-Modifizierer verwaltet die Verwendung von Bewegungen in Modellen. Die mit dem Modifizierer `bonesPlayer` verwalteten Bewegungen animieren Abschnitte des Modells, die als "Knochen“ bezeichnet werden.

Bewegungen und die Modelle, die sie verwenden, müssen in einem 3D-Modellierungsprogramm erstellt, als W3D-Dateien exportiert und dann in einen Film importiert werden. Auf in Director erstellte Modellprimitive können keine Bewegungen angewendet werden.

Wenn Sie den Modifizierer `bonesPlayer` mit dem Befehl `addModifier` zu einem Modell hinzufügen, können Sie auf die folgenden Eigenschaften des Modifizierers `bonesPlayer` zugreifen:

`playing (3D)` gibt an, ob ein Modell eine Bewegung ausführt.

`playlist` ist eine lineare Liste mit Eigenschaftslisten, die die Abspielparameter der für ein Modell vorgesehenen Bewegungen enthalten.

`currentTime (3D)` gibt die lokale Zeit der gegenwärtig abgespielten oder angehaltenen Bewegung in Millisekunden an.

`playRate (3D)` ist eine Zahl, die mit dem Parameter *scale* des Befehls `play()` oder `queue()` multipliziert wird, um die Wiedergabegeschwindigkeit der Bewegung zu bestimmen.

`playlist.count (3D)` gibt die Anzahl von Bewegungen zurück, die sich gegenwärtig in der Abspielliste befinden.

`rootLock` gibt an, ob die Translationskomponente der Bewegung verwendet oder ignoriert wird.

`currentLoopState` gibt an, ob die Bewegung einmal abgespielt oder kontinuierlich wiederholt wird.

`blendTime` gibt die Dauer des durch den Modifizierer erzeugten Übergangs zwischen den einzelnen Bewegungen an, wenn die Eigenschaft `autoblend` des Modifizierers auf `TRUE` gesetzt ist.

`autoblend` gibt an, ob der Modifizierer einen linearen Übergang auf die aktuelle Bewegung von der vorhergehenden Bewegung erzeugt.

`blendFactor` gibt den Grad der Überblendung zwischen den einzelnen Bewegungen an, wenn die Eigenschaft `autoBlend` des Modifizierers auf `FALSE` gesetzt ist.

`bone[boneId].transform` gibt die auf den Parent-Knochen bezogene Transformation des Knochens an. Um den `boneId`-Wert zu ermitteln, testen Sie die Eigenschaft `getBoneID` der Modellressource. Wenn Sie die Transformation des Knochens selbst festlegen, wird diese nicht mehr durch die aktuelle Bewegung gesteuert. Die manuelle Steuerung endet bei Ablauf der aktuellen Bewegung.

`bone[boneId].getWorldTransform` gibt die auf die Welt bezogene Transformation des Knochens zurück.

`lockTranslation` gibt an, ob das Modell von den angegebenen Ebenen verdrängt werden kann.

`positionReset` gibt an, ob das Modell nach Ablauf einer Bewegung oder nach jedem Durchlauf einer Schleife an seine Ausgangsposition zurückkehrt.

`rotationReset` gibt das Rotationselement eines Übergangs von einer Bewegung zur nächsten bzw. den Durchlauf einer Einzelbewegung an.

Hinweis: *Weitere Informationen zu diesen Eigenschaften finden Sie in den einzelnen Beschreibungen.*

Der Modifizierer `bonesPlayer` verwendet folgende Befehle:

`pause() (3D)` hält die Bewegung an, die gegenwärtig vom Modell ausgeführt wird.

`play() (3D)` leitet die Ausführung einer Bewegung ein bzw. setzt sie fort.

`playNext() (3D)` leitet die Wiedergabe der nächsten Bewegung in der Abspielliste ein.

`queue() (3D)` fügt eine Bewegung am Ende der Abspielliste hinzu.

Der Modifizierer `bonesPlayer` generiert die folgenden Ereignisse, die von in den Befehlen `registerForEvent()` und `registerScript()` deklarierten Prozeduren verwendet werden. Der Aufruf der deklarierten Prozedur enthält drei Argumente: den Ereignistyp(entweder `#animationStarted` oder `#animationEnded`), den Namen der Bewegung und die aktuelle Zeit der Bewegung. Weitere Informationen über Benachrichtigungsereignisse finden Sie unter `registerForEvent()`.

`#animationStarted` wird gesendet, wenn die Wiedergabe einer Bewegung beginnt. Wenn zwischen den einzelnen Bewegungen eine Überblendung stattfindet, wird das Ereignis am Anfang des Übergangs gesendet.

`#animationEnded` wird am Ende einer Bewegung gesendet. Wenn zwischen den einzelnen Bewegungen eine Überblendung stattfindet, wird das Ereignis am Ende des Übergangs gesendet.

Siehe auch

`keyframePlayer (Modifizierer)`, `addModifier`, `modifiers`, `modifier`

border

Syntax

```
-- Lingo syntax
memberObjRef.border

// JavaScript syntax
memberObjRef.border;
```

Beschreibung

Diese Felddarstellereigenschaft gibt die Breite des Rahmens um den angegebenen Felddarsteller in Pixel an.

Beispiel

Die folgende Anweisung stellt die Breite des Rahmens um den Felddarsteller "Titel“ auf 10 Pixel ein.

```
--Lingo syntax
member("Title").border = 10

// JavaScript syntax
member("Title").border = 10;
```

bottom

Syntax

```
-- Lingo syntax
spriteObjRef.bottom

// JavaScript syntax
spriteObjRef.bottom;
```

Beschreibung

Diese Sprite-Eigenschaft bestimmt die untere vertikale Koordinate des Begrenzungsrechtecks eines Sprites. Lesen/Schreiben.

Beispiel

Die folgende Anweisung ordnet die vertikale Koordinate am unteren Rand des mit (i + 1) nummerierten Sprites der Variablen lowest zu.

```
-- Lingo syntax
lowest = sprite(i + 1).bottom

// JavaScript syntax
var lowest = sprite(i + 1).bottom;
```

Siehe auch

Sprite

bottom (3D)

Syntax

```
member(whichCastmember).modelResource(whichModelResource).bottom
```

Beschreibung

Diese 3D-Eigenschaft für die Modellressource `#box` gibt an, ob die Seite der Box, die von der -Y-Achse geschnitten wird, geschlossen (`TRUE`) oder offen ist (`FALSE`).

Der Standardwert dieser Eigenschaft lautet `TRUE`.

Beispiel

Die folgende Anweisung setzt die Eigenschaft „bottom“ der Modellressource `GiftBox` auf TRUE, d. h. die Unterseite der Box ist geschlossen.

```
-- Lingo syntax
nmr = member("3D World").newModelresource("GiftBox", #box)
member("3D World").modelResource("GiftBox").bottom = TRUE

// JavaScript syntax
nmr = member("3D World").newModelresource("GiftBox",symbol("box"));
member("3D World").getProp("modelresource",10).bottom = true;
```

Siehe auch

back, front, top (3D), left (3D), right (3D), bottomCap

bottomCap

Syntax

```
member(whichCastmember).modelResource(whichModelResource).bottomCap
```

Beschreibung

Diese 3D-Eigenschaft für die Modellressource `#cylinder` gibt an, ob das Ende des Zylinders, das von der -Y-Achse geschnitten wird, geschlossen (`TRUE`) oder offen ist (`FALSE`).

Der Standardwert dieser Eigenschaft lautet `TRUE`.

Beispiel

Die folgende Anweisung setzt die Eigenschaft `bottomCap` der Modellressource „Cylinder11“ auf „FALSE“, d. h. die Unterseite des Zylinders ist offen.

```
-- Lingo syntax
nmr = member("3D World").newModelresource("Cylinder11", #cylinder);
member("3D World").modelResource("Cylinder11").bottomCap = FALSE

// JavaScript syntax
nmr = member("3D World").newModelresource("Cylinder11", symbol("cylinder"));
member("3D World").getPropRef("modelResource", 11).bottomCap = false;
```

Siehe auch

topCap, bottomRadius, bottom (3D)

bottomRadius

Syntax

```
member(whichCastmember).modelResource(whichModelResource).bottomRadius
```

Beschreibung

Diese 3D-Eigenschaft für die Modellressource `#cylinder` gibt den Radius des Zylinderendes, das von der -Y-Achse geschnitten wird, in Welteinheiten an.

Der Standardwert dieser Eigenschaft lautet 25.

Beispiel

Die folgende Anweisung setzt die Eigenschaft `bottomRadius` der Modellressource "Rohr" auf 38.5.

```
member("3D World").modelResource("Tube").bottomRadius = 38.5
```

Siehe auch

topRadius, bottomCap

bottomSpacing

Syntax

```
-- Lingo syntax
chunkExpression.bottomSpacing

// JavaScript syntax
chunkExpression.bottomSpacing;
```

Beschreibung

Mit dieser Textdarsteller-Eigenschaft können zusätzliche Abstände, die am unteren Ende jedes Absatzes angewendet werden, im *chunkExpression*-Abschnitt des Textdarstellers festgelegt werden.

Der Wert selbst ist ein ganzzahliger Wert, wobei Werte unter 0 weniger und Werte über 0 mehr Abstand zwischen den Absätzen bewirken.

Der Standardwert ist 0. Dadurch werden Absätze durch den Standardabstand getrennt.

Hinweis: *Diese Eigenschaft unterstützt wie alle Textdarstellereigenschaften nur die Punktsyntax.*

Beispiel

Die folgende Anweisung fügt nach dem ersten Absatz im Darsteller "Nachrichtenmeldungen“ einen Abstand ein.

```
--Lingo syntax
member("News Items").paragraph[1].bottomSpacing=20

// JavaScript syntax
member("News Items").getPropRef("paragraph", 1).bottomSpacing=20;
```

Siehe auch

top (3D)

boundary

Syntax

```
member(whichCastmember).model(whichModel).inker.boundary
member(whichCastmember).model(whichModel).toon.boundary
```

Beschreibung

Diese 3D-Eigenschaft der Modifizierer `inker` und `toon` bestimmt, ob an den Kanten eines Modells eine Linie gezeichnet werden soll.

Die Standardeinstellung dieser Eigenschaft lautet `TRUE`.

Beispiel

Die folgende Anweisung setzt die Eigenschaft boundary des auf das Modell "Box" angewendeten Modifizierers "inker" auf TRUE. An den Kanten der Modelloberfläche werden Linien gezeichnet.

```
-- Lingo syntax
member("shapes").model("Box").addModifier(#inker)
member("shapes").model("Box").inker.boundary = TRUE

// JavaScript syntax
member("shapes").getProp("model",1).addModifier(symbol("inker"));
member("shapes").getProp("model",1).getPropRef("inker").boundary = true;
```

Siehe auch

lineColor, lineOffset, silhouettes, creases

boundingSphere

Syntax

```
member(whichCastmember).model(whichModel).boundingSphere
member(whichCastmember).group(whichGroup).boundingSphere
member(whichCastmember).light(whichLight).boundingSphere
member(whichCastmember).camera(whichCamera).boundingSphere
```

Beschreibung

Diese 3D-Eigenschaft für Modelle, Gruppen, Lichtquellen und Kameras beschreibt eine Kugel, die das Modell, die Gruppe, das Licht bzw. die Kamera samt Child-Objekten enthält.

Der Wert dieser Eigenschaft ist eine Liste mit der Vektorposition der Kugelmitte und der als Fließkommazahl ausgedrückten Länge des Kugelradius.

Diese Eigenschaft kann getestet, aber nicht eingestellt werden.

Beispiel

Das folgende Beispiel zeigt die Begrenzungskugel eines Lichts im Nachrichtenfenster an:

```
-- Lingo syntax
put member("newAlien").light[5].boundingSphere
-- [vector(166.8667, -549.6362, 699.5773), 1111.0039]

// JavaScript syntax
put(member("newAlien").getProp("light",5).boundingSphere);
// <[vector(166.8667, -549.6362, 699.5773), 1111.0039]>
```

Siehe auch

debug

boxDropShadow

Syntax

```
-- Lingo syntax
memberObjRef.boxDropShadow

// JavaScript syntax
memberObjRef.boxDropShadow;
```

Beschreibung

Diese Darstellereigenschaft bestimmt die Größe des Schlagschattens um die Box des durch *whichCastMember* angegebenen Felddarstellers in Pixel.

Beispiel

Die folgende Anweisung stellt die Breite des Schlagschattens des Felddarstellers "Titel“ auf 10 Pixel ein.

```
--Lingo syntax
member("Title").boxDropShadow = 10

// JavaScript syntax
member("Title").boxDropShadow = 10;
```

boxType

Syntax

```
-- Lingo syntax
memberObjRef.boxType

// JavaScript syntax
memberObjRef.boxType;
```

Beschreibung

Diese Darstellereigenschaft bestimmt den Typ des Textfeldes, der für den angegebenen Darsteller verwendet wird. Die möglichen Werte sind `#adjust`, `#scroll`, `#fixed` und `#limit`.

Beispiel

<Check Alignment of PHs>Die folgende Anweisung macht das Feld des Felddarstellers „Leitartikel“ zu einem rollbaren Feld.

```
--Lingo syntax
member("Editorial").boxType = #scroll

// JavaScript syntax
member("Editorial").boxType = symbol("scroll");
```

brightness

Syntax

```
member(whichCastmember).shader(whichShader).brightness
member(whichCastmember).model(whichModel).shader.brightness
member(whichCastmember).model(whichModel).shaderList{[index]}.brightness
```

Beschreibung

Diese 3D-Eigenschaft für die Shader `#newsprint` und `#engraver` gibt an, wie viel Weiß mit dem Shader gemischt werden soll.

Der Gültigkeitsbereich dieser Eigenschaft liegt zwischen 1 und 100; der Standardwert ist 0.

Beispiel

Die folgende Anweisung setzt die Helligkeit des vom Modell `gbCyl2` verwendeten Shaders auf die Hälfte des Höchstwerts:

```
-- Lingo syntax
member("scene").model("gbCyl2").shader.brightness = 50

// JavaScript syntax
member("scene").getProp("shader",1).brightness = 50;
```

Siehe auch

newShader

broadcastProps

Syntax

```
-- Lingo syntax
memberObjRef.broadcastProps

// JavaScript syntax
memberObjRef.broadcastProps;
```

Beschreibung

Diese Darstellereigenschaft bestimmt, ob die an einem Flash- oder Vektorformdarsteller vorgenommenen Änderungen sofort an alle seine Sprites, die gegenwärtig auf der Bühne sind, gesendet werden (TRUE) oder nicht (FALSE).

Wenn diese Eigenschaft auf FALSE eingestellt ist, werden die am Darsteller vorgenommenen Änderungen nur als Standard für neue Sprites verwendet. Sie haben keinen Einfluss auf die Sprites auf der Bühne.

Der Standardwert dieser Eigenschaft ist TRUE. Sie kann getestet und eingestellt werden.

Beispiel

Im folgenden Bildskript wird davon ausgegangen, dass die Eigenschaft broadcastProps des Flash-Filmdarstellers „Navigationsfilm“ auf FALSE eingestellt ist. Das Skript erlaubt vorübergehend, dass die an einem Flash-Filmdarsteller vorgenommenen Änderungen an seine Sprites, die zurzeit auf der Bühne sind, übertragen werden. Dann stellt das Skript die Eigenschaft viewScale des Flash-Filmdarstellers ein, und diese Änderung wird an sein Sprite übertragen. Danach hindert das Skript den Flash-Film daran, Änderungen an seine Sprites zu übertragen.

```
-- Lingo syntax
on enterFrame
    member("Navigation Movie").broadcastProps = TRUE
    member("Navigation Movie").viewScale = 200
    member("Navigation Movie").broadcastProps = FALSE
end

// JavaScript syntax
function enterFrame() {
    member("Navigation Movie").broadcastProps = 1;
    member("Navigation Movie").viewScale = 200;
    member("Navigation Movie").broadcastProps = 0;
}
```

bufferSize

Syntax

```
-- Lingo syntax
memberObjRef.bufferSize

// JavaScript syntax
memberObjRef.bufferSize;
```

Beschreibung

Diese Flash-Darstellereigenschaft bestimmt, wie viele Byte eines verknüpften Flash-Films zu einer bestimmten Zeit in den Speicher streamen. Die Darstellereigenschaft `bufferSize` kann nur ganzzahlige Werte enthalten. Diese Eigenschaft hat nur dann eine Wirkung, wenn die Eigenschaft `preload` des Darstellers auf `FALSE` eingestellt ist.

Diese Eigenschaft kann getestet und eingestellt werden. Der Standardwert beträgt 32.768 Byte.

Beispiel

Die folgende `startMovie`-Prozedur konfiguriert einen Flash-Filmdarsteller für das Streaming und stellt dann für ihn die Eigenschaft `bufferSize` ein.

```
-- Lingo syntax
on startMovie
    member.("Flash Demo").preload = FALSE
    member.("Flash Demo").bufferSize = 65536
end

// JavaScript syntax
function startMovie() {
    member.("Flash Demo").preload = 0;
    member.("Flash Demo").bufferSize = 65536;
}
```

Siehe auch

`bytesStreamed`, `preLoadRAM`, `stream()`, `streamMode`

bufferSize (Mixer)

Syntax

```
mixer.bufferSize
```

Beschreibung

Diese Soundobjekt- und Mixereigenschaft gibt die Länge (in Millisekunden) eines Abschnitts der Soundobjekt- oder Mixerausgabe zurück, die der Verarbeitungseinheit des Audiomoduls entspricht. Die Puffergröße `bufferSize` eines Mixers gilt für alle in diesem enthaltenen Soundobjekte.

Der Wert lässt sich anpassen, um die Latenzzeit des Films einzustellen. `bufferSize` ist ein Vielfaches von 10; der Standardwert beträgt 100.

Der Wert für `bufferSize` lässt sich nur einstellen, wenn der Mixer sich im Zustand `#stopped` befindet.

Beispiele

```
--Lingo syntax
on mouseUp me
    mixerRef.bufferSize = 40
end

// JavaScript syntax
function mouseUp(){
mixeRref.bufferSize = 40;
}
```

Siehe auch

Mixer

buttonCount

Syntax

```
-- Lingo syntax
dvdObjRef.buttonCount

// JavaScript syntax
dvdObjRef.buttonCount;
```

Beschreibung

Diese DVD-Eigenschaft gibt die Anzahl der verfügbaren Schaltflächen im aktuellen DVD-Menü zurück. Nur Lesen.

Wird zurzeit auf dem Mac nicht unterstützt.

Siehe auch

DVD

buttonsEnabled

Syntax

```
-- Lingo syntax
memberOrSpriteObjRef.buttonsEnabled

// JavaScript syntax
memberOrSpriteObjRef.buttonsEnabled;
```

Beschreibung

Diese Flash-Darstellereigenschaft und Sprite-Eigenschaft legt fest, ob die Schaltflächen in einem Flash-Film aktiv (`TRUE`, Standard) oder inaktiv (`FALSE`) sind. Schaltflächenaktionen werden nur dann ausgelöst, wenn die Eigenschaft `actionsEnabled` auf `TRUE` eingestellt ist.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Die folgende Prozedur akzeptiert einen Sprite-Bezug und schaltet die Sprite-Eigenschaft `buttonsEnabled` ein oder aus.

```
-- Lingo syntax
on ToggleButtons(whichSprite)
    sprite(whichSprite).buttonsEnabled = not(sprite(whichSprite).buttonsEnabled)
end

// JavaScript syntax
function ToggleActions(whichSprite) {
    sprite(whichSprite).buttonsEnabled = !(sprite(whichSprite).buttonsEnabled);
}
```

Siehe auch

`actionsEnabled`

buttonStyle

Syntax

```
-- Lingo syntax
_movie.buttonStyle

// JavaScript syntax
_movie.buttonStyle;
```

Beschreibung

Diese Filmeigenschaft bestimmt die visuelle Reaktion von Schaltflächen bei gedrückter Maustaste. Lesen/Schreiben.

Diese Eigenschaft kann nur für Schaltflächen verwendet werden, die mit dem Schaltflächenwerkzeug in der Werkzeugpalette erstellt wurden.

Die Eigenschaft `buttonStyle` kann die folgenden Werte haben:

- 0 (Listenstil: Standard) – Nachfolgende Schaltflächen werden hervorgehoben, wenn der Cursor sich darüber bewegt. Lässt der Benutzer die Maustaste los, wird das mit der Schaltfläche verbundene Skript aktiviert.
- `1` (Dialog) – Nur die erste angeklickte Schaltfläche wird hervorgehoben. Nachfolgende Schaltflächen werden nicht hervorgehoben. Lässt der Benutzer die Maustaste los, während sich der Cursor über einer anderen als der ursprünglich angeklickten Schaltfläche befindet, wird das mit dieser Schaltfläche verbundene Skript nicht aktiviert.

Beispiel

Die folgende Anweisung stellt die Eigenschaft `buttonStyle` auf 1 ein:

```
-- Lingo syntax
_movie.buttonStyle = 1

// JavaScript syntax
_movie.buttonStyle = 1;
```

Die folgende Anweisung speichert die aktuelle Einstellung der Eigenschaft `buttonStyle`, indem sie den aktuellen `buttonStyle`-Wert in die Variable `buttonStyleValue` setzt:

```
-- Lingo syntax
buttonStyleValue = _movie.buttonStyle

// JavaScript syntax
var buttonStyleValue = _movie.buttonStyle;
```

Siehe auch

Movie

buttonType

Syntax

```
member(whichCastMember).buttonType
the buttonType of member whichCastMember
```

Beschreibung

Diese Schaltflächen-Darstellereigenschaft gibt den Typ des angegebenen Schaltflächendarstellers an. Mögliche Werte sind `#pushButton`, `#checkBox` und `#radioButton`. Diese Eigenschaft kann nur für Schaltflächen verwendet werden, die mit dem Schaltflächenwerkzeug in der Werkzeugpalette erstellt wurden.

Beispiel

Die folgende Anweisung macht den Schaltflächendarsteller Leitartikel zu einem Kontrollkästchen.

```
--Lingo Dot syntax:
member("Editorial").buttonType = #checkBox

--Lingo Verbose syntax:
set the buttonType of member "Editorial" to #checkBox

// JavaScript syntax
member("Editorial").buttonType = symbol("checkBox");
```

byteArray

Syntax

```
memberRef.byteArray
```

Beschreibung

Diese Byte-Array-Eigenschaft legt den Inhalt eines Byte-Arrays fest, oder liefert ihn zurück.

Beispiele

In folgendem Beispiel wird einem Byte-Array-Darsteller ein Byte-Array zugewiesen. Das zugewiesene Byte-Array wird beim Speichern des Films mit gespeichert.

```
--Lingo syntax
ba_m ember=new(#byteArray)
ba = bytearray("Sample by tearray contents")
ba_member.byteArray = ba

//JavaScript syntax
ba_member=_movie.newMember(symbol("byteArray"))
ba = byteArray("Sample bytearray contents")
ba_member.byteArray = ba
```

BytesRemaining

Syntax

```
byteArrayObject.bytesRemaining
```

Beschreibung

Diese Bytearrayeigenschaft gibt die Anzahl der ab der aktuellen Position im Byte-Array bis zum Ende des Arrays zum Lesen verfügbaren Datenbytes zurück.

Beispiele

```
--Lingo syntax
bArray = byteArray("Sample byte array")
put bArray.bytesRemaining

//JavaScript syntax
bArray = byteArray("Sample byte array");
put(bArray.bytesRemaining);
```

bytesStreamed

Syntax

```
-- Lingo syntax
memberObjRef.bytesStreamed

// JavaScript syntax
memberObjRef.bytesStreamed;
```

Beschreibung

Diese Flash- und Shockwave-Audio-Darstellereigenschaft gibt an, wie viele Byte des betreffenden Darstellers bereits in den Speicher geladen wurden. Die Eigenschaft `bytesStreamed` gibt nur dann einen Wert zurück, wenn der Director-Film abgespielt wird. Sie gibt eine Ganzzahl zurück.

Diese Eigenschaft kann getestet, aber nicht eingestellt werden.

Beispiel

Die folgende Prozedur akzeptiert eine Darstellerreferenz als Parameter und verwendet dann den Befehl stream, um den Darsteller in den Speicher zu laden. Jedes Mal, wenn ein Teil des Darstellers in den Speicher gestreamt wird, gibt die Prozedur mit der Eigenschaft `bytesStreamed` im Nachrichtenfenster an, wie viele Byte bereits geladen wurden.

```
-- Lingo syntax
on fetchMovie(whichFlashMovie)
    repeat while member(whichFlashMovie).percentStreamed < 100
        stream(member(whichFlashMovie))
        put("Number of bytes streamed:" && member(whichFlashMovie).bytesStreamed)
    end repeat
end

// JavaScript syntax
function fetchMovie(whichFlashMovie)
    var i = member(whichFlashMovie).percentStreamed;
    while(i < 100) {
        stream(member(whichFlashMovie));
        trace( "Number of bytes streamed: " + member(whichFlashMovie).bytesStreamed);
    }
}
```

Siehe auch

bufferSize, percentStreamed (Darsteller), stream()

bytesStreamed (3D)

Syntax

```
member(whichCastMember).bytesStreamed
```

Beschreibung

Diese 3D-Darstellereigenschaft gibt an, wie viele Bytes der anfänglichen Dateiimportanforderung bzw. der letzten Dateiladeanforderung bereits geladen wurden.

Beispiel

Die folgende Anweisung zeigt, dass 325.300 Bytes des Darstellers "Szene" bereits geladen wurden.

```
put member("Scene").bytesStreamed
-- 325300
```

Siehe auch

streamSize (3D), state (3D)

camera

Syntax

```
member(whichCastMember).camera(whichCamera)
member(whichCastMember).camera[index]
member(whichCastMember).camera(whichCamera).whichCameraProperty
member(whichCastMember).camera[index].whichCameraProperty
sprite(whichSprite).camera{(index)}
sprite(whichSprite).camera{(index)}.whichCameraProperty
```

Beschreibung

Dieses 3D-Element ist ein Objekt an einer Vektorposition, von der aus die 3D-Welt betrachtet wird.

Jedes Sprite verfügt über eine Liste mit Kameras. Die Ansicht von den einzelnen Kameras aus erscheint über der Ansicht von Kameras mit niedrigeren *index*-Positionen. Sie können für jede Kamera die Eigenschaft `rect (camera)` einstellen, um mehrere Ansichten im Sprite darzustellen.

Kameras werden in der Kamerapalette des Darstellers gespeichert. Mit den Befehlen `newCamera` und `deleteCamera` können Sie Kameras in einem 3D-Darsteller erstellen und löschen.

Bei einem Sprite bezieht sich die Eigenschaft `camera` auf die erste Kamera in der Kameraliste des Sprites. Mit `sprite(`*whichSprite*`).camera` wird auf dieselbe Kamera Bezug genommen wie mit `sprite(`*whichSprite*`).camera(1)`. Mit den Befehlen `addCamera` und `deleteCamera` können Sie bei einem 3D-Sprite die Kameraliste erstellen.

Eine vollständige Liste von Kameraeigenschaften und -befehlen finden Sie in den Themen des Director-Benutzerhandbuchs im Director-Hilfefenster.

Beispiel

Die folgende Anweisung setzt die Kamera von Sprite 1 auf die Kamera `TreeCam` des Darstellers „Picknick“:

```
-- Lingo syntax
sprite(1).camera = member("Picnic").camera("TreeCam")
```

Die folgende Anweisung setzt die Kamera von Sprite 1 auf Kamera 2 des Darstellers „Picknick“:

```
-- Lingo syntax
sprite(1).camera = member("Picnic").camera[2]

// JavaScript syntax
sprite(1).camera = member("Picnic").getProp("camera", 2);
```

Siehe auch

`bevelDepth`, `overlay`, `modelUnderLoc`, `spriteSpaceToWorldSpace`, `fog`, `clearAtRender`

cameraPosition

Syntax

```
member(whichCastMember).cameraPosition
sprite(whichSprite).cameraPosition
```

Beschreibung

Diese 3D-Eigenschaft für Darsteller und Sprites gibt die Position der Standardkamera an.

Der Standardwert dieser Eigenschaft lautet vector(0, 0, 250). Dies ist die Position der Standardkamera in einem neu erstellten 3D-Darsteller.

Beispiel

Die folgende Anweisung zeigt, dass die Position der Standardkamera des Darstellers „Babyland“ > „vector(-117.5992, -78.9491, 129.0254)“ lautet:

```
-- Lingo syntax
member("Babyland").cameraPosition = vector(-117.5992, -78.9491, 129.0254)

// JavaScript syntax
member("Babyland").cameraPosition = vector(-117.5992, -78.9491, 129.0254);
```

Siehe auch

cameraRotation, autoCameraPosition

cameraRotation

Syntax

```
member(whichCastMember).cameraRotation
sprite(whichSprite).cameraRotation
```

Beschreibung

Diese 3D-Eigenschaft für Darsteller und Sprites gibt die Position der Standardkamera an.

Der Standardwert dieses Eigenschaft lautet vector(0, 0, 0). Dies ist die Position der Standardkamera in einem neu erstellten 3D-Darsteller.

Beispiel

Die folgende Anweisung zeigt, dass die Drehung der Standardkamera des Darstellers „Babyland“ > „vector(82.6010, -38.8530, -2.4029)“ lautet:

```
member("babyland").cameraRotation = vector(82.6010, -38.8530, -2.4029)
```

Siehe auch

cameraPosition, autoCameraPosition

castLib

Syntax

```
-- Lingo syntax
_movie.castLib[castNameOrNum]

// JavaScript syntax
_movie.castLib[castNameOrNum];
```

Beschreibung

Diese Filmeigenschaft stellt unabhängig vom Aktivitätsstatus eines Films benannten oder indizierten Zugriff auf die Besetzungsbibliotheken eines Films bereit. Nur Lesen.

Das Argument *castNameOrNum* kann eine Zeichenfolge sein, die den Namen des Films angibt, oder eine Ganzzahl, die die Nummer des Films angibt, auf den zugegriffen werden soll.

Diese Eigenschaft bietet eine ähnliche Funktion wie die Top-Level-Methode `castLib()`, mit der Ausnahme, dass die `castLib()`-Methode nur für den aktuell aktiven Film gültig ist.

Beispiel

Die folgende Anweisung zeigt die Nummer der Besetzung Schaltflächen im Nachrichtenfenster an.

```
-- Lingo syntax
put(_movie.castLib["Buttons"].number)

// JavaScript syntax
put(_movie.castLib["Buttons"].number);
```

Siehe auch

castLib(), Movie

castLibNum

Syntax

```
-- Lingo syntax
memberObjRef.castLibNum

// JavaScript syntax
memberObjRef.castLibNum;
```

Beschreibung

Diese Darstellereigenschaft bestimmt die Nummer der Besetzungsbibliothek, zu der der Darsteller gehört. Nur Lesen.

Beispiel

Die folgende Anweisung bestimmt die Nummer der Besetzung, der der Darsteller "Jazz“ zugeordnet ist.

```
-- Lingo syntax
put(member("Jazz").castLibNum)

// JavaScript syntax
put(member("Jazz").castLibNum);
```

Die folgende Anweisung ändert den Darsteller, der Sprite 5 zugeordnet ist, durch Wechsel der Besetzung auf „Plan für Mittwoch“.

```
-- Lingo syntax
sprite(5).castLibNum = castLib("Wednesday Schedule").number

// JavaScript syntax
sprite(5).castLibNum = castLib("Wednesday Schedule").number;
```

Siehe auch

Cast Library, Member

castMemberList

Syntax

```
-- Lingo syntax
memberObjRef.castMemberList

// JavaScript syntax
memberObjRef.castMemberList;
```

Beschreibung

Diese Cursor-Darstellereigenschaft enthält eine Liste von Darstellern, aus denen die Bilder eines Cursors bestehen. Ersetzen Sie bei Ihrer Eingabe den Ausdruck *whichCursorCastMember* durch den Namen (in Anführungszeichen) oder die Nummer eines Darstellers. Sie können auch Darsteller aus verschiedenen Besetzungen angeben.

Der erste Darsteller in der Liste ist das erste Bild des Cursors, der zweite Darsteller ist das zweite Bild usw.

Wenn Sie Darsteller angeben, die für einen Cursor nicht gültig sind, werden diese übergangen und die restlichen Darsteller verwendet.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Dieser Befehl erstellt eine Folge von vier Darstellern, die für den animierten Farbcursor-Darsteller mit der Bezeichnung `myCursor` verwendet werden.

```
-- Lingo syntax
member("myCursor").castmemberList = [member(1), member(2), member(1, 2), member(2, 2)]

// JavaScript syntax
member("myCursor").castmemberList = list(member(1), member(2), member(1, 2), member(2, 2));
```

center

Syntax

```
member(whichCastMember).center
the center of member whichCastMember
```

Beschreibung

Diese Darstellereigenschaft steht mit der Eigenschaft `crop` in Wechselwirkung.

- Wenn die Eigenschaft `crop` auf `FALSE` gesetzt ist, hat die Eigenschaft `center` keine Wirkung.
- Wenn `cropTRUE` ist und `center` ebenfalls `TRUE` ist, wird der Digitalvideo-Darsteller um den Mittelpunkt herum zugeschnitten.
- Wenn `cropTRUE` und `centerFALSE` ist, werden die rechte und die untere Seite des Digitalvideo-Darstellers zugeschnitten.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Die folgende Anweisung bewirkt, dass der Digitalvideo-Darsteller Interview in der linken oberen Ecke des Sprites angezeigt wird.

```
-- Lingo Dot syntax:
member("Interview").center = FALSE

-- Lingo Verbose syntax:
set the center of member "Interview" to FALSE

// JavaScript syntax
member("Interview").center = false
```

Siehe auch

crop, centerRegPoint, regPoint, scale (Darsteller)

centerRegPoint

Syntax

```
-- Lingo syntax
memberObjRef.centerRegPoint

// JavaScript syntax
memberObjRef.centerRegPoint;
```

Beschreibung

Diese Flash-, Vektorform- und Bitmapdarsteller-Eigenschaft zentriert das Registrierungskreuz des Darstellers automatisch, wenn Sie die Größe des Sprites ändern (`TRUE`, Standard) bzw. positioniert das Registrierungskreuz an seinem aktuellen Punktwert neu, wenn Sie die Größe des Sprites ändern, die Eigenschaft `defaultRect` oder die Eigenschaft `regPoint` einstellen (`FALSE`).

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Dieses Skript prüft, ob die Eigenschaft `centerRegPoint` eines Flash-Films auf TRUE eingestellt ist. Wenn dem so ist, verwendet das Skript die Eigenschaft `regPoint`, um das Registrierungskreuz des Sprites in seiner linken oberen Ecke neu zu positionieren. Durch Prüfen der Eigenschaft `centerRegPoint` stellt das Skript sicher, dass es nicht versehentlich ein Registrierungskreuz, das vorher mit der Eigenschaft `regPoint` eingestellt wurde, neu positioniert.

```
-- Lingo syntax
property spriteNum

on beginSprite me
    if sprite(spriteNum).member.centerRegPoint = TRUE then
        sprite(spriteNum).member.regPoint = point(0,0)
    end if
end

// JavaScript syntax
function beginSprite() {
    var ctrRg = sprite(this.spriteNum).member.centerRegPoint;
    if (ctrRg == 1) {
        sprite(this.spriteNum).member.regPoint = point(0,0);
    }
}
```

Siehe auch

regPoint

centerStage

Syntax

```
-- Lingo syntax
_movie.centerStage

// JavaScript syntax
_movie.centerStage;
```

Beschreibung

Diese Filmeigenschaft bestimmt, ob die Bühne beim Laden des Films auf dem Bildschirm zentriert (`TRUE`, Standard) oder nicht zentriert (`FALSE`) angezeigt wird. Lesen/Schreiben.

Setzen Sie die Anweisung, die diese Eigenschaft enthält, in den Film, der dem Film, für den sie bestimmt ist, vorangeht.

Diese Eigenschaft ist zum Prüfen der Bühnenposition geeignet, bevor Sie einen Film in einem Projektor abspielen.

Hinweis: *Denken Sie daran, dass es beim Abspielen in einem Projektor Unterschiede im Verhalten zwischen Windows und dem Mac gibt. Einstellungen, die während der Erstellung des Projektors ausgewählt wurden, haben womöglich Vorrang vor dieser Eigenschaft.*

Beispiel

Die folgende Anweisung schickt den Film zu einem bestimmten Bild, wenn die Bühne nicht zentriert ist:

```
-- Lingo syntax
if (_movie.centerStage = FALSE) then
     _movie.go("Off Center")
end if

// JavaScript syntax
if (_movie.centerStage == false) {
    _movie.go("Off Center");
}
```

Die folgende Anweisung kehrt die Eigenschaft `centerStage` ins Gegenteil ihres aktuellen Werts um:

```
-- Lingo syntax
_movie.centerStage = not(_movie.centerStage)

// JavaScript syntax
_movie.centerStage = !(_movie.centerStage)
```

Siehe auch

`fixStageSize`, `Movie`

changeArea

Syntax

```
member(whichCastMember).changeArea
the changeArea of member whichCastMember
```

Beschreibung

Diese Übergangsdarsteller-Eigenschaft bestimmt, ob ein Übergang nur auf den sich ändernden Bereich auf der Bühne (`TRUE`) oder auf die ganze Bühne (`FALSE`) angewendet wird. Ein ähnlicher Effekt kann durch Auswählen der Option „Nur Änderungsbereich“ im Dialogfeld „Bildeigenschaften: Übergang“ erzielt werden.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Die folgende Anweisung bewirkt, dass der Übergangsdarsteller "Welle“ nur auf den sich ändernden Bereich der Bühne angewendet wird.

```
-- Lingo Dot syntax:
member("Wave").changeArea = TRUE

-- Lingo Verbose syntax:
set the changeArea of member "Wave" to TRUE

// JavaScript syntax
member("Wave").changeArea = true;
```

channel

Syntax

```
mixer.channel (Read-only)
```

Beschreibung

Diese Audiomixereigenschaft gibt den mit dem Mixer verknüpften Kanal zurück. Diese Eigenschaft ist nur gültig, wenn ein Mixer abgespielt wird.

Mit dem Ausdruck `_movie.channel[1]` lässt sich Kanal 1 eines Films mit Lingo oder JavaScript ansprechen.

Beispiele

```
--Lingo syntax
on mouseUp me
    put mixer1.channel --Displays the channel that the mixer is associated with.
end

// JavaScript syntax
function mouseup()
{
put (mixer1.channel); //Displays the channel that the mixer is associated with.
}
```

Siehe auch

Mixer

channelCount (Mixer)

Syntax

```
mixer.channelCount (Read-write)
```

Beschreibung

Diese Mixereigenschaft gibt die Anzahl der Kanäle in einem Mixer aus oder legt sie fest. Diese Eigenschaft besitzt einen Lese-/Schreibzugriff.

Beispiel

```
--Lingo syntax
on mouseUp me
mixerRef.channelcount = 6 -- Sets the channelCount of the mixer to 6.
end

// JavaScript syntax
function mouseUp(){
mixerRef.channelcount = 6; // Sets the channelCount of the mixer to 6.
}
```

Siehe auch

Mixer

channelCount (Soundkanal)

Syntax

```
-- Lingo syntax
soundChannelObjRef.channelCount

// JavaScript syntax
soundChannelObjRef.channelCount;
```

Beschreibung
Diese Soundkanaleigenschaft bestimmt die Anzahl von Kanälen im aktuell wiedergegebenen oder angehaltenen Sound in einem angegebenen Soundkanal. Nur Lesen.

Diese Eigenschaft ist hilfreich, um zu ermitteln, ob ein Sound Mono oder Stereoist.

Beispiel
Die folgende Anweisung bestimmt die Anzahl der Kanäle im Sounddarsteller „Jazz“.

```
-- Lingo syntax
put(member("Jazz").channelCount)

// JavaScript syntax
put(member("Jazz").channelCount);
```

Die folgende Anweisung bestimmt die Anzahl der Kanäle im Sounddarsteller auf Kanal 2:

```
-- Lingo syntax
put(sound(2).channelCount)

// JavaScript syntax
put(sound(2).channelCount);
```

Siehe auch
Sound Channel

channelCount (Soundobjekt)

Syntax
```
soundObj.channelCount (Read-only)
```

Beschreibung
Diese Soundobjekteigenschaft gibt die Anzahl der Kanäle im Soundobjekt zurück.

Beispiele
```
--Lingo syntax
on mouseUp me
    put soundObjRef.channelCount -- Displays the number of channels in the sound
-- object associated with soundobjectRef.
end

// JavaScript syntax
function mouseUp(){
put (soundObjRef.channelCount) ; // Displays the number of channels in the sound object
// associated with soundobjectRef.
}
```

chapter

Syntax

```
-- Lingo syntax
dvdObjRef.chapter

// JavaScript syntax
dvdObjRef.chapter;
```

Beschreibung

Diese DVD-Eigenschaft gibt die Nummer des aktuellen Kapitels zurück. Lesen/Schreiben.

Beispiel

Die folgende Anweisung gibt das aktuelle Kapitel zurück:

```
-- Lingo syntax
trace (member(1).chapter)-- 1

// JavaScript syntax
trace (member(1).chapter);// 1
```

Siehe auch

DVD

chapterCount

Syntax

```
-- Lingo syntax
dvdObjRef.chapterCount

// JavaScript syntax
dvdObjRef.chapterCount;
```

Beschreibung

Diese DVD-Methode gibt die Anzahl verfügbarer Kapitel in einem Titel zurück. Nur Lesen.

Beispiel

Die folgende Anweisung gibt die Anzahl der Kapitel im aktuellen Titel zurück:

```
-- Lingo syntax
trace (member(1).chapterCount)-- 17

// JavaScript syntax
trace (member(1). chapterCount);// 17
```

Siehe auch

chapterCount(), DVD

characterSet

Syntax

```
-- Lingo syntax
memberObjRef.characterSet

// JavaScript syntax
memberObjRef.characterSet;
```

Beschreibung

Diese Textdarsteller-Eigenschaft gibt einen String mit den Zeichen zurück, die beim Erstellen des Darstellers importiert wurden. Wenn alle Zeichen der Originalschriftart eingebunden wurde, ist das Ergebnis ein leerer String.

Beispiel

Die folgende Anweisung zeigt die Zeichen an, die beim Erstellen von Darsteller 11 eingebunden wurden. Bei den beim Importieren aufgenommenen Zeichen handelt es sich um Ziffern und römische Buchstaben.

```
-- Lingo syntax
put(member(11).characterSet)

// JavaScript syntax
put(member(11).characterSet);
```

Siehe auch

recordFont, bitmapSizes, originalFont

charSpacing

Syntax

```
-- Lingo syntax
chunkExpression.charSpacing

// JavaScript syntax
chunkExpression.charSpacing;
```

Beschreibung

Diese Textdarsteller-Eigenschaft ermöglicht die Angabe weiterer Abstände, die auf die einzelnen Buchstaben im *chunkExpression*-Abschnitt des Textdarstellers angewendet werden.

Ist der Wert kleiner als Null, wird der Abstand zwischen den Buchstaben kleiner. Ist der Wert größer als Null, wird der Abstand zwischen den Buchstaben größer.

Der Standardwert 0 ergibt den Standardabstand zwischen den Buchstaben.

Beispiel

Die folgende Prozedur vergrößert den Zeichenabstand im dritten bis fünften Wort des Textdarstellers `myCaption` um den Wert 2:

```
--Lingo syntax
on myCharSpacer
    mySpaceValue = member("myCaption").word[3..5].charSpacing
    member("myCaption").word[3..5].charSpacing = (mySpaceValue + 2)
end

// JavaScript syntax
function myCharSpacer() {
    var i = 3;
    while (i < 6) {
        var mySpaceValue = member("myCaption").getPropRef("word", i).charSpacing;
        member("myCaption").getPropRef("word", i).charSpacing = (mySpaceValue + 2);
    }
}
```

checkMark

Syntax

```
the checkMark of menuItem whichItem of menu whichMenu
```

Beschreibung

Diese Menüelementeigenschaft bestimmt, ob neben dem benutzerdefinierten Menüelement ein Häkchen angezeigt wird (`TRUE`) oder nicht (`FALSE`, Standard).

Der Wert von *whichItem* kann entweder der Name oder die Nummer eines Menüelements sein. Der Wert von *whichMenu* kann entweder ein Menüname oder eine Menünummer sein.

Diese Eigenschaft kann getestet und eingestellt werden.

Hinweis: *Im Shockwave Player sind keine Menüs verfügbar.*

Beispiel

Die folgende Prozedur deaktiviert alle Elemente, die im benutzerdefinierten Menü, das vom Argument `theMenu` angegeben ist, mit einem Häkchen versehen sind. Zum Beispiel schaltet `unCheck` ("Format") alle Elemente im Menü "Format" aus.

```
-- Lingo syntax
on unCheck theMenu
    set n = the number of menuItems of menu theMenu
    repeat with i = 1 to n
        set the checkMark of menuItem i of menu theMenu to FALSE
    end repeat
end unCheck

// JavaScript syntax
function unCheck (theMenu) {
    var n = _menuBar.menu[theMenu].item.count; //the number of menuItems of menu theMenu
    for( i = 1 ; i <= n ; i++ )
        _menuBar.menu[theMenu].item[i].checkMark = false;
}
```

Siehe auch

`installMenu`, `enabled`, `name (Menüelementeigenschaft)`, `number (Menüelemente)`, `script`, `menu`

child (3D)

Syntax

```
member(whichCastmember).model(whichParentNode).child(whichChildNodeName)
member(whichCastmember).model(whichParentNode).child[index]
```

Beschreibung

Diese 3D-Eigenschaft für Modelle, Gruppen, Lichtquellen und Kameras gibt die im Parameter *whichChildNodeName* angegebene Child-Node bzw. die an der angegebenen Indexposition in der Child-Liste der Parent-Node befindliche Child-Node zurück. Ein Node kann ein Modell, eine Gruppe, eine Kamera oder ein Licht sein.

Die Transformation eines Node ist parentbezogen. Wenn Sie die Position des Parent-Node ändern, werden die zugehörigen Child-Nodes ebenfalls verschoben. Ihre Position relativ zum Parent-Node wird dabei beibehalten. Änderungen der Dreh- und Skalierungseigenschaften des Parent-Node wirken sich ebenfalls auf die Child-Nodes aus.

Verwenden Sie die Methode `addChild` des Parent-Knoten bzw. stellen Sie die Eigenschaft `parent` des Child-Knotens ein, um die Child-Liste des Parent-Knotens zu erweitern. Ein Child-Node kann nur einen Parent-Node besitzen, ein Parent-Node jedoch beliebig viele Child-Nodes. Ein Child-Node kann auch eigene Child-Nodes besitzen.

Beispiel

Die folgende Anweisung zeigt, dass die zweite Child-Node des Modells "Auto" das Modell "Reifen" ist.

```
-- Lingo syntax
put member("3D").model("Car").child[2]
-- model("Tire")

// JavaScript syntax
put ( member("3D").getProp("model", 1).child[2] );
// model("Tire")
```

Siehe auch

addChild, parent

child (XML)

Syntax

```
XMLnode.child[ childNumber ]
```

Beschreibung

Diese XML-Eigenschaft nimmt auf den angegebenen Child-Node in der verschachtelten Tag-Struktur eines geparsten XML-Dokuments Bezug.

Beispiel

Als Ausgangspunkt wird das folgende XML-Dokument verwendet:

```
<?xml version="1.0"?>
    <e1>
        <tagName attr1="val1" attr2="val2"/>
        <e2> element 2</e2>
        <e3>element 3</e3>
        here is some text
    </e1>
```

Die folgende Lingo-Anweisung gibt den Namen des ersten Child-Node im XML-Beispiel zurück:

```
put gParserObject.child[1].name
-- "e1"
```

chunkSize

Syntax

```
member(whichCastMember).chunkSize
the chunkSize of member whichCastMember
```

Beschreibung

Diese Übergangsdarsteller-Eigenschaft bestimmt die Chunk-Größe des Übergangs in Pixel von 1 bis 128 und entspricht der Einstellung des Weichheitsgrad-Reglers im Dialogfeld „Bildeigenschaften“: Je kleiner die Chunk-Größe ist, desto fließender erscheint der Übergang.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Die folgende Anweisung stellt die Chunk-Größe des Übergangsdarstellers Nebel auf 4 Pixel ein.

Punktsyntax:

```
member("Fog").chunkSize = 4
```

Ausführliche Syntax:

```
set the chunkSize of member "Fog" to 4
```

clearAtRender

Syntax

```
member(whichCastmember).camera(whichCamera).colorBuffer.clearAtRender
sprite(whichSprite).camera{(index)}.colorBuffer.clearAtRender
```

Beschreibung

Diese 3D-Eigenschaft gibt an, ob der Farbpuffer nach jedem Bild geleert wird. Wenn Sie den Wert auf `FALSE` setzen (der Puffer wird nicht geleert), ergibt sich ein ähnlicher Effekt wie beim Farbeffekt „#trails“. Der Standardwert dieser Eigenschaft lautet `TRUE`.

Beispiel

Die folgende Anweisung verhindert, dass Director vorherige Bilder aus der Kameraansicht löscht. In Bewegung befindliche Modelle erscheinen auf der Bühne verwischt.

```
-- Lingo syntax
sprite(1).camera.colorBuffer.clearAtRender = 0

// JavaScript syntax
sprite(1).camera.getPropRef("colorBuffer").clearAtRender = 0;
```

Siehe auch

clearValue

clearValue

Syntax

```
member(whichCastmember).camera(whichCamera).colorBuffer.clearValue
sprite(whichSprite).camera{(index)}.colorBuffer.clearValue
```

Beschreibung

Diese 3D-Eigenschaft gibt die Farbe an, die zum Leeren des Farbpuffers verwendet wird, wenn `colorBuffer.clearAtRender` auf `TRUE` gesetzt ist. Der Standardwert dieser Eigenschaft lautet `rgb(0, 0, 0)`.

Beispiel

Die folgende Anweisung setzt die Eigenschaft `clearValue` der Kamera auf "rgb(255, 0, 0)". Bereiche in der 3D-Welt, die nicht durch Modelle belegt sind, erscheinen rot.

```
-- Lingo syntax
sprite(1).camera.colorBuffer.clearValue= rgb(255, 0, 0)

// JavaScript syntax
sprite(1).camera.getPropRef("colorBuffer").clearValue= color(255, 0, 0);
```

Siehe auch

clearAtRender

clickLoc

Syntax

```
-- Lingo syntax
_mouse.clickLoc

// JavaScript syntax
_mouse.clickLoc;
```

Beschreibung

Diese Mauseigenschaft identifiziert die Stelle auf dem Bildschirm, auf die zuletzt mit der Maus geklickt wurde, als Punkt. Nur Lesen.

Beispiel

Die folgende Prozedur `mouseDown` zeigt die Position an, auf die zuletzt mit der Maus geklickt wurde:

```
-- Lingo syntax
on mouseDown
    put(_mouse.clickLoc)
end mouseDown

// JavaScript syntax
function mouseDown() {
    put(_mouse.clickLoc);
}
```

Wenn der Klick 50 Pixel vom linken Rand und 100 Pixel vom oberen Rand der Bühne entfernt erfolgt, zeigt das Nachrichtenfenster Folgendes an:

```
point(50, 100)
```

Siehe auch

`clickOn`, `Mouse`

clickMode

Syntax

```
-- Lingo syntax
memberOrSpriteObjRef.clickMode

// JavaScript syntax
memberOrSpriteObjRef.clickMode;
```

Beschreibung

Diese Flash-Darsteller- und Sprite-Eigenschaft legt fest, wann das Flash-Film-Sprite Mausklickereignisse (`mouseUp` und `mouseDown`) und Rollover (`mouseEnter`, `mouseWithin` und `mouseLeave`) erkennt. Die Eigenschaft `clickMode` kann folgende Werte haben:

- `#boundingBox`#boundingBox – Mausklickereignisse werden innerhalb des Begrenzungsrechtecks und Rollover an den Rändern des Sprites erkannt.
- `#opaque` (Standard) – Mausklickereignisse werden nur erkannt, wenn sich der Cursor über einem undurchsichtigen Bereichs des Sprites befindet. Rollover werden an den Rändern der undurchsichtigen Bereiche des Sprites erkannt, wenn der Farbeffekt des Sprites auf „Hintergrund transparent“ eingestellt ist. Wenn der Farbeffekt des Sprites nicht auf Hintergrund transparent eingestellt ist, hat diese Einstellung dieselbe Wirkung wie die Einstellung `#boundingBox`.
- `#object` – Mausklickereignisse werden erkannt, wenn sich der Cursor über einem gefüllten (nicht zum Hintergrund gehörenden) Bereich des Sprites befindet. Rollover werden an den Rändern des gefüllten Bereichs erkannt. Der Farbeffekt des Sprites spielt bei dieser Einstellung keine Rolle.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Dieses Skript prüft, ob das Sprite mit dem Farbeffekt „Hintergrund transparent“ gegenwärtig so eingestellt ist, dass es direkt auf der Bühne wiedergegeben wird. Wenn das Sprite nicht direkt auf der Bühne wiedergegeben wird, wird die Eigenschaft `clickMode` des Sprites auf `#opaque` eingestellt. Andernfalls wird `clickMode` des Sprites auf `#boundingBox` eingestellt (weil Director Farbeffekte für Flash-Film-Sprites, die direkt auf der Bühne wiedergegeben werden, ignoriert).

```
-- Lingo syntax
property spriteNum

on beginSprite me
    if sprite(spriteNum).directToStage = FALSE then
        sprite(spriteNum).clickMode = #opaque
    else
        sprite(spriteNum).clickMode = #boundingBox
    end if
end

// JavaScript syntax
function beginSprite(me){
    var dts = sprite(this.spriteNum).directToStage;
    if (dts == 0) {
        sprite(this.spriteNum).clickMode = symbol("opaque");
    } else {
        sprite(this.spriteNum).clickMode = symbol("boundingBox");
    }
}
```

clickOn

Syntax

```
-- Lingo syntax
_mouse.clickOn

// JavaScript syntax
_mouse.clickOn;
```

Beschreibung

Diese Mauseigenschaft gibt das aktive Sprite zurück, auf das der Benutzer zuletzt geklickt hat. Nur Lesen.

Ein aktives Sprite ist ein Sprite, das mit einem Sprite- oder Darstellerskript verbunden ist.

Wenn der Benutzer auf die Bühne klickt, gibt `clickOn` 0 zurück. Um zu erkennen, ob der Benutzer auf ein skriptloses Sprite geklickt hat, müssen Sie dem Sprite ein Mausereignisskript zuordnen, damit es von der Funktion `clickOn` erkannt wird. Beispiel:

```
-- Lingo syntax
on mouseUp me
    ...
end
```

Schaltflächen, Kontrollkästchen und Optionsfelder werden von `clickOn` auch dann erkannt, wenn an ihnen kein Skript angebracht ist.

Die Eigenschaft `clickOn` kann in einer Schleife geprüft werden. Wenn die Prozedur ausgeführt wird, ändert jedoch weder `clickOn` noch `clickLoc` ihren Wert. Sie erhalten den Wert, der vor Beginn der Prozedur bestand.

Beispiel

Die folgende Anweisung prüft, ob Sprite 7 das aktive Sprite ist, auf das zuletzt geklickt wurde:

```
-- Lingo syntax
if (_mouse.clickOn = 7) then
    _player.alert("Sorry, try again.")
end if

// JavaScript syntax
if (_mouse.clickOn == 7) {
    _player.alert("Sorry, try again.");
}
```

Die folgende Anweisung stellt die Eigenschaft `foreColor` des zuletzt geklickten aktiven Sprites auf eine nach dem Zufallsprinzip ausgewählte Farbe ein:

```
-- Lingo syntax
sprite(_mouse.clickOn).foreColor = (random(255) - 1)

// JavaScript syntax
sprite(_mouse.clickOn).foreColor = (random(255) - 1);
```

Siehe auch

`clickLoc`, `Mouse`

closed

Syntax

```
-- Lingo syntax
memberObjRef.closed

// JavaScript syntax
memberObjRef.closed;
```

Beschreibung

Diese Vektorform-Darstellereigenschaft gibt an, ob die Endpunkte eines Pfads geschlossen oder offen sind.

Vektorformen müssen geschlossen sein, um eine Füllung enthalten zu können.

Folgende Werte sind möglich:

- `TRUE` – Die Endpunkte sind geschlossen.
- `FALSE` – Die Endpunkte sind offen.

closedCaptions

Syntax

```
-- Lingo syntax
dvdObjRef.closedCaptions

// JavaScript syntax
dvdObjRef.closedCaptions;
```

Beschreibung

Diese DVD-Eigenschaft bestimmt, ob Untertitel aktiviert sind (`TRUE`) oder nicht bzw. nicht aktiviert werden konnten (`FALSE`). Wird zurzeit auf dem Mac nicht unterstützt. Lesen/Schreiben.

Beispiel

Die folgenden Anweisungen versuchen, `closedCaptions` auf „TRUE“ festzulegen, und zeigen eine Warnmeldung an, wenn die Aktivierung nicht möglich ist:

```
-- Lingo syntax
member(3).closedCaptions = TRUE
if (member(3).closedCaptions = FALSE) then
    _player.alert("Closed captions cannot be enabled.")
end if

// JavaScript syntax
member(3).closedCaptions = true
if (member(3).closedCaptions == false) {
    _player.alert("Closed captions cannot be enabled.");
}
```

Wird zurzeit auf dem Mac nicht unterstützt.

Siehe auch

DVD

collision (modifier)

Syntax

```
member(whichCastmember).model(whichModel).collision.collisionModifierProperty
```

Beschreibung

Dieser 3D-Modifizierer dient zur Erkennung und Behebung von Kollisionen. Wenn Sie den Modifizierer `collision` mit dem Befehl `addModifier` zu einem Modell hinzufügen, können Sie auf die folgenden Eigenschaften des Modifizierers collision zugreifen:

enabled (collision)gibt an, ob Kollisionen mit dem Modell entdeckt werden.

resolvegibt an, ob Kollisionen mit dem Modell behoben werden.

immovablegibt an, ob ein Modell von Bild zu Bild verschoben werden kann.

mode (collision)gibt die zur Kollisionserkennung verwendete Geometrie an.

Hinweis: *Weitere Informationen zu diesen Eigenschaften finden Sie in den einzelnen Beschreibungen.*

Der Modifizierer „collision“ generiert die folgenden Ereignisse. Weitere Informationen zu Kollisionsereignissen enthält der Eintrag zu `registerForEvent()`.

Das Ereignis `#collideAny` wird generiert, wenn eine Kollision zwischen Modellen stattfindet, an denen der Modifizierer `collision` angebracht ist.

Das Ereignis `#collideWith` wird generiert, wenn eine Kollision mit einem bestimmten Modell stattfindet, an dem der Modifizierer `collision` angebracht ist.

Das Objekt `collisionData` wird bei den Ereignissen `#collideAny` und `#collideWith` als Argument übergeben. Weitere Einzelheiten zu den zugehörigen Eigenschaften enthält der Eintrag zu `collisionData`.

Siehe auch

`addModifier`, `removeModifier`, `modifiers`

collisionData

Syntax

```
on myHandlerName me, collisionData
```

Beschreibung

Dieses 3D-Datenobjekt wird bei den Ereignissen `#collideWith` und `#collideAny` als Argument an die in den Befehlen `registerForEvent`, `registerScript` und `setCollisionCallback` angegebene Prozedur übergeben. Das `collisionData`-Objekt weist folgende Eigenschaften auf:

`modelA`ist eines der an der Kollision beteiligten Modelle.

`modelB` ist das andere an der Kollision beteiligte Modell.

`pointOfContact`ist die Weltposition der Kollision.

`collisionNormal`ist die Richtung der Kollision.

Beispiel

Dieses Beispiel setzt sich aus drei Teilen zusammen. Der erste Teil ist die erste Codezeile, die die Prozedur `#putDetails` für das Ereignis `#collideAny` registriert. Der zweite Teil ist die Prozedur `#putDetails`. Wenn zwei Modelle im Darsteller `MyScene` zusammenstoßen, wird die Prozedur `#putDetails` aufgerufen und das Argument `collisionData` an sie übergeben. Diese Prozedur zeigt die vier Eigenschaften des Objekts `collisionData` im Nachrichtenfenster an. Der dritte Teil des Beispiels zeigt die Ergebnisse aus dem Nachrichtenfenster an. Aus den ersten zwei Zeilen geht hervor, dass bei der Kollision das Modell `GreenBall` Modell A war und das Modell `YellowBall` Modell B war. In der dritten Zeile wird der Berührungspunkt der beiden Modelle angegeben. Aus der letzten Zeile geht die Kollisionsrichtung hervor.

```
-- Lingo syntax

member("MyScene").registerForEvent(#collideAny, #putDetails, 0)

on putDetails me, collisionData
    put collisionData.modelA
    put collisionData.modelB
    put collisionData.pointOfContact
    put collisionData.collisionNormal
end
-- model("GreenBall")
-- model("YellowBall")
-- vector( 24.800, 0.000, 0.000 )
-- vector( -1.000, 0.000, 0.000 )

// JavaScript syntax

member("MyScene").registerForEvent(symbol("collideAny"),symbol("putDetails"), 0);

function putDetails (me, collisionData)
{
    put (collisionData.modelA);
    put (collisionData.modelB);
    put (collisionData.pointOfContact);
    put (collisionData.collisionNormal);
}
// model("GreenBall")
// model("YellowBall")
// vector( 24.800, 0.000, 0.000 )
// vector( -1.000, 0.000, 0.000 )
```

Siehe auch

Eigenschaften von `collisionData`: `modelA`, `modelB`, `pointOfContact`, `collisionNormal`

Methoden von `collisionData`: `resolveA`, `resolveB`, `collision (modifier)`

collisionNormal

Syntax

```
collisionData.collisionNormal
```

Beschreibung

Bei dieser 3D-Eigenschaft des `collisionData`-Objekts handelt es sich um einen Vektor, der die Richtung der Kollision angibt.

Das `collisionData`-Objekt wird bei den Ereignissen `#collideWith` und `#collideAny` als Argument an die in den Befehlen `registerForEvent`, `registerScript`, und `setCollisionCallback` angegebene Prozedur übergeben.

Die Ereignisse `#collideWith` und `#collideAny` werden gesendet, wenn eine Kollision zwischen Modellen stattfindet, an denen der Modifizierer "collision" angebracht ist. Die Eigenschaft `resolve` der Modifizierer des Modells muss auf `TRUE` gesetzt sein.

Diese Eigenschaft kann getestet, aber nicht eingestellt werden.

Beispiel

Dieses Beispiel setzt sich aus zwei Teilen zusammen. Der erste Teil ist die erste Codezeile, die die Prozedur `#explode` für das Ereignis `#collideAny` registriert. Der zweite Teil ist die Prozedur `#explode`. Wenn zwei Modelle im Darsteller `MyScene` zusammenstoßen, wird die Prozedur `#explode` aufgerufen und das Argument `collisionData` an sie übergeben. Die ersten zehn Zeilen der Prozedur `#explode` erstellen die Modellressource "Funkenquelle" und legen ihre Eigenschaften fest. Bei dieser Modellressource handelt es sich um einen einzigen Partikelstrahl. Die zehnte Zeile setzt die Richtung des Strahls auf `collisionNormal`, was der Richtung der Kollision entspricht. Die elfte Zeile der Prozedur erstellt ein Modell namens `SparksModel` mithilfe der Modellressource `SparkSource`. Die letzte Zeile der Prozedur setzt die Position von `SparksModel` auf die Stelle, an der die Kollision stattfand. Bei einer Kollision entstehen dadurch Funken, die ausgehend vom Berührungspunkt in die Kollisionsrichtung fliegen.

```
-- Lingo syntax
member("MyScene").registerForEvent(#collideAny, #explode, 0)
on explode me, collisionData
    nmr = member("MyScene").newModelResource("SparkSource", #particle)
    nmr.emitter.mode = #burst
    nmr.emitter.loop = 0
    nmr.emitter.minSpeed = 30
    nmr.emitter.maxSpeed = 50
    nmr.emitter.angle = 45
    nmr.colorRange.start = rgb(0, 0, 255)
    nmr.colorRange.end = rgb(255, 0, 0)
    nmr.lifetime = 5000
    nmr.emitter.direction = vector(0,0,-1)
    nm = member("MyScene").newModel("SparksModel", nmr)
    nm.transform.position = collisionData.pointOfContact
    nm.pointAt(collisionData.pointOfContact + collisionData.collisionNormal)
end

// JavaScript syntax
member("MyScene").registerForEvent(symbol("collideAny"), symbol("explode"), 0);
function explode (me, collisionData) {
    nmr = member("MyScene").newModelResource("SparkSource", symbol("particle"));
    nmr.getPropRef("emitter").mode = symbol("burst");
    nmr.getPropRef("emitter").loop = 0;
    nmr.getPropRef("emitter").minSpeed = 30;
    nmr.getPropRef("emitter").maxSpeed = 50;
    nmr.getPropRef("emitter").angle = 45;
    nmr.getPropRef("colorRange").start = color(0, 0, 255);
    nmr.getPropRef("colorRange").end = color(255, 0, 0);
    nmr.lifetime = 5000;
    nmr.getPropRef("emitter").direction = vector(0,0,-1);
    nm = member("MyScene").newModel("SparksModel", nmr);
    nm.transform.position = collisionData.pointOfContact;
    nm.pointAt(collisionData.pointOfContact + collisionData.collisionNormal);
}
```

Siehe auch

pointOfContact, modelA, modelB, resolveA, resolveB, collision (modifier)

color()

Syntax

```
color(#rgb, redValue, greenValue, blueValue)
color(#paletteIndex, paletteIndexNumber)
rgb(rgbHexString)
rgb(redValue, greenValue, blueValue)
paletteIndex(paletteIndexNumber)
```

Beschreibung

Diese Funktion und dieser Datentyp bestimmen, ob die Farbe eines Objekts als RGB- oder als 8-Bit-Palettenindexwert angegeben wird. Hierbei handelt es sich um dieselben Werte, die auch in den Darstellereigenschaften `color` und den Sprite-Eigenschaften `color`, den Darstellereigenschaften `bgColor` und den Sprite-Eigenschaften `bgColor` sowie in der Eigenschaft `bgColor` der Bühne verwendet werden.

Mit der Funktion `color` können entweder 24-Bit- oder 8-Bit-Farbwerte manipuliert sowie auf Darsteller, Sprites und die Bühne angewendet werden.

Bei RGB-Werten hat jede Farbkomponente einen Bereich von 0 bis 255. Bei `paletteIndex`-Typen wird eine Ganzzahl von 0 bis 255 verwendet, die die Indexposition in der aktuellen Palette angibt, und alle anderen Werte werden gekürzt.

Beispiel

Die folgende Anweisung führt eine mathematische Operation durch:

```
palColorObj = paletteIndex(20)
put palColorObj
-- paletteIndex(20)
put palColorObj / 2
-- paletteIndex(10)
```

Die folgende Anweisung wandelt einen Farbtyp in einen anderen Typ um:

```
newColorObj = color(#rgb, 155, 0, 75)
put newColorObj
-- rgb(155, 0, 75)
newColorObj.colorType = #paletteIndex
put newColorObj
-- paletteIndex(106)
```

Die folgende Anweisung ruft den Hexadezimalwert einer Farbe ungeachtet ihres Typs ab:

```
someColorObj = color(#paletteIndex, 32)
put someColorObj.hexString()
-- "#FF0099"
```

Die folgende Anweisung bestimmt die einzelnen RGB-Komponenten sowie den `paletteIndex`-Wert einer Farbe, ungeachtet ihres Typs:

```
newColorObj = color(#rgb, 155, 0, 75)
put newColorObj.green
-- 0
put newColorObj.paletteIndex
-- 106
newColorObj.green = 100
put newColorObj.paletteIndex
-- 94
put newColorObj
-- rgb(155, 100, 75)
newColorObj.paletteIndex = 45
put newColorObj
-- paletteIndex(45)
```

Die folgende Anweisung ändert die Farbe des vierten bis siebten Zeichens des Textdarstellers `myQuotes`.

```
member("myQuotes").char[4..7].color = rgb(200, 150, 75)
```

Die folgende Lingo-Anweisung zeigt die Farbe von Sprite 6 im Nachrichtenfenster an und stellt sie anschließend auf einen neuen RGB-Wert ein:

```
put sprite(6).color
-- rgb( 255, 204, 102 )
sprite(6).color = rgb(122, 98, 210)
```

Hinweis: *Wenn Sie den paletteIndex-Wert eines RGB-Farbtyps einstellen, ändert sich colorType zu paletteIndex. Wenn Sie den RGB-Farbtyp einer paletteIndex-Farbe einstellen, wird der colorType-Wert auf RGB eingestellt.*

Siehe auch

bgColor (Fenster)

color (fog)

Syntax

```
member(whichCastmember).camera(whichCamera).fog.color
sprite(whichSprite).camera{(index)}.fog.color
```

Beschreibung

Diese 3D-Eigenschaft gibt die Farbe an, die von der Kamera zur Szene hinzugefügt wird, wenn die Kameraeigenschaft `fog.enabled` auf `TRUE` gesetzt ist.

Der Standardwert dieser Eigenschaft lautet `rgb(0, 0, 0)`.

Beispiel

Die folgende Anweisung setzt die Nebelfarbe der Kamera `BayView` auf `rgb(255, 0, 0)`.. Wenn die Eigenschaft `fog.enabled` der Kamera `TRUE` lautet, erscheinen Modelle im Nebel rötlich.

```
member("MyYard").camera("BayView").fog.color = rgb(255, 0, 0)
```

Siehe auch

fog

color (light)

Syntax

```
member(whichCastmember).light(whichLight).color
```

Beschreibung

Diese 3D-Lichteigenschaft gibt den RGB-Wert des Lichts an.

Der Standardwert dieser Eigenschaft ist `rgb(191,191,191)`.

Beispiel

Die folgende Anweisung setzt die Farbe des Lichts `RoomLight` auf `rgb(255, 0, 255)`.

```
member("Room").light("RoomLight").color = rgb(255,0,255)
```

Siehe auch

`fog`

colorBufferDepth

Syntax

```
getRendererServices().colorBufferDepth
```

Beschreibung

Diese 3D-Eigenschaft für `rendererServices` gibt die Farbgenauigkeit des Hardwareausgabepuffers auf dem Benutzersystem an. Je nach den Hardwareeinstellungen des Benutzers lautet dieser Wert 16 oder 32.

Diese Eigenschaft kann getestet, aber nicht eingestellt werden.

Beispiel

Die folgende Anweisung zeigt, dass der `colorBufferDepth`-Wert der Videokarte des Benutzers 32 lautet:

```
-- Lingo syntax
put getRendererServices().colorBufferDepth
-- 32

// JavaScript syntax
put (getRendererServices().colorBufferDepth);
// 32
```

Siehe auch

`getRendererServices()`, `getHardwareInfo()`, `depthBufferDepth`

colorDepth

Syntax

```
-- Lingo syntax
_system.colorDepth

// JavaScript syntax
_system.colorDepth;
```

Beschreibung

Diese Systemeigenschaft bestimmt die Farbtiefe des Computermonitors. Lesen/Schreiben.

- In Windows können Sie mit dieser Eigenschaft die Farbtiefe des Monitors prüfen und einstellen. Die Eigenschaft `colorDepth` kann mit bestimmten Videokarten- und Videotreiberkombinationen unter Umständen nicht eingestellt werden. Prüfen Sie deshalb nach jeder Änderung der Farbtiefe, ob sich die Einstellung tatsächlich geändert hat.
- Auf dem Mac können Sie mit dieser Eigenschaft die Farbtiefe verschiedener Bildschirme prüfen und entsprechende Änderungen vornehmen.

Folgende Werte sind möglich:

`1`	Schwarzweiß
`2`	4 Farben
`4`	16 Farben
`8`	256 Farben
`16`	32 768 oder 65 536 Farben
`32`	16 777 216 Farben

Wenn Sie eine Farbtiefe einzustellen versuchen, die der betreffende Monitor nicht unterstützt, ändert sich die Farbtiefe des Monitors nicht.

Bei Computern mit mehreren Bildschirmen verweist die Eigenschaft `colorDepth` auf den Bildschirm, der die Bühne anzeigt. Wenn sich die Bühne über mehrere Bildschirme hinweg erstreckt, gibt die Eigenschaft `colorDepth` die größte Farbtiefe dieser Bildschirme an und versucht, alle Bildschirme auf die angegebene Tiefeeinzustellen.

Beispiel

Die folgende Anweisung weist Director an, den Film Vollfarben nur dann zu öffnen, wenn die Farbtiefe des Bildschirms auf 256 Farben eingestellt ist:

```
-- Lingo syntax
if (_system.colorDepth = 8) then
    window("Full color").open()
end if

// JavaScript syntax
if (_system.colorDepth == 8) {
    window("Full color").open()
}
```

Die folgende Prozedur versucht, die Farbtiefe zu ändern, und zeigt einen Warnhinweis an, falls dies nicht möglich ist:

```
-- Lingo syntax
on tryToSetColorDepth(desiredDepth)
    _system.colorDepth = desiredDepth
    if (_system.colorDepth = desiredDepth) then
        return true
    else
        _player.alert("Please change your system to" && desiredDepth && "color depth and reboot.")
        return false
    end if
end

// JavaScript syntax
function tryToSetColorDepth(desiredDepth) {
    _system.colorDepth = desiredDepth;
    if (_system.colorDepth == desiredDepth) {
        return true;
    }
    else {
        _player.alert("Please change your system to " + desiredDepth + " color depth and reboot.");
        return false;
    }
}
```

Siehe auch

System

colorList

Syntax

```
member(whichCastmember).modelResource(whichModelResource).colorList
member(whichCastmember).modelResource(whichModelResource).colorList[index]
member(whichCastmember).model(whichModel).meshdeform.mesh[meshIndex].colorList
member(whichCastmember).model(whichModel).meshdeform.mesh[meshIndex].colorList[index]
```

Beschreibung

Mit dieser 3D-Eigenschaft können Sie alle in einem Gitternetz verwendeten Farben ermitteln und festlegen. Dieser Befehl ist nur für Modellressourcen vom Typ `#mesh` verfügbar. Jede Farbe kann von mehreren Scheitelpunkten (Flächen) des Gitternetzes gemeinsam verwendet werden. Sie können auch Texturkoordinaten für die Gitternetzseiten angeben und auf Modelle, die diese Modellressource nutzen, einen Shader anwenden.

Dieser Befehl muss auf eine Liste gesetzt werden, die so viele Lingo-Farbwerte enthält, wie im Aufruf von newMesh angegeben sind.

Beispiel

Die folgende Anweisung zeigt, dass die dritte Farbe in der `colorList` der Modellressource "Gitternetz2" "rgb(255, 0, 0)" ist:

```
-- Lingo syntax
put member("shapes").modelResource("mesh2").colorlist[3]
-- rgb(255,0,0)

// JavaScript syntax
put ( member("shapes").getProp("modelResource" , 1).colorlist[3] );
// color(255,0,0)
```

Siehe auch

face[ ], colors

colorRange

Syntax

```
member(whichCastmember).modelResource(whichModelResource).colorRange.start
member(whichCastmember).modelResource(whichModelResource).colorRange.end
```

Beschreibung

Diese 3D-Eigenschaften für Modellressourcen vom Typ `#particle` geben die Anfangs- und Endfarbe der Partikel eines Partikelsystems an.

Die Eigenschaft `start` gibt die Farbe der Partikel bei ihrer Erstellung an. Die Eigenschaft `end` gibt die Farbe der Partikel am Ende ihrer Lebensdauer an. Die Farbe der Partikel ändert sich allmählich vom `start`-Wert in den `end`-Endwert.

Der Standardwert der Eigenschaften `start` und `end` lautet `rgb(255, 255, 255)`.

Beispiel

Die folgende Anweisung legt die `colorRange`-Eigenschaften der Modellressource `ThermoSystem` fest. Die erste Zeile setzt den "start"-Wert auf rgb(255, 0, 0) und die zweite Zeile den "end"-Wert auf rgb(0, 0, 255). Diese Anweisung bewirkt Folgendes: Die Partikel von `ThermoSystem` sind anfänglich rot, ihre Farbe ändert sich im Laufe ihrer Lebensdauer jedoch allmählich in blau.

```
member(8,2).modelResource("ThermoSystem").colorRange.start =rgb(255,0,0)
member(8,2).modelResource("ThermoSystem").colorRange.end = rgb(0,0,255)
```

Siehe auch

emitter, blendRange, sizeRange

colors

Syntax

```
member(whichCastmember).modelResource(whichModelResource).face[faceIndex].colors
```

Beschreibung

Diese 3D-Flächeneigenschaft ist eine lineare Liste aus drei Ganzzahlen, die angeben, welche Indexpositionen in der Farbliste der Modellressource für die drei Scheitelpunkte der Fläche verwendet werden sollen. Die Farbliste ist eine lineare Liste aus `rgb`-Werten.

Die Eigenschaft `colors` wird nur bei Modellressourcen vom Typ `#mesh` verwendet.

Nach Einstellung dieser Eigenschaft müssen Sie den Modellressourcenbefehl `build()` verwenden, da die Änderungen andernfalls nicht in Kraft treten.

Beispiel

In diesem Beispiel werden eine Modellressource vom Typ `#mesh` erstellt, ihre Eigenschaften festgelegt und anhand der Modellressource dann ein neues Modell erstellt.

In Zeile 1 wird der Befehl `newMesh()` zur Erstellung einer `#mesh`-Modellressource namens „Dreieck“ verwendet, die eine Fläche, drei Scheitelpunkte und maximal drei Farben aufweist. Die Anzahl von Normalen und die Anzahl von Texturkoordinaten sind nicht festgelegt.

In Zeile 2 wird die Eigenschaft `vertexList` auf eine Liste mit drei Vektoren gesetzt.

In Zeile 3 werden die Vektoren der Eigenschaft `vertexList` den Scheitelpunkten der ersten Fläche des Gitternetzes "Dreieck" zugeordnet.

In Zeile 4 wird die Farbliste auf drei rgb-Werte gesetzt.

In Zeile 5 werden die Farben der ersten Fläche des Gitternetzes „Dreieck“ zugeordnet. Die dritte Farbe in der Farbliste wird auf den ersten Scheitelpunkt von "Dreieck" angewendet, die zweite Farbe auf den zweiten Scheitelpunkt und die erste Farbe auf den dritten Scheitelpunkt. Die Farben werden in Form von Verläufen über die erste Fläche von "Dreieck“ verteilt.

In Zeile 6 werden mit dem Befehl `generateNormals()` die Normalen für "Dreieck" erstellt.

In Zeile 7 wird der Befehl build() zur Gitternetzerstellung aufgerufen.

In Zeile 8 wird ein neues Modell namens `TriModel` erstellt, das das neue Gitternetz verwendet.

```
nm = member("Shapes").newMesh("Triangle",1,3,0,3,0)
nm.vertexList = [vector(0,0,0), vector(20,0,0), vector(20, 20, 0)]
nm.face[1].vertices = [1,2,3]
nm.colorList = [rgb(255,255,0), rgb(0, 255, 0), rgb(0,0,255)]
nm.face[1].colors = [3,2,1]
nm.generateNormals(#smooth)
nm.build()
nm = member("Shapes").newModel("TriModel", nm)
```

Siehe auch

face, vertices, vertices, flat

colorSteps

Syntax

```
member(whichCastmember).model(whichModel).toon.colorSteps
member(whichCastmember).model(whichModel).shader.colorSteps
member(whichCastmember).shader(whichShader).colorSteps
```

Beschreibung

Diese 3D-Eigenschaft für den Modifizierer `toon` und den Shader `painter` gibt an, wie viele Farben maximal vom Modifizierer `toon` bzw. dem Shader `painter` verwendet werden können. Diese Eigenschaft kann den Wert 2, 4, 8 oder 16 besitzen. Wenn Sie für `colorSteps` eine andere Zahl angeben, wird diese auf einen dieser Werte gerundet.

Der Standardwert ist 2.

Beispiel

Die folgende Anweisung beschränkt die Anzahl von Farben, die für den Modifizierer toon des Modells „Teekanne“ zur Verfügung stehen, auf 8. Die Teekanne wird mit also maximal acht Farben wiedergegeben.

```
-- Lingo syntax
member("shapes").model("Teapot").toon.colorSteps = 8

// JavaScript syntax
member("shapes").getProp("model", 1).getPropRef("toon").colorSteps = 8;
```

Siehe auch

highlightPercentage, shadowPercentage

commandDown

Syntax

```
-- Lingo syntax
_key.commandDown

// JavaScript syntax
_key.commandDown;
```

Beschreibung

Diese Tasteneigenschaft stellt fest, ob die Strg-Taste (Windows) oder die Befehlstaste (Mac) gedrückt wird. Nur Lesen.

Wird die Strg- oder Befehlstaste gedrückt, gibt die Eigenschaft `TRUE` zurück, andernfalls `FALSE`.

Sie können `commandDown` zusammen mit der Eigenschaft `key` verwenden, um festzustellen, ob die Strg- oder die Befehlstaste gleichzeitig mit einer anderen Taste gedrückt wird. Auf diese Weise können Sie Prozeduren erstellen, die ausgeführt werden, wenn der Benutzer bestimmte Strg- oder Befehlstastenkombinationen drückt.

Die Strg- und Befehlstastenäquivalente für die Authoring-Menüs von Director haben beim Abspielen des Films Vorrang, es sei denn, Sie haben benutzerdefinierte Lingo- oder JavaScript-Syntax-Menüs installiert oder spielen den Film in einem Projektor ab.

Beispiel

Die folgenden Anweisungen halten einen Projektor an, wenn der Abspielkopf in ein Bild eintritt und der Benutzer Strg+A (Windows) oder Befehlstaste+A (Mac) drückt.

```
-- Lingo syntax
on enterFrame
    if (_key.commandDown and _key.key = "a") then
        _movie.go(_movie.frame)
    end if
end

// JavaScript syntax
function enterFrame() {
    if (_key.commandDown && _key.key == "a") {
        _movie.go(_movie.frame);
    }
}
```

Siehe auch

Taste, key

comments

Syntax

```
-- Lingo syntax
memberObjRef.comments

// JavaScript syntax
memberObjRef.comments;
```

Beschreibung

Diese Darstellereigenschaft ermöglicht das Speichern Ihrer Kommentare zu einem bestimmten Darsteller oder anderer Strings, die Sie dem Darsteller zuordnen möchten. Lesen/Schreiben.

Diese Eigenschaft kann auch im Eigenschafteninspektor auf der Registerkarte „Darsteller“ eingestellt werden.

Beispiel

Die folgende Anweisung stellt die Eigenschaft „comments“ des Darstellers „Hintergrund“ auf den String „Diese Grafik muss noch lizenziert werden“ ein:

```
-- Lingo syntax
member("Backdrop").comments = "Still need to license this artwork"

// JavaScript syntax
member("Backdrop").comments = "Still need to license this artwork";
```

Siehe auch

Member

compressed

Syntax

```
member(whichCastmember).texture(whichTexture).compressed
```

Beschreibung

Diese 3D-Textureigenschaft gibt an, ob der Quelldarsteller der Textur komprimiert ist (`TRUE`) oder nicht (`FALSE`). Der Wert der Eigenschaft `compressed` ändert sich automatisch von `TRUE` in `FALSE`, wenn die Textur zum Rendern benötigt wird. Der Wert kann auf `FALSE` gesetzt werden, wenn die Textur schon früher dekomprimiert werden soll. Der Wert kann auf `TRUE` gesetzt werden, um die dekomprimierte Darstellung aus dem Speicher zu entfernen. Für Texturen verwendete Darsteller werden nicht komprimiert, wenn dieser Wert `TRUE` lautet (abgesehen von der Standardkomprimierung, die beim Speichern eines Director-Films für Bitmapdarsteller verwendet wird). Der Standardwert dieser Eigenschaft lautet `TRUE`.

Beispiel

Die folgende Anweisung setzt die Eigenschaft compressed der Textur "Plutokarte" auf TRUE.

```
-- Lingo syntax
member("scene").texture("Plutomap").compressed = TRUE

// Java Script
member("scene").getProp("texture",1).compressed = true;
```

Siehe auch

`texture`

connectionStatus (MP4Media/FLV)

Syntax

```
member1.connectionStatus
sprite1.connectionStatus
```

Beschreibung

Diese Eigenschaft für MP4Media/FLV-Darsteller oder Sprites liefert den Status der RTMP-Verbindung zurück, wenn die MP4-Ressource mit einer RTMP-Quelle verbunden wird.

`connectionStatus` liefert einen der folgenden Werte zurück:

- `#idle` – Das Sprite oder der Darsteller ist nicht mit dem Server verbunden. Lokale und HTTP-Quellen liefern stets diesen Status zurück.
- `#connecting` – Das Sprite oder der Darsteller ist dabei, sich mit dem Server zu verbinden. Der Status bleibt `#connecting`, falls die Serveradresse ungültig ist oder der Verbindungsversuch mit dem Server fehlschlägt.
- `#connected` – Das Sprite oder der Darsteller ist mit dem Server verbunden. Der Status bleibt `#connected`, falls die Serveradresse gültig ist, aber das angefragte Video auf dem Server nicht verfügbar ist.
- `#ready` – Das Sprite oder der Darsteller sind zur Wiedergabe bereit.

Beispiel

```
-- Lingo syntax
put(sprite("MP4MediaSprite").connectionStatus
put(member("MP4Media/FLV").connectionStatus

// JavaScript syntax
put(sprite("MP4MediaSprite").connectionStatus
put(member("MP4Media/FLV").connectionStatus
```

connectionStatus (Soundobjekt)

Syntax

```
soundobj.connectionStatus (Read-only)
```

Beschreibung

Diese Soundobjekteigenschaft gibt den Status der RTMP-Verbindung aus, wenn das Soundobjekt eine RTMP-Quelle besitzt.

`connectionStatus` liefert einen der folgenden Werte zurück:

- `#idle`- Das Soundobjekt hat keine Verbindung mit dem Server. Lokale und verknüpfte Soundobjekte geben diesen Status aus, da keine Verbindung herzustellen ist.
- `#connecting`- Das Soundobjekt ist im Begriff eine Verbindung zum Server herzustellen. Der Status bleibt `#connecting`, falls die Serveradresse ungültig ist oder der Verbindungsversuch mit dem Server fehlschlägt.
- `#connected`- Das Soundobjekt ist mit dem Server verbunden. Der Status bleibt weiterhin `#connected`, wenn die Audiodatei nicht wiedergegeben werden kann oder nicht verfügbar ist.
- `#ready` - Das Soundobjekt kann jetzt wiedergegeben werden. Nach dem Anhalten eines Soundobjekts muss der Status `#ready` lauten, bevor das Soundobjekt erneut wiedergegeben werden kann.

Beispiele

Bei den folgenden Beispielen wird der `connectionStatus` des Soundobjekts ausgegeben.

```
--Lingo syntax
on mouseUp me
put SoundObjectRef.connectionStatus
end

// JavaScript syntax
function mouseUp(){
put (SoundObjectRef.connectionStatus);
}
```

constraint

Syntax

```
-- Lingo syntax
spriteObjRef.constraint

// JavaScript syntax
spriteObjRef.constraint;
```

Beschreibung

Diese Sprite-Eigenschaft bestimmt, ob das Registrierungskreuz eines Sprites auf das Begrenzungsrechteck eines anderen Sprites beschränkt ist (1 oder `TRUE`) oder nicht (0 oder `FALSE`, Standard). Lesen/Schreiben.

Mit der Eigenschaft `constraint` können Sie eine Spur für einen Schieberegler simulieren oder eine Begrenzung festsetzen, innerhalb welcher der Benutzer auf dem Bildschirm ein Objekt in einem Spiel ziehenkann.

Die Eigenschaft `constraint` wirkt sich auf ziehbare Sprites sowie die Eigenschaften `locH` und `locV` aus. Der Beschränkungspunkt eines verschiebbaren Sprites kann nicht außerhalb des Begrenzungsrechtecks des beschränkenden Sprites verschoben werden. (Der Beschränkungspunkt eines Bitmap-Sprites ist sein Registrierungskreuz. Der Beschränkungspunkt eines Form-Sprites ist seine linke obere Ecke.) Wenn eine Beschränkung für ein Sprite eingestellt ist, heben die Beschränkungswerte jegliche Einstellungen der Eigenschaften `locH` und `locV` auf.

Beispiel

Die folgende Anweisung entfernt die constraint-Eigenschaft eines Sprites.

```
-- Lingo syntax
sprite(5).constraint = 0

// JavaScript syntax
sprite(5).constraint = 0;
```

Die folgende Anweisung schränkt Sprite (i + 1) auf die Begrenzung von Sprite 14 ein:

```
-- Lingo syntax
sprite(i + 1).constraint = 14

// JavaScript syntax
sprite(i + 1).constraint = 14;
```

Die folgende Anweisung prüft, ob Sprite 3 beschränkt ist, und aktiviert die Prozedur `showConstraint`, wenn dies der Fall ist.

```
-- Lingo syntax
if (sprite(3).constraint <> 0) then
    showConstraint
end if

// JavaScript syntax
if (sprite(3).constraint != 0) {
    showConstraint();
}
```

Siehe auch

`locH`, `locV`, `Sprite`

controlDown

Syntax

```
-- Lingo syntax
_key.controlDown

// JavaScript syntax
_key.controlDown;
```

Beschreibung

Diese Tasteneigenschaft stellt fest, ob die Strg-Taste gedrückt wird. Nur Lesen.

Wird die Strg-Taste gedrückt, gibt die Eigenschaft `TRUE` zurück, andernfalls `FALSE`.

Sie können `controlDown` zusammen mit der Eigenschaft `key` verwenden, um festzustellen, ob die Strg-Taste gleichzeitig mit einer anderen Taste gedrückt wird. Auf diese Weise können Sie Prozeduren erstellen, die ausgeführt werden, wenn der Benutzer bestimmte Strg-Tastenkombinationen drückt.

Die Strg-Tastenäquivalente für die Authoring-Menüs von Director haben beim Abspielen des Films Vorrang, es sei denn, Sie haben benutzerdefinierte Lingo- oder JavaScript-Syntax-Menüs installiert oder spielen den Film in einem Projektor ab.

Beispiel

Die folgende on `keyDown`-Prozedur prüft, ob die gedrückte Taste die Strg- bzw. Ctrl-Taste ist, und aktiviert in diesem Fall die Prozedur `doControlKey`. Das Argument (`_key.key`) gibt an, welche Taste zusätzlich zur Strg-Taste gedrückt wurde.

```
-- Lingo syntax
on keyDown
    if (_key.controlDown) then
        doControlKey(_key.key)
    end if
end

on doControlKey(theKey)
    trace("The " & theKey & " key is down")
end

// JavaScript syntax
function keyDown() {
    if (_key.controlDown) {
        doControlKey(_key.key);
    }
}

function doControlKey(theKey) {
    trace("The " & theKey & " key is down");
}
```

Siehe auch

Taste, key

controller

Syntax

```
member(whichCastMember).controller
the controller of member whichCastMember
```

Beschreibung

Diese Digitalvideo-Darstellereigenschaft bestimmt, ob ein Digitalvideo-Darsteller seinen Regler anzeigt oder nicht. Wenn Sie diese Eigenschaft auf 1 einstellen, wird der Regler eingeblendet. Wird sie auf 0 eingestellt, wird der Regler ausgeblendet.

Die Darstellereigenschaft `controller` ist nur auf QuickTime®-Digitalvideos anwendbar.

- Beim Einstellen der Darstellereigenschaft `controller` für ein Video für Windows-Digitalvideo wird weder ein Vorgang ausgeführt noch werden Fehlermeldungen erzeugt.
- Beim Überprüfen der Darstellereigenschaft `controller` für ein Video für Windows-Digitalvideo wird immer der Wert `FALSE` zurückgegeben.

Das Digitalvideo muss im Abspielmodus "Direkt auf Bühne“ laufen, damit der Regler angezeigt werden kann.

Beispiel

Die folgende Anweisung bewirkt, dass der Regler des QuickTime-Darstellers „Demo“ angezeigt wird.

```
--Lingo Dot syntax:
member("Demo").controller = 1

--Lingo Verbose syntax:
set the controller of member "Demo" to 1

// JavaScript syntax
member("Demo").controller = 1;
```

Siehe auch

`directToStage`

copyrightInfo (Film)

Syntax

```
-- Lingo syntax
_movie.copyrightInfo

// JavaScript syntax
_movie.copyrightInfo;
```

Beschreibung

Diese Filmeigenschaft ist ein String, der bei der Filmerstellung im Dialogfeld „Filmeigenschaften“ eingegeben wird. Sie dient möglichen Funktionsverbesserungen in zukünftigen Versionen von ShockwavePlayer. Schreibgeschützt

Siehe auch

`aboutInfo`, `Movie`

copyrightInfo (SWA)

Syntax

```
-- Lingo syntax
memberObjRef.copyrightInfo

// JavaScript syntax
memberObjRef.copyrightInfo;
```

Beschreibung

Diese Shockwave Audio (SWA)-Darstellereigenschaft zeigt den Copyright-Text in einer SWA-Datei an. Diese Eigenschaft ist erst verfügbar, nachdem die Wiedergabe des SWA-Sounds begonnen hat oder nachdem die Datei mithilfe des Befehls `preLoadBuffer` vorausgeladen wurde.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Die folgende Anweisung bewirkt, dass Director die Copyright-Informationen der Shockwave Audio-Datei „SWAfile“ im Felddarsteller „Info-Anzeige“ anzeigt..

```
-- Lingo syntax
whatState = member("SWAfile").state
if whatState > 1 AND whatState < 9 then
    member("Info Display").text = member("SWAfile").copyrightInfo
end if

// JavaScript syntax
var whatState = member("SWAfile").state;
if (whatState > 1 && whatState < 9) {
    member("Info Display").text = member("SWAfile").copyrightInfo;
}
```

count

Syntax

```
list.count
count (list)
count(theObject)
object.count
textExpression.count
```

Beschreibung

Diese Eigenschaft (nur Lingo) gibt die Anzahl der Einträge in einer linearen oder Eigenschaftsliste, die Anzahl der Eigenschaften in einem Parent-Skript (ohne Eigenschaften des Ancestor-Skripts) oder die Anzahl der Chunks (Zeichen, Zeilen oder Wörter) in einem Textausdruck zurück.

Der Befehl `count` ist auf lineare und Eigenschaftslisten, auf Objekte, die mit einem Parent-Skript erstellt wurden, und auf die Eigenschaft `the globals` anwendbar.

Ein Beispiel für `count()` in einem fertigen Film finden Sie im Film "Text“ im Ordner „Learning/Lingo Examples“ im Director-Anwendungsordner.

Beispiel

Die folgende Anweisung zeigt die Zahl 3 als Anzahl der Einträge an:

```
put [10,20,30].count
-- 3
```

Siehe auch

globals

count (3D)

Syntax

```
member(whichCastmember).light.count
member(whichCastmember).camera.count
member(whichCastmember).modelResource(whichModelResource).bone.count
member(whichCastmember).model.count
member(whichCastmember).group.count
member(whichCastmember).shader.count
member(whichCastmember).texture.count
member(whichCastmember).modelResource.count
member(whichCastmember).motion.count
member(whichCastmember).light.child.count
member(whichCastmember).camera.child.count
member(whichCastmember).model.child.count
member(whichCastmember).group.child.count
sprite(whichSprite).camera{(index)}.backdrop.count
member(whichCastmember).camera(whichCamera).backdrop.count
sprite(whichSprite).camera{(index)}.overlay.count
member(whichCastmember).camera(whichCamera).overlay.count
member(whichCastmember).model(whichModel).modifier.count
member(whichCastmember).model(whichModel).keyframePlayer.playlist.count
member(whichCastmember).model(whichModel).bonesPlayer.playlist.count
member(whichCastmember).modelResource(whichModelResource).face.count
member(whichCastmember).model(whichModel).meshDeform.mesh[index].textureLayer.count
member(whichCastmember).model(whichModel).meshDeform.mesh.count
member(whichCastmember).model(whichModel).meshDeform.mesh[index].face.count
```

Beschreibung

Diese 3D-Eigenschaft gibt die Anzahl von Einträgen in der angegebenen Liste zurück, die mit dem angegebenen 3D-Objekt verknüpft ist. Sie kann bei jedem Objekttyp verwendet werden.

Mit der Eigenschaft `face.count` können Sie für eine Modellressource vom Typ `#mesh` die Anzahl von Polygonen im Gitternetz ermitteln.

Diese Eigenschaft kann getestet, aber nicht eingestellt werden.

Beispiel

In den folgenden Beispielen wird die Anzahl von Objekten des jeweiligen Typs in einem 3D-Darsteller namens "3D-Welt" ermittelt:

```
numberOfCameras = member("3D World").camera.count
put member("3D World").light.count
-- 3
numberOfModels = member("3D World").model.count
numberOfTextures = member("3D World").texture.count
put member("3D World").modelResource("mesh2").face.count
-- 4
```

Die folgende Anweisung zeigt, dass das erste Gitternetz des Modells „Ohr“ aus 58 Polygonen besteht:

```
put member("Scene").model("Ear").meshdeform.mesh[1].face.count
-- 58
```

Die folgende Anweisung zeigt, dass das Modell „Ohr“ aus drei Gitternetzen besteht:

```
put member("Scene").model("Ear").meshdeform.mesh.count
-- 3
```

Die folgende Anweisung zeigt, dass das erste Gitternetz des Modells „Ohr“ aus zwei Texturebenen besteht:

```
put member("Scene").model("Ear").meshdeform.mesh[1].textureLayer.count
-- 2
```

Siehe auch

cameraCount()

count (castLib)

Syntax

```
-- Lingo syntax
castObjRef.count

// JavaScript syntax
castObjRef.count;
```

Beschreibung

Gibt die Nummer der im Film enthaltenen Darstellerbibliothek zurück. Nur Lesen.

Beispiel

Folgende Anweisung verwendet ein Nachrichtenfenster, um die Nummer der Darstellerbibliothek im Film anzuzeigen:

```
-- Lingo syntax
  x = castLib(1)
  put x.count

// JavaScript syntax
  var cst = castLib(2);
  trace(cst.count);
```

Diese Eigenschaft lässt sich auch mit dem Ausdruck `_movie.castLib.count` abfragen.

Siehe auch

globals

cpuHogTicks

Syntax

```
the cpuHogTicks
```

Beschreibung

Diese Systemeigenschaft bestimmt, wie oft Director die Steuerung der CPU freigibt, damit der Computer Hintergrundereignisse (z. B. Ereignisse in anderen Anwendungen, Netzwerkereignisse, Aktualisierungen der Uhrzeit oder andere Tastaturereignisse) verarbeiten kann.

Der Standardwert ist 20 Ticks. Wenn Sie die CPU-Belegung von Director verlängern oder die Reaktion des Computers auf Netzwerkvorgänge steuern möchten, stellen Sie `cpuHogTicks` auf einen höheren Wert ein.

Wenn Sie die Tastenwiederholungsrate erhöhen und Animationen zugleich verlangsamen möchten, stellen Sie `cpuHogTicks` auf einen niedrigeren Wert ein. Wenn der Benutzer in einem Film eine Taste gedrückt hält, um das Tastatursignal in rascher Folge zu wiederholen, prüft Director die Auto-Tastenfunktion in der Regel seltener, als im Kontrollfeld festgelegt ist.

Die Eigenschaft `cpuHogTicks` funktioniert nur auf demMac.

Beispiel

Die folgende Anweisung bewirkt, dass Director die Steuerung der CPU alle 6 Ticks, also nach jeweils einer Zehntelsekunde freigibt:

```
the cpuHogTicks = 6
```

Siehe auch

`milliseconds`

creaseAngle

Syntax

```
member(whichCastmember).model(whichModel).inker.creaseAngle
member(whichCastmember).model(whichModel).toon.creaseAngle
```

Beschreibung

Diese 3D-Eigenschaft für die Modifizierer `inker` und `toon` gibt an, wie empfindlich die Linienzeichnungsfunktion des Modifizierers auf das Vorhandensein von Kanten in der Geometrie des Modells anspricht. Bei höheren Einstellungen werden an mehr Kanten Linien gezeichnet.

Die Eigenschaft `creases` des Modifizierers muss auf `TRUE` gesetzt sein, damit die Eigenschaft `creaseAngle` eine Wirkung hat.

`CreaseAngle` liegt im Bereich von -1.0 bis +1.0. Die Standardeinstellung ist 0.01.

Beispiel

Die folgende Anweisung setzt die Eigenschaft `creaseAngle` des Farbeffektmodifizierers für das Modell „Teapot“ auf 0.10 ein. Es wird eine Linie um alle Crease-Elemente des Modells gezogen, für welche der angegebene Schwellenwert überschritten wird. Diese Einstellung hat nur dann eine Wirkung, wenn die Eigenschaft creases des Modifizierers inker auf TRUE gesetzt ist.

```
-- Lingo syntax
member("shapes").model("Teapot").addModifier(#inker);
member("shapes").model("Teapot").inker.creaseAngle = 0.10

// JavaScript syntax
member("shapes").getProp("model",1).addModifier(symbol("inker"));
member("shapes").getProp("model",1).getPropRef("inker").creaseAngle = 0.10;
```

Siehe auch

`creases`, `lineColor`, `lineOffset`, `useLineOffset`

creases

Syntax

```
member(whichCastmember).model(whichModel).inker.creases
member(whichCastmember).model(whichModel).toon.creases
```

Beschreibung

Diese 3D-Eigenschaft für die Modifizierer `toon` und `inker` bestimmt, ob an Kanten in der Oberfläche des Modells Linien gezeichnet werden.

Die Standardeinstellung dieser Eigenschaft lautet `TRUE`.

Beispiel

Die folgende Anweisung setzt die Eigenschaft creases des auf das Modell "Teekanne" angewendeten Modifizierers inker auf TRUE. An allen Kanten im Modell, bei denen die durch die Eigenschaft `creaseAngle` des Modifizierers inker festgelegte Schwelle überschritten wird, wird eine Linie gezeichnet.

```
-- Lingo syntax
member("shapes").model("Teapot").addModifier(#inker);
member("shapes").model("Teapot").inker.creases = TRUE

// JavaScript syntax
member("shapes").getProp("model",1).addModifier(symbol("inker"));
member("shapes").getProp("model",1).getPropRef("inker").creases = true;
```

Siehe auch

`creaseAngle`, `lineColor`, `lineOffset`, `useLineOffset`

creationDate

Syntax

```
-- Lingo syntax
memberObjRef.creationDate

// JavaScript syntax
memberObjRef.creationDate;
```

Beschreibung

Diese Darstellereigenschaft zeichnet das Datum der ersten Erstellung des Darstellers unter Angabe des Systemdatums des Computers auf. Nur Lesen.

Diese Eigenschaft dient zur Terminierung von Projekten; Director selbst verwendet diese Eigenschaft nicht.

Beispiel

In der Regel wird die Eigenschaft `creationDate` im Eigenschafteninspektor oder in der Listenansicht des Besetzungsfensters abgefragt. Sie können Sie jedoch auch im Nachrichtenfenster überprüfen:

```
-- Lingo syntax
put(member(1).creationDate)

// JavaScript syntax
put(member(1).creationDate);
```

Siehe auch

Member

crop

Syntax

```
member(whichCastMember).crop
the crop of member whichCastMember
```

Beschreibung

Diese Darstellereigenschaft skaliert entweder einen Digitalvideo-Darsteller, damit er genau in das Sprite-Rechteck passt, in dem er angezeigt wird (FALSE), oder schneidet den Darsteller zu, ohne ihn zu skalieren, damit er in das Sprite-Rechteck passt (TRUE).

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Die folgende Anweisung veranlasst Director, alle Sprites zuzuschneiden, die auf den Digitalvideo-Darsteller "Interview“ verweisen.

```
-- Lingo Dot syntax:
member("Interview").crop = TRUE

-- Lingo Verbose syntax:
set the crop of member "Interview" to TRUE

// JavaScript syntax
member("Interview").crop = true;
```

Siehe auch

center

cuePointNames

Syntax

```
-- Lingo syntax
memberObjRef.cuePointNames

// JavaScript syntax
memberObjRef.cuePointNames;
```

Beschreibung

Diese Darstellereigenschaft erstellt eine Liste von Aufrufpunktnamen oder fügt, falls ein Aufrufpunkt keinen Namen besitzt, einen leeren String (`""`) als Platzhalter in die Liste ein. Aufrufpunktnamen dienen zum Synchronisieren von Sound, QuickTime und Animation.

Diese Eigenschaft wird von `SoundEdit`-Darstellern, QuickTime-Digitalvideo-Darstellern und Xtra-Darstellern unterstützt, in denen Aufrufpunkte enthalten sind. Xtra-Erweiterungen, die Aufrufpunkte zur Laufzeit erzeugen, können unter Umständen keine Aufrufpunktnamen auflisten.

Beispiel

Die folgende Anweisung ruft den Namen des dritten Aufrufpunkts eines Darstellers ab.

```
-- Lingo syntax
put member("symphony").cuePointNames[3]

// JavaScript syntax
put(member("symphony").cuePointNames[3]);
```

Siehe auch

`cuePointTimes`, `mostRecentCuePoint`

cuePointTimes

Syntax

```
-- Lingo syntax
memberObjRef.cuePointTimes

// JavaScript syntax
memberObjRef.cuePointTimes;
```

Beschreibung

Diese Darstellereigenschaft erstellt eine Liste mit den Aufrufpunktzeiten (in Millisekunden) eines bestimmten Darstellers. Aufrufpunktzeiten dienen zum Synchronisieren von Sound, QuickTime und Animation.

Diese Eigenschaft wird von `SoundEdit`-Darstellern, QuickTime-Digitalvideo-Darstellern und Xtra-Darstellern unterstützt, in denen Aufrufpunkte enthalten sind. Xtra-Erweiterungen, die Aufrufpunkte zur Laufzeit erzeugen, können unter Umständen keine Aufrufpunktnamen auflisten.

Beispiel

Die folgende Anweisung ruft die Zeit des dritten Aufrufpunkts für einen Sounddarsteller ab.

```
-- Lingo syntax
put member("symphony").cuePointTimes[3]

// JavaScript syntax
put(member("symphony").cuePointTimes[3]);
```

Siehe auch

`cuePointNames`, `mostRecentCuePoint`

currentLoopState

Syntax

```
member(whichCastmember).model(whichModel).keyframePlayer.currentLoopState
member(whichCastmember).model(whichModel).bonesPlayer.currentLoopState
```

Beschreibung

Diese 3D-Eigenschaft für die Modifizierer `keyframePlayer` und `bonesPlayer` gibt an, ob die durch das Modell ausgeführte Bewegung kontinuierlich wiederholt (`TRUE`) oder fertig abgespielt und dann durch die nächste Bewegung in der Abspielliste des Modifizierers ersetzt wird (`FALSE`).

Die Standardeinstellung dieser Eigenschaft ist der Wert des Parameters „looped“ des Befehls `play()`, der die Wiedergabe der Bewegung eingeleitet hat, bzw. der Wert des Befehls `queue()`, durch den die Bewegung zur Abspielliste des Modifizierers hinzugefügt wurde. Bei Änderung der Eigenschaft `currentLoopState` ändert sich auch der Wert der Eigenschaft `#looped` des zur Bewegung gehörenden Eintrags in der Abspielliste des Modifizierers.

Beispiel

Die folgende Anweisung bewirkt, dass die durch das Modell "Monster" ausgeführte Bewegung kontinuierlich wiederholt wird:

```
-- Lingo syntax
member("NewAlien").model("Monster").addModifier(#keyframeplayer)
member("NewAlien").model("Monster").keyframePlayer.currentLoopState = TRUE

// JavaScript syntax
member("NewAlien").getPropRef("model",1).addModifier(symbol("keyframeplayer"));
member("NewAlien").getProp("model",1).getPropRef("keyframePlayer").currentLoopState = true;
```

Siehe auch

`loop (3D)`, `play() (3D)`, `queue() (3D)`, `playlist`

currentSpriteNum

Syntax

```
-- Lingo syntax
_player.currentSpriteNum

// JavaScript syntax
_player.currentSpriteNum;
```

Beschreibung

Diese Player-Eigenschaft gibt die Kanalnummer des Sprites an, dessen Skript zurzeit abgespielt wird. Nur Lesen.

Diese Eigenschaft ist in Verhalten und Darstellerskripten zulässig. Wird sie in Bild- oder Filmskripten verwendet, ist der Wert der Eigenschaft `currentSpriteNum` gleich 0.

Die Eigenschaft `currentSpriteNum` ähnelt der Eigenschaft `spriteNum` des Sprite-Objekts.

Hinweis: *Diese Eigenschaft war in erster Linie während des Versionswechsels zu Director 6 von Nutzen, als Verhalten eingeführt wurden. Sie ermöglichte eine gewisse verhaltensähnliche Funktionalität, ohne dass Skriptcode komplett neu geschrieben werden musste. Wenn bei der Filmerstellung Verhalten verwendet werden, ist die Eigenschaft überflüssig und wird daher im Grunde nicht mehr benötigt.*

Beispiel

Die folgende Prozedur in einem Darsteller- oder Filmskript tauscht den Darsteller aus, der dem vom `mouseDown`-Ereignis betroffenen Sprite zugeordnet ist:

```
-- Lingo syntax
on mouseDown
    sprite(_player.currentSpriteNum).member = member("DownPict")
end

// JavaScript syntax
function mouseDown() {
    sprite(_player.currentSpriteNum).member = member("DownPict");
}
```

Siehe auch

`Player`, `spriteNum`

currentTime (3D)

Syntax

```
member(whichCastmember).model(whichModel).keyframePlayer.currentTime
member(whichCastmember).model(whichModel).bonesPlayer.currentTime
```

Beschreibung

Diese 3D-Eigenschaft der Modifizierer `keyframePlayer` und `bonesPlayer` gibt die lokale Zeit der vom Modell ausgeführten Bewegung an. Die Eigenschaft `currentTime` wird in Millisekunden gemessen und entspricht nur dann der Echtzeit, wenn die Bewegung mit der Originalgeschwindigkeit wiedergegeben wird.

Die Wiedergabe einer Bewegung durch ein Modell wird durch den Befehl `play()` oder `queue()` ausgelöst. Der Parameter `scale` des Befehls `play()` oder `queue()` wird mit der Modifiziereigenschaft `playRate` multipliziert. Das Ergebnis wird mit der Originalgeschwindigkeit der Bewegung multipliziert, um zu ermitteln, wie schnell das Modell die Bewegung ausführen und wie schnell die lokale Zeit der Bewegung laufen soll. Wenn der Parameter `scale` also den Wert 2 und die Modifiziereigenschaft `playRate` den Wert 3 aufweist, führt das Modell die Bewegung sechsmal so schnell wie mit der Originalgeschwindigkeit aus und die lokale Zeit läuft sechsmal so schnell wie die Echtzeit.

Die Eigenschaft `currentTime` wird zu Beginn jeder Iteration einer in Schleife laufenden Bewegung auf den Wert des Parameters `cropStart` des Befehls `play()` bzw. `queue()` zurückgesetzt.

Beispiel

Die folgende Anweisung zeigt die lokale Zeit der vom Modell "Kreatur3" ausgeführten Bewegung an.

```
-- Lingo syntax
member("NewAlien").model("Alien3").addModifier(#keyframeplayer)
put member("NewAlien").model("Alien3").keyframePlayer.play()
put member("NewAlien").model("Alien3").keyframePlayer.currentTime
-- 1393.8599

// JavaScript syntax
member("NewAlien").getPropRef("model",1).addModifier(symbol("keyframeplayer"));
member("NewAlien").getProp("model",1).getPropRef("keyframePlayer").play();
put(member("NewAlien").getProp("model",1).getPropRef("keyframePlayer").currentTime );
// 1393.8599
```

Siehe auch

play() (3D), queue() (3D), playlist

currentTime (DVD)

Syntax

```
-- Lingo syntax
dvdObjRef.currentTime

// JavaScript syntax
dvdObjRef.currentTime;
```

Beschreibung

Diese DVD-Eigenschaft gibt die verstrichene Zeit in Millisekunden zurück. Lesen/Schreiben.

Beispiel

Die folgende Anweisung gibt die verstrichene Zeit zurück:

```
-- Lingo syntax
trace (member(1).currentTime)-- 11500

// JavaScript syntax
trace (member(1).currentTime);// 11500
```

Die folgende Anweisung legt `currentTime` auf einen bestimmten Punkt im aktuellen Titel fest:

```
-- Lingo syntax
member(1).currentTime = 22000

// JavaScript syntax
member(1).currentTime = 22000
```

Siehe auch

DVD

currentTime (QuickTime, AVI)

Syntax

```
-- Lingo syntax
spriteObjRef.currentTime

// JavaScript syntax
spriteObjRef.currentTime;
```

Beschreibung

Diese Eigenschaft von Digitalvideosprites gibt die aktuelle Zeit (in Millisekunden) für ein Quicktime-, AVI- oder MP4-Videosprite zurück. Die Eigenschaft „currentTime“ ersetzt „movieTime“ für QuickTime und AVI.

Diese Eigenschaft lässt sich für QuickTime und AVI lesen und einstellen und für MP4 nur lesen. Die Methode „Seek“ wird verwendet, um die Abspielzeit für MP4 einzustellen.

Ein Beispiel für `currentTime` in einem fertigen Film finden Sie im Film „QT and Flash“ im Ordner „Learning/Lingo_Examples“ im Director-Anwendungsordner.

Beispiel

Die folgende Anweisung zeigt die aktuelle Zeit des QuickTime-Films in Kanal 9 im Nachrichtenfenster an:

```
-- Lingo syntax
put(sprite(9).currentTime)
// JavaScript syntax
put(sprite(9).currentTime);
```

Die folgende Anweisung stellt die aktuelle Zeit des QuickTime-Films in Kanal 9 auf 2 Sekunden sein.

```
-- Lingo syntax
sprite(9).currentTime = 2000
// JavaScript syntax
sprite(9).currentTime = 2000;
```

Siehe auch

duration (Darsteller)

currentTime (MP4Media/FLV)

Syntax

```
put sprite(1).currentTime
put member(1).currentTime
```

Beschreibung

Diese MP4Media/FLV-Darsteller- oder Spriteeigenschaft gibt die aktuelle Spielzeit eines MP4Media/FLV-Darstellers oder -Sprites während der Wiedergabe zurück.

`currentTime` nimmt einen ganzzahligen Wert in Millisekunden an, der von 0 bis zur Gesamtdauer des Videodarstellers reichen kann.

- `sprite.play()` Video und Soundtrack werden wiedergegeben und `sprite.currentTime` wird aktualisiert.

- `member.play()` Nur der Soundtrack wird wiedergegeben und `member.currentTime` und `member.mixer.soundObjectList[1].currentTime` werden aktualisiert. Die Aktualisierung dauert ca. 3 Millisekunden.
- `sprite.currentTime` ist eine ganze Zahl und `member.currentTime` ist eine Fließkommazahl.

Beispiele

Folgende Beispiele geben die aktuelle Spielzeit des MP4-Videos zurück:

```
-- Lingo syntax
put sprite(7).currentTime   -- 3000
put member("MP4Media/FLV").currentTime -- 3000

// JavaScript syntax
put sprite(7).currentTime  // 3000
put member("MP4Media/FLV").currentTime  // 3000
```

currentTime (RealMedia)

Syntax

```
-- Lingo syntax
memberOrSpriteObjRef.currentTime

// JavaScript syntax
memberOrSpriteObjRef.currentTime;
```

Beschreibung

Mit dieser Eigenschaft für `RealMedia`-Sprites und -Darsteller können Sie die aktuelle Zeit des `RealMedia`-Streams einstellen und abrufen (in Millisekunden). Wenn der `RealMedia` gegenwärtig nicht abgespielt wird, ist der Wert dieser Eigenschaft 0 (Standardeinstellung). Diese Wiedergabeeigenschaft wird nicht gespeichert.

Wenn der Stream beim Einstellen oder Ändern der Eigenschaft `currentTime` abgespielt wird, findet eine Suche statt, der Stream wird neu gepuffert und die Wiedergabe an der neuen Stelle fortgesetzt. Wenn der Stream beim Einstellen oder Ändern der Eigenschaft `currentTime` angehalten ist (Wert `#paused mediaStatus`), wird das Bild an der neuen Stelle aktualisiert und die Wiedergabe fortgesetzt, sofern `pausedAtStart` auf `FALSE` gesetzt ist. Wenn der Stream im `RealMedia`-Viewer angehalten oder gestoppt ist, weist `mediaStatus` den Wert `#paused` auf. Wenn der Stream mit dem Lingo-Befehl `stop` gestoppt wird, besitzt `mediaStatus` den Wert `#closed`. Diese Eigenschaft hat keine Wirkung, wenn die Streameigenschaft `mediaStatus` auf `#closed` gesetzt ist. Wenn Sie Ganzzahlen angeben, werden sie auf den Bereich zwischen 0 und der Streamdauer zugeschnitten.

Das Einstellen von `currentTime` entspricht dem Aufruf des Befehls `seek`: `x.seek(n)` ist das Gleiche wie `x.currentTime = n`. Wenn Sie `currentTime` ändern oder `seek` aufrufen, muss der Stream neu gepuffert werden.

Beispiel

Die folgenden Beispiele zeigen, dass die aktuelle Wiedergabezeit von Sprite 2 und des Darstellers "Real“ 15.534 Millisekunden (15,534 Sekunden) seit Streambeginn ist:

```
-- Lingo syntax
put(sprite(2).currentTime) -- 15534
put(member("Real").currentTime) -- 15534

// JavaScript syntax
put(sprite(2).currentTime) // 15534
put(member("Real").currentTime) // 15534
```

In den folgenden Beispielen wird die Wiedergabezeit auf 20.000 Millisekunden (20 Sekunden) seit Beginn des Streams von Sprite 2 und des Darstellers „Real“ gesetzt:

```
-- Lingo syntax
sprite(2).currentTime = 20000
member("Real").currentTime = 20000

// JavaScript syntax
sprite(2).currentTime = 20000
member("Real").currentTime = 20000
```

Siehe auch

duration (RealMedia, SWA), seek(), mediaStatus (RealMedia, Windows Media)

currentTime (Soundobjekt)

Syntax

```
soundObject.currentTime (Read-only)
```

Beschreibung

Diese Soundobjekteigenschaft gibt die abgelaufene Zeit (in Millisekunden) für die Wiedergabe des aktuellen Soundobjekts aus. Diese Eigenschaft ist schreibgeschützt.

Beispiel

```
--Lingo syntax
on mouseUp me
put soundObjRef.currentTime -- Gives the time, in milliseconds, that the
-- current sound object has been playing for.
end

// JavaScript syntax
function mouseUp(){
put (soundObjRef.currentTime);
}
```

currentTime (Sprite)

Syntax

```
-- Lingo syntax
spriteObjRef.currentTime

// JavaScript syntax
spriteObjRef.currentTime;
```

Beschreibung

Diese Sprite- und Soundkanal-Eigenschaft gibt die aktuelle Abspielzeit in Millisekunden für ein Sound-Sprite, ein QuickTime-Digitalvideo-Sprite oder eine beliebige Xtra-Erweiterung zurück, die Aufrufpunkte unterstützt. Im Falle eines Soundkanals wird die aktuelle Abspielzeit des Sounddarstellers im angegebenen Soundkanal zurückgegeben.

Diese Eigenschaft kann immer getestet, aber nur für herkömmliche Sounddarsteller (WAV, AIFF, SND) eingestellt werden. Der zulässige Wertebereich beim Einstellen dieser Eigenschaft liegt zwischen 0 und dem Wert von `duration` des Darstellers.

Shockwave Audio (SWA)-Sounds können in Sprite-Kanälen als Sprites erscheinen, die Soundwiedergabe erfolgt jedoch in einem Soundkanal. Es wird empfohlen, auf SWA-Sound-Sprites nicht mit der Soundkanalnummer, sondern mit der Nummer ihres Sprite-Kanals zu verweisen.

Beispiel

Die folgende Anweisung zeigt die aktuelle Zeit des Sound-Sprites im Sprite-Kanal 10 in Sekunden an.

```
-- Lingo syntax
member("time").text = string(sprite(10).currentTime/ 1000)

// JavaScript syntax
member("time").text = (sprite(10).currentTime / 1000).toString();
```

Die folgende Anweisung bewirkt, dass die Wiedergabe des Sounds in Soundkanal 2 an der Stelle 2,7 Sekunden nach Beginn des Sounddarstellers gestartet wird.

```
-- Lingo syntax
sound(2).currentTime = 2700

// JavaScript syntax
sound(2).currentTime = 2700;
```

Siehe auch

duration (Darsteller)

cursor

Syntax

```
-- Lingo syntax
spriteObjRef.cursor

// JavaScript syntax
spriteObjRef.cursor;
```

Beschreibung

Diese Sprite-Eigenschaft bestimmt, welcher Cursor verwendet wird, wenn sich der Mauszeiger über einem Sprite befindet. Lesen/Schreiben.

Diese Eigenschaft bleibt so lange in Kraft, bis Sie den Cursor auf0 stellen und sie somit ausschalten. Verwenden Sie die Eigenschaft `cursor`, um den Cursor zu ändern, wenn sich der Mauszeiger über einem bestimmten Bereich des Bildschirms befindet, und um Bereiche anzugeben, in denen bestimmte Aktionen möglich sind, wenn der Benutzer darauf klickt.

Wenn Sie die Eigenschaft `cursor` in einem bestimmten Bild einstellen, überwacht Director das Sprite-Rechteck, um festzustellen, ob der Cursor geändert werden soll. Dieses Rechteck bleibt weiterhin bestehen, wenn der Film zu einem anderen Bild übergeht, sofern Sie die Eigenschaft `cursor` für den betreffenden Kanal nicht auf 0 einstellen.

- Mit der folgenden Syntax können Sie die Nummer eines Darstellers, der als Cursor verwendet werden soll, und seine optionale Maske angeben.

```
-- Lingo syntax
spriteObjRef.cursor = [castMemberObjRef, maskCastMemberObjRef]

// JavaScript syntax
spriteObjRef.cursor = [castMemberObjRef, maskCastMemberObjRef];
```

- Verwenden Sie die folgende Syntax, um Standardsystemcursor anzugeben.

```
-- Lingo syntax
spriteObjRef.cursor = castMemberObjRef

// JavaScript syntax
spriteObjRef.cursor = castMemberObjRef;
```

Die Eigenschaft `cursor` kann auf einen der folgenden Ganzzahlwerte festgelegt werden:

Wert	Beschreibung
-1, 0	Pfeil
1	Einfügemarke
2	Kreuz
3	Querstange
4	Uhr (Mac) oder Sanduhr (Windows)
5	Norden Süden Osten Westen (NSOW)
6	Norden Süden (NS)
200	Leer (Cursor ausgeblendet)
254	Hilfe
256	Bleistift
257	Radiergummi
258	Auswählen
259	Farbeimer
260	Hand
261	Rechteck, Werkzeug

Wert	Beschreibung
262	Abgerundetes Rechteck, Werkzeug
263	Kreis, Werkzeug
264	Linie, Werkzeug
265	Richt-Text, Werkzeug
266	Textfeld, Werkzeug
267	Schaltfläche, Werkzeug
268	Kontrollkästchen, Werkzeug
269	Optionsfeld, Werkzeug
270	Platzierung, Werkzeug
271	Registrierungskreuz, Werkzeug
272	Lasso
280	Finger
281	Pipette
282	Warten, Maustaste gedrückt 1
283	Warten, Maustaste gedrückt 2
284	Vertikale Größe
285	Horizontale Größe
286	Diagonale Größe
290	Geschlossene Hand
291	Nicht Ablegen, Hand
292	Kopieren (geschlossene Hand)
293	Umgekehrter Pfeil
294	Drehen
295	Neigen
296	Horizontaler Doppelpfeil
297	Vertikaler Doppelpfeil
298	Südwest Nordost, Doppelpfeil
299	Nordwest Südost, Doppelpfeil
300	Verwischen/Weichzeichnen, Pinsel
301	Airbrush
302	Vergrößern
303	Verkleinern
304	Zoom abbrechen
305	Startform

Wert	Beschreibung
306	Punkt hinzufügen
307	Form schließen
308	Kamera zoomen
309	Kamera verschieben
310	Kamera drehen
457	Benutzerdefiniert

Zur Verwendung benutzerdefinierter Cursor stellen Sie die Eigenschaft `cursor` auf eine Liste ein, die den Darsteller enthält, der als Cursor verwendet werden soll, oder auf die Nummer, die einen Systemcursor angibt.Unter Windows können nur Darsteller und keine Ressourcen als Cursor verwendet werden. Wenn ein Cursor nicht verfügbarist,weil es sich bei ihm um eine Ressource handelt, zeigt Director stattdessen den Standard-Pfeilcursoran. Die besten Resultate erzielen Sie, wenn Sie zur Erstellung von plattformübergreifenden Filmen keine benutzerdefinierten Cursor verwenden.

Benutzerdefinierte Cursor-Darsteller dürfen nicht größer als 16 x 16 Pixel sein und müssen eine Tiefe von 1 Bit haben.

Falls das Sprite eine Bitmap ist, auf die der Farbeffekt "Matt“ angewendet wurde, ändert sich der Cursor nur, wenn er über dem Mattabschnitt des Sprites steht.

Wenn sich der Cursor über der Position eines Sprites befindet, das entfernt wurde, findet das Rollover nach wie vor statt. Sie können dieses Problem vermeiden, wenn Sie an diesen Stellen keine Rollover vornehmen oder das Sprite oberhalb der Menüleiste positionieren, bevor Sie es löschen.

<Check Alignment of PHs>Auf dem Mac können Sie eine nummerierte Cursor-Ressource in der aktuell geöffneten Filmdatei als Cursor verwenden, indem Sie `cursor` auf die Nummer der Cursor-Ressource festlegen.

Beispiel

Die folgende Anweisung ändert den Cursor, der über Sprite 20 steht, in einen Uhrcursor (Mac) oder einen Sanduhr-Cursor (Windows).

```
-- Lingo syntax
sprite(20).cursor = 4

// JavaScript syntax
sprite(20).cursor = 4;
```

Siehe auch

Sprite

cursorSize

Syntax

```
-- Lingo syntax
memberObjRef.cursorSize

// JavaScript syntax
memberObjRef.cursorSize;
```

Beschreibung

Diese Cursor-Darstellereigenschaft bestimmt die Größe des animierten Farbcursor-Darstellers `whichCursorCastMember`.

Größe:	Für Cursor bis:
`16`	16 x 16 Pixel
`32`	32 x 32 Pixel

Bitmapdarsteller, die kleiner als die angegebene Größe sind, werden in ihrer vollen Größe angezeigt. Größere Bitmapdarsteller werden proportional auf die angegebene Größe skaliert.

Der Standardwert ist 32 für Windows und 16 für den Mac. Wenn Sie einen ungültigen Wert einstellen, erscheint beim Abspielen (jedoch nicht beim Kompilieren) des Films eine Fehlermeldung.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Dieser Befehl ändert die Größe des animierten Farbcursors, der in Darsteller 20 gespeichert ist, in 32 x 32 Pixel.

```
-- Lingo syntax
member(20).cursorSize = 32

// JavaScript syntax
member(20).cursorSize = 32;
```

curve

Syntax

```
-- Lingo syntax
memberObjRef.curve[curveListIndex]

// JavaScript syntax
memberObjRef.curve[curveListIndex];
```

Beschreibung

Diese Eigenschaft enthält die `vertexList` einer einzelnen Kurve (Form) eines Vektorformdarstellers. In Verbindung mit der Eigenschaft `vertex` können Sie mit der Eigenschaft „curve“ einzelne Scheitelpunkte einer bestimmten Kurve in einer Vektorform ermitteln.

Der Listentyp `vertexList` ist eine Liste mit Scheitelpunkten von denen jeder eine Eigenschaftenliste mit bis zu drei Eigenschaften ist: einer `#vertex`-Eigenschaft mit der Position des Scheitelpunkts, einer `#handle1`-Eigenschaft mit der Position des ersten Steuerpunktes des Scheitelpunktes, und einer `#handle2`-Eigenschaft mit der Position des zweiten Steuerpunktes des Scheitelpunktes. Weitere Informationen finden Sie unter `vertexList`.

Beispiel

Die folgende Anweisung zeigt eine Beispielliste der Scheitelpunkte der dritten Kurve des Vektorformdarstellers `SimpleCurves` an:

```
-- Lingo syntax
put(member("SimpleCurves").curve[3])
-- [[#vertex: point(113.0000, 40.0000), #handle1: point(32.0000, 10.0000), #handle2: point(-
32.0000, -10.0000)], [#vertex: point(164.0000, 56.0000)]]

// JavaScript syntax
put(member("SimpleCurves").curve[3]);
// [[#vertex: point(113.0000, 40.0000), #handle1: point(32.0000, 10.0000),#handle2: point(-
32.0000, -10.0000)], [#vertex: point(164.0000, 56.0000)]]
```

Die folgende Anweisung verschiebt den ersten Scheitelpunkt der ersten Kurve in einer Vektorform um 10 Pixel nach unten und nach rechts:

```
-- Lingo syntax
member(1).curve[1].vertex[1] = member(1).curve[1].vertex[1] + point(10, 10)

// JavaScript syntax
member(1).curve[1].vertex[1] = member(1).curve[1].vertex[1] + point(10, 10);
```

Der folgende Code verschiebt ein Sprite an die Position des ersten Scheitelpunkts der ersten Kurve einer Vektorform. Zu diesem Zweck muss der `originMode` der Vektorform auf `#topLeft` eingestellt werden.

```
-- Lingo syntax
vertexLoc = member(1).curve[1].vertex[1]
spriteLoc = mapMemberToStage(sprite(3), vertexLoc)
sprite(7).loc = spriteLoc

// JavaScript syntax
var vertexLoc = member(1).curve[1].vertex[1];
var spriteLoc = mapMemberToStage(sprite(3), vertexLoc);
sprite(7).loc = spriteLoc;
```

debug

Syntax

```
member(whichCastmember).model(whichModel).debug
```

Beschreibung

Diese 3D-Modelleigenschaft gibt an, ob die Begrenzungskugel und die lokalen Achsen des Modells angezeigt werden.

Beispiel

Die folgende Anweisung setzt die Eigenschaft debug des Modells "Hund" auf TRUE.

```
-- Lingo syntax
member("ParkScene").model("Dog").debug = TRUE

// JavaScript syntax
member("ParkScene").getProp("model", 1).debug = true;
```

Siehe auch

boundingSphere

debugPlaybackEnabled

Syntax

```
-- Lingo syntax
_player.debugPlaybackEnabled

// JavaScript syntax
_player.debugPlaybackEnabled;
```

Beschreibung

Unter Windows öffnet diese Player-Eigenschaft in Shockwave und in Projektoren ein Nachrichtenfenster zum Debuggen. Auf dem Mac wird eine Protokolldatei generiert, damit `put`-Anweisungen Daten zum Debuggen ausgeben können. Lesen/Schreiben.

Unter Windows ist diese Eigenschaft bei Verwendung in der Director-Anwendung wirkungslos. Wenn das Nachrichtenfenster geschlossen wird, kann es in der jeweiligen Shockwave Player- oder Projektorsitzung nicht erneut geöffnet werden. Wenn mehrere Filme mit Shockwave-Inhalt dieses Skriptelement in ein und demselben Browser verwenden, wird nur durch den ersten Film ein Nachrichtenfenster geöffnet, das dann ausschließlich mit dem ersten Film verbunden ist.

Öffnen Sie auf dem Mac zum Anzeigen von Nachrichtenfenstereinträgen die Datei "1msgdump.txt" im Ordner "\Library\Logs\".

Um dieses Nachrichtenfenster zu öffnen, setzen Sie die Eigenschaft `debugPlaybackEnabled` auf `TRUE`. Um das Fenster zu schließen, setzen Sie die Eigenschaft `debugPlaybackEnabled` auf `FALSE`.

Beispiel

Die folgende Anweisung öffnet das Nachrichtenfenster im Shockwave Player oder einem Projektor:

```
-- Lingo syntax
_player.debugPlaybackEnabled = TRUE

// JavaScript syntax
_player.debugPlaybackEnabled = true;
```

Siehe auch

`Player`, `put()`

decayMode

Syntax

```
member(whichCastmember).camera(whichCamera).fog.decayMode
sprite(whichSprite).camera{(index)}.fog.decayMode
```

Beschreibung

Diese 3D-Eigenschaft gibt an, wie sich die Nebeldichte vom Mindest- bis zum Höchstwert erhöht, wenn die Kameraeigenschaft `fog.enabled` auf `TRUE` gesetzt ist.

Mögliche Werte für diese Eigenschaft:

- `#linear`: Die Nebeldichte wird zwischen `fog.near` und `fog.far` linear interpoliert.

- `#exponential`: `fog.far` ist der Sättigungspunkt; `fog.near` wird ignoriert.
- `#exponential2`: `fog.near` ist der Sättigungspunkt; `fog.far` wird ignoriert.

Der Standardwert dieser Eigenschaft lautet `#exponential`.

Beispiel

Die folgende Anweisung setzt die Eigenschaft `decayMode`des Nebels der Kamera `Defaultview` auf `#linear`. Wenn die Eigenschaft enabled auf `TRUE` gesetzt ist, nimmt die Nebeldichte zwischen den durch die Eigenschaften near und far festgelegten Punkten ständig zu. Ist die Eigenschaft near auf 100 und die Eigenschaft "far" auf 1000 gesetzt, beginnt der Nebel 100 Welteinheiten vor der Kamera und nimmt nach und nach an Dichte zu, bis eine Entfernung von 1000 Welteinheiten von der Kamera erreicht ist.

```
-- Lingo syntax
member("3d world").camera("Defaultview").fog.decayMode = #linear

// Java Script
member("3d world").getProp("camera",1).getPropRef("fog").decayMode = symbol("linear");
```

Siehe auch

`fog`, `near (fog)`, `far (fog)`, `enabled (fog)`

defaultRect

Syntax

```
-- Lingo syntax
memberObjRef.defaultRect

// JavaScript syntax
memberObjRef.defaultRect;
```

Beschreibung

Diese Darstellereigenschaft bestimmt die Standardgröße, die für alle neuen, aus einem Flash-Film- oder Vektorformdarsteller erstellten Sprites verwendet wird. Die Einstellung `defaultRect` wirkt sich auch auf alle vorhandenen Sprites aus, die nicht auf der Bühne gestreckt wurden. Sie geben die Eigenschaftswerte als ein Director-Rechteck an, zum Beispiel „rect(0,0,32,32)“.

Die Darstellereigenschaft `defaultRect` wird von der Darstellereigenschaft `defaultRectMode` des Darstellersbeeinflusst. Die Eigenschaft `defaultRectMode` wird immer auf `#Flash` eingestellt, wenn ein Film in eine Besetzung eingefügt wird. Dies bedeutet, dass die ursprüngliche `defaultRect`-Einstellung immer der Größe des ursprünglich in Flash erstellten Films entspricht. Wenn die Eigenschaft `defaultRect` später eingestellt wird, ändert sich die Eigenschaft `defaultRectMode` des Darstellers automatisch in `#fixed`.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Die folgende Prozedur akzeptiert einen Besetzungsbezug und ein Rechteck als Parameter. Sie durchsucht daraufhin die betreffende Besetzung nach Flash-Darstellern und stellt deren `defaultRect`-Eigenschaft auf das angegebene Rechteck ein.

```
-- Lingo syntax
on setDefaultFlashRect(whichCast, whichRect)
    repeat with i = 1 to castLib(whichCast).member.count
        if member(i, whichCast).type = #flash then
            member(i, whichCast).defaultRect = whichRect
        end if
    end repeat
end

// JavaScript syntax
function setDefaultFlashRect(whichCast, whichRect) {
    var i = 1;
    while( i < (castLib(whichCast).member.count) + 1)
        var tp = member(i, whichCast).type;
        if (tp == "flash") {
            member(i, whichCast).defaultRect = whichRect;
            i++;
        }
    }
}
```

Siehe auch

defaultRectMode, flashRect

defaultRectMode

Syntax

```
-- Lingo syntax
memberObjRef.defaultRectMode

// JavaScript syntax
memberObjRef.defaultRectMode;
```

Beschreibung

Diese Darstellereigenschaft bestimmt, wie die Standardgröße für alle neuen, aus Flash-Film- oder Vektorformdarstellern erstellten Sprites eingestellt wird. Sie geben den Eigenschaftswert als ein Director-Rechteck an, zum Beispiel „rect(0,0,32,32)“.

Die Eigenschaft `defaultRectMode` stellt nicht die tatsächliche Größe des Standardrechtecks eines Flash-Films ein. Sie bestimmt lediglich, wie das Standardrechteck eingestellt wird. Die Darstellereigenschaft `defaultRectMode` kann folgende Werte haben:

- `#flash` (Standard) – Stellt das Standardrechteck auf die Größe des ursprünglich in Flash erstellten Films ein.
- `#fixed` – Stellt das Standardrechteck mit der festen Größe ein, die in der Darstellereigenschaft `defaultRect` angegeben ist.

  Die Darstellereigenschaft `defaultRect` wird von der Darstellereigenschaft `defaultRectMode` des Darstellersbeeinflusst. Die Eigenschaft `defaultRectMode` wird immer auf `#flash` eingestellt, wenn ein Film in eine Besetzung eingefügt wird. Dies bedeutet, dass die ursprüngliche `defaultRect`-Einstellung immer der Größe des ursprünglich in Flash erstellten Films entspricht. Wenn die Eigenschaft `defaultRect` später eingestellt wird, ändert sich die Eigenschaft `defaultRectMode` des Darstellers automatisch in `#fixed`.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Die folgende Prozedur akzeptiert einen Besetzungsbezug und ein Rechteck als Parameter. Danach durchsucht sie die angegebene Besetzung nach Flash-Darstellern, stellt deren Eigenschaft `defaultRectMode` auf `#fixed` und die Eigenschaft `defaultRect` auf "rect(0,0,320,240)" ein.

```
-- Lingo syntax
on setDefaultRectSize(whichCast)
    repeat with i = 1 to castLib(whichCast).member.count
        if member(i, whichCast).type = #flash then
            member(i, whichCast).defaultRectMode = #fixed
            member(i, whichCast).defaultRect = rect(0,0,320,240)
        end if
    end repeat
end

// JavaScript syntax
function setDefaultRectSize(whichCast) {
    var i = 1;
    while( i < (castLib(whichCast).member.count) + 1)
        var tp = member(i, whichCast).type;
        if (tp == "flash") {
            member(i, whichCast).defaultRectMode = symbol("fixed");
            member(i, whichCast).defaultRect = rect(0,0,320,240);
            i++;
        }
    }
}
```

Siehe auch

`flashRect`, `defaultRect`

density

Syntax

```
member(whichCastmember).shader(whichShader).density
member(whichCastmember).model(whichModel).shader.density
member(whichCastmember).model(whichModel).shaderList{[index]}.density
```

Beschreibung

Diese 3D-Eigenschaft der Shader `#engraver` und `#newsprint` ändert die Anzahl von Zeilen oder Punkten, mit denen die Effekte dieser speziellen Shader-Typen erzeugt werden. Höhere Werte führen zu mehr Zeilen bzw. Punkten.

Bei Shadern von Typ `#engraver` ändert diese Eigenschaft die Anzahl von Zeilen, die zur Erstellung des Bilds verwendet werden. Der Wert kann zwischen 0 und 100 liegen; der Standardwert ist 40.

Bei Shadern von Typ `#newsprint` ändert diese Eigenschaft die Anzahl von Punkten, die zur Erstellung des Bilds verwendet werden. Der Wert kann zwischen 0 und 100 liegen; der Standardwert ist 45.

Beispiel

Die folgende Anweisung setzt die Eigenschaft density des Shaders `EngShader` auf 10. Die von diesem `#engraver`-Shader zur Erstellung eines stilisierten Bildes verwendeten Zeilen erscheinen grob und liegen weit auseinander.

```
-- Lingo syntax
member("scene").shader("EngShader").density = 10
// JavaScript syntax
member("scene").getProp("shader",1).density = 10;
```

Die folgende Anweisung setzt die Eigenschaft density des Shaders `gbShader` auf 100. Die von diesem `#newsprint`-Shader zur Erstellung eines stilisierten Bildes verwendeten Punkte sind sehr fein und liegen nah beieinander.

```
-- Lingo syntax
member("scene").shader("gbShader").density = 100
// JavaScript syntax
member("scene").getProp("shader", 2).density = 100;
```

Siehe auch

newShader

depth (3D)

Syntax

```
member(whichCastmember).model(whichModel).sds.depth
```

Beschreibung

Diese 3D-Eigenschaft für den Modifizierer `sds` (Subdivision Surfaces, Oberflächenunterteilungen) gibt an, wie viele Auflösungsstufen das Modell bei Verwendung des Modifizierers `sds` maximal anzeigen kann.

Wenn die Einstellungen `error` und `tension` des Modifizierers `sds` niedrig sind, ergibt sich bei Erhöhung der Eigenschaft `depth` eine stärkere Auswirkung auf die Geometrie des Modells.

Der Modifizierer `sds` kann nicht zusammen mit den Modifizierern `inker` und `toon` eingesetzt werden. Sie sollten außerdem vorsichtig sein, wenn Sie den Modifizierer `sds` in Verbindung mit dem Modifizierer `lod` verwenden.

Beispiel

Die folgende Anweisung setzt die Eigenschaft depth des Modifizierers sds für das Modell „Baby“ auf 3. Wenn die Einstellungen error und tension des Modifizierers sds niedrig sind, ergibt sich eine starke Auswirkung auf die Geometrie von „Baby“.

```
-- Lingo syntax
member("Scene").model("Baby").addModifier(#sds)
member("Scene").model("Baby").sds.depth = 3

// JavaScript syntax
member("Scene").getProp("model",2).addModifier(symbol("sds"));
member("Scene").getProp("model", 2).getPropRef("sds").depth = 3;
```

Siehe auch

sds (Modifizierer), error, tension

depth (Bitmap)

Syntax

```
imageObject.depth
member(whichCastMember).depth
the depth of member whichCastMember
```

Beschreibung

Diese Grafikobjekt- oder Bitmapdarstellereigenschaft zeigt die Farbtiefe des angegebenen Grafikobjekt- oder Bitmapdarstellers an.

Tiefe	Anzahl der Farben
1	Schwarzweiß
2	4 Farben
4, 8	16 oder 256 Farben bzw. Graustufen
16	Tausende von Farben
32	Millionen von Farben

Diese Eigenschaft kann getestet, aber nicht eingestellt werden.

Beispiel

Die folgende Anweisung zeigt die Farbtiefe des in der Variablen `newImage` gespeicherten Grafikobjekts an. Das Ergebnis wird im Nachrichtenfenster angezeigt.

```
-- Lingo syntax
put(newImage.depth)

// JavaScript syntax
trace(newImage.depth);
```

Die folgende Anweisung zeigt die Farbtiefe des Darstellers „Schrein“ im Nachrichtenfenster an:

```
-- Lingo syntax
put(member("Shrine").depth)

// JavaScript syntax
put(member("Shrine").depth);
```

depthBufferDepth

Syntax

```
getRendererServices().depthBufferDepth
```

Beschreibung

Diese 3D-Eigenschaft für `rendererServices` gibt die Genauigkeit des Hardwaretiefenpuffers auf dem Benutzersystem an. Je nach den Hardwareeinstellungen des Benutzers lautet dieser Wert 16 oder 24.

Beispiel

Die folgende Anweisung zeigt, dass der `depthBufferDepth`-Wert der Videokarte des Benutzers 16 lautet:

```
-- Lingo syntax
put getRendererServices().depthBufferDepth
-- 16

// JavaScript syntax
put (getRendererServices().depthBufferDepth);
// 16
```

Siehe auch

`getRendererServices()`, `getHardwareInfo()`, `colorBufferDepth`

deskTopRectList

Syntax

```
-- Lingo syntax
_system.deskTopRectList

// JavaScript syntax
_system.deskTopRectList;
```

Beschreibung

Diese Systemeigenschaft zeigt die Größe und Position aller an einen Computer angeschlossenen Monitore an. Nur Lesen.

Mit ihr können Sie prüfen, ob Objekte wie Fenster, Sprites und Popupfenster vollständig auf einem Bildschirm dargestellt werden.

Das Ergebnis ist eine Liste mit Rechtecken, in der jedes Rechteck die Begrenzung eines Monitors darstellt. Die Koordinaten der einzelnen Monitore beziehen sich auf die linke obere Ecke von Monitor 1 mit dem Wert (0,0). Der erste Koordinatensatz gibt die Größe des ersten Monitors an. Wenn ein zweiter Monitor vorhanden ist, gibt ein zweiter Koordinatensatz an, wo sich die Ecken des zweiten Monitors in Bezug auf den ersten Monitor befinden.

Beispiel

Die folgende Anweisung testet die Größe der an den Computer angeschlossenen Monitore und zeigt das Ergebnis im Nachrichtenfenster an:

```
-- Lingo syntax
put(_system.deskTopRectList)

// JavaScript syntax
put(_system.deskTopRectList);
```

Die folgende Prozedur gibt an, wie viele Monitore an das aktuelle System angeschlossen sind:

```
-- Lingo syntax
on countMonitors
    return _system.deskTopRectList
end

// JavaScript syntax
function countMonitors() {
    return _system.deskTopRectList;
}
```

Siehe auch

System

diffuse

Syntax

```
member(whichCastmember).shader(whichShader).diffuse
member(whichCastmember).model(whichModel).shader.diffuse
member(whichCastmember).model(whichModel).shaderList{[index]}.diffuse
```

Beschreibung

Diese 3D-Eigenschaft für den Shader `#standard` gibt eine Farbe an, die mit der ersten Textur des Shaders vermischt wird, wenn die folgenden Bedingungen erfüllt sind:

- Die Shader-Eigenschaft `useDiffuseWithTexture` ist auf `TRUE` gesetzt und
- die Shader-Eigenschaft `blendFunction` ist auf `#add` oder `#multiply` gesetzt oder
- die Shader-Eigenschaft `blendFunction` ist auf `#blend` und die Shader-Eigenschaft `blendSource` auf `#constant` gesetzt, und der Wert der Shader-Eigenschaft `blendConstant` liegt unter 100.

Der Standardwert dieser Eigenschaft ist `rgb( 255, 255, 255 )`.

Beispiel

Die folgende Anweisung setzt die Eigenschaft diffuse des Shaders "Globus" auf rgb(255, 0, 0):

```
-- Lingo syntax
member("MysteryWorld").shader("Globe").diffuse = rgb(255, 0, 0)

// JavaScript syntax
member("MysteryWorld").getProp("shader", 1).diffuse = color(255, 0, 0);
```

Siehe auch

diffuseColor, useDiffuseWithTexture, blendFunction, blendSource, blendConstant

diffuseColor

Syntax

```
member(whichCastmember).diffuseColor
```

Beschreibung

Diese 3D-Darstellereigenschaft gibt eine Farbe an, die mit der ersten Textur des ersten Shaders des Darstellers vermischt wird, wenn die folgenden Bedingungen erfüllt sind:

- Die Shader-Eigenschaft `useDiffuseWithTexture` ist auf `TRUE` gesetzt und
- die Shader-Eigenschaft `blendFunction` ist auf `#add` oder `#multiply` gesetzt oder
- die Shader-Eigenschaft `blendFunction` ist auf `#blend` und die Shader-Eigenschaft `blendSource` auf `#constant` gesetzt, und der Wert der Shader-Eigenschaft `blendConstant` liegt unter 100.

Der Standardwert der Eigenschaft `diffuseColor` ist `rgb( 255, 255, 255 )`.

Beispiel

Die folgende Anweisung setzt die Eigenschaft `diffuseColor` des Darstellers "Raum" auf rgb(255, 0, 0):

```
-- Lingo syntax
member("Room").diffuseColor = rgb(255, 0, 0)

// JavaScript syntax
member("Room").diffuseColor = color(255, 0, 0);
```

Siehe auch

`diffuse`, `useDiffuseWithTexture`, `blendFunction`, `blendSource`, `blendConstant`

diffuseLightMap

Syntax

```
member(whichCastmember).shader(whichShader).diffuseLightMap
member(whichCastmember).model(whichModel).shader.diffuseLightMap
member(whichCastmember).model(whichModel).shaderList{[index]}.diffuseLightMap
```

Beschreibung

Diese 3D-Eigenschaft für den Shader `#standard` gibt die Textur an, die zum Streulicht-Mapping verwendet wird.

Beim Einstellen dieser Eigenschaft werden die folgenden Eigenschaften automatisch gesetzt:

- Die zweite Texturebene des Shaders wird auf die angegebene Textur gesetzt.
- Der Wert von `textureModeList[2]` wird auf `#diffuse` gesetzt.
- Der Wert von `blendFunctionList[2]` wird auf `#multiply` gesetzt.
- Der Wert von `blendFunctionList[1]` wird auf `#replace` gesetzt.

Beispiel

Die folgende Anweisung legt die Textur „Oval“ für die Eigenschaft `diffuseLightMap` des vom Modell `GlassBox` verwendeten Shaders fest:

```
-- Lingo syntax
member("3DPlanet").model("GlassBox").shader.diffuseLightMap =
member("3DPlanet").texture("Oval")

// JavaScript syntax
member("3DPlanet").getProp("model", 1).shaderList[1].diffuseLightMap =
member("3DPlanet").getprop("texture", 1);
```

Siehe auch

blendFunctionList, textureModeList, glossMap, region, specularLightMap

digitalVideoTimeScale

Syntax

```
-- Lingo syntax
_player.digitalVideoTimeScale

// JavaScript syntax
_player.digitalVideoTimeScale;
```

Beschreibung

Diese Player-Eigenschaft legt die Zeitskala in Einheiten pro Sekunde fest, mit der das System Digitalvideo-Darsteller überwacht. Lesen/Schreiben.

Die Eigenschaft `digitalVideoTimeScale` kann auf jeden beliebigen Wert gesetzt werden.

Der Wert dieser Eigenschaft bestimmt, welcher Sekundenbruchteil als Zeitskala für das Video verwendet wird:

- 100 – Die Zeitskala beträgt 1/100 Sekunde (und der Film wird in 100 Einheiten pro Sekunde gemessen).
- 500 – Die Zeitskala beträgt 1/500 Sekunde (und der Film wird in 500 Einheiten pro Sekunde gemessen).
- 0 – Director verwendet die Zeitskala des laufenden Films.

Mit der Einstellung von `digitalVideoTimeScale` können Sie genau auf Spuren zugreifen, indem Sie gewährleisten, dass die Zeiteinheit für Video des Systems ein Mehrfaches der Zeiteinheit des Digitalvideos ist. Setzen Sie die Eigenschaft `digitalVideoTimeScale` auf einen höheren Wert, wenn Sie die Videowiedergabe genauer steuern möchten.

Beispiel

Die folgende Anweisung stellt die Zeitskala, mit der das System Digitalvideo misst, auf 600 Einheiten pro Sekunde ein:

```
-- Lingo syntax
_player.digitalVideoTimeScale = 600

// JavaScript syntax
_player.digitalVideoTimeScale = 600;
```

Siehe auch

Player

digitalVideoType

Syntax

```
member(whichCastMember).digitalVideoType
the digitalVideoType of member whichCastMember
```

Beschreibung

Diese Darstellereigenschaft zeigt das Format des angegebenen Digitalvideos an. Mögliche Werte: `#quickTime` oder `#videoForWindows`.

Diese Eigenschaft kann getestet, aber nicht eingestellt werden.

Beispiel

Die folgende Anweisung testet, ob es sich bei dem Darsteller "Tagesnachrichten“ um ein QuickTime- oder ein AVI (Audio-Video Interleaved)-Digitalvideo handelt, und zeigt das Ergebnis im Nachrichtenfenster an:

```
-- Lingo syntax
put member("Today's Events").digitalVideoType

// JavaScript syntax
put ( member("Today's Events").digitalVideoType );
```

Siehe auch

`QuickTimeVersion()`

direction

Syntax

```
member(whichCastmember).modelResource(whichModelResource).emitter.direction
```

Beschreibung

Diese 3D-Emittereigenschaft beschreibt einen Vektor, der die Richtung angibt, in der die Partikel eines Partikelsystems ausgestrahlt werden. Ein Partikelsystem ist eine Modellressource vom Typ `#particle`.

Die Hauptrichtung der Partikelemission ist der durch die Emittereigenschaft `direction` festgelegte Vektor. Die Emissionsrichtung eines bestimmten Partikels weicht jedoch um einen willkürlich gewählten Winkel zwischen 3 und dem Wert der Emittereigenschaft `angle (3D)` von diesem Vektor ab.

Wenn Sie die Eigenschaft `direction` auf `vector(0,0,0)` setzen, werden Partikel in alle Richtungen ausgestrahlt.

Der Standardwert dieser Eigenschaft lautet `vector(1,0,0)`.

Beispiel

Im folgenden Beispiel ist `ThermoSystem` eine Modellressource vom Typ `#particle`. Die folgende Anweisung setzt die Eigenschaft direction des Emitters von `ThermoSystem` auf vector(1, 0, 0), sodass die Partikel von `ThermoSystem` in einen kegelförmigen Bereich ausgestrahlt werden, dessen Achse die x-Achse der 3D-Welt ist.

```
-- Lingo syntax
member("Fires").modelResource("ThermoSystem").emitter.direction = vector(1,0,0)

// JavaScript syntax
member("Fires").getProp("modelResource", 1).getPropRef("emitter").direction = vector(1,0,0);
```

Siehe auch

emitter, angle (3D)

directionalColor

Syntax

```
member(whichCastmember).directionalColor
```

Beschreibung

Diese 3D-Darstellereigenschaft gibt die RGB-Farbe des Standardumgebungslichts des Darstellers an.

Der Standardwert dieser Eigenschaft ist „rgb(255, 255, 255)“.

Beispiel

Die folgende Anweisung setzt die Eigenschaft `directionalColor` des Darstellers „Raum“ auf @@@rgb(0, 255, 0). Das gerichtete Standardlicht des Darstellers ist grün. Diese Eigenschaft kann auch im Eigenschafteninspektor eingestellt werden.

```
-- Lingo syntax
member("Room").directionalcolor = rgb(0, 255, 0)

// JavaScript syntax
member("Room").directionalcolor = color(0, 255, 0);
```

Siehe auch

directionalPreset

directionalPreset

Syntax

```
member(whichCastmember).directionalPreset
```

Beschreibung

Diese 3D-Darstellereigenschaft gibt die Richtung an, aus der das gerichtete Standardlicht scheint, und zwar bezogen auf die Kamera des Sprites.

Wenn Sie den Wert dieser Eigenschaft ändern, ändern sich auch die Eigenschaften `position` und `rotation` der Transformation des Lichts.

Mögliche Werte für `directionalPreset`:

- `#topLeft`

- `#topCenter`
- `#topRight`
- `#middleLeft`
- `#middleCenter`
- `#middleRight`
- `#bottomLeft`
- `#bottomCenter`
- `#bottomRight`
- `#None`

Der Standardwert dieser Eigenschaft lautet `#topCenter`.

Beispiel

Die folgende Anweisung setzt die Eigenschaft `directionalPreset` des Darstellers "Raum" auf `#middleCenter`, Die Standardlichtquelle wird so ausgerichtet, dass das Standardlicht von "Raum" auf den Mittelpunkt der aktuellen Ansicht der Sprite-Kamera fällt. Diese Eigenschaft kann auch im Eigenschafteninspektor eingestellt werden.

```
-- Lingo syntax
member("Room").directionalpreset = #middleCenter

// JavaScript syntax
member("Room").directionalpreset = symbol("middleCenter");
```

Siehe auch

`directionalColor`

directToStage

Syntax

```
-- Lingo syntax
memberOrSpriteObjRef.directToStage

// JavaScript syntax
memberOrSpriteObjRef.directToStage;
```

Beschreibung

Diese Darsteller- und Sprite-Eigenschaft legt die Ebene fest, in der ein Digitalvideo-, animierter GIF-, Vektorform-, 3D-, Windows Media- oder Flash Asset-Darsteller auftritt.

Ist diese Eigenschaft auf `TRUE` (1) gesetzt, erscheint der Darsteller vor allen anderen Ebenen auf der Bühne, und Farbeffekte haben keine Wirkung.

Ist diese Eigenschaft auf `FALSE` (0) gesetzt, kann der Darsteller in jeder beliebigen Ebene der Animationsflächen der Bühne auftreten und Farbeffekte wirken sich auf das Aussehen des Sprites aus.

- Verwenden Sie die Syntax `member(`*whichCastMember*`).directToStage` für Digitalvideos oder animierte GIFs.
- Verwenden Sie die Syntax `sprite(`*whichSprite)*`.directToStage` für Flash oder Vektorformen.

- Für 3D-Darsteller oder -Sprites sind beide Syntaxformate zulässig.

Die Verwendung dieser Eigenschaft kann die Wiedergabeleistung des Darstellers oder Sprites verbessern.

Kein anderer Darsteller kann vor einem `directToStage`-Sprite erscheinen. Außerdem haben Farbeffekte keine Auswirkungen auf die Darstellung eines `directToStage`-Sprites.

Wenn die Sprite-Eigenschaft `directToStage` auf `TRUE` gesetzt ist, zeichnet Director das Sprite direkt auf dem Bildschirm, ohne es vorher im Offscreen-Puffer von Director zusammenzusetzen. Das Ergebnis kann dem Farbeffekt Spuren der Bühne ähneln.

Um einen Bereich mit Spuren ausdrücklich zu aktualisieren, schalten Sie die Eigenschaft `directToStage` aus und wieder ein, verwenden Sie einen Vollbild-Übergang oder "wischen“ Sie ein anderes Sprite über diesen Bereich. (Wenn Sie das in Windows nicht tun, können Sie zu einem ähnlichen Bild übergehen, wobei das Video u. U. nicht völlig verschwindet.)

Ein Beispiel für `directToStage` in einem fertigen Film finden Sie im Film „QT and Flash“ im Ordner „Learning\\Lingo_Examples“ im Director-Anwendungsordner.

Beispiel

Die folgende Anweisung bewirkt, dass der QuickTime-Film "Die Bewohner“ immer auf der obersten Ebene der Bühne abgespielt wird:

```
-- Lingo syntax
member("The Residents").directToStage = 1

// JavaScript syntax
member("The Residents").directToStage = 1;
```

directToStage (MP4Media/FLV)

Syntax

```
member(1).directToStage = true
put sprite(1).directToStage
```

Beschreibung

Darsteller- und Spriteeigenschaft, die der Wiedergabeebene eines MP4Media/FLV-Darstellers auf der Bühne entspricht. Diese Eigenschaft kann ausgelesen und beschrieben werden.

Falls `directToStage` den Wert „True“ (1) enthält, wird der Darsteller im Vordergrund vor allen anderen Ebenen der Bühne abgespielt. In diesem Fall wird das Aussehen des Sprites nicht von Farbeffekten beeinflusst.

Falls die Eigenschaft den Wert „False“ (0) enthält, darf der Darsteller in einer beliebigen Animationsebene der Bühne abgespielt werden. In diesem Fall kann das Aussehen des Sprites von Farbeffekten beeinflusst werden.

Beispiele

Diese Syntax veranlasst, dass der MP4Media/FLV-Darsteller stets auf vorderster Ebene der Bühne angezeigt wird:

```
-- Lingo syntax
member("MP4Media/FLV").directToStage = TRUE
put sprite("MP4Media/FLV").directToStage   -- 1

// JavaScript syntax
member("MP4Media/FLV").directToStage = true;
put sprite("MP4Media/FLV").directToStage   // 1
```

disableImagingTransformation

Syntax

```
-- Lingo syntax
_player.disableImagingTransformation

// JavaScript syntax
_player.disableImagingTransformation;
```

Beschreibung

Diese Player-Eigenschaft bestimmt, ob Director beim Erfassen des Bildes der Bühne automatisch Roll- und Zoomvorgänge auf der Bühne berücksichtigt. Lesen/Schreiben.

Ist `TRUE` gesetzt, verhindert diese Eigenschaft, dass Director beim Erfassen des Bildes der Bühne mithilfe der Eigenschaft `image` das Rollen bzw. Zoomen der Bühne automatisch berücksichtigt. Das Zoomen und Rollen der Bühne wirken sich auf das Aussehen des mit `image` erfassten Bildes aus.

Wenn die Eigenschaft auf `FALSE` gesetzt ist, erfasst Director das Bild der Bühne immer so, als sei das Bühnenfenster auf 100% gezoomt und als habe kein Bildlauf von der Mitte des Bühnenfensters aus stattgefunden. Die Standardeinstellung lautet `FALSE`.

Beispiel

Die folgende Anweisung setzt die Eigenschaft `disableImagingTransformation` auf TRUE:

```
-- Lingo syntax
_player.disableImagingTransformation = TRUE

// JavaScript syntax
_player.disableImagingTransformation = true;
```

Siehe auch

`image (Grafik)`, `Player`

displayFace

Syntax

```
member(whichTextCastmember).displayFace
member(which3DCastmember).modelResource(whichModelResource).displayFace
```

Beschreibung

Diese 3D-Texteigenschaft ist eine lineare Liste mit der bzw. den anzuzeigenden 3D-Textseite(n). Mögliche Werte: `#front`, `#tunnel` und `#back`. Sie können eine beliebige Seitenkombination anzeigen, und die Listeneinträge können in beliebiger Reihenfolge vorliegen.

Der Standardwert dieser Eigenschaft lautet `[#front, #back, #tunnel]`.

Bei Textdarstellern handelt es sich hierbei um eine Darstellereigenschaft. Bei extrudiertem Text in einem 3D-Darsteller handelt es sich um eine Modellressourceneigenschaft.

Beispiel

In diesem Beispiel ist der Darsteller "Logo“ ein Textdarsteller. Die folgende Anweisung setzt die Eigenschaft `displayFace` von „Schild“ auf `[#tunnel]`. Wenn "Logo“ im 3D-Modus angezeigt wird, erscheinen die Vorder- und Rückseite nicht.

```
-- Lingo syntax
member("Rugsign").displayFace = [#tunnel]

// JavaScript syntax
member("Rugsign").displayFace = list( symbol("tunnel") );
```

Im folgenden Beispiel ist die Modellressource des Modells "Slogan" extrudierter Text. Die folgende Anweisung setzt die Eigenschaft `displayFace` der Modellressource von „Slogan“ auf `[#back, #tunnel]`. Die Vorderseite von "Slogan“ wird nicht gezeichnet.

```
-- Lingo syntax
member("scene").model("Slogan").resource.displayFace = [#back, #tunnel]
```

Siehe auch

`extrude3D`, `displayMode`

displayMode

Syntax

```
member(whichTextCastmember).displayMode
```

Beschreibung

Diese Textdarstellereigenschaft gibt an, ob der Text als 2D- oder 3D-Text wiedergegeben wird.

Wenn diese Eigenschaft auf `#Mode3D` gesetzt ist, erscheint der Text in 3D. Sie können die 3D-Eigenschaften (wie z. B. `displayFace` und `bevelDepth`) des Texts sowie die üblichen Texteigenschaften (wie z. B. `text` und `font`) festlegen. Das Sprite mit diesem Darsteller wird zu einem 3D-Sprite.

Wenn diese Eigenschaft auf `#ModeNormal` gesetzt ist, erscheint der Text in 2D.

Der Standardwert dieser Eigenschaft lautet `#ModeNormal`.

Beispiel

In diesem Beispiel ist der Darsteller "Logo“ ein Textdarsteller. Die folgende Anweisung bewirkt, dass "Logo“ in 3D erscheint:

```
-- Lingo syntax
member("Logo").displayMode = #mode3D

// JavaScript syntax
member("Logo").displayMode = symbol("mode3D");
```

Siehe auch

extrude3D

displayRealLogo

Syntax

```
-- Lingo syntax
memberOrSpriteObjRef.displayRealLogo

// JavaScript syntax
memberOrSpriteObjRef.displayRealLogo;
```

Beschreibung

Mit dieser Eigenschaft für RealMedia-Sprites und -Darsteller können Sie ermitteln und festlegen, ob das RealNetworks®-Logo angezeigt wird (TRUE) oder nicht (FALSE). Wenn Sie diese Eigenschaft auf TRUE setzen, wird das Logo RealNetworks im RealMedia-Viewer am Anfang des Streams und beim Anhalten und Zurückspulen des Videos im angezeigt.

Der Standardwert dieser Eigenschaft lautet TRUE (1). Andere Ganzzahlen als 1 oder 0 werden als TRUE interpretiert.

Beispiel

Die folgenden Beispiele zeigen, dass die Eigenschaft displayRealLogo für Sprite 2 und den Darsteller „Real“ auf TRUE gesetzt ist, d. h. dass das RealNetworks-Logo bei Wiedergabebeginn und beim Anhalten und Zurückspulen des Videos angezeigt wird.

```
-- Lingo syntax
put(sprite(2).displayRealLogo) -- 1
put(member("Real").displayRealLogo) -- 1

// JavaScript syntax
trace(sprite(2).displayRealLogo); // 1
put(member("Real").displayRealLogo); // 1
```

In den folgenden Beispielen wird die Eigenschaft displayRealLogo für Sprite 2 und den Darsteller "Real" auf FALSE gesetzt, damit das RealNetworks-Logo nicht angezeigt wird.

```
-- Lingo syntax
sprite(2).displayRealLogo = FALSE
member("Real").displayRealLogo = FALSE

// JavaScript syntax
sprite(2).displayRealLogo = 0;
member("Real").displayRealLogo = 0;
```

displayTemplate

Syntax

```
-- Lingo syntax
_movie.displayTemplate

// JavaScript syntax
_movie.displayTemplate;
```

Beschreibung

Diese Filmeigenschaft bietet Zugriff auf eine Liste von Eigenschaften, die auf das Fenster angewendet werden, in dem ein Film wiedergegeben wird. Lesen/Schreiben.

Die Eigenschaft `displayTemplate` bietet Zugriff auf die Eigenschaften des Window-Objekts, mit deren Hilfe Standardfenstereinstellungen festgelegt werden. Deshalb wird `displayTemplate` auf dieselbe Weise zum Abrufen bzw. Festlegen von Standardfenstereinstellungen für das Movie-Objekt verwendet, wie die Eigenschaften `appearanceOptions` und `titlebarOptions` mit dem Window-Objekt verwendet werden.

Die Eigenschaft `displayTemplate` bietet Zugriff auf folgende Eigenschaften.

Eigenschaft	Beschreibung
`appearanceOptions`	Eine Eigenschaftsliste, die Darstellungsoptionen für ein Fenster speichert. Die Darstellungsoptionen lauten `mask`, `border`, `metal`, `dragRegionMask`, `shadow` und `liveresize`. Weitere Informationen finden Sie unter `appearanceOptions`.
`dockingEnabled`	Bestimmt, ob ein Film in einem Fenster (MIAW) andockbar ist, wenn er während der Erstellung geöffnet wird. Ist diese Eigenschaft `TRUE`, kann das Fenster angedockt werden. Ist diese Eigenschaft `FALSE`, kann das Fenster nicht angedockt werden. Der Standardwert ist `FALSE`. Weitere Informationen finden Sie unter `dockingEnabled`.
resizable	Bestimmt, ob ein Fenster in der Größe angepasst werden kann. Ist diese Eigenschaft `TRUE`, kann das Fenster in der Größe angepasst werden. Ist diese Eigenschaft `FALSE`, kann das Fenster nicht in der Größe angepasst werden. Der Standardwert ist `TRUE`. Weitere Informationen finden Sie unter `resizable`.
title	Gibt den Titel der Anzeigevorlage zurück oder legt ihn fest. Weitere Informationen finden Sie unter `title`.
`titlebarOptions`	Eine Eigenschaftsliste, die Titelleistenoptionen für ein Fenster speichert. Die Titelleistenoptionen lauten `icon`, `visible`, `closebox`, `minimizebox`, `maximizebox` und `sideTitlebar`. Weitere Informationen finden Sie unter `titlebarOptions`.
`systemTrayIcon`	(Nur Microsoft Windows) Bestimmt, ob einem Fenster ein Symbol im Infobereich der Taskleiste eines Benutzerdesktops zugeordnet ist.
`systemTrayTooltip`	(Nur Microsoft Windows) Bestimmt die Zeichenfolge, die im QuickInfo-Popup des Infobereichsymbols angezeigt wird.
type	Gibt den Typ eines Fensters zurück oder legt ihn fest. Durch das Festlegen des Typs eines Fensters werden alle zu diesem Fenstertyp gehörenden Eigenschaften ebenfalls festgelegt. Die Fenstertypen lauten `tool`, `document` und `dialog`. Weitere Informationen finden Sie unter `type`.

Beispiel

Die folgenden Anweisungen zeigen die `displayTemplate`-Eigenschaften mit ihren entsprechenden Werten im Nachrichtenfenster an:

```
-- Lingo syntax
trace(_movie.displayTemplate)

// JavaScript syntax
trace(_movie.displayTemplate);
```

Die folgenden Anweisungen legen verschiedne `displayTemplate`-Eigenschaften fest.

```
-- Lingo syntax
_movie.displayTemplate.dockingEnabled = TRUE
_movie.displayTemplate.resizable = FALSE
_movie.displayTemplate.appearanceOptions.mask = member("mask")
_movie.displayTemplate.titlebarOptions.sideTitlebar = TRUE

// JavaScript syntax
_movie.displayTemplate.dockingEnabled = true;
_movie.displayTemplate.resizable = false;
_movie.displayTemplate.appearanceOptions.mask = member("mask");
_movie.displayTemplate.titlebarOptions.sideTitlebar = true;
```

Siehe auch

appearanceOptions, dockingEnabled, Movie, resizable, systemTrayIcon, title (Fenster), titlebarOptions, type (Fenster), Window

distribution

Syntax

```
member(whichCastmember).modelResource(whichModelResource).emitter.distribution
```

Beschreibung

Diese 3D-Emittereigenschaft gibt an, wie die Partikel eines Partikelsystems bei ihrer Erstellung über den Emitterbereich hinweg verteilt werden. Mögliche Werte für diese Eigenschaft: `#gaussian` oder `#linear`. Der Standardwert ist `#linear`.

Beispiel

Im folgenden Beispiel ist `ThermoSystem` eine Modellressource vom Typ `#particle`. Die folgende Anweisung setzt die Eigenschaft „distribution“ des Emitters von `ThermoSystem` auf `#linear`, damit die Partikel von `ThermoSystem` bei ihrer Entstehung gleichmäßig über ihren Ursprungsbereich hinweg verteilt werden.

```
-- Lingo syntax
member("Fires").modelResource("ThermoSystem").emitter.distribution = #linear

// JavaScript syntax
member("Fires").getProp("modelResource", 1).getPropRef("emitter").distribution =
symbol("linear");
```

Siehe auch

emitter, region

dither

Syntax

```
-- Lingo syntax
memberObjRef.dither

// JavaScript syntax
memberObjRef.dither;
```

Beschreibung

Diese Bitmapdarstellereigenschaft rastert den Darsteller, der mit einer Farbtiefe von 8 Bit oder weniger (256 Farben) dargestellt wird, wenn ein im Darsteller fehlender Farbton angezeigt werden muss (`TRUE`), oder weist Director an, die nächstgelegene Farbe in der aktuellen Palette auszuwählen (`FALSE`).

Aus Leistungs- und Qualitätsgründen sollten Sie `dither` nur dann auf `TRUE` einstellen, wenn eine Anzeige in höherer Qualität erforderlich ist. Rastern dauert länger als Zuordnen einer ähnlichen Farbe, und Artefakte sind eher sichtbar, wenn über einer gerasterten Grafik animiert wird.

Bei einer Bildschirmfarbtiefe über 8 Bit hat diese Eigenschaft keine Wirkung.

Siehe auch

`depth (Bitmap)`

dockingEnabled

Syntax

```
-- Lingo syntax
_movie.displayTemplate.dockingEnabled
windowObjRef.dockingEnabled

// JavaScript syntax
_movie.displayTemplate.dockingEnabled;
windowObjRef.dockingEnabled;
```

Beschreibung

Diese Film- und Fenstereigenschaft gibt an, ob ein Film in einem Fenster (MIAW) andockbar ist, wenn er während der Erstellung geöffnet wird. Lesen/Schreiben.

Auf diese Eigenschaft kann nicht direkt von einem Movie-Objekt aus zugegriffen werden. Der Zugriff darauf erfolgt über die Eigenschaft `displayTemplate` des Movie-Objekts.

Der Standardwert dieser Eigenschaft ist `FALSE`, wodurch angegeben ist, dass ein MIAW beim Öffnen während der Erstellung nicht andockbar ist. Ist die Eigenschaft auf `TRUE` festgelegt, bestimmt der Wert der Eigenschaft `type` des Window-Objekts, die Darstellungsweise des Fensters während der Erstellung.

- Hat `dockingEnabled` den Wert `TRUE` und ist `type` auf `#document` festgelegt, sieht das MIAW aus wie ein Dokumentfenster in Director und verhält sich auch so. Das Fenster wird im Anzeigebereich angezeigt und kann an die Bühnen-, Drehbuch- und Besetzungsfenster sowie an Medien-Editoren und Nachrichtenfenster angedockt werden. Das Fenster kann aber mit keinem dieser Fenster gruppiert werden.

- Hat `dockingEnabled` den Wert `TRUE` und ist `type` auf `#tool` festgelegt, sieht das MIAW aus wie eins der Werkzeugfenster in Director und verhält sich auch so. Das Fenster kann mit allen Werkzeugfenstern gruppiert werden, mit Ausnahme des Eigenschafteninspektors und der Werkzeugpalette.
- Hat `dockingEnabled` den Wert `TRUE` und ist `type` auf `#dialog` festgelegt, wird der Typ ignoriert und das Fenster ist ein Erstellungsfenster.

Diese Eigenschaft wird in Projektoren ignoriert.

Beispiel

Die folgenden Anweisungen legen die Eigenschaft `dockingEnabled` auf TRUE fest,

```
-- Lingo syntax
_movie.displayTemplate.dockingEnabled = TRUE -- from the Movie object
window("Instructions").dockingEnabled = TRUE -- from the Window object

// JavaScript syntax
_movie.displayTemplate.dockingEnabled = true; // from the Movie object
window("Instructions").dockingEnabled = true; // from the Window object
```

Siehe auch

appearanceOptions, displayTemplate, Movie, titlebarOptions, type (Fenster), Window

domain

Syntax

```
-- Lingo syntax
dvdObjRef.domain

// JavaScript syntax
dvdObjRef.domain;
```

Beschreibung

Diese DVD-Eigenschaft gibt ein Symbol zurück, das die aktuelle Domäne angibt. Nur Lesen.

Beispiel

Die folgende Anweisung gibt die aktuelle Domäne zurück:

```
-- Lingo syntax
trace (member(1).domain)-- #title

// JavaScript syntax
trace (member(1).domain);// #title
```

Siehe auch

DVD

doubleClick

Syntax

```
-- Lingo syntax
_mouse.doubleClick

// JavaScript syntax
_mouse.doubleClick;
```

Beschreibung

Diese Mauseigenschaft testet, ob zwei Mausklicks als Doppelklick behandelt werden sollen, wenn sie in der für einen Doppelklick eingestellten Zeitspanne stattfanden (`TRUE`), oder als einzelne Klicks (`FALSE`). Nur Lesen.

Beispiel

Die folgende Anweisung bringt den Abspielkopf zum Bild "Angebot eingeben“, wenn der Benutzer mit der Maustaste doppelklickt:

```
-- Lingo syntax
if (_mouse.doubleClick) then
    _movie.go("Enter Bid")
end if

// JavaScript syntax
if (_mouse.doubleClick) {
    _movie.go("Enter Bid");
}
```

Siehe auch

`clickLoc`, `clickOn`, `Mouse`

drag

Syntax

```
member(whichCastmember).modelResource(whichModelResource).drag
```

Beschreibung

Diese 3D-Eigenschaft für Modellressourcen vom Typ `#particle` gibt an, welcher Prozentssatz der Geschwindigkeit eines Partikels bei jedem Simulationsschritt verloren geht. Der Gültigkeitsbereich dieser Eigenschaft liegt zwischen 0 (kein Geschwindigkeitsverlust) und 100 (totaler Geschwindigkeitsverlust, Partikel bewegt sich nicht mehr). Der Standardwert ist 0.

Beispiel

Im folgenden Beispiel ist `ThermoSystem` eine Modellressource vom Typ `#particle`. Die folgende Anweisung setzt die Eigenschaft drag von `ThermoSystem` auf 5. Dadurch wird ein starker Widerstand auf die Bewegung der Partikel von `ThermoSystem` ausgeübt und verhindert, dass sie sich sehr weit bewegen.

```
-- Lingo syntax
member("Fires").modelResource("ThermoSystem").drag = 5

// JavaScript syntax
member("Fires").getProp("modelResource", 1).drag = 5;
```

Siehe auch

wind, gravity

drawRect

Syntax

```
-- Lingo syntax
windowObjRef.drawRect

// JavaScript syntax
windowObjRef.drawRect;
```

Beschreibung

Diese Fenstereigenschaft identifiziert die rechteckigen Bühnenkoordinaten des Films, der in einem Fenster erscheint. Lesen/Schreiben.

Die Koordinaten werden als Rechteck angegeben, wobei die Eingaben in der Reihenfolge links, oben, rechts und unten erfolgen.

Diese Eigenschaft eignet sich zum Skalieren und Schwenken von Filmen, skaliert aber keine Text- und Felddarsteller neu. Das Skalieren von Bitmaps kann die Leistung beeinträchtigen.

Beispiel

Die folgende Anweisung zeigt die aktuellen Koordinaten des Filmfensters "Steuerpult“ an:

```
-- Lingo syntax
put(window("Control Panel").drawRect)

// JavaScript syntax
put(window("Control Panel").drawRect);
```

Die folgende Anweisung stellt das Rechteck des Films auf die Werte des Rechtecks `movieRectangle` ein. Der Filmausschnitt, der sich innerhalb des Rechtecks befindet, erscheint im Fenster.

```
-- Lingo syntax
movieRectangle = rect(10, 20, 200, 300)
window("Control Panel").drawRect = movieRectangle

// JavaScript syntax
var movieRectangle = rect(10, 20, 200, 300);
window("Control Panel").drawRect = movieRectangle;
```

Die folgenden Zeilen bewirken, dass die Bühne den Hauptbildschirmbereich ausfüllt:

```
-- Lingo syntax
_movie.stage.drawRect = _system.deskTopRectList[1]
_movie.stage.rect = _system.deskTopRectList[1]

// JavaScript syntax
_movie.stage.drawRect = _system.deskTopRectList[1];
_movie.stage.rect = _system.deskTopRectList[1];
```

Siehe auch

rect(), Window

dropShadow

Syntax

```
-- Lingo syntax
memberObjRef.dropShadow

// JavaScript syntax
memberObjRef.dropShadow;
```

Beschreibung

Diese Darstellereigenschaft bestimmt die Größe des Schlagschattens (in Pixel) für den Text in einem Felddarsteller.

Beispiel

Die folgende Anweisung stellt den Schlagschatten des Felddarstellers "Kommentar“ auf 5 Pixel ein:

```
--Lingo syntax
member("Comment").dropShadow = 5

// JavaScript syntax
member("Comment").dropShadow = 5;
```

duration (3D)

Syntax

```
member(whichCastmember).motion(whichMotion).duration
motionObjectReference.duration
```

Beschreibung

Mit dieser 3D-Eigenschaft können Sie ermitteln, wie lange (in Millisekunden) es dauert, bis die im Parameter *whichMotion* angegebene Bewegung fertig abgespielt ist. Diese Eigenschaft ist immer größer oder gleich 0.

Beispiel

Die folgende Anweisung zeigt die Dauer der Bewegung "Tritt“ in Millisekunden an:

```
-- Lingo syntax
put member("GbMember").motion("Kick").duration
-- 5100.0000

// JavaScript syntax
put( member("GbMember").getProp("motion", 1).duration );
// 5100.0000
```

Siehe auch

motion, currentTime (3D), play() (3D), queue() (3D)

duration (DVD)

Syntax

```
-- Lingo syntax
dvdObjRef.duration

// JavaScript syntax
dvdObjRef.duration;
```

Beschreibung

Diese DVD-Eigenschaft gibt die Titelgesamtzeit in Millisekunden zurück. Nur Lesen.

Beispiel

Die folgende Anweisung gibt die Dauer des aktuellen Titels zurück:

```
--Lingo syntax
trace (member(1).duration)-- 1329566

// JavaScript syntax
trace (member(1).duration);// 1329566
```

Siehe auch

DVD

duration (Darsteller)

Syntax

```
-- Lingo syntax
memberObjRef.duration

// JavaScript syntax
memberObjRef.duration;
```

Beschreibung
Diese Darstellereigenschaft bestimmt die Dauer des angegebenen Shockwave Audio (SWA)-, Übergangs-, Windows Media- oder QuickTime-Darstellers.

- Wenn *whichCastmember* eine streamende Sounddatei ist, gibt diese Eigenschaft die Dauer des Sounds an. Die Eigenschaft `duration` gibt 0 zurück, bis das Streaming beginnt. Wird`preLoadTime` auf 1 Sekunde eingestellt, kann die Bitrate die tatsächliche Dauer zurückgeben.
- Wenn *whichCastMember* ein Digitalvideodarsteller ist, gibt diese Eigenschaft die Dauer des Digitalvideos an. Der Wert wird in Ticks gemessen.
- Wenn *whichCastMember* ein Übergangsdarsteller ist, gibt diese Eigenschaft die Dauer des Übergangs an. Der Wert des Übergangs wird in Millisekunden gemessen. Beim Abspielen hat diese Einstellung dieselbe Wirkung wie die Einstellung „Dauer“ im Dialogfeld „Bildübergang“.

Diese Eigenschaft kann für alle Darsteller, die sie unterstützen, getestet werden. Eingestellt werden kann sie allerdings nur für Übergänge.

Ein Beispiel für `duration` in einem fertigen Film finden Sie im Film „QT and Flash“ im Ordner „Learning\\Lingo_Examples“ im Director-Anwendungsordner.

Beispiel
Wenn der SWA-Darsteller "Louie Prima“ vorausgeladen wurde, zeigt die folgende Anweisung die Sounddauer im Felddarsteller "Daueranzeige“ an:

```
-- Lingo syntax
on exitFrame
    if member("Louie Prima").state = 2 then
        member("Duration Displayer").text = string(member("Louie Prima").duration)
    end if
end

// JavaScript syntax
function exitFrame() {
    if (member("Louie Prima").state == 2) {
        member("Duration Displayer").text = member("Louie Prima").duration.toString()
    }
}
```

duration (MP4Media/FLV)

Syntax
```
put member(1).duration
```

Beschreibung
Diese MP4Media/FLV-Sprite- oder -Darstellereigenschaft, die der Länge eines MP4Media-Darstellers in Sekunden entspricht. Diese Eigenschaft ist schreibgeschützt. Sie kann ausgelesen, aber nicht eingestellt werden.

Beispiele
Das folgende Beispiel zeigt die Dauer des MP4Media/FLV-Darstellers in Sprite 2 an: 100.009 Sekunden.

```
-- Lingo syntax
put(sprite(2).duration) - 100.009
put(member("MP4Media/FLV ").duration) - 100.009

// JavaScript syntax
put(sprite(2).duration); // 100.009
put(member("MP4Media/FLV ").duration); // 100.009
```

duration (RealMedia, SWA)

Syntax

```
-- Lingo syntax
memberOrSpriteObjRef.duration

// JavaScript syntax
memberOrSpriteObjRef.duration;
```

Beschreibung

Diese Eigenschaft für `RealMedia` oder Shockwave Audio-Sprites und Darsteller gibt die Dauer eines `RealMedia` oder Shockwave Audio-Streams in Millisekunden zurück. Die Dauer des Streams ist unbekannt, bis die Wiedergabe des Darstellers beginnt. Wenn der Stream aus einem Live-Feed stammt oder noch nie abgespielt wurde, lautet der Wert dieser Eigenschaft 0. Diese Eigenschaft kann getestet, aber nicht eingestellt werden.

Beispiel

Die folgenden Beispiele zeigen, dass die Dauer des `RealMedia`-Streams in Sprite 2 und im Darsteller "Real“ 100.500 Millisekunden (100,500 Sekunden) beträgt.

```
-- Lingo syntax
put(sprite(2).duration) -- 100500
put(member("Real").duration) -- 100500

// JavaScript syntax
put(sprite(2).duration); // 100500
put(member("Real").duration); // 100500
```

Siehe auch

play() (RealMedia, SWA, Windows Media), seek(), currentTime (RealMedia)

editable

Syntax

```
-- Lingo syntax
spriteObjRef.editable

// JavaScript syntax
spriteObjRef.editable;
```

Beschreibung

Diese Sprite-Eigenschaft bestimmt, ob ein angegebenes Sprite auf der Bühne bearbeitet werden kann (`TRUE`) oder nicht (`FALSE`). Lesen/Schreiben.

Wenn die Darstellereigenschaft gesetzt ist, wird die Einstellung auf alle Sprites, die das Feld enthalten, angewendet.

Wenn diese Eigenschaft eingestellt ist, wirkt sie sich nur auf das angegebene Spriteaus.

Sie können auch ein Feld-Sprite durch Auswahl der Option „Bearbeitbar“ im Dialogfeld „Felddarsteller-Eigenschaften“ bearbeitbar machen.

Sie können ein Feld-Sprite anhand der Option „Bearbeitbar“ im Drehbuch bearbeitbar machen.

Damit der mit Skriptcode festgesetzte Wert über das aktuelle Sprite hinweg gültig ist, muss das Sprite ein mit Skript erstelltes Sprite sein.

Beispiel

Die folgende Prozedur macht zuerst aus dem Sprite-Kanal ein Puppet und dann das Sprite bearbeitbar:

```
-- Lingo syntax
on myNotes
    _movie.puppetSprite(5, TRUE)
    sprite(5).editable = TRUE
end

// JavaScript syntax
function myNotes() {
    _movie.puppetSprite(5, true);
    sprite(5).editable = true;
}
```

Die folgende Anweisung überprüft, ob ein Sprite bearbeitbar ist, und zeigt eine Nachricht an, wenn dies der Fall ist:

```
-- Lingo syntax
if (sprite(13).editable = TRUE) then
    member("Notice").text = "Please enter your answer below."
end if

// JavaScript syntax
if (sprite(13).editable == true) {
    member("Notice").text = "Please enter your answer below.";
}
```

Siehe auch

Sprite

editShortCutsEnabled

Syntax

```
-- Lingo syntax
_movie.editShortCutsEnabled

// JavaScript syntax
_movie.editShortCutsEnabled;
```

Beschreibung

Diese Filmeigenschaft gibt an, ob die Befehle „Ausschneiden“, „Kopieren“ und „Einfügen“ sowie die entsprechenden Tastaturbefehle im aktuellen Film funktionieren. Lesen/Schreiben.

Wenn diese Eigenschaft auf `TRUE` gesetzt ist, funktionieren diese Textbearbeitungsbefehle. Wenn sie auf `FALSE` gesetzt ist, sind diese Befehle nicht verfügbar. Die Standardeinstellung ist `TRUE` für Filme, die in Director 8 und später erstellt wurden, und `FALSE` für Filme, die aus früheren Director-Versionen stammen.

Beispiel

Die folgende Anweisung deaktiviert die Textbearbeitungsfunktionen „Ausschneiden“, „Kopieren“ und „Einfügen“:

```
-- Lingo syntax
_movie.editShortCutsEnabled = 0

// JavaScript syntax
_movie.editShortCutsEnabled = 0;
```

Siehe auch

Movie

elapsedTime

Syntax

```
-- Lingo syntax
soundChannelObjRef.elapsedTime

// JavaScript syntax
soundChannelObjRef.elapsedTime;
```

Beschreibung

Diese Soundkanaleigenschaft gibt an, wie viele Millisekunden des aktuellen Sounddarstellers in einem angegebenen Soundkanal bereits abgespielt wurden. Nur Lesen.

Die verstrichene Zeit ist zu Beginn der Soundwiedergabe 0 und nimmt unabhängig von Soundschleifen, der Einstellung von `currentTime` oder anderen Manipulationen während der Wiedergabe zu. Mit `currentTime` können Sie die jeweils aktuelle absolute Zeit im Sound testen.

Da der Wert dieser Eigenschaft eine Fließkommazahl ist, lässt sich die Dauer der Soundwiedergabe auf den Bruchteil einer Millisekunde genau bestimmen.

Beispiel

Diese idle-Prozedur zeigt während der Wartezeiten die abgelaufene Zeit für Soundkanal 4 in einem Feld auf der Bühne an:

```
-- Lingo syntax
on idle
    member("time").text = string(sound(4).elapsedTime)
end idle

// JavaScript syntax
function idle() {
    member("time").text = sound(4).elapsedTime.toString();
}
```

Siehe auch

currentTime (Sprite), Sound Channel

elapsedTime (Mixer)

Syntax

```
mixerRef.elapsedTime (Read-only)
```

Beschreibung

Diese Mixereigenschaft gibt die abgelaufene Zeit (in Millisekunden) für die Wiedergabe des aktuellen Mixers aus. `elapsedTime` ist eine schreibgeschützte Eigenschaft.

Beispiele

```
--Lingo syntax
on mouseUp me
put mixerRef.elapsedTime -- Gives the time, in milliseconds, that the
-- current mixer object has played for.
end

// JavaScript syntax
function mouseUp(){
put (mixerRef.elapsedTime); // Gives the time, in milliseconds, that the current
// mixer object has played for.
}
```

Siehe auch

Mixer

elapsedTime (Soundobjekt)

Syntax

```
soundObject.elapsedTime (Read-only)
```

Beschreibung

Diese Soundobjekteigenschaft gibt die Zeit (in Millisekunden) zurück, die während der Wiedergabe des Soundobjekts verstrichen ist. `elapsedTime` ist eine schreibgeschützte Eigenschaft.

Beispiele

```
--Lingo syntax
on mouseUp me
    put soundObjRef.elapsedTime -- Gives the time, in milliseconds, that the
-- current sound object has played for.
end

// JavaScript syntax
function mouseUp(){
put (soundObjRef.elapsedTime) ; // Gives the time, in milliseconds, that the current
// sound object has played for.
}
```

emissive

Syntax

```
member(whichCastmember).shader(whichShader).emissive
member(whichCastmember).model(whichModel).shader.emissive
member(whichCastmember).model(whichModel).shaderList{[index]}.emissive
```

Beschreibung

Diese 3D-Eigenschaft für den Shader `#standard` fügt unabhängig von der Szenenbeleuchtung Licht zum Shader hinzu. Ein Modell, das einen Shader verwendet, bei dem die Eigenschaft `emissive` auf `rgb(255, 255, 255)` gesetzt ist, scheint beispielsweise durch ein weißes Licht beleuchtet zu sein, auch wenn es in der Szene keine Lichtquellen gibt. Das Modell beleuchtet jedoch weder andere Modelle noch wirft es Licht auf die Szene.

Der Standardwert dieser Eigenschaft lautet `rgb(0, 0, 0)`.

Beispiel

Die folgende Anweisung setzt die Eigenschaft emissive des Shaders "Globus" auf rgb(255, 0, 0). Modelle, die diesen Shader verwenden, scheinen durch ein rotes Licht beleuchtet zu sein.

```
-- Lingo syntax
member("MysteryWorld").shader("Globe").emissive = rgb(255, 0, 0)

// JavaScript syntax
member("MysteryWorld").getProp("shader", 1).emissive = color(255, 0, 0);
```

Siehe auch

silhouettes

emitter

Syntax

```
--Lingo Usage
member(whichCastmember).modelResource(whichModelResource).emitter.numParticles
member(whichCastmember).modelResource(whichModelResource).emitter.mode
member(whichCastmember).modelResource(whichModelResource).emitter.loop
member(whichCastmember).modelResource(whichModelResource).emitter.direction
member(whichCastmember).modelResource(whichModelResource).emitter.region
member(whichCastmember).modelResource(whichModelResource).emitter.distribution
member(whichCastmember).modelResource(whichModelResource).emitter.angle
member(whichCastmember).modelResource(whichModelResource).emitter.path
member(whichCastmember).modelResource(whichModelResource).emitter.pathStrength
member(whichCastmember).modelResource(whichModelResource).emitter.minSpeed
member(whichCastmember).modelResource(whichModelResource).emitter.maxSpeed

// JavaScript usage
member(whichCastmember).getPropRef("modelresource",whichModelResourceIndex).getPropRef("emit
ter").numParticles
member(whichCastmember).getPropRef("modelresource",whichModelResourceIndex).getPropRef("emit
ter").mode
member(whichCastmember).getPropRef("modelresource",whichModelResourceIndex).getPropRef("emit
ter").loop
member(whichCastmember).getPropRef("modelresource",whichModelResourceIndex).getPropRef("emit
ter").direction
member(whichCastmember).getPropRef("modelresource",whichModelResourceIndex).getPropRef("emit
ter").region
member(whichCastmember).getPropRef("modelresource",whichModelResourceIndex).getPropRef("emit
ter").distribution
member(whichCastmember).getPropRef("modelresource",whichModelResourceIndex).getPropRef("emit
ter").angle
member(whichCastmember).getPropRef("modelresource",whichModelResourceIndex).getPropRef("emit
ter").path
member(whichCastmember).getPropRef("modelresource",whichModelResourceIndex).getPropRef("emit
ter").pathStrength
member(whichCastmember).getPropRef("modelresource",whichModelResourceIndex).getPropRef("emit
ter").minSpeed
member(whichCastmember).getPropRef("modelresource",whichModelResourceIndex).getPropRef("emit
ter").maxSpeed
```

Beschreibung

Dieses 3D-Element für Partikelsysteme steuert den anfänglichen Antrieb von Partikeln aus einer Modellressource vom Typ `#particle`.

Eine vollständige Auflistung aller Emittereigenschaften finden Sie unter "Siehe auch“. Weitere Informationen finden Sie in den Einträgen zu den einzelnen Eigenschaften.

Siehe auch

numParticles, loop (emitter), direction, distribution, region, angle (3D), path (3D), pathStrength, minSpeed, maxSpeed

emulateMultibuttonMouse

Syntax

```
-- Lingo syntax
_player.emulateMultibuttonMouse

// JavaScript syntax
_player.emulateMultibuttonMouse;
```

Beschreibung

Diese Player-Eigenschaft bestimmt, ob ein Film einen Mausklick bei gedrückter Ctrl-Taste auf dem Mac wie einen Klick mit der rechten Maustaste in Windows interpretiert (`TRUE`) oder nicht (`FALSE`, Standard). Lesen/Schreiben.

Für Rechtsklicks gibt es auf dem Mac kein direktes Äquivalent.

Wird diese Eigenschaft auf `TRUE` gesetzt, ist die Reaktion auf Maustastenaktionen immer gleich, unabhängig von der Plattform, auf der die Filme abgespielt werden.

Beispiel

Die folgende Anweisung legt die Eigenschaft `emulateMultibuttonMouse` auf TRUE fest:

```
-- Lingo syntax
_player.emulateMultibuttonMouse = TRUE

// JavaScript syntax
_player.emulateMultibuttonMouse = true;
```

Siehe auch

`Player`

enabled

Syntax

```
the enabled of menuItem whichItem of menu whichMenu
```

Beschreibung

Diese Menüelement-Eigenschaft bestimmt, ob das in *whichItem* angegebene Menüelement als Standardtext angezeigt wird und ausgewählt werden kann (`TRUE`, Standard) oder grau erscheint und nicht ausgewählt werden kann (`FALSE`).

Beim Ausdruck *whichItem* kann es sich entweder um den Name oder um die Nummer eines Menüelements handeln. Der Ausdruck *whichMenu* kann entweder ein Menüname oder eine Menünummer sein.

Die Eigenschaft „enabled“ kann getestet und eingestellt werden.

Hinweis: *Im Shockwave Player sind keine Menüs verfügbar.*

Beispiel

Die folgende Prozedur aktiviert oder deaktiviert alle Elemente des angegebenen Menüs. Das Argument `theMenu` gibt das Menü an, und das Argument vSetting gibt TRUE oder FALSE an. Zum Beispiel deaktiviert die Aufrufsanweisung `ableMenu` ("Spezial", FALSE) alle Elemente im Menü "Spezial".

```
-- Lingo syntax
on ableMenu theMenu, vSetting
    set n = the number of menuItems of menu theMenu
    repeat with i = 1 to n
        set the enabled of menuItem i of menu theMenu to vSetting
    end repeat
end ableMenu
// JavaScript syntax
function ableMenu (theMenu, vSetting){
    n = _menuBar.menu[theMenu].item.count;
    for( i = 1 ; i <= n ; i++)
        _menuBar.menu[theMenu].item[i].enabled = vSetting;
}
```

Siehe auch

name (Menüeigenschaft), number (Menüs), checkMark, script, number (Menüelemente)

enabled (collision)

Syntax

```
member(whichCastmember).model(whichModel).collision.enabled
```

Beschreibung

Mit dieser 3D-Kollisionseigenschaft können Sie ermitteln und festlegen, ob Modellkollisionen entdeckt werden (TRUE) oder nicht (FALSE). Wenn Sie diese Eigenschaft auf FALSE setzen, wird der Modifizierer collision vorübergehend deaktiviert, aber nicht aus dem Modell entfernt.

Die Standardeinstellung dieser Eigenschaft lautet TRUE.

Beispiel

Die folgende Anweisung aktiviert den Modifizierer collision für das Modell "Box".

```
-- Lingo syntax
member("3d world").model("box").collision.enabled = TRUE

// JavaScript syntax
member("3d world").getProp("model",1).getPropRef("collision").enabled = true;
```

Siehe auch

addModifier, collision (modifier), modifier

enabled (Filter)

Syntax

```
audioFilter(#NameOfFilter, [#enabled:isEnabled])
```

Beschreibung

Die Eigenschaft „enabled“ steht für alle Filter zur Verfügung. Sie kann zum Aktivieren oder Deaktivieren von Filtern verwendet werden.

Eigenschaft	Beschreibung	Bereich	Standard
enabled:bool	Aktiviert (enabled:true) oder deaktiviert (enabled:false) den Filter	True, false	True

Beispiel

```
--Lingo Syntax
audioFilter(#EchoFilter, [#enabled:true])
//Javascript Syntax:
audioFilter(symbol("EchoFilter"),propList(symbol("enabled"),true));
```

enabled (fog)

Syntax

```
member(whichCastmember).camera(whichCamera).fog.enabled
sprite(whichSprite).camera{(index)}.fog.enabled
```

Beschreibung

Diese 3D-Kameraeigenschaft gibt an, ob die Kamera Nebel zur Kameraansicht hinzufügt. Die Standardeinstellung dieser Eigenschaft lautet `FALSE`.

Beispiel

Die folgende Anweisung erzeugt Nebel in der von der Kamera `BayView` erfassten Ansicht:

```
member("MyYard").camera("BayView").fog.enabled = TRUE
```

Siehe auch

`fog`

enabled (sds)

Syntax

```
member(whichCastmember).model(whichModel).sds.enabled
```

Beschreibung

Diese 3D-Eigenschaft für den Modifizierer `sds` gibt an, ob der an einem Modell angebrachte Modifizierer `sds` vom Modell auch tatsächlich verwendet wird.

Die Standardeinstellung dieser Eigenschaft lautet `TRUE`.

Der Versuch, den Modifizierer `sds` zu einem Modell hinzuzufügen, an dem bereits der Modifizierer `inker` oder `toon` angebracht ist, scheitert ohne Fehlermeldung. Analog dazu scheitert auch der Versuch, den Modifizierer `inker` oder `toon` zu einem Modell hinzuzufügen, an dem bereits der Modifizierer `sds` angebracht ist, ohne Fehlermeldung. Vorsicht bei Verwendung des Modifizierers `sds` in Verbindung mit dem Modifizierer `lod`! Weitere Informationen enthält der Eintrag zu `sds (modifier)`.

Beispiel

Die folgende Anweisung aktiviert den Modifizierer sds, der am Modell „Baby“ angebracht ist:

```
member("Scene").model("Baby").sds.enabled = TRUE
```

Siehe auch

`sds (Modifizierer)`, `modifier`, `addModifier`

enableFlashLingo

Syntax

```
-- Lingo syntax
_movie.enableFlashLingo

// JavaScript syntax
_movie.enableFlashLingo;
```

Beschreibung

Diese Filmeigenschaft bestimm, ob ein Sprite mit Flash-Inhalt direkte Skriptrückrufe ausführen kann, wenn die Flash-Methode `getURL()` verwendet wird. Lesen/Schreiben.

Die Flash-Methode `getURL()` lädt eine neue URL in ein leeres Browserfenster.

Wenn `enableFlashLingo` auf `TRUE` festgelegt ist, kann ein Sprite mit Flash-Inhalt alle (im Rahmen der Standardregeln für Shockwave Player-Sicherheit) gültigen Skriptbefehle ausführen, wenn `getURL()` aufgerufen wird.

Wenn `enableFlashLingo` auf `FALSE` festgelegt ist, kann ein Sprite mit Flash-Inhalt keine Skriptbefehle ausführen, wenn `getURL()` aufgerufen wird. Der Standardwert dieser Eigenschaft lautet `FALSE`.

Diese Eigenschaft eignet sich besonders für das Erstellen eines Films, der Flash-Inhalte unbekannter Herkunft anzeigt, wie z. B. in einem Projektor, der einen Systemordner nach SWF-Dateien durchsucht, oder ein Film mit Shockwave-Inhalt, der von einem Endbenutzer eine URL zu einer SWF-Datei akzeptiert.

Beispiel

Die folgende Anweisung setzt die Eigenschaft `enableFlashLingo` auf TRUE:

```
-- Lingo syntax
_movie.enableFlashLingo = TRUE

// JavaScript syntax
_movie.enableFlashLingo = true;
```

Siehe auch

`Movie`

endAngle

Syntax

```
member(whichCastmember).modelResource(whichModelResource).endAngle
```

Beschreibung

Diese 3D-Eigenschaft für Modellressourcen vom Typ `#cylinder` oder `#sphere` gibt an, welcher Anteil der Kugel bzw. des Zylinders gezeichnet wird.

Die Oberfläche einer Kugel wird durch Zeichnen eines 2D-Halbkreisbogens um die Y-Achse der Kugel zwischen `startAngle` und `endAngle` erzeugt. Wenn `startAngle` auf 0 und `endAngle` auf 360 gesetzt ist, ergibt sich eine komplette Kugel. Um einen Kugelausschnitt zu zeichnen, setzen Sie `endAngle` auf einen Wert unter 360.

Die Oberfläche eines Zylinders wird durch Zeichnen einer 2D-Linie um die Y-Achse des Zylinders zwischen `startAngle` und `endAngle` erzeugt. Wenn `startAngle` auf 0 und `endAngle` auf 360 gesetzt ist, ergibt sich ein kompletter Zylinder. Um einen Zylinderausschnitt zu zeichnen, setzen Sie `endAngle` auf einen Wert unter 360.

Der Standardwert dieser Eigenschaft ist 360.

Beispiel

In diesem Beispiel sei angenommen, dass der Darsteller `MyMember` ein Modell enthält, das die Modellressource "Kugel4" verwendet, bei der die Eigenschaft `endAngle` auf 310 gesetzt ist, sodass sich eine Öffnung von 50° ergibt. Die Prozedur `closeSphere` schließt diese Öffnung mit einem Schiebeeffekt. Die Wiederholungsschleife ändert den `endAngle`-Wert der Kugel um jeweils 1°. Der in der Wiederholungsschleife verwendete Befehl `updateStage` bewirkt, dass die Bühne nach jedem 1°-Schritt neu gezeichnet wird.

```
-- Lingo syntax
on closeSphere
    MyAngle = member("MyMember").modelresource("Sphere4").endAngle
    repeat with r = 1 to 50
        MyAngle = MyAngle + 1
        member("MyMember").modelresource("Sphere4").endAngle = MyAngle
        updateStage
    end repeat
end

// JavaScript syntax
function closeSphere(){
    var MyAngle = member("MyMember").getProp("modelresource", 1).endAngle;
    for( r = 1; r <= 50; r++)
    {
        MyAngle++;
        member("MyMember").getProp("modelresource", 1).endAngle = MyAngle;
        _movie.updateStage();
    }
}
```

Siehe auch

state (3D)

endColor

Syntax

```
-- Lingo syntax
memberObjRef.endColor

// JavaScript syntax
memberObjRef.endColor;
```

Beschreibung

Diese Vektorform-Darstellereigenschaft bezeichnet die Schlußfarbe einer Verlaufsformfüllung, die als RGB-Wert angegeben wird.

`endColor` ist nur gültig, wenn `fillMode` auf `#gradient` und die Anfangsfarbe mit `fillColor` eingestellt wird.

Diese Eigenschaft kann getestet und eingestellt werden.

Ein Beispiel für `endColor` in einem fertigen Film finden Sie im Film "Vector Shapes“ im Ordner „Learning/Lingo Examples“ im Director-Anwendungsordner.

Siehe auch

`color()`, `fillColor`, `fillMode`

endFrame

Syntax

```
-- Lingo syntax
spriteObjRef.endFrame

// JavaScript syntax
spriteObjRef.endFrame;
```

Beschreibung

Diese Sprite-Eigenschaft gibt die Bildnummer des letzten Bildes im Sprite-Einschluss zurück. Nur Lesen.

Diese Eigenschaft ist nützlich, um den Einschluss im Drehbuch eines bestimmten Sprites zu bestimmen.

Sie ist nur in einem Bild verfügbar, welches das Sprite enthält. Sie kann nicht aufSprites in verschiedenen Bildern des Films angewendet werden.

Beispiel

Die Ausgabe der folgenden Anweisung gibt das Endbild des Sprites in Kanal 5 im Nachrichtenfenster an:

```
-- Lingo syntax
put(sprite(5).endFrame)

// JavaScript syntax
put(sprite(5).endFrame);
```

Siehe auch

`Sprite`, `startFrame`

endian

Syntax

```
byteArrayObject.endian
```

Beschreibung

Diese Bytearrayeigenschaft wird beim Lesen oder Schreiben von Zahlenwerten in das Bytearray verwendet.

Beispiel

```
--Lingo syntax
bArray = byteArray("Sample byte array")
put bArray.endian

//JavaScript syntax
bArray = byteArray("Sample byte array");
put(bArray.endian);
```

endTime (Soundkanal)

Syntax

```
-- Lingo syntax
soundChannelObjRef.endTime

// JavaScript syntax
soundChannelObjRef.endTime;
```

Beschreibung

Diese Soundkanaleigenschaft gibt die Endzeit des gegenwärtig abgespielten, unterbrochenen oder wartenden Sounds an. Lesen/Schreiben.

Bei der Endzeit handelt es sich um die interne Zeit des Sounddarstellers, zu der dieser die Wiedergabe beendet. Da es sich bei diesem Wert um eine Fließkommazahl handelt, lässt sich die Dauer der Soundwiedergabe auf den Bruchteil einer Millisekunde genau messen und steuern. Der Standardwert ist das normale Ende des Sounds.

Diese Eigenschaft kann nur dann auf einen anderen Wert als das normale Soundende eingestellt werden, wenn sie als Parameter mit einer der Methoden `queue()` oder `setPlayList()` übergeben wird.

Beispiel

Die folgende Anweisung prüft, ob der Sounddarsteller „Jingle“ so eingestellt ist, dass er in Soundkanal 1 bis zum Ende abgespielt wird:

```
-- Lingo syntax
if (sound(1).startTime > 0 and sound(1).endTime < member("Jingle").duration) then
_player.alert("Not playing the whole sound.")
end if

// JavaScript syntax
if (sound(1).startTime > 0 && sound(1).endTime < member("Jingle").duration) {
    _player.alert("Not playing the whole sound.");
}
```

Siehe auch

queue(), setPlayList(), Sound Channel

endTime (Soundobjekt)

Syntax

```
soundObject.endTime
```

Beschreibung

Diese Soundobjekteigenschaft gibt die Endzeit (in Millisekunden) des aktuellen Soundobjekts zurück.

Beispiele

```
soundObject.endTime = 46890
--Lingo syntax
on mouseUp me
    put soundObjRef.endTime -- Displays the end time for the sound object
-- associated with soundobjectRef.
end

// JavaScript syntax
function mouseUp(){
put (soundObjRef.endTime) ; // Displays the end time for the sound object
// associated with soundobjectRef.
}
```

Siehe auch

startTime (Soundobjekt)

environmentPropList

Syntax

```
-- Lingo syntax
_system.environmentPropList

// JavaScript syntax
_system.environmentPropList;
```

Beschreibung

Diese Systemeigenschaft enthält eine Liste mit Informationen über die Umgebung, in der der Director-Inhalt gegenwärtig abgespielt wird. Nur Lesen.

Dieses Design ermöglicht es Adobe, der Eigenschaft `environmentPropList` in der Zukunft Informationen hinzuzufügen, ohne dass sich dies auf bestehende Filme auswirkt.

Die Informationen enthalten Eigenschafts-/Wertepaare für diesen Bereich.

`#platform`	String mit „Mac,`PowerPC`“ oder „Windows,32". Dies hängt davon ab, unter welchem Betriebssystem und auf welcher Hardware der Film abgespielt wird.
`#runMode`	String mit „Author“, „Projector“ oder „Plugin“, je nachdem, mit welcher Anwendung der Film abgespielt wird.
`#colorDepth`	Ganzzahl zur Angabe der Farbtiefe des Bildschirms, auf dem die Bühne angezeigt wird. Mögliche Werte sind 1, 2, 4, 8, 16 oder 32.
`#internetConnected`	Symbol, das angibt, ob der Computer, auf dem der Film abgespielt wird, über eine aktive Internetverbindung verfügt. Mögliche Werte: `#online` und `#offline`.
`#uiLanguage`	Zeichenfolge zur Angabe der Sprache, in der die Benutzeroberfläche des Players angezeigt wird.
`#osLanguage`	String zur Angabe der Sprache, die vom Betriebssystem des Computers verwendet wird.

`#osVersion`	Der Wert von `#osVersion` ist eine Zeichenfolge (String).
	Unter Windows wird die Eigenschaft `#osVersion` mit Informationen ausgefüllt, die mithilfe des Systemaufrufs `GetVersionEx()` abgerufen werden. Die Werte in der Zeichenfolge stammen aus der OSVERSIONINFO-Struktur:
	"Windows CE" oder "Windows NT" oder "Windows 2000" oder Windows XP" oder "Windows 95" oder
	"Windows 98" oder "Windows ME"
	`dwMajorVersion`
	`dwMinorVersion`
	`dwOSVersionInfoSize`
	`dwPlatformId`
	`szCSDVersion`
	Auf dem Mac stammen die Werte in der Zeichenfolge aus dem Aufruf `Gestalt(gestaltSystemVersion)`:
	"Mac OS"
	major version
	minor version
	sub-version

`#productBuildVersion`	String mit der internen Build-Nummer der Wiedergabeanwendung.

Die Eigenschaften enthalten dieselben Informationen wie die gleichnamigen Eigenschaften und Funktionen.

Beispiel

Die folgende Anweisung zeigt die Umgebungsliste im Nachrichtenfenster an:

```
-- Lingo syntax
put(_system.environmentPropList)

// JavaScript syntax
put(_system.environmentPropList);
```

Siehe auch

`System`

error

Syntax

```
member(whichCastmember).model(whichModel).sds.error
```

Beschreibung

Diese 3D-Eigenschaft für den Modifizierer `#sds` gibt die Fehlertoleranz des Modifizierers beim Synthetisieren geometrischer Details in Modellen.

Diese Eigenschaft funktioniert nur, wenn die Eigenschaft `subdivision` des Modifizierers auf `#adaptive` gesetzt ist. Die Modifiziereigenschaften `tension` und `depth (3D)` steuern im Zusammenspiel mit der Eigenschaft `error` die vom Modifizierer durchgeführte Unterteilungsaktivität.

Beispiel

Die folgende Anweisung setzt die Eigenschaft error des Modifizierers `#sds` für das Modell „Baby“ auf 0. Wenn die Modifiziereinstellung tension niedrig und die Einstellung depth hoch ist und die Einstellung subdivision auf `#adaptive` gesetzt ist, ergibt sich eine starke Auswirkung auf die Geometrie von „Baby“.

```
-- Lingo syntax
member("Scene").model("Baby").addModifier(#sds)
member("Scene").model("Baby").sds.subdivision = #adaptive
member("Scene").model("Baby").sds.error = 0

// JavaScript syntax
member("Scene").getProp("model",2).addModifier(symbol("sds"));
member("Scene").getProp("model",2).getPropRef("sds").subdivision = symbol("adaptive");
member("Scene").getProp("model",2).getPropRef("sds").error = 0;
```

Siehe auch

`sds (Modifizierer)`, `subdivision`, `depth (3D)`, `tension`

eventPassMode

Syntax

```
-- Lingo syntax
memberOrSpriteObjRef.eventPassMode

// JavaScript syntax
memberOrSpriteObjRef.eventPassMode;
```

Beschreibung

Diese Flash-Darsteller- und Sprite-Eigenschaft steuert, wann ein Flash-Film Mausereignisse an Verhalten übergibt, die an den dem Flash-Sprite zu Grunde liegenden Sprites angebracht sind. Die Eigenschaft `eventPassMode` kann die folgenden Werte haben:

- `#passAlways` (Standard) – Übergibt immer Mausereignisse.
- `#passButton` – Übergibt Mausereignisse nur dann, wenn im Flash-Film auf eine Schaltfläche geklickt wird.
- `#passNotButton` – Übergibt Mausereignisse nur dann, wenn auf ein Objekt geklickt wird, das keine Schaltfläche ist.

- `#passNever` – Übergibt niemals Mausereignisse.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Das folgende Bildskript überprüft, ob die Schaltflächen in einem Flash-Film gegenwärtig aktiviert sind, und setzt in diesem Fall `eventPassMode` auf `#passNotButton`. Wenn die Schaltflächen deaktiviert sind, setzt das Skript `eventPassMode` auf `#passAlways`. Dies hat den folgenden Effekt:

- Mausereignisse an Objekten, die keine Schaltflächen sind, werden stets an Sprite-Skripts übergeben.
- Mausereignisse an Schaltflächenobjekten werden an Sprite-Skripts übergeben, wenn die Schaltflächen deaktiviert sind. Sind die Schaltflächen aktiviert, werden die entsprechenden Mausereignisse unterbrochen.

```
-- Lingo syntax
on enterFrame
    if sprite(5).buttonsEnabled = TRUE then
        sprite(5).eventPassMode= #passNotButton
    else
        sprite(5).eventPassMode = #passAlways
    end if
end

// JavaScript syntax
function enterFrame() {
    var btEn = sprite(5).buttonsEnabled;
    if (btEn == 1) {
        sprite(5).eventPassMode= symbol("passNotButton");
    } else {
        sprite(5).eventPassMode = symbol("passAlways");
    }
}
```

exitLock

Syntax

```
-- Lingo syntax
_movie.exitLock

// JavaScript syntax
_movie.exitLock;
```

Beschreibung

Diese Filmeigenschaft bestimmt, ob der Benutzer einen Projektor beenden und somit zum Windows Desktop oder Mac-Finder zurückkehren kann (`FALSE`, Standard) oder nicht (`TRUE`). Lesen/Schreiben.

Der Benutzer kann das Programm durch Drücken der folgenden Tastenkombinationen beenden: Strg+Punkt (Windows) oder Befehl+Punkt (Mac), Strg+Q (Windows) oder Befehl+Q (Mac) sowie Strg+W (Windows) oder Befehl+W (Mac). In Windows wird außerdem die Esc-Taste unterstützt.

Beispiel

Die folgende Anweisung setzt die Eigenschaft `exitLock` auf TRUE:

```
-- Lingo syntax
_movie.exitLock = TRUE

// JavaScript syntax
_movie.exitLock = true;
```

Wenn `exitLock` auf TRUE gesetzt ist, wird beim Drücken der Tastenkombination Strg+Punkt, Strg+Q, Strg+W oder der Esc-Taste (Windows) bzw. der Tastenkombination Befehl+Punkt, Befehl+Q oder Befehl+W (Macintosh) kein Vorgang automatisch ausgeführt. Diese Prozedur überprüft Tastatureingaben danach, ob Tasten zum Beenden gedrückt wurden, und führt den Benutzer zu einer benutzerdefinierten Beenden-Sequenz:

```
-- Lingo syntax
on checkExit
    if ((_key.commandDown) and (_key.key = "." or _key.key = "q") and (_movie.exitLock = TRUE))
then _movie.go("quit sequence")
end checkExit

// JavaScript syntax
function checkExit() {
    if ((_key.commandDown) && (_key.key == "." || _key.key == "q") && (_movie.exitLock == true)) {
        _movie.go("quit sequence");
    }
}
```

Siehe auch

Movie

externalParamCount

Syntax

```
-- Lingo syntax
_player.externalParamCount

// JavaScript syntax
_player.externalParamCount;
```

Beschreibung

Diese Player-Eigenschaft gibt die Anzahl der Parameter an, die von einem HTML-Tag „<EMBED>“ oder „<OBJECT>“ an einen Film mit Shockwave-Inhalt übergeben werden. Nur Lesen.

Diese Eigenschaft gilt nur für Filme mit Shockwave-Inhalt, die in einem Browser abgespielt werden. Für Filme in der Authoring-Umgebung oder Projektoren ist sie nicht verfügbar.

Weitere Informationen zu den gültigen externen Parametern finden Sie unter `externalParamName()` und `externalParamValue()`.

Beispiel

Die folgende Prozedur stellt fest, ob ein „<OBJECT>“- oder „<EMBED>“-Tag externe Parameter an einen Film mit Shockwave-Inhalt übergibt, und führt in diesem Fall Lingo-Anweisungen aus:

```
-- Lingo syntax
if (_player.externalParamCount > 0) then
    -- perform some action
end if

// JavaScript syntax
if (_player.externalParamCount > 0) {
    // perform some action;
}
```

Siehe auch

externalParamName(), externalParamValue(), Player

face

Syntax

```
\-- Lingo Usage
member(whichCastmember).modelResource(whichModelResource).face.count
member(whichCastmember).modelResource(whichModelResource).face[index].colors
member(whichCastmember).modelResource(whichModelResource).face[index].normals
member(whichCastmember).modelResource(whichModelResource).face[index].shader
member(whichCastmember).modelResource(whichModelResource).face[index].textureCoordinates
member(whichCastmember).modelResource(whichModelResource).face[index].vertices
member(whichCastmember).model(whichModel).meshdeform.face.count
member(whichCastmember).model(whichModel).meshdeform.mesh[index].face.count
member(whichCastmember).model(whichModel).meshdeform.mesh[meshIndex].face[faceIndex]
member(whichCastmember).model(whichModel).meshdeform.mesh[meshIndex].face[faceIndex].neighbo
r{[neighborIndex]}

// JavaScript Usage
member(whichCastmember).getProp("modelResource", whichModelResourceIndex).getPropRef("face",
index).colors
member(whichCastmember).getProp("modelResource", whichModelResourceIndex).getPropRef("face",
index).normals
member(whichCastmember).getProp("modelResource", whichModelResourceIndex).getPropRef("face",
index).shader
member(whichCastmember).getProp("modelResource", whichModelResourceIndex).getPropRef("face",
index).textureCoordinates
member(whichCastmember).getProp("modelResource", whichModelResourceIndex).getPropRef("face",
index).vertices
member(whichCastmember).getProp("model",
whichModelIndex).getPropRef("meshdeform").getPropRef("mesh", meshIndex).face[faceIndex]
member(whichCastmember).getProp("model",
whichModelIndex).getPropRef("meshdeform").getPropRef("mesh",
meshIndex).face[faceIndex].neighbor{[neighborIndex]}
```

Beschreibung

3D-Eigenschaft für Modellressourcen vom Typ `#mesh` und für den Modifizierer `meshdeform`. Alle Modellressourcen sind aus Polygonen bestehende Gitternetze. Jedes Polygon ist eine Fläche.

Sie können auf die Eigenschaften der Flächen von Modellressourcen vom Typ `#mesh` zugreifen. Änderungen dieser Eigenschaften treten erst nach Aufruf des Befehls `build()` in Kraft.

Hinweis: *Weitere Informationen zu den folgenden Eigenschaften finden Sie in den einzelnen Beschreibungen.*

- `count` gibt die Anzahl der Polygone im Gitternetz an.
- `colors` gibt an, welche Indizes in der Farbliste der Modellressource für die einzelnen Scheitelpunkte der Fläche verwendet werden sollen.
- `normals` gibt an, welche Indizes in der Normalenliste der Modellressource für die einzelnen Scheitelpunkte der Fläche verwendet werden sollen.
- `shadowPercentage` gibt an, welcher Shader beim Rendern der Fläche verwendet wird.
- `textureCoordinates` gibt an, welche Indizes in der Texturkoordinatenliste der Modellressource für die einzelnen Scheitelpunkte der Fläche verwendet werden sollen.
- `vertices` gibt an, welche Indizes in der Scheitelpunktliste der Modellressource zum Definieren der Fläche verwendet werden sollen.

Eine Beschreibung der Flächeneigenschaften enthält der Eintrag zu `meshDeform`.

Siehe auch

`build()`, `newMesh`, `meshDeform (Modifizierer)`

face[ ]

Syntax

```
member(whichCastmember).model(whichModel).meshdeform.mesh[meshIndex].face[faceIndex]
```

Beschreibung

Diese 3D-Eigenschaft für den Modifizierer `meshdeform` gibt an, welche Indizes in der Scheitelpunktliste der Modellressource zum Definieren der Fläche verwendet wurden.

Diese Eigenschaft kann getestet, aber nicht eingestellt werden. Bei einer Modellressource vom Typ `#mesh` können Sie die Scheitelpunkte einer Fläche durch Einstellen der Eigenschaften `vertexList` und `vertices` und Aufruf des Befehls `build` festlegen.

Beispiel

Die folgende Anweisung zeigt, dass die erste Seite des ersten Gitternetzes des Modells „Boden“ durch die ersten drei Vektoren in der Scheitelpunktliste der von „Boden“ verwendeten Modellressource definiert wird:

```
put member("Scene").model("Floor").meshdeform.mesh[1].face[1]
-- [1, 2, 3]
```

Siehe auch

`meshDeform (Modifizierer)`, `face`, `vertexList (mesh deform)`, `vertices`

far (fog)

Syntax

```
member(whichCastmember).camera(whichCamera).fog.far
sprite(whichSprite).camera{(index)}.fog.far
```

Beschreibung

Diese 3D-Kameraeigenschaft gibt an, in welcher Entfernung von der Kamera (in Welteinheiten) der Nebel seine maximale Dichte erreicht, wenn die Kameraeigenschaft `fog.enabled` auf `TRUE` gesetzt ist.

Der Standardwert dieser Eigenschaft lautet 1000.

Beispiel

Die folgende Anweisung setzt die Eigenschaft „far“ für den Nebel der Kamera `BayView` auf 5000. Wenn die Nebeleigenschaft enabled auf TRUE gesetzt ist, ist der Nebel 5000 Welteinheiten vor der Kamera am dichtesten.

```
-- Lingo syntax
member("MyYard").camera("BayView").fog.far = 5000

// JavaScript syntax
member("MyYard").getProp("camera",1).getPropRef("fog").far = 5000;
```

Siehe auch

`fog`, `near (fog)`

fieldOfView

Syntax

```
-- Lingo syntax
spriteObjRef.fieldOfView

// JavaScript syntax
spriteObjRef.fieldOfView;
```

Beschreibung

Diese QTVR-Sprite-Eigenschaft gibt das aktuelle Blickfeld des angegebenen Sprites in Grad an.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Die folgende Anweisung setzt die Eigenschaft `fieldOfView` von Kamera 1 auf 90:

```
-- Lingo syntax
member("3d world").camera[1].fieldOfView = 90

// JavaScript syntax
member("3d world").getProp("camera",1).fieldOfView = 90;
```

fieldOfView (3D)

Syntax

```
member(whichCastmember).camera(whichCamera).fieldOfView
sprite(whichSprite).camera{(index)}.fieldOfView
```

Beschreibung

Die Eigenschaft „3D-Kamera“ entspricht dem Winkel zwischen zwei Strahlen: einem von der Kamera an die obere Grade der Projektionsebene gezeichneten Strahl, und einem Strahl von der Kamera an die untere Grade der Projektsebene.

Die Bilder der Modelle in der 3D-Welt werden auf der Projektionsebene abgebildet, die sich vor der Kamera befindet, ähnlich wie eine Leinwand vor einem Filmprojektor. Die Projektionsebene ist das, was Sie im 3D-Sprite sehen. Die Ober- und Unterkante der Projektionsebene werden durch die Eigenschaft `fieldOfView` definiert. Wenn sich der Wert der Eigenschaft `fieldOfView` ändert, wird das Sprite jedoch nicht vergrößert oder verkleinert. Stattdessen wird das Bild der Projektionsebene skaliert, damit es in das Sprite-Rechteck passt.

Der Wert dieser Eigenschaft ist nur dann relevant, wenn die Kameraeigenschaft `projection` auf `#perspective` gesetzt ist. Ist die Eigenschaft `projection` auf `#orthographic` gesetzt, können Sie mit der Kameraeigenschaft `orthoHeight` die Ober- und Unterkante der Projektionsebene definieren.

Der Standardwert dieser Eigenschaft ist 30.0.

Beispiel

Die folgende Anweisung setzt die Eigenschaft `fieldOfView` von Kamera 1 auf 90:

```
member("3d world").camera[1].fieldOfView = 90
```

Siehe auch

`orthoHeight`

fileFreeSize

Syntax

```
-- Lingo syntax
_movie.fileFreeSize

// JavaScript syntax
_movie.fileFreeSize;
```

Beschreibung

Diese Filmeigenschaft gibt die Anzahl der ungenutzten Bytes im aktuellen Film zurück, die auf Änderungen an den Besetzungsbibliotheken oder Darstellern in einem Film zurückzuführen sind. Nur Lesen.

Mit den Befehlen `Save` und `Compact` und `Save as` kann die Datei neu geschrieben werden, um diesen ungenutzten Platz zu löschen.

Wenn im Film kein ungenutzter Speicherplatz vorliegt, gibt `fileFreeSize` den Wert 0 zurück.

Beispiel

Die folgende Anweisung zeigt die Anzahl der ungenutzten Byte im aktuellen Film an:

```
-- Lingo syntax
put(_movie.fileFreeSize)

// JavaScript syntax
put(_movie.fileFreeSize);
```

Siehe auch

Movie

fileName (Besetzung)

Syntax

```
-- Lingo syntax
castObjRef.fileName

// JavaScript syntax
castObjRef.fileName;
```

Beschreibung

Diese Besetzungsbibliothekseigenschaft gibt den Dateinamen einer Besetzungsbibliothek zurück oder legt ihn fest. Nur Lesen für interne Besetzungsbibliotheken. Lesen/Schreiben für externe Besetzungsbibliotheken.

Bei externen Besetzungsbibliotheken gibt `fileName` den vollen Pfad- und Dateinamen der Besetzung an.

Bei internen Besetzungsbibliotheken gibt `fileName` einen Wert zurück, der davon abhängig ist, welche interne Besetzungsbibliothek angegeben wurde.

- Ist die erste interne Besetzungsbibliothek angegeben, gibt `fileName` den Namen des Films zurück.
- Ist eine andere interne Besetzungsbibliothek angegeben, gibt `fileName` eine leere Zeichenfolge zurück.

Diese Eigenschaft akzeptiert URLs als Verweise. Soll jedoch eine Besetzungsbibliothek aus dem Internet verwendet und die Ladezeit minimiert werden, laden Sie zuerst die Datei der Besetzung mit der `downloadNetThing()`- oder `preloadNetThing()`-Methode auf einen lokalen Datenträger herunter und legen dann `fileName` auf die Datei auf dem Datenträger fest.

Beispiel

Die folgende Anweisung zeigt den Pfad- und Dateinamen der externen Besetzung "Buttons“ im Nachrichtenfenster an:

```
-- Lingo syntax
trace(castLib("Buttons").fileName)

// JavaScript syntax
trace(castLib("Buttons").fileName);
```

Die folgende Anweisung setzt den Dateinamen der externen Besetzung „Buttons“ auf „Inhalt.cst“:

```
-- Lingo syntax
castLib("Buttons").fileName = _movie.path & "Content.cst"

// JavaScript syntax
castLib("Buttons").fileName = _movie.path + "Content.cst";
```

Der Film verwendet jetzt die externe Besetzungsdatei „Inhalt.cst“ anstelle der Besetzung „Buttons“.

Die folgenden Anweisungen laden eine externe Besetzung von einer URL in den Anwendungsordner von Director und machen dann diese Datei zur externen Besetzung mit dem Namen „Besetzung der Tausende“:

```
-- Lingo syntax
downloadNetThing("http://wwwcbDeMille.com/Thousands.cst", _player.applicationPath &
"Thousands.cst")
castLib("Cast of Thousands").fileName = _player.applicationPath & "Thousands.cst"

// JavaScript syntax
downloadNetThing("http://wwwcbDeMille.com/Thousands.cst", _player.applicationPath +
"Thousands.cst");
castLib("Cast of Thousands").fileName = _player.applicationPath + "Thousands.cst";
```

Siehe auch

Cast Library, downloadNetThing, preloadNetThing()

fileName (Darsteller)

Syntax

```
-- Lingo syntax
memberObjRef.fileName

// JavaScript syntax
memberObjRef.fileName;
```

Beschreibung

Diese Darstellereigenschaft verweist auf den Namen der Datei, die einem verknüpften Darsteller zugeordnet ist. Lesen/Schreiben.

Diese Eigenschaft ist nützlich, um – wie beim Darstellertausch – eine externe verknüpfte Datei, die einem Darsteller zugeordnet ist, beim Abspielen des Films auszutauschen. Befindet sich die verknüpfte Datei in einem anderen Ordner als der Film, müssen Sie den Pfadnamen der Datei angeben.

Außerdem können Sie unverknüpfte Medien mit einem Darsteller verknüpfen, indem Sie den Dateinamen von Darstellertypen einstellen, die verknüpfte Medien unterstützen.

Diese Eigenschaft akzeptiert ebenfalls URLs als Verweise. Um jedoch eine Datei von einer URL verwenden und die Ladezeit minimieren zu können, laden Sie zuerst die Datei mit der `downloadNetThing()`- oder `preloadNetThing()`-Methode auf einen lokalen Datenträger herunter und legen dann die Eigenschaft `fileName` auf die Datei auf dem lokalen Datenträger fest.

Nach der Einstellung des Dateinamens verwendet Director diese Datei beim nächsten Gebrauch des Darstellers.

Beispiel

Die folgende Anweisung verknüpft den QuickTime-Film `ChairAnimation` mit dem Darsteller 40:

```
-- Lingo syntax
member(40).fileName = "ChairAnimation"

// JavaScript syntax
member(40).fileName = "ChairAnimation";
```

Die folgenden Anweisungen laden eine externe Datei von einer URL in den Anwendungsordner von Director und machen diese Datei zum Medium für den Sounddarsteller „Norma Desmond spricht“:

```
-- Lingo syntax
downloadNetThing("http://wwwcbDeMille.com/Talkies.AIF", _player.applicationPath &
"Talkies.AIF")
member("Norma Desmond Speaks").fileName = _player.applicationPath & "Talkies.AIF"

// JavaScript syntax
downloadNetThing("http://wwwcbDeMille.com/Talkies.AIF", _player.applicationPath +
"Talkies.AIF");
member("Norma Desmond Speaks").fileName = _player.applicationPath + "Talkies.AIF";
```

Siehe auch

downloadNetThing, Member, preloadNetThing()

fileName (MP4Media/FLV)

Syntax

```
put(sprite(1).fileName)
Member(1).fileName = "C:\downloads\movie.mp4"
```

Beschreibung

Diese MP4Media/FLV-Darstellereigenschaft legt den Dateinamen eines MP4Media/FLV-Darstellers fest. Diese Darstellereigenschaft verweist auf den Namen der Datei, die einem verknüpften Darsteller zugeordnet ist.

`fileName` ist eine schreib-/lesbare Eigenschaft.

Beispiele

In folgendem Beispiel wird die Eigenschaft `fileName` von „Sprite 7“ eingestellt.

```
-- Lingo syntax
sprite(7).fileName = "C:\Documents and Settings\user1\Desktop\DEMO_VIDEO\Ratatoullie.mp4"

// JavaScript syntax
sprite(7).fileName = "C:\Documents and Settings\user1\Desktop\DEMO_VIDEO\Ratatoullie.mp4";
```

fileName (Fenster)

Syntax

```
-- Lingo syntax
windowObjRef.fileName

// JavaScript syntax
windowObjRef.fileName;
```

Beschreibung

Diese Fenstereigenschaft verweist auf den Dateinamen des Films, der einem Fenster zugeordnet ist. Lesen/Schreiben.

Befindet sich die verknüpfte Datei in einem anderen Ordner als der Film, müssen Sie den Pfadnamen der Datei angeben.

Damit der Film in einem Fenster abgespielt werden kann, muss die Eigenschaft `fileName` auf den Dateinamen des Films eingestellt werden.

Die Eigenschaft `fileName` akzeptiert URLs als Verweis. Um jedoch eine Filmdatei von einer URL verwenden und die Ladezeit minimieren zu können, laden Sie zuerst die Datei mit der `downloadNetThing()`- oder `preloadNetThing()`-Methode auf einen lokalen Datenträger herunter und legen dann die Eigenschaft `fileName` auf die Datei auf dem lokalen Datenträger fest.

Beispiel

Die folgende Anweisung ordnet die Datei "Steuerpult“ dem Fenster "Werkzeugkasten“ zu:

```
-- Lingo syntax
window("Tool Box").fileName = "Control Panel"

// JavaScript syntax
window("Tool Box").fileName = "Control Panel";
```

Die folgende Anweisung zeigt den Namen der Datei an, die dem Fenster „Navigator“ zugewiesen ist:

```
-- Lingo syntax
trace(window("Navigator").fileName)

// JavaScript syntax
trace(window("Navigator").fileName);
```

Die folgenden Anweisungen laden eine Filmdatei von einer URL in den Anwendungsordner von Director und ordnen anschließend diese Datei dem Fenster „Meine Nahaufnahme“ zu:

```
-- Lingo syntax
downLoadNetThing("http://www.cbDeMille.com/Finale.DIR", _player.applicationPath &
"Finale.DIR")
window("My Close Up").fileName = _player.applicationPath & "Finale.DIR"

// JavaScript syntax
downLoadNetThing("http://www.cbDeMille.com/Finale.DIR", _player.applicationPath +
"Finale.DIR");
window("My Close Up").fileName = _player.applicationPath + "Finale.DIR";
```

Siehe auch

`downloadNetThing`, `preloadNetThing()`, `Window`

fileSize

Syntax

```
-- Lingo syntax
_movie.fileSize

// JavaScript syntax
_movie.fileSize;
```

Beschreibung

Diese Filmeigenschaft gibt die Anzahl der Byte im aktuellen Film zurück, die auf einem Datenträger gespeichert sind. Nur Lesen.

Die gleiche Zahl wird auch bei Auswahl des Befehls „Eigenschaften“ im Menü „Datei“ (Windows) bzw. „Information“ im Finder (Macintosh) zurückgegeben.

Beispiel

Die folgende Anweisung zeigt die Anzahl der Byte im aktuellen Film an:

```
-- Lingo syntax
put(_movie.fileSize)

// JavaScript syntax
put(_movie.fileSize);
```

Siehe auch

Movie

fileVersion

Syntax

```
-- Lingo syntax
_movie.fileVersion

// JavaScript syntax
_movie.fileVersion;
```

Beschreibung

Diese Filmeigenschaft gibt die Version von Director als Zeichenfolge (string) zurück, in der der Film zuletzt gespeichert wurde. Nur Lesen.

Beispiel

Die folgende Anweisung gibt die Version von Director zurück, in der der aktuelle Film zuletzt gespeichert wurde:

```
-- Lingo syntax
put(_movie.fileVersion)

// JavaScript syntax
put(_movie.fileVersion);
```

Siehe auch

Movie

fillColor

Syntax

```
-- Lingo syntax
memberObjRef.fillColor

// JavaScript syntax
memberObjRef.fillColor;
```

Beschreibung

Diese Vektorform-Darstellereigenschaft bezeichnet die Farbe einer Formfüllung, die als RGB-Wert angegeben wird.

`FillColor` kann verwendet werden, wenn die Eigenschaft `fillMode` der Form auf `#solid` oder `#gradient` gesetzt ist, nicht jedoch, wenn sie `#none` lautet. Ist `fillMode` auf `#gradient` eingestellt, gibt `fillColor` die Anfangsfarbe des Verlaufs an. Die Endfarbe wird mit der Eigenschaft `endColor` angegeben.

Diese Eigenschaft kann getestet und eingestellt werden.

Ein Beispiel für `fillColor` in einem fertigen Film finden Sie im Film „Vector Shapes“ im Ordner „Learning\\Lingo_Examples“ im Director-Anwendungsordner.

Beispiel

Die folgende Anweisung stellt die Füllfarbe des Darstellers "Archie“ auf einen neuen RGB-Wert ein.

```
-- Lingo syntax
member("Archie").fillColor = color( 24, 15, 153)

// JavaScript syntax
member("Archie").fillColor = color( 24, 15, 153);
```

Siehe auch

`endColor`, `fillMode`

fillCycles

Syntax

```
-- Lingo syntax
memberObjRef.fillCycles

// JavaScript syntax
memberObjRef.fillCycles;
```

Beschreibung

Diese Vektorform-Darstellereigenschaft gibt die Anzahl der Füllzyklen in der Füllung einer Verlaufsvektorform als ganzzahligen Wert zwischen 1 und 7 an.

Diese Eigenschaft ist nur gültig, wenn die Eigenschaft `fillMode` der Form auf `#gradient` gesetzt ist.

Diese Eigenschaft kann getestet und eingestellt werden.

Ein Beispiel für `fillCycles` in einem fertigen Film finden Sie im Film „Vector Shapes“ im Ordner „Learning\\Lingo_Examples“ im Director-Anwendungsordner.

Beispiel

Die folgende Anweisung setzt die Eigenschaft `fillCycles` des Darstellers „Archie“ auf 3:

```
-- Lingo syntax
member("Archie").fillCycles = 3

// JavaScript syntax
member("Archie").fillCycles = 3;
```

Siehe auch

endColor, fillColor, fillMode

fillDirection

Syntax

```
-- Lingo syntax
memberObjRef.fillDirection

// JavaScript syntax
memberObjRef.fillDirection;
```

Beschreibung

Diese Vektorform-Darstellereigenschaft gibt die Gradzahl an, die zum Drehen der Formfüllung notwendig ist.

Diese Eigenschaft ist nur gültig, wenn die Eigenschaft `fillMode` der Form auf `#gradient` gesetzt ist.

Diese Eigenschaft kann getestet und eingestellt werden.

Ein Beispiel für `fillDirection` in einem fertigen Film finden Sie im Film „Vector Shapes“ im Ordner „Learning\\Lingo_Examples“ im Director-Anwendungsordner.

Beispiel

Das folgende Verhalten ändert die Eigenschaft `fillMode` der Vektorform auf „gradient“ und generiert eine einfache Animation durch kontinuierliches Ändern der Eigenschaft `fillDirection` der Vektorform.

```
-- Lingo syntax
on beginSprite(me)
    member("VectorShape").fillMode = #gradient
end

on exitFrame(me)
    member("VectorShape").fillDirection = (member("VectorShape").fillDirection + 10 ) mod \ 360
end

// JavaScript syntax
function beginSprite(me)
{
    member("VectorShape").fillMode = symbol("gradient");
}

function exitFrame(me)
{
    member("VectorShape").fillDirection = (member("VectorShape").fillDirection + 10 ) % 360;
}
```

Siehe auch

fillMode

filled

Syntax

```
member(whichCastMember).filled
the filled of member whichCastMember
```

Beschreibung

Diese Formdarstellereigenschaft zeigt an, ob der angegebene Darsteller mit einem Muster (`TRUE`) gefüllt ist oder nicht (`FALSE`).

Beispiel

Die folgenden Anweisungen machen den Formdarsteller "Zielbereich“ zu einer gefüllten Form und ordnen ihm das Muster mit der Nummer 1, eine Vollfarbe, zu:

```
-- Lingo syntax
member("Target Area").filled = TRUE
member("Target Area").pattern = 1

// Java Script
member("Target Area").filled = true;
member("Target Area").pattern = 1;
```

Siehe auch

`fillColor`, `fillMode`

fillMode

Syntax

```
-- Lingo syntax
memberObjRef.fillMode

// JavaScript syntax
memberObjRef.fillMode;
```

Beschreibung

Diese Vektorform-Darstellereigenschaft zeigt die Füllmethode der Form an; folgende Werte sind möglich:

- `#none` – Die Form ist durchsichtig.
- `#solid` – Die Form wird mit einer Farbe gefüllt.
- `#gradient` – Die Form wird mit einem Verlauf aus zwei Farben gefüllt.

Diese Eigenschaft kann getestet und eingestellt werden, wenn die Form geschlossen ist; offen Formen haben keine Füllung.

Ein Beispiel für `fillMode` in einem fertigen Film finden Sie im Film „Vector Shapes“ im Ordner „Learning\\Lingo_Examples“ im Director-Anwendungsordner.

Beispiel

Die folgende Anweisung setzt die Eigenschaft `fillMode` des Darstellers "Archie" auf einen Verlauf:

```
-- Lingo syntax
member("Archie").fillMode = #gradient

// JavaScript syntax
member("Archie").fillMode = symbol("gradient");
```

Siehe auch

endColor, fillColor

fillOffset

Syntax

```
-- Lingo syntax
memberObjRef.fillOffset

// JavaScript syntax
memberObjRef.fillOffset;
```

Beschreibung

Diese Vektorform-Darstellereigenschaft gibt an, um wie viele Pixel der Mittelpunkt der Formfüllung horizontal und vertikal (innerhalb des `defaultRect`-Bereichs) verschoben wird.

Diese Eigenschaft ist nur gültig, wenn die Eigenschaft `fillMode` der Form auf `#gradient` gesetzt ist, kann jedoch getestet und eingestellt werden.

Ein Beispiel für `fillOffset` in einem fertigen Film finden Sie im Film „Vector Shapes“ im Ordner „Learning\\Lingo_Examples“ im Director-Anwendungsordner.

Beispiel

Die folgende Anweisung verschiebt den Mittelpunkt des Vektorformdarstellers „miette“ um 33 Pixel horizontal und um 27 Pixel vertikal:

```
-- Lingo syntax
member("miette").fillOffset = point(33, 27)

// JavaScript syntax
member("miette").fillOffset = point(33, 27);
```

Siehe auch

defaultRect, fillMode

fillScale

Syntax

```
-- Lingo syntax
memberObjRef.fillScale

// JavaScript syntax
memberObjRef.fillScale;
```

Beschreibung

Diese Vektorform-Darstellereigenschaft gibt an, in welchem Maße die Formfüllung skaliert werden soll. Im Vektorformfenster wird diese Eigenschaft auch als "Verteilung“ bezeichnet.

Diese Eigenschaft ist nur gültig, wenn die Eigenschaft `fillMode` der Form auf `#gradient` gesetzt ist, kann jedoch getestet und eingestellt werden.

Ein Beispiel für `fillScale` in einem fertigen Film finden Sie im Film „Vector Shapes“ im Ordner „Learning\\Lingo_Examples“ im Director-Anwendungsordner.

Beispiel

Die folgende Anweisung setzt die Eigenschaft `fillScale` des Darstellers "Archie" auf 33:

```
-- Lingo syntax
member("Archie").fillScale = 33.00

// JavaScript syntax
member("Archie").fillScale = 33.00;
```

Siehe auch

`fillMode`

filterlist

Syntax

```
-- Lingo syntax
spriteObjRef.filterlist
// JavaScript syntax
spriteObjRef.filterlist;
```

Beschreibung

Diese Sprite-Eigenschaft bestimmt, ob Bitmap-Filter auf ein Sprite angewendet werden. Es handelt sich um eine Liste, die über die „append“-Methode auf das Sprite angewendet werden kann.

Hinweis: *Die Methode „duplicate()“ funktioniert nicht für „filterlist“.*

Beispiel

Die erste Anweisung legt die Variable `myFilter` auf den Filter „Blur“ fest. Die nächste Zeile wendet den Weichzeichnungsfilter („blur“) auf „sprite(1)“ an.

```
--Lingo syntax
MyFilter=filter(#BlurFilter)
sprite(1).filterlist.append(MyFilter)
// JavaScript syntax
var MyFilter = filter(symbol("BlurFilter"));
sprite(1).filterlist.append(MyFilter);
```

Siehe auch

`Bitmap filters in Using Director.`

filterList (Mixer)

Syntax

```
mixer.filterList
```

Beschreibung

Diese Audiomixereigenschaft gibt die Liste der Audiofilter zurück, die zum Zeitpunkt des Aufrufs auf den Mixer angewendet werden.

Beispiele

```
--Lingo syntax
on mouseUp me
    put mixer1.filterlist -- Returns the filters currently applied to mixer1.
end

// JavaScript syntax
function mouseup()
{
put (mixer1.filterList); // Returns the filters currently applied to mixer1.
}
```

Siehe auch

Mixer

filterList (Soundobjekt)

Syntax

```
soundObject.filterList
```

Beschreibung

Diese Soundobjekteigenschaft gibt die Liste der Audiofilter zurück, die auf das Soundobjekt angewendet werden.

Beispiele

```
--Lingo syntax
on mouseUp me
    put soundObjRef.filterlist -- Displays the filter list for the sound object
-- associated with soundobjectRef.
end

// JavaScript syntax
function mouseUp(){
put (soundObjRef.filterList) ; //Displays the filter list for the sound object
// associated with soundobjectRef.

}
```

firstIndent

Syntax

```
-- Lingo syntax
chunkExpression.firstIndent

// JavaScript syntax
chunkExpression.firstIndent;
```

Beschreibung

Diese Textdarsteller-Eigenschaft enthält die Anzahl der Pixel, um die die erste Zeile von *chunkExpression* gegenüber dem linken Rand von *chunkExpression* eingerückt ist.

Eine Zahl kleiner als Null zeigt eine hängende Einrückung an, Null stellt keine Einrückung dar, und eine Zahl größer als Null zeigt eine normale Einrückung an.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Die folgende Anweisung setzt die Einrückung der ersten Zeile des Darstellers "Schreibtisch“ auf 0 Pixel.

```
--Lingo syntax
member("Desk").firstIndent = 0

// JavaScript syntax
member("Desk").firstIndent = 0;
```

Siehe auch

leftIndent, rightIndent

fixedLineSpace

Syntax

```
-- Lingo syntax
chunkExpression.fixedLineSpace

// JavaScript syntax
chunkExpression.fixedLineSpace;
```

Beschreibung

Diese Textdarsteller-Eigenschaft legt die Höhe jeder Zeile im Chunk-Ausdruck *chunkExpression* des Textdarstellers fest.

Der Wert selbst ist eine Ganzzahl und zeigt die absolute Pixelhöhe jeder Zeile an.

Der Standardwert ist Null und ergibt die natürliche Zeilenhöhe.

Beispiel

Die folgende Anweisung setzt die Höhe jeder Zeile des Darstellers "Schreibtisch“ auf 24 Pixel:

```
--Lingo syntax
member("Desk").fixedLineSpace = 24

// JavaScript syntax
member("Desk").fixedLineSpace = 24;
```

fixedRate

Syntax

```
-- Lingo syntax
memberOrSpriteObjRef.fixedRate

// JavaScript syntax
memberOrSpriteObjRef.fixedRate;
```

Beschreibung

Diese Darsteller- und Sprite-Eigenschaft steuert die Bildrate eines Flash-Films oder eines animierten GIFs. Die Eigenschaft `fixedRate` kann ganzzahlige Werte haben. Der Standardwert ist 15.

Diese Eigenschaft wird ignoriert, wenn die Eigenschaft `playbackMode` des Sprites nicht `#fixed` ist.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Die folgende Prozedur passt die Bildrate eines Flash-Film-Sprites an. Als Parameter akzeptiert die Prozedur einen Sprite-Bezug, einen Hinweis darauf, ob der Flash-Film beschleunigt oder verlangsamt werden soll, sowie die Zahl, um die die Geschwindigkeit angepasst werden soll.

```
-- Lingo syntax
on adjustFixedRate(whichSprite, adjustType, howMuch)
    case adjustType of
        #faster:
            sprite(whichSprite).fixedRate = sprite(whichSprite).fixedRate + howMuch
        #slower:
            sprite(whichSprite).fixedRate = sprite(whichSprite).fixedRate - howMuch
    end case
end

// JavaScript syntax
function adjustFixedRate(whichSprite, adjustType, howMuch) {
    switch(adjustType) {
        case "faster":
            sprite(whichSprite).fixedRate = sprite(whichSprite).fixedRate + howMuch;
            break;
        case "slower":
            sprite(whichSprite).fixedRate = sprite(whichSprite).fixedRate - howMuch;
            break;
    }
}
```

Siehe auch

playBackMode

fixStageSize

Syntax

```
-- Lingo syntax
_movie.fixStageSize

// JavaScript syntax
_movie.fixStageSize;
```

Beschreibung

Diese Filmeigenschaft bestimmt, ob die Bühnengröße nach dem Laden eines neuen Films unverändert bleibt (`TRUE`, Standard) oder nicht (`FALSE`). Dabei spielen die für diesen Film gespeicherte Bühnengröße oder die Einstellungen für `centerStage` keine Rolle. Lesen/Schreiben.

Die Eigenschaft `fixStageSize` kann die Bühnengröße eines laufenden Films nicht ändern.

Beispiel

Die folgende Anweisung bestimmt, ob die Eigenschaft `fixStageSize` aktiviert ist. Wenn `fixStageSize` den Wert FALSE aufweist, wird der Abspielkopf zum angegebenen Bild geschickt.

```
-- Lingo syntax
if (_movie.fixStageSize = FALSE) then
    _movie.go("proper size")
end if

// JavaScript syntax
if (_movie.fixStageSize == false) {
    _movie.go("proper size");
}
```

Die folgende Anweisung kehrt die Einstellung der Eigenschaft `fixStageSize` ins Gegenteil um:

```
-- Lingo syntax
_movie.fixStageSize = not(_movie.fixStageSize)

// JavaScript syntax
_movie.fixStageSize = !(_movie.fixStageSize);
```

Siehe auch

`centerStage`, `Movie`

flashRect

Syntax

```
-- Lingo syntax
memberObjRef.flashRect

// JavaScript syntax
memberObjRef.flashRect;
```

Beschreibung

Diese Darstellereigenschaft zeigt die Originalgröße eines Flash-Films oder Vektorformdarstellers an. Die Werte der Eigenschaften werden als Director-Rechteck angegeben, beispielsweise rect(0,0,32,32).

Bei verknüpften Flash-Darstellern gibt die Darstellereigenschaft `FlashRect` nur dann einen gültigen Wert zurück, wenn die Kopfzeile des Darstellers bereits in den Speicher geladen wurde.

Diese Eigenschaft kann getestet, aber nicht eingestellt werden.

Beispiel

Dieses Sprite-Skript passt die Größe des Flash-Film-Sprites an die Originalgröße seines Flash-Filmdarstellers an:

```
-- Lingo syntax
property spriteNum

on beginSprite me
    sprite(spriteNum).rect = sprite(spriteNum).member.FlashRect
end

// JavaScript syntax
function beginSprite() {
    sprite(this.spriteNum).rect = sprite(this.spriteNum).member.FlashRect;
}
```

Siehe auch

defaultRect, defaultRectMode, state (Flash, SWA)

flat

Syntax

```
member(whichCastmember).shader(whichShader).flat
member(whichCastmember).model(whichModel).shader.flat
member(whichCastmember).model(whichModel).shaderList{[index]}.flat
```

Beschreibung

Diese 3D-Eigenschaft für den Shader `#standard` gibt an, ob das Gitternetz mit flacher Schattierung (`TRUE`) oder mit Gouraud-Schattierung (`FALSE`) gerendert werden soll.

Bei der flachen Schattierung wird für jede Gitternetzfläche jeweils eine Farbe verwendet, und zwar die Farbe des ersten Scheitelpunkts der Fläche. Die flache Schattierung ist schneller als die Gouraud-Schattierung.

Bei der Gouraud-Schattierung wird jedem Scheitelpunkt einer Fläche eine Farbe zugeordnet; die restlichen Farben für die Fläche werden interpoliert und in einem Farbverlauf dargestellt. Die Gouraud-Schattierung erfordert zwar mehr Zeit und Rechenaufwand, erzeugt aber eine glattere Oberfläche.

Der Standardwert dieser Eigenschaft lautet `FALSE`.

Beispiel

Die folgende Anweisung setzt die Eigenschaft flat des Shaders "Wand" auf TRUE. Das Gitternetz eines Modells, das diesen Shader verwendet, wird mit einer Farbe pro Fläche gerendert.

```
-- Lingo syntax
member("MysteryWorld").shader("Wall").flat = TRUE

// JavaScript syntax
member("MysteryWorld").getProp("shader", 1).flat = true;
```

Siehe auch

mesh (Eigenschaft), colors, vertices, generateNormals()

flipH

Syntax

```
-- Lingo syntax
spriteObjRef.flipH

// JavaScript syntax
spriteObjRef.flipH;
```

Beschreibung

Diese Sprite-Eigenschaft gibt an, ob die Grafik eines Sprites horizontal auf der Bühne gespiegelt wurde (TRUE) oder nicht (FALSE). Nur Lesen.

Die eigentliche Grafik wird um ihr Registrierungskreuz gespiegelt.

Dabei wird jede Drehung oder Neigung beibehalten, und nur die eigentlichen Grafikdaten werden gespiegelt.

Beispiel

Die folgende Anweisung zeigt die Eigenschaft flipH von Sprite 5 an:

```
-- Lingo syntax
put(sprite(5).flipH)

// JavaScript syntax
put(sprite(5).flipH);
```

Siehe auch

flipV, rotation, skew, Sprite

flipV

Syntax

```
-- Lingo syntax
spriteObjRef.flipV

// JavaScript syntax
spriteObjRef.flipV;
```

Beschreibung

Diese Sprite-Eigenschaft gibt an, ob die Grafik eines Sprites vertikal auf der Bühne gespiegelt wurde (`TRUE`) oder nicht (`FALSE`). Nur Lesen.

Die eigentliche Grafik wird um ihr Registrierungskreuz gespiegelt.

Dabei wird jede Drehung oder Neigung beibehalten, und nur die eigentlichen Grafikdaten werden gespiegelt.

Beispiel

Die folgende Anweisung zeigt die Eigenschaft flipV von Sprite 5 an:

```
-- Lingo syntax
put(sprite(5).flipV)

// JavaScript syntax
put(sprite(5).flipV);
```

Siehe auch

`flipH`, `rotation`, `skew`, `Sprite`

floatPrecision

Syntax

```
the floatPrecision
```

Beschreibung

Diese Filmeigenschaft rundet Fließkommazahlen bei der Anzeige auf die Anzahl der angegebenen Dezimalstellen auf oder ab. Der Wert von `floatPrecision` muss eine Ganzzahl sein. Es sind maximal 15 Dezimalstellen möglich, der Standardwert beträgt 4.

Die Eigenschaft `floatPrecision` legt lediglich die Zahl der Ziffern zur Anzeige von Fließkommazahlen fest. Sie ändert nicht die Zahl der Dezimalziffern, die bei der Ausführung von Rechenoperationen verwendet werden.

- Wenn `floatPrecision` eine Zahl zwischen 1 und 15 ist, weisen Fließkommazahlen die angegebene Anzahl von Dezimalziffern nach dem Dezimalpunkt auf. Nachgestellte Nullen werden beibehalten.
- Wenn `floatPrecision` Null ist, werden Fließkommazahlen auf die nächste Ganzzahl auf- oder abgerundet. Es wird kein Dezimalpunkt angezeigt.
- Wenn `floatPrecision` eine negative Zahl ist, werden Fließkommazahlen auf den absoluten Wert der Dezimalstellenzahl auf- oder abgerundet. Nachgestellte Nullen fallen weg.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Die folgende Anweisung rundet die Quadratwurzel von 3.0 auf drei Dezimalstellen:

```
the floatPrecision = 3
x = sqrt(3.0)
put x
-- 1.732
```

Die folgende Anweisung rundet die Quadratwurzel von 3.0 auf acht Dezimalstellen:

```
the floatPrecision = 8
put x
-- 1.73205081
```

fog

Syntax

```
\-- Lingo Usage
member(whichCastmember).camera(whichCamera).fog.color
sprite(whichSprite).camera{(index)}.fog.color
member(whichCastmember).camera(whichCamera).fog.decayMode
sprite(whichSprite).camera{(index)}.fog.decayMode
member(whichCastmember).camera(whichCamera).fog.enabled
sprite(whichSprite).camera{(index)}.fog.enabled
member(whichCastmember).camera(whichCamera).fog.far
sprite(whichSprite).camera{(index)}.fog.far
member(whichCastmember).camera(whichCamera).fog.near
sprite(whichSprite).camera{(index)}.fog.near
// JavaScript Usage
member(whichCastmember).getProp("camera", whichCameraIndex).getPropRef("fog").color
sprite(whichSprite).camera.getPropRef("fog").color
member(whichCastmember).getProp("camera", whichCameraIndex).getPropRef("fog").decayMode
sprite(whichSprite).camera.getPropRef("fog").decayMode
member(whichCastmember).getProp("camera", whichCameraIndex).getPropRef("fog").enabled
sprite(whichSprite).camera.getPropRef("fog").enabled
member(whichCastmember).getProp("camera", whichCameraIndex).getPropRef("fog").far
sprite(whichSprite).camera.getPropRef("fog").far
member(whichCastmember).getProp("camera", whichCameraIndex).getPropRef("fog").near
sprite(whichSprite).camera.getPropRef("fog").near
```

Beschreibung

Diese 3D-Kameraeigenschaft erzeugt Nebel, der Modelle verfärbt und verwischt. Dieser Effekt steigert sich mit zunehmender Entfernung von der Kamera und ist echtem Nebel ähnlich, außer dass jede beliebige Farbe verwendet werden kann.

Siehe auch

color (fog), decayMode, enabled (fog), far (fog), near (fog)

folder

Syntax

```
-- Lingo syntax
dvdObjRef.folder

// JavaScript syntax
dvdObjRef.folder;
```

Beschreibung
Diese DVD-Eigenschaft bestimmt den Pfadnamen des Ordners, aus dem eine DVD wiedergegeben wird. Lesen/Schreiben.

Der Pfadname muss eine Zeichenfolge (string) sein.

Die Eigenschaft „folder“ kann entweder im Eigenschafteninspektor oder mit Skriptcode eingestellt werden. Für die aktuelle Implementierung gelten folgende Voraussetzungen:

Windows:

- Am Ende des Dateipfades für das lokale DVD-Zielmedium muss `video_ts` stehen. Beispielsweise „C:\video_ts“ oder „C:\myLocalDVDContent\video_ts“.

  Mac:

- Der Wert des in der Eigenschaft „folder“ angegebenen Pfades muss mit „/Volumes/“ beginnen.
- Das Hinzufügen von `video_ts` an den als Wert für die Eigenschaft „folder“ eingegebenen Pfad ist zurzeit optional. Wenn sich der DVD-Ordner `video_ts` beispielsweise im Stammordner des Startlaufwerks befindet, kann der Wert für die Eigenschaft „folder“ auf eine der folgenden zwei Arten eingegeben werden:
  - /Volumes/Mac HD/myLocalDVDContent/video_ts

    oder

  - /Volumes/Mac HD/`myLocalDVDContent`

So bearbeiten Sie den Wert der Eigenschaft „folder“ im Eigenschafteninspektor

1. Wählen Sie den DVD-Darsteller aus und aktivieren Sie in der Listenansicht des Eigenschafteninspektors die Registerkarte „DVD“.
2. Wählen Sie im Bereich „Wiedergabeeigenschaften“ das bearbeitbare Wertfeld der Eigenschaft „folder “ aus und geben Sie den Dateipfad des Speicherortes des DVD-Zielmediums ein.

Verwenden Sie folgende Beispiele als Richtlinie zum Festlegen der Ordnereigenschaft mithilfe von Skriptcode. Diese Anweisungen legen den Pfadnamen der DVD-Ordnereigenschaft fest:

Beispiel
Windows:

```
-- Lingo syntax
member(2).folder = "C:\myLocalDVDContent\video_ts"

// JavaScript syntax
member(2).folder = "C:\\myLocalDVDContent\video_ts";
```

Mac:

```
-- Lingo syntax
member(2).folder = "/Volumes/Mac HD/myLocalDVDContent"

// JavaScript syntax
member(2).folder = "/Volumes/Mac HD/myLocalDVDContent";
```

Hinweis: *Wird während des Erstellens des ersten DVD-Darstellers kein Ordner* `video_ts` *gefunden, wird der Warnhinweis "Unable to locate DVD volume." (Das DVD-Volume kann nicht gefunden werden.) angezeigt. Diese Warnung wird nur einmal pro Sitzung angezeigt. Zu diesem Zeitpunkt können Sie immer noch alle neu erstellten DVD-Darsteller benennen und dann deren Eigenschaft „folder“ auf einen Speicherort festlegen, der einen gültigen Ordner* `video_ts` *enthält.*

Probleme mit DVD-Ordnerpfadnamen auf Mac-Computern

Auf Mac-Computern muss im Pfadnamen für die Ordnereigenschaft anstelle des Mac-Standardtrennzeichens – dem Doppelpunkt (:) – ein Schrägstrich (/) als Trennzeichen verwendet werden. Zusätzlich muss am Anfang des Pfadnamens des DVD-Ordners "/volumes/" eingefügt werden. Befindet sich beispielsweise der DVD-Ordner im Stammverzeichnis des Startlaufwerks, sähe das Format wie folgt aus:

```
member (2).folder = "/Volumes/Mac HD/Test_DVD/video_ts"
```

Wird der Befehl `_movie.path` verwendet, um den Pfad des Projektors oder Films auf einem Mac-Computer abzurufen, enthält dieser anstelle eines Schrägstrichs (/) einen Doppelpunkt (:). Die Verwendung des Doppelpunkts im Pfadnamen des DVD-Ordners verursacht einen Fehler. Als Problemumgehung können Entwickler mithilfe eines Skripts die Doppelpunkte im Pfadnamen durch Schrägstriche ersetzen.

Siehe auch

DVD

font

Syntax

```
-- Lingo syntax
memberObjRef.font

// JavaScript syntax
memberObjRef.font;
```

Beschreibung

Diese Text- und Felddarstellereigenschaft legt die Schriftart fest, die zur Anzeige des angegebenen Darstellers verwendet wird. Dazu muss der Darsteller Zeichen enthalten, selbst wenn es sich nur um ein Leerzeichen handelt. Der Parameter *whichCastMember* kann entweder ein Darstellername oder eine Darstellernummer sein.

Die Darstellereigenschaft `font` kann getestet und eingestellt werden.

Ein Beispiel für `font` in einem fertigen Film finden Sie im Film "Text“ im Ordner „Learning/Lingo Examples“ im Director-Anwendungsordner.

Beispiel

Die folgende Anweisung setzt die Variable `oldFont` auf die aktuelle Fonteinstellung des Felddarstellers "Rokujo spricht":

```
-- Lingo syntax
oldFont = member("Rokujo Speaks").font

// JavaScript syntax
var oldFont = member("Rokujo Speaks").font;
```

Siehe auch

text, alignment, fontSize, fontStyle, lineHeight

fontSize

Syntax

```
-- Lingo syntax
memberObjRef.fontSize

// JavaScript syntax
memberObjRef.fontSize;
```

Beschreibung

Diese Felddarstellereigenschaft legt die Schriftgröße fest, die zur Anzeige des angegebenen Felddarstellers verwendet wird. Dazu muss der Darsteller Zeichen enthalten, selbst wenn es sich nur um ein Leerzeichen handelt. Der Parameter *whichCastMember* kann entweder ein Darstellername oder eine Darstellernummer sein.

Diese Eigenschaft kann getestet und eingestellt werden. Beim Testen der Eigenschaft wird die Höhe der ersten Zeile im Feld zurückgegeben. Beim Einstellen wirkt sie sich auf jede Zeile im Feld aus.

Ein Beispiel für `fontSize` in einem fertigen Film finden Sie im Film „Text“ im Ordner „Learning/Lingo Examples“ im Director-Anwendungsordner.

Beispiel

Die folgende Anweisung setzt die Variable `oldSize` auf die aktuelle Darstellereinstellung `fontSize` des Felddarstellers "Rokujo spricht":

```
--Lingo syntax
oldSize = member("Rokujo Speaks").fontSize

// JavaScript syntax
var oldSize = member("Rokujo Speaks").fontSize;
```

Die folgende Anweisung setzt die dritte Zeile des Textdarstellers `myMenu` auf 24 Punkt:

```
member("myMenu").fontSize = 12

// JavaScript syntax
member("myMenu").fontSize = 12;
```

Siehe auch

text, alignment, font, fontStyle, lineHeight

fontStyle

Syntax

```
-- Lingo syntax
memberObjRef.fontStyle
memberObjRef.char[whichChar].fontStyle
memberObjRef.line[whichLine].fontStyle
memberObjRef.word[whichWord].fontStyle

// JavaScript syntax
memberObjRef.fontStyle;
memberObjRef.getPropRef("char", whichChar).fontStyle
memberObjRef.getPropRef("line", whichLine).fontStyle;
memberObjRef.getPropRef("word", whichWord).fontStyle;
```

Beschreibung

Diese Darstellereigenschaft legt den Schriftstil für den angegebenen Felddarsteller, das angegebene Zeichen, die angegebene Zeile, das angegebene Wort oder den angegebenen Chunk-Ausdruck fest. Dazu muss der Felddarsteller Zeichen enthalten, selbst wenn es sich nur um ein Leerzeichen handelt.

Der Wert der Eigenschaft ist ein String mit Stilen, die durch Kommas voneinander getrennt sind. Lingo verwendet eine Schrift, die eine Kombination der Stile im String darstellt. Verfügbare Stile sind „plain“ (Normal), „bold“ (Fett), „italic“ (Kursiv), „underline“ (Unterstrichen), „shadow“ (Schatten), „outline“ (Kontur) und „extended“ (Erweitert). Auf dem Mac ist außerdem noch „condensed“ (Eng) verfügbar.

Mit der Stilart „plain“ können Sie alle aktuellen Stile entfernen. Der Parameter *whichCastMember* kann entweder ein Darstellername oder eine Darstellernummer sein.

Diese Eigenschaft kann getestet und eingestellt werden.

Ein Beispiel für `fontStyle` in einem fertigen Film finden Sie im Film „Text“ im Ordner „Learning/Lingo Examples“ im Director-Anwendungsordner.

Beispiel

Die folgende Anweisung setzt die Variable `oldStyle` auf die aktuelle `fontStyle`-Einstellung des Felddarstellers "Rokujo spricht":

```
--Lingo syntax
oldStyle = member("Rokujo Speaks").fontStyle

// JavaScript syntax
var oldStyle = member("Rokujo Speaks").fontStyle;
```

Die folgende Anweisung setzt die Darstellereigenschaft `fontStyle` des Darstellers „Gedicht“ auf @@@bold und @@@italic:

```
--Lingo syntax
member("Poem").fontStyle = "bold, italic"

// JavaScript syntax
member("Poem").fontStyle = "bold, italic";
```

Die folgende Anweisung setzt die Eigenschaft `fontStyle` des dritten Worts des Darstellers „Namen des Sohnes“ auf @@@#italic:

```
--Lingo syntax
member("Son's Names").word[3].fontStyle = "italic"

// JavaScript syntax
member("Son's Names").getPropRef("word", 3).fontStyle = "italic";
```

Diese Anweisung legt die Eigenschaft `fontStyle` des Textdarstellers auf ein fettes bzw. ein fettes und kursives Format fest:

```
--Lingo syntax
member("text").fontStyle=[#bold]
or
member("text").fontStyle=[#bold, #italic]

// JavaScript syntax
member("text").fontStyle = list(symbol("bold"));
```

Siehe auch

`text`, `alignment`, `fontSize`, `font`, `lineHeight`

foreColor

Syntax

```
-- Lingo syntax
spriteObjRef.foreColor

// JavaScript syntax
spriteObjRef.foreColor;
```

Beschreibung

Diese Sprite-Eigenschaft gibt die Vordergrundfarbe eines Sprites zurück oder legt sie fest. Lesen/Schreiben.

Diese Eigenschaft sollte nicht auf Bitmapdarsteller mit einer größeren Farbtiefe als 1 Bit angewendet werden, da dies zu unerwarteten Ergebnissen führen kann.

Anstelle der Eigenschaft `foreColor` sollte die neuere Eigenschaft `color` verwendet werden.

Beispiel

Die folgende Anweisung setzt die Variable `oldColor` auf die Vordergrundfarbe von Sprite 5:

```
-- Lingo syntax
oldColor = sprite(5).foreColor

// JavaScript syntax
var oldColor = sprite(5).foreColor;
```

Die folgende Anweisung legt 36 als Nummer der Vordergrundfarbe eines nach dem Zufallsprinzip ausgewählten Sprites zwischen 11 und 13 fest.

```
-- Lingo syntax
sprite(10 + random(3)).foreColor = 36

// JavaScript syntax
sprite(10 + random(3)).foreColor = 36;
```

Siehe auch

backColor, color(), Sprite

frame

Syntax

```
-- Lingo syntax
_movie.frame

// JavaScript syntax
_movie.frame;
```

Beschreibung

Diese Filmeigenschaft gibt die Nummer des aktuellen Bildes im Film zurück. Nur Lesen.

Beispiel

Die folgende Anweisung schickt den Abspielkopf zum Bild vor dem aktuellen Bild:

```
-- Lingo syntax
_movie.go(_movie.frame - 1)

// JavaScript syntax
_movie.go(_movie.frame - 1);
```

Siehe auch

go(), Movie

frameCount

Syntax

```
-- Lingo syntax
memberObjRef.frameCount

// JavaScript syntax
memberObjRef.frameCount;
```

Beschreibung

Diese Flash-Darstellereigenschaft gibt die Anzahl der Bilder im Flash-Filmdarsteller an. Die Darstellereigenschaft `frameCount` kann ganzzahlige Werte aufweisen.

Diese Eigenschaft kann getestet, aber nicht eingestellt werden.

Beispiel

Das folgende Sprite-Skript gibt im Nachrichtenfenster die Kanalnummer und die Anzahl der Bilder in einem Flash-Film an.

```
-- Lingo syntax
property spriteNum

on beginSprite me
    put(""The Flash movie in channel"" && spriteNum && has"" &&
/sprite(spriteNum).member.frameCount && ""frames.""
end

// JavaScript syntax
function beginSprite() {
    trace("The Flash movie in channel " + (this.spriteNum) + " has " +
sprite(this.spriteNum).member.frameCount + " frames.");
}
```

frameLabel

Syntax

```
-- Lingo syntax
_movie.frameLabel

// JavaScript syntax
_movie.frameLabel;
```

Beschreibung

Diese Filmeigenschaft gibt die Beschriftung an, die dem aktuellen Bild zugeordnet ist. Lesen/Schreiben; nur während einer Drehbuchaufzeichnungssitzung.

Hat das aktuelle Bild keine Beschriftung, ist der Wert der Eigenschaft `frameLabel` 0.

Beispiel

Die folgende Anweisung überprüft die Beschriftung des aktuellen Bildes. In diesem Fall ist der aktuelle Wert von `frameLabel` "Start":

```
-- Lingo syntax
put(_movie.frameLabel)

// JavaScript syntax
put(_movie.frameLabel);
```

Siehe auch

`labelList`, `Movie`

framePalette

Syntax

```
-- Lingo syntax
_movie.framePalette

// JavaScript syntax
_movie.framePalette;
```

Beschreibung

Diese Filmeigenschaft kennzeichnet die Darstellernummer der Palette, die im aktuellen Bild verwendet wird. Dabei handelt es sich entweder um die aktuelle oder die im aktuellen Bild eingestellte Palette. Lesen/Schreiben; nur während einer Drehbuchaufzeichnungssitzung.

Beispiel

Die folgende Anweisung überprüft die Palette des aktuellen Bildes. In diesem Fall ist die Palette Darsteller 45.

```
-- Lingo syntax
put(_movie.framePalette)

// JavaScript syntax
put(_movie.framePalette);
```

Die folgende Anweisung macht Palettendarsteller 45 zur Palette für das aktuelle Bild:

```
-- Lingo syntax
_movie.framePalette = 45

// JavaScript syntax
_movie.framePalette = 45;
```

Siehe auch

Movie

frameRate

Syntax

```
-- Lingo syntax
memberObjRef.frameRate

// JavaScript syntax
memberObjRef.frameRate;
```

Beschreibung

Diese Darstellereigenschaft gibt die Bildrate an, mit der das angegebene Digitalvideo bzw. der angegebene Flash-Filmdarsteller abgespielt wird.

Die möglichen Werte für die Bildrate eines Digitalvideo-Darstellers stimmen mit den Optionsfeldern zur Auswahl der Abspieloptionen von Digitalvideos überein.

- Wenn die Darstellereigenschaft `frameRate`zwischen 1 und 255 liegt, spielt der Digitalvideofilm jedes Bild mit dieser Bildrate ab. Der Wert für die Darstellereigenschaft `frameRate` kann nicht größer als 255 sein.
- Wenn die Darstellereigenschaft `frameRate` auf –1 oder 0 gesetzt ist, spielt der Digitalvideofilm jedes Bild mit der normalen Bildrate ab. Auf diese Weise lässt sich das Video mit seiner Tonspur synchronisieren. Wenn die Bildrate `frameRate` auf einen andern Wert als -1 oder 0 gesetzt ist, wird die Tonspur des Digitalvideos nicht abgespielt.
- Wenn die Darstellereigenschaft `frameRate` auf –2 eingestellt ist, spielt der Digitalvideofilm jedes Bild so schnell wie möglichab.

Bei einem Flash-Filmdarsteller gibt die Eigenschaft die Bildrate des in Flash erstellten Films an.

Diese Eigenschaft kann getestet, aber nicht eingestellt werden.

Beispiel

Die folgende Anweisung setzt die Bildrate des QuickTime-Digitalvideo-Darstellers "Drehstuhl“ auf 30 Bilder pro Sekunde:

```
-- Lingo syntax
member("Rotating Chair").frameRate = 30

// JavaScript syntax
member("Rotating Chair").frameRate = 30;
```

Die folgende Anweisung weist den QuickTime-Digitalvideo-Darsteller „Drehstuhl“ an, jedes Bild so schnell wie möglich abzuspielen:

```
-- Lingo syntax
member("Rotating Chair").frameRate = -2

// JavaScript syntax
member("Rotating Chair").frameRate = -2;
```

Das folgende Sprite-Skript prüft, ob der Darsteller des Sprites ursprünglich in Flash mit einer Bildrate von unter 15 Bildern pro Sekunde erstellt wurde. Wenn die Bildrate des Films niedriger als 15 Bilder pro Sekunde ist, stellt das Skript die Eigenschaft `playBackMode` für das Sprite so ein, dass es mit einer anderen Bildrate wiedergegeben werden kann. Anschließend setzt das Skript die Sprite-Eigenschaft `fixedRate` auf 15 BpS.

```
-- Lingo syntax
property spriteNum

on beginSprite me
    if sprite(spriteNum).member.frameRate < 15 then
        sprite(spriteNum).playBackMode = #fixed
        sprite(spriteNum).fixedRate = 15
    end if
end

// JavaScript syntax
function beginSprite () {
    var fr = sprite(this.spriteNum).member.frameRate;
    if (fr < 15) {
        sprite(this.spriteNum).playBackMode = symbol("fixed");
        sprite(this.spriteNum).fixedRate = 15;
    }
}
```

Siehe auch

fixedRate, playRate (QuickTime, AVI, MP4, FLV), currentTime (QuickTime, AVI), playBackMode

frameRate (DVD)

Syntax

```
-- Lingo syntax
dvdObjRef.frameRate

// JavaScript syntax
dvdObjRef.frameRate;
```

Beschreibung
Diese DVD-Eigenschaft gibt den `frameRate`-Wert der DVD zurück. Nur Lesen.

Der `frameRate`-Wert wird als eine der folgenden Fließkommazahlen zurückgegeben:

Fließkommazahl (float)	Beschreibung
0.0	Der `frameRate`-Wert konnte nicht ermittelt werden, weil er sich nicht in der Titeldomäne befindet oder weil der Titel kein sequenzieller Videotitel ist.
25.0	Die DVD ist mit einer Wiedergabegeschwindigkeit von 25 Bildern pro Sekunde erstellt.
30.0	Die DVD ist mit einer Wiedergabegeschwindigkeit von 30 Bildern pro Sekunde erstellt.
29.97	Die DVD ist mit einer Wiedergabegeschwindigkeit von 29.97 Bildern pro Sekunde erstellt.

Siehe auch
DVD

frameRate (MP4Media/FLV)

Syntax
```
put member(1).frameRate
```

Beschreibung
Darstellereigenschaft, die der Wiedergabe-Bildfrequenz des angegebenen MP4Media-Darstellers entspricht.

Beispiel
Die folgende Syntax zeigt die Bildfrequenz des MP4Media-Darstellers in Bilder pro Sekunde an:

```
-- Lingo syntax
put member("MP4Media/FLV").frameRate

// JavaScript syntax
put member("MP4Media/FLV").frameRate;
```

frameScript

Syntax
```
-- Lingo syntax
_movie.frameScript

// JavaScript syntax
_movie.frameScript;
```

Beschreibung
Diese Filmeigenschaft enthält die eindeutige Darstellernummer des Bildskripts, das dem aktuellen Bild zugeordnetist. Lesen/Schreiben; nur während einer Drehbuchaufzeichnungssitzung.

Während einer Drehbucherstellungssitzung können Sie ebenfalls ein Bildskript dem aktuellen Bild zuordnen, indem Sie die Eigenschaft `frameScript` einstellen.

Ist dem aktuellen Bild kein Bildskript zugewiesen, gibt die Eigenschaft 0 zurück.

Beispiel

Die folgende Anweisung zeigt die Nummer des dem aktuellen Bild zugeordneten Skripts an. In diesem Fall ist die Skriptnummer 25.

```
-- Lingo syntax
put(_movie.frameScript)

// JavaScript syntax
put(_movie.frameScript);
```

Die folgende Anweisung macht den Skriptdarsteller „Schaltflächenreaktionen“ zum Bildskript des aktuellen Bildes:

```
-- Lingo syntax
_movie.frameScript = member("Button responses")

// JavaScript syntax
_movie.frameScript = member("Button responses");
```

Siehe auch

Movie

frameSound1

Syntax

```
-- Lingo syntax
_movie.frameSound1

// JavaScript syntax
_movie.frameSound1;
```

Beschreibung

Diese Filmeigenschaft ermittelt die Nummer des Darstellers, der dem ersten Soundkanal im aktuellen Bild zugeordnet ist. Lesen/Schreiben.

Diese Eigenschaft kann auch während einer Drehbuchaufzeichnungssitzung eingestellt werden.

Beispiel

Als Teil der Drehbuchaufzeichnung ordnet die folgende Anweisung den Sounddarsteller "Jazz“ dem ersten Soundkanal zu:

```
-- Lingo syntax
_movie.frameSound1 = member("Jazz").number

// JavaScript syntax
_movie.frameSound1 = member("Jazz").number;
```

Siehe auch

frameSound2, Movie

frameSound2

Syntax

```
-- Lingo syntax
_movie.frameSound2

// JavaScript syntax
_movie.frameSound2;
```

Beschreibung

Diese Filmeigenschaft ermittelt die Nummer des Darstellers, der dem zweiten Soundkanal im aktuellen Bild zugeordnet ist. Lesen/Schreiben.

Diese Eigenschaft kann auch während einer Drehbuchaufzeichnungssitzung eingestellt werden.

Beispiel

Als Teil der Drehbuchaufzeichnung ordnet die folgende Anweisung den Sounddarsteller "Jazz“ dem zweiten Soundkanal zu:

```
-- Lingo syntax
_movie.frameSound2 = member("Jazz").number

// JavaScript syntax
_movie.frameSound2 = member("Jazz").number;
```

Siehe auch

frameSound1, Movie

frameTempo

Syntax

```
-- Lingo syntax
_movie.frameTempo

// JavaScript syntax
_movie.frameTempo;
```

Beschreibung

Diese Filmeigenschaft gibt das Tempo an, das dem aktuellen Bild zugeordnet ist. Lesen/Schreiben; nur während einer Drehbuchaufzeichnungssitzung.

Beispiel

Die folgende Anweisung überprüft das Tempo des aktuellen Bildes. In diesem Fall beträgt das Tempo 15 Bilder pro Sekunde.

```
-- Lingo syntax
put(_movie.frameTempo)

// JavaScript syntax
put(_movie.frameTempo);
```

Siehe auch

Movie, puppetTempo()

frameTransition

Syntax

```
-- Lingo syntax
_movie.frameTransition

// JavaScript syntax
_movie.frameTransition;
```

Beschreibung

Diese Filmeigenschaft gibt die Nummer des dem aktuellen Bild zugeordneten Übergangsdarstellers an. Lesen/Schreiben; nur während einer Drehbuchaufzeichnungssitzung zum Angeben von Übergängen.

Beispiel

Bei einer Drehbuchaufzeichnungssitzung macht die folgende Anweisung den Darsteller „Nebel“ zum Übergang für das Bild, das Lingo momentan aufzeichnet

```
-- Lingo syntax
_movie.frameTransition = member("Fog")

// JavaScript syntax
_movie.frameTransition = member("Fog");
```

Siehe auch

Movie

front

Syntax

```
member(whichCastmember).modelResource(whichModelResource).front
```

Beschreibung

Diese 3D-Eigenschaft für die Modellressource `#box` gibt an, ob die Seite der Box, die von der -Z-Achse geschnitten wird, geschlossen (`TRUE`) oder offen ist (`FALSE`).

Der Standardwert dieser Eigenschaft lautet `TRUE`.

Beispiel

Die folgende Anweisung setzt die Eigenschaft front der Modellressource "Kiste" auf FALSE, d. h. die Vorderseite der Box ist offen:

```
member("3D World").modelResource("Crate").front = FALSE
```

Siehe auch

back, bottom (3D), top (3D), left (3D), right (3D)

frontWindow

Syntax

```
-- Lingo syntax
_player.frontWindow

// JavaScript syntax
_player.frontWindow;
```

Beschreibung

Diese Player-Eigenschaft gibt an, welcher Film im Fenster (MIAW) gerade auf dem Bildschirm an vorderster Stelle steht. Nur Lesen.

Wenn die Bühne ganz vorne steht, ist das vorderste Fenster (`frontWindow`) die Bühne. Wenn ein Medien-Editor oder eine schwebende Palette an vorderster Stelle steht, gibt `frontWindowVOID` (Lingo) oder `null` (JavaScript-Syntax) zurück.

Beispiel

Die folgende Anweisung stellt fest, ob das Fenster „Musik“ gegenwärtig das vorderste Fenster ist, und bringt in diesem Fall das Fenster „Versuche dies“ in den Vordergrund:

```
-- Lingo syntax
if (_player.frontWindow = "Music") then
    window("Try This").moveToFront()
end if

// JavaScript syntax
if (_player.frontWindow == "Music") {
    window("Try This").moveToFront();
}
```

Siehe auch

Player

fullScreen

Syntax

```
-- Lingo syntax
dvdObjRef.fullScreen

// JavaScript syntax
dvdObjRef.fullScreen;
```

Beschreibung

Diese DVD-Eigenschaft gibt an, ob die DVD im Vollbildmodus wiedergegeben werden soll. Lesen/Schreiben.

Durch Drücken der ESC-Taste wird der Vollbildmodus deaktiviert und die Eigenschaft auf „false“ festgelegt.

Wird zurzeit auf dem Mac nicht unterstützt.

Beispiel

Die folgende Anweisung veranlasst die DVD zur Wiedergabe im Vollbildmodus.

```
-- Lingo syntax
member("DVDMember").fullScreen = TRUE

// Java Script
member("DVDMember").fullScreen = true;
```

Siehe auch

DVD

getBoneID

Syntax

```
memberReference.modelResource.getBoneID("boneName")
```

Beschreibung

Diese 3D-Eigenschaft für Modellressourcen gibt die Indexnummer des Knochens *boneName* in der Modellressource zurück. Wenn kein Knochen mit dem angegebenen Namen zu finden ist, wird „0“ zurückgegeben.

Beispiel

Die folgende Anweisung gibt eine ID-Nummer für den Knochen "SchienbeinL" zurück:

```
put member("ParkScene").modelResource("LittleKid").getBoneId("ShinL")
-- 40
```

Siehe auch

bone

globals

Syntax

```
the globals
```

Beschreibung

Diese Systemeigenschaft enthält eine spezielle Eigenschaftsliste aller aktuellen globalen Variablen mit einem Wert ungleich `VOID`. Jede globale Variable ist als Eigenschaft zusammen mit ihrem Wert in der Liste aufgeführt.

Sie können die folgenden Listenoperationen mit `globals` verwenden:

- `count()` – Gibt die Anzahl der Listeneinträge zurück.
- `getPropAt(n)` – Gibt den Namen des *n*-ten Eintrags zurück.
- `getProp(x)` – Gibt den Wert eines Eintrags mit dem angegebenen Namen zurück.
- `getAProp(x)` – Gibt den Wert eines Eintrags mit dem angegebenen Namen zurück.

Hinweis: *Die Eigenschaft `globals` enthält automatisch die Eigenschaft `#version`, welche die aktuell ausgeführte Version von Director angibt. Aus diesem Grund enthält die Liste immer mindestens einen Eintrag, selbst wenn noch keine globalen Variablen deklariert wurden.*

Diese Eigenschaft unterscheidet sich von `showGlobals` insofern, als dass `the globals` auch in einem anderen Kontext als dem Nachrichtenfenster verwendet werden kann. Wenn Sie `the globals` im Nachrichtenfenster anzeigen möchten, verwenden Sie den Befehl`showGlobals`.

Siehe auch

`showGlobals()`, `clearGlobals()`

glossMap

Syntax

```
member(whichCastmember).shader(whichShader).glossMap
member(whichCastmember).model(whichModel).shader.glossMap
member(whichCastmember).model(whichModel).shaderList{[index]}.glossMap
```

Beschreibung

Diese 3D-Eigenschaft für den Shader `#standard` gibt die Textur an, die zum Gloss Mapping verwendet wird.

Beim Einstellen dieser Eigenschaft werden die folgenden Eigenschaften automatisch gesetzt:

- Die vierte Texturebene des Shaders wird auf die angegebene Textur gesetzt.
- Der Wert von `textureModeList[4]` wird auf `#none` gesetzt.
- Der Wert von `blendFunctionList[4]` wird auf `#multiply` gesetzt.

Beispiel

Die folgende Anweisung legt die Textur "Oval" als `glossMap`-Wert des vom Modell `Glassbox` verwendeten Shaders fest:

```
-- Lingo syntax
member("3DPlanet").model("GlassBox").shader.glossMap = member("3DPlanet").texture("Oval")

// Java Script
member("House").getPropRef("model", 1).shaderList[1].glossMap =
member("House").getPropRef("texture",1);
```

Siehe auch

blendFunctionList, textureModeList, region, specularLightMap, diffuseLightMap

gravity

Syntax

```
member(whichCastmember).modelResource(whichModelResource).gravity
```

Beschreibung

Mit dieser 3D-Eigenschaft für Partikelmodellressourcen können Sie bei einer Modellressource vom Typ `#particle` die Eigenschaft `gravity` der Ressource als Vektor ermitteln und festlegen.

Diese Eigenschaft definiert die Schwerkraft, die in jedem Simulationsschritt auf alle Partikel ausgeübt wird.

Der Standardwert dieser Eigenschaft lautet `vector(0,0,0)`.

Beispiel

In diesem Beispiel ist `ThermoSystem` eine Modellressource vom Typ `#particle`. Die folgende Anweisung setzt die Eigenschaft gravity von `ThermoSystem` auf vector(0, -.1, 0), damit die Partikel von `thermoSystem` entlang der y-Achse leicht nach unten gezogen werden:

```
-- Lingo syntax
member("Fires").modelResource("ThermoSystem").gravity = vector(0, -.1, 0)

// JavaScript syntax
member("Fires").getProp("modelResource", 1).gravity = vector(0, -.1, 0);
```

Siehe auch

drag, wind

gradientType

Syntax

```
-- Lingo syntax
memberObjRef.gradientType

// JavaScript syntax
memberObjRef.gradientType;
```

Beschreibung

Diese Vektorform-Darstellereigenschaft gibt den aktuellen Verlauf in der Füllung des Darstellers an.

Mögliche Werte: `#linear` oder `#radial`. Die Eigenschaft `gradientType` ist nur gültig, wenn `fillMode` auf `#gradient` gesetzt ist.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Die folgende Prozedur schaltet zwischen linearen und radialen Verläufen im Darsteller "backdrop“ um.

```
-- Lingo syntax
on mouseUp me
    if member("backdrop").gradientType = #radial then
        member("backdrop").gradientType = #linear
    else
        member("backdrop").gradientType = #radial
    end if
end

// JavaScript syntax
function mouseUp() {
    var gt = member("backdrop").gradientType;
    if (gt == "radial") {
        member("backdrop").gradientType = symbol("linear");
    } else {
        member("backdrop").gradientType = symbol("radial");
    }
}
```

Siehe auch

`fillMode`

group

Syntax

```
member(whichCastmember).group(whichGroup)
member(whichCastmember).group[index]
```

Beschreibung

Dieses 3D-Element ist ein Node in der 3D-Welt, der einen Namen, eine Transformation, einen Parent-Node und Child-Nodes aufweist, aber keine anderen Eigenschaften.

Jeder 3D-Darsteller besitzt eine Standardgruppe namens "World“, die nicht gelöscht werden kann. Die Parent-Hierarchie aller in der 3D-Welt vorhandenen Modelle, Lichtquellen, Kameras und Gruppen endet mit `group("world")`.

Beispiel

Die erste Zeile des folgenden Beispiels zeigt die zweite Gruppe des Darstellers „3Dobjects“ an. Die zweite Zeile zeigt die Gruppe „RBCollection01“ des Darstellers „3Dobjects“ an.

```
-- Lingo syntax
put member("3Dobjects").group("RBCollection01")
put member("3Dobjects").group[2]

// Javascript
put(member("3Dobjects").getPropRef("group",2));
```

Siehe auch

newGroup, deleteGroup, child (3D), parent

height

Syntax

```
-- Lingo syntax
imageObjRef.height
memberObjRef.height
spriteObjRef.height

// JavaScript syntax
imageObjRef.height;
memberObjRef.height;
spriteObjRef.height;
```

Beschreibung

Diese Grafik-, Darsteller- und Sprite-Eigenschaft bestimmt die Höhe (in Pixel) des auf der Bühne angezeigten Vektorform-, Flash-, animierten GIF-, `RealMedia`-, Windows Media-, Bitmap- bzw. Formdarstellers. Nur Lesen für Darsteller und Grafikobjekte. Lesen/Schreiben für Sprites.

Beispiel

Die folgende Anweisung ordnet die Höhe des Darstellers "Schlagzeile" der Variablen vHeight zu:

```
-- Lingo syntax
vHeight = member("Headline").height

// JavaScript syntax
var vHeight = member("Headline").height;
```

Die folgende Anweisung setzt die Höhe von Sprite 10 auf 26 Pixel:

```
-- Lingo syntax
sprite(10).height = 26

// JavaScript syntax
sprite(10).height = 26;
```

Siehe auch

Member, Sprite, width

height (3D)

Syntax

```
member(whichCastmember).modelResource(whichModelResource).height
member(whichCastmember).texture(whichTexture).height
```

Beschreibung

Diese 3D-Eigenschaft für Modellressourcen vom Typ `#box` und `#cylinder` sowie für Texturen gibt die Höhe des Objekts an.

Die Höhe einer `#box`- oder `#cylinder`-Modellressource wird in Welteinheiten gemessen und kann getestet und eingestellt werden. Der Standardwert dieser Eigenschaft lautet 50.

Die Höhe einer Textur wird in Pixel gemessen und kann getestet, aber nicht gesetzt werden. Die Texturhöhe wird von der Höhe der Texturquelle auf die nächste Potenz von 2 gerundet.

Beispiel

Die folgende Anweisung setzt die Höhe der Modellressource "Turm" auf 225.0 Welteinheiten:

```
member("3D World").modelResource("Tower").height = 225.0
```

Die folgende Anweisung zeigt, dass die Höhe der Textur „Marskarte“ 512 Pixel beträgt:

```
put member("scene").texture("Marsmap").height
-- 512
```

Siehe auch

`length (3D)`, `width (3D)`

height (MP4Media/FLV)

Syntax

```
put member(1).height
Sprite(1).height = 30
```

Beschreibung

Darsteller- und Sprite-Eigenschaft, die für MP4Media/FLV-Darsteller deren Höhe in Pixel des MP4-Originalvideos des Autors entspricht, das auf der Bühne angezeigt wird. Diese Eigenschaft kann für MP4Media/FLV-Darsteller nur ausgelesen werden und nur für MP4Media-Sprites beschrieben werden.

Beispiele

In folgendem Beispiel wird die Höhe des Darstellers „MP4Media/FLV“ der Variable `vHeight` zugewiesen:

```
-- Lingo syntax
vHeight = member("MP4Media/FLV").height

// JavaScript syntax
var vHeight = member("MP4Media/FLV").height;
```

Das folgende Beispiel setzt die Höhe des MP4Video-Sprites auf 10 bis 26 Pixel:

```
-- Lingo syntax
sprite(10).height = 26

// JavaScript syntax
sprite(10).height = 26;
```

heightVertices

Syntax

```
member(whichCastmember).modelResource(whichModelResource).\heightVertices
```

Beschreibung

Diese 3D-Eigenschaft für Modellressourcen vom Typ `#box` gibt die Anzahl von Gitternetzscheitelpunkten entlang der Boxhöhe an. Wenn Sie einen höheren Wert angeben, erhöht sich die Anzahl von Flächen und es ergibt sich ein feineres Gitternetz.

Die Höhe einer Box wird entlang ihrer Y-Achse gemessen.

Setzen Sie die Eigenschaft `renderStyle` des Shaders eines Modells auf `#wire`, um die Flächen des Gitternetzes der Modellressource zu sehen. Setzen Sie die Eigenschaft `renderStyle` auf `#point`, um nur die Scheitelpunkte des Gitternetzes zu sehen.

Der Wert dieser Eigenschaft muss größer oder gleich 2 sein. Der Standardwert ist 4.

Beispiel

Die folgende Anweisung setzt die Eigenschaft `heightVertices` der Modellressource "Turm" auf 10. Die Geometrie der Modellressource wird entlang der Z-Achse durch neun Polygone definiert; insgesamt gibt es also zehn Scheitelpunkte.

```
member("3D World").modelResource("Tower").heightVertices = 10
```

Siehe auch

`height (3D)`

highlightPercentage

Syntax

```
member(whichCastmember).model(whichModel).toon.highlightPercentage
member(whichCastmember).model(whichModel).shader.highlightPercentage
member(whichCastmember).shader(whichShader).highlightPercentage
```

Beschreibung

Diese 3D-Eigenschaft für den Modifizierer `toon` und den Shader `#painter` gibt an, welcher Prozentsatz der verfügbaren Farben in dem Bereich der Modelloberfläche verwendet wird, in dem das Licht Highlights erzeugt.

Der Gültigkeitsbereich dieser Eigenschaft liegt zwischen 50 und 100; die Standardeinstellung ist 0.

Die vom Modifizierer `toon` und dem Shader `#painter` für ein Modell verwendete Anzahl von Farben richtet sich nach der Eigenschaft `colorSteps`des Modifizierers `toon` bzw. des Shaders `#painter` des Modells.

Beispiel

Das folgende Beispiel legt die Eigenschaft `highlightPercentage` des im Modell "Kugel" enthaltenen Modifizierers toon fest.

```
-- Lingo syntax
member("3Dobjects").model("Sphere01").toon.highlightPercentage = 25

// Javascript
member("3Dobjects").getPropRef("model",2).toon.highlightPercentage = 25;
```

Siehe auch

highlightStrength, brightness

highlightStrength

Syntax

```
member(whichCastmember).model(whichModel).toon.highlightStrength
member(whichCastmember).model(whichModel).shader.highlightStrength
member(whichCastmember).shader(whichShader).highlightStrength
```

Beschreibung

Diese 3D-Eigenschaft für den Modifizierer `toon` und den Shader `#painter` gibt die Helligkeit des Bereichs der Modelloberfläche an, in dem das Licht Highlights erzeugt.

Der Standardwert dieser Eigenschaft lautet 1.0.

Beispiel

Das folgende Beispiel legt die Eigenschaft `highlightStrength` des im Modell "Kugel" enthaltenen Modifizierers toon fest.

```
-- Lingo syntax
member("3Dobjects").model("Sphere01").toon.highlightStrength = 0.25

// Javascript
member("3Dobjects").getPropRef("model",2).toon.highlightStrength = 0.25;
```

Siehe auch

highlightPercentage, brightness

hilite

Syntax

```
-- Lingo syntax
memberObjRef.hilite

// JavaScript syntax
memberObjRef.hilite;
```

Beschreibung

Diese Darstellereigenschaft bestimmt, ob ein mit dem Schaltflächenwerkzeug erstelltes Kontrollkästchen oder Optionsfeld ausgewählt ist (`TRUE`) oder nicht (`FALSE`, Standard). Lesen/Schreiben.

Beispiel

Die folgende Anweisung prüft, ob die Schaltfläche "Sound ein“ ausgewählt ist, und dreht die Lautstärke von Soundkanal 1 voll auf, wenn dies der Fall ist:

```
-- Lingo syntax
if (member("Sound On").hilite = TRUE) then
    sound(1).volume = 255
end if

// JavaScript syntax
if (member("Sound On").hilite == true) {
    sound(1).volume = 255;
}
```

Die folgende Anweisung aktiviert den Schaltflächendarsteller `powerSwitch`, indem die Darstellereigenschaft „hilite“ für den Darsteller auf TRUE festgelegt wird:

```
-- Lingo syntax
member("powerSwitch").hilite = TRUE

// JavaScript syntax
member("powerSwitch").hilite = true;
```

Siehe auch

`Member`

hinting

Syntax

```
-- Lingo syntax
member("text").hinting

// JavaScript syntax
member("text").hinting;
```

Beschreibung

Textdarstellereigenschaft, aktiviert die Optionen für Schriftarthinweise Mögliche Werte:

- `#Auto` – Verwendet die Informationen der Schriftartdatei für Schriftarthinweise. Dies ist der Standardwert.
- `#Algorithmic` – Verwendet für Schriftarthinweise das Textmodul.
- `#TVMode` – Zeichen werden horizontal und vertikal symmetrisch dargestellt und für Schriftarthinweise wird das Textmodul verwendet.
- `#None` – Für den aktuellen Darsteller keine Schriftarthinweise erstellt werden.

Beispiel

```
-- Lingo syntax
member("text").hinting = #Auto

// JavaScript syntax
member("text").hinting = symbol("Auto");
```

hither

Syntax

```
member(whichCastmember).camera(whichCamera).hither
sprite(whichSprite).camera{(index)}.hither
```

Beschreibung

Diese 3D-Kameraeigenschaft gibt an, ab welcher Entfernung von der Kamera (in Welteinheiten) Modelle gezeichnet werden. Objekte, die sich näher an der Kamera befinden als `hither`, werden nicht gezeichnet.

Der Wert dieser Eigenschaft muss größer oder gleich 1.0 sein. Der Standardwert ist 5.0.

Beispiel

Die folgende Anweisung setzt die Eigenschaft hither von Kamera 1 auf 1000. Modelle, die weniger als 1000 Welteinheiten von der Kamera entfernt sind, sind nicht sichtbar.

```
member("SolarSystem").camera[1].hither = 1000
```

Siehe auch

yon

hotSpot

Syntax

```
-- Lingo syntax
memberObjRef.hotSpot

// JavaScript syntax
memberObjRef.hotSpot;
```

Beschreibung

Diese Cursor-Darstellereigenschaft gibt die horizontale und vertikale Punktposition der Pixel an, die den Hotspot im animierten Farbcursor-Darsteller `whichCursorCastMember` definieren. Director verfolgt mit diesem Punkt die Position des Cursors auf dem Bildschirm (wenn das Programm z. B. die Werte für die Lingo-Funktionen `mouseH` und `mouseV` und zurückgibt) und stellt fest, wo ein Rollover (das durch die Lingo-Nachricht `mouseEnter` signalisiert wird) auftritt.

Die linke obere Ecke eines Cursors ist „point(0,0)“, der Standardwert von `hotSpot`. Beim Versuch, einen Punkt außerhalb der Cursorbegrenzung einzustellen, wird ein Fehler erzeugt. So wird beispielsweise ein Fehler verursacht, wenn der Hotspot eines 16 x 16 Pixel großen Cursors auf point(16,16) eingestellt wird (da der Ausgangspunkt 0,0 und nicht 1,1 ist).

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Die folgende Prozedur stellt den Hotspot eines 32 x 32 Pixel großen Cursors (dessen Darstellernummer in der Variablen `cursorNum` gespeichert ist) auf die Cursormitte ein:

```
-- Lingo syntax
on startMovie
    member(cursorNum).hotSpot = point(16,16)
end

// JavaScript syntax
function startMovie() {
    member(cursorNum).hotSpot = point(16,16);
}
```

hotSpotEnterCallback

Syntax

```
-- Lingo syntax
spriteObjRef.hotSpotEnterCallback

// JavaScript syntax
spriteObjRef.hotSpotEnterCallback;
```

Beschreibung

Diese QuickTime VR Sprite-Eigenschaft enthält den Namen der Prozedur, die ausgeführt wird, wenn der Cursor auf einen auf der Bühne sichtbaren QuickTime VR-Hotspot bewegt wird. Das QuickTime VR-Sprite erhält die Nachricht zuerst. Diese Nachrichte hat zwei Argumente: den Parameter`me` und die ID des Hotspots, in den der Cursor eingetreten ist.

Wenn Sie den Rückruf löschen möchten, setzen Sie diese Eigenschaft auf 0.

Zur Vermeidung von Leistungsabfällen sollten Sie Rückrufeigenschaften nur einstellen, wenn dies wirklich notwendig ist.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Das folgende Beispiel zeigt den Namen der Prozedur an, die ausgeführt wird, wenn der Cursor auf einen auf der Bühne sichtbaren QuickTime VR-Hotspot bewegt wird.

```
-- Lingo syntax
put sprite("multinode").hotSpotEnterCallback

// Javascript
put(sprite("multinode").hotSpotEnterCallback);
```

Siehe auch

`hotSpotExitCallback`, `nodeEnterCallback`, `nodeExitCallback`, `triggerCallback`

hotSpotExitCallback

Syntax

```
-- Lingo syntax
spriteObjRef.hotSpotExitCallback

// JavaScript syntax
spriteObjRef.hotSpotExitCallback;
```

Beschreibung

Diese QuickTime VR Sprite-Eigenschaft enthält den Namen der Prozedur, die ausgeführt wird, wenn der Cursor von einem auf der Bühne sichtbaren QuickTime VR-Hotspot fort bewegt wird. Das QuickTime VR-Sprite erhält die Nachricht zuerst. Diese Nachrichte hat zwei Argumente: den Parameter`me` und die ID des Hotspots, in den der Cursor eingetreten ist.

Wenn Sie den Rückruf löschen möchten, setzen Sie diese Eigenschaft auf 0.

Zur Vermeidung von Leistungsabfällen sollten Sie Rückrufeigenschaften nur einstellen, wenn dies wirklich notwendig ist.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Das folgende Beispiel zeigt den Namen der Prozedur an, die ausgeführt wird, wenn der Cursor von einem auf der Bühne sichtbaren QuickTime VR-Hotspot wegbewegt wird.

```
-- Lingo syntax
put sprite("multinode").hotSpotExitCallback

// Javascript
put(sprite("multinode").hotSpotExitCallback);
```

Siehe auch

`hotSpotEnterCallback`, `nodeEnterCallback`, `nodeExitCallback`, `triggerCallback`

HTML

Syntax

```
-- Lingo syntax
memberObjRef.HTML

// JavaScript syntax
memberObjRef.HTML;
```

Beschreibung

Diese Darstellereigenschaft greift auf Text und Tags zu, die das Textlayout in einem HTML-formatierten Textdarsteller steuern.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Die folgende Anweisung zeigt im Nachrichtenfenster die HTML-Formatierungsinformationen an, die im Textdarsteller „Homepage“ eingebettet sind:

```
--Lingo syntax
put(member("Home Page").HTML)

// JavaScript syntax
trace(member("Home Page").HTML);
```

Siehe auch

importFileInto(), RTF

hyperlink

Syntax

```
-- Lingo syntax
chunkExpression.hyperlink

// JavaScript syntax
chunkExpression.hyperlink;
```

Beschreibung

Diese Textdarstellereigenschaft gibt den Hyperlinkstring für den im Textdarsteller angegebenen Chunk-Ausdruck zurück.

Diese Eigenschaft kann getestet und eingestellt werden.

Beim Abrufen dieser Eigenschaft wird der Hyperlink verwendet, der das erste Zeichen von *chunkExpression* enthält.

Hyperlinks dürfen sich nicht überschneiden. Wird ein Hyperlink über einen vorhandenen Link gesetzt, wenn auch nur teilweise, wird der ursprüngliche Link durch den neuen ersetzt.

Wird ein Hyperlink auf einen leeren String gesetzt, wird er entfernt.

Beispiel

Die folgende Prozedur erstellt einen Hyperlink im ersten Wort des Textdarstellers `MacroLink`. Dieser Link verweist auf die Adobe-Website.

```
--Lingo syntax
on startMovie
    member("MacroLink").word[1].hyperlink = "http://www.adobe.com"
end

// JavaScript syntax
function startMovie() {
    member("MacroLink").getPropRef("word", 1).hyperlink = "http://www.adobe.com";
}
```

Siehe auch

hyperlinkRange, hyperlinkState

hyperlinkRange

Syntax

```
-- Lingo syntax
chunkExpression.hyperlinkRange

// JavaScript syntax
chunkExpression.hyperlinkRange;
```

Beschreibung

Diese Textdarstellereigenschaft gibt den Bereich des Hyperlinks zurück, der das erste Zeichen des Chunk-Ausdrucks enthält.

Diese Eigenschaft kann getestet, aber nicht eingestellt werden.

Wie auch bei `hyperLink` und `hyperLinkState` enthält der zurückgegebene Bereich des Links das erste Zeichen von *chunkExpression*.

Beispiel

Das folgende Beispiel zeigt den Bereich des Hyperlinks an, der das erste Zeichen des Chunk-Ausdrucks enthält.

```
-- Lingo syntax
put member("MyText").hyperlinkRange

// Javascript
put(member("MyText").hyperlinkRange);
```

Siehe auch

`hyperlink`, `hyperlinkState`

hyperlinks

Syntax

```
-- Lingo syntax
chunkExpression.hyperlinks

// JavaScript syntax
chunkExpression.hyperlinks;
```

Beschreibung

Diese Textdarstellereigenschaft gibt eine lineare Liste zurück, die alle Hyperlink-Bereiche des angegebenen Chunks eines Textdarstellers enthält. Jeder Bereich wird als lineare Liste mit zwei Elementen angezeigt – einem für das Anfangszeichen des Links und einem für das Endzeichen.

Beispiel

Die folgende Anweisung gibt alle Links für den Textdarsteller „Glossar“ zurück und zeigt sie im Nachrichtenfenster an:

```
--Lingo syntax
put(member("Glossary").hyperlinks) -- [[3, 8], [10, 16], [41, 54]]

// JavaScript syntax
trace(member("Glossary").hyperlinks); // [[3, 8], [10, 16], [41, 54]]
```

hyperlinkState

Syntax

```
-- Lingo syntax
chuckExpression.hyperlinkState

// JavaScript syntax
chuckExpression.hyperlinkState;
```

Beschreibung

Diese Textdarstellereigenschaft enthält den aktuellen Zustand eines Hyperlinks. Mögliche Werte für den Status sind: `#normal`, `#active` und `#visited`.

Diese Eigenschaft kann getestet und eingestellt werden.

Wie auch bei `hyperLink` und `hyperlinkRange` enthält der zurückgegebene Bereich des Links das erste Zeichen von *chunkExpression*.

Beispiel

Die folgende Prozedur prüft, ob es sich bei dem angeklickten Hyperlink um eine Webadresse handelt. Wenn ja, wird der Zustand des Hyperlinktexts auf `#visited` gesetzt, und der Film verzweigt zu der betreffenden Webadresse.

```
--Lingo syntax
property spriteNum

on hyperlinkClicked me, data, range
    if data starts "http://" then
        currentMember = sprite(spriteNum).member
        currentMember.word[4].hyperlinkState = #visited
        gotoNetPage(data)
    end if
end

// JavaScript syntax
function hyperlinkClicked(data, range) {
    var st = data.slice(0,7);
    var ht = "http://";
    if (st == ht) {
        currentMember = sprite(spriteNum).member;
        currentMember.getPropRef("word", 4).hyperlinkState = symbol("visited");
        gotoNetPage(data);
    }
}
```

Siehe auch

hyperlink, hyperlinkRange

idleHandlerPeriod

Syntax

```
-- Lingo syntax
_movie.idleHandlerPeriod

// JavaScript syntax
_movie.idleHandlerPeriod;
```

Beschreibung

Diese Filmeigenschaft bestimmt, wie viel Zeit (in Ticks) maximal verstreichen darf, bis der Film eine `idle`-Nachricht sendet. Lesen/Schreiben.

Der Standardwert ist 1 und weist den Film an, eine `idle`-Prozedurnachricht maximal 60-mal pro Sekunde zu senden.

Wenn der Abspielkopf in ein Bild eintritt, startet Director einen Timer, zeichnet die geeigneten Sprites auf der Bühne neu und gibt ein `enterFrame`-Ereignis aus. Wenn die für das Tempo eingestellte Zeit bereits abgelaufen ist, erzeugt Director ein `exitFrame`-Ereignis und geht zum nächsten angegebenen Bild über. Andernfalls erzeugt Director in regelmäßigen Abständen eine `idle`-Nachricht, bis die Zeit abgelaufen ist. Das Intervall zwischen den einzelnen `idle`-Ereignissen wird durch den Wert für `idleHandlerPeriod` festgelegt.

Mögliche Einstellungen für `idleHandlerPeriod`:

- 0 – So viele idle-Ereignisse wie möglich
- 1 – Maximal 60 pro Sekunde
- 2 – Maximal 30 pro Sekunde
- 3 – Maximal 20 pro Sekunde
- *n* – Maximal 60/*n* pro Sekunde

Die Zahl der `idle`-Ereignisse pro Bild hängt auch von der Bildrate des Films und anderen Aktivitäten ab (z. B. davon, ob Skripts ausgeführt werden). Wenn das Tempo 60 Bilder pro Sekunden (BpS) beträgt und der Wert von `idleHandlerPeriod` 1 ist, findet genau ein `idle`-Ereignis pro Bild statt. Wenn das Tempo 20 BpS beträgt, finden drei `idle`-Ereignisse pro Bild statt. Es kommt zu einem Wartezustand, wenn Director gegenwärtig keine Aufgaben durchzuführen hat und keine Ereignisse generieren kann.

Wird dagegen die Eigenschaft `idleHandlerPeriod` bei ausgesprochen niedrigem Tempo auf 0 gesetzt, können Tausende von `idle`-Ereignisse generiert werden.

Der Standardwert dieser Eigenschaft lautet 1.

Beispiel

Die folgende Anweisung bewirkt, dass der Film maximal einmal pro Sekunde eine idle-Nachricht sendet:

```
-- Lingo syntax
_movie.idleHandlerPeriod = 60

// JavaScript syntax
_movie.idleHandlerPeriod = 60;
```

Siehe auch

`on idle`, `idleLoadMode`, `idleLoadPeriod`, `idleLoadTag`, `idleReadChunkSize`, `Movie`

idleLoadMode

Syntax

```
-- Lingo syntax
_movie.idleLoadMode

// JavaScript syntax
_movie.idleLoadMode;
```

Beschreibung

Diese Filmeigenschaft bestimmt, wann die Methoden `preLoad()` und `preLoadMember()` während Wartezeiten versuchen, Darsteller zu laden. Lesen/Schreiben.

Wartezeiten können einen der folgenden Werte aufweisen:

- 0 – Im Wartezustand werden keine Darsteller geladen.
- 1 – Darsteller werden im Wartezustand geladen, wenn freie Zeit zwischen Bildern verfügbar ist.
- 2 – Darsteller werden während `idle` -Ereignissen im Wartezustand geladen.
- 3 – Darsteller werden im Wartezustand so oft wie möglich geladen.

Die Eigenschaft `idleLoadMode` führt keine Funktion aus und kann nur in Verbindung mit den Methoden `preLoad()` und `preLoadMember()` verwendet werden.

Darsteller, die während des Wartezustands geladen werden, bleiben so lange komprimiert, bis sie vom Film verwendet werden. Deshalb kann es beim Abspielen des Films zu beträchtlichen Pausen kommen, wenn die Darsteller dekomprimiert werden.

Beispiel

Die folgende Anweisung bewirkt, dass der Film Darsteller, die durch die Befehle `preLoad` und `preLoadMember` zum Vorausladen bestimmt sind, so oft wie möglich zu laden versucht:

```
-- Lingo syntax
_movie.idleLoadMode = 3

// JavaScript syntax
_movie.idleLoadMode = 3;
```

Siehe auch

`on idle`, `Movie`, `preLoad() (Film)`, `preLoadMember()`

idleLoadPeriod

Syntax

```
-- Lingo syntax
_movie.idleLoadPeriod

// JavaScript syntax
_movie.idleLoadPeriod;
```

Beschreibung

Diese Filmeigenschaft bestimmt, wie viele Ticks Director warten soll, bevor das Programm versucht, Darsteller zu laden, die darauf warten, geladen zu werden. Lesen/Schreiben.

Der Standardwert für `idleLoadPeriod` ist 0 und weist Director an, die Ladewarteschlange so oft wie möglich abzufertigen.

Beispiel

Die folgende Anweisung weist Director an, in Abständen von 0,5 Sekunden (30 Ticks) zu versuchen, alle wartenden Darsteller zu laden:

```
-- Lingo syntax
_movie.idleLoadPeriod = 30

// JavaScript syntax
_movie.idleLoadPeriod = 30;
```

Siehe auch

`on idle`, `Movie`

idleLoadTag

Syntax

```
-- Lingo syntax
_movie.idleLoadTag

// JavaScript syntax
_movie.idleLoadTag;
```

Beschreibung

Diese Filmeigenschaft identifiziert oder markiert Darsteller, die im Ruhezustand des Computers auf das Vorausladen warten, mit einer Nummer. Lesen/Schreiben.

Die Eigenschaft `idleLoadTag` erleichtert das Identifizieren von Darstellern in einer Gruppe, die vorausgeladen werdensollen. Sie kann einen beliebigen Wert annehmen.

Beispiel

Die folgende Anweisung macht die Zahl 10 zum Idle Load Tag:

```
-- Lingo syntax
_movie.idleLoadTag = 10

// JavaScript syntax
_movie.idleLoadTag = 10;
```

Siehe auch

`on idle`, `Movie`

idleReadChunkSize

Syntax

```
-- Lingo syntax
_movie.idleReadChunkSize

// JavaScript syntax
_movie.idleReadChunkSize;
```

Beschreibung

Diese Filmeigenschaft bestimmt, wie viel Bytes Director maximal laden kann, wenn das Programm versucht, Darsteller aus einer Ladewarteschlange zu laden. Lesen/Schreiben.

Der Standardwert für `idleReadChunkSize` ist 32 KB.

Beispiel

Die folgende Anweisung gibt 500 KB als maximale Speichermenge an, die Director beim Versuch, Darsteller aus einer Ladewarteschlange zu laden, auf einmal einlesen kann:

```
-- Lingo syntax
_movie.idleReadChunkSize = (500 * 1024)

// JavaScript syntax
_movie.idleReadChunkSize = (500 * 1024);
```

Siehe auch

`on idle`, `Movie`

image (Grafik)

Syntax

```
-- Lingo syntax
imageObjRef.image

// JavaScript syntax
imageObjRef.image;
```

Beschreibung

Diese Grafikeigenschaft bezieht sich auf das Grafikobjekt eines Bitmap- oder Textdarstellers, der Bühne oder eines Fensters. Lesen/Schreiben für die Grafik eines Darstellers. Nur Lesen für eine Grafik der Bühne oder eines Fensters.

Wenn Sie die Eigenschaft `image` eines Darstellers einstellen, wirkt sich dies unmittelbar auf dessen Inhalt aus. Wenn Sie jedoch die Grafik eines Darstellers oder Fensters abrufen, erzeugt Director einen Verweis auf diese Grafik. Änderungen an den Fenstern wirken sich unmittelbar auf den Darsteller oder das Fenster aus.

Wenn Sie umfangreiche Änderungen an der Eigenschaft `image` eines Elements vornehmen möchten, sollten Sie aus Zeitgründen die Eigenschaft „image“ des Elements mit der `duplicate()`-Methode in ein neues Grafikobjekt kopieren, die Änderungen an dem neuen Grafikobjekt vornehmen und die Grafik des ursprünglichen Elements anschließend auf das neue Grafikobjekt setzen. Nicht-Bitmapdarsteller lassen sich mit der `duplicate()`-Methode grundsätzlich schneller bearbeiten.

Beispiel

Die folgende Anweisung stellt die Grafik des Darstellers `originalFlower` in den Darsteller `newFlower`:

```
-- Lingo syntax
member("newFlower").image = member("originalFlower").image

// JavaScript syntax
member("newFlower").image = member("originalFlower").image;
```

Die folgenden Anweisungen schreiben einen Bezug auf die Grafik der Bühne in die Variable `myImage` und stellen diese Grafik anschließend in den Darsteller „Blume“.

```
-- Lingo syntax
myImage = _movie.stage.image
member("flower").image = myImage

// JavaScript syntax
var myImage = _movie.stage.image;
member("flower").image = myImage;
```

Siehe auch

copyPixels(), draw(), duplicate() (Grafik), fill(), image(), setPixel()

image (RealMedia)

Syntax

```
-- Lingo syntax
memberOrSpriteObjRef.image

// JavaScript syntax
memberOrSpriteObjRef.image;
```

Beschreibung

Diese Eigenschaft für `RealMedia`-Sprites oder -Darsteller gibt ein Lingo-Grafikobjekt mit dem aktuellen Bild des `RealMedia`-Videostreams zurück. Anhand dieser Eigenschaft können Sie `RealVideo®` einem 3D-Modell zuordnen (siehe Beispiel unten).

Beispiel

Die folgende Anweisung kopiert das aktuelle Bild des `RealMedia`-Darstellers "Real“ in den Bitmapdarsteller "Standbild“:

```
-- Lingo syntax
member("Still").image = member("Real").image

// JavaScript syntax
member("Still").image = member("Real").image;
```

image (MP4Media/FLV)

Syntax

```
img = sprite(1).image
img = member(1).image
```

Beschreibung

Diese MP4Media-Darsteller- oder Spriteeigenschaft gibt ein Lingo-Bildobjekt für das aktuelle Bild des MP4Media/FLV-Darstellers oder -Sprites zurück.

Beispiele

Folgende Beispiele geben das Lingo-Bildobjekt für das aktuelle Bild `currentFrame` des MP4-Videos zurück.

```
-- Lingo syntax
img = sprite(1).image
img = member(1).image

// JavaScript syntax
img = sprite(1).image;
img = member(1).image;
```

image (Fenster)

Syntax

```
-- Lingo syntax
windowObjRef.image

// JavaScript syntax
windowObjRef.image;
```

Beschreibung

Diese Fenstereigenschaft verweist auf das Grafikobjekt eines Fensters. Nur Lesen.

Wenn Sie die Grafik eines Fensters abrufen, erzeugt Director einen Verweis auf die Grafik des angegebenen Fensters. Änderungen an der Grafik wirken sich unmittelbar auf den Inhalt des Fensters aus.

Wenn Sie umfangreiche Änderungen an der Eigenschaft `image` vornehmen möchten, sollten Sie aus Zeitgründen die Eigenschaft `image` mit der `duplicate()`-Methode des Member-Objekts in ein neues Grafikobjekt kopieren, die Änderungen an dem neuen Grafikobjekt vornehmen und die Grafik des ursprünglichen Elements anschließend auf das neue Grafikobjekt setzen. Nicht-Bitmapdarsteller lassen sich mit der `duplicate()`-Methode grundsätzlich schneller bearbeiten.

Beispiel

Die folgenden Anweisungen schreiben einen Verweis auf die Grafik der Bühne in die Variable `myImage` und stellen diese Grafik anschließend in das Fenster "Blume":

```
-- Lingo syntax
myImage = _movie.stage.image
window("Flower").image = myImage

// JavaScript syntax
var myImage = _movie.stage.image;
window("Flower").image = myImage;
```

Siehe auch

duplicate() (Darsteller), Window

imageCompression

Syntax

```
-- Lingo syntax
_movie.imageCompression
memberObjRef.imageCompression

// JavaScript syntax
_movie.imageCompression;
memberObjRef.imageCompression;
```

Beschreibung

Diese Film- und Bitmapdarstellereigenschaft gibt an, welche Art von Komprimierung Director für interne (nicht verknüpfte) Bitmapdarsteller verwendet, wenn ein Film im Shockwave Player-Format gespeichert wird. Lesen/Schreiben.

`imageCompression` kann folgende gültige Werte annehmen:

Wert	Bedeutung
`#standard`	Verwendet das interne Director-Standardkomprimierungsformat.
`#movieSetting`	Verwendet die in der Eigenschaft `_movie.imageCompression` gespeicherten Komprimierungseinstellungen des Films. Dies entspricht bei Grafikformaten, die nicht auf die Standardkomprimierung beschränkt sind, der Standardeinstellung.
`#jpeg`	Verwendet JPEG-Komprimierung Siehe `imageQuality`.

Diese Eigenschaft wird normalerweise innerhalb von Director im Dialogfeld „Veröffentlichungseinstellungen“ festgelegt.

Beispiel

Die folgende Anweisung zeigt im Nachrichtenfenster die `imageCompression` an, die für den aktuell wiedergegebenen Film gilt:

```
-- Lingo syntax
put(_movie.imageCompression)

// JavaScript syntax
put(_movie.imageCompression);
```

Siehe auch

imageQuality, Movie

imageEnabled

Syntax

```
-- Lingo syntax
memberOrSpriteObjRef.imageEnabled

// JavaScript syntax
memberOrSpriteObjRef.imageEnabled;
```

Beschreibung

Diese Darsteller- und Sprite-Eigenschaft bestimmt, ob ein Flash-Film oder die Grafiken einer Vektorform zu sehen sind (TRUE, Standard) oder nicht (FALSE).

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Das folgende beginSprite-Skript richtet ein verknüpftes Flash-Film-Sprite so ein, dass dessen Grafiken ausgeblendet werden, wenn es zum ersten Mal auf der Bühne erscheint und in den Speicher geladen wird, und speichert seine Sprite-Nummer in einer globalen Variable namens gStreamingSprite, damit diese später in einem Bildskript im Drehbuch verwendet werden kann:

```
-- Lingo syntax
global gStreamingSprite

on beginSprite me
    gStreamingSprite = me.spriteNum
    sprite(gStreamingSprite).imageEnabled = FALSE
end

// JavaScript syntax
function beginSprite() {
    _global.gStreamingSprite = this.spriteNum;
    sprite(_global.gStreamingSprite).imageEnabled = 0;
}
```

In einem späteren Bild des Films prüft dieses Bildskript, ob das in der globalen Variablen gStreamingSprite angegebene Flash-Film-Sprite vollständig in den Speicher geladen wurde. Wenn nicht, lässt das Skript den Abspielkopf im aktuellen Bild so lange in Schleife abspielen, bis der Film vollständig in den Speicher geladen ist. Anschließend wird die Eigenschaft imageEnabled auf TRUE gesetzt, damit die Grafiken eingeblendet werden, und der Abspielkopf zum nächsten Bild im Drehbuch geschickt.

```
-- Lingo syntax
global gStreamingSprite

on exitFrame me
    if sprite(gStreamingSprite).member.percentStreamed < 100 then
        _movie.go(_movie.frame)
    else
        sprite(gStreamingSprite).imageEnabled = TRUE
        _movie.updatestage()
    end if
end

// JavaScript syntax
function exitFrame() {
    var stmSp = sprite(_global.gStreamingSprite).member.percentStreamed;
    if (stmSp < 100) {
        _movie.go(_movie.frame);
    } else {
        sprite(_global.gStreamingSprite).imageEnabled = 1;
        _movie.updatestage();
    }
}
```

imageQuality

Syntax

```
-- Lingo syntax
_movie.imageQuality
memberObjRef.imageQuality

// JavaScript syntax
_movie.imageQuality;
memberObjRef.imageQuality;
```

Beschreibung

Diese Film- und Bitmapdarstellereigenschaft gibt den Komprimierungsfaktor an, der verwendet werden soll, wenn die Eigenschaft `imageCompression` eines Films auf `#jpeg` gesetzt ist. Lesen/Schreiben; nur während der Erstellung.

Gültiger Bereich: 0 bis 100. Der Wert 0 ergibt die niedrigste Bildqualität und die höchste Komprimierung, der Wert 100 die höchste Bildqualität und die niedrigste Komprimierung.

Diese Eigenschaft kann nur bei der Filmerstellung eingestellt werden und wirkt sich erst dann aus, wenn der Film im Shockwave-Format gespeichert wird.

Beispiel

Die folgende Anweisung zeigt im Nachrichtenfenster die `imageQuality` an, die für den aktuell wiedergegebenen Film gilt:

```
-- Lingo syntax
put(_movie.imageQuality)

// JavaScript syntax
put(_movie.imageQuality);
```

Siehe auch

imageCompression, Movie

immovable

Syntax

```
member(whichCastmember).model(whichModel).collision.immovable
```

Beschreibung

Diese 3D-Eigenschaft für den Modifizierer `#collision` gibt an, ob ein Modell aufgrund von während der Animation stattfindenden Kollisionen verschoben werden kann. Bei Angabe von `TRUE` wird das Modell stationär; bei Angabe von `FALSE` kann es verschoben werden. Mit dieser Eigenschaft lässt sich bei der Animation die Leistung steigern, da stationäre Modelle auch nicht mit Lingo auf Kollisionen hin überprüft werden müssen.

Der Standardwert dieser Eigenschaft lautet `FALSE`.

Beispiel

Die folgende Anweisung setzt die Eigenschaft immovable des am zweiten Modell des Darstellers "3Dobjects" angebrachten Modifizierers collision auf TRUE:

```
-- Lingo syntax
member("3Dobjects").model[2].collision.immovable = TRUE

// Javascript
member("3Dobjects").getPropRef("model",2).collision.immovable = 1;
```

Siehe auch

collision (modifier)

ink

Syntax

```
-- Lingo syntax
spriteObjRef.ink

// JavaScript syntax
spriteObjRef.ink;
```

Beschreibung

Diese Sprite-Eigenschaft bestimmt den Farbeffekt, der auf ein Sprite angewendet wird. Lesen/Schreiben.

`ink` kann folgende gültige Werte annehmen:

0Copy—	32Blend—
1Transparent—	33Add pin—
2Reverse—	34Add—

3Ghost—	35Subtract pin—
4Not copy—	36Background transparent —
5Not transparent—	37Lightest—
6Not reverse—	38Subtract—
7Not ghost—	39Darkest—
8Matte—	40Lighten—
9Mask—	41Darken—

Bei 36 (Hintergrund transparent) können Sie ein Sprite im Drehbuch auswählen und dann eine transparente Farbe im Feld „Hintergrundfarbe“ des Werkzeugfensters auswählen. Sie können dazu auch die Eigenschaft `backColor` einstellen.

Wenn Sie diese Eigenschaft in einem Skript einstellen, während der Abspielkopf stillsteht, müssen Sie unbedingt die `updateStage()`-Methode des Movie-Objekts verwenden, damit die Bühne neu gezeichnet wird. Wenn Sie mehrere Sprite-Eigenschaften – oder mehrere Sprites – ändern, sollten Sie nur eine einzige `updateStage()` -Methode verwenden, und zwar nachdem Sie alle Änderungen vorgenommen haben.

Beispiel

Die folgende Anweisung setzt die Variable `currentInk` auf den Wert des Farbeffekts von Sprite (3):

```
-- Lingo syntax
currentInk = sprite(3).ink

// JavaScript syntax
var currentInk = sprite(3).ink;
```

Die folgende Anweisung verleiht Sprite (i + 1) den Farbeffekt „Matt“. Hierzu wird die Sprite-Eigenschaft ink auf 8 gesetzt:

```
-- Lingo syntax
sprite(i + 1).ink = 8

// JavaScript syntax
sprite(i + 1).ink = 8;
```

Siehe auch

`backColor`, `Sprite`, `updateStage()`

inker (Modifizierer)

Syntax

```
member(whichCastmember).modelResource(whichModelResource).inker.inkerModifierProperty
modelResourceObjectReference.inker.inkerModifierProperty
```

Beschreibung

3D-Modifizierer; nachdem Sie den Modifizierer `#inker` mit `addModifier` zu einer Modellressource hinzugefügt haben, können Sie die `#inker`-Modifiziereigenschaften abrufen und einstellen.

Der Modifizierer `#inker` fügt Silhouetten, Falten und Begrenzungskanten zu einem vorhandenen Modell hinzu. Mit den `#inker`-Eigenschaften können Sie die Definition und Stärke dieser Merkmale steuern.

Bei Verwendung des Modifizierers `#inker` in Verbindung mit dem Modifizierer `#toon` ergibt sich beim Rendering ein kumulativer Effekt, der je nachdem, welcher Modifizierer zuerst angewandt wurde, variiert. Die von der Eigenschaft `modifier` zurückgegebene Modifiziererliste enthält entweder `#inker` oder `#toon` (je nachdem, welcher Modifizierer zuerst hinzugefügt wurde), aber nicht beide. Der Modifizierer `#inker` kann nicht zusammen mit dem Modifizierer `#sds` verwendet werden.

Der Modifizierer `#inker` weist folgende Eigenschaften auf:

- Mit `lineColor` können Sie die Farbe der von diesem Modifizierer gezeichneten Linien abrufen und einstellen.
- Mit `silhouettes` können Sie ermitteln und festlegen, ob zum Definieren der Kanten entlang der Modellgrenze Linien gezeichnet werden, durch die die Form des Modells hervorgehoben wird.
- Mit `creases` können Sie abrufen und einstellen, ob in Falten Linien gezeichnet werden.
- Mit `creaseAngle` können Sie die Empfindlichkeit der Kantenwinkelerkennung für diesen Modifizierer ermitteln und festlegen.
- Mit `boundary` können Sie ermitteln und festlegen, ob um die Grenzen der Oberfläche herum Linien gezeichnet werden.
- Mit `lineOffset` können Sie abrufen und einstellen, wo relativ zur schattierten Oberfläche und der Kamera Linien gezeichnet werden.
- Mit `useLineOffset` können Sie ermitteln und festlegen, ob `lineOffset` aktiviert ist oder nicht.

Hinweis: *Weitere Informationen zu den folgenden Eigenschaften finden Sie in den einzelnen Beschreibungen.*

Beispiel

Die folgende Anweisung fügt dem zweiten Modell des Darstellers `3Dobjects` den Modifizierer inker hinzu: Nachdem Sie den Modifizierer `#inker` mit `addModifier` zu einer Modellressource hinzugefügt haben, können Sie `#inker`-Modifiziereigenschaften abrufen und einstellen.

```
-- Lingo syntax
member("3Dobjects").model[2].addModifier(#inker)

// JavaScript syntax
member("3Dobjects").getPropRef("model",2).addModifier(symbol("inker"));
```

Siehe auch

`addModifier`, `modifiers`, `toon (Modifizierer)`, `shadowPercentage`

inlineImeEnabled

Syntax

```
-- Lingo syntax
_player.inlineImeEnabled

// JavaScript syntax
_player.inlineImeEnabled;
```

Beschreibung

In Director 11 ist die Eigenschaft `inlineImeEnabledalwaysTRUE`, unabhängig davon, ob sie mithilfe von Skriptcode auf „TRUE“ oder „FALSE“ festgelegt wird. Diese Eigenschaft wird nur aus Gründen der Rückwärtskompatibilität unterstützt.

Beispiel

Die folgende Anweisung setzt die Eigenschaft `inlineImeEnabled` des Players auf TRUE.

```
-- Lingo syntax
_player.inlineImeEnabled=TRUE

// JavaScript syntax
_player.inlineImeEnabled=1;
```

Siehe auch

`Player`

interval

Syntax

```
-- Lingo syntax
memberObjRef.interval

// JavaScript syntax
memberObjRef.interval;
```

Beschreibung

Diese Cursor-Darstellereigenschaft gibt das Intervall in Millisekunden (ms) zwischen den einzelnen Bildern des animierten Farbcursor-Darstellers *whichCursorCastMember* an. Das Standardintervall ist 100 ms.

Das Cursorintervall ist unabhängig von der Bildrate, die mit dem Tempokanal oder dem Lingo-Befehl `puppetTempo` für den Film eingestellt wurde.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

In diesem Sprite-Skript wird das Intervall auf 50 ms eingestellt, um die Animation zu beschleunigen, wenn der animierte Farbcursor, der im Darsteller „Schmetterling“ gespeichert ist, in das Sprite eintritt. Wenn der Cursor das Sprite verlässt, wird das Intervall auf 100 ms zurückgesetzt, um die Animation zu verlangsamen.

```
-- Lingo syntax
on mouseEnter
    member("Butterfly").interval = 50
end

on mouseLeave
    member("Butterfly").interval = 100
end

// JavaScript syntax
function mouseEnter() {
    member("Butterfly").interval = 50;
}

function mouseLeave() {
    member("Butterfly").interval = 100;
}
```

invertMask

Syntax

```
-- Lingo syntax
memberObjRef.invertMask

// JavaScript syntax
memberObjRef.invertMask;
```

Beschreibung

Diese QuickTime-Darstellereigenschaft bestimmt, ob Director QuickTime-Filme in den weißen Pixeln der Maske des Films (`TRUE`) oder in den schwarzen Pixeln (`FALSE`, Standard) zeichnet.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Die folgende Prozedur kehrt die aktuelle Einstellung der Eigenschaft `invertMask` eines QuickTime-Films mit dem Namen "Sonnenrad" um:

```
-- Lingo syntax
on toggleMask
    member("Starburst").invertMask = not(member("Starburst").invertMask)
end

// JavaScript syntax
function toggleMask() {
    member("Starburst").invertMask = !(member("Starburst").invertMask);
}
```

Siehe auch

mask

isPlayable (MP4Media/FLV)

Syntax

```
member("MP4Media").isPlayable = true
member("FLV").isPlayable = true
sprite("MP4Media").isPlayable = true
sprite("FLV").isPlayable = true
```

Beschreibung

Diese MP4Media/FLV-Darsteller- oder Spriteeigenschaft zeigt mit dem Boole'schen Rückgabewert an, ob das MP4/FLV-Element Audio- oder Videoinhalt enthält, der von Director unterstützt wird. Diese Eigenschaft ist schreibgeschützt.

Beispiele

Das folgende Beispiel zeigt, ob das MP4Media-Element oder das FLV-Video in „Sprite 1“ mit Director abgespielt werden kann.

```
-- Lingo syntax
put(sprite(1).isPlayable) -- 1
put(member(1).isPlayable) -- 1

// JavaScript syntax
put(sprite(1).isPlayable); // 1
put(member(1).isPlayable); // 1
```

isSaving (Mixer)

Syntax

```
mixer.isSaving(ReadOnly)
```

Beschreibung

Diese Eigenschaft des Audiomixers gibt 1 zurück, wenn die Mixerausgabe gerade gespeichert wird, sonst 0.

Hinweis: *Verwenden Sie diese Eigenschaft nicht mit den Methoden* `startSave()` *und* `stopSave()`.

Beispiele

```
-- Lingo syntax
on exitFrame me
    put mixerRef.issaving --Returns 0 if the mixer associated with mixerRef is not being
-- saved. Returns 1 otherwise.
end

// JavaScript syntax
function exitframe(){
put mixerRef.isSaving) ; //Returns 0 if the mixer associated with mixerRef is not being
// saved. Returns 1 otherwise.

}
```

Siehe auch

Mixer

isSaving (Soundobjekt)

Syntax

```
soundObj.isSaving(ReadOnly)
```

Beschreibung

Diese Soundobjekteigenschaft gibt 1 zurück, wenn das Soundobjekt gerade gespeichert wird, sonst 0.

Hinweis: *Verwenden Sie diese Eigenschaft nicht mit den Methoden* `startSave()` *und* `stopSave()`.

Beispiele

```
-- Lingo syntax
on exitFrame me
    put soundObjRef.issaving -- Returns 0 if the sound object associated
-- with soundobjectRef is not being saved. Returns 1 otherwise.
end

// JavaScript syntax
function exitframe(){
put (soundObjRef.isSaving) ; // Returns 0 if the sound object associated
// with soundobjectRef is not being saved. Returns 1 otherwise.

}
```

isVRMovie

Syntax

```
-- Lingo syntax
memberOrSpriteObjRef.isVRMovie

// JavaScript syntax
memberOrSpriteObjRef.isVRMovie;
```

Beschreibung

Diese QuickTime-Darsteller- und Sprite-Eigenschaft gibt an, ob es sich bei einem Darsteller oder Sprite um einen QuickTime VR-Film, der noch nicht heruntergeladen wurde (`TRUE`), oder um einen anderen Darstellertyp handelt (`FALSE`).

Wird diese Eigenschaft in einem anderen Element als `#quickTimeMedia` getestet, wird eine Fehlermeldung angezeigt.

Diese Eigenschaft kann getestet, aber nicht eingestellt werden.

Beispiel

Die folgende Prozedur prüft, ob es sich bei dem Sprite-Darsteller um einen QuickTime-Film handelt. Wenn ja, wird außerdem geprüft, ob es ein QTVR-Film ist. In beiden Fällen wird eine Hinweismeldung ausgegeben.

```
-- Lingo syntax
on checkForVR(theSprite)
    if sprite(theSprite).member.type = #quickTimeMedia then
        if sprite(theSprite).isVRMovie then
            _player.alert("This is a QTVR asset.")
        else
            _player.alert("This is not a QTVR asset.")
        end if
    else
        _player.alert("This is not a QuickTime asset.")
    end if
end

// JavaScript syntax
function checkForVR(theSprite) {
    var memType = sprite(theSprite).member.type;
    if (memType == "quickTimeMedia") {
        var isType = sprite(theSprite).isVRMovie;
        if (isType == 1) {
            _player.alert("This is a QTVR asset.");
        } else {
            _player.alert("This is not a QTVR asset.");
        } else {
            _player.alert("This is not a QuickTime asset.");
        }
    }
}
```

itemDelimiter

Syntax

```
_player.itemDelimiter
```

Beschreibung

Diese Player-Eigenschaft gibt das spezielle Zeichen an, das zur Trennung von Elementen verwendet wird.

Sie können mithilfe von `itemDelimiter` Dateinamen parsen, indem Sie `itemDelimiter` in Windows auf einen umgekehrten Schrägstrich (\) oder auf dem Mac auf einen Doppelpunkt (:) setzen. Für den normalen Betrieb setzen Sie das Zeichen für `itemDelimiter` auf ein Komma (,) zurück.

Diese Funktion kann getestet und eingestellt werden.

Beispiel

Die folgende Prozedur sucht die letzte Komponente in einem Mac-Pfadnamen. Die Prozedur zeichnet zuerst den aktuellen Begrenzer auf und ändert ihn dann in einen Doppelpunkt (:). Wenn der Begrenzer ein Doppelpunkt ist, kann Lingo unter Verwendung von the last item of das letzte Element im Chunk bestimmen, aus dem ein Mac-Pfadname besteht. Bevor die Prozedur beendet wird, wird der Begrenzer auf seinen ursprünglichen Wert zurückgesetzt.

```
on getLastComponent pathName
    save = _player.itemDelimiter
    _player.itemDelimiter = ":"
    f = the last item of pathName
    _player.itemDelimiter = save
    return f
end
```

Siehe auch

Player

kerning

Syntax

```
-- Lingo syntax
memberObjRef.kerning

// JavaScript syntax
memberObjRef.kerning;
```

Beschreibung

Diese Textdarstellereigenschaft gibt an, ob der Text bei Änderungen des Textdarstellerinhalts automatisch unterschnitten wird.

Wenn der Wert auf TRUE gesetzt ist, erfolgt die Unterschneidung automatisch. Ist er auf FALSE gesetzt, findet keine Unterschneidung statt.

Diese Eigenschaft ist standardmäßig auf TRUE gesetzt.

Beispiel

Die folgende Anweisung legt die Eigenschaft kerning des Textdarstellers auf TRUE fest.

```
-- Lingo syntax
member("MyText").kerning=TRUE

// JavaScript syntax
member("MyText").kerning=1;
```

Siehe auch

kerningThreshold

kerningThreshold

Syntax

```
-- Lingo syntax
memberObjRef.kerningThreshold

// JavaScript syntax
memberObjRef.kerningThreshold;
```

Beschreibung
Diese Textdarstellereigenschaft bestimmt die Größe, bei der in einem Textdarsteller eine automatische Unterschneidung erfolgt. Sie ist nur dann wirksam, wenn die Eigenschaft „kerning“ (Unterschneiden) des Darstellers auf „TRUE“ gesetzt ist.

Die Einstellung selbst ist eine Ganzzahl, die den Schriftgrad angibt, bei dem eine Unterschneidung stattfindet.

Der Standardwert dieser Eigenschaft lautet 14 Punkte.

Beispiel
Die folgende Anweisung legt die Schriftgröße der Eigenschaft `kerningThreshold` des Textdarstellers auf 14 fest.

```
-- Lingo syntax
member("MyText").kerningThreshold=14

// JavaScript syntax
member("MyText").kerningThreshold=14;
```

Siehe auch
`kerning`

key

Syntax

```
-- Lingo syntax
_key.key

// JavaScript syntax
_key.key;
```

Beschreibung
Diese Tasteneigenschaft gibt als Wert die Taste zurück, die zuletzt gedrückt wurde. Nur Lesen.

Bei dem zurückgegebenen Wert handelt es sich um den der Taste zugeordneten ANSI-Wert und nicht um den numerischen Wert.

Sie können `key` in Prozeduren verwenden, die bestimmte Aktionen ausführen, wenn der Benutzer bestimmte Tasten als Tastaturkurzbefehle drückt und andere interaktive Aktionen stattfinden. Werden sie in einer primären Ereignisprozedur verwendet, werden die von Ihnen angegebenen Aktionen zuerst ausgeführt.

Hinweis: *Der Wert von* `key` *wird nicht aktualisiert, wenn der Benutzer eine Taste drückt, während sich Lingo oder JavaScript-Syntax in einer Schleife befindet.*

Prüfen Sie mit dem Beispielfilm "Tastatur-Lingo", welche Zeichen den verschiedenen Tasten auf unterschiedlichen Tastaturen entsprechen.

Beispiel
Die folgenden Anweisungen bewirken, dass der Film zur Markierung „Hauptmenü“ zurückkehrt, wenn der Benutzer die Q-Taste drückt. Da die Eigenschaft `keyDownScript` auf `checkKey` gesetzt ist, macht die Prozedur on `prepareMovie` die Prozedur on `checkKey` zur ersten Ereignisprozedur, die beim Drücken einer Taste ausgeführt wird. Die Prozedur on `CheckKey` prüft, ob die Q-Taste gedrückt wird, und springt zur Markierung "Hauptmenü", wenn dies der Fall ist.

```
-- Lingo syntax
on prepareMovie
    keyDownScript = "checkKey"
end

on checkKey
    if (_key.key = "q") then _movie.go("Main Menu")
    end if
end

// JavaScript syntax
function prepareMovie() {
    keyDownScript = checkKey();
}

function checkKey() {
    if (_key.key == "q") {
         _movie.go("Main Menu");
    }
}
```

Die Prozedur on `keyDown` prüft, ob es sich bei der zuletzt gedrückten Taste um die Z-Taste handelt, und ruft die Prozedur on `addNumbers` auf, wenn dies der Fall ist:

```
-- Lingo syntax
on keyDown
    if (_key.key = "z") then addNumbers
end

// JavaScript syntax
function keyDown() {
    if (_key.key == "z") {
        addNumbers();
    }
}
```

Siehe auch

commandDown, Taste

keyboardFocusSprite

Syntax

```
-- Lingo syntax
_movie.keyboardFocusSprite

// JavaScript syntax
_movie.keyboardFocusSprite;
```

Beschreibung

Diese Filmeigenschaft ermöglicht es dem Benutzer, den Fokus für Tastatureingaben auf ein bestimmtes Text-Sprite zu setzen, das sich gegenwärtig auf dem Bildschirm befindet, ohne die Cursorposition zu steuern. Lesen/Schreiben.

Dies entspricht der Benutzung der Tab-Taste, wenn die Eigenschaft `autoTab` des Darstellers ausgewählt ist.

Wenn Sie `keyboardFocusSprite` auf -1 setzen, wird die Steuerung des Tastaturfokus an das Drehbuch zurückgegeben. Wenn Sie diesen Wert auf 0 setzen, wird die Tastatureingabe in einem bearbeitbaren Sprite deaktiviert.

Beispiel

Die folgende Anweisung deaktiviert die Tastatureingabe für alle bearbeitbaren Sprites.

```
-- Lingo syntax
_movie.keyboardFocusSprite=0

// JavaScript syntax
_movie.keyboardFocusSprite=0;
```

Siehe auch

Movie

keyCode

Syntax

```
-- Lingo syntax
_key.keyCode

// JavaScript syntax
_key.keyCode;
```

Beschreibung

Diese Tasteneigenschaft gibt den numerischen Code für die zuletzt gedrückte Taste zurück. Nur Lesen.

Bei dem zurückgegebenen Wert handelt es sich um den numerischen Wert der Taste, nicht um ihren ANSI-Wert.

Mit `keyCode` können Sie erkennen, wann der Benutzer eine Pfeil- oder Funktionstaste gedrückt hat. Dieses Tasten können mit der Eigenschaft `key` nicht angegeben werden.

Prüfen Sie mit dem Beispielfilm "Tastatur-Lingo", welche Zeichen den verschiedenen Tasten auf unterschiedlichen Tastaturen entsprechen.

Beispiel

Die folgende Prozedur zeigt für jede gedrückte Taste den entsprechenden Tastencode im Nachrichtenfenster an:

```
-- Lingo syntax
on enterFrame
    keyDownScript = put(_key.keyCode)
end

// JavaScript syntax
function enterFrame() {
    keyDownScript = put(_key.keyCode);
}
```

Die folgende Anweisung prüft, ob die Nach-oben-Taste (Tastencode 126) gedrückt wurde, und geht zur vorherigen Markierung, wenn dies der Fall ist:

```
-- Lingo syntax
if (_key.keyCode = 126) then
    _movie.goPrevious()
end if

// JavaScript syntax
if (_key.keyCode == 126) {
    _movie.goPrevious();
}
```

Die folgende Prozedur prüft, ob eine Pfeiltaste gedrückt wurde, und reagiert in diesem Fall entsprechend:

```
-- Lingo syntax
on keyDown
    case (_key.keyCode) of
        123: TurnLeft
        126: GoForward
        125: BackUp
        124: TurnRight
    end case
end keyDown

// JavaScript syntax
function keyDown() {
    switch (_key.keyCode) {
        case 123: TurnLeft();
            break;
        case 126: GoForward();
            break;
        case 125: BackUp();
            break;
        case 124: TurnRight();
            break;
    }
}
```

Siehe auch

Taste, key

keyDownScript

Syntax

```
the keyDownScript
```

Beschreibung

Diese Systemeigenschaft bestimmt, welcher Lingo-Code bei einem Tastendruck ausgeführt wird. Der Lingo-Code wird als ein in Anführungszeichen stehender String angegeben und kann eine einfache Anweisung oder ein Skript sein, das eine Prozedur aufruft.

Wenn eine Taste gedrückt wird und die Eigenschaft `keyDownScript` definiert ist, führt Lingo zuerst die für die Eigenschaft `keyDownScript` angegebenen Anweisungen aus. Es werden keine anderen `on keyDown`-Prozeduren ausgeführt, es sei denn, die Anweisungen enthalten den Befehl `pass`, damit die `keyDown`-Nachricht an andere Objekte im Film weitergegeben werden kann.

Die Einstellung der Eigenschaft `keyDownScript` hat die gleiche Wirkung wie die Benutzung des Befehls `when keyDown then`, der in früheren Versionen von Director verwendet wurde.

Wenn die von Ihnen für die Eigenschaft `keyDownScript` angegebenen Anweisungen nicht mehr zutreffen, deaktivieren Sie diese mit der Anweisung `set the keyDownScript to EMPTY`.

Beispiel

Die folgende Anweisung gibt das Skript an, das bei einem Tastendruck ausgeführt wird.

```
-- Lingo syntax
    the keyDownScript = "go to the frame"

// JavaScript syntax
    _system.keyDownScript = "_movie.go(_movie.frame)";
```

Siehe auch

`on keyDown`, `keyUpScript`, `mouseDownScript`, `mouseUpScript`

keyframePlayer (Modifizierer)

Syntax

```
member(whichCastmember).model(whichModel).keyframePlayer.keyframePlayerModifierProperty
```

Beschreibung

Dieser 3D-Modifizierer verwaltet die Verwendung von Bewegungen in Modellen. Die durch den Modifizierer `keyframePlayer` verwalteten Bewegungen animieren das gesamte Modell auf einmal – im Gegensatz zu bonesPlayer-Bewegungen, die nur als "Knochen“ bezeichnete Modellsegmente animieren.

Bewegungen und die Modelle, die sie verwenden, müssen in einem 3D-Modellierungsprogramm erstellt, als W3D-Dateien exportiert und dann in einen Film importiert werden. Auf in Director erstellte Modellprimitive können keine Bewegungen angewendet werden.

Wenn Sie den Modifizierer `keyframePlayer` mit dem Befehl `addModifier` zu einem Modell hinzufügen, können Sie auf die folgenden Eigenschaften des Modifizierers `keyframePlayer` zugreifen:

- `playing` gibt an, ob ein Modell eine Bewegung ausführt.
- `playList` ist eine lineare Liste mit Eigenschaftslisten, die die Abspielparameter der für ein Modell vorgesehenen Bewegungen enthalten.
- `currentTime` gibt die lokale Zeit der gegenwärtig abgespielten oder angehaltenen Bewegung in Millisekunden an.
- `playRate` ist eine Zahl, die mit dem Parameter *scale* des Befehls `play()` oder `queue()` multipliziert wird, um die Wiedergabegeschwindigkeit der Bewegung zu bestimmen.
- `playlist.count` gibt die Anzahl von Bewegungen zurück, die sich gegenwärtig in der Abspielliste befinden.
- `rootLock` gibt an, ob die Translationskomponente der Bewegung verwendet oder ignoriert wird.
- `currentLoopState` gibt an, ob die Bewegung einmal abgespielt oder kontinuierlich wiederholt wird.
- `blendTime` gibt die Dauer des durch den Modifizierer erzeugten Übergangs zwischen den einzelnen Bewegungen an, wenn die Eigenschaft `autoblend` des Modifizierers auf `TRUE` gesetzt ist.
- `autoblend` gibt an, ob der Modifizierer einen linearen Übergang auf die aktuelle Bewegung von der vorhergehenden Bewegung erzeugt.

- `blendFactor` gibt den Grad der Überblendung zwischen den einzelnen Bewegungen an, wenn die Eigenschaft `autoBlend` des Modifizierers auf `FALSE` gesetzt ist.
- `lockTranslation` gibt an, ob das Modell von den angegebenen Ebenen verdrängt werden kann.
- `positionReset` gibt an, ob das Modell nach Ablauf einer Bewegung oder nach jedem Durchlauf einer Schleife an seine Ausgangsposition zurückkehrt.
- `rotationReset` gibt das Rotationselement eines Übergangs von einer Bewegung zur nächsten bzw. den Durchlauf einer Einzelbewegung an.

  Hinweis: *Weitere Informationen zu diesen Eigenschaften finden Sie in den einzelnen Beschreibungen.*

Der Modifizierer `keyframePlayer` verwendet folgende Befehle:

- `pause` hält die Bewegung an, die gegenwärtig vom Modell ausgeführt wird.
- `play()` leitet die Ausführung einer Bewegung ein bzw. setzt sie fort.
- `playNext()` leitet die Wiedergabe der nächsten Bewegung in der Abspielliste ein.
- `queue()` fügt eine Bewegung am Ende der Abspielliste hinzu.

Der Modifizierer `keyframePlayer` generiert die folgenden Ereignisse, die von in den Befehlen `registerForEvent()` und `registerScript()` deklarierten Prozeduren verwendet werden. Der Aufruf der deklarierten Prozedur enthält drei Argumente: den Ereignistyp (entweder `#animationStarted` oder `#animationEnded`), den Namen der Bewegung und die aktuelle Zeit der Bewegung. Weitere Informationen über Benachrichtigungsereignisse enthält der Eintrag zu `registerForEvent()`.

`#animationStarted` wird gesendet, wenn die Wiedergabe einer Bewegung beginnt. Wenn zwischen den einzelnen Bewegungen eine Überblendung stattfindet, wird das Ereignis am Anfang des Übergangs gesendet.

`#animationEnded` wird am Ende einer Bewegung gesendet. Wenn zwischen den einzelnen Bewegungen eine Überblendung stattfindet, wird das Ereignis am Ende des Übergangs gesendet.

Siehe auch

`addModifier`, `modifiers`, `bonesPlayer (Modifizierer)`, `motion`

keyUpScript

Syntax

```
the keyUpScript
```

Beschreibung

Diese Systemeigenschaft gibt Lingo-Code an, der beim Loslassen einer Taste ausgeführt wird. Der Lingo-Code wird als ein in Anführungszeichen stehender String angegeben und kann eine einfache Anweisung oder ein Skript sein, das eine Prozedur aufruft.

Wenn eine Taste losgelassen wird und die Eigenschaft `keyUpScript` definiert ist, führt Lingo zuerst die für die Eigenschaft `keyUpScript` angegebenen Anweisungen aus. Es werden keine anderen `on keyUp`-Prozeduren ausgeführt, es sei denn, die Anweisungen enthalten den Befehl `pass`, damit die `keyUp`-Nachricht an andere Objekte im Film weitergegeben werden kann.

Wenn die von Ihnen für die Eigenschaft `keyUpScript` angegebenen Anweisungen nicht mehr zutreffen, deaktivieren Sie diese unter Verwendung der Anweisung `set the keyUpScript to empty`.

Beispiel

Die folgende Anweisung gibt das Skript an, das bei einem Tastendruck ausgeführt wird.

```
-- Lingo syntax
    the keyUpScript = "go to the frame +1 "

// JavaScript syntax
    _system.keyUpScript = "_movie.go(_movie.frame+1)";
```

Siehe auch

on keyUp

labelList

Syntax

```
the labelList
```

Beschreibung

Diese Systemeigenschaft listet die Bildbeschriftungen im aktuellen Film als einen durch Zeilenumbrüche begrenzten String (keine Liste) mit je einer Beschriftung pro Zeile auf. Beschriftungen werden entsprechend ihrer Reihenfolge im Drehbuch aufgelistet. (Da die Einträge durch den Zeilenumbruch begrenzt sind, ist das Ende des Strings eine leere Zeile nach dem letzten Zeilenumbruch. Vergessen Sie nicht, diese leere Zeile ggf. zu entfernen.)

Beispiel

Die folgende Anweisung listet die Bildbeschriftungen im aktuellen Film als einen durch Zeilenumbrüche begrenzten String (keine Liste) mit je einer Beschriftung pro Zeile auf.

```
put the labelList
```

Siehe auch

frameLabel, label(), marker()

lastChannel

Syntax

```
-- Lingo syntax
_movie.lastChannel

// JavaScript syntax
_movie.lastChannel;
```

Beschreibung

Diese Filmeigenschaft ist die Nummer des letzten Kanals im Film, die im Dialogfeld „Filmeigenschaften“ eingegeben wurde. Nur Lesen.

Ein Beispiel für `lastChannel` in einem fertigen Film finden Sie im Film „QT and Flash“ im Ordner „Learning\\Lingo_Examples“ im Director-Anwendungsordner.

Beispiel

Die folgende Anweisung zeigt die Nummer des letzten Kanals des Films im Nachrichtenfenster an:

```
-- Lingo syntax
put(_movie.lastChannel)

// JavaScript syntax
put(_movie.lastChannel);
```

Siehe auch

Movie

lastClick

Syntax

```
-- Lingo syntax
_player.lastClick

// JavaScript syntax
_player.lastClick;
```

Beschreibung

Diese Player-Eigenschaft gibt die in Ticks (1/60 Sekunde) gemessene Zeit zurück, die seit der letzten Betätigung der Maustaste vergangen ist. Nur Lesen.

Beispiel

Die folgende Anweisung prüft, ob seit dem letzten Mausklick 10 Sekunden vergangen sind, und schickt in diesem Fall den Abspielkopf zur Markierung "Kein Klick“:

```
-- Lingo syntax
if (_player.lastClick > (10 * 60)) then
    _movie.go("No Click")
end if

// JavaScript syntax
if (_player.lastClick > (10 * 60)) {
    _movie.go("No Click");
}
```

Siehe auch

lastEvent, lastKey, lastRoll, Player

lastError

Syntax

```
-- Lingo syntax
memberOrSpriteObjRef.lastError

// JavaScript syntax
memberOrSpriteObjRef.lastError;
```

Beschreibung

Mit dieser Eigenschaft für `RealMedia`-Sprites oder -Darsteller können Sie das letzte von RealPlayer® zurückgegebene Fehlersymbol als Lingo-Symbol abrufen. Die von RealPlayer zurückgegebenen Fehlersymbole sind englischsprachige Textstrings, die bei der Fehlerbehebung von Nutzen sein können. Diese Eigenschaft ist während der Wiedergabe dynamisch und kann getestet, aber nicht eingestellt werden.

Der Wert `#PNR_OK` bedeutet, dass alles normal ist.

Beispiel

Die folgenden Beispiele zeigen, dass der letzte von RealPlayer für Sprite 2 und den Darsteller "Real" zurückgegebene Fehler #PNR_OUTOFMEMORY war:

```
-- Lingo syntax
put(sprite(2).lastError) -- #PNR_OUTOFMEMORY
put(member("Real").lastError) -- #PNR_OUTOFMEMORY

// JavaScript syntax
trace(sprite(2).lastError); // #PNR_OUTOFMEMORY
put(member("Real").lastError); // #PNR_OUTOFMEMORY
```

lastEvent

Syntax

```
-- Lingo syntax
_player.lastEvent

// JavaScript syntax
_player.lastEvent;
```

Beschreibung

Diese Player-Eigenschaft gibt die in Ticks (1/60 Sekunde) gemessene Zeit zurück, die seit dem letzten Mausklick, Rollover oder Tastenanschlag vergangen ist. Nur Lesen.

Beispiel

Die folgende Anweisung prüft, ob seit dem letzten Mausklick, Rollover oder Tastenanschlag 10 Sekunden vergangen sind, und schickt in diesem Fall den Abspielkopf zur Markierung "Hilfe“:

```
-- Lingo syntax
if (_player.lastEvent > (10 * 60)) then
    _movie.go("Help")
end if

// JavaScript syntax
if (_player.lastEvent > (10 * 60)) {
    _movie.go("Help");
}
```

Siehe auch

lastClick, lastKey, lastRoll, Player

lastFrame

Syntax

```
-- Lingo syntax
_movie.lastFrame

// JavaScript syntax
_movie.lastFrame;
```

Beschreibung

Diese Filmeigenschaft zeigt die Nummer des letzten Bildes im Film an. Nur Lesen.

Beispiel

Die folgende Anweisung zeigt die Nummer des letzten Bildes des Films im Nachrichtenfenster an:

```
-- Lingo syntax
put(_movie.lastFrame)

// JavaScript syntax
put(_movie.lastFrame);
```

Siehe auch

Movie

lastKey

Syntax

```
-- Lingo syntax
_player.lastKey

// JavaScript syntax
_player.lastKey;
```

Beschreibung
Diese Systemeigenschaft gibt die in Ticks (ein Sechzigstel einerSekunde) gemessene Zeit an, die seit dem letzten Tastenanschlag vergangen ist. Nur Lesen.

Beispiel
Die folgende Anweisung prüft, ob seit dem letzten Tastenanschlag 10 Sekunden vergangen sind, und schickt in diesem Fall den Abspielkopf zur Markierung "Keine Taste“:

```
-- Lingo syntax
if (_player.lastKey > (10 * 60)) then
    _movie.go("No Key")
end if

// JavaScript syntax
if (_player.lastKey > (10 * 60)) {
    _movie.go("No Key");
}
```

Siehe auch
lastClick, lastEvent, lastRoll, Player

lastRoll

Syntax

```
-- Lingo syntax
_player.lastRoll

// JavaScript syntax
_player.lastRoll;
```

Beschreibung
Diese Player-Eigenschaft gibt die in Ticks (1/60 Sekunde) gemessene Zeit an, die seit der letzten Mausbewegung vergangen ist. Nur Lesen.

Beispiel
Die folgende Anweisung prüft, ob seit der letzten Mausbewegung 45 Sekunden vergangen sind, und schickt in diesem Fall den Abspielkopf zur Markierung „Aufwachen-Schleife“:

```
-- Lingo syntax
if (_player.lastRoll > (45 * 60)) then
    _movie.go("Attract Loop")
end if

// JavaScript syntax
if (_player.lastRoll > (45 * 60)) {
    _movie.go("Attract Loop");
}
```

Siehe auch
lastClick, lastEvent, lastKey, Player

left

Syntax

```
-- Lingo syntax
spriteObjRef.left

// JavaScript syntax
spriteObjRef.left;
```

Beschreibung

Diese Sprite-Eigenschaft identifiziert die linke horizontale Koordinate des Begrenzungsrechtecks eines Sprites. Lesen/Schreiben.

Sprite-Koordinaten werden in Pixel gemessen, wobei mit (0,0) in der linken oberen Ecke der Bühne begonnen wird.

Beispiel

Die folgende Anweisung stellt fest, ob sich der linke Rand des Sprites links neben dem linken Rand der Bühne befindet. Falls sich der linke Rand des Sprites links neben dem linken Bühnenrand befindet, führt das Skript die Prozedur `offLeftEdge` aus:

```
-- Lingo syntax
if (sprite(3).left < 0) then
    offLeftEdge()
end if

// JavaScript syntax
if (sprite(3).left < 0) {
    offLeftEdge();
}
```

Die folgende Anweisung misst die linke horizontale Koordinate des Sprites mit der Nummer (i + 1) und ordnet diesen Wert der Variablen mit dem Namen vLowest zu:

```
-- Lingo syntax
vLowest = sprite(i + 1).left

// JavaScript syntax
var vLowest = sprite(i + 1).left
```

Siehe auch

bottom, height, locH, locV, right, Sprite, top, width

left (3D)

Syntax

```
member(whichCastmember).modelResource(whichModelResource).left
```

Beschreibung

Diese 3D-Eigenschaft der Modellressource `#box` gibt an, ob die Seite der Box, die von der -X-Achse geschnitten wird, geschlossen (`TRUE`) oder offen ist (`FALSE`).

Der Standardwert dieser Eigenschaft lautet `TRUE`.

Beispiel

Die folgende Anweisung setzt die Eigenschaft left der Modellressource "Kiste" auf FALSE, d. h. die linke Seite der Box ist offen:

```
member("3D World").modelResource("crate").left = FALSE
```

Siehe auch

`back`, `front`, `bottom (3D)`, `top (3D)`, `right (3D)`

leftIndent

Syntax

```
chunkExpression.leftIndent
```

Beschreibung

Diese Textdarstellereigenschaft enthält die Anzahl der Pixel, um die der linke Rand von *chunkExpression* von der linken Seite des Textdarstellers versetzt ist.

Bei diesem Wert handelt es sich um eine Ganzzahl größer oder gleich 0.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Die folgende Anweisung rückt den Textdarsteller `myText` um zehn Pixel nach links ein.

```
-- Lingo syntax
member("myText").leftIndent=10

// JavaScript syntax
member("myText").leftIndent=10;
```

Siehe auch

`firstIndent`, `rightIndent`

length (byte array)

Syntax

```
byteArrayObject.length
```

Beschreibung

Diese Bytearrayeigenschaft gibt die Größe des Bytearrays in Byte zurück.

Beispiele

```
--Lingo syntax
bArray = byteArray("Sample byte array")
put bArray.length

//JavaScript syntax
bArray = byteArray("Sample byte array");
put(bArray.length);
```

length (3D)

Syntax

```
member(whichCastmember).modelResource(whichModelResource).length
vectorReference.length
```

Beschreibung

Diese 3D-Eigenschaft für Modellressourcen vom Typ `#box` oder `#plane` sowie für Vektoren gibt die Länge der Box oder Ebene in Welteinheiten an.

Die Länge einer Box wird entlang ihrer Z-Achse gemessen. Die Standardlänge einer Box beträgt 50.

Die Länge einer Ebene wird entlang ihrer Y-Achse gemessen. Die Standardlänge einer Ebene beträgt 1.

Die Länge eines Vektors ist seine Entfernung von `vector(0, 0, 0)` in Welteinheiten. Dieser Wert ist mit dem Vektorbetrag identisch.

Beispiel

Die folgende Anweisung setzt die Variable `myBoxLength` auf die Länge der Modellressource `GiftBox`.

```
myBoxLength = member("3D World").modelResource("GiftBox").length
```

Siehe auch

`height (3D)`, `width (3D)`, `magnitude`

lengthVertices

Syntax

```
member(whichCastmember).modelResource(whichModelResource).lengthVertices
```

Beschreibung

Diese 3D-Eigenschaft für Modellressourcen vom Typ `#box` und `#plane` gibt die Anzahl von Gitternetzscheitelpunkten entlang der Box- oder Ebenenlänge an. Wenn Sie einen höheren Wert angeben, erhöht sich die Anzahl von Flächen und es ergibt sich ein feineres Gitternetz.

Die Länge einer Box wird entlang ihrer Z-Achse gemessen. Die Länge einer Ebene wird entlang ihrer Y-Achse gemessen.

Setzen Sie die Eigenschaft `renderStyle` des Shaders eines Modells auf `#wire`, um die Flächen des Gitternetzes der Modellressource zu sehen. Setzen Sie die Eigenschaft `renderStyle` auf `#point`, um nur die Scheitelpunkte des Gitternetzes zu sehen.

Der Wert dieser Eigenschaft muss größer oder gleich 2 sein. Der Standardwert ist 4.

Beispiel

Die folgende Anweisung setzt die Eigenschaft `lengthVertices` der Modellressource "Turm" auf 10. Die Geometrie der Modellressource wird entlang der Y-Achse durch neun Dreiecke definiert; insgesamt gibt es also zehn Scheitelpunkte.

```
member("3D World").modelResource("Tower").lengthVertices = 10
```

Siehe auch

`length (3D)`

level

Syntax

```
member(whichCastmember).model(whichModel).lod.level
```

Beschreibung

Diese 3D-Eigenschaft für den Modifizierer `lod` gibt an, wie viele Details durch den Modifizierer entfernt werden, wenn die Eigenschaft `auto` auf `FALSE` gesetzt ist. Der Gültigkeitsbereich dieser Eigenschaft liegt zwischen 0.0 und 100.00.

Wenn die Modifiziereigenschaft `auto` auf `TRUE` gesetzt ist, wird der Wert der Eigenschaft `level` zwar dynamisch aktualisiert, kann aber nicht festgelegt werden.

Der Modifizierer `#lod` kann nur zu Modellen hinzugefügt werden, die außerhalb von Director in einem 3D-Modellierungsprogramm erstellt wurden. Der Wert der Eigenschaft `type` der von diesen Modellen verwendeten Modellressourcen lautet `#fromFile`. Der Modifizierer kann nicht zu in Director erstellten Primitiven hinzugefügt werden.

Beispiel

Die folgende Anweisung setzt die Eigenschaft level des Modifizierers lod im Modell „Raumschiff“ auf 50. Wenn die Eigenschaft auto des Modifizierers lod auf FALSE gesetzt ist, wird das Raumschiff mit mittlerem Detailliertheitsgrad gezeichnet. Ist die Eigenschaft auto des Modifizierers lod auf TRUE gesetzt, ist dieser Code wirkungslos.

```
member("3D World").model("Spaceship").lod.level = 50
```

Siehe auch

`lod (Modifizierer)`, `auto`, `bias`

lifetime

Syntax

```
member(whichCastmember).modelResource(modelResource).lifetime
```

Beschreibung

Diese 3D-Eigenschaft für Modellressourcen vom Typ `#particle` gibt die Lebensdauer alle Partikel in einem Partikelsystem in Millisekunden an, d. h. den Zeitraum zwischen ihrem Entstehen und Erlöschen.

Der Standardwert dieser Eigenschaft lautet 10,000.

Beispiel

Das folgende Beispiel legt die Eigenschaft lifetime eines Partikels im Darsteller "3Dobjects" auf 100 Millisekunden fest.

```
-- Lingo syntax
member("3Dobjects").modelResource("Particle01").lifetime = 100.0

// JavaScript syntax
member("3Dobjects").getPropRef("modelResource",10).lifetime = 100.0;
```

Siehe auch

emitter

light

Syntax

```
member(whichCastmember).light(whichLight)
member(whichCastmember).light[index]
member(whichCastmember).light(whichLight).whichLightProperty
member(whichCastmember).light[index].whichLightProperty
```

Beschreibung

Dieses 3D-Element ist ein Objekt an einer Vektorposition, von der Licht ausstrahlt.

Beispiel

Das folgende Beispiel zeigt das erste Licht im Darsteller „3Dobjects“ an.

```
-- Lingo syntax
putmember("3Dobjects").light(1)

// JavaScript syntax
put( member("3Dobjects").getPropRef("light",1));
```

Siehe auch

newLight, deleteLight

lineColor

Syntax

```
member(whichCastmember).model(whichModel).inker.lineColor
member(whichCastmember).model(whichModel).toon.lineColor
```

Beschreibung

Diese 3D-Eigenschaft für die Modifizierer `toon` und `inker` gibt die Farbe der Linien an, die vom Modifizierer im Modell gezeichnet werden. Damit diese Eigenschaft eine Wirkung hat, muss die Modifiziereregenschaft `creases`, `silhouettes` oder `boundary` auf `TRUE` gesetzt sein.

Der Standardwert dieser Eigenschaft lautet `rgb(0, 0, 0)`.

Beispiel

Die folgende Anweisung setzt die Farbe aller Linien, die vom Modifizierer toon im zweiten gezeichnet werden, auf "rgb(255, 0, 0)" (Rot):

```
-- Lingo syntax
member("3Dobjects").model[2].toon.lineColor = rgb(255, 0, 0)

// JavaScript syntax
member("3Dobjects").getPropRef("model",2).toon.lineColor =color(100,0,0);
```

Siehe auch

`creases`, `silhouettes`, `boundary`, `lineOffset`

lineCount

Syntax

```
-- Lingo syntax
memberObjRef.lineCount

// JavaScript syntax
memberObjRef.lineCount;
```

Beschreibung

Diese Darstellereigenschaft gibt die Anzahl der Zeilen an, die im Felddarsteller auf der Bühne erscheinen, wobei diese Zahl der Anzahl der Zeilenumbrüche und nicht der Anzahl der Zeilenschaltungen im String entspricht.

Beispiel

Die folgende Anweisung bestimmt, wie viele Zeilen der Felddarsteller „Nachrichten“ bei seinem Auftritt auf der Bühne enthält, und weist diesen Wert der Variablen `numberOfLines` zu:

```
--Lingo syntax
numberOfLines = member("Today's News").lineCount

// JavaScript syntax
var numberOfLines = member("Today's News").lineCount;
```

lineDirection

Syntax

```
member(whichCastMember).lineDirection
```

Beschreibung

Diese Formdarsteller-Eigenschaft enthält eine 0 oder 1, wodurch die Steigung der gezeichneten Linie angezeigt wird.

Wenn die Linie von links nach rechts ansteigt, ist diese Eigenschaft auf 1, fällt sie von links nach rechts ab, ist die Eigenschaft auf 0 eingestellt.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Das folgende Beispiel wechselt zwischen steigender und fallender Linie im Darsteller `theLine` und erzeugt so einen Sägezahneffekt:

```
-- Lingo syntax
member("theLine").lineDirection = not member("theLine").lineDirection

// JavaScript syntax
member("theLine").lineDirection = ! (member("theLine").lineDirection);
```

lineHeight

Syntax

```
member(whichCastMember).lineHeight
the lineHeight of member whichCastMember
```

Beschreibung

Diese Darstellereigenschaft legt den Zeilenabstand fest, der für die Anzeige des angegebenen Felddarstellers verwendet wird. Der Parameter *whichCastMember* kann entweder ein Darstellername oder eine Darstellernummer sein.

Durch Einstellen der Darstellereigenschaft `lineHeight` wird die Systemeinstellung vorübergehend außer Kraft gesetzt, bis der Film endet. Wenn Sie den gewünschten Zeilenabstand im gesamten Film verwenden wollen, stellen Sie die Darstellereigenschaft `lineHeight` in einer `on prepareMovie`-Prozedur ein.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Die folgende Anweisung legt die Linienhöhe im Darsteller `myText` fest.

```
-- Lingo syntax
member("myText").lineHeight=20

// JavaScript syntax
member("myText").lineHeight=20;
```

Siehe auch

`text`, `alignment`, `font`, `fontSize`, `fontStyle`

lineOffset

Syntax

```
member(whichCastmember).model(whichModel).toon.lineOffset
member(whichCastmember).model(whichModel).inker.lineOffset
```

Beschreibung

Diese 3D-Eigenschaft für die Modifizierer `toon` und `inker` gibt an, in welcher scheinbaren Entfernung von der Modelloberfläche vom Modifizierer Linien gezeichnet werden. Damit diese Eigenschaft eine Wirkung hat, muss die Modifiziereigenschaft `useLineOffset` auf `TRUE` gesetzt sein. Darüber hinaus müssen die Eigenschaften `creases`, `silhouettes` und/oder `boundary` ebenfalls auf `TRUE` gesetzt sein.

Der Wert dieser Eigenschaft liegt im Bereich -100,00 bis +100,00. Die Standardeinstellung ist -2.0.

Beispiel

Die folgende Anweisung setzt die Eigenschaft `lineOffset` des Modifizierers "toon" im Modell "Teekanne" auf 10. Die von diesem Modifizierer auf der Modelloberfläche gezeichneten Linien stehen weiter hervor als bei der Standardeinstellung (-2).

```
member("shapes").model("Teapot").toon.lineOffset = 10
```

Siehe auch

`creases`, `silhouettes`, `boundary`, `useLineOffset`, `lineColor`

lineSize

Syntax

```
member(whichCastMember).lineSize
the lineSize of member whichCastMember
sprite whichSprite.lineSize
the lineSize of sprite whichSprite
```

Beschreibung

Diese Formdarstellereigenschaft legt die Rahmenstärke des angegebenen Formdarstellers, der auf der Bühne dargestellt wird, in Pixeln fest. Bei nicht rechteckigen Formen ist der Rahmen der Rand der Form und nicht das Begrenzungsrechteck.

Die `lineSize`-Einstellung `des`Sprites hat Vorrang vor der `lineSize`-Einstellung `des` Darstellers. Wenn Lingo die Einstellung `lineSize` des Darstellers ändert, während sich ein Sprite auf der Bühne befindet, bleibt die `lineSize`-Einstellung des Sprites so lange gültig, bis das Sprite abgeschlossen ist.

Der von Lingo eingestellte Wert ist nur dann über das aktuelle Sprite hinaus gültig, wenn es sich bei diesem um einmit Skript erstelltes Sprite handelt.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Die folgende Anweisung stellt die Stärke des Formdarstellers `myShape` auf 5 Pixel ein.

```
--Lingo syntax
member("myShape").lineSize=5

// JavaScript syntax
member("myShape").lineSize=5;
```

linked

Syntax

```
-- Lingo syntax
memberObjRef.linked

// JavaScript syntax
memberObjRef.linked;
```

Beschreibung

Diese Darstellereigenschaft bestimmt, ob ein Skript, Flash-Film oder animiertes GIF in einer externen Datei (`TRUE`, Standard) oder in der Besetzung von Director (`FALSE`) gespeichert ist. Lesen/Schreiben für Skript-, Flash- und animierte GIF-Darsteller. Nur Lesen für alle anderen Darstellertypen.

Wenn die Daten extern in einer verknüpften Datei gespeichert sind, muss die Eigenschaft `pathName` des Darstellers auf die Position verweisen, an der sich die Filmdatei befindet.

Beispiel

Die folgende Anweisung konvertiert den Flash-Darsteller `homeBodies` von einem verknüpften in einen intern gespeicherten Darsteller.

```
-- Lingo syntax
member("homeBodies").linked = 0

// JavaScript syntax
member("homeBodies").linked = 0;
```

Siehe auch

Member

loaded

Syntax

```
-- Lingo syntax
memberObjRef.loaded

// JavaScript syntax
memberObjRef.loaded;
```

Beschreibung

Diese Darstellereigenschaft gibt an, ob ein angegebener Darsteller in den Speicher geladen wurde (`TRUE`) oder nicht (`FALSE`). Nur Lesen.

Die verschiedenen Darstellertypen weisen ein unterschiedliches Ladeverhalten auf:

- Form- und Skriptdarsteller sind stets im Speicher geladen.
- Filmdarsteller werden nie entladen.
- Digitalvideo-Darsteller können unabhängig davon, ob sie benutzt werden, vorausgeladen und entladen werden. (Ein Digitalvideo-Darsteller wird schneller aus dem Speicher abgespielt als vom Datenträger.)

Beispiel

Die folgende Anweisung prüft, ob der Darsteller "Demofilm“ im Speicher geladen ist, und geht zu einem anderen Film, wenn dies nicht der Fall ist:

```
-- Lingo syntax
if member("Demo Movie").loaded = FALSE then
    _movie.go(1, "Waiting.dir")
end if

// JavaScript syntax
if (member("Demo Movie").loaded == false) {
    _movie.go(1, "Waiting.dir")
}
```

Siehe auch

Member

loc (backdrop and overlay)

Syntax

```
sprite(whichSprite).camera{(index)}.backdrop[index].loc
member(whichCastmember).camera(whichCamera).backdrop[index].loc
sprite(whichSprite).camera{(index)}.overlay[index].loc
member(whichCastmember).camera(whichCamera).overlay[index].loc
```

Beschreibung

Diese 3D-Eigenschaft für Hintergründe und Überlagerungen gibt die 2D-Position des Hintergrunds bzw. der Überlagerung von der oberen linken Ecke des Sprites gemessen an.

Diese Eigenschaft wird anfänglich als Parameter des Befehls `addBackdrop`, `addOverlay`, `insertBackdrop` oder `insertOverlay` festgelegt, mit dem der Hintergrund bzw. die Überlagerung erstellt wird.

Beispiel

Die folgende Anweisung positioniert den ersten Hintergrund der Kamera von Sprite 2:

```
sprite(2).camera.backdrop[1].loc = point(120, 120)
```

Siehe auch

bevelDepth, overlay, regPoint (3D)

locH

Syntax

```
-- Lingo syntax
spriteObjRef.locH

// JavaScript syntax
spriteObjRef.locH;
```

Beschreibung

Diese Sprite-Eigenschaft gibt die horizontale Position des Registrierungskreuzes eines Sprites an. Lesen/Schreiben.

Sprite-Koordinaten beziehen sich auf die linke obere Ecke der Bühne.

Wenn der Wert über das aktuelle Sprite hinaus gültig bleiben soll, machen Sie das Sprite zu einem mit Skript erstellten Sprite.

Beispiel

Die folgende Anweisung stellt Sprite 15 an dieselbe horizontale Position wie die mit der Maus angeklickte Stelle:

```
-- Lingo syntax
sprite(15).locH = _mouse.mouseH

// JavaScript syntax
sprite(15).locH = _mouse.mouseH;
```

Siehe auch

`bottom`, `height`, `left`, `locV`, `point()`, `right`, `Sprite`, `top`, `updateStage()`

lockTranslation

Syntax

```
member(whichCastmember).model(whichModel).bonesPlayer.lockTranslation
member(whichCastmember).model(whichModel).keyframePlayer.lockTranslation
```

Beschreibung

Diese 3D-Eigenschaft für die Modifizierer `#bonesPlayer` und `#keyframePlayer` verhindert eine Verschiebung aus der bzw. den angegebenen Ebene(n). Die einzige Ausnahme ist bei einer absoluten Translation der Bewegungsdaten. Zusätzliche Translationen, die manuell oder durch kumulativen Fehler eingeführt werden, werden entfernt. Mögliche Werte: `#none`, `#x`, `#y`, `#z`, `#xy`, `#yz`, `#xz` und `#all`. Diese Werte bestimmen, welche der drei Translationskomponenten für jedes Bild gesteuert werden. Bei Aktivierung einer Achsensperre wird die aktuelle Verschiebung entlang dieser Achse gespeichert und dann als feste Verschiebung verwendet, an der sich die Animation orientiert. Diese Verschiebung kann durch Deaktivieren der Achsensperre, Verschieben des Objekts und Reaktivieren der Achsensperre geändert werden.

Diese Eigenschaft definiert somit die bei der Wiedergabe einer Bewegung zu ignorierende Translationsachse. Wenn ein Modell so auf einer Grundebene fixiert werden soll, dass seine Oberseite entlang der *Z*-Achse positioniert ist, setzen Sie `lockTranslation` auf `#z`. Der Standardwert dieser Eigenschaft lautet `#none`.

Beispiel

Die folgende Anweisung setzt die Eigenschaft `lockTranslation` des Modells "Spaziergänger" auf `#z`.

```
member("ParkScene").model("Walker").bonesPlayer.lockTranslation = #z
```

Siehe auch

`immovable`

locV

Syntax

```
-- Lingo syntax
spriteObjRef.locV

// JavaScript syntax
spriteObjRef.locV;
```

Beschreibung

Diese Sprite-Eigenschaft gibt die vertikale Position des Registrierungskreuzes eines Sprites an. Lesen/Schreiben.

Sprite-Koordinaten beziehen sich auf die linke obere Ecke der Bühne.

Wenn der Wert über das aktuelle Sprite hinaus gültig bleiben soll, machen Sie das Sprite zu einem mit Skript erstellten Sprite.

Beispiel

Die folgende Anweisung stellt Sprite 15 an dieselbe vertikale Position wie die mit der Maus angeklickte Stelle:

```
-- Lingo syntax
sprite(15).locV = _mouse.mouseV

// JavaScript syntax
sprite(15).locV = _mouse.mouseV;
```

Siehe auch

`bottom`, `height`, `left`, `locH`, `point()`, `right`, `Sprite`, `top`, `updateStage()`

locZ

Syntax

```
-- Lingo syntax
spriteObjRef.locZ

// JavaScript syntax
spriteObjRef.locZ;
```

Beschreibung

Diese Sprite-Eigenschaft gibt die dynamische Z-Folge eines Sprites an und steuert die Anordnung in Ebenen, ohne dass Sprite-Kanäle und Eigenschaften manipuliert werden müssen. Lesen/Schreiben.

Diese Eigenschaft kann einen ganzzahligen Wert zwischen minus 2 Milliarden und plus 2 Milliarden haben. Höhere Werte führen dazu, dass das Sprite vor anderen Sprites mit niedrigeren Werten erscheint. Falls zwei Sprites denselben `locZ`-Wert aufweisen, wird die endgültige Reihenfolge zur Anzeige dieser beiden Sprites durch die Kanalnummer entschieden. Sprites in Kanälen mit niedrigeren Nummern erscheinen demzufolge hinter Sprites in Kanälen mit höheren Nummern, auch wenn die `locZ`-Werte gleich sind.

Standardmäßig entspricht der `locZ`-Wert eines Sprites seiner Kanalnummer.

Von Ebenen abhängige Vorgänge, wie beispielsweise die Registrierung von Klicks oder anderen Mausereignissen, werden von den Sprite-Werten für `locZ` bestimmt. Daher kann die Änderung des `locZ`-Wertes eines Sprites dazu führen, dass dieses ganz oder teilweise von anderen Sprites verdeckt wird und der Benutzer nicht mehr dazu in der Lage ist, es anzuklicken.

Andere Director-Funktionen beachten die `locZ`-Folge von Sprites nicht. Erzeugte Ereignisse beginnen ungeachtet der Z-Folge der Sprites immer mit Kanal 1 und gehen von dort aus aufwärts.

Beispiel

Die folgende Prozedur verwendet eine globale Variable namens `gHighestSprite`, die in der Prozedur `startMovie` initialisiert und auf die Anzahl der genutzten Sprites eingestellt wurde. Wenn der Benutzer auf das Sprite klickt, wird sein „locZ“-Wert auf `gHighestSprite` + 1 gesetzt, sodass es in den Vordergrund der Bühne geholt wird. Gleichzeitig wird `gHighestSprite` als Vorbereitung auf den nächsten `mouseUp`-Aufruf um 1 erhöht.

```
-- Lingo syntax
on mouseUp me
    global gHighestSprite
    sprite(me.spriteNum).locZ = gHighestSprite + 1
    gHighestSprite = gHighestSprite + 1
end

// JavaScript syntax
function mouseUp() {
    _global.gHighestSprite;
    sprite(this.spriteNum).locZ = _global.gHighestSprite + 1
    _global.gHighestSprite = _global.gHighestSprite + 1
}
```

Siehe auch

`locH`, `locV`, `Sprite`

lod (Modifizierer)

Syntax

```
member(whichCastmember).model(whichModel).lod.lodModifierProperty
```

Beschreibung

Durch diesen 3D-Modifizierer werden Details dynamisch aus Modellen entfernt, wenn diese sich von der Kamera weg bewegen.

Dieser Modifizierer kann nur zu Modellen hinzugefügt werden, die außerhalb von Director in einem 3D-Modellierungsprogramm erstellt wurden. Der Wert der Eigenschaft `type` der von diesen Modellen verwendeten Modellressourcen lautet `#fromFile`. Bei allen solchen Modellen erfolgt eine Reduzierung der Details, unabhängig davon, ob der Modifizierer `lod` angebracht ist oder nicht. Durch Anbringung des Modifizierers können Sie die Eigenschaften der Detailentfernung steuern. Der Modifizierer kann nicht zu in Director erstellten Primitiven hinzugefügt werden.

Die Daten für den Modifizierer `lod` werden für alle Modelle durch 3D-Modellierungsprogramme erstellt. Wenn Sie die Eigenschaft `userData` auf `"sw3d_no_lod = true"` setzen, werden die `lod`-Modifiziererdaten und der dadurch belegte Speicher nach Abschluss des Streamings freigegeben.

Sie sollten die Modifizierer `sds` und `lod` nicht zusammen verwenden, da sie gegenteilige Funktionen ausüben: Der Modifizierer `sds` fügt geometrische Details hinzu, der Modifizierer `lod` entfernt sie. Vor Verwendung des Modifizierers `sds` sollten Sie wie folgt die Modifiziereigenschaft `lod.auto` deaktivieren und die Modifiziereigenschaft `lod.level` auf die höchstmögliche Auflösung setzen:

```
member("myMember").model("myModel").lod.auto = 0
member("myMember").model("myModel").lod.level = 100
member("myMember").model("myModel").addmodifier(#sds)
```

Der Modifizierer `lod` weist folgende Eigenschaften auf:

- `auto` ermöglicht es dem Modifizierer, den Grad der Detailentfernung dynamisch einzustellen, wenn sich die Entfernung zwischen Modell und Kamera ändert. Der Wert der Modifiziereigenschaft `level` wird zwar aktualisiert, doch die Einstellung der Eigenschaft `level` bleibt ohne Wirkung, wenn die Eigenschaft `auto` auf `TRUE` gesetzt ist.
- `bias` bestimmt, wie aggressiv der Modifizierer Details aus dem Modell entfernt, wenn die Eigenschaft `auto` auf `TRUE` gesetzt ist. Der Gültigkeitsbereich dieser Eigenschaft liegt zwischen 0.0 (alle Polygone entfernen) und 100.0 (keine Polygone entfernen). Der Standardwert dieser Eigenschaft ist 100.0.
- `level` gibt den Grad der Detailentfernung an, wenn die Modifiziereigenschaft `auto` auf `FALSE` gesetzt ist. Der Gültigkeitsbereich dieser Eigenschaft liegt zwischen 0.0 und 100.00.

Hinweis: *Weitere Informationen zu diesen Eigenschaften finden Sie in den einzelnen Beschreibungen.*

Beispiel

Das folgende Beispiel entfernt dynamisch das Modell „Sphere01“ aus dem Darsteller „3Dobjects“.

```
--Lingo
member("3Dobjects").model("Sphere01").lod.auto=1
member("3Dobjects").model("Sphere01").lod.bias=0

// Javascript
member("3Dobjects").getPropRef("model",2).lod.auto=1;
member("3Dobjects").getPropRef("model",2).lod.bias=0;
```

Siehe auch

`sds (Modifizierer)`, `auto`, `bias`, `level`, `addModifier`

loop (3D)

Syntax

```
member(whichCastmember).loop
```

Beschreibung

Diese 3D-Darstellereigenschaft bestimmt, ob Bewegungen, die auf das erste Modell im Darsteller angewendet werden, kontinuierlich wiederholt (TRUE) oder einmal abgespielt und dann gestoppt werden (FALSE).

Die Standardeinstellung dieser Eigenschaft lautet TRUE.

Beispiel

Die folgende Anweisung setzt die Eigenschaft loop des Darstellers „Spaziergänger“ auf TRUE. Vom ersten Modell in "Spaziergänger“ ausgeführte Bewegungen werden kontinuierlich wiederholt.

```
member("Walkers").loop = TRUE
```

Siehe auch

motion, play() (3D), queue() (3D), animationEnabled

loop (emitter)

Syntax

```
member(whichCastmember).modelResource(whichModelResource).emitter.loop
```

Beschreibung

Mit dieser 3D-Eigenschaft können Sie bei einer Modellressource vom Typ #particle ermitteln und festlegen, was mit Partikeln am Ende ihrer Lebensdauer geschieht. Bei TRUE werden Partikel am Ende ihrer Lebensdauer an der durch die Emittereigenschaft region definierten Emitterposition regeneriert. Bei FALSE erlöschen die Partikel am Ende ihrer Lebensdauer. Die Standardeinstellung dieser Eigenschaft lautet TRUE.

Beispiel

In diesem Beispiel ist ThermoSystem eine Modellressource vom Typ #particle. Die folgende Anweisung setzt die Eigenschaft „emitter.loop“ von ThermoSystem auf 1, was zu einer kontinuierlichen Emission der Partikel von ThermoSystem führt:

```
member("Fires").modelResource("ThermoSystem").emitter.loop = 1
```

Siehe auch

emitter

loop (Darsteller)

Syntax

```
-- Lingo syntax
memberObjRef.loop

// JavaScript syntax
memberObjRef.loop;
```

Beschreibung
Diese Darstellereigenschaft bestimmt, ob der angegebene Digitalvideo-, Sound- oder Flash-Filmdarsteller in Schleife abgespielt wird (TRUE) oder nicht (FALSE).

Beispiel
Die folgende Anweisung bewirkt, dass der QuickTime-Filmdarsteller „Demo“ in Schleife abgespielt wird:

```
-- Lingo syntax
member("Demo").loop = 1

// JavaScript syntax
member("Demo").loop = 1;
```

loop (MP4Media/FLV)

Syntax

```
member(1).loop = true
sprite(1).loop = true
```

Beschreibung
Sprite- und Darstellereigenschaft, die festlegt, ob der angegebene MP4Media-Darsteller in einer Schleife wiederholt abgespielt wird (True) oder nicht (False).

Beispiele
Folgendes Beispiel versetzt den MP4Media-Darsteller `Demo` in eine Schleife:

```
-- Lingo syntax
member("Demo").loop = 1

// JavaScript syntax
member("Demo").loop = 1;
```

loop (Flash)

Syntax

```
sprite(whichFlashSprite).loop
the loop of sprite whichFlashSprite
member (whichFlashMember).loop
the loop of member whichFlashMember
```

Beschreibung
Diese Flash-Sprite- und Darstellereigenschaft bestimmt, ob ein Flash-Film in einer kontinuierlichen Schleife (TRUE) oder nur einmal abgespielt wird und dann stoppt (FALSE).

Diese Eigenschaft kann sowohl getestet als auch eingestellt werden.

Beispiel

Das folgende Bildskript prüft den Streamstatus eines verknüpften Flash-Darstellers namens `flashObj` unter Verwendung der Eigenschaft `percentStreamed`. Während `flashObj` heruntergeladen wird, wird der Film im aktuellen Bild in Schleife abgespielt. Nachdem `flashObj` heruntergeladen ist, springt der Film zum nächsten Bild und die Eigenschaft loop des Flash-Films in Kanal 1 wird auf FALSE gesetzt, sodass der Film bis zum Ende abgespielt und dann beendet wird (dieses Sprite wurde beim Herunterladen von `flashObj` in Schleife abgespielt).

```
-- Lingo syntax
on exitFrame me
if member("flashObj").percentStreamed = 100 then
    member(1).loop = FALSE
    go the frame + 1
else
    go to the frame
end if

end
// JavaScript syntax
function exitFrame(me) {
if(member("flashObj").percentStreamed == 100)
    {
        member(1).loop = 0;
        _movie.go(_movie.frame+1);
    }
    else
    {
        _movie.go(_movie.frame);
    }
}
```

loop (Windows Media)

Syntax

```
-- Lingo syntax
windowsMediaObjRef.loop

// JavaScript syntax
windowsMediaObjRef.loop;
```

Beschreibung

Diese Windows Media-Eigenschaft bestimmt, ob ein fertiger Film in Schleife wiedergegeben wird (`TRUE`, Standard) oder nicht (`FALSE`). Lesen/Schreiben.

Beispiel

Die folgende Anweisung gibt an, dass der Darsteller „Classical“ nach der Fertigstellung in Schleife wiedergegeben wird:

```
-- Lingo syntax
member("Classical").loop = TRUE

// JavaScript syntax
member("Classical").loop = true;
```

Siehe auch

Windows Media

loopBounds

Syntax

```
-- Lingo syntax
spriteObjRef.loopBounds

// JavaScript syntax
spriteObjRef.loopBounds;
```

Beschreibung

Diese QuickTime-Sprite-Eigenschaft stellt interne Schleifenpunkte für einen QuickTime-Darsteller oder ein QuickTime-Sprite ein. Die Schleifenpunkte werden in Form einer Director-Liste vorgegeben: [*startTime*, *endTime*].

Die Parameter *startTime* und *endTime* müssen folgende Anforderungen erfüllen:

- Beide Parameter müssen ganzzahlige Werte sein, die die Zeiten in Director-Ticks angeben.
- Die Werte müssen zwischen 0 und der Dauer des QuickTime-Darstellers liegen.
- Die Startzeit muss niedriger als die Schlusszeit sein.

Falls eine dieser Anforderungen nicht erfüllt ist, wird der QuickTime-Film über seine gesamte Länge hinweg in Schleife abgespielt.

Die Eigenschaft `loopBounds` hat keinen Einfluss, wenn die Eigenschaft `loop` des Films auf `FALSE` gesetzt ist. Wird die Eigenschaft `loop` beim Abspielen des Films auf `TRUE` gesetzt, wird die Filmwiedergabe fortgesetzt. Director entscheidet anhand der folgenden Regeln, ob und wie der Film in Schleife abgespielt wird:

- Wenn die durch `loopBounds` angegebene Schlusszeit erreicht ist, kehrt der Film zur Startzeit zurück.
- Wenn das Ende des Films erreicht ist, kehrt der Film zum Filmanfang zurück.

Wird die Eigenschaft `loop` während der Wiedergabe des Films deaktiviert, läuft der Film weiter. Director stoppt, wenn das Ende des Films erreicht ist.

Diese Eigenschaft kann getestet und eingestellt werden. Die Standardeinstellung ist [0,0].

Beispiel

Das folgende Sprite-Skript stellt die Start- und Schlusszeiten für die Wiedergabe eines QuickTime-Sprites in Schleife ein. Die Zeiten werden durch Angabe von Sekunden festgelegt, die dann durch Multiplizieren mit 60 in Ticks konvertiert werden.

```
-- Lingo syntax
on beginSprite me
    sprite(me.spriteNum).loopBounds = [(16 * 60),(32 * 60)]
end

// JavaScript syntax
function beginSprite() {
    sprite(me.spriteNum).loopBounds = list((16 * 60),(32 * 60));
}
```

loopCount

Syntax

```
-- Lingo syntax
soundChannelObjRef.loopCount

// JavaScript syntax
soundChannelObjRef.loopCount;
```

Beschreibung

Diese Soundkanaleigenschaft gibt an, wie oft der aktuelle Sound in einem Soundkanal insgesamt in Schleife abgespielt wird. Nur Lesen.

Die Standardeinstellung für den Wert dieser Eigenschaft ist 1 für Sounds, die sich einfach in der Warteschlange befinden und nicht in einer internen Schleife abgespielt werden.

Sie können auch nur einen Teil eines Sounds in Schleife abspielen, indem Sie die Parameter `loopStartTime`, `loopEndTime` und `loopCount` mit einer `queue()`- oder `setPlayList()`-Methode übergeben. Dies sind die einzigen Möglichkeiten, diese Eigenschaft einzustellen.

Wenn `loopCount` auf 0 gesetzt ist, wird die Schleife endlos wiederholt. Wenn die Eigenschaft `loop` des Sounddarstellers auf `TRUE` gesetzt ist, gibt `loopCount` den Wert 0 zurück.

Beispiel

Die folgende Prozedur stellt zwei Sounds in die Warteschlange und spielt sie in Soundkanal 2 ab. Der erste Sound, Sounddarsteller `introMusic`, wird zwischen Sekunde 8 und Sekunde 8,9 fünf Mal wiederholt. Der zweite Sound, Sounddarsteller `creditsMusic`, wird drei Mal abgespielt. Da jedoch keine Werte für `#loopStartTime` und `#loopEndTime` angegeben sind, werden standardmäßig die Werte für `#startTime` und `#endTime` verwendet.

```
-- Lingo syntax
on playMusic
    sound(2).queue([#member:member("introMusic"), #startTime:3000, #loopCount:5,
#loopStartTime:8000, #loopEndTime:8900])
    sound(2).queue([#member:member("creditsMusic"), #startTime:3000, #endTime:3000,
#loopCount:3])
    sound(2).play()
end playMusic

// JavaScript syntax
function playMusic() {
    sound(2).queue(propList("member",member("introMusic"), "startTime",3000,"loopCount",5,
"loopStartTime",8000, "loopEndTime",8900));
    sound(2).queue(propList("member",member("creditsMusic"),
"startTime",3000,"endTime",3000, "loopCount",3]);
    sound(2).play();
}
```

Siehe auch

`loopEndTime (Soundkanal)`, `loopStartTime`, `queue()`, `setPlayList()`, `Sound Channel`

loopCount (Soundobjekt)

Syntax

```
soundObject.loopCount
```

Beschreibung

Diese Soundobjekteigenschaft gibt die Häufigkeit wieder, mit der das Soundobjekt noch wiedergegeben wird.

Beispiele

```
soundObject.loopCount = 8
--Lingo syntax
on mouseUp me
    put soundObjRef.loopcount -- Displays the number of times the sound object
-- associated with soundobjectRef plays.
end

//JavaScript syntax
function mouseUp(){
put (soundObjRef.loopCount) ; // Displays the number of times the sound object
// associated with soundobjectRef plays.
}
```

loopEndTime (Soundkanal)

Syntax

```
-- Lingo syntax
soundChannelObjRef.loopEndTime

// JavaScript syntax
soundChannelObjRef.loopEndTime;
```

Beschreibung

Diese Soundkanaleigenschaft gibt die Endzeit der laufenden Schleife im aktuellen Sound in einem Soundkanal in Millisekunden an. Nur Lesen.

Der Wert dieser Eigenschaft ist eine Fließkommazahl, wodurch sich die Dauer der Soundwiedergabe auf den Bruchteil einer Millisekunde genau bestimmen lässt.

Diese Eigenschaft kann nur eingestellt werden, wenn sie als Eigenschaft im Befehl `queue()` oder `setPlaylist()` übergeben wird.

Beispiel

Die folgende Prozedur spielt den Sounddarsteller `introMusic` in Soundkanal 2 ab. Die Wiedergabe wird zwischen Sekunde 8 und Sekunde 8,9 fünf Mal wiederholt.

```
-- Lingo syntax
on playMusic
    sound(2).play([#member:member("introMusic"), #startTime:3000, #loopCount:5 \
#loopStartTime:8000, #loopEndTime:8900])
end playMusic

// JavaScript syntax
function playMusic() {
    sound(2).play(propList("member",member("introMusic"), "startTime",3000,"loopCount",5,
"loopStartTime",8000, "loopEndTime",8900));
}
```

Siehe auch

queue(), setPlayList(), Sound Channel

loopEndTime (Soundobjekt)

Syntax

```
soundObject.loopEndTime
```

Beschreibung

Diese Soundobjekteigenschaft legt das Ende des wiederholten Bereichs des Soundobjekts fest. `loopEndTime` wird in Millisekunden vom Anfang des aktuellen Soundobjekts festgelegt.

Falls die Anfangszeit `loopStartTime` 100 beträgt, die Endzeit `loopEndTime` 200 und der Zähler `loopCount` 3, dann wird der Teil des Soundobjekts zwischen 100 Millisekunden und 200 Millisekunden nach dem Start dreimal abgespielt.

Beispiele

```
soundObject.loopEndTime = 46890
--Lingo syntax
on mouseUp me
    put soundObjRef.loopEndTime -- Displays the end time for the sound portion
-- that loops.
end

// JavaScript syntax
function mouseUp(){
put (soundObjRef.loopEndTime) ; // Displays the end time for the sound portion
// that loops.
}
```

Siehe auch

loopStartTime (Soundobjekt)

loopsRemaining

Syntax

```
-- Lingo syntax
soundChannelObjRef.loopsRemaining

// JavaScript syntax
soundChannelObjRef.loopsRemaining;
```

Beschreibung

Diese Soundkanaleigenschaft gibt an, wie oft die laufende Schleife im aktuellen Sound in einem Soundkanal noch wiederholt wird. Nur Lesen.

Wenn dem Sound beim Einreihen in die Warteschlange keine Schleife zugewiesen wurde, ist diese Eigenschaft 0. Wenn diese Eigenschaft unmittelbar nach Beginn der Soundwiedergabe getestet wird, gibt sie einen Wert zurück, der um 1 niedriger ist als die mit der Eigenschaft `#loopCount` in der Methode `queue()` oder `setPlayList()` festgelegte Anzahl von Schleifen.

Ausnahmen

Das folgende Beispiel zeigt die im Soundkanal 2 verbleibenden Schleifen an.

```
-- Lingo syntax
put sound(2).loopsRemaining

// JavaScript syntax
put(sound(2).loopsRemaining);
```

Siehe auch

`loopCount`, `queue()`, `setPlayList()`, `Sound Channel`

loopsRemaining (Soundobjekt)

Syntax

```
soundObject.loopsRemaining
```

Beschreibung

Gibt die Anzahl anhängiger Wiedergabeschleifen zurück, die vom Soundobjekt wiedergegeben werden.

Beispiele

```
--Lingo syntax
on mouseUp me
    put soundObjRef.loopsRemaining -- Displays the number of remaining playback loops.
end

// JavaScript syntax
function mouseUp(){
put (soundObjRef.loopsRemaining) ; // Displays the number of remaining playback loops.
}
```

loopStartTime

Syntax

```
-- Lingo syntax
soundChannelObjRef.loopStartTime

// JavaScript syntax
soundChannelObjRef.loopStartTime;
```

Beschreibung

Diese Soundkanaleigenschaft gibt die Startzeit der Schleife für den aktuellen Sound, der in einem Soundkanal abgespielt wird, in Millisekunden an. Nur Lesen.

Da der Wert dieser Eigenschaft eine Fließkommazahl ist, lässt sich die Dauer der Soundwiedergabe auf den Bruchteil einer Millisekunde genau bestimmen. Der Standardwert ist die `startTime` des Sounds, wenn keine Schleife definiert wurde.

Diese Eigenschaft kann nur eingestellt werden, wenn sie als Eigenschaft in der Methode `queue()` oder `setPlaylist()` übergeben wird.

Beispiel

Die folgende Prozedur spielt den Sounddarsteller `introMusic` in Soundkanal 2 ab. Die Wiedergabe wird zwischen Sekunde 8 und Sekunde 8,9 fünf Mal wiederholt.

```
-- Lingo syntax
on playMusic
    sound(2).play([#member:member("introMusic"), #startTime:3000, #loopCount:5
\#loopStartTime:8000, #loopEndTime:8900])
end playMusic

// JavaScript syntax
function playMusic() {
    sound(2).play(propList("member",member("introMusic"), "startTime",3000,"loopCount",5,
"loopStartTime",8000, "loopEndTime",8900));
}
```

Siehe auch

`queue()`, `setPlayList()`, `Sound Channel`, `startTime (Sound Channel)`

loopStartTime (Soundobjekt)

Syntax

```
soundObject.loopStartTime
```

Beschreibung

Diese Soundobjekteigenschaft legt den Beginn des wiederholten Bereichs des Soundobjekts fest. `loopStartTime` wird in Millisekunden vom Anfang des aktuellen Soundobjekts festgelegt.

Beispiele

```
soundObject.loopStartTime = 6890
--Lingo syntax
on mouseUp me
    put soundObjRef.loopStartTime -- Displays the loop start time for the sound object
-- associated with soundobjectRef.
end

// JavaScript syntax
function mouseUp(){
put (soundObjRef.loopStartTime) ; // Displays the loop start time for the sound object
// associated with soundobjectRef.

}
```

Siehe auch

loopEndTime (Soundobjekt)

magnitude

Syntax

```
whichVector.magnitude
```

Beschreibung

Diese 3D-Eigenschaft gibt den Betrag eines Vektors zurück. Der Wert ist eine Fließkommazahl. Der Vektorbetrag ist die Länge eines Vektors und immer größer oder gleich 0.0. (`vector (0, 0, 0)` entspricht 0.)

Beispiel

Die folgende Anweisung zeigt an, dass „magnitude“ von `MyVec1` 100 ist.

```
-- Lingo syntax
MyVec1 = vector(100, 0, 0)
put MyVec1.magnitude

// JavaScript syntax
MyVec1 = vector(100, 0, 0);
put(MyVec1.magnitude);
```

Siehe auch

length (3D), identity()

margin

Syntax

```
-- Lingo syntax
memberObjRef.margin

// JavaScript syntax
memberObjRef.margin;
```

Beschreibung
Diese Felddarstellereigenschaft legt die Größe des Randes innerhalb des Feldrahmens in Pixeln fest.

Beispiel
Die folgende Anweisung setzt den Rand innerhalb des Rahmens für den Felddarsteller „Aktuelle Nachrichten“ auf 15 Pixel:

```
--Lingo syntax
member("Today's News").margin = 15

// JavaScript syntax
member("Today's News").margin = 15;
```

markerList

Syntax

```
-- Lingo syntax
_movie.markerList

// JavaScript syntax
_movie.markerList;
```

Beschreibung
Diese Filmeigenschaft enthält eine Skripteigenschaftsliste der Markierungen im Drehbuch. Nur Lesen.

Die Liste weist das folgende Format auf:

```
frameNumber: "markerName"
```

Beispiel
Die folgende Anweisung zeigt die Liste der Markierungen im Nachrichtenfenster an:

```
-- Lingo syntax
put(_movie.markerList)

// JavaScript syntax
put(_movie.markerList);
```

Siehe auch
Movie

mask

Syntax

```
-- Lingo syntax
memberObjRef.mask

// JavaScript syntax
memberObjRef.mask;
```

Beschreibung

Diese Darstellereigenschaft gibt einen schwarzweißen (1-Bit-) Darsteller an, der als Maske für direkt auf der Bühne wiedergegebene Medien verwendet werden soll, wobei Medien in den Bereichen erscheinen, in denen die Pixel der Maske schwarz sind. Sie können sich mit der Eigenschaft `mask` die Leistungsvorteile eines direkt auf der Bühne wiedergegebenen Digitalvideos zu Nutze machen, während Sie einen QuickTime-Film in einem nicht rechteckigen Bereich abspielen. Die Eigenschaft `mask` wirkt sich nicht nur auf Darsteller aus, die direkt auf der Bühne wiedergegeben werden.

Director richtet das Registrierungskreuz des Maskendarstellers immer am linken oberen Rand des QuickTime-Film-Sprites aus. Vergessen Sie nicht, das Registrierungskreuz einer Bitmap auf die linke obere Ecke zurückzusetzen, da es standardmäßig auf die Mitte eingestellt wird. Die Einstellung des Registrierungskreuzes des QuickTime-Darstellers in der linken oberen Ecke kann nicht geändert werden. Der Maskendarsteller kann nicht verschoben werden und wird durch die Eigenschaften `center` und `crop` des zugehörigen Darstellers nicht beeinflusst.

Die besten Ergebnisse erzielen Sie, wenn Sie die Eigenschaft „mask“ eines QuickTime-Darstellers in der Ereignisprozedur `onbeginSprite` einstellen, bevor seine Sprites auf der Bühne erscheinen. Wenn Sie die Eigenschaft `mask` einstellen oder ändern, während sich der Darsteller auf der Bühne befindet, kann dies unvorhersehbare Folgen haben (zum Beispiel kann die Maske als ein Standbild des Digitalvideos erscheinen, das beim Aktivieren der Eigenschaft `mask` erstellt wurde).

Die Verwendung von Masken ist eine fortgeschrittene Funktion. Sie müssen deshalb unter Umständen ein wenig experimentieren, um das gewünschte Resultat zu erzielen.

Diese Eigenschaft kann getestet und eingestellt werden. Wenn Sie eine Maske entfernen möchten, setzen Sie die Eigenschaft `mask` auf 0.

Beispiel

Das folgende Bildskript stellt eine Maske für ein QuickTime-Sprite ein, bevor Director mit dem Zeichnen des Bildes beginnt:

```
-- Lingo syntax
on prepareFrame
    member("Peeping Tom").mask = member("Keyhole")
end

// JavaScript syntax
function prepareFrame() {
    member("Peeping Tom").mask = member("Keyhole");
}
```

Siehe auch

`invertMask`

maxInteger

Syntax

```
the maxInteger
```

Beschreibung

Diese Systemeigenschaft gibt die größte Ganzzahl zurück, die vom System unterstützt wird. Auf den meisten PCs ist dies 2.147.483.647 (2 hoch 31 minus1).

Diese Funktion kann nützlich sein, um Begrenzungsvariablen vor einer Schleife zu initialisieren oder um Grenzwerte zu testen.

Wenn Sie Zahlen verwenden möchten, die den Bereich der adressierbaren Ganzzahlen übersteigen, benutzen Sie stattdessen Fließkommazahlen. Diese werden langsamer verarbeitet als Ganzzahlen, unterstützen aber einen größeren Wertebereich.

Beispiel

Die folgende Anweisung erstellt im Nachrichtenfenster eine Tabelle mit dem höchsten Dezimalwert, der durch eine bestimmte Anzahl Binärziffern dargestellt werden kann:

```
on showMaxValues
    b = 31
    v = the maxInteger
    repeat while v > 0
        put b && "-" && v
        b = b-1
        v = v/2
    end repeat
end
```

maxSpeed

Syntax

```
member(whichCastmember).modelResource(whichModelResource).emitter.maxSpeed
```

Beschreibung

Mit dieser 3D-Eigenschaft können Sie bei Modellressourcen vom Typ `#particle` die Höchstgeschwindigkeit für Partikelemissionen ermitteln und festlegen. Die Anfangsgeschwindigkeit eines Partikels wird nach dem Zufallsprinzip gewählt und liegt zwischen den Emittereigenschaften `minSpeed` und `maxSpeed`.

Der Wert ist eine Fließkommazahl und muss größer als 0.0 sein.

Beispiel

Das folgende Beispiel legt die maximale Geschwindigkeit des Partikels auf 15 im Darsteller „3Dobjects“ fest.

```
-- Lingo syntax
member("3Dobjects").modelResource("Particle01").emitter.maxSpeed=15

// JavaScript syntax
member("3Dobjects").getPropRef("modelResource",10).emitter.maxSpeed=15;
```

Siehe auch

`minSpeed`, `emitter`

media

Syntax

```
-- Lingo syntax
memberObjRef.media

// JavaScript syntax
memberObjRef.media;
```

Beschreibung

Diese Darstellereigenschaft identifiziert den angegebenen Darsteller als einen Satz Zahlen. Lesen/Schreiben.

Da die Einstellung dieser Eigenschaft viel Speicher beanspruchen kann, sollte sie am besten nur während der Filmerstellung verwendet werden.

Sie können mit der Darstellereigenschaft `media` den Inhalt eines Darstellers in einen anderen Darsteller kopieren, indem Sie den `media`-Wert des zweiten Darstellers auf den `media`-Wert des ersten Darstellers setzen.

Bei einem Filmschleifendarsteller gibt die Eigenschaft `media` eine Auswahl an Bildern und Kanälen im Drehbuch an.

Um Medien in einem Projektor auszutauschen, ist es effizienter, die Sprite-Eigenschaft `member` zu verwenden.

Beispiel

Die folgende Anweisung kopiert den Inhalt des Darstellers "Sonnenaufgang" in den Darsteller "Morgenröte", indem der media-Wert des Darstellers "Morgenröte" auf den media-Wert des Darstellers "Sonnenaufgang" gesetzt wird:

```
-- Lingo syntax
member("Dawn").media = member("Sunrise").media

// JavaScript syntax
member("Dawn").media = member("Sunrise").media;
```

Siehe auch

`Member`

mediaReady

Syntax

```
-- Lingo syntax
memberObjRef.mediaReady

// JavaScript syntax
memberObjRef.mediaReady;
```

Beschreibung

Diese Darstellereigenschaft bestimmt, ob der Inhalt eines Darstellers, eine Film- oder Besetzungsbibliotheksdatei oder ein verknüpfter Darsteller aus dem Internet heruntergeladen wurde und auf dem lokalen Datenträger verfügbar ist (`TRUE`) oder nicht (`FALSE`). Nur Lesen.

Diese Eigenschaft dient ausschließlich zum Streamen eines Films oder einer Besetzungsbibliotheksdatei. Das Streaming eines Films wird aktiviert, wenn Sie die Filmabspieleigenschaften im Menü „Modifizieren“ auf die Option „Abspielen, während Film geladen wird“ (Standardeinstellung) setzen.

Eine Demonstration der Eigenschaft `mediaReady` sehen Sie im Beispielfilm „Streaming Shockwave“ in der Director-Hilfe.

Beispiel

Die folgende Anweisung ändert Darsteller, wenn der gewünschte Darsteller heruntergeladen wurde und lokal verfügbar ist:

```
-- Lingo syntax
if member("background").mediaReady = TRUE then sprite(2).member = member("background").number
end if

// JavaScript syntax
if (member("background").mediaReady == true) {
    sprite(2).member = member("background").number;
}
```

Siehe auch

Member

mediaStatus (DVD)

Syntax

```
-- Lingo syntax
dvdObjRef.mediaStatus

// JavaScript syntax
dvdObjRef.mediaStatus;
```

Beschreibung

Diese DVD-Eigenschaft gibt ein Symbol zurück, das den aktuellen Zustand des DVD-Players angibt. Nur Lesen.

Folgende Symbole sind möglich:

Symbol	Beschreibung
`#stopped`	Die DVD ist beendet.
`#playing`	Die DVD wird wiedergegeben.
`#paused`	Die DVD ist angehalten.
`#scanning`	Die DVD wird gescannt.
`#uninitialized`	Die DVD ist nicht initialisiert.
`#volumeInvalid`	Die angegebene DVD ist nicht gültig.

Symbol	Beschreibung
`#volumeUknown`	Die DVD ist nicht vorhanden oder es liegt keine Disc im Laufwerk.
`#systemSoftwareMissing`	Die DVD-Decoder sind nicht installiert.
`#systemSoftwareBusy`	Die für die Wiedergabe der DVD erforderliche Systemsoftware wird von einer anderen Anwendung verwendet.

Siehe auch

DVD

mediaStatus (MP4Media/FLV)

Syntax

```
put sprite(1).mediaStatus
put member(1).mediaStatus
```

Beschreibung

MP4Media/FLV-Darsteller oder -Sprite-Eigenschaft, mit der ein Symbol abgerufen wird, dass dem Zustand des MP4Media/FLV-Streams entspricht. Die Eigenschaft kann nur ausgelesen werden und ihr Wert kann sich während der Wiedergabe ändern.

Gültige Werte für diese Eigenschaft:

Wert	Eigenschaft
`#playing`	Gibt an, dass der MP4Media/FLV-Stream gerade abgespielt wird.
`#paused`	Gibt an, dass die Wiedergabe unterbrochen wurde, möglicherweise, weil der Benutzer im MP4Media-Player auf die Schaltfläche „Pause“ geklickt hat oder weil die `pause()`-Methode von einem Skript aufgerufen wurde.
`#stopped`	Gibt an, dass der Stream durch das Klicken der Schaltfläche `Stop` angehalten wurde, oder weil die Methode `stop()` von einem Skript aufgerufen wurde.

Beispiele

Folgendes Beispiel zeigt, dass der MP4Media/FLV-Darsteller in „Sprite 1“ abgespielt wird.

```
-- Lingo syntax
put(sprite(1).mediaStatus)  -- #playing
put(member(1).mediaStatus) -- #playing

// JavaScript syntax
put(sprite(1).mediaStatus);  // #playing
put(member(1).mediaStatus); // #playing
```

mediaStatus (RealMedia, Windows Media)

Syntax

```
-- Lingo syntax
memberOrSpriteObjRef.mediaStatus

// JavaScript syntax
memberOrSpriteObjRef.mediaStatus;
```

Beschreibung

Mit dieser Eigenschaft für `RealMedia`- und Windows Media-Darsteller und -Sprites können Sie ein Symbol abrufen, das den Zustand des `RealMedia`- bzw. Windows Media-Streams angibt. Nur Lesen.

Der Wert der Eigenschaft kann sich während der Wiedergabe ändern.

Gültige Werte für diese Eigenschaft:

- `#closed` gibt an, dass der `RealMedia`- oder Windows Media-Darsteller nicht aktiv ist. Der Wert von `mediaStatus` bleibt `#closed`, bis die Wiedergabe beginnt.
- `#connecting` bedeutet, dass eine Verbindung zum `RealMedia`- oder Windows Media-Stream hergestellt wird.
- `#opened` bedeutet, dass eine Verbindung zum `RealMedia`- oder Windows Media-Stream hergestellt wurde und geöffnet ist. Dies ist ein vorübergehender Zustand, auf den sehr schnell `#buffering` folgt.
- `#buffering` gibt an, dass der `RealMedia`- oder Windows Media-Stream in den Wiedergabepuffer heruntergeladen wird. Nach Abschluss der Pufferung (`percentBuffered` = 100) beginnt die Wiedergabe des Streams, sofern die Eigenschaft `pausedAtStart` auf `FALSE` gesetzt ist. Weitere Informationen hierzu finden Sie unter `percentBuffered`.
- `#playing` gibt an, dass der `RealMedia`- oder Windows Media-Stream gegenwärtig abgespielt wird.
- `#seeking` gibt an, dass die Wiedergabe durch den Befehl `seek` unterbrochen wurde.
- `#paused` gibt an, dass die Wiedergabe unterbrochen wurde, möglicherweise, weil der Benutzer im `RealMedia`- oder Windows Media-Viewer auf die Schaltfläche „Stop“ geklickt hat oder weil die Methode `pause()` von einem Skript aufgerufen wurde.
- `#error` gibt an, dass der Stream aus irgendeinem Grund nicht verbunden, gepuffert oder abgespielt werden konnte. Der genaue Fehler lässt sich mit der Eigenschaft `lastError` ermitteln.

Je nach Wert `state (RealMedia)` des Darstellers wird ein anderer Darstellerwert `mediaStatus` zurückgegeben. Jeder `mediaStatus`-Wert entspricht nur jeweils einem `state`-Wert.

Beispiel

Die folgenden Beispiele zeigen, dass der `RealMedia`-Stream in „Sprite 1“ und im Darsteller „Real“ gegenwärtig abgespielt wird:

```
-- Lingo syntax
put(sprite(1).mediaStatus)
put(member("RealDemo").mediaStatus)

// JavaScript syntax
put(sprite(1).mediaStatus);
put(member("RealDemo").mediaStatus);
```

Siehe auch

state (RealMedia), percentBuffered, lastError

mediaXtraList

Syntax

```
-- Lingo syntax
_player.mediaXtraList

// JavaScript syntax
_player.mediaXtraList;
```

Beschreibung

Diese Player-Eigenschaft gibt eine lineare Liste aller im Director-Player verfügbaren Xtra-Medienerweiterungen zurück. Nur Lesen.

Beispiel

Die folgende Anweisung zeigt alle für den Director-Player verfügbaren Xtra-Medienerweiterungen im Nachrichtenfenster an.

```
-- Lingo syntax
put(_player.mediaXtraList)

// JavaScript syntax
put(_player.mediaXtraList);
```

Siehe auch

Medientypen, Player, scriptingXtraList, toolXtraList, transitionXtraList, xtraList (Player)

member

Syntax

```
member(whichCastmember).texture(whichTexture).member
member(whichCastmember).model(whichModel).shader.texture.member
member(whichCastmember).model(whichModel).shaderList[shaderListIndex].textureList[textureLis
tIndex].member
```

Beschreibung

Diese 3D-Textureigenschaft gibt bei Texturen vom Typ `#fromCastMember` den Darsteller an, der als Texturquelle verwendet wird.

Diese Eigenschaft kann getestet und eingestellt werden.

Wenn der Texturtyp `#importedFromFile` lautet, besitzt diese Eigenschaft den Wert „void“ und kann nicht eingestellt werden. Wenn der Texturtyp `#fromImageObject` lautet, besitzt diese Eigenschaft zwar ebenfalls den Wert `void`, kann aber eingestellt werden.

Beispiel

Der folgende Lingo-Code fügt eine neue Textur hinzu. Die zweite Anweisung zeigt, dass zur Erstellung der Textur `gbTexture` Darsteller 16 in Besetzung 1 verwendet wurde.

```
member("scene").newTexture("gbTexture", #fromCastmember, member(16, 1))
put member("scene").texture("gbTexture").member
-- (member 16 of castLib 1)
```

member (Besetzung)

Syntax

```
-- Lingo syntax
castObjRef.member[memberNameOrNum]

// JavaScript syntax
castObjRef.member[memberNameOrNum]
```

Beschreibung

Diese Besetzungsbibliothekseigenschaft bietet indizierten oder benannten Zugriff auf die Darsteller einer Besetzungsbibliothek. Schreibgeschützt

Das Argument *memberNameOrNum* kann eine Zeichenfolge sein, die den Darsteller anhand des Namens identifiziert, oder eine Ganzzahl, die den Darsteller anhand der Nummer angibt.

Beispiel

Das folgende Beispiel bietet Zugriff auf den zweiten Darsteller in der Besetzungsbibliothek namens „Internal“.

```
-- Lingo syntax
myMember = castLib("Internal").member[2]

// JavaScript syntax
var myMember = castLib("Internal").member[2];
```

Siehe auch

Cast Library

member (Film)

Syntax

```
-- Lingo syntax
_movie.member[memberNameOrNum]

// JavaScript syntax
_movie.member[memberNameOrNum];
```

Beschreibung

Diese Filmeigenschaft bietet indizierten oder benannten Zugriff auf die Darsteller der Besetzungsbibliothek eines Films. Schreibgeschützt

Das Argument *memberNameOrNum* kann eine Zeichenfolge sein, die den Darsteller anhand des Namens identifiziert, oder eine Ganzzahl, die den Darsteller anhand der Nummer angibt.

Beispiel

Die folgende Anweisung greift unter Verwendung von benanntem als auch indiziertem Zugriff auf einen Darsteller zu und legt die Variable `myMember` auf das Ergebnis fest.

```
-- Lingo syntax
myMember = _movie.member[2] -- using numbered access
myMember = _movie.member["Athlete"] -- using named access

// JavaScript syntax
var myMember = _movie.member[2]; // using numbered access;
var myMember = _movie.member["Athlete"]; // using named access;
```

Siehe auch

`Movie`

member (Soundkanal)

Syntax

```
-- Lingo syntax
soundChannelObjRef.member

// JavaScript syntax
soundChannelObjRef.member;
```

Beschreibung

Diese Soundkanaleigenschaft gibt den Sounddarsteller an, der gegenwärtig in einem Soundkanal abgespielt wird. Nur Lesen.

Diese Eigenschaft gibt `VOID` (Lingo) oder `null` (JavaScript-Syntax) zurück, wenn kein Sound wiedergegeben wird.

Beispiel

Die folgende Anweisung zeigt den Namen des Darstellers des in Soundkanal 2 abgespielten Sounds im Nachrichtenfenster an:

```
-- Lingo syntax
put(sound(2).member)

// JavaScript syntax
put(sound(2).member);
```

Siehe auch

`Sound Channel`

member (Soundobjekt)

Syntax

```
soundObj.member (Read-only)
```

Beschreibung

Diese Soundobjekteigenschaft gibt eine Referenz auf den Darsteller des aktuellen Soundobjekts zurück. Wenn das Soundobjekt über einen Dateipfad erstellt wurde, zeigt der Darsteller auf NULL.

Beispiel

```
-- Lingo syntax
on mouseUp me
    put soundObjRef.  -- Displays the reference of the cast member of the sound object
-- associated with soundobjectRef.
end

// JavaScript syntax
function mouseUp(){
put (soundObjRef.mixer) ; // Displays the reference of the cast member of the sound
// object associated with soundobjectRef.

}
```

member (Sprite)

Syntax

```
-- Lingo syntax
spriteObjRef.member

// JavaScript syntax
spriteObjRef.member;
```

Beschreibung

Diese Sprite-Eigenschaft gibt den Darsteller eines Sprites und dessen Besetzungsbibliothek an. Lesen/Schreiben.

Die Sprite-Eigenschaft `member` unterscheidet sich von der Sprite-Eigenschaft `spriteNum`, die nur die Nummer des Sprites zur Angabe seiner Position in der Besetzungsbibliothek, aber nicht die Besetzungsbibliothek selbst angibt. Die Eigenschaft `member` unterscheidet sich auch von der Eigenschaft `mouseMember` des Mouse-Objekts, die keine Besetzungsbibliothek eines Sprites angibt.

Wenn Sie die Eigenschaft `member` eines Sprites zuordnen, verwenden Sie eines der folgenden Formate:

- Geben Sie die vollständige Darsteller- und Besetzungsbibliotheksbeschreibung an (*spriteObjRef*`.member = member(`*intMemberNum* `{, ` *castLibraryNameOrNum}*`)`).
- Geben Sie den Namen des Darstellers an (*spriteObjRef*`.member = member("`*stringMemberName*"`)`).
- Geben Sie die eindeutige Ganzzahl an, die alle Besetzungsbibliotheken einschließt und der Eigenschaft `mouseMember` entspricht (*spriteObjRef*`.member = 132`).

Falls Sie nur den Darstellernamen verwenden, sucht Director in allen aktuellen Besetzungsbibliotheken den ersten Darsteller mit diesem Namen. Sollte dieser Name in zwei Besetzungsbibliotheken vorkommen, wird nur der erste gefundene Darsteller verwendet.

Wenn Sie einen Darsteller anhand seiner Nummer angeben möchten und mehrere Besetzungen vorhanden sind, verwenden Sie die Sprite-Eigenschaft `memberNum`, die die Position des Darstellers in seiner Besetzungsbibliothek ohne Auswirkung auf die Besetzungsbibliothek des Sprites ändert (*spriteObjRef*`.memberNum = 10`).

Der einem Sprite-Kanal zugeordnete Darsteller ist nur eine der Eigenschaften dieses Sprites. Andere Eigenschaften variieren je nach Typ des Medienelements in diesem Kanal im Drehbuch. Wenn Sie zum Beispiel eine Bitmap anhand der Sprite-Eigenschaft `member` durch eine nicht gefüllte Form ersetzen, wird die Eigenschaft `lineSize` des Form-Sprites nicht automatisch geändert und Sie werden die Form wahrscheinlich nicht sehen.

Ähnliche Unstimmigkeiten bei Sprite-Eigenschaften können auftreten, wenn Sie den Darsteller eines Feld-Sprites in ein Video ändern. Im Allgemeinen ist es sinnvoller und auch vorhersagbarer, Darsteller durch ähnliche Darsteller zu ersetzen. Tauschen Sie zum Beispiel Bitmap-Sprites gegen Bitmapdarsteller aus.

Beispiel

Die folgende Anweisung ordnet Darsteller 3 in Besetzung Nummer 4 dem Sprite 15 zu:

```
-- Lingo syntax
sprite(15).member = member(3, 4)

// JavaScript syntax
sprite(15).member = member(3, 4);
```

Die folgende Prozedur verwendet die Funktion `mouseMember` in Verbindung mit der Sprite-Eigenschaft @@@member, um festzustellen, ob sich der Mauszeiger über einem bestimmten Sprite befindet:

```
-- Lingo syntax
on exitFrame
    mm = _mouse.mouseMember
    target = sprite(1).member
    if (target = mm) then
        put("Above the hotspot.")
        _movie.go(_movie.frame)
    end if
end

// JavaScript syntax
function exitFrame() {
    var mm = _mouse.mouseMember;
    var target = sprite(1).member;
    if (target == mm) {
        put("Above the hotspot.");
        _movie.go(_movie.frame);
    }
}
```

Siehe auch

lineSize, mouseMember, Sprite, spriteNum

memorySize

Syntax

```
the memorySize
```

Beschreibung

Diese Systemeigenschaft gibt den Speicherplatz zurück, der dem Programm insgesamt zugewiesen wurde, wobei es keine Rolle spielt, ob er belegt oder frei ist. Diese Eigenschaft dient zum Überprüfen der Mindestspeicheranforderungen. Der Wert wird in Byteangegeben.

Unter Windows ist dieser Wert der gesamte verfügbare physikalische Speicher, und auf dem Mac ist dieser Wert die gesamte Partition, die der Anwendung zugeordnet ist.

Beispiel

Die folgende Anweisung prüft, ob mehr als 500 KB Speicherplatz zugewiesen ist, und blendet in diesem Fall einen Warnhinweis ein.

```
-- Lingo syntax
if the memorySize > 500 * 1024 then alert "There is enough memory to run this movie."

// JavaScript syntax
if ( _system.memorySize > 500 * 1024)
{
    _player.alert( "There is enough memory to run this movie.");
}
```

Siehe auch

`freeBlock()`, `freeBytes()`, `ramNeeded()`, `size`

meshDeform (Modifizierer)

Syntax

```
member(whichCastmember).model(whichModel).meshDeform.propertyName
```

Beschreibung

Mit diesem 3D-Modifizierer können Sie die verschiedenen Aspekte der Gitternetzstruktur des referenzierten Modells steuern. Nachdem Sie den Modifizierer `#meshDeform` mit dem Befehl `addModifier` zu einem Modell hinzugefügt haben, können Sie auf die folgenden Eigenschaften des `#meshDeform`-Modifizierers zugreifen:

Hinweis: *Weitere Informationen über die folgenden Eigenschaften enthalten die einzelnen Einträge (siehe Abschnitt "Siehe auch").*

- `face.count` gibt die Gesamtzahl von Seiten im referenzierten Modell zurück.
- `mesh.count` gibt die Anzahl von Gitternetzen im referenzierten Modell zurück.
- `mesh[index]` ermöglicht den Zugriff auf die Eigenschaften des angegebenen Gitternetzes.

Beispiel

Die folgende Anweisung zeigt die Anzahl von Flächen im Modell `gbFace` an:

```
put member("3D World").model("gbFace").meshDeform.face.count
-- 432
```

Die folgende Anweisung zeigt die Anzahl von Gitternetzen im Modell `gbFace` an:

```
put member("3D World").model("gbFace").meshDeform.mesh.count
-- 2
```

Die folgende Anweisung zeigt die Anzahl von Flächen im zweiten Gitternetz des Modells `gbFace` an:

```
put member("3D World").model("gbFace").meshDeform.mesh[2].face.count
-- 204
```

Siehe auch

mesh (Eigenschaft), addModifier

milliseconds

Syntax

```
-- Lingo syntax
_system.milliseconds

// JavaScript syntax
_system.milliseconds;
```

Beschreibung

Diese Systemeigenschaft gibt die aktuelle Zeit in Millisekunden (ein Tausendstel einer Sekunde) zurück. Nur Lesen.

Es wird vom Einschalten des Computers an gezählt.

Beispiel

Die folgende Anweisung konvertiert Millisekunden in Sekunden und Minuten, indem die Anzahl an Millisekunden durch 1000 und das Ergebnis durch 60 geteilt wird, und setzt dann die Variable `currentMinutes` auf das Ergebnis:

```
-- Lingo syntax
currentSeconds = _system.milliseconds/1000
currentMinutes = currentSeconds/60

// JavaScript syntax
var currentSeconds = _system.milliseconds/1000;
var currentMinutes = currentSeconds/60;
```

Die Auflösungsgenauigkeit des Zählers hängt vom Computer und vom Betriebssystem ab.

Die folgende Prozedur zählt die Millisekunden und sendet einen Warnhinweis, wenn Sie zu lange gearbeitet haben.

```
-- Lingo syntax
on idle
    if (_system.milliseconds > (1000 * 60 * 60 * 4)) then
        _player.alert("Take a break")
    end if
end

// JavaScript syntax
function idle() {
    if (_system.milliseconds > (1000 * 60 * 60 * 4)) {
        _player.alert("Take a break");
    }
}
```

Siehe auch

System

minSpeed

Syntax

```
member(whichCastmember).modelResource(whichModelResource).emitter.minSpeed
```

Beschreibung

Bei Modellressourcen vom Typ `#particle` können Sie mit dieser 3D-Eigenschaft die Mindestgeschwindigkeit für Partikelemissionen ermitteln und festlegen. Die Anfangsgeschwindigkeit eines Partikels wird nach dem Zufallsprinzip gewählt und liegt zwischen den Emittereigenschaften `minSpeed` und `maxSpeed`.

Der Wert ist eine Fließkommazahl und muss größer als 0.0 sein.

Beispiel

Das folgende Beispiel legt die minimale Geschwindigkeit des Partikels auf 4 im Darsteller „3Dobjects“ fest.

```
-- Lingo syntax
member("3Dobjects").modelResource("Particle01").emitter.minSpeed=4

// JavaScript syntax
member("3Dobjects").getPropRef("modelResource",10).emitter.minSpeed=4;
```

Siehe auch

maxSpeed, emitter

missingFonts

Syntax

```
member(textCastMember).missingFonts
```

Beschreibung
Diese Textdarstellereigenschaft enthält eine Liste mit den Namen der Schriften, auf die im Text verwiesen wird, die aber gegenwärtig im System nicht zur Verfügung stehen.

Auf diese Weise kann der Entwickler während der Laufzeit feststellen, ob eine bestimmte Schrift verfügbar ist.

Diese Eigenschaft kann getestet, aber nicht eingestellt werden.

Beispiel
Das folgende Beispiel zeigt eine Liste mit den Namen der Schriften an, auf die im Text verwiesen wird, die aber gegenwärtig im System nicht zur Verfügung stehen.

```
-- Lingo syntax
put member(4).missingFonts

// JavaScript syntax
put(member(4).missingFonts);
```

Siehe auch
substituteFont

mixer

Syntax
```
soundObject.mixer (Read-only)
```

Beschreibung
Diese Soundobjekteigenschaft gibt den Mixer des Soundobjekts zurück.

Beispiele
```
--Lingo syntax
on mouseUp me
    put soundObjRef.mixer -- Displays the sound object's mixer.
end

//JavaScript syntax
function mouseUp(){
put (soundObjRef.mixer) ; // Displays the sound object's mixer.
}
```

mode (emitter)

Syntax
```
member(whichCastmember).modelResource(whichModelResource).emitter.mode
```

Beschreibung
Mit dieser 3D-Eigenschaft können Sie bei Modellressourcen vom Typ `#particle` die Eigenschaft `mode` des Partikelemitters der Ressource ermitteln und festlegen.

Mögliche Werte: `#burst` und `#stream` (Standard). Wenn `mode` den Wert `#burst` aufweist, werden alle Partikel zum gleichen Zeitpunkt ausgestrahlt. Beim Wert `#stream` wird in jedem Bild eine Gruppe von Partikeln emittiert. Die in den einzelnen Bildern emittierte Anzahl von Partikeln richtet sich nach der folgenden Gleichung:

```
particlesPerFrame = resourceObject.emitter.numParticles (resourceObject.lifetime x
millisecondsPerRenderedFrame)
```

Beispiel

In diesem Beispiel ist `ThermoSystem` eine Modellressource vom Typ `#particle`. Die folgende Anweisung setzt die Eigenschaft `emitter.mode` von `ThermoSystem` auf `#burst`, sodass die Partikel von `ThermoSystem` stoßweise erscheinen. Um einen einzigen Partikelstoß zu erzeugen, verwenden Sie `emitter.mode = #burst` und`emitter.loop`= 0.

```
member("Fires").modelResource("ThermoSystem").emitter.mode = #burst
```

Siehe auch

`emitter`

mode (collision)

Syntax

```
member(whichCastmember).model(whichModel).collision.mode
```

Beschreibung

Diese 3D-Eigenschaft für den Modifizierer „collision“ gibt die Geometrie an, die im Kollisionserkennungsalgorithmus verwendet werden soll. Einfache Geometrien wie die Begrenzungskugel führen zu besserer Leistung. Mögliche Werte für diese Eigenschaft:

- `#mesh` verwendet die eigentliche Gitternetzgeometrie der Modellressource. Dieser Wert hat eine Genauigkeit von einem Dreieck und ist in der Regel langsamer als `#box` oder `#sphere`.
- `#box` verwendet das Begrenzungsrechteck des Modells. Diese Einstellung bietet sich für Objekte an, die besser in einen Quader passen als in eine Kugel, wie z. B. eine Wand.
- `#sphere` ist der schnellste Modus, da hierbei die Begrenzungskugel des Modells verwendet wird. Dies ist die Standardeinstellung für diese Eigenschaft.

Beispiel

Die folgenden Anweisungen fügen den Modifizierer "collision" zum Modell deinModell hinzu und setzen die Eigenschaft mode auf `#mesh`:

```
member("3d").model("yourModel").addModifier(#collision)
member("3d").model("yourModel").collision.mode = #mesh
```

model

Syntax

```
member(whichCastmember).model(whichModel)
member(whichCastmember).model[index]
member(whichCastmember).model.count
member(whichCastmember).model(whichModel).propertyName
member(whichCastmember).model[index].propertyName
```

Beschreibung

Dieser 3D-Befehl gibt das im referenzierten Darsteller gefundene Modell zurück, das den im Parameter *whichModel* angegebenen Namen besitzt bzw. das sich an der durch *index* angegebenen Indexposition befindet. Wenn für den angegebenen Parameter kein Modell gefunden wird, gibt der Befehl `void` zurück. In der Form `model.count` gibt der Befehl die im referenzierten Darsteller gefundene Anzahl von Modellen zurück. Über diesen Befehl können Sie außerdem auf die Eigenschaften des jeweiligen Modells zugreifen.

Beim Vergleich von Modellnamen spielt die Groß-/Kleinschreibung keine Rolle. Die Indexposition eines bestimmten Modells kann sich ändern, wenn Objekte an niedrigeren Indexpositionen gelöscht werden.

Wenn kein Modell mit dem angegebenen Namen bzw. an der angegebenen Indexposition gefunden wird, gibt dieser Befehl „void“ zurück.

Beispiel

Die folgende Anweisung speichert einen Verweis auf das Modell "Spieleravatar" in der Variablen diesesModell:

```
thismodel = member("3DWorld").model("Player Avatar")
```

Die folgende Anweisung speichert einen Bezug auf das achte Modell des Darstellers „3DWelt“ in der Variablen diesesModell:

```
thismodel = member("3DWorld").model[8]
```

Die folgende Anweisung zeigt, dass es im Darsteller von Sprite 1 vier Modelle gibt.

```
put sprite(1).member.model.count
-- 4
```

modelA

Syntax

```
collisionData.modelA
```

Beschreibung

Diese 3D-Eigenschaft für `collisionData` gibt eines der an einer Kollision beteiligten Modelle an; das andere Modell ist `modelB`.

Das `collisionData`-Objekt wird bei den Ereignissen `#collideWith` und `#collideAny` als Argument an die in den Befehlen `registerForEvent, registerScript` und `setCollisionCallback` angegebene Prozedur übergeben.

Die Ereignisse `#collideWith` und `#collideAny` werden gesendet, wenn eine Kollision zwischen Modellen stattfindet, an denen der Modifizierer "collision" angebracht ist. Die Eigenschaft `resolve` der Modifizierer des Modells muss auf `TRUE` gesetzt sein.

Diese Eigenschaft kann getestet, aber nicht eingestellt werden.

Beispiel

Dieses Beispiel setzt sich aus drei Teilen zusammen. Der erste Teil ist die erste Codezeile, die die Prozedur `#putDetails` für das Ereignis `#collideAny` registriert. Der zweite Teil ist die Prozedur `#putDetails`. Wenn zwei Modelle im Darsteller `MyScene` zusammenstoßen, wird die Prozedur `#putDetails` aufgerufen und das Argument `collisionData` an sie übergeben. Die folgende Prozedur zeigt die Eigenschaften modelA und modelB des Objekts `collisionData` im Nachrichtenfenster an. Der dritte Teil des Beispiels zeigt die Ergebnisse aus dem Nachrichtenfenster an. Daraus geht hervor, dass bei der Kollision das Modell `GreenBall` „modelA“ und das Modell `YellowBall` „modelB“ war.

```
member("MyScene").registerForEvent(#collideAny, #putDetails, 0)
on putDetails me, collisionData
    put collisionData.modelA
    put collisionData.modelB
end
-- model("GreenBall")
-- model("YellowBall")
```

Siehe auch

`registerScript()`, `registerForEvent()`, `sendEvent`, `modelB`, `setCollisionCallback()`

modelB

Syntax

```
collisionData.modelB
```

Beschreibung

Diese 3D-Eigenschaft für `collisionData` gibt eines der an einer Kollision beteiligten Modelle an; das andere Modell ist `modelA`.

Das `collisionData`-Objekt wird bei den Ereignissen `#collideWith` und `#collideAny` als Argument an die in den Befehlen `registerForEvent`, `registerScript` und `setCollisionCallback` angegebene Prozedur übergeben.

Die Ereignisse `#collideWith` und `#collideAny` werden gesendet, wenn eine Kollision zwischen Modellen stattfindet, an denen der Modifizierer "collision" angebracht ist. Die Eigenschaft `resolve` der Modifizierer des Modells muss auf `TRUE` gesetzt sein.

Diese Eigenschaft kann getestet, aber nicht eingestellt werden.

Beispiel

Dieses Beispiel setzt sich aus drei Teilen zusammen. Der erste Teil ist die erste Codezeile, die die Prozedur `#putDetails` für das Ereignis `#collideAny` registriert. Der zweite Teil ist die Prozedur `#putDetails`. Wenn zwei Modelle im Darsteller `MyScene` zusammenstoßen, wird die Prozedur `#putDetails` aufgerufen und das Argument `collisionData` an sie übergeben. Die folgende Prozedur zeigt die Eigenschaften modelA und modelB des Objekts `collisionData` im Nachrichtenfenster an. Der dritte Teil des Beispiels zeigt die Ergebnisse aus dem Nachrichtenfenster an. Daraus geht hervor, dass bei der Kollision das Modell `GreenBall` „modelA“ und das Modell `YellowBall` „modelB“ war.

```
member("MyScene").registerForEvent(#collideAny, #putDetails, 0)
on putDetails me, collisionData
    put collisionData.modelA
    put collisionData.modelB
end
-- model("GreenBall")
-- model("YellowBall")
```

Siehe auch

registerScript(), registerForEvent(), sendEvent, modelA, collisionNormal, setCollisionCallback()

modelResource

Syntax

```
member(whichCastmember).modelResource(whichModelResource)
member(whichCastmember).modelResource[index]
member(whichCastmember).modelResource.count
member(whichCastmember).modelResource(whichModelResource).propertyName
member(whichCastmember).modelResource[index].propertyName
```

Beschreibung

Dieser 3D-Befehl gibt die im referenzierten Darsteller gefundene Modellressource zurück, die den im Parameter *whichModelResource* angegebenen Namen besitzt bzw. die sich an der durch *index* angegebenen Indexposition befindet. Wenn für den angegebenen Parameter keine Modellressource gefunden wird, gibt der Befehl `void` zurück. In der Form `modelResource.count` gibt der Befehl die im referenzierten Darsteller gefundene Anzahl von Modellressourcen zurück. Über diesen Befehl können Sie außerdem auf die Eigenschaften des jeweiligen Modells zugreifen.

Beim Vergleich von Modellressourcennamenstrings spielt die Groß-/Kleinschreibung keine Rolle. Die Indexposition einer bestimmten Modellressource kann sich ändern, wenn Objekte an niedrigeren Indexpositionen gelöscht werden.

Beispiel

Die folgende Anweisung zeigt die erste Modellressource des Darstellers „3Dobjects“ an.

```
-- Lingo syntax
putmember("3Dobjects").modelResource[1]

// JavaScript syntax
put( member("3Dobjects").getPropRef("modelResource",1));
```

modified

Syntax

```
-- Lingo syntax
memberObjRef.modified

// JavaScript syntax
memberObjRef.modified;
```

Beschreibung

Diese Darstellereigenschaft zeigt an, ob ein Darsteller seit dem Einlesen aus der Filmdatei modifiziert wurde. Nur Lesen.

- Wenn die Eigenschaft `modified` den Wert `TRUE` (`1`) aufweist, wurde der Darsteller seit dem Einlesen aus der Filmdatei modifiziert.
- Wenn die Eigenschaft `modified` den Wert `FALSE` (`0`) aufweist, wurde der Darsteller seit dem Einlesen aus der Filmdatei nicht modifiziert.

Beispiel

Die folgende Anweisung prüft, ob der Darsteller "Einleitung“ seit dem Einlesen aus der Filmdatei modifiziert wurde:

```
-- Lingo syntax
if (member("Introduction").modified) then
    _player.alert("Introduction has been modified")
else
    _player.alert("Introduction has not been modified")
end if

// JavaScript syntax
if (member("Introduction").modified) {
    _player.alert("Introduction has been modified");
}
else {
    _player.alert("Introduction has not been modified");
}
```

Siehe auch

Member

modifiedBy

Syntax

```
-- Lingo syntax
memberObjRef.modifiedBy

// JavaScript syntax
memberObjRef.modifiedBy;
```

Beschreibung

Diese Darstellereigenschaft zeichnet den Namen des Benutzers auf, von dem der Darsteller zuletzt bearbeitet wurde. Nur Lesen.

Der Name stammt aus den Benutzernameninformationen, die während der Director-Installation bereitgestellt werden. Sie können diese Angaben im Director-Dialogfeld „Allgemeine Voreinstellungen“ ändern.

Diese Eigenschaft dient zum Überwachen und Koordinieren von Director-Projekten, an denen mehrere Autoren beteiligt sind, und können auch auf der Registerkarte „Darsteller“ des Eigenschafteninspektors angezeigt werden.

Beispiel

Die folgende Anweisung zeigt den Namen des Benutzers an, der Darsteller 1 zuletzt bearbeitet hat.

```
-- Lingo syntax
put(member(1).modifiedBy)

// JavaScript syntax
put(member(1).modifiedBy);
```

Siehe auch

Member

modifiedDate

Syntax

```
-- Lingo syntax
memberObjRef.modifiedDate

// JavaScript syntax
memberObjRef.modifiedDate;
```

Beschreibung

Diese Darstellereigenschaft zeigt das Datum und die Uhrzeit gemäß der Systemzeit auf dem Erstellungscomputer an, zu dem/r der Darsteller zuletzt geändert wurde. Nur Lesen.

Diese Eigenschaft dient zur Überwachung und Koordinierung von Director-Projekten. Sie kann auch auf der Registerkarte „Darsteller“ des Eigenschafteninspektor sowie in der Listenansicht des Besetzungsfensters angezeigt werden.

Beispiel

Die folgende Anweisung zeigt das Datum an, an dem Darsteller 1 zuletzt geändert wurde:

```
-- Lingo syntax
put(member(1).modifiedDate)

// JavaScript syntax
put(member(1).modifiedDate);
```

Siehe auch

Member

modifier

Syntax

```
member(whichCastmember).model(whichModel).modifier
member(whichCastmember).model(whichModel).modifier.count
```

Beschreibung

Diese 3D-Modelleigenschaft gibt eine Liste aller an einem bestimmten Modell angebrachten Modifizierer zurück. In der Form `modifier.count` gibt dieser Befehl die Anzahl der am Modell angebrachten Modifizierer zurück.

Wenn sowohl der Modifizierer `toon` als auch der Modifizierer `inker` am Modell angebracht ist, wird nur der erste zum Modell hinzugefügte Modifizierer zurückgegeben.

Diese Eigenschaft kann getestet, aber nicht eingestellt werden. Mit den Befehlen `addModifier` und `removeModifier` können Sie Modifizierer zu Modellen hinzufügen bzw. daraus entfernen.

Beispiel

Die folgende Anweisung zeigt an, welche Modifizierer am Modell „Sphere01“ des Darstellers „3Dobjects“ angebracht sind.

```
-- Lingo syntax
put member("3Dobjects").model("Sphere01").modifier

// JavaScript syntax
put( member("3Dobjects").getPropRef("model",2).modifier);
```

Siehe auch

`modifier[]`, `modifiers`, `addModifier`, `removeModifier`

modifier[]

Syntax

```
member(whichCastmember).model(whichModel).modifier[index]
```

Beschreibung

Diese 3D-Modelleigenschaft gibt den Typ des Modifizierers zurück, der in der Modifiziererliste des Modells an der durch *index* angegebenen Position zu finden ist. Beim Rückgabewert handelt es sich um ein Symbol.

Wenn an der angegebenen Position kein Modifizierer zu finden ist, hat diese Eigenschaft den Wert „void“.

Mit der Eigenschaft `modifier` können Sie Informationen über die Modifiziererliste eines Modells abrufen.

Der direkte Zugriff auf die Eigenschaften eines angebrachten Modifizierers ist mit diesem Befehl jedoch nicht möglich.

Beispiel

```
put member("3d world").model("box").modifier[1]
-- #lod
```

Siehe auch

`modifier`, `modifiers`, `addModifier`, `removeModifier`

modifiers

Syntax

```
getRendererServices().modifiers
```

Beschreibung
Diese globale 3D-Eigenschaft gibt eine Liste von Modifizierern zurück, die Modellen in 3D-Darstellern zur Verfügung stehen.

Beispiel
Die folgende Anweisung gibt die Liste aller gegenwärtig verfügbaren Modifizierer zurück:

```
-- Lingo syntax
put getRendererServices().modifiers

// JavaScript syntax
put(getRendererServices().modifiers);
```

Siehe auch
`getRendererServices()`, `addModifier`

mostRecentCuePoint

Syntax

```
-- Lingo syntax
spriteObjRef.mostRecentCuePoint

// JavaScript syntax
spriteObjRef.mostRecentCuePoint;
```

Beschreibung
Diese Soundkanal- und Sprite-Eigenschaft gibt für Sound-Sprites, QuickTime-Digitalvideos und Xtra-Erweiterungen, die Aufrufpunkte unterstützen, die Nummer an, die den zuletzt im Sprite oder Sound passierten Aufrufpunkt identifiziert. Der Wert ist die Ordnungszahl des Aufrufpunkts. Falls keine Aufrufpunkte passiert wurden, ist der Wert 0.

Shockwave Audio (SWA)-Sounds können in Sprite-Kanälen als Sprites erscheinen, die Soundwiedergabe erfolgt jedoch in einem Soundkanal. Es wird empfohlen, mit der Nummer des Sprite-Kanals und nicht mit der Nummer des Soundkanals auf SWA-Sound-Sprites Bezug zu nehmen.

Beispiel
Die folgende Anweisung weist das Nachrichtenfenster an, die Nummer des Aufrufpunkts anzuzeigen, der zuletzt im Sprite in Sprite-Kanal 1 passiert wurde:

```
-- Lingo syntax
put sprite(1).mostRecentCuePoint

// JavaScript syntax
put(sprite(1).mostRecentCuePoint);
```

Die folgende Anweisung gibt die Ordnungszahl des zuletzt passierten Aufrufpunkts im laufenden Sound in Soundkanal 2 zurück:

```
-- Lingo syntax
put sound(2).mostRecentCuePoint

// JavaScript syntax
put(sound(2).mostRecentCuePoint);
```

Siehe auch

`cuePointNames`, `isPastCuePoint()`, `cuePointTimes`, `on cuePassed`

mostRecentCuePoint (Soundobjekt)

Syntax

```
soundobj.mostRecentCuePoint (Read-only)
```

Beschreibung

Diese Soundobjekteigenschaft ermittelt den zuletzt wiedergegebenen Cue-Punkt des Soundobjekts.

Beispiele

```
-- Lingo syntax
on exitFrame me
    put soundObjRef.mostRecentCuePoint -- Returns the most recent cue point hit by
-- the sound object.
end

// JavaScript syntax
function exitFrame(){
put (soundObjRef.mostRecentCuePoint) ; // Returns the most recent cue point hit by
// the sound object.

}
```

motion

Syntax

```
member(whichCastmember).motion(whichMotion)
member(whichCastmember).motion[index]
member(whichCastmember).motion.count
```

Beschreibung

Dieser 3D-Befehl gibt die im referenzierten Darsteller gefundene Bewegung zurück, die den im Parameter *whichMotion* angegebenen Namen besitzt bzw. die sich an der durch *index* angegebenen Indexposition befindet. In der Form `motion.count` gibt diese Eigenschaft die Gesamtzahl von Bewegungen im Darsteller zurück.

Beim Vergleich von Objektnamenstrings spielt die Groß-/Kleinschreibung keine Rolle. Die Indexposition einer bestimmten Bewegung kann sich ändern, wenn Objekte an niedrigeren Indexpositionen gelöscht werden.

Wenn keine Bewegung mit dem angegebenen Namen bzw. an der angegebenen Indexposition gefunden wird, gibt dieser Befehl „void“ zurück.

Beispiel

Die folgende Anweisung zeigt die erste im referenzierten Darsteller gefundene Bewegung an.

```
-- Lingo syntax
put member("3Dobjects").motion[1]

// JavaScript syntax
put(member("3Dobjects").getPropRef("motion",1));
```

Siehe auch

duration (3D), map (3D)

motionQuality

Syntax

```
-- Lingo syntax
spriteObjRef.motionQuality

// JavaScript syntax
spriteObjRef.motionQuality;
```

Beschreibung

Diese QuickTime VR-Sprite-Eigenschaft gibt die Codec-Qualität an, die verwendet wird, wenn der Benutzer auf das QuickTime VR-Sprite klickt und es zieht. Mögliche Werte: `#minQuality`, `#maxQuality` und `#normalQuality`.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Die folgende Anweisung setzt die Eigenschaft `motionQuality` von Sprite 1 auf `#minQuality`:

```
-- Lingo syntax
sprite(1).motionQuality = #minQuality

// JavaScript syntax
sprite(1).motionQuality = symbol("minQuality");
```

mouseChar

Syntax

```
-- Lingo syntax
_mouse.mouseChar

// JavaScript syntax
_mouse.mouseChar;
```

Beschreibung

Diese Mauseigenschaft enthält für Feld-Sprites die Nummer des Zeichens, das sich unter dem Mauszeiger befindet, wenn die Eigenschaft aufgerufen wird. Nur Lesen.

Es wird vom Anfang des Feldes an gezählt. Falls sich die Maus nicht über einem Feld oder im Rahmenabstand eines Feldes befindet, ist das Ergebnis -1.

Der Wert von `mouseChar` kann sich in einer Prozedur oder Schleife ändern. Falls eine Eigenschaft mehrmals in einer Prozedur oder Schleife verwendet wird, ist es normalerweise sinnvoll, die Funktion einmal aufzurufen und ihren Wert einer lokalen Variablen zuzuordnen.

Beispiel

Die folgende Anweisung bestimmt, ob sich der Zeiger über einem Feld-Sprite befindet. Wenn nicht, wird der Inhalt des Felddarstellers „Instructions“ in „Please point to a character.“ geändert:

```
-- Lingo syntax
if (_mouse.mouseChar = -1) then
    member("Instructions").text = "Please point to a character."
end if

// JavaScript syntax
if (_mouse.mouseChar == -1) {
    member("Instructions").text = "Please point to a character.";
}
```

Die folgende Anweisung ordnet das Zeichen unter dem Mauszeiger im angegebenen Feld der Variablen `currentChar` zu:

```
-- Lingo syntax
currentChar = member(_mouse.mouseMember).char[_mouse.mouseChar]

// JavaScript syntax
var currentChar = member(_mouse.mouseMember).getProp("char",_mouse.mouseChar);
```

Siehe auch

`Mouse`, `mouseItem`, `mouseLine`

mouseDown

Syntax

```
-- Lingo syntax
_mouse.mouseDown

// JavaScript syntax
_mouse.mouseDown;
```

Beschreibung

Diese Mauseigenschaft zeigt an, ob die Maustaste momentan gedrückt wird (`TRUE`) oder nicht (`FALSE`). Nur Lesen.

Beispiel

Die folgende `mouseEnter`-Prozedur, die an einem Sprite angebracht ist, ruft eine Prozedur auf, wenn die Maus bei nicht gedrückter Taste in ein Sprite eintritt, und eine andere Prozedur, wenn die Maus bei gedrückter Taste in das Sprite eintritt.

```
-- Lingo syntax
on mouseEnter
    if (_mouse.mouseDown) then
        runMouseDownScript
    else
        runMouseUpScript
    end if
end

// JavaScript syntax
function mouseEnter() {
    if (_mouse.mouseDown) {
        runMouseDownScript();
    }
    else {
        runMouseUpScript();
    }
}
```

Siehe auch

Mouse, on mouseDown (Ereignisprozedur), mouseH, mouseUp, on mouseUp (Ereignisprozedur), mouseV

mouseDownScript

Syntax

```
the mouseDownScript
```

Beschreibung

Diese Systemeigenschaft gibt den Lingo-Code an, der ausgeführt werden soll, wenn der Benutzer die Maustaste drückt. Dieser Lingo-Code wird als String in Anführungszeichen geschrieben; hierbei kann es sich um eine einfache Anweisung oder ein Skript handeln, das eine Prozedur aufruft. Der Standardwert ist `EMPTY`, was bedeutet, dass der Eigenschaft `mouseDownScript` kein Lingo-Code zugeordnet ist.

Wenn die Maustaste gedrückt wird und die Eigenschaft `mouseDownScript` definiert ist, führt Lingo zuerst die für die `mouseDownScript`-Eigenschaft angegebenen Anweisungen aus. Es werden keine anderen `on mouseDown`-Prozeduren ausgeführt, es sei denn, die Anweisungen enthalten den Befehl `pass`, damit die Nachricht `mouseDown` an andere Objekte im Film weitergegeben werden kann.

Die Einstellung der Eigenschaft `mouseDownScript` hat dieselbe Funktion wie der Befehl `when keyDown then` in früheren Versionen von Director.

Wenn Sie die Anweisungen, die Sie für die Eigenschaft `mouseDownScript` angegeben haben, ausschalten möchten, verwenden Sie die Anweisung `set the mouseDownScript to empty`.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Die folgende Anweisung gibt das Skript an, das ausgeführt wird, wenn der Benutzer mit der Maustaste klickt.

```
-- Lingo syntax
the mouseDownScript = "go to the frame"

// JavaScript syntax
_system.mouseDownScript = "_movie.go(_movie.frame)";
```

Siehe auch

stopEvent(), mouseUpScript, on mouseDown (Ereignisprozedur), on mouseUp (Ereignisprozedur)

mouseH

Syntax

```
-- Lingo syntax
_mouse.mouseH

// JavaScript syntax
_mouse.mouseH;
```

Beschreibung

Diese Mauseigenschaft zeigt die horizontale Position des Mauszeigers an. Nur Lesen.

Der Wert für `mouseH` ist die Anzahl der Pixel, die der Cursor vom linken Bühnenrand entfernt ist.

Die Eigenschaft `mouseH` eignet sich dazu, Sprites zur horizontalen Position des Mauszeigers zu verschieben und zu überprüfen, ob sich der Zeiger innerhalb eines bestimmten Bereichs auf der Bühne befindet. Wenn Sie die Eigenschaften `mouseH` und `mouseV` zusammen benutzen, können Sie die genaue Position des Cursors feststellen.

Beispiel

Die folgende Prozedur schiebt Sprite 10 an die Position des Mauszeigers und aktualisiert die Bühne, wenn der Benutzer die Maustaste klickt:

```
-- Lingo syntax
on mouseDown
    sprite(10).locH = _mouse.mouseH
    sprite(10).locV = _mouse.mouseV
end

// JavaScript syntax
function mouseDown() {
    sprite(10).locH = _mouse.mouseH;
    sprite(10).locV = _mouse.mouseV;
}
```

Die folgende Anweisung prüft, ob sich der Mauszeiger um mehr als 10 Pixel rechts oder links von einem Ausgangspunkt entfernt befindet, und setzt in diesem Fall die Variable Far auf TRUE:

```
-- Lingo syntax
startH = 7
if (abs(_mouse.mouseH - startH) > 10) then
    Far = TRUE
end if

// JavaScript syntax
var startH = 7;
if (Math.abs(_mouse.mouseH - startH) > 10) {
    var Far = true;
}
```

Siehe auch

locH, locV, Mouse, mouseLoc, mouseV

mouseItem

Syntax

```
-- Lingo syntax
_mouse.mouseItem

// JavaScript syntax
_mouse.mouseItem;
```

Beschreibung

Diese Mauseigenschaft enthält die Nummer des Elements unter dem Mauszeiger, wenn die Eigenschaft aufgerufen wird und sich der Zeiger über einem Feld-Sprite befindet. Nur Lesen.

Ein Element ist eine beliebige Zeichensequenz, die durch den aktuellen Begrenzer, der von der Eigenschaft `the itemDelimiter` eingestellt wird, begrenzt ist. Die Zeichen werden vom Anfang des Felds aus gezählt. Falls sich der Mauszeiger nicht über einem Feld befindet, lautet das Ergebnis -1.

Der Wert der Eigenschaft `mouseItem` kann sich in einer Prozedur oder Schleife ändern. Falls eine Prozedur oder Schleife auf dem Anfangswert für `mouseItem` beruht, wenn sie beginnt, rufen Sie die Eigenschaft einmal auf und ordnen ihren Wert einer lokalen Variablen zu.

Beispiel

Folgende Anweisung stellt fest, ob sich der Mauszeiger über einem Feld-Sprite befindet. Wenn nicht, wird der den Inhalt des Felddarstellers „Instructions“ in „Please point to an item.“ geändert:

```
-- Lingo syntax
if (mouse.mouseItem = -1) then
    member("Instructions").text = "Please point to an item."
end if

// JavaScript syntax
if (_mouse.mouseItem == -1) {
    member("Instructions").text = "Please point to an item.";
}
```

Die folgende Anweisung ordnet der Variablen `currentItem` das Element unter dem Zeiger im angegebenen Feld zu:

```
-- Lingo syntax
currentItem = member(_mouse.mouseMember).item[_mouse.mouseItem]

// JavaScript syntax
var currentItem = member(_mouse.mouseMember).getProp("item",_mouse.mouseItem);
```

Siehe auch

itemDelimiter, Mouse, mouseChar, mouseLine, mouseWord

mouseLevel

Syntax

```
-- Lingo syntax
spriteObjRef.mouseLevel

// JavaScript syntax
spriteObjRef.mouseLevel;
```

Beschreibung

Diese QuickTime-Sprite-Eigenschaft bestimmt, wie Director Mausklicks auf einem QuickTime-Sprite an QuickTime übergibt. Die Fähigkeit, Mausklicks innerhalb des Begrenzungsrechtecks des Sprites zu übergeben, kann für interaktive Medien wie QuickTime VR nützlich sein. Die Sprite-Eigenschaft `mouseLevel` kann die folgenden Werte haben:

- `#controller` – Übergibt nur Klicks auf die Filmsteuerung an QuickTime. Director reagiert nur auf Mausklicks, die außerhalb der Steuerung stattfinden. Dies ist das Standardverhalten für alle QuickTime-Sprites mit Ausnahme von QuickTime VR.
- `#all` – Übergibt alle Mausklicks innerhalb des Begrenzungsrechtecks des Sprites an QuickTime. Es werden keine Klicks an andere Lingo-Prozeduren übergeben.
- `#none` – Übergibt keine Mausklicks an QuickTime. Director reagiert auf alle Mausklicks.
- `#shared` – Übergibt alle Mausklicks innerhalb des Begrenzungsrechtecks des QuickTime VR-Sprites an QuickTime und übergibt diese Ereignisse dann an Lingo-Prozeduren. Dies ist der Standardwert für QuickTime VR.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Das folgende Bildskript prüft, ob der Name des QuickTime-Sprites in Kanal 5 den String „QTVR“ enthält. Wenn ja, setzt dieses Skript `mouseLevel` auf `#all`. Andernfalls wird `mouseLevel` auf `#none` gesetzt.

```
on prepareFrame
    if sprite(5).member.name contains "QTVR" then
        sprite(5).mouseLevel = #all
    else
        sprite(5).mouseLevel = #none
    end if
end

// JavaScript syntax
function prepareFrame() {
    var nm = sprite(5).member.name;
    var nmStr = nm.indexOf("QTVR");
    if (nmStr != -1) {
        sprite(5).mouseLevel = symbol("all");
    } else {
        sprite(5).mouseLevel = symbol("none");
    }
}
```

mouseLine

Syntax

```
-- Lingo syntax
_mouse.mouseLine

// JavaScript syntax
_mouse.mouseLine;
```

Beschreibung

Diese Mauseigenschaft enthält die Nummer der Zeile unter dem Zeiger, wenn die Eigenschaft aufgerufen wird und sich der Zeiger über einem Feld-Sprite befindet. Nur Lesen.

Es wird vom Anfang des Feldes aus gezählt. Eine Zeile wird durch ein Zeilenumbruchzeichen definiert, nicht durch den Umbruch am Rand des Feldes. Wenn sich der Mauszeiger nicht über einem Feld-Sprite befindet, ist das Ergebnis -1.

Der Wert der Eigenschaft `mouseLine` kann sich in einer Prozedur oder Schleife ändern. Falls eine Eigenschaft mehrmals in einer Prozedur oder Schleife verwendet wird, ist es normalerweise sinnvoll, die Funktion einmal aufzurufen und ihren Wert einer lokalen Variablen zuzuordnen.

Beispiel

Die folgende Anweisung prüft, ob sich der Zeiger über einem Feld-Sprite befindet. Wenn nicht, wird der Inhalt des Felddarstellers „Instructions“ in „Please point to a line.“ geändert.

```
-- Lingo syntax
if (_mouse.mouseLine = -1) then
    member("Instructions").text = "Please point to a line."
end if

// JavaScript syntax
if (_mouse.mouseLine == -1) {
    member("Instructions").text = "Please point to a line.";
}
```

Die folgende Anweisung weist den Inhalt der Zeile unter dem Zeiger im angegebenen Feld der Variablen currentLine zu:

```
-- Lingo syntax
currentLine = member(_mouse.mouseMember).line[_mouse.mouseLine]

// JavaScript syntax
var currentLine = member(_mouse.mouseMember).getProp("line",_mouse.mouseLine);
```

Siehe auch

Mouse, mouseChar, mouseItem, mouseWord

mouseLoc

Syntax

```
-- Lingo syntax
_mouse.mouseLoc

// JavaScript syntax
_mouse.mouseLoc;
```

Beschreibung

Diese Mauseigenschaft gibt die aktuelle Position des Mauszeigers als Punkt (point()) zurück. Nur Lesen.

Die Punktposition wird als zwei Koordinaten angegeben, und zwar zuerst die horizontale, dann die vertikale Position.

Beispiel

Die folgende Anweisung zeigt die aktuelle Position der Maus an.

```
-- Lingo syntax
trace(_mouse.mouseLoc)

// JavaScript syntax
trace(_mouse.mouseLoc);
```

Siehe auch

Mouse, mouseH, mouseV

mouseMember

Syntax

```
-- Lingo syntax
_mouse.mouseMember

// JavaScript syntax
_mouse.mouseMember;
```

Beschreibung

Diese Mauseigenschaft gibt den Darsteller zurück, der dem Sprite zugeordnet ist, das sich beim Aufruf der Eigenschaft unter dem Zeiger befindet. Nur Lesen.

Befindet sich der Zeiger nicht über einem Sprite, gibt diese Eigenschaft das Ergebnis `VOID` (Lingo) oder `null` (JavaScript-Syntax) zurück.

Anhand dieser Eigenschaft können Sie einen Film spezifische Aktionen durchführen lassen, wenn der Zeiger über ein Sprite rollt und das Sprite einen bestimmten Darsteller verwendet.

Der Wert der Eigenschaft `mouseMember` kann sich häufig ändern. Wenn diese Eigenschaft mehrmals in einer Prozedur mit dem gleichen Wert verwendet werden soll, weisen Sie den Wert von `mouseMember` zur Beginn der Prozedur einer lokalen Variablen zu, und verwenden Sie dann die Variable.

Beispiel

Die folgende Anweisung prüft, ob der Darsteller "Nicht betreten“ der Darsteller ist, der dem Sprite unter dem Zeiger zugeordnet ist, und blendet in diesem Fall eine Hinweismeldung ein. Dieses Beispiel zeigt, wie Sie auf Grundlage des dem Sprite zugeordneten Darstellers eine Aktion angeben können.

```
-- Lingo syntax
if (_mouse.mouseMember = member("Off Limits")) then
    _player.alert("Stay away from there!")
end if

// JavaScript syntax
if (_mouse.mouseMember == member("Off Limits")) {
    _player.alert("Stay away from there!");
}
```

Die folgende Anweisung ordnet den Darsteller des Sprites unter dem Zeiger der Variablen `lastMember` zu:

```
-- Lingo syntax
lastMember = _mouse.mouseMember

// JavaScript syntax
var lastMember = _mouse.mouseMember;
```

Siehe auch

castLibNum, Mouse

mouseOverButton

Syntax

```
-- Lingo syntax
spriteObjRef.mouseOverButton
// JavaScript syntax
spriteObjRef.mouseOverButton;
```

Beschreibung

Diese Flash-Sprite-Eigenschaft gibt an, ob sich der Mauszeiger über einer Schaltfläche in einem Flash-Film-Sprite befindet, das durch den Parameter *whichFlashSprite* angegeben wird (TRUE), oder ob sich der Mauszeiger außerhalb der Grenzen des Sprites bzw. innerhalb der Grenzen des Sprites, jedoch über einem Objekt wie dem Hintergrund befindet, das keine Schaltfläche ist (FALSE).

Diese Eigenschaft kann getestet, aber nicht eingestellt werden.

Beispiel

Dieses Bildskript prüft, ob sich der Mauszeiger im Flash-Film in Sprite 3 über einer Navigationsschaltfläche befindet. Befindet sich der Zeiger über der Schaltfläche, aktualisiert das Skript ein Textfeld anhand einer entsprechenden Meldung. Andernfalls löscht das Skript die Meldung.

```
-- Lingo syntax
on enterFrame
    case sprite(3).mouseOverButton of
        TRUE:
        member("Message Line").text = "Click here to go to the next page."
        FALSE:
        member("Message Line").text = " "
    end case
    _movie.updatestage()
end

// JavaScript syntax
function enterFrame() {
    switch(sprite(3).mouseOverButton)
    {
        case 1:
            member("Message Line").text = "Click here to go to the next page.";
            break;
        case 0:
            member("Message Line").text = " ";
            break;
    }
    _movie.updatestage();
}
```

mouseUp

Syntax

```
-- Lingo syntax
_mouse.mouseUp

// JavaScript syntax
_mouse.mouseUp;
```

Beschreibung

Diese Mauseigenschaft zeigt an, ob die Maustaste losgelassen (TRUE) oder gedrückt (FALSE) ist. Nur Lesen.

Beispiel

Die folgende Prozedur bewirkt, dass der Film abgespielt wird, solange der Benutzer die Maustaste drückt. Der Abspielkopf hält an, sobald der Benutzer die Maustaste loslässt.

```
-- Lingo syntax
on exitFrame me
    if (_mouse.mouseUp) then
        _movie.go(_movie.frame)
    end if
end

// JavaScript syntax
function exitFrame() {
     if (_mouse.mouseUp) {
        _movie.go(_movie.frame);
    }
}
```

Die folgende Anweisung fordert Lingo auf, die Wiederholungsschleife oder die Prozedur zu beenden, wenn der Benutzer die Maustaste loslässt:

```
-- Lingo syntax
if (_mouse.mouseUp) then exit

// JavaScript syntax
if (_mouse.mouseUp) {
    return;
}
```

Siehe auch

Mouse, mouseDown, mouseH, mouseV

mouseUpScript

Syntax

```
the mouseUpScript
```

Beschreibung

Diese Systemeigenschaft bestimmt, welcher Lingo-Code beim Loslassen einer Maustaste ausgeführt wird. Der Lingo-Code wird als ein in Anführungszeichen stehender String angegeben und kann eine einfache Anweisung oder ein Skript sein, das eine Prozedur aufruft.

Wenn die Maustaste losgelassen wird und die Eigenschaft `mouseUpScript` definiert ist, führt Lingo zuerst die für die Eigenschaft `mouseUpScript` angegebenen Anweisungen aus. Es werden keine weiteren `on mouseUp`-Prozeduren ausgeführt, es sei denn, die Anweisungen enthalten den Befehl `pass`, mit dem die Nachricht `mouseUp` an andere Objekte im Film weitergegeben werden kann.

Wenn die von Ihnen für die Eigenschaft `mouseUpScript` angegebenen Anweisungen nicht mehr zutreffen, können Sie diese mit der Anweisung `set the mouseUpScript to empty` deaktivieren.

Das Einstellen der Eigenschaft `mouseUpScript` hat die gleiche Wirkung wie die Verwendung des Befehls `when mouseUp then` in früheren Versionen von Director.

Diese Eigenschaft kann getestet und eingestellt werden. Der Standardwert ist `EMPTY`.

Beispiel

Wenn die folgende Anweisung in Kraft ist und der Film angehalten wird, fährt der Film immer dann fort, wenn der Benutzer die Maustaste loslässt:

```
the mouseUpScript = "go to the frame +1"
```

Die folgende Anweisung bewirkt, dass ein Warnton zu hören ist, sobald der Benutzer nach dem Klicken auf die Bühne die Maustaste loslässt:

```
the mouseUpScript = "if the clickOn = 0 then beep"
```

Die folgende Anweisung setzt `mouseUpScript` auf die benutzerdefinierte Prozedur *myCustomHandler*. Eine benutzerdefinierte Lingo-Prozedur muss zwischen Anführungszeichen stehen, wenn sie zusammen mit der Eigenschaft `mouseUpScript` verwendet wird.

```
the mouseUpScript = "myCustomHandler"
```

Siehe auch

`stopEvent()`, `mouseDownScript`, `on mouseDown (Ereignisprozedur)`, `on mouseUp (Ereignisprozedur)`

mouseV

Syntax

```
-- Lingo syntax
_mouse.mouseV

// JavaScript syntax
_mouse.mouseV;
```

Beschreibung

Diese Mauseigenschaft gibt die vertikale Position des Mauscursors gemessen vom oberen Bühnenrand in Pixeln an. Nur Lesen.

Der Wert dieser Eigenschaft nimmt zu, wenn sich der Cursor nach unten bewegt, und ab, wenn sich der Cursor nach oben bewegt.

Die Eigenschaft `mouseV` eignet sich dazu, Sprites zur vertikalen Position des Mauscursors zu verschieben und zu überprüfen, ob sich der Cursor innerhalb eines bestimmten Bereichs auf der Bühne befindet. Wenn Sie die Eigenschaften `mouseH` und `mouseV` zusammen benutzen, können Sie die genaue Position des Cursors feststellen.

Beispiel

Die folgende Prozedur schiebt Sprite 1 an die Position des Mauszeigers und aktualisiert die Bühne, wenn der Benutzer die Maustaste klickt:

```
-- Lingo syntax
on mouseDown
    sprite(1).locH = _mouse.mouseH
    sprite(1).locV = _mouse.mouseV
end

// JavaScript syntax
function mouseDown() {
    sprite(1).locH = _mouse.mouseH;
    sprite(1).locV = _mouse.mouseV;
}
```

Die folgende Anweisung testet, ob sich der Cursor um mehr als 10 Pixel ober- oder unterhalb eines bestimmten Ausgangspunktes befindet, und stellt die Variable vFar auf TRUE ein, wenn dies der Fall ist:

```
-- Lingo syntax
startV = 7
if (abs(_mouse.mouseV - startV) > 10) then
    vFar = TRUE
end if

// JavaScript syntax
var startV = 7
if (Math.abs(_mouse.mouseV - startV) > 10) {
    var vFar = true;
}
```

Siehe auch

locH, locV, Mouse, mouseH, mouseLoc

mouseWord

Syntax

```
-- Lingo syntax
_mouse.mouseWord

// JavaScript syntax
_mouse.mouseWord;
```

Beschreibung

Diese Mauseigenschaft enthält die Nummer des Wortes unter dem Mauszeiger, wenn die Eigenschaft aufgerufen wird und sich der Zeiger über einem Feld-Sprite befindet. Nur Lesen.

Es wird vom Anfang des Feldes aus gezählt. Wenn sich die Maus nicht über einem Feld befindet, ist das Ergebnis -1.

Der Wert der Eigenschaft `mouseWord` kann sich in einer Prozedur oder Schleife ändern. Falls eine Eigenschaft mehrmals in einer Prozedur oder Schleife verwendet wird, ist es normalerweise sinnvoll, die Funktion einmal aufzurufen und ihren Wert einer lokalen Variablen zuzuordnen.

Beispiel

Die folgende Anweisung prüft, ob sich der Zeiger über einem Feld-Sprite befindet. Wenn nicht, wird der Inhalt des Felddarstellers „Instructions“ in „Please point to a word.“ geändert.

```
-- Lingo syntax
if (_mouse.mouseWord = -1) then
    member("Instructions").text = "Please point to a word."
else
    member("Instructions").text = "Thank you."
end if

// JavaScript syntax
if (_mouse.mouseWord == -1) {
    member("Instructions").text = "Please point to a word.";
}
else {
    member("Instructions").text = "Thank you.";
}
```

Die folgende Anweisung ordnet die Nummer des Wortes unter dem Zeiger im angegebenen Feld der Variablen `currentWord` zu:

```
-- Lingo syntax
currentWord = member(_mouse.mouseMember).word[_mouse.mouseWord]

// JavaScript syntax
var currentWord = member(_mouse.mouseMember).getProp("word",_mouse.mouseWord);
```

Siehe auch

Mouse, mouseChar, mouseItem

moveableSprite

Syntax

```
sprite(whichSprite).moveableSprite
the moveableSprite of sprite whichSprite
```

Beschreibung

Diese Sprite-Eigenschaft gibt an, ob ein Sprite vom Benutzer verschoben werden kann (`TRUE`) oder nicht (`FALSE`).

Mithilfe der Option „Verschiebbar“ im Drehbuch können Sie ein Sprite verschiebbar machen. Sie müssen jedoch Lingo verwenden, um zu steuern, ob ein Sprite verschiebbar ist, und um diese Bedingung je nach Bedarf zu aktivieren und deaktivieren. Damit Benutzer jeweils immer nur ein Sprite ziehen können und das Sprite nach der Positionierung nicht mehr verschoben werden kann, müssen Sie die Sprite-Eigenschaft `moveableSprite` zum entsprechenden Zeitpunkt ein- und ausschalten.

Hinweis: *Wenn Sie individuellere Steuerungsmöglichkeiten wünschen (z. B. das Zurückspringen zum Ausgangspunkt oder das Animieren beim Ziehen), erstellen Sie ein Verhalten, mit dem die zusätzliche Funktionalität gesteuert wird.*

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Die folgende Prozedur macht das Sprite in Kanal 5 verschiebbar:

```
on spriteMove
     sprite(5).moveableSprite = TRUE
end
```

Die folgende Anweisung prüft, ob ein Sprite verschiebbar ist, und zeigt eine Meldung an, falls dies nicht der Fall ist:

```
if sprite(13).moveableSprite = FALSE thenmember("Notice").text = "You can't drag this item by
using the mouse."
```

Siehe auch

mouseLoc

movie

Syntax

```
-- Lingo syntax
windowObjRef.movie

// JavaScript syntax
windowObjRef.movie;
```

Beschreibung

Diese Fenstereigenschaft gibt einen Verweis auf das Movie-Objekt zurück, das in einem angegebenen Fenster wiedergegeben wird. Nur Lesen.

Beispiel

Die folgende Anweisung zeigt das Movie-Objekt, das im Fenster „Empires“ wiedergegeben wird, im Nachrichtenfenster an:

```
-- Lingo syntax
trace(window("Empires").movie)

// JavaScript syntax
trace(window("Empires").movie);
```

Siehe auch

Window

multiSound

Syntax

```
the multiSound
```

Beschreibung

Diese Systemeigenschaft gibt an, ob das System mehr als einen Soundkanal unterstützt und auf einem Windows-Computer eine Mehrkanal-Soundkarte erfordert (TRUE) oder nicht (FALSE).

Beispiel

Die folgende Anweisung spielt die Sounddatei "Music“ im Soundkanal 2 ab, wenn der Computer mehr als einen Soundkanal unterstützt:

```
if the multiSound then sound playFile 2, "Music.wav"
```

name

Syntax

```
-- Lingo syntax
castObjRef.name
memberObjRef.name
_movie.name
windowObjRef.name

// JavaScript syntax
castObjRef.name;
memberObjRef.name;
_movie.name;
windowObjRef.name;
```

Beschreibung

Diese Besetzungs-, Darsteller-, Film- und Fenstereigenschaft gibt den Namen eines Objekts zurück oder legt ihn fest. Lesen/Schreiben für Cast-, Member- und Window-Objekte. Nur Lesen für Movie-Objekte.

Beispiel

Die folgende Anweisung ändert den Namen des Darstellers 2 in „Newname“.

```
-- Lingo syntax
member(1).name="Newname"

// JavaScript syntax
member(1).name="Newname";
```

Siehe auch

Cast Library, Member, Movie, Window

name (3D)

Syntax

```
member(whichCastmember).texture(whichTexture).name
member(whichCastmember).shader(whichShader).name
member(whichCastmember).motion(whichMotion).name
member(whichCastmember).modelResource(whichModelResource).name
member(whichCastmember).model(whichModel).name
member(whichCastmember).light(whichLight).name
member(whichCastmember).camera(whichCamera).name
member(whichCastmember).group(whichGroup).name
node.name
```

Beschreibung

Mit dieser 3D-Eigenschaft können Sie bei einem Objektbezug den Namen des referenzierten Objekts ermitteln. Der Name kann abgerufen, jedoch nicht geändert werden.

Alle Namen müssen eindeutig sein. Bei Erstellung per Lingo wird der in der Konstruktorfunktion angegebene Name zurückgegeben. Bei Erstellung über ein 3D-Modellierungsprogramm wird u.U. der Name des Modells zurückgegeben.

Beispiel

Die folgende Anweisung setzt den Namen der fünften Kamera im Darsteller `TableScene` auf `BirdCam`:

```
member("TableScene").camera[5].name = "BirdCam"
```

name (Menüeigenschaft)

Syntax

```
the name of menu(whichMenu)
the name of menu whichMenu
```

Beschreibung

Diese Menüeigenschaft gibt einen String zurück, der den Namen der angegebenen Menünummer enthält.

Diese Eigenschaft kann getestet, aber nicht eingestellt werden. Mit dem Befehl `installMenu` können Sie eine benutzerdefinierte Menüleiste einrichten.

Hinweis: *Im Shockwave Player sind keine Menüs verfügbar.*

Beispiel

Die folgende Anweisung weist den Namen des Menüs mit der Nummer 1 der Variablen `firstMenu` zu:

```
firstMenu = menu(1).name
```

Die folgende Prozedur gibt eine Liste von Menünamen zurück (einen pro Zeile):

```
on menuList
    theList = []
    repeat with i = 1 to the number of menus
        theList[i] = the name of menu i
    end repeat
    return theList
end menuList
```

Siehe auch

`number (Menüs)`, `name (Menüelementeigenschaft)`

name (Menüelementeigenschaft)

Syntax

```
the name of menuItem(whichItem) of menu(whichMenu)
the name of menuItem whichItem of menu whichMenu
```

Beschreibung

Diese Menüeigenschaft bestimmt den Text, der im durch *whichItem* bezeichneten Menüelement erscheint, das sich im durch *whichMenu* bezeichneten Menü befindet. Das Argument *whichItem* ist entweder der Name oder die Nummer eines Menüelements, und *whichMenu* ist entweder ein Menüname oder eine Menünummer.

Diese Eigenschaft kann getestet und eingestellt werden.

Hinweis: *Im Shockwave Player sind keine Menüs verfügbar.*

Beispiel

Die folgende Anweisung setzt die Variable `itemName` auf den Namen des achten Elements im Menü "Bearbeiten":

```
set itemName = the name of menuItem(8) of menu("Edit")
```

Die folgende Anweisung bewirkt, dass ein bestimmter Dateiname auf das Wort *Öffnen* im Menü Datei folgt:

```
the name of menuItem("Open") of menu("fileMenu") = "Open" && fileName
```

Siehe auch

name (Menüeigenschaft), number (Menüelemente)

name (Mixer)

Syntax

```
mixer.name
```

Beschreibung

Diese Audiomixereigenschaft stellt den Namen eines Mixerdarstellers ein oder gibt ihn zurück. Diese Eigenschaft kann ausgelesen und beschrieben werden.

Beispiele

```
--Lingo syntax
on mouseUp me
    mixerRef.name = "BackgroundMixer"
    put mixerRef.name -- Displays the name of the mixer associated with mixerRef.
end

// JavaScript syntax
function mouseUp(){
    mixerRef.name = "BackgroundMixer"
    put (mixerRef.name) ; // Displays the name of the mixer associated with mixerRef.
}
```

Siehe auch

Mixer

name (Soundobjekt)

Syntax

```
soundObject.name (Read-write)
```

Beschreibung

Soundobjekteigenschaft, gibt den Namen des Soundobjekts zurück Diese Eigenschaft kann ausgelesen und beschrieben werden.

Beispiele

```
--Lingo syntax
on mouseUp me
    put soundObjRef.name -- Displays the name of the sound object associated with
-- soundobjectRef.
end

// JavaScript syntax
function mouseUp(){
put (soundObjRef.name) ; // Displays the name of the sound object associated with
// soundobjectRef.
}
```

name (Sprite)

Syntax

```
-- Lingo syntax
spriteObjRef.name

// JavaScript syntax
spriteObjRef.name;
```

Beschreibung

Diese Sprite-Eigenschaft identifiziert den Namen eines Sprites. Lesen/Schreiben; nur während einer Drehbuchaufzeichnungssitzung.

Im Gegensatz zu Sprite-Anzeigeeigenschaften wie `backColor` oder `blend` kann ein Sprite `name` kein mit Skript erstelltes Sprite sein. Das bedeutet, dass `name` nur während einer Drehbuchaufzeichnungssitzung festgelegt werden kann, also zwischen den Aufrufen der Methoden `beginRecording()` und `endRecording()` des Movie-Objekts. `name` kann nur festgelegt werden, wenn `beginRecording()` in oder vor einem Bild im Drehbuch aufgerufen, das das Sprite enthält.

Hinweis: *Wenn eine Drehbuchaufzeichnungssitzung mithilfe von `beginRecording()` begonnen wird, werden die Eigenschaften aller mit Skript erstellten Sprites und Sprite-Kanäle zurückgesetzt.*

Wenn Sie mithilfe von Skriptcode ein neues Sprite während einer Drehbuchaufzeichnungssitzung erstellen und die Sprite-Daten mithilfe von `updateFrame()` auf die Sitzung anwenden, können Sie den Namen des Sprites erst festlegen, wenn Sie zu dem Bild zurückwechseln, in dem das Sprite erstellt wurde. Zu einem bestimmten Bild zurückwechseln können Sie, indem Sie eine Methode wie `go()` verwenden.

Beispiel

Die folgende Anweisung setzt den Namen von Sprite 5 auf „Hintergrundsound“:

```
-- Lingo syntax
sprite(5).name = "Background Sound"

// JavaScript syntax
sprite(5).name = "Background Sound";
```

Siehe auch

`beginRecording()`, `endRecording()`, `go()`, `Sprite`, `updateFrame()`

name (Sprite-Kanal)

Syntax

```
-- Lingo syntax
spriteChannelObjRef.name

// JavaScript syntax
spriteChannelObjRef.name;
```

Beschreibung

Diese Sprite-Kanal-Eigenschaft identifiziert den Namen eines Sprite-Kanals. Lesen/Schreiben; nur während einer Drehbuchaufzeichnungssitzung.

Legen Sie `name` eines Sprite-Kanals während einer Drehbuchaufzeichnungssitzung fest, also zwischen den Aufrufen der Methoden `beginRecording()` und `endRecording()` des Movie-Objekts.

Hinweis: *Wenn eine Drehbuchaufzeichnungssitzung mithilfe von `beginRecording()` begonnen wird, werden die Eigenschaften aller mit Skript erstellten Sprites und Sprite-Kanäle zurückgesetzt.*

Im Gegensatz zur Eigenschaft `name` eines Sprite-Objekts, die nur in oder nach einem Bild festgelegt werden kann, in dem ein Sprite im Drehbuch vorkommt, kann die Eigenschaft `name` eines Sprite Channel-Objekts in einem leeren Kanal festgelegt werden. Dies bedeutet, dass `updateFrame()` nicht vor dem Festlegen von `name` für den Sprite- Kanal aufgerufen werden muss.

Eine Änderung am Namen eines Sprite-Kanals mithilfe von Skriptcode wird im Drehbuchfenster nicht angezeigt.

Beispiel

Die folgende Anweisung legt den Namen von Sprite-Kanal 6 während einer Drehbuchaufzeichnung auf „Kite String“ fest:

```
-- Lingo syntax
on mouseDown
    _movie.beginRecording()
    channel(6).name = "Kite string"
    _movie.endRecording()
end

// JavaScript syntax
function mouseDown() {
    _movie.beginRecording();
    channel(6).name = "Kite string";
    _movie.endRecording();
}
```

Siehe auch

`beginRecording()`, `endRecording()`, `Sprite Channel`

name (timeout)

Syntax

```
timeoutObject.name
```

Beschreibung

Diese Timeout-Eigenschaft ist der Name des Timeout-Objekts, der bei der Erstellung des Objekts definiert wurde. Zur Erstellung von Timeout-Objekten wird der Befehl `new()` verwendet.

Beispiel

Die folgende Timeout-Prozedur blendet eine Hinweismeldung mit dem Namen des Timeouts ein, von dem das Ereignis gesendet wurde:

```
on handleTimeout timeoutObject
    alert "Timeout:" && timeoutObject.name
end
```

Siehe auch

`forget() (Timeout)`, `new()`, `period`, `persistent`, `target`, `time (Timeout-Objekt)`, `timeout()`, `timeoutHandler`, `timeoutList`

name (XML)

Syntax

```
XMLnode.name
```

Beschreibung

Diese Eigenschaft gibt den Namen des angegebenen XML-Nodes zurück.

Beispiel

Als Ausgangspunkt wird das folgende XML-Dokument verwendet:

```
<?xml version="1.0"?>
    <e1>
<tagName attr1="val1" attr2="val2"/>
<e2> element 2</e2>
<e3>element 3</e3>
    </e1>
```

Die folgende Lingo-Anweisung gibt den Namen des zweiten Tags zurück, das im Tag „<e1>“ verschachtelt ist:

```
put gParserObject.child[1].child[2].name
-- "e2"
```

Siehe auch

`attributeName`

near (fog)

Syntax

```
member(whichCastmember).camera(whichCamera).fog.near
cameraReference.fog.near
member(whichCastmember).camera(whichCamera).fog.far
cameraReference.fog.far
```

Beschreibung

Mit dieser 3D-Eigenschaft können Sie die Entfernung zwischen der Vorderseite der Kamera und dem Punkt abrufen und festlegen, an dem der Nebeleffekt beginnt, wenn `fog.enabled` auf `TRUE` gesetzt ist.

Der Standardwert dieser Eigenschaft lautet 0.0.

Beispiel

Die folgende Anweisung setzt die Eigenschaft near des Nebels der Kamera "Standardansicht" auf 100.

```
-- Lingo syntax
member("3dobjects").camera("defaultview").fog.near = 100.0

// JavaScript syntax
member("3dobjects").getPropRef("camera",1).fog.near = 100.00;
```

Siehe auch

`fog`, `far (fog)`, `enabled (fog)`, `decayMode`

nearFiltering

Syntax

```
member(whichCastmember).texture(whichTexture).nearFiltering
member(whichCastmember).shader(whichShader).texture(whichTexture).nearFiltering
member(whichCastmember).model(whichModel).shader.texture(whichTexture).nearFiltering
member(whichCastmember).model(whichModel).shaderList[shaderListIndex].texture(whichTexture).
nearFiltering
```

Beschreibung

Mit dieser 3D-Textureigenschaft können Sie ermitteln und festlegen, ob beim Rendern einer projizierten Texture-Map, die eine größere Bildschirmfläche abdeckt als die ursprüngliche Texturquelle, bilineare Filterung verwendet wird. Bei der bilinearen Filterung werden Texturfehler ausgeglichen und so das Aussehen der Textur verbessert. Die Fehlerkorrektur erfolgt bei der bilinearen Filterung in zwei, bei der trilinearen Filterung in drei Dimensionen. Die Filterung verbessert zwar das Aussehen, wirkt sich aber nachteilig auf die Leistung aus. Damit ist die bilineare Filterung weniger leistungsabträglich als die trilineare Filterung.

Wenn die Eigenschaft den Wert `TRUE` aufweist, wird die bilineare Filterung verwendet. Wenn der Wert `FALSE` lautet, wird die bilineare Filterung nicht verwendet. Der Standardwert ist `TRUE`.

Beispiel

Die folgende Anweisung deaktiviert die bilineare Filterung für die erste Textur im Darsteller „3Dobjects“.

```
-- Lingo syntax
member("3dobjects").texture[1].nearFiltering = FALSE

// JavaScript syntax
member("3dobjects").getPropRef("texture",1).nearFiltering = 0;
```

netPresent

Syntax

```
-- Lingo syntax
_player.netPresent

// JavaScript syntax
_player.netPresent;
```

Beschreibung

Diese Player-Eigenschaft ermittelt, ob die für den Zugang zum Internet erforderlichen Xtra-Erweiterungen verfügbar sind. Sie gibt aber nicht an, ob momentan eine Internetverbindung besteht. Nur Lesen.

Sind keine Xtra-Erweiterungen für die Netzwerkunterstützung verfügbar, funktioniert zwar `netPresent` korrekt, die Methode `netPresent()` führt jedoch zu einem Skriptfehler.

Beispiel

Die folgende Anweisung sendet einen Warnhinweis, wenn die Xtra-Erweiterungen nicht verfügbar sind:

```
-- Lingo syntax
if (not(_player.netPresent)) then
    _player.alert("Sorry, the Network Support Xtras could not be found.")
end if

// JavaScript syntax
if (!(_player.netPresent)) {
    _player.alert("Sorry, the Network Support Xtras could not be found.");
}
```

Siehe auch

Player

netThrottleTicks

Syntax

```
-- Lingo syntax
_player.netThrottleTicks

// JavaScript syntax
_player.netThrottleTicks;
```

Beschreibung

Mit dieser Player-Eigenschaft können Sie auf dem Mac festlegen, wie häufig ein Netzwerkvorgang durchgeführt wird. Lesen/Schreiben.

Der Standardwert ist 15. Je höher dieser Wert ist, desto flüssiger verlaufen das Abspielen von Filmen und die Animation, aber desto weniger Zeit wird auf die Durchführung von Netzwerkaktivitäten verwendet. Eine niedrige Einstellung stellt mehr Zeit für Netzwerkvorgänge bereit, wirkt sich jedoch negativ auf die Abspiel- und Animationsleistung aus.

Diese Eigenschaft betrifft nur die Erstellungsumgebung und Projektoren auf dem Mac. Sie wird unter Windows und von Shockwave Player für den Mac ignoriert.

Beispiel

Die folgende Anweisung zeigt den Wert von netthrottleticks im Player an.

```
-- Lingo syntax
put _player.netThrottleTicks

// JavaScript syntax
    put( _player.netThrottleTicks);
```

Siehe auch

Player

node

Syntax

```
-- Lingo syntax
spriteObjRef.node

// JavaScript syntax
spriteObjRef.node;
```

Beschreibung

Diese QuickTime VR-Sprite-Eigenschaft ist die aktuelle Node-ID, die vom Sprite angezeigt wird.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Die folgende Anweisung zeigt die aktuelle Node-ID an, die vom Sprite angezeigt wird.

```
-- Lingo syntax
put sprite(3).node

// JavaScript syntax
put( sprite(3).node);
```

nodeEnterCallback

Syntax

```
-- Lingo syntax
spriteObjRef.nodeEnterCallback

// JavaScript syntax
spriteObjRef.nodeEnterCallback;
```

Beschreibung
Diese QuickTime VR-Sprite-Eigenschaft enthält den Namen der Prozedur, die ausgeführt wird, wenn der QuickTime VR-Film zu einem neuen aktiven Node auf der Bühne übergeht. Die Nachricht hat zwei Argumente: den Parameter `me` und die ID des anzuzeigenden Knotens.

Das QuickTime VR-Sprite erhält die Nachricht zuerst.

Wenn Sie den Rückruf löschen möchten, setzen Sie diese Eigenschaft auf 0.

Zur Vermeidung von Leistungsabfällen sollten Sie Rückrufeigenschaften nur einstellen, wenn dies wirklich notwendig ist.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel
Die folgende Anweisung zeigt den Namen der Prozedur an, die ausgeführt wird, nachdem der QuickTime VR-Film zu einem neuen aktiven Node auf der Bühne übergegangen ist.

```
-- Lingo syntax
put sprite(3).nodeEnterCallback

// JavaScript syntax
put( sprite(3).nodeEnterCallback);
```

nodeExitCallback

Syntax
```
-- Lingo syntax
spriteObjRef.nodeExitCallback

// JavaScript syntax
spriteObjRef.nodeExitCallback;
```

Beschreibung
Diese QuickTime VR-Sprite-Eigenschaft enthält den Namen der Prozedur, die ausgeführt wird, wenn der QuickTime VR-Film zu einem neuen aktiven Node auf der Bühne übergeht. Die Nachricht hat drei Argumente: den Parameter `me`, die ID des Nodes, den der Film gleich verlassen wird, und die ID des Nodes, zu dem der Film anschließend überwechselt.

Der von der Prozedur zurückgegebene Wert bestimmt, ob der Film zum nächsten Node übergeht. Wenn die Prozedur `#continue` zurückgibt, fährt das QuickTime VR-Sprite mit einem normalen Node-Übergang fort. Wenn die Prozedur `#cancel` zurückgibt, findet kein Übergang statt, und der Film verbleibt im ursprünglichen Node.

Setzen Sie die Eigenschaft auf 0, um den Rückruf zu löschen.

Das QuickTime VR-Sprite erhält die Nachricht zuerst.

Zur Vermeidung von Leistungsabfällen sollten Sie Rückrufeigenschaften nur einstellen, wenn dies wirklich notwendig ist.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel
Die folgende Anweisung zeigt den Namen der Prozedur an, die ausgeführt wird, wenn der QuickTime VR-Film im Begriff ist, zu einem neuen aktiven Node auf der Bühne überzugehen.

```
-- Lingo syntax
put sprite(3).nodeExitCallback

// JavaScript syntax
put( sprite(3).nodeExitCallback);
```

nodeType

Syntax

```
-- Lingo syntax
spriteObjRef.nodeType

// JavaScript syntax
spriteObjRef.nodeType;
```

Beschreibung

Diese QuickTime VR-Sprite-Eigenschaft gibt den Typ des Nodes zurück, der sich momentan für das angegebene Sprite auf der Bühne befindet. Mögliche Werte: `#object`, `#panorama` oder `#unknown`. (`#unknown` ist der Wert für ein Sprite, das kein QuickTime VR-Sprite ist.)

Diese Eigenschaft kann getestet, aber nicht eingestellt werden.

Beispiel

Die folgende Anweisung zeigt den Typ des Nodes an, der sich momentan für das angegebene Sprite auf der Bühne befindet.

```
-- Lingo syntax
put sprite(3).nodeType

// JavaScript syntax
put( sprite(3).nodeType);
```

normalList

Syntax

```
member(whichCastmember).modelResource(whichModelResource).normalList
model.meshDeform.mesh[index].normalList
```

Beschreibung

Mit dieser 3D-Eigenschaft können Sie bei einer Modellressource vom Typ `#mesh` die Eigenschaft `normalList` der Modellressource ermitteln und festlegen.

Die Eigenschaft `normalList` ist eine lineare Vektorliste, aus der Sie beim Generieren der Gitternetzflächen Scheitelpunktnormalen angeben können.

Diese Eigenschaft muss auf eine Liste gesetzt werden, die so viele Vektoren enthält, wie im Aufruf von `newMesh()` angegeben sind.

Die Eigenschaft `normalList` kann auch durch die `generateNormals()`-Methode für Gitternetzmodellressourcen generiert werden.

Analog dazu ist beim Modifizierer `meshDeform` die Eigenschaft `normalList` eine lineare Vektorliste, aus der Sie beim Verformen des Gitternetzes Scheitelpunktnormale angeben können.

Weitere Informationen zu Flächen- und Scheitelpunktnormalen enthält der Eintrag zu `normals`.

Beispiel

Die folgende Anweisung zeigt die Eigenschaft `normalList` der Modellressource "pyramid" des Darstellers "3Dobjects" an.

```
-- Lingo syntax
put member("3Dobjects").modelResource("pyramid").normalList[2]

// JavaScript syntax
put(member("3Dobjects").getPropRef("modelResource",10).normalList[2]);
```

Siehe auch

face, meshDeform (Modifizierer)

normals

Syntax

```
member(whichCastmember).modelResource(whichModelResource).face[index].normals
```

Beschreibung

Bei Modellressourcen vom Typ `#mesh` (mit dem Befehl `newMesh` erstellt) können Sie mit dieser 3D-Flächeneigenschaft die Liste der normierten Vektoren (Normalen) ermitteln und festlegen, die von der durch den Parameter *index* angegebenen Fläche verwendet werden.

Setzen Sie diese Eigenschaft auf eine lineare Liste mit Ganzzahlen, die den Indexpositionen der Scheitelpunktnormalen in der Modellressourceneigenschaft `normalList` entsprechen.

Diese Eigenschaft muss die gleiche Länge aufweisen wie die Liste `face[index].vertices`oder leer sein („[]“).

Geben Sie für diese Eigenschaft keinen Wert an, wenn Sie vorhaben, normierte Vektoren (Normale) mit dem Befehl `generateNormals()` zu generieren.

Wenn Sie diese Eigenschaft ändern oder den Befehl `generateNormals()` verwenden, müssen Sie den Befehl `build()` zur Neuerstellung des Gitternetzes aufrufen.

Beispiel

Die folgende Anweisung zeigt die Eigenschaft normals der fünften Fläche der Modellressource "pyramid" des Darstellers "3Dobjects" an.

```
-- Lingo syntax
put member("3Dobjects").modelResource("pyramid").face[5].normals

// JavaScript syntax
put(member("3Dobjects").getPropRef("modelResource",10).face[5].normals);
```

Siehe auch

face, normalList, vertices

number (Besetzung)

Syntax

```
-- Lingo syntax
castObjRef.number

// JavaScript syntax
castObjRef.number;
```

Beschreibung

Diese Besetzungsbibliothekseigenschaft gibt die Nummer einer angegebenen Besetzungsbibliothek zurück. Nur Lesen.

Beispiel

Bei der folgenden Wiederholungsschleife wird die Anzahl der Darsteller in jeder Besetzung des Films im Nachrichtenfenster angezeigt:

```
-- Lingo syntax
repeat with n = 1 to _movie.castLib.count
    put(castLib(n).name && "contains" && castLib(n).member.count && "cast members.")
end repeat

// JavaScript syntax
for (var n=1; n<=_movie.castLib.count; n++) {
    put(castLib(n).name + " contains " + castLib(n).member.count + " cast members.")
}
```

Siehe auch

Cast Library

number (Zeichen)

Syntax

```
the number of chars in chunkExpression
```

Beschreibung

Dieser Chunk-Ausdruck gibt die Anzahl der Zeichen in einem Chunk-Ausdruck zurück.

Chunk-Ausdrücke sind beliebige Zeichen (darunter Leer- und Steuerzeichen wie Tabulator und Zeilenschaltung), Wörter, Elemente und Zeilen in einem beliebigen Zeichencontainer. Zu Containern zählen Felddarsteller, Textdarsteller und Variablen, die Strings enthalten, sowie spezifische Zeichen, Wörter, Elemente, Zeilen und Bereiche in Containern.

Hinweis: *Die Funktion* `count()` *stellt eine effizientere Alternative zur Bestimmung der Anzahl von Zeichen in einem Chunk-Ausdruck dar.*

Beispiel

Die folgende Anweisung zeigt die Anzahl der Zeichen in der Zeichenfolge „Adobe, the Multimedia Company“ im Nachrichtenfenster an:

```
put the number of chars in "Adobe, the Multimedia Company"
```

Das Ergebnis ist 29.

Die folgende Anweisung setzt die Variable `charCounter` auf die Anzahl der Zeichen im Wort "i" im String "Namen":

```
charCounter = the number of chars in member("Names").word[i]
```

Das gleiche Ergebnis erzielen Sie mit Textdarstellern unter Verwendung der folgenden Syntax:

```
charCounter = member("Names").word[i].char.count
```

Siehe auch

`length()`, `char...of`, `count()`, `number (Elemente)`, `number (Zeilen)`, `number (Wörter)`

number (Elemente)

Syntax

```
the number of items in chunkExpression
```

Beschreibung

Dieser Chunk-Ausdruck gibt die Anzahl von Elementen in einem Chunk-Ausdruck zurück. Ein Element-Chunk ist eine beliebige Folge von Zeichen, die durch Kommas getrennt sind.

Chunk-Ausdrücke sind beliebige Zeichen, Wörter, Elemente oder Zeilen in einem Zeichencontainer. Zu Containern zählen Felder (Felddarsteller) und Variablen, die Strings enthalten, sowie spezifische Zeichen, Wörter, Elemente, Zeilen und Bereiche in Containern.

Hinweis: *Die Funktion* `count()` *stellt eine effizientere Alternative zur Bestimmung der Anzahl von Elementen in einem Chunk-Ausdruck dar.*

Beispiel

Die folgende Anweisung zeigt die Anzahl der Elemente in der Zeichenfolge „Adobe, the Multimedia Company“ im Nachrichtenfenster an:

```
put the number of items in "Adobe, the Multimedia Company"
```

Das Ergebnis ist 2.

Die folgende Anweisung setzt die Variable `itemCounter` auf die Anzahl von Elementen im Feld "Namen":

```
itemCounter = the number of items in member("Names").text
```

Das gleiche Ergebnis erzielen Sie mit Textdarstellern unter Verwendung der folgenden Syntax:

```
itemCounter = member("Names").item.count
```

Siehe auch

`item...of`, `count()`, `number (Zeichen)`, `number (Zeilen)`, `number (Wörter)`

number (Zeilen)

Syntax

```
the number of lines in chunkExpression
```

Beschreibung

Dieser Chunk-Ausdruck gibt die Anzahl von Zeilen in einem Chunk-Ausdruck zurück. (Mit Zeile ist hier eine Zeichenfolge gemeint, die durch eine Zeilenschaltung begrenzt ist und nicht durch einen automatischen Textumbruch gebildet wird.)

Chunk-Ausdrücke sind beliebige Zeichen, Wörter, Elemente oder Zeilen in einem Zeichencontainer. Zu Containern zählen Felddarsteller, Textdarsteller und Variablen, die Strings enthalten, sowie spezifische Zeichen, Wörter, Elemente, Zeilen und Bereiche in Containern.

Hinweis: *Die Funktion* `count()` *stellt eine effizientere Alternative zur Bestimmung der Anzahl von Zeilen in einem Chunk-Ausdruck dar.*

Beispiel

Die folgende Anweisung zeigt die Anzahl der Zeilen in der Zeichenfolge „Adobe, the Multimedia Company“ im Nachrichtenfenster an:

```
put the number of lines in "Adobe, the Multimedia Company"
```

Das Ergebnis ist 1.

Die folgende Anweisung setzt die Variable `lineCounter` auf die Anzahl der Zeilen im Feld "Namen":

```
lineCounter = the number of lines in member("Names").text
```

Das gleiche Ergebnis erzielen Sie mit Textdarstellern unter Verwendung der folgenden Syntax:

```
lineCounter = member("Names").line.count
```

Siehe auch

`line...of`, `count()`, `number (Zeichen)`, `number (Elemente)`, `number (Wörter)`

number (Darsteller)

Syntax

```
-- Lingo syntax
memberObjRef.number

// JavaScript syntax
memberObjRef.number;
```

Beschreibung

Diese Darstellereigenschaft zeigt die Nummer der Besetzungsbibliothek eines angegebenen Darstellers an. Nur Lesen.

Der Wert dieser Eigenschaft ist eine eindeutige Kennzeichnung des Darstellers in Form einer einzelnen Ganzzahl, die dessen Standort und Position in einer Besetzung beschreibt.

Beispiel

Die folgende Anweisung ordnet die Darstellernummer des Darstellers "Netzschalter" der Variablen *whichCastMember* zu:

```
-- Lingo syntax
whichCastMember = member("Power Switch").number

// JavaScript syntax
var whichCastMember = member("Power Switch").number;
```

Die folgende Anweisung ordnet den Darsteller „Roter Luftballon“ dem Sprite 1 zu:

```
-- Lingo syntax
sprite(1).member = member("Red Balloon").number

// JavaScript syntax
sprite(1).member = member("Red Balloon").number;
```

Mit der folgenden Anweisung können Sie bestätigen, dass ein Darsteller tatsächlich existiert, bevor Sie versuchen, den Darsteller im Sprite auszutauschen:

```
-- Lingo syntax
property spriteNum

on mouseUp me
    if (member("Mike's face").number > 0) then
        sprite(spriteNum).member = "Mike's face"
    end if
end

// JavaScript syntax
function mouseUp() {
    if (member("Mike's face").number > 0) {
        sprite(this.spriteNum).member = "Mike's face"
    }
}
```

Siehe auch

`castLib()`, `Member`

number (Menüs)

Syntax

```
the number of menus
```

Beschreibung

Diese Menüeigenschaft gibt die Anzahl der im aktuellen Film installierten Menüs an.

Diese Menüeigenschaft kann getestet, aber nicht eingestellt werden. Mit dem Befehl `installMenu` können Sie eine benutzerdefinierte Menüleiste einrichten.

Hinweis: *Im Shockwave Player sind keine Menüs verfügbar.*

Beispiel

Die folgende Anweisung bestimmt, ob benutzerdefinierte Menüs im Film installiert sind. Falls noch keine Menüs installiert sind, wird das Menü „Menüleiste“ installiert:

```
if the number of menus = 0 then installMenu "Menubar"
```

Die folgende Anweisung gibt im Nachrichtenfenster die Anzahl von Menüs an, die im aktuellen Film enthalten sind:

```
put the number of menus
```

Siehe auch

`installMenu`, `number (Menüelemente)`

number (Menüelemente)

Syntax

```
the number of menuItems of menu whichMenu
```

Beschreibung

Diese Menüeigenschaft gibt die Anzahl von Menüelementen im benutzerdefinierten Menü an, das durch *whichMenu* angegeben wird. Der Parameter *whichMenu* kann ein Menüname oder eine Menünummer sein.

Diese Menüeigenschaft kann getestet, aber nicht eingestellt werden. Mit dem Befehl `installMenu` können Sie eine benutzerdefinierte Menüleiste einrichten.

Hinweis: *Im Shockwave Player sind keine Menüs verfügbar.*

Beispiel

Die folgende Anweisung stellt die Variable `fileItems` auf die Anzahl von Menüelementen im benutzerdefinierten Menü "Datei" ein:

```
fileItems = the number of menuItems of menu "File"
```

Die folgende Anweisung setzt die Variable `itemCount` auf die Anzahl von Menüelementen im benutzerdefinierten Menü, dessen Menünummer der Variablen "i" entspricht:

```
itemCount = the number of menuItems of menu i
```

Siehe auch

`installMenu`, `number (Menüs)`

number (Sprite-Kanal)

Syntax

```
-- Lingo syntax
spriteChannelObjRef.number

// JavaScript syntax
spriteChannelObjRef.number;
```

Beschreibung

Diese Sprite-Kanal-Eigenschaft gibt die Nummer eines Sprite-Kanals zurück. Nur Lesen.

Beispiel

Die folgende Anweisung zeigt im Nachrichtenfenster die Nummer eines benannten Sprite-Kanals an:

```
-- Lingo syntax
put(channel("Kite String").number)

// JavaScript syntax
put(channel("Kite String").number);
```

Siehe auch

Sprite

number (System)

Syntax

```
the number of castLibs
```

Beschreibung

Diese Systemeigenschaft gibt die Anzahl von Besetzungen im aktuellen Film zurück.

Diese Eigenschaft kann getestet, aber nicht eingestellt werden.

Beispiel

Bei der folgenden Wiederholungsschleife wird die Anzahl der Darsteller in jeder Besetzung des Films im Nachrichtenfenster angezeigt:

```
repeat with n = 1 to the number of castLibs
    put castLib(n).name && "contains" && the number of members of castLib(n) && "cast members."
end repeat
```

number (Wörter)

Syntax

```
the number of words in chunkExpression
```

Beschreibung

Dieser Chunk-Ausdruck gibt die Anzahl von Wörtern in dem durch *chunkExpression* angegebenen Chunk-Ausdruck zurück.

Chunk-Ausdrücke sind beliebige Zeichen, Wörter, Elemente oder Zeilen in einem Zeichencontainer. Zu Containern zählen Felddarsteller, Textdarsteller und Variablen, die Strings enthalten, sowie spezifische Zeichen, Wörter, Elemente, Zeilen und Bereiche in Containern.

Unter dem Eintrag `count` erfahren Sie, wie diese Funktionalität mit Textdarstellern erzielt werden kann.

Hinweis: *Die Funktion* `count()` *stellt eine effizientere Alternative zur Bestimmung der Anzahl von Wörtern in einem Chunk-Ausdruck dar.*

Beispiel

Die folgende Anweisung zeigt im Nachrichtenfenster die Anzahl der Wörter in der Zeichenfolge „Adobe, the multimedia company“ an:

```
put the number of words in "Adobe, the multimedia company"
```

Das Ergebnis ist 3.

Die folgende Prozedur kehrt die Reihenfolge der Wörter in dem in Argument `wordList` angegebenen String um:

```
on reverse wordList
    theList = EMPTY
    repeat with i = 1 to the number of words in wordList
        put word i of wordList & " " before theList
    end repeat
    delete theList.char[thelist.char.count]
    return theList
end
```

Siehe auch

`count()`, `number (Zeichen)`, `number (Elemente)`, `number (Zeilen)`, `word...of`

number of members

Syntax

```
the number of members of castLib whichCast
```

Beschreibung

Diese Darstellereigenschaft gibt die Nummer des letzten Darstellers in der angegebenen Besetzung zurück.

Diese Eigenschaft kann getestet, aber nicht eingestellt werden.

Beispiel

Die folgende Anweisung zeigt im Nachrichtenfenster den Typ jedes Darstellers in der Besetzung „Zentralbesetzung“ an. Anhand der Eigenschaft „number of members“ der Eigenschaft `castLib` wird bestimmt, wie oft die Schleife wiederholt wird.

```
repeat with i = 1 to the number of members of castLib("Central Casting")
    put "Cast member" && i && "is a" && member(i, "Central Casting").type
end repeat
```

number of xtras

Syntax

```
the number of xtras
```

Beschreibung

Diese Systemeigenschaft gibt die Anzahl der für den Film verfügbaren Xtra-Skripterweiterungen an. Die Xtra-Erweiterungen wurden entweder mit dem Befehl `openxlib` geöffnet oder sind im Ordner „Configuration\Xtras“ vorhanden.

Diese Eigenschaft kann getestet, aber nicht eingestellt werden.

Beispiel

Die folgende Anweisung zeigt die Anzahl der für den Film verfügbaren Xtra-Skripterweiterungen im Nachrichtenfenster an:

```
put the number of xtras
```

numBuffersToPreload

Syntax

```
mixer.numBuffersToPreload
```

Beschreibung

Diese Audioeigenschaft bestimmt die Anzahl der Puffer, die berechnet werden, bevor ein Aufruf von `play` zurückkehrt. Standardmäßig beträgt dieser Wert 0. Diese Eigenschaft kann nur festgelegt werden, falls sich der Mixer im Zustand `#stopped` befindet.

Beispiele

```
--Lingo syntax
on mouseUp me
    mixerRef.numBuffersToPreload = 2
end

// JavaScript syntax
function mouseUp(){
mixerRef.numBuffersToPreload = 2;
}
```

Siehe auch

Mixer

numChannels

Syntax

```
-- Lingo syntax
memberObjRef.numChannels

// JavaScript syntax
memberObjRef.numChannels;
```

Beschreibung

Diese Shockwave Audio (SWA)-Darstellereigenschaft gibt die Anzahl der Kanäle im angegebenen SWA-Streaming-Darsteller zurück. Der Wert ist entweder 1 für Mono oder 2 für Stereo.

Diese Eigenschaft steht erst zur Verfügung, wenn die Wiedergabe des SWA-Streaming-Darstellers begonnen hat oder nachdem die Datei unter Verwendung des Befehls `preLoadBuffer` vorausgeladen wurde.

Diese Eigenschaft kann getestet, aber nicht eingestellt werden.

Beispiel

Im folgenden Beispiel wird dem Felddarsteller „Kanalanzeige“ die Anzahl der Soundkanäle des SWA-Streaming-Darstellers „Duke Ellington“ zugewiesen:

```
-- Lingo syntax
myVariable = member("Duke Ellington").numChannels
if myVariable = 1 then
    member("Channel Display").text = "Mono"
else
    member("Channel Display").text = "Stereo"
end if

// JavaScript syntax
var myVariable = member("Duke Ellington").numChannels;
if (myVariable == 1) {
    member("Channel Display").text = "Mono";
} else {
    member("Channel Display").text = "Stereo";
}
```

numParticles

Syntax

```
member(whichCastmember).modelResource(whichModelResource).emitter.numParticles
modelResourceObjectReference.emitter.numParticles
```

Beschreibung

Mit dieser 3D-Eigenschaft können Sie bei Modellressourcen vom Typ `#particle` die Eigenschaft `numParticles` des Partikelemitters der Ressource ermitteln und festlegen. Der Wert muss größer als 0 und kleiner oder gleich 100000 sein. Der Standardwert ist 1000.

Beispiel

Die folgende Anweisung legt die Anzahl von Partikeln auf 50000 im Darsteller „3Dobjects“ fest.

```
-- Lingo syntax
member("3Dobjects").modelResource("Particle01").emitter.numParticles = 50000
// JavaScript syntax
member("3Dobjects").getPropRef("modelResource",10).emitter.numParticles = 50000;
```

Siehe auch

emitter

numSegments

Syntax

```
member(whichCastmember).modelResource(whichModelResource).numSegments
```

Beschreibung

Mit dieser 3D-Eigenschaft können Sie bei einer Modellressource vom Typ `#cylinder` die Eigenschaft `numSegments` der Modellressource ermitteln und festlegen.

Die Eigenschaft `numSegments` bestimmt die Anzahl von Segmenten zwischen der Ober- und Unterseite des Zylinders. Diese Eigenschaft muss größer oder gleich dem Standardwert 2 sein.

Die Glätte der Zylinderoberfläche hängt vom Wert dieser Eigenschaft ab. Je größer der Wert, desto glatter ist die Zylinderoberfläche.

Beispiel

Die folgende Anweisung setzt die Eigenschaft `numSegments` der Modellressource "Cylinder01" auf 10.

```
-- Lingo syntax
member("3Dobjects").modelResource("Cylinder01").numSegments = 10

// JavaScript syntax
member("3Dobjects").getPropRef("modelResource",11).numSegments = 10;
```

obeyScoreRotation

Syntax

```
member(flashMember).obeyScoreRotation
```

Beschreibung

Diese Flash-Darstellereigenschaft kann auf `TRUE` oder `FALSE` gesetzt werden, um zu bestimmen, ob das Flash-Film-Sprite die Drehungsinformationen des Drehbuchs oder die ältere Eigenschaft „rotation“ der Flash-Elemente verwenden soll.

Diese Eigenschaft wird für alle vor Director-Version 7 erstellten Filme automatisch auf `FALSE` gesetzt, um bei allen Sprites, die diesen Flash-Darsteller enthalten, die alte Funktionalität der Eigenschaft „rotation“ beizubehalten.

Bei neuen Elementen, die in Version 7 oder später erstellt wurden, wird diese Eigenschaft automatisch auf `TRUE` gesetzt.

Bei `TRUE` wird die Darstellereigenschaft „rotation“ ignoriert und stattdessen die Drehungseinstellungen des Drehbuchs verwendet.

Beispiel

Das folgende Sprite-Skript setzt die Eigenschaft `obeyScoreRotation` des Darstellers `FlashObj` auf 1 (TRUE) und dreht dann das Sprite, das den Darsteller enthält, um 180°.

```
-- Lingo syntax
member("FlashObj").obeyScoreRotation = 1
sprite(1).rotation = sprite(1).rotation + 180

// JavaScript syntax
member("FlashObj").obeyScoreRotation = 1;
sprite(1).rotation = sprite(1).rotation + 180;
```

Siehe auch

`rotation`

optionDown

Syntax

```
-- Lingo syntax
_key.optionDown

// JavaScript syntax
_key.optionDown;
```

Beschreibung

Diese Tasteneigenschaft bestimmt, ob die Alt-Taste (Windows) bzw. die Wahltaste (Mac) gedrückt wird. Nur Lesen.

Wird die Alt- oder Wahltaste gedrückt, gibt die Eigenschaft `TRUE` zurück, andernfalls `FALSE`.

In Windows-Projektoren funktioniert `optionDown` bei gedrückter Alt-Taste nicht, es sei denn, es wird außer der Alt-Taste gleichzeitig noch eine normale Taste gedrückt. Vermeiden Sie die Verwendung von `optionDown`, wenn Sie beabsichtigen, den Film als Windows-Projektor zu erstellen. Falls nur erkannt zu werden braucht, ob die Zusatztaste gedrückt ist, sollten Sie lieber `controlDown` oder `shiftDown` verwenden.

Wenn auf dem Mac die Wahltaste gedrückt wird, ändert sich der `key`-Wert. Sie sollten deshalb stattdessen `keyCode` verwenden.

Beispiel

Die folgende Prozedur prüft, ob der Benutzer die Alt- oder Wahltaste drückt. Wenn ja, wird die Prozedur `doOptionKey` aufgerufen:

```
-- Lingo syntax
on keyDown
    if (_key.optionDown) then
        doOptionKey(_key.key)
    end if
end

// JavaScript syntax
function keyDown() {
    if (_key.optionDown) {
        doOptionKey(_key.key);
    }
}
```

Siehe auch

`controlDown`, `Taste`, `key`, `keyCode`, `shiftDown`

organizationName

Syntax

```
-- Lingo syntax
_player.organizationName

// JavaScript syntax
_player.organizationName;
```

Beschreibung

Diese eigenschaft enthält den während der Installation von Director eingegebenen Firmennamen. Nur Lesen.

Diese Eigenschaft ist nur in der Authoring-Umgebung verfügbar. Diese Eigenschaft kann in einem MIAW (Film in einem Fenster)-Werkzeug verwendet werden, dass die persönlichen Angaben eines Benutzers anzeigt.

Beispiel

Die folgende Prozedur würde sich in diesem Falle im Filmskript eines MIAW befinden. Die folgende Prozedur stellt den Benutzernamen und die Seriennummer in ein Anzeigefeld, wenn das Fenster geöffnet wird.

```
-- Lingo syntax
on prepareMovie
    displayString = _player.userName & RETURN & _player.organizationName & RETURN &
_player.serialNumber
    member("User Info").text = displayString
end

// JavaScript syntax
function prepareMovie() {
    var displayString = _player.userName + "\n" + _player.organizationName+ "\n" +
_player.serialNumber;
    member("User Info").text = displayString;
}
```

Siehe auch

Player

originalFont

Syntax

```
-- Lingo syntax
memberObjRef.originalFont

// JavaScript syntax
memberObjRef.originalFont;
```

Beschreibung

Diese Schriftdarstellereigenschaft gibt den genauen Namen der Originalschriftart zurück, die bei Erstellung des jeweiligen Darstellers importiert wurde.

Beispiel

Die folgende Anweisung zeigt den Namen der Schriftart an, die bei Erstellung von Darsteller 11 importiert wurde:

```
-- Lingo syntax
put(member(11).originalFont)

// JavaScript syntax
put(member(11).originalFont);
```

Siehe auch

recordFont, bitmapSizes, characterSet

originH

Syntax

```
-- Lingo syntax
memberOrSpriteObjRef.originH

// JavaScript syntax
memberOrSpriteObjRef.originH;
```

Beschreibung

Diese Darsteller- und Sprite-Eigenschaft steuert die in Pixel angegebene horizontale Koordinate des Ursprungspunkts eines Flash-Films oder einer Vektorform. Es kann hier ein Fließkommawert verwendet werden.

Der Ursprungspunkt in einem Flash-Film oder einer Vektorform ist die Koordinate, die als Ausgangspunkt für die Skalierung und Drehung verwendet wird. Der Ursprungspunkt kann als Fließkommawert eingestellt werden, und zwar über die separaten Eigenschaften `originH` und `originV`. Er kann auch über die Eigenschaft `originPoint` als Ganzzahl festgelegt werden.

Sie können `originH` nur dann einstellen, wenn die Eigenschaft `originMode` auf `#point` gesetzt ist.

Diese Eigenschaft kann getestet und eingestellt werden. Der Standardwert ist 0.

Hinweis: *Diese Eigenschaft muss auf den Standardwert gesetzt sein, wenn die Eigenschaft* `scaleMode` *auf* `#autoSize` *gesetzt ist. Andernfalls wird das Sprite nicht richtig angezeigt.*

Beispiel

Das folgende Sprite-Skript verwendet die Eigenschaft `originMode`, um ein Flash-Film-Sprite so zu konfigurieren, dass sein Ursprungspunkt auf einen bestimmten Punkt eingestellt werden kann. Anschließend stellt das Skript den horizontalen und vertikalen Ursprungspunkt ein.

```
-- Lingo syntax
property spriteNum

on beginSprite me
    sprite(spriteNum).originMode = #point
    sprite(spriteNum).originH = 100
    sprite(spriteNum).originV = 80
end

// JavaScript syntax
function beginSprite() {
    sprite(this.spriteNum).originMode = symbol("point");
    sprite(this.spriteNum).originH = 100;
    sprite(this.spriteNum).originV = 80;
}
```

Siehe auch

originV, originMode, originPoint, scaleMode

originMode

Syntax

```
-- Lingo syntax
memberOrSpriteObjRef.originMode

// JavaScript syntax
memberOrSpriteObjRef.originMode;
```

Beschreibung

Diese Darsteller- und Sprite-Eigenschaft legt wie folgt den Ursprungspunkt für die Skalierung und Drehung fest:

- `#center` (Standard) – Der Ursprungspunkt befindet sich in der Mitte des Flash-Films.
- `#topleft` – Der Ursprungspunkt befindet sich im Flash-Film oben links.
- `#point` – Der Ursprungspunkt wird durch die Eigenschaften `originPoint`, `originH` und `originV` bestimmt.

Diese Eigenschaft kann getestet und eingestellt werden.

Hinweis: *Diese Eigenschaft muss auf den Standardwert eingestellt sein, wenn die Eigenschaft* `scaleMode` *auf* `#autoSize` *gesetzt ist. Andernfalls wird das Sprite nicht richtig angezeigt.*

Beispiel

Das folgende Sprite-Skript verwendet die Eigenschaft `originMode`, um ein Flash-Film-Sprite so einzurichten, dass es auf einen bestimmten Ursprungspunkt eingestellt werden kann. Anschließend stellt das Skript den horizontalen und vertikalen Ursprungspunkt ein.

```
-- Lingo syntax
property spriteNum

on beginSprite me
    sprite(spriteNum).originMode = #point
    sprite(spriteNum).originH = 100
    sprite(spriteNum).originV = 80
end

// JavaScript syntax
function beginSprite() {
    sprite(this.spriteNum).originMode = symbol("point");
    sprite(this.spriteNum).originH = 100;
    sprite(this.spriteNum).originV = 80;
}
```

Siehe auch

originH, originV, originPoint, scaleMode

originPoint

Syntax

```
-- Lingo syntax
memberOrSpriteObjRef.originPoint

// JavaScript syntax
memberOrSpriteObjRef.originPoint;
```

Beschreibung

Diese Darsteller- und Sprite-Eigenschaft steuert den Ursprungspunkt, der in einem Flash-Film oder einer Vektorform als Ausgangspunkt für die Skalierung und Drehung verwendet wird.

Die Eigenschaft `originPoint` wird als Director-Punktwert angegeben, beispielsweise „point(100,200)“. Die Einstellung des Ursprungspunkts eines Flash-Films oder einer Vektorform über die Eigenschaft `originPoint` bewirkt das Gleiche wie die separate Einstellung der Eigenschaften `originH` und `originV`. Wenn Sie z. B. die Eigenschaft `originPoint` auf „point(50,75)“ setzen, ergibt sich die gleiche Wirkung, als wenn `originH` auf 50 und `originV` auf 75 gesetzt wird.

Die Director-Punktwerte für die Eigenschaft `originPoint` können nur als Ganzzahlen angegeben werden, während für `originH` und `originV` auch Fließkommazahlen möglich sind. Wenn Sie die Eigenschaft `originPoint` testen, werden die Punktwerte auf Ganzzahlen gekürzt. In der Regel werden die Eigenschaften `originH` und `originV` empfohlen, wenn es auf Genauigkeit ankommt, und die Eigenschaft `originPoint`, wenn Geschwindigkeit und leichte Handhabung wichtiger sind.

Die Eigenschaft `originPoint` kann nur dann eingestellt werden, wenn die Eigenschaft `originMode` auf `#point` gesetzt ist.

Diese Eigenschaft kann getestet und eingestellt werden. Der Standardwert ist 0.

Hinweis: *Diese Eigenschaft muss auf den Standardwert eingestellt sein, wenn die Eigenschaft* `scaleMode` *auf* `#autoSize` *gesetzt ist. Andernfalls wird das Sprite nicht richtig angezeigt.*

Beispiel

Das folgende Sprite-Skript verwendet die Eigenschaft `originMode`, um ein Flash-Film-Sprite so einzurichten, dass es auf einen bestimmten Ursprungspunkt eingestellt werden kann. Anschließend stellt das Skript dann den Ursprungspunkt ein.

```
-- Lingo syntax
property spriteNum

on beginSprite me
    sprite(spriteNum).scaleMode = #showAll
    sprite(spriteNum).originMode = #point
    sprite(spriteNum).originPoint = point(100, 80)
end

// JavaScript syntax
function beginSprite() {
    sprite(this.spriteNum).scaleMode = symbol("showAll");
    sprite(this.spriteNum).originMode = symbol("point");
    sprite(this.spriteNum).originPoint = point(100, 80);
}
```

Siehe auch

originH, originV, scaleMode

originV

Syntax

```
-- Lingo syntax
memberOrSpriteObjRef.originv

// JavaScript syntax
memberOrSpriteObjRef.originV;
```

Beschreibung

Diese Darsteller- und Sprite-Eigenschaft steuert die in Pixel angegebene vertikale Koordinate des Ursprungspunkts eines Flash-Films oder einer Vektorform, der als Ausgangspunkt für die Skalierung und Drehung dient. Es kann hier ein Fließkommawert verwendet werden.

Der Ursprungspunkt kann als Fließkommawert eingestellt werden, und zwar über die separaten Eigenschaften `originH` und `originV`. Er kann auch über die Eigenschaft `originPoint` als Ganzzahl festgelegt werden.

Sie können die Eigenschaft `originV` nur dann einstellen, wenn die Eigenschaft `originMode` auf `#point` gesetzt ist.

Diese Eigenschaft kann getestet und eingestellt werden. Der Standardwert ist 0.

Hinweis: *Diese Eigenschaft muss auf den Standardwert gesetzt sein, wenn die Eigenschaft* `scaleMode` *auf* `#autoSize` *gesetzt ist. Andernfalls wird das Sprite nicht richtig angezeigt.*

Beispiel

Das folgende Sprite-Skript verwendet die Eigenschaft `originMode`, um ein Flash-Film-Sprite so einzurichten, dass es auf einen bestimmten Ursprungspunkt eingestellt werden kann. Anschließend stellt das Skript den horizontalen und vertikalen Ursprungspunkt ein.

```
-- Lingo syntax
property spriteNum

on beginSprite me
    sprite(spriteNum).scaleMode = #showAll
    sprite(spriteNum).originMode = #point
    sprite(spriteNum).originH = 100
    sprite(spriteNum).originV = 80
end

// JavaScript syntax
function beginSprite() {
    sprite(this.spriteNum).scaleMode = symbol("showAll");
    sprite(this.spriteNum).originMode = symbol("point");
    sprite(this.spriteNum).originH = 100;
    sprite(this.spriteNum).originV = 80;
}
```

Siehe auch

originH, originPoint, scaleMode

orthoHeight

Syntax

```
member(whichCastmember).camera(whichCamera).orthoHeight
member(whichCastmember).camera[cameraindex].orthoHeight
sprite(whichSprite).camera.orthoHeight
```

Beschreibung

Wenn `camera.projection` auf `#orthographic` gesetzt ist, gibt die 3D-Eigenschaft `camera.orthoHeight` an, wie viele senkrechte Welteinheiten vertikal in das Sprite passen. Welteinheiten sind die Maßeinheiten für die jeweilige 3D-Welt. Sie sind zwar willkürlich gewählt, aber intern einheitlich und können von einer 3D-Welt zur anderen variieren.

Sie müssen den Kameraindex (*whichCamera*) nicht angeben, um auf die erste Kamera des Sprites zuzugreifen.

Der Standardwert dieser Eigenschaft lautet 200.0.

Beispiel

Folgende Anweisung setzt den Wert `orthoHeight` der Kamera von „Sprite 1“ auf 200. Das bedeutet, dass 200 Welteinheiten vertikal in den Sprite hineinpassen.

```
-- Lingo syntax
sprite(1).camera.orthoheight = 200.0

// JavaScript syntax
sprite(1).camera.orthoheight = 200.0;
```

Siehe auch

`projection`

overlay

Syntax

```
member(whichCastmember).camera(whichCamera).overlay[overlayIndex].propertyName
member(whichCastmember).camera(whichCamera).overlay.count
```

Beschreibung

Mit dieser 3D-Eigenschaft können Sie den Zugriff auf die Eigenschaften der Überlagerungen in der Überlagerungsliste der Kamera bestimmen und festlegen. In der Form `overlay.count` gibt diese Eigenschaft die Gesamtzahl von Überlagerungen in der Überlagerungsliste der Kamera zurück.

Überlagerungen sind Texturen, die vor allen Modellen angezeigt werden, die im Sichtkegel einer bestimmten Kamera erscheinen. Die Überlagerungen werden in der Reihenfolge gezeichnet, in der sie in der Überlagerungsliste der Kamera aufgeführt sind; der erste Eintrag in der Liste erscheint hinter, der letzte Eintrag vor allen anderen Überlagerungen.

Jede Überlagerung in der Überlagerungsliste der Kamera besitzt folgende Eigenschaften:

- Mit `loc` können Sie die Position des `regPoint` der Überlagerung bezogen auf die obere linke Ecke des Kamerarechtecks ermitteln und festlegen.
- Mit `source` können Sie die Textur, die als Quellgrafik für die Überlagerung dienen soll, ermitteln und festlegen.

- Mit `scale` können Sie den von der Überlagerung verwendeten Skalierungswert ermitteln und festlegen. Die Skalierung bestimmt die Vergrößerung der Überlagerung; der Standardwert dieser Eigenschaft ist 1.0.
- Mit `rotation` können Sie die Drehung der Überlagerung in Grad ermitteln und festlegen.
- Mit `regPoint` können Sie das Registrierungskreuz der Überlagerung bezogen auf die obere linke Ecke der Textur ermitteln und festlegen.
- Mit `blend` können Sie die Mischung der Überlagerung als Ganzzahl zwischen 0 und 100 ermitteln und festlegen. Dieser Wert gibt an, wie durchsichtig (0) oder undurchsichtig (100) die Überlagerung ist.

Beispiel

Die folgende Anweisung zeigt die Eigenschaft scale der ersten Überlagerung in der Überlagerungsliste der Sprite-Kamera an:

```
-- Lingo syntax
put sprite(2).camera.overlay[1].scale
```

Siehe auch

addOverlay, removeOverlay

pageHeight

Syntax

```
-- Lingo syntax
memberObjRef.pageHeight

// JavaScript syntax
memberObjRef.pageHeight;
```

Beschreibung

Diese Felddarstellereigenschaft gibt die Höhe des auf der Bühne sichtbaren Felddarstellerbereichs in Pixel zurück.

Diese Eigenschaft kann getestet, aber nicht eingestellt werden.

Beispiel

Die folgende Anweisung gibt die Höhe des sichtbaren Teils des Felddarstellers "Aktuelle Nachrichten“ zurück:

```
--Lingo syntax
trace(member("Today's News").pageHeight)

// JavaScript syntax
trace(member("Today's News").pageHeight);
```

palette

Syntax

```
-- Lingo syntax
memberObjRef.palette

// JavaScript syntax
memberObjRef.palette;
```

Beschreibung

Diese Darstellereigenschaft wird nur für Bitmapdarsteller verwendet und bestimmt, welche Palette dem in *whichCastMember* angegebenen Darsteller zugeordnet ist.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Die folgende Anweisung zeigt die dem Darsteller "Laub“ zugeordnete Palette im Nachrichtenfenster an.

```
-- Lingo syntax
put(member("Leaves").palette)

// JavaScript syntax
put(member("Leaves").palette);
```

paletteMapping

Syntax

```
-- Lingo syntax
_movie.paletteMapping

// JavaScript syntax
_movie.paletteMapping;
```

Beschreibung

Diese Filmeigenschaft bestimmt, ob der Film die Paletten für Darsteller, deren Paletten nicht der des aktuellen Films entspricht, neu zuordnet (`TRUE`) oder nicht (`FALSE`, Standard). Lesen/Schreiben.

Diese Eigenschaft hat die gleiche Auswirkung wie das Kontrollkästchen „Palette bei Bedarf anpassen“ im Dialogfeld „Filmeigenschaften“.

Um verschiedene Bitmaps mit unterschiedlichen Paletten gleichzeitig anzuzeigen, können Sie `paletteMapping` auf `TRUE` setzen. Director überprüft die Referenzpalette (d. h. die über das Dialogfeld Darstellereigenschaften zugeordnete Palette) jedes Darstellers auf der Bühne und sucht in der neuen Palette den passenden Wert für die einzelnen Pixel aus, falls sie sich von der aktuellen Palette unterscheidet.

Die nicht genau passenden Farben der Bitmap sind den Originalfarben trotzdem relativ ähnlich.

Die Neuzuordnung erfordert Verarbeitungszeit. Es ist daher zu empfehlen, die Bitmappalette im voraus entsprechend anzupassen.

Das Neuzuordnen kann in manchen Fällen auch zu unerwünschten Ergebnissen führen. Wenn sich die Palette z. B. in der Mitte eines Sprite-Einschlusses ändert, passt sich die Bitmap sofort der neuen Palette an und erscheint dann in den falschen Farben. Sobald der Bildschirm jedoch aktualisiert wird – z. B. durch einen Übergang oder ein Sprite, das sich über die Bühne bewegt – erscheint das betroffene Rechteck dann in den neu zugeordneten Farben.

Beispiel

Die folgende Anweisung sorgt dafür, dass die Filmpalette bei Bedarf angepasst wird:

```
-- Lingo syntax
_movie.paletteMapping = TRUE

// JavaScript syntax
_movie.paletteMapping = true;
```

Siehe auch

Movie

paletteRef

Syntax

```
member(whichCastMember). paletteRef
the paletteRef
```

Beschreibung

Diese Bitmapdarstellereigenschaft bestimmt die Palette, die mit einem Bitmapdarsteller verbunden ist. Integrierte Director-Paletten werden durch Symbole (wie z. B. `#systemMac`, `#rainbow` usw.) angegeben. Paletten, die Darsteller sind, werden als Darstellerbezüge behandelt. Dieses Verhalten unterscheidet sich von dem der Darstellereigenschaft `palette`, die eine positive Zahl für Besetzungspaletten und eine negative Zahl für integrierte Director-Paletten zurückgibt.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Die folgende Anweisung ordnet dem Bitmapdarsteller „Muschel“ die Mac-Systempalette zu:

```
member("Shell").paletteRef = #systemMac
```

pan

Syntax

```
-- Lingo syntax
soundChannelObjRef.pan

// JavaScript syntax
soundChannelObjRef.pan;
```

Beschreibung

Diese Soundkanaleigenschaft zeigt die linke/rechte Balance des Sounds an, der gegenwärtig in einem Soundkanal abgespielt wird. Lesen/Schreiben.

Die möglichen Werte liegen zwischen -100 und 100. -100 zeigt an, dass nur der linke Kanal zu hören ist. 100 bedeutet, dass nur der rechte Kanal zu hören ist. Der Wert 0 bedeutet, dass die linke/rechte Balance mittig eingestellt ist, so dass der Sound in beiden Kanälen gleich laut zu hören ist. Bei Mono-Sounds bestimmt `pan`, auf welchem Lautsprecher (links oder rechts) der Sound zu hören ist.

Sie können die Schwenkung eines Soundobjekts jederzeit ändern, aber wenn im Soundkanal gegenwärtig ein Ein-/Ausblendvorgang stattfindet, tritt die neue Schwenkeinstellung erst nach Abschluss des Ein-/Ausblendvorgangs in Kraft.

Ein Beispiel für `pan` in einem fertigen Film finden Sie im Film „Sound Control“ im Ordner „Learning/Lingo Examples“ im Director-Anwendungsordner.

Beispiel

Die folgenden Anweisungen schwenken den Sound in Soundkanal 2 vom linken auf den rechten Kanal:

```
-- Lingo syntax
repeat with x = -100 to 100
    sound(2).pan = x
end repeat

// JavaScript syntax
for (var x = -100; x <= 100; x++) {
    sound(2).pan = x;
}
```

Siehe auch

`Sound Channel`

pan (QTVR-Eigenschaft)

Syntax

```
-- Lingo syntax
spriteObjRef.pan

// JavaScript syntax
spriteObjRef.pan;
```

Beschreibung

Diese QuickTime VR-Sprite-Eigenschaft bestimmt den aktuellen Schwenkungsgrad des QuickTime VR-Films. Der Wert wird in Grad angegeben.

Diese Eigenschaft kann getestet und eingestellt werden.

panMatrix, toChannels und useMatrix (Mixer)

Syntax

```
mixerRef.toChannels
mixerRef.useMatrix
mixerRef.panMatrix
```

Beschreibung

Mixereigenschaften. Unter „Panning“ versteht man den Verlauf der Audiosignalstärke in verschiedenen Richtungen. Sie können ein Panning für die Stärke eines Audiosignals nicht nur über die bestehende Anzahl von Kanälen durchführen, sondern auch über eine zusätzliche Anzahl von Kanälen hinweg (bis zu 5.1 Kanälen insgesamt).

Beispielsweise lässt sich die Anzahl der Kanäle im Audiosignal von 2 auf 5.1 oder von 5.1 auf Mono ändern.

Die beim Panning unterstützten Audiokanäle sind Mono, Stereo, 2.1, 3.1, 4.1 und 5.1.

- 2.1 bedeutet linken Kanal, rechten Kanal und Niederfrequenzkanal (LHE, Tiefbass)
- 3.1 bedeutet linken Kanal, rechten Kanal, mittleren Kanal und Niederfrequenzkanal
- 4.1 bedeutet linken Kanal, rechten Kanal, linken Surrounkanal, rechten Surroundkanal und Niederfrequenzkanal
- 5.1 bedeutet linken Kanal, rechten Kanal, mittleren Kanal, linken Surrounkanal, rechten Surroundkanal und Niederfrequenzkanal

Übersicht: Eigenschaften

Eigenschaft	Beschreibung	Bereich	Standard
panMatrix	Ein zweidimensionales Array mit Panwerten panMatrix ist eine n X m-Matrix, wobei n die Anzahl der Kanäle der Audiodaten vor dem Panning und m die Anzahl der Kanäle in den Audiodaten nach dem Panning darstellt. panMatrix[i][j] liefert Informationen über die Mitwirkung des i. Kanal (Lautsprecher) des Eingangsaudio bei der Erstellung des j. Kanals (Lautsprecher) des Ausgangsaudio.	0-5	Der Wert beträgt entweder 0 oder 1, je nach Quell- oder Zielkanal.
toChannels	Anzahl der Kanäle beim Panning	1-6	0 (Kein Panning).
useMatrix	Dieses Kennzeichen ist standardmäßig gelöscht. Falls es aktiviert ist und Sie keine Panmatrix zur Verfügung stellen, wird eine Standardmatrix verwendet.	True/False	False

Beispiel

```
-- Lingo syntax
-- Panning  mixer output to 3.1 channels.
on mouseUp me
 gmixer = new(#mixer)   -- Creates a mixer.
 so = gmixer.createSoundObject("so1", member(4),[#loopcount:0])
 gmixer.toChannels = 4  -- Sets the channel property of the mixer to 4.
 gmixer.usematrix = true
 myPanMatrix = newMatrix(2,4)   -- Creates a 2x4 myPanmatrix. The stereo file is
-- panned to 3.1 channels.
  repeat with i = 1 to 2
  repeat with j = 1 to 4
   myPanMatrix.setval(i,j,1)  -- Updates all values of myPanmatrix to 1.
  end repeat
 end repeat
gmixer.panMatrix = myPanMatrix   -- Assigns myPanmatrix to the mixer's PanMatrix.

gmixer.play()
end
```

Siehe auch

Mixer

panMatrix, toChannels und useMatrix (Soundobjekte)

Syntax

```
soundObj.toChannels
soundObj.useMatrix
soundObj.panMatrix
```

Beschreibung

Soundobjekteigenschaften. Unter „Panning“ versteht man den Verlauf der Audiosignalstärke in verschiedenen Richtungen. Sie können ein Panning für die Stärke eines Audiosignals nicht nur über die bestehende Anzahl von Kanälen durchführen, sondern auch über eine zusätzliche Anzahl von Kanälen hinweg (bis zu 5.1 Kanälen insgesamt).

Beispielsweise lässt sich die Anzahl der Kanäle im Audiosignal von 2 auf 5.1 oder von 5.1 auf Mono ändern.

Die beim Panning unterstützten Audiokanäle sind Mono, Stereo, 2.1, 3.1, 4.1 und 5.1.

- 2.1 bedeutet linken Kanal, rechten Kanal und Niederfrequenzkanal (LHE, Tiefbass)
- 3.1 bedeutet linken Kanal, rechten Kanal, mittleren Kanal und Niederfrequenzkanal
- 4.1 bedeutet linken Kanal, rechten Kanal, linken Surrounkanal, rechten Surroundkanal und Niederfrequenzkanal
- 5.1 bedeutet linken Kanal, rechten Kanal, mittleren Kanal, linken Surrounkanal, rechten Surroundkanal und Niederfrequenzkanal

Übersicht: Eigenschaften

Eigenschaft	Beschreibung	Bereich	Standard
panMatrix	Ein zweidimensionales Array mit Panwerten panMatrix ist eine n X m-Matrix, wobei n die Anzahl der Kanäle der Audiodaten vor dem Panning und m die Anzahl der Kanäle in den Audiodaten nach dem Panning darstellt. panMatrix[i][j] liefert Informationen über die Mitwirkung des i. Kanal (Lautsprecher) des Eingangsaudio bei der Erstellung des j. Kanals (Lautsprecher) des Ausgangsaudio.	0-5	Der Wert beträgt entweder 0 oder 1, je nach Quell- oder Zielkanal.
toChannels	Anzahl der Kanäle nach dem Panning	1-6	0 (Kein Panning).
useMatrix	Dieses Kennzeichen ist standardmäßig gelöscht. Falls es aktiviert ist und Sie keine Panmatrix zur Verfügung stellen, wird eine Standardmatrix verwendet.	True/False	False

Hinweis: *Um ein korrektes Panning durchzuführen, soll der* `channelCount` *des Mixers größer oder gleich den* `tochannels` *für das Soundobjekt sein.*

Beispiel

```
-- Lingo syntax
-- Panning to 3.1 channels.
on mouseUp me
 gmixer = new(#mixer)   -- Creates a mixer.
 gmixer.channelCount = 4 -- Sets the channelCount of the mixer to 4. This channelCount
-- value should be greater than or equal to the value of the sound object's
-- toChannels property.
 so = gmixer.createsoundobject("so1", member(4),[#loopcount:0])
 so.toChannels = 4  -- Sets the channel property of the sound object to 4.
 so.usematrix = true
 myPanMatrix = newMatrix(2,4)   -- Creates a 2x4 myPanmatrix. The stereo file is panned
-- to 3.1 channels.
  repeat with i = 1 to 2
   repeat with j = 1 to 4
    myPanMatrix.setval(i,j,1)  -- Updates all values of myPanmatrix to 1.
   end repeat
  end repeat
 so.panMatrix = myPanMatrix   -- Assigns myPanmatrix to the sound object's panMatrix.

gmixer.play()
end
```

paragraph

Syntax

```
chunkExpression.paragraph[whichParagraph]
chunkExpression.paragraph[firstParagraph..lastParagraph]
```

Beschreibung

Diese Textdarsteller-Eigenschaft ist ein Chunk-Ausdruck, über den Sie auf verschiedene Absätze innerhalb eines Textdarstellers zugreifen können.

Ein Absatz wird durch ein Zeilenschaltungszeichen begrenzt.

```
put member("AnimText").paragraph[3]
```

Beispiel

Die folgende Anweisung gibt den zweiten Absatz des Textdarstellers `myText` zurück.

```
-- Lingo syntax
put member("myText").paragraph[2]
```

Siehe auch

`line...of`

parent

Syntax

```
member(whichCastmember).model(whichModel).parent
member(whichCastmember).camera(whichCamera).parent
member(whichCastmember).light(whichLight).parent
member(whichCastmember).group(whichGroup).parent
```

Beschreibung

Mit dieser 3D-Eigenschaft können Sie bei einem Modell-, Kamera-, Licht- oder Gruppenbezug den Parent-Node des referenzierten Objekts abrufen und einstellen. Der Parent-Node kann ein beliebiges anderes Modell-, Kamera-, Licht- oder Gruppenobjekt sein.

Die Objekteigenschaft `transform` definiert die Skalierung, Position und Ausrichtung des Objekts im Verhältnis zu seinem Parent-Objekt.

Wenn Sie bei einem Objekt die Eigenschaft „parent“ auf `VOID` setzen, wird das Objekt wie mit dem Befehl `removeFromWorld()` aus der Welt entfernt.

Wenn Sie die Eigenschaft „parent“ auf das Gruppenobjekt "World" setzen (`group("World")`), wird das Objekt wie mit dem Befehl `addToWorld()` zur Welt hinzugefügt.

Sie können den Wert dieser Eigenschaft auch mit dem Befehl `addChild` ändern.

Beispiel

Die folgende Anweisung legt die Eigenschaft parent des Modells „Reifen“ fest. Als Parent-Objekt wird das Modell "Auto" verwendet.

```
-- Lingo syntax
member("3Dobjects").model("Tire").parent = member("3Dobjects").model("Car")

// JavaScript syntax
member("3Dobjects").getPropRef("model",2).parent =
member("3Dobjects").getPropRef("model",3);
```

Siehe auch

child (3D), addChild

password

Syntax

```
-- Lingo syntax
memberOrSpriteObjRef.password

// JavaScript syntax
memberOrSpriteObjRef.password;
```

Beschreibung

Mit dieser Eigenschaft für `RealMedia`-Sprites und -Darsteller können Sie das Kennwort festlegen, das zum Zugriff auf einen geschützten `RealMedia`-Stream erforderlich ist. Aus Sicherheitsgründen können Sie mit dieser Eigenschaft kein bereits festgelegtes Kennwort abrufen. Wenn bereits ein Kennwort festgelegt ist, gibt diese Eigenschaft den String `"********"` zurück. Wenn noch kein Kennwort festgelegt ist, gibt diese Eigenschaft einen leeren String zurück.

Beispiel

Die folgenden Beispiele zeigen, dass für den `RealMedia`-Stream im Darsteller "Real" bzw. in Sprite 2 ein Kennwort festgelegt wurde:

```
-- Lingo syntax
put(sprite(2).password) -- "********"
put(member("Real").password) -- "********"

// JavaScript syntax
put(sprite(2).password); // "********"
put(member("Real").password); // "********"
```

Die folgenden Beispiele zeigen, dass für den `RealMedia`-Stream im Darsteller „Real“ bzw. in Sprite 2 noch kein Kennwort festgelegt wurde:

```
-- Lingo syntax
put(sprite(2).password) -- ""
put(member("Real").password) -- ""

// JavaScript syntax
put(sprite(2).password); // ""
put(member("Real").password); // ""
```

In den folgenden Beispielen wird das Kennwort für den `RealMedia`-Stream in Sprite 2 und im Darsteller „Real“ auf „abrakadabra“ gesetzt:

```
-- Lingo syntax
sprite(2).password = "abracadabra"
member("Real").password = "abracadabra"

// JavaScript syntax
sprite(2).password = "abracadabra";
member("Real").password = "abracadabra";
```

Siehe auch

userName (RealMedia)

path (Film)

Syntax

```
-- Lingo syntax
_movie.path

// JavaScript syntax
_movie.path;
```

Beschreibung

Diese Filmeigenschaft gibt den Pfadnamen des Ordners an, in dem der aktuelle Film gespeichert ist. Schreibgeschützt

Verwenden Sie bei Pfadnamen, die sowohl für Windows- als auch für Mac-Computer geeignet sind, den @-Pfadnamenoperator.

Ein Beispiel für `path` in einem fertigen Film finden Sie im Film „Read and Write Text“ im Ordner „Learning/Lingo_Examples“ im Director-Anwendungsordner.

Beispiel

Die folgende Anweisung zeigt den Pfadnamen des Ordners an, der den aktuellen Film enthält:

```
-- Lingo syntax
trace(_movie.path)

// JavaScript syntax
trace(_movie.path);
```

Die folgende Anweisung spielt die Sounddatei „Crash.aif“ ab, die im Unterordner „Sounds“ des aktuellen Filmordners gespeichert ist:

```
-- Lingo syntax
sound(1).playFile(_movie.path & "Sounds\Crash.aif")

// JavaScript syntax
sound(1).playFile(_movie.path + "Sounds\\Crash.aif");
```

Siehe auch

Movie

path (3D)

Syntax

```
member(whichCastmember).modelResource(whichModelResource).emitter.path
```

Beschreibung

Mit dieser 3D-Eigenschaft können Sie bei Modellressourcen vom Typ `#particle` die Eigenschaft `path` des Partikelemitters der Ressource ermitteln und festlegen.

Diese Eigenschaft ist eine Liste mit Vektoren, die den Pfad definieren, den Partikel über ihre Lebensdauer hinweg folgen. Der Standardwert dieser Eigenschaft ist eine leere Liste (`[]`).

Beispiel

In diesem Beispiel ist `ThermoSystem` eine Modellressource vom Typ `#particle`. Die folgende Anweisung gibt an, dass die Partikel von `ThermoSystem` dem durch die Vektorliste vorgegebenen Pfad folgen:

```
member("Fires").modelResource("ThermoSystem").emitter.path = [vector(0,0,0), vector(15,0,0),
vector(30,30,-10)]
```

Siehe auch

`pathStrength`, `emitter`

pathName (Flash-Darsteller)

Syntax

```
member(whichFlashMember).pathName
the pathName of member whichFlashMember
```

Beschreibung

Diese Darstellereigenschaft steuert die Position einer externen Datei, in der die Elemente eines Flash-Filmdarstellers gespeichert sind. Sie können einen Flash-Film mit jedem beliebigen Pfadnamen auf dem lokalen Laufwerk oder dem Netzwerklaufwerk und mit einer URL-Adresse verknüpfen.

Wenn Sie den Pfadnamen eines nicht verknüpften Darstellers angeben, wird dieser dadurch in einen verknüpften Darsteller konvertiert.

Diese Eigenschaft kann getestet und eingestellt werden. Der `pathName` eines nicht verknüpften Darstellers besteht aus einem leeren String.

Diese Eigenschaft hat die gleiche Wirkung wie die Eigenschaft `fileName` für andere Darstellertypen, d. h. `fileName` kann anstelle von `pathName` verwendet werden.

Beispiel

Die folgende Anweisung ändert die Einstellung `pathName` des Darstellers in den Speicherort eines Flash-Films im Internet.

```
-- Lingo syntax
member("FlashObj").pathName = "http://www.someURL.com/myFlash.swf"

// JavaScript syntax
member("FlashObj").pathName = "http://www.someURL.com/myFlash.swf";
```

Siehe auch

fileName (Darsteller), linked

pathStrength

Syntax

```
member(whichCastmember).modelResource(whichModelResource).emitter.pathStrength
```

Beschreibung

Bei Modellressourcen vom Typ `#particle` bestimmt diese 3D-Eigenschaft, wie eng sich die Partikel an den durch die Emittereigenschaft `path` vorgegebenen Pfad halten. Der Gültigkeitsbereich dieser Eigenschaft beginnt bei 0.0 (keine Stärke, d. h. die Partikel werden nicht vom Pfad angezogen) und läuft bis unendlich. Der Standardwert ist 0.1. Es ist sinnvoll, den Wert `pathStrength` auf 0,0 zu setzen, um den Pfad vorübergehend abzuschalten.

Je größer der Wert von `pathStrength` wird, desto „steifer“ wird das gesamte Partikelsystem. Bei großen `pathStrength` -Werten springen die Partikel sehr schnell herum, es sei denn, mit einer Partikeleigenschaft wie `drag` wird eine gewisse Dämpfungskraft ausgeübt.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

In diesem Beispiel ist `ThermoSystem` eine Modellressource vom Typ `#particle`. Folgende Anweisung setzt die Eigenschaft `pathStrength` von `ThermoSystem` auf 0,97. Pfade, die mit der `ThermoSystem`-Eigenschaft „emitter.path“ ausgezeichnet wurde, werden von den Partikeln sehr genau verfolgt.

```
member("Fires").modelResource("ThermoSystem").emitter.pathStrength = 0.97
```

Siehe auch

path (3D), emitter

pattern

Syntax

```
member(whichCastMember).pattern
the pattern of member whichCastMember
```

Beschreibung

Diese Darstellereigenschaft bestimmt das Füllmuster, das der angegebenen Form zugeordnet wird. Sie können für diese Eigenschaft die Werte verwenden, die den Feldern in der Füllmusterpalette des Werkzeugfensters entsprechen. Falls der Formdarsteller nicht gefüllt ist, wird das Füllmuster auf die Kontur des Darstellers angewendet.

Diese Eigenschaft kann in Shockwave-Filmen zur Grafikänderung sehr nützlich sein, da Sie auf diese Weise das in der Form angewandte Kachelmuster ändern und somit Speicherplatz sparen können, der sonst für größere Bitmaps erforderlich wäre.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Die folgenden Anweisungen wandeln den Formdarsteller `myShape` in eine gefüllte Form um und ordnen ihm dann das Muster 1 zu, bei dem es sich um eine Vollfarbe handelt.

```
-- Lingo syntax
member("myShape").filled = TRUE
member("myShape").pattern = 1

// JavaScript syntax
member("myShape").filled = 1;
member("myShape").pattern = 1;
```

pausedAtStart (Flash, Digitalvideo)

Syntax

```
member(whichFlashOrDigitalVideoMember).pausedAtStart
the pausedAtStart of member whichFlashOrDigitalVideoMember
```

Beschreibung

Diese Darstellereigenschaft bestimmt, ob das Digitalvideo oder der Flash-Film beim Erscheinen des Sprites auf der Bühne abgespielt werden soll oder nicht. Wenn diese Eigenschaft `TRUE` lautet, wird das Digitalvideo oder der Flash-Film beim Erscheinen des Sprites nicht abgespielt. Lautet diese Eigenschaft dagegen `FALSE`, wird beim Erscheinen des Sprites das Video oder der Film sofort abgespielt.

Bei einem Digitalvideo-Darsteller wird durch diese Eigenschaft angegeben, ob das Kontrollkästchen Angehalten beim Start im Dialogfeld Digitalvideo-Darstellereigenschaften mit einem Häkchen versehen ist oder nicht.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Die folgende Anweisung aktiviert das Kontrollkästchen „Angehalten beim Start“ im Dialogfeld „Digitalvideo-Darstellereigenschaften“ für den QuickTime-Film „Drehstuhl“:

```
member("Rotating Chair").pausedAtStart = TRUE
```

pausedAtStart (MP4Media/FLV)

Syntax

```
member(1).pausedAtStart = true
sprite(1).pausedAtStart = true
```

Beschreibung

Mit dieser MP4Media/FLV-Sprite- oder -Darstellereigenschaft können Sie festlegen, ob MP4Media/FLV automatisch abgespielt wird oder beim Starten des Films pausiert wird. Damit der Film automatisch abgespielt wird, muss der Wert für „MP4Media/FLV“ auf „False“ oder 0 gesetzt werden. Um den Film anzuhalten, muss der Wert auf „True“ oder 1 gesetzt werden.

Die Eigenschaft `pausedAtStart` lässt sich mit jedem Ausdruck definieren, der „True“ oder „False“ entspricht. Andere Ganzzahlen als 1 oder 0 werden als TRUE interpretiert. Die Standardeinstellung dieser Eigenschaft lautet FALSE.

Falls `pausedAtStart` auf „False“ gesetzt wird, muss der Benutzer im MP4Media/FLV-Viewer die Schaltfläche zur Wiedergabe klicken (bzw. eine im Film dafür vorgesehene Schaltfläche). Alternativ hierzu kann auch die Methode `play()` aufgerufen werden, um das Sprite auf der Bühne abzuspielen.

Diese Eigenschaft betrifft nur die besetzungsbasierte Wiedergabe. Die Wiedergabe im MP4Media/FLV-Viewer ist nicht betroffen.

Beispiele

Im folgenden Beispiel wird die Eigenschaft `pausedAtStart` von „Sprite 2“ und dem Darsteller „MP4Media“ auf „False“ gesetzt. In diesem Fall wird der MP4Media-Stream automatisch abgespielt.

```
-- Lingo syntax
put(sprite(2).pausedAtStart) -- 0
put(member("MP4Media/FLV").pausedAtStart) -- 0

// JavaScript syntax
put(sprite(2).pausedAtStart); // 0
put(member("MP4Media/FLV").pausedAtStart); // 0
```

Im folgenden Beispiel wird die Eigenschaft `pausedAtStart` von Sprite 2 und dem Darsteller „MP4Media“ auf „True“ gesetzt. In diesem Fall wird der MP4Media-Stream nicht automatisch abgespielt, falls nicht der Befehl `play` aufgerufen wird.

```
-- Lingo syntax
sprite(2).pausedAtStart = TRUE
member("MP4Media/FLV").pausedAtStart = TRUE

// JavaScript syntax
sprite(2).pausedAtStart = 1;
member("MP4Media/FLV").pausedAtStart = 1;
```

pausedAtStart (RealMedia, Windows Media)

Syntax

```
-- Lingo syntax
memberOrSpriteObjRef.pausedAtStart

// JavaScript syntax
memberOrSpriteObjRef.pausedAtStart;
```

Beschreibung

Mit dieser Eigenschaft für `RealMedia-` und Windows Media-Sprites und -Darsteller können Sie ermitteln und festlegen, ob die Wiedergabe eines `RealMedia-` oder Windows Media-Streams auf der Bühne nach Abschluss der Pufferung automatisch beginnt (`FALSE` bzw. 0) oder nicht (`TRUE` bzw. 1). Lesen/Schreiben.

Diese Eigenschaft kann auf einen Ausdruck gesetzt werden, der bei Auswertung TRUE oder FALSE ergibt. Andere Ganzzahlen als 1 oder 0 werden als TRUE interpretiert. Die Standardeinstellung dieser Eigenschaft lautet FALSE. Durch Auswahl des Kontrollkästchens „Pause“ in der Grafikansicht des Eigenschafteninspektors können Sie diese Eigenschaft auf TRUE setzen.

Ist diese Eigenschaft auf FALSE gesetzt, muss der Benutzer auf die Schaltfläche „Abspielen“ im RealMedia- bzw. Windows Media-Viewer klicken (bzw. auf eine Schaltfläche, die Sie zu diesem Zweck in Ihrem Film erstellt haben) oder Sie müssen die Methode play() aufrufen, um das Sprite nach Abschluss der Pufferung abzuspielen.

Diese Eigenschaft wirkt sich nur auf die drehbuchbasierte Wiedergabe aus, nicht auf die Wiedergabe im RealMedia- bzw. Windows Media-Viewer.

Beispiel

Die folgenden Beispiele zeigen, dass die Eigenschaft pausedAtStart für Sprite 2 und den Darsteller "Real" auf FALSE gesetzt ist, d. h. dass die Wiedergabe des RealMedia-Streams automatisch beginnt, sobald die Pufferung abgeschlossen ist.

```
-- Lingo syntax
put(sprite(2).pausedAtStart) -- 0
put(member("Real").pausedAtStart) -- 0

// JavaScript syntax
put(sprite(2).pausedAtStart); // 0
put(member("Real").pausedAtStart); // 0
```

In den folgenden Beispielen ist die Eigenschaft pausedAtStart für Sprite 2 und den Darsteller "Real" auf TRUE gesetzt, d. h. dass die Wiedergabe RealMedia nicht gespielt wird, es sei denn, der Befehl "Spielen" wird aufgerufen.

```
-- Lingo syntax
sprite(2).pausedAtStart = TRUE
member("Real").pausedAtStart = TRUE

// JavaScript syntax
sprite(2).pausedAtStart = 1;
member("Real").pausedAtStart = 1;
```

Im folgenden Beispiel wird mithilfe der Eigenschaft pausedAtStart ein RealMedia-Sprite außerhalb der Bühne gepuffert und anschließend auf der Bühne abgespielt. In diesem Beispiel wurde bei dem RealMedia-Darsteller die Eigenschaft pausedAtStart bereits auf TRUE gesetzt. Eine Instanz dieses Darstellers befindet sich außerhalb der Bühne in Sprite-Kanal 1. Das folgende Bildskript sollte in den Sprite-Einschluss gestellt werden:

```
-- Lingo syntax
on exitFrame me
    if sprite(1).state > 3 then -- check to see if buffering is complete
        sprite(1).locH = 162
        sprite(1).locV = 118
        sprite(1).play() -- position and play the sprite
    end if
end

// JavaScript syntax
function exitFrame() {
    var st = sprite(1).state;
    if (st > 3) { // check to see if buffering is complete
        sprite(1).locH = 162;
        sprite(1).locV = 118;
        sprite(1).play(); // position and play the sprite
    }
}
```

Das `RealMedia`-Sprite wird außerhalb der Bühne gepuffert. Nach Abschluss der Pufferung erscheint es auf der Bühne und wird dann sofort abgespielt.

percentBuffered

Syntax

```
-- Lingo syntax
memberOrSpriteObjRef.percentBuffered

// JavaScript syntax
memberOrSpriteObjRef.percentBuffered;
```

Beschreibung

Diese Eigenschaft für `RealMedia`-Sprites und -Darsteller gibt an, welchen Prozentsatz des Puffers der aus einer lokalen Datei bzw. vom Server heruntergeladene `RealMedia`-Stream bereits belegt. Wenn diese Eigenschaft den Wert 100 erreicht, ist der Puffer voll und die Wiedergabe des `RealMedia`-Streams beginnt, sofern die Eigenschaft `pausedAtStart` nicht auf `TRUE` gesetzt ist. Diese Eigenschaft ist während der Wiedergabe dynamisch und kann nicht eingestellt werden.

Der Puffer ist ein Zwischenspeicher, der den Filmabschnitt enthält, der als nächstes abgespielt wird (in der Regel handelt es sich dabei nur um einige Sekunden Filmmaterial). Der Stream wird beim Herunterladen der Daten in den Puffer gestellt und verlässt ihn, sobald RealPlayer den Clip abspielt. Der Puffer ermöglicht die Anzeige des Inhalts, ohne die gesamte Datei herunterladen zu müssen, und sorgt dafür, dass die Wiedergabe des Streams nicht durch Netzwerkprobleme beeinträchtigt wird.

Die Pufferung wird durch den Befehl `play` eingeleitet. Sobald der Puffer zu 100% voll ist, beginnt die Wiedergabe des zwischengespeicherten Streamabschnitts. Da die anfängliche Pufferung einige Sekunden dauern kann, kommt es zu einer Verzögerung zwischen dem Aufruf des Befehls `play` und der eigentlichen Wiedergabe des Streams. Dies lässt sich mit dem Befehl `pausedAtStart` verhindern: Beginnen Sie die Wiedergabe des Streams schon während der Pufferung außerhalb der Bühne und zeigen Sie den Stream dann bei Beginn der Wiedergabe auf der Bühne an. (Weitere Informationen hierzu finden Sie im Beispiel zu pausedAtStart (RealMedia, Windows Media).)

Beispiel

Die folgenden Beispiele zeigen, dass der `RealMedia`-Stream in Sprite 2 und im Darsteller „Real“ zu 56% gepuffert ist:

```
-- Lingo syntax
put(sprite(2).percentBuffered) -- 56
put(member("Real").percentBuffered) -- 56

// JavaScript syntax
put(sprite(2).percentBuffered); // 56
put(member("Real").percentBuffered); // 56
```

Siehe auch

`mediaStatus (RealMedia, Windows Media)`, `pausedAtStart (RealMedia, Windows Media)`, `state (RealMedia)`

percentPlayed

Syntax

```
member(whichCastMember).percentPlayed
the percentPlayed of member whichCastMember
```

Beschreibung

Diese Shockwave Audio (SWA)-Darstellereigenschaft gibt den bereits abgespielten Teil der angegebenen SWA-Datei in Prozent zurück.

Diese Eigenschaft kann nur getestet werden, wenn der SWA-Sound bereits läuft oder mit dem Befehl `preLoadBuffer` vorausgeladen wurde. Diese Eigenschaft kann nicht eingestellt werden.

Beispiel

Die folgende Prozedur zeigt den bereits abgespielten Prozentsatz des SWA-Streaming-Darstellers „Frank Sinatra“ an und stellt diesen Wert in den Felddarsteller „Prozent abgespielt“:

```
on exitFrame
    whatState = member("Frank Sinatra").state
    if whatState > 1 AND whatState < 9 then
        member("Percent Played").text = string(member("Frank Sinatra").percentPlayed)
    end if
end
```

Siehe auch

`percentStreamed (Darsteller)`

percentStreamed (3D)

Syntax

```
member(whichCastMember).percentStreamed
```

Beschreibung
Mit dieser 3D-Eigenschaft können Sie ermitteln, welcher Prozentsatz eines 3D-Darstellers bereits gestreamt wurde. Diese Eigenschaft bezieht sich entweder auf den ursprünglichen Dateiimport oder die letzte Dateiladeanforderung. Der Rückgabewert ist ein Integer zwischen 0 und 100. Diese Eigenschaft hat keinen Standardwert.

Beispiel
Die folgende Anweisung zeigt, dass der Darsteller `PartyScene` fertig geladen ist:

```
put member("PartyScene").percentStreamed
-- 100
```

percentStreamed (Darsteller)

Syntax

```
-- Lingo syntax
memberOrSpriteObjRef.percentStreamed

// JavaScript syntax
memberOrSpriteObjRef.percentStreamed;
```

Beschreibung
Eigenschaft für Shockwave Audio (SWA)- und Flash-Darsteller sowie für QuickTime-Sprites.

Bei SWA-Streaming-Sounds ermittelt diese Eigenschaft, wie viel Prozent einer SWA-Datei bereits von einem HTTP- oder FTP-Server heruntergeladen wurde. Bei SWA unterscheidet sich diese Eigenschaft darin von der Eigenschaft `percentPlayed`, dass sie die Dateimenge mit einschließt, die sich zwar bereits im Puffer befindet, aber noch nicht abgespielt wurde. Diese Eigenschaft kann nur getestet werden, wenn der SWA-Sound bereits läuft oder mit dem Befehl `preLoadBuffer` vorausgeladen wurde.

Bei Flash-Filmdarstellern ermittelt diese Eigenschaft, wie viel Prozent eines Flash-Films bereits in den Speicher geladen wurde.

Bei QuickTime-Sprites ermittelt diese Eigenschaft, wie viel Prozent der QuickTime-Datei bereits abgespielt wurde.

Diese Eigenschaft kann einen Wert zwischen 0 und 100 % aufweisen. Bei einer Datei auf einem lokalen Datenträger beträgt dieser Wert 100. Bei aus dem Internet heruntergeladenen Dateien nimmt der `percentStreamed`-Wert zu, je mehr Byte empfangen werden. Diese Eigenschaft kann nicht eingestellt werden.

Beispiel
Das folgende Beispiel zeigt den Prozentsatz des SWA-Streaming-Darstellers „Ray Charles“ an, der bereits gestreamt wurde, und stellt diesen Wert in ein Feld:

```
-- Lingo syntax
on exitFrame
    whatState = member("Ray Charles").state
    if whatState > 1 AND whatState < 9 then
        member("Percent Streamed Displayer").text = string(member("Ray Charles").percentStreamed)
    end if
end

// JavaScript syntax
function exitFrame() {
    var whatState = member("Ray Charles").state;
    var pcStm = new String(member("Ray Charles").percentStreamed);
    if (whatState > 1 && whatState < 9) {
        member("Percent Streamed Displayer").text = pcStm;
    }
}
```

Das folgende Bild-Skript lässt den Abspielkopf im aktuellen Bild solange in Schleife abspielen, bis 60 % des Flash-Films „Splash-Bildschirm“ in den Speicher geladen wurden:

```
-- Lingo syntax
on exitFrame
    if member("Splash Screen").percentStreamed < 60 then
        _movie.go(_movie.frame)
    end if
end

// JavaScript syntax
function exitFrame() {
    var ssStrm = member("Splash Screen").percentStreamed;
    if (ssStrm < 60) {
        _movie.go(_movie.frame);
    }
}
```

Siehe auch

`percentPlayed`

percentStreamed (MP4Media/FLV)

Syntax

```
sprite(1).percentStreamed
member(1).percentStreamed
```

Beschreibung

Diese MP4Media/FLV- oder Sprite-Eigenschaft stellt die Datenmenge dar, die für ein Video von einer RTMP-URL gestreamt wird. `percentStreamed` ist ein Prozentwert der insgesamt gestreamten Datenmenge.

Diese Eigenschaft ist schreibgeschützt.

Beispiel

```
-- Lingo syntax
put(sprite("Mp4MediaSprite").percentStreamed
put(member("MP4Media/FLV").percentStreamed

// JavaScript syntax
put(sprite("Mp4MediaSprite").percentStreamed
put(member("MP4Media/FLV").percentStreamed
```

percentStreamed (Soundobjekt)

Syntax

```
soundobj.percentStreamed (Read-only)
```

Beschreibung

Diese Soundobjekteigenschaft gibt die Dauer der gestreamten Daten als Prozentsatz der Gesamtdauer der zu streamenden Datei aus.

Diese Eigenschaft ist schreibgeschützt.

Beispiel

Die folgenden Beispiele geben den Prozentsatz der für das Soundobjekt gestreamten Daten aus:

```
--Lingo syntax
on mouseUp me
put SoundObjectRef.percentStreamed
end

// JavaScript syntax
function mouseUp(){
put (SoundObjectRef.percentStreamed);
}
```

period

Syntax

```
timeoutObject.period
```

Beschreibung

Diese Objekteigenschaft gibt an, wie viele Millisekunden zwischen den durch `timeOutObject` an die Timeout-Prozedur gesendeten Timeout-Ereignissen verstreichen.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Die folgende Timeout-Prozedur reduziert die Zeitüberschreitung (`period`) bei jedem Aufruf um jeweils eine Sekunde, bis eine Mindestdauer von 2 Sekunden (2000 Millisekunden) erreicht ist:

```
on handleTimeout timeoutObject
    if timeoutObject.period > 2000 then
        timeoutObject.period = timeoutObject.period - 1000
    end if
end handleTimeout
```

Siehe auch

name (timeout), persistent, target, time (Timeout-Objekt), timeout(), timeoutHandler, timeoutList

persistent

Syntax

```
timeoutObject.persistent
```

Beschreibung

Diese Objekteigenschaft bestimmt, ob das angegebene *timeoutObject* aus `the timeoutList` entfernt wird, wenn die Wiedergabe des aktuellen Films gestoppt wird. Bei `TRUE` bleibt das *timeoutObject* aktiv. Bei `FALSE` wird das Timeout-Objekt gelöscht, wenn die Wiedergabe des Films gestoppt wird. Der Standardwert ist `FALSE`.

Wenn Sie diese Eigenschaft auf `TRUE` setzen, kann ein Timeout-Objekt Timeout-Ereignisse in anderen Filmen erzeugen. Dies ist besonders dann nützlich, wenn ein Film mit dem Befehl `go to movie` auf einen anderen Film verzweigt.

Beispiel

Die folgende Anweisung erstellt ein Timeout-Objekt und kennzeichnet es als persistent.

```
-- Lingo syntax
gTO = timeout().new("test",50000,"sampleTimeout",0)
gTO.persistent = TRUE

// JavaScript syntax
_global.gTO = new timeout("test",50000,"sampleTimeout",0)
_global.gTO.persistent = 1;
```

Siehe auch

name (timeout), period, target, time (Timeout-Objekt), timeout(), timeoutHandler, timeoutList

picture (Darsteller)

Syntax

```
-- Lingo syntax
memberObjRef.picture

// JavaScript syntax
memberObjRef.picture;
```

Beschreibung

Diese Darstellereigenschaft bestimmt, welche Grafik mit einem Bitmap-, Text- oder PICT-Darsteller verbunden ist. Verwenden Sie folgende Anweisung, um die am Registrierungskreuz eines Darstellers vorgenommenen Änderungen zu aktualisieren oder um Grafikänderungen zu aktualisieren, nachdem die Grafik mithilfe der Eigenschaft `fileName` neu verknüpft wurde:

```
member(whichCastMember).picture = member(whichCastMember).picture
```

In dieser Anweisung muss *whichCastMember* durch den Namen oder die Nummer des betreffenden Darstellers ersetzt werden.

Da Darstelleränderungen im RAM gespeichert werden, sollte diese Eigenschaft am besten bei der Anwendungserstellung verwendet werden. Vermeiden Sie diese Eigenschaft in Projektoren.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Die folgende Anweisung setzt die Variable `pictHolder` auf die Grafik im Darsteller "Sonnenuntergang":

```
-- Lingo syntax
pictHolder = member("Sunset").picture

// JavaScript syntax
var pictHolder = member("Sunset").picture;
```

Siehe auch

`type (sprite)`

picture (Fenster)

Syntax

```
-- Lingo syntax
windowObjRef.picture

// JavaScript syntax
windowObjRef.picture;
```

Beschreibung

Mit dieser Fenstereigenschaft können Sie ein Bild des aktuellen Fensterinhalts abrufen (d. h. ein Bild des Bühnenfensters oder eines Films in einem Fenster (MIAW)). Nur Lesen.

Sie können die daraus resultierenden Bitmapdaten auf eine vorhandene Bitmap anwenden oder daraus eine neue Bitmap erstellen.

Ist kein Bild vorhanden, gibt diese Eigenschaft `VOID` (Lingo) oder `null` (JavaScript-Syntax) zurück.

Beispiel

Die folgende Anweisung erfasst den aktuellen Bühneninhalt und stellt ihn in einen Bitmapdarsteller:

```
-- Lingo syntax
member("Stage image").picture = _movie.stage.picture

// JavaScript syntax
member("Stage image").picture = _movie.stage.picture;
```

Siehe auch

Window

platform

Syntax

```
the platform
```

Beschreibung

Diese Systemeigenschaft zeigt die Plattform an, für die der Projektor erstellt wurde.

Diese Eigenschaft kann getestet, aber nicht eingestellt werden.

Folgende Werte sind möglich:

Möglicher Wert	Entsprechende Plattform
`Mac,PowerPC`	`PowerPC` Mac
`Windows,32`	Windows 95 oder Windows NT

Um für Aufwärtskompatibilität zu sorgen und das Hinzufügen von Werten zu ermöglichen, ist zu empfehlen, die Plattform mithilfe von `contains` zu testen.

Beispiel

Die folgende Anweisung gibt die Plattform zurück, auf der der Film erstellt wurde.

```
-- Lingo syntax
if the platform contains "Windows,32" then
    alert ("This movie has been created using Windows")
end if
```

Siehe auch

runMode

playBackMode

Syntax

```
-- Lingo syntax
memberOrSpriteObjRef.playBackMode

// JavaScript syntax
memberOrSpriteObjRef.playBackMode;
```

Beschreibung

Diese Darsteller- und Sprite-Eigenschaft bestimmt das Tempo eines Flash-Filmdarstellers oder eines animierten GIF-Darstellers, und zwar mithilfe folgender Werte:

- `#normal` (*Standardeinstellung*) – Spielt den Flash-Film oder die GIF-Datei möglichst im Originaltempo ab.
- `#lockStep` – Spielt den Flash-Film oder die GIF-Datei Bild für Bild mit dem Director-Film ab.
- `#fixed`– Spielt den Flash-Film oder die GIF-Datei mit der durch die Eigenschaft `fixedRate` angegebenen Geschwindigkeit ab.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Das folgende Sprite-Skript stellt die Bildrate für das Flash-Film-Sprite so ein, dass sie mit der Bildrate des Director-Films übereinstimmt:

```
-- Lingo syntax
property spriteNum

on beginSprite(me)
    sprite(spriteNum).playBackMode = #lockStep
end

// JavaScript syntax
function beginSprite() {
    sprite(this.spriteNum).playBackMode = symbol("lockStep");
}
```

Siehe auch

`fixedRate`

playing

Syntax

```
-- Lingo syntax
spriteObjRef.playing

// JavaScript syntax
spriteObjRef.playing;
```

Beschreibung

Diese Flash-Sprite-Eigenschaft gibt an, ob der Flash-Film abgespielt wird (`TRUE`) oder gestoppt ist (`FALSE`).

Diese Eigenschaft kann getestet, aber nicht eingestellt werden.

Beispiel

Das folgende Bildskript prüft, ob das Flash-Film-Sprite in Kanal 5 abgespielt wird. Falls das nicht der Fall ist, wird der Film durch das Skript gestartet:

```
-- Lingo syntax
on enterFrame
    if not sprite(5).playing then
        sprite(5).play()
    end if
end

// JavaScript syntax
function enterFrame() {
    var plg = sprite(5).playing;
    if (plg == 0) {
        sprite(5).play();
    }
}
```

playing (3D)

Syntax

```
member(whichCastmember).model(whichModel).keyframePlayer.playing
member(whichCastmember).model(whichModel).bonesPlayer.playing
```

Beschreibung

Diese 3D-Eigenschaft für die Modifizierer `#keyframePlayer` und `#bonesPlayer` gibt an, ob die Animationswiedergabe-Engine des Modifizierers läuft (`TRUE`) oder angehalten ist (`FALSE`).

Diese Eigenschaft kann getestet, aber nicht eingestellt werden.

Beispiel

Die folgende Anweisung zeigt an, dass die Animationswiedergabe-Engine des Modifizierers `#keyframePlayer` für das Modell "Kreatur3" gegenwärtig ausgeführt wird.

```
put member("newaliens").model("Alien3").keyframePlayer.playing
-- 1
```

Siehe auch

`play() (3D)`, `pause() (3D)`, `playlist`, `queue() (3D)`

playlist

Syntax

```
member(whichCastmember).model(whichModel).keyframePlayer.playlist
member(whichCastmember).model(whichModel).bonesPlayer.playlist
```

Beschreibung

Diese 3D-Eigenschaft für die Modifizierer `#keyframePlayer` und `#bonesPlayer` gibt eine lineare Listen mit Eigenschaftslisten zurück, von denen jede einer Bewegung entspricht, die auf Wiedergabe durch den Modifizierer wartet.

Jede Eigenschaftsliste enthält folgende Eigenschaften:

- `#name` ist der Name der abzuspielenden Bewegung.
- `#loop` gibt an, ob die Bewegung kontinuierlich wiederholt werden soll.
- `#startTime` ist die Zeit (in Millisekunden), zu der die Wiedergabe der Animation beginnen soll.
- `#endTime` ist die Zeit (in Millisekunden), zu der die Wiedergabe der Animation enden bzw. die kontinuierliche Wiederholung der Bewegung (Looping bzw. Schleifenbetrieb) beginnen soll. Bei einem negativen Wert wird die Bewegung bis zum Ende abgespielt.
- `#scale` ist der Faktor, mit dem die vom Modifizierer verwendete Eigenschaft `playRate` multipliziert wird, um die tatsächliche Wiedergabegeschwindigkeit der Bewegung zu bestimmen.

Die Eigenschaft `playlist` kann getestet, aber nicht eingestellt werden. Sie kann mit den Befehlen `queue()`, `play()`, `playNext()` und `removeLast()` manipuliert werden.

Beispiel

Die folgende Anweisung zeigt die Bewegungen für das Modell „Spaziergänger“, die gegenwärtig auf die Ausführung warten, im Nachrichtenfenster an.

```
-- Lingo syntax
put member("3Dobjects").model("Stroller").bonesPlayer.playList
// JavaScript syntax
put(member("3Dobjects").getPropRef("model",2).bonesPlayer.playList);
```

Siehe auch

`play() (3D)`, `playNext() (3D)`, `removeLast()`, `queue() (3D)`

playRate (3D)

Syntax

```
member(whichCastmember).model(whichModel).bonesPlayer.playRate
member(whichCastmember).model(whichModel).keyframePlayer.playRate
```

Beschreibung

Diese 3D-Eigenschaft für die Modifizierer `#keyframePlayer` und `#bonesPlayer` fungiert als Skalierungsmultiplikator für die lokale Zeit der abgespielten Bewegungen. Die Geschwindigkeit, mit der Bewegungen vom Modell ausgeführt werden, wird durch diese Eigenschaft nur zum Teil bestimmt.

Die Wiedergabe einer Bewegung durch ein Modell wird durch den Befehl `play()` oder `queue()` ausgelöst. Der Parameter `scale` des Befehls `play()` oder `queue()` wird mit der Eigenschaft `playRate` des Modifizierers multipliziert; das Produkt ist die Geschwindigkeit, mit der die jeweilige Bewegung abgespielt wird.

Beispiel

Die folgende Anweisung setzt die Eigenschaft `playRate` des Modifizierers `keyframePlayer` für das Modell `GreenAlien` auf 3:

```
member("newAliens").model("GreenAlien").keyframePlayer.playRate = 3
```

Siehe auch

`play() (3D)`, `queue() (3D)`, `playlist`, `currentTime (3D)`

playRate (DVD)

Syntax

```
-- Lingo syntax
dvdObjRef.playRate

// JavaScript syntax
dvdObjRef.playRate;
```

Beschreibung

Diese DVD-Eigenschaft gibt die Geschwindigkeit an, mit der eine DVD ab der aktuellen Position vorwärts oder rückwärts wiedergegeben wird. Lesen/Schreiben.

Ein negativer Wert bewirkt, dass die DVD rückwärts läuft, während ein positiver Wert die DVD vorwärts wiedergibt.

Siehe auch

DVD

playRate (QuickTime, AVI, MP4, FLV)

Syntax

```
-- Lingo syntax
spriteObjRef.playRate

// JavaScript syntax
spriteObjRef.playRate;
```

Beschreibung

Diese Digitalvideo-Sprite-Eigenschaft steuert die Geschwindigkeit, mit der ein Digitalvideo in einem bestimmten Kanal abgespielt wird.

Die Wiedergabe des Digitalvideos wird durch die Filmgeschwindigkeit bestimmt. Der Wert 1 legt den normalen Abspielmodus (vorwärts) fest, -1 bedeutet Rücklauf und 0 gibt an, dass das Video pausiert wird. Mit Zwischenwerten lassen sich schnellere oder langsamere Abspielweisen festlegen. Der Wert 0,5 erzeugt eine doppelt so langsame Abspielgeschwindigkeit des Videos. Der Wert 2 erzeugt eine doppelt so schnelle Abspielgeschwindigkeit des Videos. Bilder können ausgelassen werden, wenn die Eigenschaft `playRate` den Wert 1 überschreitet. Das Ausmaß dieser Auslassungen hängt auch davon ab, ob das Digitalvideosprite gestreckt ist, und wie leistungsfähig der Computer ist, auf dem das Video abgespielt wird.

Hinweis: *Für MP4Media-Darsteller liegen gültige Werte für* `playRate` *zwischen 0 und 4. Negative Werte für* `playRate` *werden für MP4Media-Darsteller nicht unterstützt. Für Quicktime und AVI-Darsteller lassen sich Videos rückwärts abspielen, indem negative* `playRate`*-Werte angegeben werden.*

Beispiel

Die folgende Anweisung stellt die Geschwindigkeit für ein Digitalvideo in Sprite-Kanal 9 auf die normale Abspielgeschwindigkeit ein:

```
-- Lingo syntax
sprite(9).playRate = 1

// JavaScript syntax
sprite(9).playRate = 1;
```

Die folgende Anweisung bewirkt, dass das Digitalvideo in Sprite-Kanal 9 rückwärts abgespielt wird:

```
-- Lingo syntax
sprite(9).playRate = -1

// JavaScript syntax
sprite(9).playRate = -1;
```

Siehe auch

duration (Darsteller), currentTime (QuickTime, AVI)

playRate (Soundobject)

Syntax

```
soundobj.playRate
```

Beschreibung

Diese Soundobjekteigenschaft ändert die Wiedergabegeschwindigkeit. Diese Eigenschaft hat außerdem Auswirkungen auf die Audiotonhöhe.

`playRate` bestimmt die erforderliche Audiogeschwindigkeit, mit der die Wiedergabe im Vergleich zur ursprünglichen Audiogeschwindigkeit erfolgen soll Die Granularität der Wiedergabegeschwindigkeit ist 0,01.

`playRate` kann Werte zwischen 0,25 und 4 annehmen. Der Standardwert ist 1.

Beispiel

```
on mouseUp me
soundobjectref.playrate = 0.50  --  Changes the play rate of the sound object to 0.50.
end

// JavaScript syntax
function mouseUp(){
soundobjectRef.playrate = 0.25;  // Changes the play rate of the sound object to 0.25.
}
```

playRate (Windows Media)

Syntax

```
-- Lingo syntax
windowsMediaObjRef.playRate

// JavaScript syntax
windowsMediaObjRef.playRate;
```

Beschreibung
Diese Windows Media-Eigenschaft bestimmt die Wiedergabegeschwindigkeit eines Windows Media-Darstellers. Nur Lesen.

Beispiel
Die folgende Anweisung zeigt die Wiedergabegeschwindigkeit von Darsteller 10 im Nachrichtenfenster an:

```
-- Lingo syntax
trace(member(10).playRate)

// JavaScript syntax
trace(member(10).playRate);
```

Siehe auch
Windows Media

pointAtOrientation

Syntax
```
member(whichCastmember).model(whichModel).pointAtOrientation
member(whichCastmember).group(whichGroup).pointAtOrientation
member(whichCastmember).light(whichLight).pointAtOrientation
member(whichCastmember).camera(whichCamera).pointAtOrientation
```

Beschreibung
Mit dieser 3D-Eigenschaft für Modelle, Lichtquellen und Kameras können Sie ermitteln und festlegen, wie das referenzierte Objekt auf den Befehl `pointAt` reagiert. Diese Eigenschaft ist eine lineare Liste aus zwei objektbezogenen Vektoren. Der erste Vektor in der Liste bestimmt, welche Richtung als die Vorwärtsrichtung des Objekts gilt, der zweite, welche Richtung als die Aufwärtsrichtung des Objekts gilt.

Die Vorwärts- und Aufwärtsrichtungen brauchen zwar nicht senkrecht zueinander zu verlaufen, dürfen aber auch nicht parallel sein.

Beispiel
Die folgende Anweisung zeigt die objektbezogenen Vorwärts- und Aufwärtsrichtungsvektoren des Modells „bip01“ an:

```
put member("scene").model("bip01").pointAtOrientation
-- [vector(0.0000, 0.0000, 1.0000), vector(0.0000, 1.0000, 0.0000)]
```

Siehe auch
pointAt

pointOfContact

Syntax
```
collisionData.pointOfContact
```

Beschreibung

Diese 3D-Eigenschaft für `collisionData`-Objekte gibt einen Vektor zurück, der den Berührungspunkt zwischen zwei Modellen bei einer Kollision beschreibt.

Das `collisionData`-Objekt wird bei den Ereignissen `#collideWith` und `#collideAny` als Argument an die in den Befehlen `registerForEvent, registerScript,` und `setCollisionCallback` angegebene Prozedur übergeben.

Die Ereignisse `#collideWith` und `#collideAny` werden gesendet, wenn eine Kollision zwischen Modellen stattfindet, an denen der Modifizierer "collision" angebracht ist. Die Eigenschaft `resolve` der Modifizierer des Modells muss auf `TRUE` gesetzt sein.

Diese Eigenschaft kann getestet, aber nicht eingestellt werden.

Beispiel

Dieses Beispiel setzt sich aus zwei Teilen zusammen. Der erste Teil ist die erste Codezeile, die die Prozedur `#explode` für das Ereignis `#collideAny`. registriert. Der zweite Teil ist die Prozedur `#explode`. Wenn zwei Modelle im Darsteller `MyScene` zusammenstoßen, wird die Prozedur `#explode` aufgerufen und das Argument `collisionData` an sie übergeben. Die ersten neun Zeilen der Prozedur `#explode` erstellen die Modellressource `SparkSpource` und legen ihre Eigenschaften fest. Bei dieser Modellressource handelt es sich um einen einzigen Partikelstrahl. Die zehnte Zeile der Prozedur erstellt ein Modell namens `SparksModel` mithilfe der Modellressource `SparkSource`. Die letzte Zeile der Prozedur setzt die Position von `SparksModel` auf die Stelle, an der die Kollision stattfand. Der Gesamteffekt ist eine durch eine Kollision verursachte Funkenexplosion.

```
member("MyScene").registerForEvent(#collideAny, #explode, 0)

on explode me, collisionData
    nmr = member("MyScene").newModelResource("SparkSource",#particle)
    nmr.emitter.mode = #burst
    nmr.emitter.loop = 0
    nmr.emitter.minSpeed = 30
    nmr.emitter.maxSpeed = 50
    nmr.emitter.direction = vector(0, 0, 1)
    nmr.colorRange.start = rgb(0, 0, 255)
    nmr.colorRange.end = rgb(255, 0, 0)
    nmr.lifetime = 5000
    nm = member("MyScene").newModel("SparksModel", nmr)
    nm.transform.position = collisionData.pointOfContact
end
```

Siehe auch

`modelA`, `modelB`

position (byte array)

Syntax

```
byteArrayObject.position
```

Beschreibung

Diese Bytearrayeigenschaft bewegt den Dateizeiger im `ByteArray`-Objekt oder gibt dessen aktuelle Position in Byte zurück.

Beispiele

```
--Lingo syntax
bArray = byteArray("Sample ByteArray")
put bArray.position
bArray.position=1

//JavaScript syntax
bArray = byteArray("Sample ByteArray");
put(bArray.position);
bArray.position=1;
```

position (Transformation)

Syntax

```
member(whichCastmember).node(whichNode).transform.position
member(whichCastmember).node(whichNode).getWorldTransform().position
transform.position
```

Beschreibung

Mit dieser 3D-Eigenschaft können Sie die Positionskomponente einer Transformation abrufen und einstellen. Eine Transformation definiert eine Skalierung, Position und Drehung in einem bestimmten Bezugsrahmen. Der Standardwert dieser Eigenschaft lautet `vector(0,0,0)`.

Ein Node kann ein Kamera-, Gruppen-, Licht- oder Modellobjekt sein. Durch Festlegung der Eigenschaft `position` der Transformation eines Node wird die Position dieses Objekts im Bezugsrahmen der Transformation definiert. Wenn Sie mit `getWorldTransform().position` die Eigenschaft „position“ der weltbezogenen Transformation eines Objekts festlegen, wird die Position des Objekts bezogen auf den Weltursprung definiert. Wenn Sie mit `transform.position` die Eigenschaft position der Parent-bezogenen Transformation eines Objekts festlegen, wird die Position des Objekts bezogen auf den Parent-Node definiert.

Die Eigenschaft `worldPosition` eines Modell-, Licht-, Kamera- oder Gruppenobjekts ist eine Kurzform für `getWorldTransform().position`.

Beispiel

Die folgende Anweisung zeigt die parentbezogene Position des Modells „Sphere01“ an.

```
-- Lingo syntax
    put(member("3Dobjects").model("Sphere01").transform.position)

// JavaScript syntax
    put(member("3Dobjects").getPropRef("model",2).transform.position);
```

Siehe auch

transform (Eigenschaft), getWorldTransform(), rotation (Transformation), scale (Transformation)

positionReset

Syntax

```
member(whichCastmember).model(whichModel).bonesPlayer.positionReset
member(whichCastmember).model(whichModel).keyframePlayer.positionReset
```

Beschreibung

Diese 3D-Eigenschaft für die Modifizierer `keyframePlayer` und `bonesPlayer` gibt an, ob das Modell nach Abschluss einer Bewegung an seine Ausgangsposition zurückkehrt (`TRUE`) oder nicht (`FALSE`).

Der Standardwert dieser Eigenschaft lautet `TRUE`.

Beispiel

Die folgende Anweisung verhindert, dass das Modell „Monster“ nach Ausführung einer Bewegung an seine Ausgangsposition zurückkehrt:

```
member("NewAlien").model("Monster").keyframePlayer.positionReset = FALSE
```

Siehe auch

`currentLoopState`

posterFrame

Syntax

```
-- Lingo syntax
memberObjRef.posterFrame

// JavaScript syntax
memberObjRef.posterFrame;
```

Beschreibung

Diese Flash-Darstellereigenschaft bestimmt, welches Bild eines Flash-Filmdarstellers für das zugehörige Piktogramm verwendet wird. Diese Eigenschaft gibt eine Ganzzahl an, die einer Bildnummer im Flash-Film entspricht.

Diese Eigenschaft kann getestet und eingestellt werden. Der Standardwert ist 1.

Beispiel

Für die folgende Prozedur können als Parameter ein Bezug auf einen Flash-Filmdarsteller und eine Bildnummer angegeben werden. Die Prozedur setzt dann das Piktogramm für den jeweiligen Film auf die angegebene Bildnummer:

```
-- Lingo syntax
on resetThumbnail(whichFlashMovie, whichFrame)
    member(whichFlashMovie).posterFrame = whichFrame
end

// JavaScript syntax
function resetThumbnail(whichFlashMovie, whichFrame) {
    member(whichFlashMovie).posterFrame = whichFrame;
}
```

preferred3dRenderer

Syntax

```
-- Lingo syntax
_movie.preferred3dRenderer

// JavaScript syntax
_movie.preferred3dRenderer;
```

Beschreibung

Mit dieser Filmeigenschaft können Sie den Standard-Renderer abrufen oder festlegen, mit dem 3D-Sprites in einem bestimmten Film gezeichnet werden sollen, sofern dieser Renderer auf dem Clientcomputer verfügbar ist. Lesen/Schreiben.

Wenn der angegebene Renderer auf dem Client-Computer nicht verfügbar ist, wählt der Film den geeignetsten verfügbaren Renderer.

Mögliche Werte für diese Eigenschaft:

- `#openGL` gibt die `openGL`-Hardwarebeschleunigungstreiber an, die sowohl auf Mac- als auch auf Windows-Plattformen verwendet werden können.
- `#directX9` gibt die DirectX® 9-Hardwarebeschleunigungstreiber an, die nur auf Windows-Plattformen verwendet werden können. `#auto` legt unter Windows den Renderer auf „DirectX 9“ fest. Auf Macintosh®-Intel® steht nur der Renderer `#OpenGL` zur Verfügung. DirectX 9 wurde im Eigenschafteninspektor auf der Registerkarte „Film“ dem Menü „Bevorzugter 3D-Renderer“ hinzugefügt. Bei der Verwendung von DirectX 9 liegen keine Änderungen an vorhandenen Funktionalitäten vor. Es kann aber zu einer Leistungsverbesserung kommen.
- `#directX7_0` gibt die DirectX 7-Hardwarebeschleunigungstreiber an, die nur auf Windows-Plattformen verwendet werden können.
- `#directX5_2` gibt die DirectX 5.2-Hardwarebeschleunigungstreiber an, die nur auf Windows-Plattformen verwendet werden können.
- `#software` gibt den in Director integrierten Software-Renderer an, der sowohl auf Mac- als auch auf Windows-Plattformen verwendet werden kann.
- `#auto` bestimmt, dass der jeweils am besten geeignete Renderer ausgewählt werden soll. Dies ist die Standardeinstellung für diese Eigenschaft.

Der für diese Eigenschaft festgelegte Wert wird als Standardeinstellung für die Eigenschaft `renderer` des Renderer-Dienstobjekts verwendet.

Der Unterschied zwischen dieser Eigenschaft und der Eigenschaft `renderer` des `getRendererServices()`-Objekts besteht darin, dass `preferred3dRenderer` den bevorzugten Renderer angibt, die Eigenschaft `renderer` des `getRendererServices()`-Objekts aber den tatsächlich vom Film verwendeten Renderer.

Shockwave Player-Benutzer können den gewünschten Renderer vom Programm aus im Kontextmenü „3D-Renderer“ auswählen. Bei Auswahl der Option Inhaltseinstellungen berücksichtigen wird zum Zeichnen des Films der durch die Eigenschaft `renderer` bzw. `preferred3DRenderer` angegebene Renderer verwendet (falls auf dem Benutzersystem vorhanden), andernfalls der vom Benutzer gewählte Renderer.

Beispiel

Die folgende Anweisung ermöglicht es dem Film, den besten auf dem Benutzersystem verfügbaren 3D-Renderer auszuwählen:

```
-- Lingo syntax
_movie.preferred3dRenderer = #auto

// JavaScript syntax
_movie.preferred3dRenderer = "auto";
```

Siehe auch

getRendererServices(), Movie, renderer

preLoad (3D)

Syntax

```
member(whichCastmember).preload
memberReference.preload
```

Beschreibung

Mit dieser 3D-Eigenschaft können Sie ermitteln oder festlegen, ob Daten schon vor dem Abspielen geladen (TRUE) oder erst während des Abspielvorgangs gestreamt werden (FALSE). Diese Eigenschaft kann nur für verknüpfte Dateien verwendet werden. Der Standardwert ist FALSE.

Wenn Sie in Director die Eigenschaft preLoad auf TRUE setzen, wird der Darsteller vor Wiedergabebeginn vollständig geladen. Wenn Sie im Shockwave Player die Eigenschaft preLoad auf TRUE setzen, fängt der Darsteller zu streamen an, wenn die Wiedergabe des Films beginnt. Vor Durchführung einer Lingo-Operation bei einem streamenden 3D-Darsteller müssen Sie prüfen, ob die Eigenschaft state des Darstellers einen Wert größer oder gleich 2 aufweist.

Beispiel

Die folgende Anweisung setzt die Eigenschaft preload des Darstellers PartyScene auf FALSE, damit während der Wiedergabe extern verknüpfte Medien in PartyScene gestreamt werden können:

```
member("PartyScene").preload = FALSE
member("3D world").preload
```

Siehe auch

state (3D)

preLoad (Darsteller)

Syntax

```
-- Lingo syntax
memberObjRef.preLoad

// JavaScript syntax
memberObjRef.preLoad;
```

Beschreibung

Diese Darstellereigenschaft bestimmt, ob der in *whichCastMember* angegebene Digitalvideo-Darsteller in den Speicher vorausgeladen wird (TRUE) oder nicht (FALSE, Standard). Der Status TRUE hat die gleiche Wirkung, als wenn Sie im Dialogfeld Digitalvideo-Darstellereigenschaften die Option Vorausladen aktivieren wählen.

Bei Flash-Filmdarstellern wird durch diese Eigenschaft bestimmt, ob der Flash-Film erst vollkommen in den Arbeitsspeicher geladen werden muss, bevor das erste Bild eines Sprites angezeigt werden kann (TRUE) oder ob der Film einfach während bei der Wiedergabe in den Speicher gestreamt werden kann (FALSE, Standard). Diese Eigenschaft kann nur für verknüpfte Flash-Filme verwendet werden, deren Elemente in einer externen Datei gespeichert sind. Die Eigenschaft hat keine Wirkung auf Darsteller, deren Elemente in der Besetzung gespeichert sind. Die Eigenschaften `streamMode` und `bufferSize` bestimmen, wie der Darsteller in den Speicher gestreamt wird.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Die folgende Anweisung zeigt im Nachrichtenfenster an, ob der QuickTime-Film „Drehstuhl“ im Voraus in den Speicher geladen werden kann:

```
-- Lingo syntax
put(member("Rotating Chair").preload)

// JavaScript syntax
put(member("Rotating Chair").preload);
```

Siehe auch

bufferSize, streamMode

preLoadEventAbort

Syntax

```
-- Lingo syntax
_movie.preLoadEventAbort

// JavaScript syntax
_movie.preLoadEventAbort;
```

Beschreibung

Diese Filmeigenschaft gibt an, ob das Vorausladen der Darsteller durch Tastendruck oder Mausklick gestoppt werden kann (TRUE) oder nicht (FALSE, Standard). Lesen/Schreiben.

Diese Eigenschaft wirkt sich lediglich auf den aktuellen Film aus.

Beispiel

Die folgende Anweisung bewirkt, dass der Benutzer das Vorausladen von Darstellern stoppen kann, indem er eine Taste drückt oder mit der Maus klickt:

```
-- Lingo syntax
_movie.preLoadEventAbort = TRUE

// JavaScript syntax
_movie.preLoadEventAbort = true;
```

Siehe auch

Movie

preLoadMode

Syntax

```
-- Lingo syntax
castObjRef.preLoadMode

// JavaScript syntax
castObjRef.preLoadMode;
```

Beschreibung

Diese Besetzungsbibliothekseigenschaft bestimmt den Vorausladem odus einer angegebenen Besetzungsbibliothek. Lesen/Schreiben.

`preLoadMode` kann folgende gültige Werte annehmen:

- 0. Besetzungsbibliothek bei Bedarf laden. (Standardeinstellung).
- 1. Besetzungsbibliothek vor Bild 1 laden.
- 2. Besetzungsbibliothek nach Bild 1 laden.

Die Einstellung dieser Eigenschaft hat die gleiche Wirkung wie das Festlegen von „Besetzung laden“ im Dialogfeld „Darstellereigenschaften“.

Beispiel

Die folgende Anweisung veranlasst Director, die Darsteller der Besetzung „Schaltflächen“ zu laden, bevor der Film in Bild 1 eintritt:

```
-- Lingo syntax
castLib("Buttons").preLoadMode = 1

// JavaScript syntax
castLib("Buttons").preLoadMode = 1;
```

Siehe auch

Cast Library

preLoadRAM

Syntax

```
the preLoadRAM
```

Beschreibung

Diese Systemeigenschaft gibt an, wie viel Arbeitsspeicher (RAM) zum Vorausladen eines Digitalvideos verwendet werden kann. Diese Eigenschaft kann getestet und eingestellt werden.

Diese Eigenschaft kann bei der Speicherverwaltung sehr nützlich sein, da dadurch Digitalvideo-Darsteller auf eine gewisse Speichermenge beschränkt werden, sodass noch Platz zum Vorausladen von anderen Darstellertypen bleibt. Wenn `preLoadRAM` den Wert `FALSE` besitzt, kann der gesamte verfügbare Speicherplatz zum Vorausladen von Digitalvideo-Darstellern verwendet werden.

Es ist jedoch nicht möglich, mit Sicherheit vorauszusagen, wie viel RAM ein vorausgeladenes Digitalvideo tatsächlich in Anspruch nehmen wird, da dies vom Inhalt des Films, vom Ausmaß der Komprimierung, von der Anzahl der Schlüsselbilder und von Grafikänderungen usw. abhängt.

In der Regel brauchen nur die ersten paar Sekunden des Videos vorausgeladen zu werden. Danach kann das Video gestreamt werden.

Falls Sie die Datendurchsatzrate des Films kennen, können Sie die Einstellung für `preLoadRAM` ziemlich genau abschätzen. Wenn der Film beispielsweise eine Datendurchsatzrate von 300 KB pro Sekunde hat, sollten Sie `preloadRAM` auf 600 KB einstellen, sofern Sie die ersten zwei Sekunden der Videodatei vorausgeladen wollen. Dies ist natürlich nur eine Schätzung, aber in den meisten Situationen liegen Sie damit ziemlich richtig.

Beispiel

Die folgende Anweisung stellt `preLoadRAM` auf 600 KB ein, damit die ersten zwei Sekunden eines Films mit einer Datendurchsatzrate von 300 KB pro Sekunde vorausgeladen werden.

```
--Lingo
set the preLoadRAM = 600

// Javascript
_system.preLoadRAM = 600;
```

Siehe auch

`loop (Schlüsselwort)`, `next`

preLoadTime

Syntax

```
-- Lingo syntax
memberObjRef.preLoadTime

// JavaScript syntax
memberObjRef.preLoadTime;
```

Beschreibung

Diese Darsteller- und Soundkanaleigenschaft gibt an, wie viele Sekunden des Shockwave Audio (SWA)-Streaming-Darstellers vor Beginn der Wiedergabe heruntergeladen werden sollen oder wann der Befehl `preLoadBuffer` verwendet werden soll. Der Standardwert ist 5 Sekunden.

Diese Eigenschaft kann nur dann eingestellt werden, wenn der SWA-Streaming-Darsteller gestoppt ist.

Bei Soundkanälen bezieht sich der Wert auf den angegebenen Sound in der Warteschlange bzw. auf den gegenwärtig abgespielten Sound, wenn nichts angegeben ist.

Beispiel

Die folgende Prozedur stellt die Zeitspanne zum Vorausladen des SWA-Streaming-Darstellers „Louis Armstrong“ auf 6 Sekunden ein. Das eigentliche Vorausladen findet erst statt, wenn der Befehl `preLoadBuffer` oder play erteilt wird.

```
-- Lingo syntax
on mouseDown
    member("Louis Armstrong").stop()
    member("Louis Armstrong").preLoadTime = 6
end

// JavaScript syntax
function mouseDown() {
    member("Louis Armstrong").stop();
    member("Louis Armstrong").preLoadTime = 6;
}
```

Die folgende Anweisung gibt den `preLoadTime`-Wert des gegenwärtig abgespielten Sounds in Soundkanal 1 zurück:

```
-- Lingo syntax
put sound(1).preLoadTime

// JavaScript syntax
trace(sound(1).preLoadTime);
```

Siehe auch

preLoadBuffer()

primitives

Syntax

```
getRendererServices().primitives
```

Beschreibung

Diese 3D-Funktion gibt eine Liste der Primitiventypen zurück, die zur Erstellung neuer Modellressourcen verwendet werden können.

Beispiel

Die folgende Anweisung zeigt die verfügbaren Primitiventypen an:

```
--Lingo
put getRendererServices().primitives

// Javascript
put( getRendererServices().primitives);
```

Siehe auch

getRendererServices(), newModelResource

productName

Syntax

```
-- Lingo syntax
_player.productName

// JavaScript syntax
_player.productName;
```

Beschreibung

Diese Player-Eigenschaft gibt den Namen der Director-Anwendung zurück. Nur Lesen.

Beispiel

Die folgende Anweisung zeigt den Namen der Director-Anwendung im Nachrichtenfenster an.

```
-- Lingo syntax
trace(_player.productName)

// JavaScript syntax
trace(_player.productName);
```

Siehe auch

Player

productVersion

Syntax

```
-- Lingo syntax
_player.productVersion

// JavaScript syntax
_player.productVersion;
```

Beschreibung

Diese Player-Eigenschaft gibt die Versionsnummer der Director-Anwendung zurück. Nur Lesen.

Beispiel

Die folgende Anweisung zeigt die Version der Director-Anwendung im Nachrichtenfenster an.

```
-- Lingo syntax
trace(_player.productVersion)

// JavaScript syntax
trace(_player.productVersion);
```

Siehe auch

Player

projection

Syntax

```
sprite(whichSprite).camera.projection
camera(whichCamera).projection
member(whichCastmember).camera(whichCamera).projection
```

Beschreibung

Mit dieser 3D-Eigenschaft können Sie die Projektionsart der Kamera ermitteln und festlegen. Mögliche Werte: `#perspective` (Standard) und `#orthographic`.

Bei der Projektionsart `#perspective` erscheinen Objekte, die sich näher an der Kamera befinden, größer als Objekte, die weiter von der Kamera entfernt sind. Die Eigenschaften `projectionAngle` und `fieldOfView` geben den vertikalen Projektionswinkel an, der bestimmt, wie viel von der Welt Sie sehen. Der horizontale Projektionswinkel wird durch das Seitenverhältnis der Kameraeigenschaft `rect` bestimmt.

Bei der Projektionsart `#orthographic` hängt die scheinbare Größe von Objekten nicht von der Entfernung zur Kamera ab. Die Eigenschaft `orthoHeight` gibt an, wie viele Welteinheiten vertikal in das Sprite passen; danach richtet sich, wie viel von der Welt Sie sehen. Die orthografische Projektionsbreite wird durch das Seitenverhältnis der Kameraeigenschaft `rect` bestimmt.

Beispiel

Die folgende Anweisung setzt die Eigenschaft @@@projection der Kamera von Sprite 5 auf `#orthographic`:

```
sprite(5).camera.projection = #orthographic
```

Siehe auch

`fieldOfView (3D)`, `orthoHeight`, `fieldOfView (3D)`

purgePriority, Eigenschaft

Syntax

```
-- Lingo syntax
memberObjRef.purgePriority

// JavaScript syntax
memberObjRef.purgePriority;
```

Beschreibung

Diese Darstellereigenschaft gibt die Löschpriorität eines Darstellers an. Lesen/Schreiben.

Die Löschprioritäten eines Darstellers bestimmen die Priorität, mit der Director entscheidet, welche Darsteller bei vollem Speicher gelöscht werden sollen. Je höher die Löschpriorität, desto wahrscheinlicher ist es, dass der Darsteller gelöscht wird. Folgende `purgePriority`-Einstellungen stehen zur Verfügung:

- 0 – Nie
- 1 – Zuletzt
- 2 – Als nächstes
- 3 – Normal (Standard)

Bei der Einstellung „Normal“ löscht Director die Darsteller nach dem Zufallsprinzip aus dem Speicher. Bei den Einstellungen „Als nächstes“, „Zuletzt“ und „Nie“ haben Sie eine gewisse Kontrolle über den Löschvorgang, aber bei „Zuletzt“ oder „Nie“ könnte dem Film leicht die Speicherkapazität ausgehen, wenn mehrere Darsteller auf diese Werte eingestellt sind.

Die Darstellereinstellung `purgePriority`eignet sich besonders dann für die Speicherverwaltung, wenn die verfügbare Speicherkapazität für die Besetzungsbibliothek des Films nicht ausreicht. In der Regel können Sie Unterbrechungen beim Laden von Darstellern und auch das Nachladen von Darstellern so gering wie möglich halten, indem Sie den Darstellern, die im Verlauf des Films häufig gebraucht werden, eine niedrige Löschpriorität zuordnen.

Beispiel

Die folgende Anweisung setzt die Löschpriorität des Darstellers „Hintergrund“ auf 3. Dieser Darsteller ist somit einer der ersten, die bei mangelndem Speicherplatz gelöscht werden.

```
-- Lingo syntax
member("Background").purgePriority = 3

// JavaScript syntax
member("Background").purgePriority = 3;
```

Siehe auch

`Member`

quad

Syntax

```
-- Lingo syntax
spriteObjRef.quad

// JavaScript syntax
spriteObjRef.quad;
```

Beschreibung

Diese Sprite-Eigenschaft enthält eine Liste von vier Punkten, bei denen es sich um Fließkommawerte handelt, durch die die Eckpunkte eines Sprites auf der Bühne beschrieben werden. Lesen/Schreiben.

Die Punkte des Quaders stehen in folgender Reihenfolge: oben links, oben rechts, untern rechts, unten links.

Die Punkte selbst können manipuliert werden, um Perspektiven- und andere Grafikverzerrungen zu erzeugen.

Nachdem Sie das Quad eines Sprites manipuliert haben, können Sie es auf die Drehbuchwerte zurücksetzen, indem Sie das mit Skript erstellte Sprite mit `puppetSprite(`*intSpriteNum*`,FALSE)` deaktivieren. Wenn das Quad eines Sprites deaktiviert ist, lässt sich das Sprite nicht drehen oder neigen.

Beispiel

Die folgende Anweisung zeigt eine typische Liste zur Beschreibung eines Sprites an:

```
-- Lingo syntax
put(sprite(1).quad)

// JavaScript syntax
put(sprite(1).quad);
```

Wenn Sie die Sprite-Eigenschaft quad ändern, müssen Sie die Punktliste nach dem Ändern von Werten auch entsprechend zurücksetzen. Wenn Sie eine Variable auf den Wert einer Eigenschaft setzen, stellen Sie nämlich nur eine Kopie der Liste (und nicht die Liste selbst) in die Variable. Um eine Änderung durchzuführen, verwenden Sie folgende Syntax (nur für Lingo gültig):

```
-- Lingo syntax
currQuadList = sprite(5).quad
currQuadList[1] = currQuadList[1] + point(50, 50)
sprite(5).quad = currQuadList
```

Siehe auch

`point()`, `puppetSprite()`, `Sprite`

quality

Syntax

```
-- Lingo syntax
memberOrSpriteObjRef.quality

// JavaScript syntax
memberOrSpriteObjRef.quality;
```

Beschreibung

Diese Flash-Darsteller- und Sprite-Eigenschaft bestimmt, ob ein Flash-Film-Sprite mithilfe von Anti-Aliasing wiedergegeben wird, um so eine bessere Wiedergabequalität zu erreichen. Dadurch kann allerdings die Wiedergabe des Films verlangsamt werden. Für die Eigenschaft `quality` stehen folgende Werte zur Verfügung:

- `#autoHigh` – Director verwendet bei der Sprite-Darstellung anfänglich Anti-Aliasing. Falls die tatsächliche Bildrate dann aber unter die für den Film angegebene Bildrate absinkt, schaltet Director das Anti-Aliasing ab. Bei dieser Einstellung hat die Abspielgeschwindigkeit Priorität vor der Bildqualität.
- `#autoLow` – Die Filmwiedergabe erfolgt anfänglich ohne Anti-Aliasing. Wenn der Flash-Player dann aber feststellt, dass der Prozessor des Computers leistungsfähig genug ist, wird das Anti-Aliasing eingeschaltet. Bei dieser Einstellung wird der Bildqualität nach Möglichkeit Priorität eingeräumt.
- `#high` (Standard) – Bei dieser Einstellung wird der Film stets mit Anti-Aliasing abgespielt.
- `#low` – Bei dieser Einstellung wird der Film immer ohne Anti-Aliasing abgespielt.

Die Eigenschaft `quality` kann getestet und eingestellt werden.

Beispiel

Das folgende Sprite-Skript prüft die Farbtiefe des Computers, auf dem der Film abgespielt wird. Falls die Farbtiefe auf 8 Bit (256 Farben) oder darunter eingestellt ist, setzt das Skript die Qualität des Sprites in Kanal 5 auf `#low`.

```
-- Lingo syntax
on beginSprite me
    if _system.colorDepth <= 8 then
        sprite(1).quality = #low
    end if
end

// JavaScript syntax
function beginSprite() {
    var clrDp = _system.colorDepth;
    if (clrDp <= 8) {
        sprite(1).quality = symbol("low");
    }
}
```

quality (3D)

Syntax

```
member(whichCastmember).texture(whichTexture).quality
member(whichCastmember).shader(whichShader).texture(whichTexture).quality
member(whichCastmember).model(whichModel).shader.texture(whichTexture).quality
member( whichCastmember ).model( whichModel ).shader.texturelist[TextureListIndex].quality
member(whichCastmember).model(whichModel).shaderList[shaderListIndex].
texture(whichTexture).quality
member( whichCastmember ).model( whichModel ).shaderList[ shaderListIndex ]. texturelist[
TextureListIndex ].quality
```

Beschreibung

Mit dieser 3D-Textureigenschaft können Sie die Bildqualität einer Textur ermitteln und festlegen, indem Sie steuern, wie viel Mipmapping auf die Textur angewendet wird. Mipmapping ist ein Prozess, bei dem zusätzliche Versionen des Texturbildes in mehreren Größen erstellt werden, die kleiner als das Original sind. Das 3D-Xtra verwendet dann die Version des Bildes, die für die aktuelle Größe des Modells auf dem Bildschirm am besten geeignet ist, und ändert bei Bedarf die Version des benutzten Bildes. Trilineares Mipmapping führt zu höherer Qualität und belegt mehr Speicher als bilineares Mipmapping.

Mipmapping ist nicht das Gleiche wie Filtern, obwohl beide Prozesse das Aussehen der Textur verbessern. Beim Filtern werden Fehler über den gesamten Texturbereich hinweg verteilt, damit sie weniger konzentriert sind. Beim Mipmapping wird ein Neusampling des Bildes durchgeführt, um es entsprechend zu verkleinern.

Diese Eigenschaft kann folgende Werte aufweisen:

- `#low` ist das Gleiche wie "aus"; bei beiden Einstellungen wird für die Textur kein Mipmapping verwendet.
- Bei `#medium` wird für die Textur ein bilineares Mipmapping (niedrige Qualität) aktiviert.
- Bei `#high` wird für die Textur ein trilineares Mipmapping (hohe Qualität) aktiviert.

Der Standardwert ist `#low`.

Beispiel

Die folgende Anweisung setzt die Eigenschaft quality der Textur "Marskarte" auf `#medium`:

```
member("scene").texture("Marsmap").quality = #medium
```

Siehe auch

nearFiltering

radius

Syntax

```
modelResourceObjectReference.radius
member(whichCastmember).modelResource(whichModelResource).radius
```

Beschreibung

Mit dieser 3D-Modelleigenschaft können Sie bei Modellressourcen vom Typ `#sphere` oder `#cylinder` den Radius des Modells ermitteln und festlegen.

Die Eigenschaft `radius` bestimmt den Ablenkradius zur Erzeugung der Modellressource. Der Wert dieser Eigenschaft muss größer als 0.0 sein. Der Standardwert ist 25.0.

Beispiel

Die folgende Anweisung zeigt den Radius der Modellressource „Cylinder01“ an.

```
--Lingo
put member("3Dobjects").modelResource("Cylinder01").radius

// Javascript
put(member("3Dobjects").getPropRef("modelResource",11).radius);
```

randomSeed

Syntax

```
the randomSeed
```

Beschreibung

Diese Systemeigenschaft gibt den Ausgangswert zum Generieren von Zufallszahlen an, auf die dann über die Funktion `random()` zugegriffen werden kann.

Bei Verwendung des gleichen Ausgangswertes erhalten Sie stets die gleiche Folge von Zufallszahlen. Diese Eigenschaft eignet sich insbesondere für das Debugging in der Entwicklungsumgebung. Mit der Eigenschaft `ticks` können Sie schnell und einfach einen eindeutigen Ausgangswert für Zufallszahlen generieren, da es sehr unwahrscheinlich ist, dass der gleiche `ticks`-Wert bei nachfolgender Verwendung der Eigenschaft erneut auftritt.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Die folgende Anweisung zeigt im Nachrichtenfenster den Ausgangswert für den Zufallsgenerator an.

```
--Lingo
put the randomSeed

// Javascript
put(_system.randomSeed);
```

Siehe auch

`random()`, `milliseconds`

recordFont

Syntax

```
recordFont(whichCastMember, font {[,face]} {,[bitmapSizes]} {,characterSubset} {,
userFontName})
```

Beschreibung

Dieser Befehl bettet eine `TrueType`- oder Type 1-Schrift als Darsteller ein. Nach Einbettung stehen diese Schriften dem Autor genauso wie alle anderen im System installierten Schriften zur Verfügung.

Bevor Sie den Befehl `recordFont` verwenden können, müssen Sie mit dem Befehl `new()` einen leeren Schriftdarsteller erstellen.

- *font* – Name der aufzuzeichnenden Originalschrift.
- *face* – Liste mit Schriftbildsymbolen für die Originalschrift. Mögliche Werte: `#plain`, `#bold`, `#italic`. Wenn Sie keinen Wert für dieses Argument angeben, wird `#plain`verwendet.
- *bitmapSizes* – Liste der Ganzzahlen, durch die die Größen für die aufzuzeichnenden Bitmaps angegeben werden. Dieses Argument kann weggelassen werden. Wenn Sie dieses Argument weglassen, werden keine Bitmaps generiert. Bitmaps ergeben in der Regel bei kleineren Punktgrößen (d. h. unter 14 Punkt) ein besseres Schriftbild, belegen aber auch mehr Speicherplatz.
- *characterSubset* – Zu kodierender Zeichenstring. Nur die angegebenen Zeichen sind in der Schrift verfügbar. Wenn Sie dieses Argument weglassen, werden alle Zeichen kodiert. Falls nur bestimmte Zeichen kodiert sind, aber ein nicht kodiertes Zeichen verwendet wird, erscheint dieses Zeichen als leeres Feld.
- *userFontName* – Ein String, der als Name für den neu aufgezeichneten Schriftdarsteller verwendet werden soll.

Dieser Befehl erstellt einen Shock Font in *whichCastMember*, und zwar unter Verwendung der im Argument *font* angegebenen Schrift. Aus dem durch diesen Befehl zurückgegebenen Wert ist ersichtlich, ob der Vorgang erfolgreich war oder nicht. Der Wert 0 bedeutet einen erfolgreichen Abschluss.

Beispiel

Die folgende Anweisung erstellt einen einfachen Shock Font unter Verwendung der zwei Argumente für den Darsteller und die aufzuzeichnende Schrift:

```
myNewFontMember = new(#font)
recordFont(myNewFontMember, "Lunar Lander")
```

Die folgende Anweisung gibt die zu erstellenden Bitmapgrößen und die Zeichen an, für die Schriftdaten erstellt werden sollen:

```
myNewFontMember = new(#font)
recordfont(mynewmember,"lunar lander",[],[14, 18, 45], "Lunar Lander Game High Score First
Last Name")
```

Hinweis: *Da die Schriftdaten durch "recordFont" nur neu synthetisiert, aber nicht direkt verwendet werden, kann der Shock Font ohne rechtliche Beschränkungen verteilt werden.*

Siehe auch

`new()`

rect (camera)

Syntax

```
sprite(whichSprite).camera(whichCamera).rect
```

Beschreibung

Mit dieser 3D-Kameraeigenschaft können Sie das Rechteck ermitteln und festlegen, das die Größe und Position der Kamera steuert. Dieses Rechteck entspricht dem Rechteck, das durch den Sucher einer echten Kamera zu sehen ist.

Der Standardwert dieser Eigenschaft ist bei allen Kameras „rect(0,0,1,1)“, d. h. die Kamera ist unsichtbar, bis Sie die Einstellung ändern. Beim Rendern von `sprite.camera(1)` wird das Rechteck jedoch auf `rect(0, 0, sprite(`*whichSprite*`).width, sprite(`*whichSprite*`).height)` gesetzt, damit die Kamera den Bildschirm füllt. Alle Koordinaten des Kamerarechtecks werden ausgehend von der linken oberen Ecke des Sprites angegeben.

Wenn *whichCamera* größer als 1 ist, wird `rect` beim Skalieren des Sprites nicht skaliert. Falls gewünscht, muss dies per Skriptcode nachgeholt werden.

Wenn *whichCamera* größer ist als 1, müssen die Eigenschaften `rect.top` und `rect.left` größer oder gleich den Einstellungen `rect.top` und `rect.left` von `sprite.camera(1)` sein.

Beispiel

Die folgende Anweisung zeigt die Koordinaten des Bitmapdarstellers 1 an.

```
-- Lingo syntax
put(member(1).rect)

// JavaScript syntax
put(member(1).rect);
```

Siehe auch

`cameraPosition`, `cameraRotation`

rect (Grafik)

Syntax

```
-- Lingo syntax
imageObjRef.rect

// JavaScript syntax
imageObjRef.rect;
```

Beschreibung

Diese Grafikeigenschaft gibt ein Rechteck zurück, das die Größe einer vorgegebenen Grafik beschreibt. Nur Lesen.

Die zurückgegebenen Rechteckkoordinaten werden relativ zur linken oberen Ecke der Grafik angegeben. Der linke und obere Wert des Rechtecks ist somit immer 0; der rechte und untere Wert entspricht der Breite bzw. Höhe des Darstellers.

Beispiel

Die folgende Anweisung gibt das Rechteck des 300 x 400 Pixel großen Darstellers „Sonnenaufgang“ im Nachrichtenfenster zurück:

```
-- Lingo syntax
member("Sunrise").image.rect -- rect(0, 0, 300, 400)

// JavaScript syntax
member("Sunrise").image.rect; // rect(0, 0, 300, 400)
```

Der folgende Lingo-Code untersucht die ersten 50 Darsteller und zeigt das Rechteck und den Namen aller Darsteller an, die Bitmaps sind:

```
-- Lingo syntax
on showAllRects
    repeat with x = 1 to 50
    if member(x).type = #bitmap then
        put member(x).image.rect && "-" && member(x).name
    end if
    end repeat
end

// JavaScript syntax
function showAllRects() {
    var x = 1;
    while(x < 51) {
        var tp = member(x).type;
        if (tp == "bitmap") {
            trace(member(x).image.rect + " - " + member(x).name);
            i++;
        }
    }
}
```

Siehe auch

height, image(), width

rect (Darsteller)

Syntax

```
-- Lingo syntax
memberObjRef.rect

// JavaScript syntax
memberObjRef.rect;
```

Beschreibung

Diese Darstellereigenschaft gibt die linke, obere, rechte und untere Koordinate für das Rechteck eines Grafikdarstellers (wie z. B. Bitmap, Form, Film oder Digitalvideo) zurück. Die Koordinaten werden als Rechteck zurückgegeben. Nur Lesen für alle Darsteller. Lesen/Schreiben nur für Felddarsteller.

Bei einer Bitmap wird die Eigenschaft `rect` von der linken oberen Ecke der Bitmap gemessen und nicht wie im Malfenster von der linken oberen Ecke der Leinwand.

Bei einem Xtra-Erweiterungsdarsteller besteht die Eigenschaft `rect` aus einem Rechteck, dessen linke obere Ecke bei(0,0) liegt.

Beispiel

Die folgende Anweisung zeigt die Koordinaten des Bitmapdarstellers 20 an:

```
-- Lingo syntax
put(member(20).rect)

// JavaScript syntax
put(member(20).rect);
```

Die folgende Anweisung stellt die Koordinaten für den Bitmapdarsteller „Banner“ ein:

```
-- Lingo syntax
member("Banner").rect = rect(100, 150, 300, 400)

// JavaScript syntax
member("Banner").rect = rect(100, 150, 300, 400);
```

Siehe auch

Member

rect (Sprite)

Syntax

```
-- Lingo syntax
spriteObjRef.rect

// JavaScript syntax
spriteObjRef.rect;
```

Beschreibung

Diese Sprite-Eigenschaft gibt die linke, obere, rechte und untere Koordinate für das Rechteck eines Grafik-Sprites (wie z. B. Bitmap, Form, Film oder Digitalvideo) an. Die Koordinaten werden als Rechteck zurückgegeben. Lesen/Schreiben.

Beispiel

Die folgende Anweisung zeigt die Koordinaten des Bitmap-Sprites 20 an:

```
-- Lingo syntax
put(sprite(20).rect)

// JavaScript syntax
put(sprite(20).rect);
```

Siehe auch

rect(), Sprite

rect (Fenster)

Syntax

```
-- Lingo syntax
windowObjRef.rect

// JavaScript syntax
windowObjRef.rect;
```

Beschreibung

Diese Fenstereigenschaft gibt die linke, obere, rechte und untere Koordinate für ein Fenster an. Die Koordinaten werden als Rechteck zurückgegeben. Lesen/Schreiben.

Wenn das angegebene Rechteck kleiner als die Bühne ist, auf welcher der Film erstellt wurde, wird der Film auf das Fenster zugeschnitten, d. h. nicht entsprechend verkleinert.

Um den im Fenster ablaufenden Film zu schwenken oder zu skalieren, müssen Sie die Fenstereigenschaft `drawRect` oder `sourceRect` einstellen.

Beispiel

Die folgende Anweisung zeigt die Koordinaten des Fensters `Control_panel` an:

```
-- Lingo syntax
put(window("Control_panel").rect)

// JavaScript syntax
put(window("Control_panel").rect);
```

Siehe auch

drawRect, sourceRect, Window

ref

Syntax

```
chunkExpression.ref
```

Beschreibung

Mit dieser Eigenschaft für Text-Chunk-Ausdrücke können Sie in einem Textdarsteller mühelos auf einen Chunk-Ausdruck Bezug nehmen.

Beispiel
Ohne Bezüge wären beispielsweise folgende Anweisungen erforderlich:

```
member(whichTextMember).line[whichLine].word[firstWord..lastWord].font = "Palatino"
member(whichTextMember).line[whichLine].word[firstWord..lastWord].fontSize = 36
member(whichTextMember).line[whichLine].word[firstWord..lastWord].fontStyle = [#bold]
```

Mit der Eigenschaft ref können Sie dagegen wie folgt auf den gleichen Chunk Bezug nehmen:

```
myRef = member(whichTextMember).line[whichLine].word[firstWord..lastWord].ref
```

Die Variable `myRef` wird somit zur Kurzformel für den gesamten Chunk-Ausdruck. Dadurch können Sie dann mit folgendem Lingo-Code arbeiten:

```
put myRef.font
-- "Palatino"
```

Sie können auch wie folgt eine Chunk-Eigenschaft einstellen:

```
myRef.fontSize = 18
myRef.fontStyle = [#italic]
```

Außerdem können Sie unter Verwendung der Eigenschaft text des Bezugs wie folgt auf den referenzierten String zugreifen:

```
put myRef.text
```

Dadurch erhalten Sie dann die tatsächlichen Stringdaten und nicht nur Informationen über den String.

reflectionMap

Syntax
```
member(whichCastmember).shader(whichShader).reflectionMap
```

Beschreibung
Mit dieser 3D-Shader-Eigenschaft können Sie die Textur ermitteln und festlegen, die zur Erzeugung von Reflexionen auf der Oberfläche eines Modells verwendet werden soll. Diese Textur wird auf die dritte Texturebene des Shaders angewendet. Diese Eigenschaft wird ignoriert, wenn der Modifizierer `toon` auf die Modellressource angewendet wird.

Diese Hilfseigenschaft bietet eine einfachere Schnittstelle zur Vereinheitlichung des Reflection Mapping. Die gleiche Wirkung lässt sich mit den folgenden Eigenschaften erzielen:

```
shader.textureModeList[3] = #reflection
shader.blendFunctionList[3] = #blend
shader.blendSourceList[3] = #constant
shader.blendConstantList[3] = 50.0
```

Wenn diese Eigenschaft getestet wird, gibt sie die Textur der dritten Texturebene des Modells zurück. Die Standardeinstellung ist `void`.

Beispiel
Die folgende Anweisung bewirkt, dass beim Modell „Sphere01“ die Standardtextur der Modelloberfläche reflektiert wird.

```
-- Lingo syntax
member("3Dobjects").model("Sphere01").shader.reflectionMap=member("3Dobjects").texture[1]

// JavaScript syntax
member("3Dobjects").getPropRef("model",2).shader.reflectionMap=member("3Dobjects").getPropRe
f("texture",1);
```

Siehe auch

textureModeList, blendFunctionList, blendConstantList

reflectivity

Syntax

```
member(whichCastmember).reflectivity
```

Beschreibung

Mit dieser 3D-Shader-Eigenschaft können Sie ermitteln und festlegen, wie stark der Standard-Shader des referenzierten Darstellers glänzt. Der Wert ist eine Fließkommazahl, die den Prozentsatz des Lichts angibt, das von der Oberfläche eines Modells, das den Standard-Shader verwendet, reflektiert werden soll (von 100,00 bis 0,0).

Beispiel

Die folgende Anweisung setzt den Reflexionsgrad des Standard-Shaders im Darsteller „Szene“ auf 50%.

```
-- Lingo syntax
member("Scene").reflectivity = 50

// JavaScript syntax
member("Scene").reflectivity = 50;
```

region

Syntax

```
member(whichCastmember).modelResource(whichModelResource).emitter.region
modelResourceObjectReference.emitter.region
```

Beschreibung

Mit dieser 3D-Emittereigenschaft können Sie bei Modellressourcen vom Typ `#particle` die Eigenschaft `region` des Partikelemitters der Ressource ermitteln und festlegen.

Die Eigenschaft „region“ definiert die Stelle, von der Partikel emittiert werden. Wenn ihr Wert ein einzelner Vektor ist, wird anhand dieses Vektors ein Punkt in der 3D-Welt definiert, von dem Partikel emittiert werden.

Wenn ihr Wert eine Liste aus zwei Vektoren ist, definieren diese die Endpunkte eines Liniensegments, von dem Partikel emittiert werden.

Wenn ihr Wert eine Liste aus vier Vektoren ist, definieren diese die Scheitelpunkte eines Vierecks, von dem Partikel emittiert werden.

Der Standardwert dieser Eigenschaft lautet [vector(0,0,0)].

Beispiel

In diesem Beispiel ist `ThermoSystem` eine Modellressource vom Typ `#particle`. Die folgende Anweisung gibt die vier Ecken eines Rechtecks an, von dem die Partikel von `ThermoSystem` ausgehen:

```
member("Fires").modelResource("ThermoSystem").emitter.region = [vector(20,90,100),
vector(30,90,100), vector(30,100,100), vector(20,100,100)]
```

Siehe auch

`emitter`

regPoint

Syntax

```
-- Lingo syntax
memberObjRef.regPoint

// JavaScript syntax
memberObjRef.regPoint;
```

Beschreibung

Diese Darstellereigenschaft gibt das Registrierungskreuz eines Darstellers an. Lesen/Schreiben.

Das Registrierungskreuz wird als horizontale und vertikale Koordinaten eines Punkts im Format `point(`*horizontal, vertical*`)`. Unsichtbare Darsteller (wie z. B. Sounds) haben keine sinnvolle `regPoint` -Eigenschaft.

Sie können die Eigenschaft `regPoint` zum Animieren einzelner Grafiken in einer Filmschleife verwenden, indem Sie die Filmschleifenposition im Verhältnis zu anderen Objekten auf der Bühne ändern.

Außerdem können Sie über `regPoint` die Position einer im Sprite verwendeten Maske entsprechend anpassen.

Wenn ein Flash-Filmdarsteller in die Besetzungsbibliothek eingefügt wird, ist das Registrierungskreuz anfänglich auf den Mittelpunkt des Darstellers und die Eigenschaft `centerRegPoint` auf `TRUE` eingestellt. Wenn Sie anschließend das Registrierungskreuz über die Eigenschaft `regPoint` neu positionieren, wird die Eigenschaft `centerRegPoint` automatisch auf `FALSE` gesetzt.

Beispiel

Die folgende Anweisung zeigt das Registrierungskreuz des Bitmapdarstellers „Schreibtisch“ im Nachrichtenfenster an:

```
-- Lingo syntax
put(member("Desk").regPoint)

// JavaScript syntax
put(member("Desk").regPoint);
```

Die folgende Anweisung ändert das Registrierungskreuz des Bitmapdarstellers „Schreibtisch“ in die Werte in der Liste:

```
-- Lingo syntax
member("Desk").regPoint = point(300, 400)

// JavaScript syntax
member("Desk").regPoint = point(300, 400);
```

Siehe auch

Member, Sprite

regPoint (3D)

Syntax

```
sprite(whichSprite).camera.backdrop[backdropIndex].regPoint
member(whichCastmember).camera(whichCamera).backdrop
[backdropIndex].regPoint
```

Beschreibung

Mit dieser 3D-Hintergrund- und -Überlagerungseigenschaft können Sie das Registrierungskreuz des Hintergrunds bzw. der Überlagerung abrufen und einstellen. Das Registrierungskreuz stellt die *x*-, *y*- und *z*-Koordinaten des Hintergrund- bzw. Überlagerungsmittelpunkts im 3D-Raum dar. Der Standardwert dieser Eigenschaft lautet point(0,0).

Beispiel

Die folgende Anweisung ändert das Registrierungskreuz des ersten Hintergrunds der Kamera von Sprite 13 in „point(50,0)“, gemessen von der oberen linken Ecke des Hintergrunds:

```
sprite(13).camera.backdrop[1].regPoint = point(50, 0)
```

Siehe auch

loc (backdrop and overlay)

regPointVertex

Syntax

```
-- Lingo syntax
memberObjRef.regPointVertex

// JavaScript syntax
memberObjRef.regPointVertex;
```

Beschreibung

Diese Darstellereigenschaft gibt an, ob ein Scheitelpunkt von *vectorCastMember* als Registrierungskreuz für diesen Darsteller verwendet wird. Wenn der Wert 0 lautet, wird das Registrierungskreuz normal anhand der Eigenschaften `centerRegPoint` und `regPoint` bestimmt. Ein Wert ungleich 0 gibt die Position des als Registrierungskreuz zu verwendenden Scheitelpunkts in der `vertextList` an. Die Eigenschaft `centerRegPoint` wird auf `FALSE` und die Eigenschaft `regPoint` auf die Position dieses Scheitelpunkts gesetzt.

Beispiel

Die folgende Anweisung passt das Registrierungskreuz für den Vektorformdarsteller „Squiggle“ an die Position des dritten Scheitelpunkts an:

```
-- Lingo syntax
member("squiggle").regPointVertex=3

// JavaScript syntax
member("squiggle").regPointVertex=3;
```

Siehe auch

centerRegPoint, regPoint

renderer

Syntax

```
getRendererServices().renderer
```

Beschreibung

Mit dieser 3D-Eigenschaft können Sie den aktuellen in einem Film verwendeten Renderer ermitteln und festlegen. Welche Werte für diese Eigenschaft gültig sind, richtet sich nach den verfügbaren Renderern, die von der Eigenschaft `rendererDeviceList` des Renderer-Dienstobjekts zurückgegeben wurden.

Shockwave Player-Benutzer können den gewünschten Renderer vom Programm aus im Kontextmenü „3D-Renderer“ auswählen. Bei Auswahl der Option Inhaltseinstellungen berücksichtigen wird zum Zeichnen des Films der durch die Eigenschaft `renderer` bzw. `preferred3DRenderer` angegebene Renderer verwendet (falls auf dem Benutzersystem vorhanden), andernfalls der vom Benutzer gewählte Renderer.

Der Standardwert dieser Eigenschaft wird durch die Eigenschaft `preferred3DRenderer` bestimmt.

Diese Eigenschaft gibt den gleichen Wert zurück wie die Filmeigenschaft `the active3dRenderer`.

Beispiel

Die folgende Anweisung zeigt den aktuell auf dem Benutzersystem verwendeten Renderer an.

```
-- Lingo syntax
put getRendererServices().renderer

// JavaScript syntax
put( getRendererServices().renderer);
```

Siehe auch

getRendererServices(), preferred3dRenderer, rendererDeviceList, active3dRenderer

rendererDeviceList

Syntax

```
getRendererServices().rendererDeviceList
```

Beschreibung

Diese 3D-Renderer-Eigenschaft gibt eine Liste mit Symbolen zurück, die die auf dem Client-Computer verfügbaren Renderer identifizieren. Der Inhalt dieser Liste bestimmt, welche Werte für die Eigenschaften `renderer` und `preferred3DRenderer` angegeben werden können. Diese Eigenschaft kann getestet, aber nicht eingestellt werden.

Diese Eigenschaft ist eine Liste, die folgende mögliche Werte enthalten kann:

- `#openGL`gibt die `openGL`-Hardwarebeschleunigungstreiber an, die sowohl auf Mac- als auch auf Windows-Plattformen verwendet werden können.
- `#directX9` gibt die DirectX 9-Hardwarebeschleunigungstreiber an, die nur auf Windows-Plattformen verwendet werden können.`#auto` legt den Renderer auf DirectX 9 fest. Auf Mac-Intel ist nur der Renderer `#OpenGL` verfügbar.
- `#directX7_0` gibt die DirectX 7-Hardwarebeschleunigungstreiber an, die nur auf Windows-Plattformen verwendet werden können.
- `#directX5_2` gibt die DirectX 5.2-Hardwarebeschleunigungstreiber an, die nur auf Windows-Plattformen verwendet werden können.
- `#software` gibt den in Director integrierten Software-Renderer an, der sowohl auf Mac- als auch auf Windows-Plattformen verwendet werden kann.

Beispiel

Die folgende Anweisung zeigt die Renderer an, die auf dem aktuellen System verfügbar sind.

```
-- Lingo syntax
put getRendererServices().rendererDeviceList

// JavaScript syntax
put( getRendererServices().rendererDeviceList);
```

Siehe auch

`getRendererServices()`, `renderer`, `preferred3dRenderer`, `active3dRenderer`

renderFormat

Syntax

```
member(whichCastmember).texture(whichTexture).renderFormat
member(whichCastmember).texture[index].renderFormat
member(whichCastmember).shader(whichShader).texture.renderFormat
member(whichCastmember).model(whichModel).shader.texture.renderFormat
member(whichCastmember).model(whichModel).shader.textureList[index].renderFormat
member(whichCastmember).model(whichModel).shaderList[index].texture(whichTexture).renderForm
at
member(whichCastmember).model(whichModel).shaderList[index].textureList[index].renderFormat
```

Beschreibung

Mit dieser 3D-Eigenschaft können Sie durch Angabe eines der folgenden Werte das `textureRenderFormat` für eine bestimmte Textur ermitteln oder festlegen:

`#default` verwendet das von `getRendererServices().textureRenderFormat` zurückgegebene Format.

```
#rgba8888
#rgba8880
#rgba5650
#rgba5550
#rgba5551
#rgba4444
```

Weitere Informationen zu diesen Werten finden Sie unter `textureRenderFormat`.

Bei Festlegung dieser Eigenschaft für eine Einzeltextur wird die mit `textureRenderFormat` festgelegte globale Einstellung überschrieben.

Die Eigenschaft `renderFormat` bestimmt das Pixelformat, das beim Rendern der angegebenen Textur vom Renderer verwendet wird. Pixelformate bestehen aus mehreren Ziffern – jede Ziffer gibt die Farbtiefe für Rot, Grün, Blau und Alpha an. Der von Ihnen gewählte Wert bestimmt die Farbtreue (einschließlich der Genauigkeit des optionalen Alphakanals) und somit den auf der Videokarte belegten Speicherplatz. Sie können einen Wert wählen, der die Farbtreue verbessert, oder einen Wert, der es erlaubt, mehr Texturen in den Speicher auf der Videokarte zu laden. Hierbei beanspruchen 16-Bit-Texturen ungefähr nur halb so viel Speicherplatz wie 32-Bit-Texturen.

Beispiel

Die folgende Anweisung setzt die Eigenschaft `renderFormat` der Textur `TexPic` auf `#rgba4444`. Die Rot-, Blau-, Grün- und Alphakomponenten der Textur bestehen aus jeweils 4 Bit.

```
-- Lingo syntax
member("3Dobjects").texture[1].renderFormat = #rgba4444
put(member("3Dobjects").texture[1].renderFormat )
```

Siehe auch

`textureRenderFormat`, `getHardwareInfo()`

renderStyle

Syntax

```
member(whichCastmember).shader(whichShader).renderStyle
```

Beschreibung

Mit dieser 3D-Eigenschaft für den Standard-Shader können Sie den durch die Geometrie der zugrunde liegenden Modellressource bestimmten `renderStyle` eines Shaders ermitteln und festlegen. Diese Eigenschaft kann folgende Werte aufweisen:

#fill gibt an, dass der Shader die Oberfläche der Modellressource komplett ausfüllt.

#wire gibt an, dass der Shader nur an den Kanten der Modellressourcenflächen gezeichnet wird.

#point gibt an, dass der Shader nur an den Scheitelpunkten der Modellressource gezeichnet wird.

Alle Shader können auf die Eigenschaften des `#standard`-Shaders zugreifen. Darüber hinaus haben die Shader `#engraver`, `#newsprint` und `#painter` noch weitere Eigenschaften, die speziell für den jeweiligen Shader-Typ gelten. Weitere Informationen hierzu finden Sie unter dem Eintrag `newShader`.

Beispiel

Die folgende Anweisung bewirkt, dass der erste Shader nur an den Stellen gerendert wird, an denen er sich über einem Scheitelpunkt der zugrunde liegenden Modellressource befindet.

```
-- Lingo syntax
member("3Dobjects").shader[1].renderStyle = #point

// JavaScript syntax
    member("3Dobjects").getPropRef("shader",1).renderStyle = symbol("point");
```

resizable

Syntax

```
-- Lingo syntax
windowObjRef.resizable

// JavaScript syntax
windowObjRef.resizable;
```

Beschreibung

Diese Fenstereigenschaft gibt an, ob die Fenstergröße angepasst werden kann (`TRUE`, Standard) oder nicht (`FALSE`).

Lesen/Schreiben.

Beispiel

Die folgenden Anweisungen maximieren das Fenster namens „Empire“, wenn dessen Größe anpassbar ist.

```
-- Lingo syntax
if (window("Empire").resizable = TRUE) then
    window("Empire").maximize()
end if

// JavaScript syntax
if (window("Empire").resizable == true) {
    window("Empire").maximize();
}
```

Siehe auch

`Window`

resolution (3D)

Syntax

```
member(whichCastmember).modelResource(whichModelResource).resolution
```

Beschreibung

Mit dieser 3D-Eigenschaft können Sie die Auflösung einer Modellressource vom Typ `#sphere` oder `#cylinder` ermitteln und festlegen.

Die Auflösung bestimmt die Anzahl von Polygonen, die zur Erzeugung der Geometrie der Modellressource verwendet werden. Bei einem größeren Wert werden mehr Polygone und somit eine glattere Oberfläche erzeugt. Der Standardwert dieser Eigenschaft lautet 20.

Beispiel

Die folgende Anweisung setzt die Auflösung der Modellressource „Kugel01“ auf 10,0:

```
member("3D World").modelResource("sphere01").resolution = 10.0
```

resolution (DVD)

Syntax

```
-- Lingo syntax
dvdObjRef.resolution

// JavaScript syntax
dvdObjRef.resolution;
```

Beschreibung

Diese DVD-Eigenschaft gibt eine Eigenschaftsliste zurück, die die Quellauflösung der `x`-Achse (Breite) und `y`-Achse (Höhe) enthält. Nur Lesen.

Beispiel

Die folgende Anweisung gibt eine Beispieleigenschaftsliste von Auflösungen zurück:

```
-- Lingo syntax
trace(member(1).resolution) -- [#width: 720, #height: 480]

// JavaScript syntax
trace(member(1).resolution); // ["width": 720, "height": 480]
```

Siehe auch

DVD

resolve

Syntax

```
member(whichCastmember).model(whichModel).collision.resolve
```

Beschreibung

Mit dieser 3D-Kollisionseigenschaft können Sie ermitteln und festlegen, ob Kollisionen beim Zusammenstoß von zwei Modellen aufgelöst werden. Wenn diese Eigenschaft für beide an einer Kollision beteiligte Modelle auf `TRUE` gesetzt ist, kommen beide Modelle am Kollisionspunkt zum Stillstand. Wenn nur bei einem Modell die Eigenschaft `resolve` auf `TRUE` gesetzt ist, kommt dieses Modell zum Stillstand und das Modell, bei dem diese Eigenschaft nicht definiert bzw. auf `FALSE` gesetzt ist, bewegt sich auch weiterhin. Der Standardwert dieser Eigenschaft lautet `TRUE`.

Beispiel

Die folgende Anweisung setzt die Eigenschaft resolve des auf das Modell "Box" angewendeten Modifizierers collision auf TRUE. Wenn das Modell „Box“ mit einem anderen Modell zusammenstößt, an dem der Modifizierer #collision angebracht ist, kommt dieses zum Stillstand.

```
member("3d world").model("Box").collision.resolve = TRUE
```

Siehe auch

collisionData, collisionNormal, modelA, modelB, pointOfContact

resource

Syntax

```
member(whichCastmember).model(whichModel).resource
```

Beschreibung

Mit dieser 3D-Eigenschaft können Sie die Eigenschaft "resource" ermitteln und festlegen, die die Geometrie der referenzierten Modellressource definiert. Über diese Eigenschaft können Sie auch auf das Ressourcenobjekt des referenzierten Modells und die zugehörigen Eigenschaften zugreifen.

Beispiel

Die folgende Anweisung zeigt die Eigenschaft radius der vom zweiten Modell verwendeten Modellressource an.

```
-- Lingo syntax
put member("3Dobjects").model[2].resource.radius

// JavaScript syntax
put(member("3Dobjects").getPropRef("model",2).resource.radius);
```

right

Syntax

```
-- Lingo syntax
spriteObjRef.right

// JavaScript syntax
spriteObjRef.right;
```

Beschreibung

Diese Sprite-Eigenschaft gibt an, wie weit (in Pixel) der rechte Rand eines Sprites vom linken Rand der Bühne entfernt ist. Lesen/Schreiben.

Sprite-Koordinaten werden relativ zur linken oberen Ecke der Bühne ausgedrückt.

Beispiel

Die folgende Anweisung gibt den Abstand des rechten Randes eines Sprites zurück:

```
-- Lingo syntax
put(sprite(6).right)

// JavaScript syntax
put(sprite(6).right);
```

Siehe auch

bottom, height, left, locH, locV, Sprite, top, width

right (3D)

Syntax

```
member(whichCastmember).modelResource
(whichModelResource).right
modelResourceObjectReference.right
```

Beschreibung

Mit dieser 3D-Eigenschaft können Sie die Eigenschaft `right` einer Modellressource vom Typ `#box` ermitteln und festlegen.

Die Eigenschaft `right` bestimmt, ob die rechte Seite der Box geschlossen (`TRUE`) oder offen ist (`FALSE`). Der Standardwert ist `TRUE`.

Beispiel

Die folgende Anweisung setzt die Eigenschaft "right" der Modellressource "Kiste" auf TRUE, d. h. die rechte Seite der Box ist geschlossen:

```
member("3D World").modelResource("crate").right = TRUE
```

Siehe auch

bottom (3D), left (3D), top (3D)

rightIndent

Syntax

```
chunkExpression.rightIndent
```

Beschreibung

Die folgende Anweisung rückt den Textdarsteller `myText` um zehn Pixel nach rechts ein.

```
-- Lingo syntax
member("myText").rightIndent=10

// JavaScript syntax
member("myText").rightIndent=10;
```

Siehe auch

firstIndent, leftIndent

rightMouseDown

Syntax

```
-- Lingo syntax
_mouse.rightMouseDown

// JavaScript syntax
_mouse.rightMouseDown;
```

Beschreibung

Diese Mauseigenschaft zeigt an, ob die rechte Maustaste (Windows) bzw. die Ctrl- und Maus-Taste (Mac) gedrückt sind (`TRUE`) oder nicht (`FALSE`). Nur Lesen.

Auf dem Mac ist `rightMouseDown` nur dann `TRUE`, wenn die Eigenschaft `emulateMultiButtonMouse` ebenfalls `TRUE` ist.

Beispiel

Die folgende Anweisung prüft, ob in Windows die rechte Maustaste gedrückt wird. Wenn ja, wird im Soundkanal 2 der Sound „Hoppla“ abgespielt.

```
-- Lingo syntax
if (_mouse.rightMouseDown) then
    sound(2).play(member("Oops"))
end if

// JavaScript syntax
if (_mouse.rightMouseDown) {
    sound(2).play(member("Oops"));
}
```

Siehe auch

`emulateMultibuttonMouse`, `Mouse`

rightMouseUp

Syntax

```
-- Lingo syntax
_mouse.rightMouseUp

// JavaScript syntax
_mouse.rightMouseUp;
```

Beschreibung

Diese Mauseigenschaft zeigt an, ob die rechte Maustaste (Windows) bzw. Maus- und Ctrl-Taste (Mac) derzeit nicht gedrückt (`TRUE`) oder gedrückt (`FALSE`) werden. Nur Lesen.

Auf dem Mac ist `rightMouseUp` nur dann `TRUE`, wenn die Eigenschaft `emulateMultiButtonMouse` ebenfalls `TRUE` ist.

Beispiel

Die folgende Anweisung prüft in Windows, ob die rechte Maustaste losgelassen ist. In diesem Fall wird der Sound „Klick mich“ abgespielt.

```
-- Lingo syntax
if (_mouse.rightMouseUp) then
    sound(2).play(member("Click Me"))
end if

// JavaScript syntax
if (_mouse.rightMouseUp) {
    sound(2).play(member("Click Me"));
}
```

Siehe auch

emulateMultibuttonMouse, Mouse

romanLingo

Syntax

```
the romanLingo
```

Beschreibung

Diese Systemeigenschaft bestimmt, ob Lingo einen Einzelbyte-Interpreter (TRUE) oder einen Doppelbyte-Interpreter (FALSE) verwendet.

Bei Einzelbyte-Zeichensätzen läuft der Lingo-Interpreter schneller. Einige Versionen der Mac-Systemsoftware (z. B. die japanische Version) verwenden einen Doppelbyte-Zeichensatz. Die US-Systemsoftware verwendet den Einzelbyte-Zeichensatz. In der Regel wird die Eigenschaft romanLingo beim erstmaligen Start von Director eingestellt und hängt ganz von der lokalen Version der Systemsoftware ab.

Wenn Sie ein Schriftsystem verwenden, das nicht auf lateinischen Buchstaben beruht, aber im Skript keine Doppelbyte-Zeichen benutzen, sollten Sie diese Eigenschaft auf TRUE setzen, damit Lingo-Skripts schneller ausgeführt werden können.

Beispiel

Die folgende Anweisung setzt die Eigenschaft romanLingo auf TRUE, damit Lingo dann mit einem Einzelbyte-Zeichensatz arbeitet:

```
-- Lingo
set the romanLingo to TRUE
```

Siehe auch

inlineImeEnabled

rootLock

Syntax

```
member(whichCastmember).model(whichModel).keyframePlayer.rootLock
member(whichCastmember).model(whichModel).bonesPlayer.rootLock
```

Beschreibung

Diese 3D-Eigenschaft für die Modifizierer `#keyframePlayer` und `#bonesPlayer` gibt an, ob die Translationskomponenten einer Bewegung verwendet (`FALSE`) oder ignoriert werden (`TRUE`).

Der Standardwert dieser Eigenschaft lautet `FALSE`.

Beispiel

Die folgende Anweisung bewirkt, dass das Modell „Kreatur3“, bei dem es ich um das erste Modell handelt, bei der Ausführung von Bewegungen an seiner Ausgangsposition bleibt, sodass eine Person erscheint, die auf der Stelle geht:

```
-- Lingo
member("newalien").model("Alien3").keyframePlayer.rootLock = 1

// Javascript
member("newalien").getPropRef("model",1).keyframePlayer.rootLock=1
```

rootNode

Syntax

```
member(whichCastmember).camera(whichCamera).rootNode
sprite(whichSprite).camera.rootNode
```

Beschreibung

Mit dieser 3D-Eigenschaft können Sie ermitteln und festlegen, welche Objekte in einem Sprite sichtbar sind. Wenn eine Kamera erstellt wird, zeigt sie alle Nodes in der Welt an. Mit der Eigenschaft `rootNode` können Sie dies ändern, indem Sie eine andere Standardansicht erstellen, mit der die für einen bestimmten Node und seine Child-Objekte sichtbaren Elemente eingeschränkt werden.

Angenommen, Licht C sei ein Child-Objekt von Modell A. Wenn Sie die Eigenschaft `rootNode` auf `camera("defaultView").rootNode=model(A)` setzen, zeigt das Sprite nur Modell A, wie es durch Licht C beleuchtet wird. Der Standardwert ist `group("world")`, d. h. es werden alle Nodes benutzt.

Beispiel

Die folgende Anweisung setzt den `rootNode` der Kamera von Sprite 5 auf das Modell "Pluto". In Sprite 5 sind nur Pluto und seine Child-Objekte sichtbar.

```
-- Lingo
sprite(5).camera.rootNode = member("Scene").model("Pluto")

// Javascript
sprite(5).getPropRef("camera",1).rootNode=member("Scene").getPropRef("model",1);
```

rotation

Syntax

```
-- Lingo syntax
spriteObjRef.rotation

// JavaScript syntax
spriteObjRef.rotation;
```

Beschreibung

Diese Sprite-Eigenschaft steuert die Drehung eines QuickTime-Film-, animierten GIF-, Flash-Film- oder Bitmap-Sprites innerhalb des Begrenzungsrechtecks des Sprites, ohne dass dabei das Rechteck oder (im Falle von QuickTime) der Sprite-Regler gedreht wird. Lesen/Schreiben.

Das Begrenzungsrechteck des Sprites fungiert praktisch als ein Fenster, durch das Sie den Flash- oder QuickTime-Film betrachten können. Das Begrenzungsrechteck von Bitmaps und animierten GIFs ändert sich automatisch, um die drehende Grafik aufnehmen zu können.

Die Drehbuchdrehung funktioniert bei einem Flash-Film nur dann, wenn `obeyScoreRotation` auf `TRUE` gesetzt ist.

Ein Flash-Film dreht sich um den in der Eigenschaft `originMode` angegebenen Ursprungspunkt. Ein QuickTime-Film dreht sich dagegen um die Mitte des Begrenzungsrechtecks des Sprites. Eine Bitmap dreht sich um das Registrierungskreuz der Grafik.

Wenn bei QuickTime-Medien die Sprite-Eigenschaft `crop` auf `TRUE` gesetzt ist, kann es beim Drehen des Sprites häufig passieren, dass ein Teil der Grafik nicht mehr zu sehen ist. Wenn die Sprite-Eigenschaft `crop` dagegen auf `FALSE` gesetzt ist, wird die Grafik so skaliert, dass sie genau in das Begrenzungsrechteck passt (wodurch die Grafik aber evtl. verzerrt wird).

Sie geben die Drehung in Grad als Fließkommazahl an.

Das Drehbuch kann Informationen zum Drehen einer Grafik von +21.474.836,47° bis -21.474.836,48° speichern, wodurch 59.652 volle Drehungen in beide Richtungen möglich sind.

Sobald die Drehungsgrenze erreicht ist (d. h. gleich nach der 59.652sten Drehung), wird die Drehung auf +116,47° oder -116,48° (d. h. nicht auf 0,00°) zurückgesetzt. Der Grund hierfür ist, dass +21,474,836.47° das Gleiche ist wie +116.47° und -21,474,836.48° dem Wert -116.48° (oder +243.12°) entsprechen. Wenn Sie Skriptcode zur kontinuierlichen Drehung verwenden und verhindern möchten, dass eine Rücksetzung erfolgt, begrenzen Sie die Winkel auf ±360 °.

Der Standardwert dieser Eigenschaft lautet 0.

Beispiel

Das folgende Verhalten bewirkt, dass ein Sprite bei jedem Abspielkopfvorschub kontinuierlich um 2° gedreht wird, wobei der Winkel auf insgesamt 360° begrenzt ist:

```
-- Lingo syntax
property spriteNum

on prepareFrame me
    sprite(spriteNum).rotation = integer(sprite(spriteNum).rotation + 2) mod 360
end

// JavaScript syntax
function prepareFrame() {
        sprite(this.spriteNum).rotation = parseInt(sprite(this.spriteNum).rotation + 2) % 360;
}
```

Das folgende Bildskript bewirkt, dass der Abspielkopf im aktuellen Bild eine Schleife durchläuft, während ein QuickTime-Sprite in Kanal 5 in Schritten von jeweils 16° um volle 360° gedreht wird. Sobald das Sprite um insgesamt 360° gedreht ist, geht der Abspielkopf zum nächsten Bild über.

```
-- Lingo syntax
on rotateMovie(whichSprite)
    repeat with i = 1 to 36
        sprite(whichSprite).rotation = i * 10
        _movie.updateStage()
    end repeat
end

// JavaScript syntax
function rotateMovie(whichSprite) {
    for (var i = 1; i <= 36; i++) {
        sprite(whichSprite).rotation = i * 10;
        _movie.updateStage();
    }
}
```

Siehe auch

obeyScoreRotation, originMode, Sprite

rotation (backdrop and overlay)

Syntax

```
sprite(whichSprite).camera.backdrop[backdropIndex].rotation
member(whichCastmember).camera(whichCamera).backdrop
[backdropIndex].rotation
sprite(whichSprite).camera.overlay[overlayIndex].rotation
member(whichCastmember).camera[cameraIndex].overlay
[overlayIndex].rotation
```

Beschreibung

Mit dieser 3D-Eigenschaft können Sie die Drehung des Hintergrunds bzw. der Überlagerung zur Standardkamera hin ermitteln und festlegen. Der Standardwert dieser Eigenschaft lautet 0.0.

Beispiel

Die folgende Anweisung dreht den Hintergrund 60° um sein Registrierungskreuz:

```
sprite(4).camera.backdrop[1].rotation = 60.0
```

Siehe auch

bevelDepth, transform (Eigenschaft)

rotation (engraver shader)

Syntax

```
member(whichCastmember).shader(whichShader).rotation
member(whichCastmember).model(whichModel).shader.rotation
member(whichCastmember).model(whichModel).shaderList[index].rotation
```

Beschreibung

Mit dieser 3D-Eigenschaft für Shader vom Typ #engraver können Sie einen Winkel (in Grad als Fließkommazahl) ermitteln und festlegen, der die 2D-Rotationsverschiebung für gravierte Linien beschreibt. Der Standardwert dieser Eigenschaft lautet 0.0.

Beispiel

Die folgende Anweisung dreht die zum Zeichnen des Shaders #engraver verwendeten Linien beim Modell `gbCyl3` um 1°:

```
member("scene").model("gbCyl3").shader.rotation = \
member("scene").model("gbCyl3").shader.rotation + 1
```

Siehe auch

transform (Eigenschaft)

rotation (Transformation)

Syntax

```
member(whichCastmember).node(whichNode).transform.rotation
member(whichCastmember).node(whichNode).getWorldTransform().rotation
transform.rotation
```

Beschreibung

Mit dieser 3D-Eigenschaft können Sie die Drehungskomponente einer Transformation ermitteln und festlegen. Eine Transformation definiert eine Skalierung, Position und Drehung in einem bestimmten Bezugsrahmen. Der Standardwert dieser Eigenschaft lautet `vector(0,0,0)`.

Ein Node kann ein Kamera-, Gruppen-, Licht- oder Modellobjekt sein. Durch Festlegung der Eigenschaft `rotation` der Transformation eines Nodes wird die Drehung dieses Objekts im Bezugsrahmen der Transformation definiert. Wenn Sie mit `getWorldTransform().rotation` die Eigenschaft `rotation` der weltbezogenen Transformation eines Objekts festlegen, wird die Drehung des Objekts bezogen auf den Weltursprung definiert. Wenn Sie mit `transform.rotation` die Eigenschaft `rotation` der Parent-bezogenen Transformation eines Objekts festlegen, wird die Drehung des Objekts bezogen auf die Parent-Node definiert.

Wenn Sie die Ausrichtung einer Transformation ändern möchten, sollten Sie statt dieser Eigenschaft die Methoden `rotate` und `prerotate` verwenden.

Beispiel

Die folgende Anweisung setzt die parentbezogene Drehung der ersten Kamera im Darsteller auf vector(0,0,0):

```
member("Space").camera[1].transform.rotation = vector(0, 0, 0)
```

Im folgenden Beispiel wird zuerst die Parent-bezogene Drehung des Modells „Mond“ angezeigt, dann die Ausrichtung des Modells mit dem Befehl rotate geändert und abschließend die daraus resultierende weltbezogene Drehung des Modells angezeigt:

```
put member("SolarSys").model("Moon").transform.rotation
-- vector( 0.0000, 0.0000, 45.0000)
member("SolarSys").model("Moon").rotate(15,15,15)
put member("SolarSys").model("Moon").getWorldTransform().rotation
--vector( 51.3810, 16.5191, 65.8771 )
```

Siehe auch

`getWorldTransform()`, `preRotate`, `rotate`, `transform (Eigenschaft)`, `position (Transformation)`, `scale (Transformation)`

rotationReset

Syntax

```
member(whichCastmember).model(whichModel).bonesPlayer.rotationReset
member(whichCastmember).model(whichModel).keyframePlayer.rotationReset
```

Beschreibung

Diese 3D-Eigenschaft für die Modifizierer `keyframePlayer` und `bonesPlayer` gibt die Achsen an, um die Drehungsänderungen zwischen dem Ende einer Bewegung und dem Anfang der nächsten Bewegung bzw. zwischen dem Ende einer Iteration einer in Schleife ablaufenden Bewegung und dem Anfang der nächsten Iteration beibehalten werden.

Mögliche Werte: `#none`, `#x`, `#y`, `#z`, `#xy`, `#yz`, `#xz` und `#all`. Der Standardwert ist `#all`.

Beispiel

Die folgende Anweisung setzt die Eigenschaft `rotationReset` des Modells "Monster" auf die *z*-Achse. Das Modell behält die Drehung um die *z*-Achse bei, wenn das Ende der aktuellen Bewegung bzw. Schleife erreicht ist.

```
member("NewAlien").model("Monster").bonesPlayer.rotationReset = #z
```

Siehe auch

`positionReset`, `bonesPlayer (Modifizierer)`

RTF

Syntax

```
-- Lingo syntax
memberObjRef.RTF

// JavaScript syntax
memberObjRef.RTF;
```

Beschreibung

Diese Darstellereigenschaft ermöglicht den Zugriff auf den Text und die Tags, durch die das Textlayout in einem Textdarsteller mit Text im Rich-Text-Format (RTF) gesteuert wird.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Die folgende Anweisung zeigt im Nachrichtenfenster die im Textdarsteller "Lebenslauf" eingebetteten RTF-Formatierungsinformationen an:

```
--Lingo syntax
put(member("Resume").RTF)

// JavaScript syntax
trace(member("Resume").RTF);
```

Siehe auch

HTML, importFileInto()

safePlayer

Syntax

```
-- Lingo syntax
_player.safePlayer

// JavaScript syntax
_player.safePlayer;
```

Beschreibung

Diese Player-Eigenschaft bestimmt, ob in Director die Sicherheitsfunktionen aktiviert sind oder nicht. Lesen/Schreiben.

In einem Film mit Shockwave-Inhalt kann die Eigenschaft getestet, aber nicht eingestellt werden. Diese Eigenschaft ist in Shockwave stets auf `TRUE` gesetzt.

In der Erstellungsumgebung und in Projektoren lautet der Standardwert `FALSE`. Diese Eigenschaft kann zurückgegeben, aber nur auf `TRUE` gesetzt werden. Sobald sie auf `TRUE` gesetzt wurde, kann sie nur dann erneut auf `FALSE` zurückgesetzt werden, wenn Director oder der Projekt neu gestartet wird.

Wenn `safePlayer` auf `TRUE` gesetzt ist, sind die folgende Sicherheitsfunktionen aktiviert:

- Es können nur sichere Xtra-Erweiterungen verwendet werden.

- Die Eigenschaft `safePlayer` kann nicht zurückgesetzt werden.
- Beim Einfügen von Daten aus der Zwischenablage mit der `pasteClipBoardInto()`-Methode erscheint ein Warndialogfeld, in dem der Benutzer den Vorgang abbrechen kann.
- Filme oder Besetzungen können nicht mit Skriptcode gespeichert werden.
- Mit der `printFrom()`-Methode kann nicht gedruckt werden.
- Anwendungen können nicht mit der `open()`-Methode geöffnet werden.
- Anwendungen oder der Computer des Benutzers können nicht mit der Methode `restart()` oder `shutDown()` beendet bzw. heruntergefahren werden.
- Dateien außerhalb des Ordners "DSWMedia" können nicht geöffnet werden.
- Es können keine lokalen Dateinamen ermittelt werden.
- Die Verwendung von `getNetText()` oder `postNetText()` bzw. der Zugriff auf eine URL mit einer anderen Domäne als der Film führt zum Einblenden eines Sicherheitsdialogfelds.

Beispiel

Die folgende Anweisung überprüft den für die Einstellung `safePlayer` in der Erstellungsumgebung festgelegten Wert.

```
-- Lingo
put( _player.safePlayer)
--0

// Javascript
trace(_player.safePlayer)
// 0
```

Siehe auch

`Player`

sampleCount (Soundkanal)

Syntax

```
-- Lingo syntax
soundChannelObjRef.sampleCount

// JavaScript syntax
soundChannelObjRef.sampleCount;
```

Beschreibung

Diese Soundkanaleigenschaft gibt die Anzahl der Soundsamples im aktuell in einem Soundkanal wiedergegebenen Sound an. Nur Lesen.

Hierbei handelt es sich um die Gesamtzahl von Samples; dieser Wert hängt von der Samplerate, `sampleRate`, und der Dauer des Sounds, `duration`, ab. Die Kanalzahl, `channelCount`, des Sounds ist dabei irrelevant.

Ein 1 Sekunde langer Sound mit der Frequenz 44,1 kHz enthält 44.100 Samples.

Beispiel

Die folgende Anweisung zeigt den Namen und die Anzahl von Samples (`sampleCount`) des Darstellers, der gegenwärtig in Soundkanal 1 abgespielt wird, im Nachrichtenfenster an:

```
-- Lingo syntax
put("Sound cast member" && sound(1).member.name && "contains" && sound(1).sampleCount &&
"samples.")

// JavaScript syntax
put("Sound cast member " + sound(1).member.name + " contains " +sound(1).sampleCount + "
samples.");
```

Siehe auch

sampleRate (Soundkanal), Sound Channel

sampleCount (Soundobjekt)

Syntax

```
soundObject.sampleCount (Read-only)
```

Beschreibung

Diese Soundobjekteigenschaft gibt die Anzahl der Samples (Abtastwerte) im aktuellen Soundobjekt zurück.

Beispiele

```
--Lingo syntax
on mouseUp me
    put soundObjRef.sampleCount -- Displays the number of samples in the sound
-- associated with soundobjectRef.
end

// JavaScript syntax
function mouseUp(){
put (soundObjRef.sampleCount); // Displays the number of samples in the sound
// associated with soundobjectRef.
}
```

sampleRate (Mixer)

Syntax

```
mixer.sampleRate (Read-write)
```

Beschreibung

Diese Mixereigenschaft gibt die Samples (Abtastwerte) pro Sekunde für den Mixer aus. Sie können die `sampleRate` auf jede beliebige Standard-Samplerate einstellen.

Beispiel

```
--Lingo syntax
on mouseUp me
mixerRef.sampleRate = 44100 -- Sets the sample rate of the mixer to 44100.
end

// JavaScript syntax
function mouseUp(){
mixerRef.sampleRate = 44100; // Sets the sample rate of the mixer to 44100.
}
```

Siehe auch

Mixer

sampleRate (Soundkanal)

Syntax

```
-- Lingo syntax
soundChannelObjRef.sampleRate

// JavaScript syntax
soundChannelObjRef.sampleRate;
```

Beschreibung

Diese Soundkanaleigenschaft gibt die Samplerate des Sounddarstellers (in Samples pro Sekunde) bzw. bei SWA-Sounds die Shockwave Audio-kodierte Originaldatei zurück. Nur Lesen.

Diese Eigenschaft ist erst verfügbar, nachdem die Wiedergabe des SWA-Sounds begonnen hat oder nachdem die Datei mithilfe der `preLoadBuffer()`-Methode vorausgeladen wurde. Bei Angabe eines Soundkanals ist das Ergebnis die Samplerate des Sounddarstellers, der gegenwärtig im angegebenen Soundkanal abgespielt wird.

Typische Werte dieser Eigenschaft: 8000, 11025, 16000, 22050 und44100.

Wenn mehrere Sounds in einem Soundkanal auf die Wiedergabe warten, verwendet Director zum Abspielen aller Sounds die Eigenschaften `channelCount`, `sampleRate` und `sampleSize` des ersten Sounds in der Warteschlange und führt für den Rest ein Resampling durch, um eine flüssige Wiedergabe zu gewährleisten. Director setzt diese Eigenschaften erst dann zurück, wenn alle Sounds in der Warteschlange des Kanals abgespielt sind oder wenn die `stop()`-Methode ausgeführt wird. Die neuen Einstellungen richten sich dann nach dem nächsten Sound, der in die Warteschlange gestellt oder abgespielt wird.

Beispiel

Die folgende Anweisung ordnet dem Felddarsteller „Soundqualität“ die ursprüngliche Samplerate der im SWA-Streaming-Darsteller „Paul Robeson“ verwendeten Datei zu:

```
-- Lingo syntax
member("Sound Quality").text = string(member("Paul Robeson").sampleRate)

// JavaScript syntax
member("Sound Quality").text = member("Paul Robeson").sampleRate.toString();
```

Die folgende Anweisung zeigt die Samplerate des Sounds, der in Soundkanal 1 abgespielt wird, im Nachrichtenfenster an:

```
-- Lingo syntax
trace(sound(1).sampleRate)

// JavaScript syntax
trace(sound(1).sampleRate);
```

Siehe auch

channelCount (Soundkanal), sampleSize, preLoadBuffer(), Sound Channel, stop() (Soundkanal)

sampleRate (Soundobjekt)

Syntax

```
soundObject.sampleRate (Read-only)
```

Beschreibung

Diese Soundobjekteigenschaft gibt die Samplingfrequenz (Abtastrate pro Sekunde) des aktuellen Soundobjekts zurück.

Beispiele

```
-- Lingo syntax
on mouseUp me
    put soundObjRef.sampleRate -- Displays the samples per second of the
-- sound object associated with soundobjectRef.
end

// JavaScript syntax
function mouseUp(){
put (soundObjRef.sampleRate) ; // Displays the samples per second of the
// sound object associated with soundobjectRef.
}
```

sampleSize

Syntax

```
-- Lingo syntax
memberObjRef.sampleSize

// JavaScript syntax
memberObjRef.sampleSize;
```

Beschreibung

Diese Darstellereigenschaft bestimmt die Samplegröße des angegebenen Darstellers. Das Ergebnis ist normalerweise 8 oder 16 Bits. Bei Angabe eines Soundkanals wird der Wert für den Sounddarsteller zurückgegeben, der gegenwärtig im angegebenen Soundkanal abgespielt wird.

Diese Eigenschaft kann getestet, aber nicht eingestellt werden.

Beispiel

Die folgende Anweisung prüft die Samplegröße des Sounddarstellers "Begleitkommentar" und ordnet den Wert der Variablen `soundSize` zu:

```
-- Lingo syntax
soundSize = member("Voice Over").sampleSize

// JavaScript syntax
var soundSize = member("Voice Over").sampleSize;
```

Die folgende Anweisung zeigt die Samplegröße des Sounds, der in Soundkanal 1 abgespielt wird, im Nachrichtenfenster an:

```
-- Lingo syntax
put(sound(1).sampleSize)

// JavaScript syntax
put(sound(1).sampleSize);
```

savew3d

Syntax

```
--Lingo syntax
member(whichcastmember).savew3d(Absolute path (optional argument))
// JavaScript syntax
member(whichcastmember).savew3d(Absolute path (optional argument));
```

Beschreibung

Speichert die 3D-Welt von einem Projektor aus.

Diese Methode verhält sich wie folgt:

- Wird der absolute Pfad angegeben, speichert diese Methode die 3D-Szene in diesem Augenblickt in der Datei, deren Pfadnamen angegeben ist (vorausgesetzt, die Pfadangabe stimmt). Dies erfolgt sowohl im Authoring- als auch im Projektormodus.
- Wird der absolute Pfad nicht angegeben (d. h. der Aufruf lautet „darstellername.savew3d()“), ist das Verhalten wie folgt:
  - Wenn der „w3d“-Darsteller extern verknüpft ist und „savew3d“ im Projektormodus aufgerufen wird, dann wird die 3D-Szene in diesem Augenblick in der extern verknüpften „w3d“-Datei gespeichert, wobei die Datei überschrieben wird.
  - Wenn der „w3d“-Darsteller intern ist und „savew3d“ im Projektormodus aufgerufen wird, dann wird dieser Methodenaufruf ignoriert.

Im Authoring-Modus verhält sich die „savew3d“-Methode wie die `saveWorld`-Methode. Der Darsteller ist für den Schreibvorgang markiert, und der 3D-Speichervorgang erfolgt, nachdem Sie die Director-Datei gespeichert haben.

Beispiel

Diese Anweisung speichert die Änderungen am Darsteller („3dworld“) in der getrennten Datei „savedworld.w3d“.

```
-- Lingo syntax
member("3dworld").savew3d(the moviepath & "savedworld.w3d")

// JavaScript syntax
member("3dworld").savew3d(the moviepath & "savedworld.w3d");
```

saveWorld

Syntax

```
--Lingo syntax
member(whichcastmember).saveWorld()
// JavaScript syntax
member(whichcastmember).saveWorld();
```

Beschreibung

Speichert die vorhandene 3D-Welt im 3D-Darsteller, nachdem Sie den Film gespeichert haben.

Beispiel

Diese Anweisung markiert den Darsteller als bearbeitet, damit er beim Speichern des Director-Films in die Datei geschrieben wird. Der Darsteller wird dabei nicht eine separate Datei gespeichert.

```
-- Lingo syntax
member("3dworld").saveWorld()

// JavaScript syntax
member("3dworld").saveWorld();
```

scale (3D)

Syntax

```
member(whichCastmember).camera(whichCamera).backdrop[backdropIndex].scale
member(whichCastmember).camera(whichCamera).overlay[overlayIndex].scale
```

Beschreibung

Mit dieser 3D-Eigenschaft können Sie den Skalierungswert, der von einer bestimmten Überlagerung oder einem bestimmten Hintergrund in der Überlagerungs- oder Hintergrundliste der referenzierten Kamera verwendet wird, ermitteln und festlegen. Die Breite und Höhe des Hintergrunds bzw. der Überlagerung werden mit dem Skalierungswert multipliziert. Der Standardwert dieser Eigenschaft ist 1.0.

Beispiel

Die folgende Anweisung verdoppelt die Hintergrundgröße:

```
-- Lingo
sprite(25).camera.backdrop[1].scale = 2.0

// Javascript
sprite(25).getPropRef("camera",1).getProp("backDrop",1).scale=2.0;
```

Siehe auch

bevelDepth, overlay

scale (backdrop and overlay)

Syntax

```
member(whichCastmember).camera(whichCamera).backdrop[backdropIndex].scale
member(whichCastmember).camera(whichCamera).overlay[overlayIndex].scale
```

Beschreibung

Mit dieser 3D-Eigenschaft können Sie den Skalierungswert, der von einer bestimmten Überlagerung oder einem bestimmten Hintergrund in der Überlagerungs- oder Hintergrundliste der referenzierten Kamera verwendet wird, ermitteln und festlegen. Die Breite und Höhe des Hintergrunds bzw. der Überlagerung werden mit dem Skalierungswert multipliziert. Der Standardwert dieser Eigenschaft ist 1.0.

Beispiel

Die folgende Anweisung verdoppelt die Hintergrundgröße:

```
sprite(25).camera.backdrop[1].scale = 2.0
```

Siehe auch

bevelDepth, overlay

scale (Darsteller)

Syntax

```
-- Lingo syntax
memberOrSpriteObjRef.scale

// JavaScript syntax
memberOrSpriteObjRef.scale;
```

Beschreibung

Diese Darsteller- und Sprite-Eigenschaft bestimmt, wie ein QuickTime-, Vektorform- oder Flash-Film-Sprite skaliert wird.

Bei QuickTime-Sprites wird durch diese Eigenschaft weder das Begrenzungsrechteck noch der Sprite-Regler skaliert. Stattdessen wird die Grafik im Begrenzungsrechteck um den Mittelpunkt skaliert. Die Skalierung wird in Form einer Director-Liste angegeben, die zwei Prozentwerte als Fließkommazahlen enthält:

```
[xPercent, yPercent]
```

Der Parameter *xPercent* gibt die horizontale Skalierung und der Parameter *yPercent* die vertikale Skalierung an.

Wenn die Eigenschaft `crop` des Sprites auf `TRUE` gesetzt ist, kann die Eigenschaft `scale` dazu verwendet werden, innerhalb des Begrenzungsrechtecks des Sprites das Zoomen zu simulieren. Wenn die Sprite-Eigenschaft `crop` dagegen auf `FALSE` gesetzt ist, wird `scale` einfach ignoriert.

Diese Eigenschaft kann getestet und eingestellt werden. Der Standardwert ist [1.0000,1.0000].

Bei Flash-Film- oder Vektorformdarstellern wird die Skalierung in einem Fließkommawert ausgedrückt. Der Film wird vom Ursprungspunkt aus skaliert, der in der Eigenschaft `originMode` angegeben ist.

Hinweis: *Diese Eigenschaft muss auf den Standardwert eingestellt sein, wenn die Eigenschaft `scaleMode` auf `#autoSize` gesetzt ist. Andernfalls wird das Sprite nicht richtig angezeigt.*

Beispiel

Die folgende Prozedur akzeptiert einen Bezug auf ein Flash-Film-Sprite als Parameter, um die Skalierung des Films auf 0% zu reduzieren (sodass der Film nicht mehr zu sehen ist) und dann wieder auf 100% zu erhöhen, und zwar in Schritten von jeweils 5%:

```
-- Lingo syntax
on scaleMovie whichSprite
    sprite(whichSprite).scale = 0
    _movie.updatestage()
    repeat with i = 1 to 20
        sprite(whichSprite).scale = i * 5
        _movie.updatestage()
    end repeat
end

// JavaScript syntax
function scaleMovie(whichSprite) {
    sprite(whichSprite).scale = 0;
    _movie.updateStage();
    var i = 1;
    while (i < 21) {
        sprite(whichSprite).scale = i * 5;
        _movie.updateStage();
        i++;
    }
}
```

Siehe auch

`scaleMode`, `originMode`

scale (Transformation)

Syntax

```
member(whichCastmember).node(whichNode).transform.scale
member(whichCastmember).node(whichNode).getWorldTransform().scale
transform.scale
```

Beschreibung

Mit dieser 3D-Eigenschaft können Sie die Skalierungskomponente einer Transformation ermitteln und festlegen. Eine Transformation definiert eine Skalierung, Position und Drehung in einem bestimmten Bezugsrahmen. Mit der Eigenschaft `scale` können Sie den Skalierungsfaktor der Transformation entlang der drei Achsen ermitteln und festlegen. Der Standardwert dieser Eigenschaft lautet `vector(1.0,1.0,1.0)`.

Ein Node kann ein Kamera-, Gruppen-, Licht- oder Modellobjekt sein. Dieser Befehl hat keine sichtbare Auswirkung auf Lichtquellen oder Kameras, da diese keine Geometrie enthalten. Durch Festlegung der Eigenschaft `scale` der Transformation eines Nodes wird die Skalierung dieses Objekts entlang der x-, y- und z-Achse im Bezugsrahmen der Transformation definiert. Wenn Sie mit `getWorldTransform().scale` die Eigenschaft `scale` der weltbezogenen Transformation eines Objekts abrufen, wird die Skalierung des Objekts bezogen auf den Weltursprung definiert. Wenn Sie mit `transform.scale` die Eigenschaft `scale` der parentbezogenen Transformation eines Objekts festlegen, wird die Skalierung des Objekts bezogen auf den Parent-Node definiert.

Beispiel

Die folgende Anweisung setzt die Eigenschaft scale der Transformation des Modells "Mond" auf "vector(2,5,3)":

```
member("Scene").model("Moon").transform.scale = vector(2,5,3)
```

Siehe auch

`transform (Eigenschaft)`, `getWorldTransform()`, `position (Transformation)`, `rotation (Transformation)`, `scale (Befehl)`

scaleMode

Syntax

```
-- Lingo syntax
memberOrSpriteObjRef.scaleMode

// JavaScript syntax
memberOrSpriteObjRef.scaleMode;
```

Beschreibung

Diese Darsteller- und Sprite-Eigenschaft steuert, wie ein Flash-Film oder eine Vektorform innerhalb des Begrenzungsrechtecks des Sprites skaliert wird. Wenn Sie ein Flash-Film-Sprite durch Einstellen der Eigenschaften `scale` und `viewScale` skalieren, wird das Sprite selbst nicht skaliert, sondern nur die Filmansicht innerhalb des Sprites. Für die Eigenschaft `scaleMode` sind folgende Werte möglich:

- `#showAll` (Standardwert für Director-Filme vor Version 7) – Behält das Seitenverhältnis des ursprünglichen Flash-Filmdarstellers bei. Bei Bedarf können Sie mithilfe der Hintergrundfarbe Lücken in der horizontalen oder vertikalen Dimension ausfüllen.
- `#noBorder` – Behält das Seitenverhältnis des ursprünglichen Flash-Filmdarstellers bei. Bei Bedarf können Sie die horizontale oder vertikale Dimension zuschneiden.
- `#exactFit` – Behält das Seitenverhältnis des ursprünglichen Flash-Filmdarstellers nicht bei. Sie können den Flash-Film aber durch Strecken an die genauen Sprite-Abmessungen anpassen.
- `#noScale` – Behält die Originalgröße der Flash-Medien bei, ganz gleich, wie groß das Sprite auf der Bühne ist. Falls das Sprite kleiner als der Original-Flash-Film ist, wird der im Sprite angezeigte Film entsprechend zugeschnitten, damit er in das Begrenzungsrechteck des Sprites passt.
- `#autoSize` (Standard) – Dieser Wert bedeutet, dass das Rechteck des Sprites automatisch den Werten von `rotation`, `skew`, `flipH` und `flipV` entsprechend angepasst und positioniert wird. Wenn ein Flash-Sprite gedreht wird, wird es nicht zugeschnitten, wie es in früheren Versionen von Director der Fall war. Die Einstellung `#autoSize` funktioniert nur dann richtig, wenn für `scale`, `viewScale`, `originPoint` und `viewPoint` die Standardwerte verwendet werden.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Das folgende Sprite-Skript prüft die Bühnenfarbe des Director-Films und setzt die Eigenschaft `scaleMode` des Flash-Film-Sprites auf `#showAll`, wenn die Bühnenfarbe in der aktuellen Palette auf 0 indiziert ist. Falls die Bühnenfarbe nicht auf 0 indiziert ist, wird die Eigenschaft `scaleMode` auf `#noBorder` gesetzt.

```
-- Lingo syntax
property spriteNum

on beginsprite me
    if _movie.stage.bgColor = 0 then
        sprite(spriteNum).scaleMode = #showAll
    else
        sprite(spriteNum).scaleMode = #noBorder
    end if
end

// JavaScript syntax
function beginsprite() {
    var stgClr = _movie.stage.bgColor;
    if (stgClr == 0) {
        sprite(this.spriteNum).scaleMode = symbol("showAll");
    } else {
        sprite(this.spriteNum).scaleMode = symbol("noBorder");
    }
}
```

Siehe auch

scale (Darsteller)

score

Syntax

```
-- Lingo syntax
_movie.score

// JavaScript syntax
_movie.score;
```

Beschreibung

Diese Filmeigenschaft bestimmt, welches Drehbuch dem aktuellen Film zugeordnet ist. Lesen/Schreiben.

Diese Eigenschaft ist insbesondere dazu geeignet, um den aktuellen Inhalt des Drehbuchs zu speichern, bevor dieses gelöscht und ein neues Drehbuch erstellt wird, oder um den aktuellen Drehbuchinhalt einer Filmschleife zuzuordnen.

Beispiel

Die folgende Anweisung ordnet den Filmschleifendarsteller „Wasserfall“ dem Drehbuch des aktuellen Films zu:

```
-- Lingo syntax
_movie.score = member("Waterfall").media

// JavaScript syntax
_movie.score = member("Waterfall").media;
```

Siehe auch

Movie

scoreColor

Syntax

```
sprite(whichSprite).scoreColor
the scoreColor of sprite whichSprite
```

Beschreibung

Diese Sprite-Eigenschaft gibt die Drehbuchfarbe an, die dem in *whichSprite* angegebenen Sprite zugeordnet ist. Die möglichen Werte entsprechen den Farbfeldern 0 bis 5 der aktuellen Palette.

Diese Eigenschaft kann getestet und eingestellt werden. Die Einstellung diese Eigenschaft ist nur in der Erstellungsumgebung und beim Aufzeichnen des Drehbuchs von Nutzen.

Beispiel

Die folgende Anweisung zeigt im Nachrichtenfenster den Wert der Drehbuchfarbe an, die Sprite 7 zugeordnet ist:

```
put sprite(7).scorecolor
```

scoreSelection

Syntax

```
-- Lingo syntax
_movie.scoreSelection

// JavaScript syntax
_movie.scoreSelection;
```

Beschreibung

Diese Filmeigenschaft bestimmt, welche Kanäle im Drehbuchfenster ausgewählt werden. Lesen/Schreiben.

Die Informationen sind als lineare Liste aus linearen Listen formatiert. Jede aneinander angrenzende Auswahl besteht aus einer Liste, die die Anfangskanal-, Endkanal-, Anfangsbild- und Endbildnummer enthält. Sprite-Kanäle sind anhand der Kanalnummer anzugeben. Verwenden Sie zur Angabe der anderen Kanäle die folgenden Nummern:

Kanal:	Nummer:
Bildskriptkanal	0
Soundkanal 1	-1
Soundkanal 2	-2

Kanal:	Nummer:
Übergangskanal	-3
Palettenkanal	-4
Tempokanal	-5

Sie können auch nicht aneinander angrenzende Kanäle oder Bilder auswählen.

Beispiel

Die folgende Anweisung wählt die Sprite-Kanäle 15 bis 25 in den Bildern 100 bis 200 aus:

```
-- Lingo syntax
_movie.scoreSelection = [[15, 25, 100, 200]]

// JavaScript syntax
_movie.scoreSelection = list(list(15, 25, 100, 200));
```

Die folgende Anweisung wählt die Sprite-Kanäle 15 bis 25 und 40 bis 50 in den Bildern 100 bis 200 aus:

```
-- Lingo syntax
_movie.scoreSelection = [[15, 25, 100, 200], [40, 50, 100, 200]]

// JavaScript syntax
_movie.scoreSelection = list(list(15, 25, 100, 200), list(40, 50, 100, 200));
```

Die folgende Anweisung wählt das Bildskript in den Bildern 100 bis 200 aus:

```
-- Lingo syntax
_movie.scoreSelection = [[0, 0, 100, 200]]

// JavaScript syntax
_movie.scoreSelection = list(list(0, 0, 100, 200));
```

Siehe auch

Movie

script

Syntax

```
-- Lingo syntax
_movie.script[scriptNameOrNum]

// JavaScript syntax
_movie.script[scriptNameOrNum];
```

Beschreibung

Diese Filmeigenschaft bietet indizierten oder benannten Zugriff auf die Skriptdarsteller eines Films. Schreibgeschützt

Das Argument *scriptNameOrNum* kann eine Zeichenfolge sein, die den Namen des Skriptdarstellers angibt, oder eine Ganzzahl, die die Nummer des Skriptdarstellers angibt.

- Ist *scriptNameOrNum* eine Zeichenfolge, bietet die Eigenschaft `script` Zugriff auf den Skript-Darsteller, unabhängig von der Besetzungsbibliothek, in der sich der Darsteller befindet.

- Ist *scriptNameOrNum* eine Ganzzahl, bietet die Eigenschaft `script` nur Zugriff auf den Skriptdarsteller, der in der ersten Besetzungsbibliothek des referenzierten Films gefunden wird. Auch mit indiziertem Zugriff kann keine andere Besetzungsbibliothek als die erste angegeben werden.

Beispiel

Die folgende Anweisung greift auf ein benanntes Skript zu.

```
-- Lingo syntax
bugScript = _movie.script["Warrior Ant"]

// JavaScript syntax
var bugScript = _movie.script["Warrior Ant"];
```

Siehe auch

Movie

scripted

Syntax

```
-- Lingo syntax
spriteChannelObjRef.scripted

// JavaScript syntax
spriteChannelObjRef.scripted;
```

Beschreibung

Diese Sprite-Kanal-Eigenschaft gibt an, ob ein Sprite-Kanal skriptgesteuert (`TRUE`) oder durch das Drehbuch gesteuert (`FALSE`) ist. Nur Lesen.

Beispiel

Die folgenden Anweisungen erstellen ein mit Skript erstelltes Sprite aus dem Darsteller „kite“ in Sprite-Kanal 5, wenn dieser Kanal nicht bereits skriptgesteuert ist.

```
-- Lingo syntax
if (channel(5).scripted = FALSE) then
    channel(5).makeScriptedSprite(member("kite"))
end if

// JavaScript syntax
if (channel(5).scripted == false) {
    channel(5).makeScriptedSprite(member("kite"));
}
```

Siehe auch

Sprite Channel

scriptingXtraList

Syntax

```
-- Lingo syntax
_player.scriptingXtraList

// JavaScript syntax
_player.scriptingXtraList;
```

Beschreibung

Diese Player-Eigenschaft gibt eine lineare Liste aller im Director-Player verfügbaren Xtra-Skripterweiterungen zurück. Nur Lesen.

Bei den Xtra-Erweiterungen handelt es sich um die im Ordner „Configuration\Xtras“ vorhandenen.

Beispiel

Die folgende Anweisung zeigt alle verfügbaren Xtra-Skripterweiterungen im Nachrichtenfenster an:

```
-- Lingo syntax
trace(_player.scriptingXtraList)

// JavaScript syntax
trace(_player.scriptingXtraList);
```

Siehe auch

`mediaXtraList`, `Player`, `Skriptobjekte`, `toolXtraList`, `transitionXtraList`, `xtraList (Player)`

scriptInstanceList

Syntax

```
sprite(whichSprite).scriptInstanceList
the scriptInstanceList of sprite whichSprite
```

Beschreibung

Diese Sprite-Eigenschaft erstellt eine Liste der Skriptbezüge, die an einem Sprite angebracht sind. Diese Eigenschaft ist nur zur Laufzeit verfügbar. Die Liste ist leer, wenn der Film nicht ausgeführt wird. In der Liste vorgenommene Änderungen werden nicht im Drehbuch gespeichert. Diese Eigenschaft bietet sich für die folgenden Aufgaben an:

- Anbringen eines Verhaltens an einem Sprite, das dann zur Laufzeit verwendet wird
- Ermitteln, ob Verhalten an einem Sprite angebracht sind und um welche Verhalten es sich handelt
- Ermitteln eines Verhaltensskriptbezugs, der mit dem Befehl `sendSprite` verwendet werden kann

Diese Eigenschaft kann getestet und eingestellt werden. (Diese Eigenschaft kann nur dann eingestellt werden, wenn das Sprite bereits vorhanden und zumindest eine Verhaltensinstanz am Sprite angebracht ist.)

Beispiel

Die folgende Prozedur zeigt die Liste der an einem Sprite angebrachten Skriptbezüge an:

```
-- Lingo
on showScriptRefs spriteNum
    put sprite(spriteNum).scriptInstanceList
end

// Javascript
function showScriptRefs(spriteNum)
{
    trace(sprite(spriteNum).scriptInstanceList);
}
```

Die folgenden Anweisungen bringen das Skript „Großer Lärm“ an Sprite 5 an:

```
x = script("Big Noise").new()
sprite(5).scriptInstanceList.add(x)
```

Siehe auch

scriptNum, sendSprite()

scriptList

Syntax

```
sprite(whichSprite).scriptList
the scriptList of sprite whichSprite
```

Beschreibung

Diese Sprite-Eigenschaft gibt eine Liste mit den am Sprite angebrachten Verhalten und ihren Eigenschaften zurück. Diese Eigenschaft kann nur über `setScriptList()` eingestellt werden. Sie kann bei der Drehbuchaufzeichnung nicht eingestellt werden.

Beispiel

Die folgende Anweisung zeigt eine Liste der an Sprite 1 angebrachten Skripts im Nachrichtenfenster an:

```
put sprite(1).scriptList
-- [[(member 2 of castLib 1), "[#myRotateAngle: 10.0000, #myClockwise: 1, #myInitialAngle:
0.0000]"], [(member 3 of castLib 1), "[#myAnglePerFrame: 10.0000, #myTurnFrames: 10,
#myHShiftPerFrame: 10, #myShiftFrames: 10, #myTotalFrames: 60, #mySurfaceHeight: 0]"]]
```

Siehe auch

setScriptList(), value()

scriptNum

Syntax

```
sprite(whichSprite). scriptNum
scriptNum of sprite whichSprite
```

Beschreibung

Diese Sprite-Eigenschaft kennzeichnet die Nummer des Skripts, das an dem in *whichSprite* angegebenen Sprite angebracht ist. Falls an dem Sprite mehrere Skripts angebracht sind, gibt die Sprite-Eigenschaft `scriptNum` die Nummer des ersten Skripts zurück. (Wenn Sie eine komplette Liste der an einem Sprite angebrachten Skripts einsehen möchten, brauchen Sie sich nur die Verhalten anzusehen, die für das betreffende Sprite im Verhaltensinspektor aufgelistet sind.)

Diese Eigenschaft kann während der Aufzeichnung des Drehbuchs sowohl getestet als auch eingestellt werden.

Beispiel

Die folgende Anweisung zeigt die Nummer des Skripts an, das an Sprite 4 angebracht ist:

```
-- Lingo
put sprite(4).scriptNum

// Javascript
trace(sprite(4).scriptNum);
```

Siehe auch

`scriptInstanceList`

scriptsEnabled

Syntax

```
-- Lingo syntax
memberObjRef.scriptsEnabled

// JavaScript syntax
memberObjRef.scriptsEnabled;
```

Beschreibung

Diese Director-Filmdarstellereigenschaft bestimmt, ob Skripts in einem verknüpften Film aktiviert (`TRUE` oder 1) oder deaktiviert (`FALSE` oder 0) werden.

Diese Eigenschaft ist nur für verknüpfte Director-Filmdarsteller verfügbar.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Die folgende Anweisung deaktiviert die Skripts im verknüpften Film „Jazz-Chronik“:

```
-- Lingo syntax
member("Jazz Chronicle").scriptsEnabled = FALSE

// JavaScript syntax
member("Jazz Chronicle").scriptsEnabled = 0;
```

scriptSyntax

Syntax

```
member whichScript.scriptSyntax
```

Beschreibung

Diese Darstellereigenschaft gibt ein Symbol zurück, dass die Skriptsyntax der Darsteller bezeichnet. Die möglichen Werte sind „`#lingo`“ oder „`#javaScript`“. Nur Lesen.

Beispiel

Die folgende Anweisung zeigt den Syntaxtyp des Hauptdarstellerskripts an.

```
-- Lingo
put member("Main Script").scriptSyntax
// JavaScript
put(member("Main Script").scriptSyntax);
```

scriptText

Syntax

```
-- Lingo syntax
memberObjRef.scriptText

// JavaScript syntax
memberObjRef.scriptText;
```

Beschreibung

Diese Darstellereigenschaft zeigt gegebenenfalls den Inhalt des Skripts an, das einem Darsteller zugeordnet ist.

Lesen/Schreiben.

Der Text eines Skripts wird entfernt, sobald der Film in einen Projektor konvertiert, geschützt oder für Shockwave Player komprimiert wird. Solche Filme verlieren dann ihre Werte für die Eigenschaft `scriptText`. Die Werte der Eigenschaft `scriptText` können also nicht abgerufen werden, wenn der Film in einem Projektor abgespielt wird. Director kann jedoch neue Werte für die Eigenschaft `scriptText` im Projektor einstellen. Diese neu zugeordneten Skripts werden automatisch kompiliert und können daher sehr schnell ausgeführt werden.

Beispiel

Die folgende Anweisung macht den Inhalt des Felddarstellers 20 zum Skript des Darstellers 30:

```
-- Lingo syntax
member(20).text = member(30).scriptText

// JavaScript syntax
member(20).text = member(30).scriptText;
```

Siehe auch

Member

scriptType

Syntax

```
member whichScript.scriptType
the scriptType of member whichScript
```

Beschreibung

Diese Darstellereigenschaft kennzeichnet den Typ des angegebenen Skripts. Folgende Werte sind möglich: `#movie`, `#score` und `#parent`.

Beispiel

Die folgende Anweisung macht den Skriptdarsteller „Main Script“ zum Filmskript:

```
-- Lingo
member("Main Script").scriptType = #movie

// Javascript
member("Main Script").scriptType = Symbol("movie");
```

scrollTop

Syntax

```
-- Lingo syntax
memberObjRef.scrollTop

// JavaScript syntax
memberObjRef.scrollTop;
```

Beschreibung

Diese Darstellereigenschaft bestimmt den Abstand (in Pixel) zwischen dem oberen Rand eines Felddarstellers und dem oberen Rand des Felds, das derzeit im Rollfeld sichtbar ist. Wenn Sie den Wert der Darstellereigenschaft `scrollTop` ändern, während der Film abgespielt wird, ändert sich dadurch auch der Feldbereich, der im Rollfeld sichtbar ist.

Auf diese Weise können Sie benutzerdefinierte Rollverhalten für Text- und Felddarsteller erstellen.

Die folgende Lingo-Anweisung verschiebt beispielsweise den Felddarsteller „Nachspann“ je nach Wert der Variablen `sliderVal` innerhalb des Feldes nach oben oder unten:

```
global sliderVal

on prepareFrame
    newVal = sliderVal * 100
    member("Credits").scrolltop = newVal
end
```

Mit Hilfe der globalen Variablen `sliderVal` kann z. B. gemessen werden, wie weit der Benutzer einen Schieberegler zieht. Die Anweisung `set newVal = sliderVal * 100` multipliziert die Variable `sliderVal` zur Berechnung eines Werts, der größer ist als die Entfernung, die der Schieberegler gezogen wird. Falls `sliderVal` ein positiver Wert ist, bewegt sich der Text nach oben. Wenn `sliderVal` dagegen einen negativen Wert darstellt, bewegt sich der Text nach unten.

Beispiel

Die folgende Wiederholungsschleife bewirkt das Rollen des Feldes "Nachspann" durch kontinuierliche Erhöhung des Werts von `scrollTop`:

```
--Lingo syntax
on wa
    member("Credits").scrollTop = 1
    repeat with count = 1 to 150
        member("Credits").scrollTop = member("Credits").scrollTop + 1
        _movie.updateStage()
    end repeat
end

// JavaScript syntax
function wa() {
    member("Credits").scrollTop = 1;
    for (var count = 1; count <= 150; count++) {
        member("Credits").scrollTop = member("Credits").scrollTop + 1;
        _movie.updateStage();
    }
}
```

sds (Modifizierer)

Syntax

```
member(whichCastmember).model(whichModel).sds.whichProperty
```

Beschreibung

Dieser 3D-Modifizierer fügt geometrische Details zu Modellen hinzu und synthetisiert zusätzliche Details, um Kurven zu glätten, wenn sich das Modell der Kamera nähert. Nachdem Sie den Modifizierer `sds` mit `addModifier()` an einem Modell angebracht haben, können Sie die Eigenschaften des `sds`-Modifizierers festlegen.

Der Modifizierer `sds` wirkt sich direkt auf die Modellressource aus. Sie sollten die Modifizierer `sds` und `lod` nicht zusammen verwenden, da sie gegenteilige Funktionen ausüben: Der Modifizierer `sds` fügt geometrische Details hinzu, der Modifizierer `lod` entfernt sie. Vor Verwendung des Modifizierers `sds` sollten Sie wie folgt die Modifiziereigenschaft `lod.auto` auf `FALSE` und die Modifiziereigenschaft `lod.level` auf die gewünschte Auflösung setzen:

```
member("myMember").model("myModel").lod.auto = 0
member("myMember").model("myModel").lod.level = 100
member("myMember").model("myModel").addmodifier(#sds)
```

Der Modifizierer `sds` kann nicht bei Modellen verwendet werden, die bereits den Modifizierer `inker` oder `toon` benutzen.

Nachdem Sie den Modifizierer `sds` an einer Modellressourcen angebracht haben, können Sie die folgenden Eigenschaften ermitteln oder festlegen:

`enabled` gibt an, ob die Oberflächenunterteilungsfunktion aktiviert ist (`TRUE`) oder nicht (`FALSE`). Die Standardeinstellung dieser Eigenschaft lautet `TRUE`.

`depth` gibt an, wie viele Auflösungsstufen das Modell bei Verwendung des Modifizierers `sds` maximal anzeigen kann.

`error` gibt die Fehlertoleranzstufe der Oberflächenunterteilungsfunktion an. Diese Eigenschaft ist nur dann relevant, wenn `sds.subdivision` auf `#adaptive` gesetzt ist.

`subdivision` gibt die Betriebsart dieses Modifizierers an. Mögliche Werte:

- Bei Angabe von `#uniform` wird das Gitternetz einheitlich hochskaliert; jede Fläche wird gleich oft unterteilt.
- Bei Angabe von `#adaptive` werden zusätzliche Details nur dann hinzugefügt, wenn größere Ausrichtungsänderungen stattfinden, und selbst dann nur zu den gegenwärtig sichtbaren Bereichen des Gitternetzes.

Hinweis: *Weitere Informationen zu diesen Eigenschaften finden Sie in den einzelnen Beschreibungen.*

Beispiel

Die folgende Anweisung zeigt den Wert der Eigenschaft sds.depth für das Modell "Gelände" an:

```
-- Lingo
put member("3D").model("Terrain").sds.depth
-- 2
// Javascript
trace(member("3D").getPropRef("model",1).sds.depth
// 2
```

Siehe auch

`lod (Modifizierer)`, `toon (Modifizierer)`, `inker (Modifizierer)`, `depth (3D)`, `enabled (sds)`, `error`, `subdivision`, `addModifier`

searchCurrentFolder

Syntax

```
-- Lingo syntax
_player.searchCurrentFolder

// JavaScript syntax
_player.searchCurrentFolder;
```

Beschreibung

Diese Player-Eigenschaft bestimmt, ob Director bei der Suche nach Dateinamen den aktuellen Ordner durchsucht oder nicht. Lesen/Schreiben.

- Wenn die Eigenschaft`searchCurrentFolder` auf `TRUE` (1) gesetzt ist, durchsucht Director bei der Verarbeitung von Dateinamen den aktuellen Ordner.
- Wenn die Eigenschaft`searchCurrentFolder` dagegen auf `FALSE` (0) gesetzt ist, wird der aktuelle Ordner bei der Verarbeitung von Dateinamen nicht durchsucht.

Diese Eigenschaft ist standardmäßig auf `TRUE` gesetzt.

Hinweis: *`_player.searchCurrentFolder` steht nur in den Ausführungsmodi (`runModes`) „Author“ und „Projector“ zur Verfügung und ist im Modus „Plugin“ (`runMode`) nicht verfügbar.*

Beispiel

Die folgende Anweisung zeigt im Nachrichtenfenster den Status der Eigenschaft `searchCurrentFolder` an. Das Ergebnis ist 1, das numerische Gegenstück zu TRUE:

```
-- Lingo syntax
put(_player.searchCurrentFolder)

// JavaScript syntax
put(_player.searchCurrentFolder);
```

Siehe auch

`Player`

searchPathList

Syntax

```
-- Lingo syntax
_player.searchPathList

// JavaScript syntax
_player.searchPathList;
```

Beschreibung

Diese Player-Eigenschaft zeigt eine Liste von Pfaden an, die Director bei der Suche nach verknüpften Medien wie z. B. Digitalvideos, GIFs, Bitmaps und Sounddateien durchsucht. Lesen/Schreiben.

Jedes Element in der Liste von Pfaden ist ein vollqualifizierter Pfadname, wie er auf der aktuellen Plattform zur Laufzeit angezeigt wird.

Der Wert von `searchPathList` ist eine lineare Liste, die Sie genau wie jede andere Liste mit den Befehlen `add()`, `addAt()`, `append()`, `deleteAt()` und `setAt()` bearbeiten können. Der Standardwert ist eine leere Liste.

URLs sollten in den Suchpfaden nicht als Dateibezüge verwendet werden.

Das Hinzufügen einer großen Anzahl von Pfaden in `searchPaths` verlangsamt den Suchvorgang. Versuchen Sie daher, die Anzahl der Pfade in dieser Liste auf ein Minimum zu beschränken.

Hinweis: *Diese Eigenschaft gilt nach ihrer Einstellung für alle folgenden Filme. Da die aktuellen Filmelemente bereits geladen wurden, wirkt sich das Ändern der Einstellung nicht auf diese Elemente aus.*

Beispiel

Die folgende Anweisung zeigt die Pfade an, die Director bei der Verarbeitung von Dateinamen durchläuft:

```
-- Lingo syntax
trace(_player.searchPathList)

// JavaScript syntax
trace(_player.searchPathList);
```

In Windows werden `searchPaths` durch die folgende Anweisung zwei Ordner zugeordnet:

```
-- Lingo syntax
_player.searchPathList = ["C:\Director\Projects\", "D:\CDROM\Sources\"]

// JavaScript syntax
_player.searchPathList = list("C:\\Director\Projects\","D:\CDROM\Sources\");
```

Auf dem Mac werden `searchPaths` durch die folgende Anweisung zwei Ordner zugeordnet:

```
-- Lingo syntax
_player.searchPathList = ["Hard Drive:Director:Projects:", "CDROM:Sources:"]

// JavaScript syntax
_player.searchPathList = list("Hard Drive:Director:Projects:","CDROM:Sources:");
```

Siehe auch

`Player`, `searchCurrentFolder`

selectedButton

Syntax

```
-- Lingo syntax
dvdObjRef.selectedButton

// JavaScript syntax
dvdObjRef.selectedButton;
```

Beschreibung

Diese DVD-Eigenschaft gibt den Index der Schaltfläche zurück, die zurzeit den Fokus hat. Nur Lesen.

Beispiel

Die folgende Anweisung zeigt die Schaltfläche an, die im DVD-Objekt aktiviert ist.

```
-- Lingo
put sprite(1).selectedButton
--1

// Javascript
trace(sprite(1).selectedButton)
//1
```

Siehe auch

`DVD`

selectedText

Syntax

```
-- Lingo syntax
memberObjRef.selectedText

// JavaScript syntax
memberObjRef.selectedText;
```

Beschreibung

Diese Textdarstellereigenschaft gibt den aktuell ausgewählten Text-Chunk als einzelnen Objektbezug zurück. Damit wird der Zugriff auf Schriftmerkmale und Stringinformationen der einzelnen Zeichen ermöglicht.

Beispiel

Die folgende Prozedur zeigt den aktuell ausgewählten Text an, der in ein lokales Variablenobjekt gesetzt wird. Dieses Objekt wird dann verwendet, um auf verschiedene Textmerkmale Bezug zu nehmen, die im Nachrichtenfenster angezeigt werden.

```
--Lingo syntax
property spriteNum

on mouseUp(me)
    mySelectionObject = sprite(spriteNum).member.selectedText
    put(mySelectionObject.text)
    put(mySelectionObject.font)
    put(mySelectionObject.fontSize)
    put(mySelectionObject.fontStyle)
end

// JavaScript syntax
function mouseUp() {
    var mySelectionObject = sprite(this.spriteNum).member.selectedText;
    trace(mySelectionObject.text);
    trace(mySelectionObject.font);
    trace(mySelectionObject.fontSize);
    trace(mySelectionObject.fontStyle);
}
```

selection

Syntax

```
-- Lingo syntax
castObjRef.selection

// JavaScript syntax
castObjRef.selection;
```

Beschreibung

Diese Besetzungsbibliothekseigenschaft gibt die Darsteller zurück, die in einem angegebenen Besetzungsfenster ausgewählt sind. Lesen/Schreiben.

Beispiel

In dieser Anweisung sind Darsteller 1 bis 10 in `castLib` 1 ausgewählt:

```
-- Lingo syntax
castLib(1).selection = [[1, 10]]

// JavaScript syntax
castLib(1).selection = list( list(1, 10) );
```

In dieser Anweisung sind Darsteller 1 bis 10 und 30 bis 40 in `castLib` 1 ausgewählt:

```
-- Lingo syntax
castLib(1).selection = [[1, 10], [30, 40]]

// JavaScript syntax
castLib(1).selection = list( list(1, 10), list(30, 40) );
```

Siehe auch

`Cast Library`

selection (Textdarstellereigenschaft)

Syntax

```
member(whichTextMember).selection
```

Beschreibung

Diese Textdarstellereigenschaft gibt eine Liste des ersten und letzten ausgewählten Zeichens in einem Textdarsteller zurück.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Die folgende Anweisung stellt die vom Sprite des Textdarstellers `myAnswer` angezeigte Auswahl so ein, dass die Zeichen 6 bis 10 markiert sind:

```
member("myAnswer").selection = [6, 10]
```

Siehe auch

`color()`, `selStart`, `selEnd`

selEnd

Syntax

```
-- Lingo syntax
_movie.selEnd

// JavaScript syntax
_movie.selEnd;
```

Beschreibung

Diese Filmeigenschaft gibt das letzte Zeichen einer Auswahl an. Sie wird mit `selStart` verwendet, um eine Auswahl im aktuellen bearbeitbaren Feld zu identifizieren, wobei vom Anfangszeichen an gezählt wird. Der Höchstwert für `selEnd` ist die Anzahl der Zeichen im gegenwärtig bearbeitbaren Feld. Diese Eigenschaft gilt für Felddarsteller-, jedoch nicht für Textdarsteller-Sprites. Wenn Sie einen höheren Wert für `selStart` als für `selEnd` zuweisen, wird der Wert von `selEnd` auf den für `selStart` zurückgesetzt.

Diese Eigenschaft kann getestet und eingestellt werden. Der Standardwert ist 0.

Beispiel

Die folgenden Anweisungen wählen "cde" im Feld "abcdefg" aus:

```
-- Lingo syntax
_movie.selStart = 2
_movie.selEnd = 5
// JavaScript syntax
_movie.selStart = 2;
_movie.selEnd = 5;
```

Die folgende Anweisung macht eine Auswahl 20 Zeichen lang:

```
-- Lingo syntax
_movie.selEnd = _movie.selStart + 20
// JavaScript syntax
_movie.selEnd = _movie.selStart + 20;
```

Siehe auch

editable, hilite (Befehl), selection() (Funktion), selStart, text

selStart

Syntax

```
-- Lingo syntax
_movie.selStart
// JavaScript syntax
_movie.selStart;
```

Beschreibung

Diese Filmeigenschaft gibt die vorherige Position eines Anfangszeichens in einer Auswahl an. Sie wird mit `selEnd` verwendet, um eine Auswahl in dem aktuellen bearbeitbaren Feld zu identifizieren. Der Wert 0 gibt eine Position vor dem ersten Zeichen an. Der Höchstwert für `selStart` ist die Anzahl der Zeichen im gegenwärtig bearbeitbaren Feld. Diese Eigenschaft gilt für Felddarsteller-, jedoch nicht für Textdarsteller-Sprites. Wenn Sie einen höheren Wert für `selStart`als für `selEnd` zuweisen, wird der Wert von `selEnd` auf den für `selStart` zurückgesetzt.

Diese Eigenschaft kann getestet und eingestellt werden. Der Standardwert ist 0.

Beispiel

Die folgenden Anweisungen wählen "cde" im Feld "abcdefg" aus:

```
-- Lingo syntax
_movie.selStart = 2
_movie.selEnd = 5

// JavaScript syntax
_movie.selStart = 2;
_movie.selEnd = 5;
```

Die folgende Anweisung macht eine Auswahl 20 Zeichen lang:

```
-- Lingo syntax
_movie.selEnd = _movie.selStart + 20

// JavaScript syntax
_movie.selEnd = _movie.selStart + 20;
```

Siehe auch

selection() (Funktion), selEnd, text

serialNumber

Syntax

```
-- Lingo syntax
_player.serialNumber
// JavaScript syntax
_player.serialNumber;
```

Beschreibung

Diese Filmeigenschaft ist ein String, der die während der Installation von Director eingegebene Seriennummer enthält. Schreibgeschützt

Diese Eigenschaft ist nur in der Authoring-Umgebung verfügbar. Sie kann in einem MIAW (Film in einem Fenster)-Werkzeug verwendet werden, dass die persönlichen Angaben eines Benutzers anzeigt.

Beispiel

Die folgende Prozedur würde sich in diesem Falle im Filmskript eines MIAW befinden. Sie stellt den Benutzernamen und die Seriennummer in ein Anzeigefeld, wenn das Fenster geöffnet wird.

```
-- Lingo syntax
on prepareMovie
displayString = _player.userName & RETURN & _player.organizationName & RETURN &
_player.serialNumber
member("User Info").text = displayString
end

// JavaScript syntax
function prepareMovie() {
var displayString = _player.userName + "\n" + _player.organizationName + "\n" +
_player.serialNumber;
member("User Info").text = displayString;
}
```

Siehe auch

Player

shader

Syntax

```
member(whichCastmember).shader(whichShader)
member(whichCastmember).shader[index]
member(whichCastmember).model(whichModel).shader
member(whichCastmember).modelResource(whichModelResource).face[index].shader
```

Beschreibung

3D-Element, -Modelleigenschaft und -Flächeneigenschaft; Objekt, das das Aussehen der Modelloberfläche definiert. Der Shader ist die "Haut", die um die vom Modell verwendete Modellressource gewickelt wird.

Der Shader selbst ist keine Grafik. Die sichtbare Komponente eines Shaders besteht aus bis zu acht Texturebenen, `texture`. Diese acht Texturebenen werden wiederum in Director aus Bitmapdarstellern oder Grafikobjekten erstellt oder zusammen mit Modellen aus 3D-Modellierungsprogrammen importiert. Weitere Informationen hierzu finden Sie unter `texture`.

Jedes Modell besitzt eine lineare Shader-Liste, die so genannte `shaderList`. Die Anzahl von Einträgen in dieser Liste entspricht der Anzahl von Gitternetzen in der vom Modell verwendeten Modellressource. Jedes Gitternetz kann nur von einem Shader schattiert werden.

Der 3D-Darsteller besitzt einen Standard-Shader namens `DefaultShader`, der nicht gelöscht werden kann. Dieser Shader wird verwendet, wenn einem Modell kein Shader zugeordnet ist und wenn ein von einem Modell verwendeter Shader gelöscht wird.

Anhand der Syntax `member(`*whichCastmember*`).model(`*whichModel*`).shader` können Sie auf den ersten Shader in der `shaderList` des Modells zugreifen. Dieses Format entspricht `member(`*whichCastmember*`).model(`*whichModel*`).shaderList[1]`.

Shader werden mit den Befehlen `newShader()` und `deleteShader()` erstellt und gelöscht.

Shader werden in der Shader-Palette des 3D-Darstellers gespeichert. Auf sie kann nach Namen (*whichShader*) oder Palettenindex (*shaderIndex*) Bezug genommen werden. Ein Shader kann von beliebig vielen Modellen benutzt werden. Shader-Änderungen erscheinen in allen Modellen, die den jeweiligen Shader verwenden.

Es gibt vier Arten von Shadern:

Bei Shadern vom Typ `#standard` werden Texturen realitätsgetreu angezeigt.

Bei Shadern vom Typ `#painter`, `#engraver` und `#newsprint` erscheinen die Texturen stilisiert, damit Mal-, Gravur- und Zeitungseffekte erzielt werden können. Diese Shader-Typen weisen neben den Eigenschaften des Shaders `#standard` noch eine Reihe spezieller Eigenschaften auf.

Weitere Informationen zu Shader-Eigenschaften finden Sie in den Themen des Director-Benutzerhandbuchs im Director-Hilfefenster.

Die von den Einzelseiten von `#mesh`-Primitiven verwendete Shader können mit der Syntax `member(`*whichCastmember*`).modelResource(`*whichModelResource*`).face[`*index*`].shader` festgelegt werden. Wenn sich diese Eigenschaft ändert, muss der Befehl `build()` aufgerufen werden.

Beispiel

Die folgende Anweisung setzt die Eigenschaft shader des Modells `WallSurface` auf den Shader "Wandoberfläche":

```
member("Room").model("Wall").shader = member("Room").shader("WallSurface")
```

Siehe auch

`shaderList`, `newShader`, `deleteShader`, `face`, `texture`

shaderList

Syntax

```
member(whichCastmember).model(whichModel).shaderList
member(whichCastmember).model(whichModel).shaderList[index]
```

Beschreibung

Diese 3D-Modelleigenschaft ist eine lineare Liste der auf das Modell angewendeten `ShadowPercentage`. Die Anzahl von Einträgen in dieser Liste entspricht der Anzahl von Gitternetzen in der vom Modell verwendeten Modellressource. Jedes Gitternetz kann nur von einem Shader schattiert werden.

Der Shader wird mit folgender Syntax an der angegebenen Positon *index* in die Shaderliste `shaderlist`:
`member(`*whichCastmember*`).model(`*whichModel*`).shaderList[`*index*`] = shaderReference`

Bei 3D-Text ist jedes Zeichen ein eigenes Gitternetz. Setzen Sie den Wert von `index` auf die Nummer des Zeichens, dessen Shader Sie festlegen möchten.

Mit der folgenden Syntax können Sie alle Indexpositionen *index* in der `shaderList` auf denselben Shader setzen (beachten Sie, dass es für die `shaderList` keinen Index gibt:
`member(`*whichCastmember*`).model(`*whichModel*`).shaderList = shaderReference`.

Eigenschaften eines Shaders lassen sich in der Shaderliste `shaderlist` mit folgender Syntax einstellen:
`member(`*whichCastmember*`).model(`*whichModel*`).shaderList[`*index*`].`*whichProperty* `=` *propValue*

Mit der folgenden Syntax können Sie eine Eigenschaft aller Shader eines Modells auf denselben Wert setzen (beachten Sie, dass es für die Shader-Liste `shaderList` keinen Index gibt):

```
member(whichCastmember).model(whichModel).shaderList.whichProperty = propValue
```

Beispiel

Die folgende Anweisung setzt den zweiten Shader in der `shaderList` des Modells "Stossstange" auf den Shader "Chrom":

```
member("Car").model("Bumper").shaderList[2] = member("Car").shader("Chrome")
```

Die folgende Anweisung setzt alle Shader in der `shaderList` des Modells „Stossstange“ auf den Shader „Chrom“:

```
member("Car").model("Bumper").shaderList = member("Car").shader("Chrome")
```

Siehe auch

`shadowPercentageshaderdeleteShadernewShader`

shadowPercentage

Syntax

```
member(whichCastmember).model(whichModel).toon.shadowPercentage
member(whichCastmember).model(whichModel).shader.shadowPercentage
member(whichCastmember).shader(whichShader).shadowPercentage
```

Beschreibung

Diese 3D-Eigenschaft für den Modifizierer `toon` und den Shader `painter` gibt an, welcher Prozentsatz der verfügbaren Farben in dem Bereich der Modelloberfläche verwendet wird, in dem das Licht keine Highlights erzeugt.

Der Gültigkeitsbereich dieser Eigenschaft liegt zwischen 50 und 100; die Standardeinstellung ist 0.

Die vom Modifizierer `toon` und dem Shader `painter` für ein Modell verwendete Anzahl von Farben richtet sich nach der Eigenschaft `colorSteps` des Modifizierers `toon` bzw. des Shaders `painter` des Modells.

Beispiel

Die folgende Anweisung setzt die Eigenschaft `shadowPercentage` des Modifizierers toon für das Modell „Teekanne“ auf 50. Für den Schattenbereich der Modelloberfläche werden 50% der Farben verwendet, die dem Modifizierer toon für dieses Modell zur Verfügung stehen.

```
-- Lingo
member("shapes").model("Teapot").toon.shadowPercentage = 50

// Javascript
member("shapes").getPropRef("model",1).getProp("toon").shadowPercentage=50;
```

Siehe auch

`colorSteps`, `shadowStrength`

shadowStrength

Syntax

```
member(whichCastmember).model(whichModel).toon.shadowStrength
member(whichCastmember).model(whichModel).shader.shadowStrength
member(whichCastmember).shader(whichShader).shadowStrength
```

Beschreibung

Diese 3D-Eigenschaft für den Modifizierer `toon` und den Shader `#painter` gibt die Helligkeit des Bereichs der Modelloberfläche an, in dem das Licht keine Highlights erzeugt.

Der Standardwert dieser Eigenschaft lautet 1.0.

Beispiel

Die folgende Anweisung stellt die Eigenschaft `shadowStrength` eines „toon“-Modifizierers für das Modell „Sphere“ auf 0,1. Die Fläche des Modells, die nicht bestrahlt wird, ist dann sehr dunkel.

```
member("Shapes").model("Sphere").toon.shadowStrength = 0.1
```

shapeType

Syntax

```
member(whichCastMember).shapeType
the shapeType of member whichCastMember
```

Beschreibung

Diese Formdarstellereigenschaft gibt den Typ der angegebenen Form an. Mögliche Typen sind `#rect`, `#roundRect`, `#oval` und `#line`. Anhand dieser Eigenschaft können Sie einen Formdarstellertyp spezifizieren, nachdem Sie mithilfe von Lingo den Formdarsteller erstellt haben.

Beispiel

Diese Anweisungen erstellen einen Formdarsteller mit der Nummer 100 und definieren ihn dann als Oval:

```
-- Lingo
new(#shape, member 100)
member(100).shapeType = #oval

// Javascript
var t = _movie.newMember(symbol("shape"));
t.shapeType=symbol("oval");
```

shiftDown

Syntax

```
-- Lingo syntax
_key.shiftDown

// JavaScript syntax
_key.shiftDown;
```

Beschreibung

Diese Tasteneigenschaft gibt an, ob der Benutzer die Umschalttaste drückt. Nur Lesen.

Wird die Umschalttaste gedrückt, gibt die Eigenschaft `TRUE` zurück, andernfalls `FALSE`.

Diese Eigenschaft muss in Zusammenhang mit einer anderen Taste getestet werden.

Beispiel

Die folgende Anweisung prüft, ob die Umschalttaste gedrückt wird, und ruft die Prozedur `doCapitalA` auf, wenn dies der Fall ist:

```
-- Lingo syntax
if (_key.shiftDown) then
    doCapitalA(_key.key)
end if

// JavaScript syntax
if (_key.shiftDown) {
    doCapitalA(_key.key);
}
```

Siehe auch

controlDown, Taste, key, keyCode, optionDown

shininess

Syntax

```
member(whichCastmember).shader(whichShader).shininess
member(whichCastmember).model(whichModel).shader.shininess
member(whichCastmember).model(whichModel).shaderList[shaderListIndex].shininess
```

Beschreibung

Mit dieser 3D-Eigenschaft für Standard-Shader können Sie ermitteln oder festlegen, wie stark eine Oberfläche glänzt. Dies richtet sich nach dem Anteil der Shader-Oberfläche, der für die hellsten Stellen (Highlights) verwendet wird. Der Wert ist eine Ganzzahl zwischen 0 und 100. Der Standardwert lautet 30.

Alle Shader können auf die Eigenschaften des `#standard`-Shaders zugreifen. Darüber hinaus haben die Shader `#engraver`, `#newsprint` und `#painter` noch weitere Eigenschaften, die speziell für den jeweiligen Shader-Typ gelten. Weitere Informationen finden Sie unter newShader.

Beispiel

Die folgende Anweisung setzt die Eigenschaft „shininess“ des ersten Shaders in der Shader-Liste des Modells `gbCyl3` auf 60, d. h. 60 % der Oberfläche des Shaders sind für Highlights vorgesehen:

```
-- Lingo
member("Scene").model("gbCyl3").shader.shininess = 60

// Javascript
member("Scene").getPropRef("model",1).getProp("shader").shininess=60;
```

silhouettes

Syntax

```
member(whichCastmember).model(whichModel).inker.silhouettes
member(whichCastmember).model(whichModel).toon.silhouettes
```

Beschreibung

Diese 3D-Eigenschaft für die Modifizierer `toon` und `inker` gibt an, ob vom Modifizierer an den sichtbaren Kanten des Modells Linien gezeichnet werden `(TRUE)` oder nicht `(FALSE)`.

Silhouettenlinien werden auf der Projektionsebene der Kamera um das 2D-Bild des Modells gezeichnet. Ihre Beziehung zum Gitternetz des Modells ist nicht fest – im Gegensatz zu Falten- oder Begrenzungslinien, die auf den Merkmalen des Gitternetzes gezeichnet werden.

Silhouettenlinien sind ähnlich wie die Konturlinien, die oft in Malbüchern für Kinder zu finden sind.

Der Standardwert dieser Eigenschaft lautet `TRUE`.

Beispiel

Die folgende Anweisung setzt die Eigenschaft silhouettes des auf das Modell "Kugel" angewendeten Modifizierers inker auf FALSE. Um das Profil des Modells werden keine Linien gezeichnet.

```
-- Lingo
member("Shapes").model("Sphere").inker.silhouettes = FALSE

// Javascript
member("Shapes").getPropRef("model",1).inker.silhouetess = false;
```

size

Syntax

```
-- Lingo syntax
memberObjRef.size

// JavaScript syntax
memberObjRef.size;
```

Beschreibung

Diese Darstellereigenschaft gibt den Speicherplatz in Byte zurück, den ein bestimmter Darsteller belegt. Nur Lesen.

Sie können Byte in Kilobyte konvertieren, indem Sie die Zahl durch 1024 dividieren.

Beispiel

Die folgende Zeile gibt die Größe von Darsteller „Schrein“ im Feld „Wie groß“ an:

```
-- Lingo syntax
member("How Big").text = string(member("shrine").size)

// JavaScript syntax
member("How Big").text = member("shrine").size.toString();
```

Siehe auch

Member

sizeRange

Syntax

```
member(whichCastmember).modelResource
(whichModelResource).sizeRange.start
modelResourceObjectReference.sizeRange.start
member(whichCastmember).modelResource
(whichModelResource).sizeRange.end
modelResourceObjectReference.sizeRange.end
```

Beschreibung

Mit dieser 3D-Eigenschaft können Sie bei einer Modellressource vom Typ `#particle` die Eigenschaften `start` und `end` des Größenbereichs (`sizeRange`) der Modellressource ermitteln und festlegen. Partikel werden in Welteinheiten gemessen.

Die Größe der Partikel im System wird über die Lebensdauer jedes Partikels hinweg linear zwischen `sizeRange.start` und `sizeRange.end` interpoliert.

Diese Eigenschaft muss eine Ganzzahl größer als 0 sein. Der Standardwert ist 1.

Beispiel

In diesem Beispiel ist `mrFount` eine Modellressource vom Typ `#particle`. Die folgende Anweisung legt die `sizeRange`-Eigenschaften von `mrFount` fest. Die erste Zeile setzt den Startwert auf 4, die zweite Zeile den Endwert auf 1. Diese Anweisung bewirkt, dass die Partikel der Ressource `mrFount` anfänglich die Größe 4 aufweisen und im Laufe ihrer Lebenszeit auf die Größe 1 abnehmen.

```
-- Lingo
member("fountain").modelResource("mrFount").sizeRange.start = 4
member("fountain").modelResource("mrFount").sizeRange.end = 1

// Javascript
member("fountain").getPropRef("modelResource",1).getProp("sizeRange").start=4;
member("fountain").getPropRef("modelResource",1).getProp("sizeRange").end=4;
```

sizeState

Syntax

```
-- Lingo syntax
windowObjRef.sizeState

// JavaScript syntax
windowObjRef.sizeState;
```

Beschreibung

Diese Fenstereigenschaft gibt den Größenzustand eines Fensters zurück. Nur Lesen.

Der zurückgegebene Größenzustand kann einer der folgenden Werte sein:

Größenzustand	Beschreibung
`#minimized`	Gibt an, dass das Fenster zurzeit minimiert ist.
`#maximized`	Gibt an, dass das Fenster zurzeit maximiert ist.
`#normal`	Gibt an, dass das Fenster zurzeit weder minimiert noch maximiert ist.

Beispiel

Die folgenden Anweisungen maximieren das Fenster namens „Artists“, wenn es nicht bereits maximiert ist.

```
-- Lingo syntax
if (window("Artists").sizeState <> #maximized) then
    window("Artists").maximize()
end if

// JavaScript syntax
if (window("Artists").sizeState != symbol("maximized")) {
    window("Artists").maximize();
}
```

Siehe auch

Window

skew

Syntax

```
-- Lingo syntax
spriteObjRef.skew

// JavaScript syntax
spriteObjRef.skew;
```

Beschreibung

Diese Sprite-Eigenschaft gibt den Winkel der Schrägstellung (Neigung) der vertikalen Ränder eines Sprites im Vergleich zur Vertikalen als Fließkommazahl in Hundertstelgrad an. Lesen/Schreiben.

Negative Werte geben eine Neigung nach links an, während positive Werte eine Neigung nach rechts angeben. Alle Werte über 90° kippen die Grafik senkrecht.

Das Drehbuch kann Informationen zum Neigen einer Grafik von +21.474.836,47° bis -21.474.836,48° speichern, wodurch 59.652 volle Drehungen in beide Richtungen möglich sind.

Sobald die Neigungsgrenze erreicht ist (d. h. gleich nach der 59.652sten Neigung), wird die Neigung auf +116,47° oder -116,48° – nicht 0,00° – zurückgesetzt. Der Grund dafür ist, dass +21,474,836.47° dem Wert +116.47° entsprechen und -21,474,836.48° dem Wert -116.48° (oder +243.12°). Um diese Zurücksetzung beim Ausführen von fortlaufenden Neigungsbefehlen durch Skriptcode zu umgehen, beschränken Sie die Winkeleinstellung auf ±360° für beide Richtungen.

Beispiel

Das folgende Verhalten bewirkt, dass ein Sprite jedes Mal, wenn der Abspielkopf in ein neues Bild eintritt, fortlaufend um 2° geneigt wird, der Gesamtwinkel jedoch auf 360° beschränkt ist:

```
-- Lingo syntax
property spriteNum

on prepareFrame me
    sprite(spriteNum).skew = integer(sprite(spriteNum).skew + 2) mod 360
end

// JavaScript syntax
function prepareFrame() {
    sprite(this.spriteNum).skew = parseInt(sprite(this.spriteNum).skew + 2) % 360;
}
```

Siehe auch

flipH, flipV, rotation, Sprite

smoothness

Syntax

```
member(whichTextmember).smoothness
member(whichCastMember).modelResource(whichExtruderModelResource).smoothness
```

Beschreibung

Mit dieser 3D-Eigenschaft für Extrudermodellressourcen und Text können Sie die Anzahl von Segmenten zur Erstellung eines 3D-Textdarstellers ermitteln und festlegen (Ganzzahl). Je höher die Zahl, desto glätter erscheint der Text. Der Gültigkeitsbereich dieser Eigenschaft liegt zwischen 1 und 10; die Standardeinstellung ist 5.

Weitere Informationen zum Arbeiten mit Extrudermodellressourcen und Textdarstellern finden Sie unter dem Eintrag extrude3D.

Beispiel

In diesem Beispiel ist der Darsteller "Logo" ein Textdarsteller. Die folgende Anweisung setzt die Eigenschaft „smoothness“ von "Logo“ auf 8. Wenn „Logo“ im 3D-Modus angezeigt wird, sind die Kanten der Buchstaben sehr glatt.

```
-- Lingo
member("Logo").smoothness = 8

// Javascript
member("Logo").smoothness = 8;
```

In diesem Beispiel ist die Modellressource des Modells „Slogan“ extrudierter Text. Die folgende Anweisung setzt die Eigenschaft „smoothness“ der Modellressource von "Slogan" auf 1, sodass die Buchstaben von "Slogan" sehr kantig erscheinen:

```
-- Lingo
member("Scene").model("Slogan").resource.smoothness = 1

// Javascript
member("Scene").getPropRef("model",1).getProp("resource").smoothness=1;
```

Siehe auch

extrude3D

sound (Darsteller)

Syntax

```
-- Lingo syntax
memberObjRef.sound

// JavaScript syntax
memberObjRef.sound;
```

Beschreibung

Diese Darstellereigenschaft steuert, ob der Sound eines Films, eines Digitalvideos oder eines Flash-Films aktiviert („TRUE“, Standard) oder deaktiviert („FALSE“) ist. Lesen/Schreiben.

Bei Flash-Darstellern tritt diese neue Einstellung in Kraft, nachdem der aktuelle Sound fertig abgespielt ist.

Ein Beispiel für `sound` in einem fertigen Film finden Sie im Film „Sound Control“ im Ordner „Learning/Lingo Examples“ im Director-Anwendungsordner.

Beispiel

Die folgende Prozedur akzeptiert einen Darstellerbezug und schaltet die Darstellereigenschaft sound ein und aus:

```
-- Lingo syntax
on ToggleSound(whichMember)
    member(whichMember).sound = not(member(whichMember).sound)
end

// JavaScript syntax
function ToggleSound(whichMember) {
    member(whichMember).sound = !(member(whichMember).sound);
}
```

Siehe auch

Flash Movie

sound (Player)

Syntax

```
-- Lingo syntax
_player.sound[intSoundChannelNum]

// JavaScript syntax
_player.sound[intSoundChannelNum];
```

Beschreibung

Diese Player-Eigenschaft bietet indizierten Zugriff mithilfe einer Player-Eigenschaft auf ein Sound Channel-Objekt. Nur Lesen.

Das Argument *intSoundChannelNum* ist eine Ganzzahl, die die Nummer des Soundkanals angibt, auf den zugegriffen werden soll.

Die Funktionalität dieser Eigenschaft ist mit der der Top-Level-Methode `sound()` identisch.

Beispiel

Die folgende Anweisung setzt die Variable `mySound` auf den Sound in Soundkanal 3:

```
-- Lingo syntax
mySound = _player.sound[3]

// JavaScript syntax
var mySound = _player.sound[3];
```

Siehe auch

`Player`, `sound()`, `Sound Channel`

soundChannel (SWA)

Syntax

```
-- Lingo syntax
memberObjRef.soundChannel

// JavaScript syntax
memberObjRef.soundChannel;
```

Beschreibung

Diese Shockwave Audio (SWA)-Darstellereigenschaft gibt den Soundkanal an, in dem der SWA-Sound abgespielt wird.

Wird keine Kanalnummer bzw. Kanal 0 angegeben, ordnet der SWA-Streaming-Darsteller den Sound dem freien Soundkanal mit der höchsten Nummer zu.

Gestreamte Shockwave Audio-Sounds können als Sprites in Sprite-Kanälen erscheinen, werden aber als Sounds in einem Soundkanal abgespielt. Nehmen Sie auf SWA-Sound-Sprites über die Sprite-Kanalnummer – nicht über die Soundkanalnummer – Bezug.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Die folgende Anweisung bewirkt, dass der SWA-Streaming-Darsteller „Frank Zappa“ in Soundkanal 3 abgespielt wird:

```
-- Lingo syntax
member("Frank Zappa").soundChannel = 3

// JavaScript syntax
member("Frank Zappa").soundChannel = 3;
```

soundChannel (RealMedia)

Syntax

```
-- Lingo syntax
memberOrSpriteObjRef.soundChannel

// JavaScript syntax
memberOrSpriteObjRef.soundChannel;
```

Beschreibung

Mit dieser Eigenschaft für `RealMedia`-Sprites und -Darsteller können Sie den Soundkanal abrufen und einstellen, der zur Wiedergabe der Audiodaten im `RealMedia`-Stream verwendet wird. Diese Eigenschaft ermöglicht es Ihnen, den Audioabschnitt eines `RealMedia`-Streams mit den in Lingo verfügbaren Soundmethoden und -eigenschaften zu steuern. Wenn Sie diese Eigenschaft auf einen Wert außerhalb des Bereichs 0–8 setzen, tritt ein Lingo-Fehler auf. Diese Eigenschaft ist wirkungslos, wenn `realPlayerNativeAudio()` auf `TRUE` gesetzt ist.

Die Standardeinstellung dieser Eigenschaft ist 0, d. h. die `RealMedia`-Audiodaten werden im höchsten verfügbaren Soundkanal abgespielt. Der Wert dieser Eigenschaft ändert sich während der Wiedergabe, je nachdem, welcher Kanal verwendet wird. Wenn der `RealMedia`-Darsteller abgespielt wird, gibt diese Eigenschaft den gegenwärtig benutzten Soundkanal an. Wenn der `RealMedia`-Darsteller angehalten wird, lautet der Wert dieser Eigenschaft wieder 0.

Wenn Sie einen Kanal (1–8) angeben und in diesem Kanal gegenwärtig Sounds abgespielt werden (aus anderen Teilen des Films), werden diese gestoppt und stattdessen der `RealMedia`-Audioabschnitt abgespielt.

Die gleichzeitige Wiedergabe mehrerer `RealMedia`-Darsteller ist nicht möglich. Wenn ein Film `RealMedia`-Darsteller enthält, die simultan abgespielt werden, sind ihre Sounds gleichzeitig im selben Soundkanal zu hören.

Beispiel

Die folgenden Beispiele zeigen, dass der Sound im `RealMedia`-Stream in Sprite 2 und im Darsteller „Real“ in Soundkanal 2 abgespielt wird:

```
-- Lingo syntax
put(sprite(2).soundChannel) -- 2
put(member("Real").soundChannel) -- 2

// JavaScript syntax
put(sprite(2).soundChannel); // 2
put(member("Real").soundChannel); // 2
```

In den folgenden Beispielen wird Soundkanal 1 dem `RealMedia`-Stream in Sprite 2 und im Darsteller "Real" zugeordnet:

```
-- Lingo syntax
sprite(2).soundChannel = 1
member("Real").soundChannel = 1

// JavaScript syntax
sprite(2).soundChannel = 1;
member("Real").soundChannel = 1;
```

Siehe auch

`realPlayerNativeAudio()`

soundDevice

Syntax

```
-- Lingo syntax
_sound.soundDevice

// JavaScript syntax
_sound.soundDevice;
```

Beschreibung

Mit dieser Soundeigenschaft können Sie bei der Filmwiedergabe ein Soundmischgerät einstellen. Lesen/Schreiben.

Die möglichen Einstellungen für `soundDevice` sind die in `soundDeviceList` aufgelisteten Geräte.

Sie können auf verschiedene Soundgeräte Bezug nehmen. Die Soundgeräte für Windows haben jeweils unterschiedliche Vorteile.

- `MacroMix` (Windows®) – Der niedrigste gemeinsame Nenner zur Wiedergabe in Windows. Dieses Gerät funktioniert auf jedem Windows-Computer; seine Latenz ist jedoch nicht so gut wie die anderer Geräte.
- QT3Mix (Windows) – Mischt Sound mit QuickTime Audio und anderen Anwendungen, sofern diese `DirectSound` unterstützen. Für die Verwendung dieses Geräts muss QuickTime installiert sein, das über eine geringere Latenz verfügt als `MacroMix`.
- `DirectSound`® (Windows) – Ähnlich wie QT3Mix, bietet aber höhere Latenz.
- `MacSoundManager` (Macintosh) – Das einzige für den Macintosh verfügbare Soundgerät.

Beispiel

Die folgende Anweisung stellt das Soundgerät auf `MacroMix` für einen Windows-Computer ein. Wenn der Einsatz des neu zugewiesenen Geräts fehlschlägt, wird die Eigenschaft `soundDevice` nicht geändert.

```
-- Lingo syntax
_sound.soundDevice = "MacroMix"

// JavaScript syntax
_sound.soundDevice = "MacroMix";
```

Siehe auch

`Sound`, `soundDeviceList`

soundDeviceList

Syntax

```
-- Lingo syntax
_sound.soundDeviceList

// JavaScript syntax
_sound.soundDeviceList;
```

Beschreibung
Diese Soundeigenschaft erstellt eine lineare Liste aus allen auf dem aktuellen Computer verfügbaren Soundgeräten. Nur Lesen.

Für den Mac listet diese Eigenschaft nur ein Gerät auf, den `MacSoundManager`.

Beispiel
Folgende Anweisung zeigt eine typische Soundgeräteliste auf einem Windows-Computer an:

```
-- Lingo syntax
trace(_sound.soundDeviceList)

// JavaScript syntax
trace(_sound.soundDeviceList);
```

Siehe auch
`Sound`, `soundDevice`

soundEnabled

Syntax

```
-- Lingo syntax
_sound.soundEnabled

// JavaScript syntax
_sound.soundEnabled;
```

Beschreibung
Diese Soundeigenschaft bestimmt, ob der Sound eingeschaltet (`TRUE`, Standard) oder ausgeschaltet (`FALSE`) ist. Lesen/Schreiben.

Wenn Sie diese Eigenschaft auf `FALSE` setzen, wird der Sound ausgeschaltet, die Lautstärkeeinstellung jedoch nicht geändert.

Beispiel
Die folgende Anweisung kehrt `soundEnabled` ins Gegenteil der aktuellen Einstellung um, d. h. sie schaltet den Sound ein, wenn er ausgeschaltet ist, und schaltet ihn aus, wenn er eingeschaltet ist:

```
-- Lingo syntax
_sound.soundEnabled = not(_sound.soundEnabled)

// JavaScript syntax
_sound.soundEnabled = !(_sound.soundEnabled);
```

Siehe auch
`Sound`

soundKeepDevice

Syntax

```
-- Lingo syntax
_sound.soundKeepDevice

// JavaScript syntax
_sound.soundKeepDevice;
```

Beschreibung

Diese nur für Windows geltende Soundeigenschaft bestimmt, ob der Soundtreiber jedes Mal, wenn ein Sound abgespielt werden muss, entladen und neu geladen wird. Lesen/Schreiben.

Der Standardwert dieser Eigenschaft ist `TRUE` und verhindert, dass der Soundtreiber jedes Mal, wenn ein Sound abgespielt werden muss, entladen und neu geladen wird.

Eventuell müssen Sie diese Eigenschaft auf `FALSE` setzen, bevor Sie einen Sound abspielen, um sicherzustellen, dass das Soundgerät nach der Wiedergabe des Sounds entladen wird und anderen Anwendungen oder Prozessen auf dem Computer zur Verfügung steht.

Das Einstellen dieser Eigenschaft auf `FALSE` kann die Leistung beeinträchtigen, wenn in der Director-Anwendung häufig Sound abgespielt wird.

Beispiel

Die folgende Anweisung setzt die Eigenschaft `soundKeepDevice` auf FALSE:

```
-- Lingo syntax
_sound.soundKeepDevice = FALSE

// JavaScript syntax
_sound.soundKeepDevice = false;
```

Siehe auch

Sound

soundLevel

Syntax

```
-- Lingo syntax
_sound.soundLevel

// JavaScript syntax
_sound.soundLevel;
```

Beschreibung

Diese Soundeigenschaft gibt die Lautstärke des Sounds zurück, der über den Computerlautsprecher abgespielt wird, oder legt sie fest. Lesen/Schreiben.

Die möglichen Werte liegen zwischen 0 (kein Sound) und 7 (maximale Lautstärke, Standard).

In Windows wird die Systemsoundeinstellung mit der Lautstärkeeinstellung der externen Lautsprecher verbunden. Daher kann die tatsächliche Lautstärke nach Einstellung des Soundpegels unterschiedlich ausfallen. Vermeiden Sie aus diesem Grund die Verwendung der Eigenschaft `soundLevel`, es sei denn, Sie sind sicher, dass die Endlautstärke für den Benutzer in jedem Fall akzeptabel ist. Es ist besser, die einzelnen Lautstärkewerte der Kanäle und Sprites mit Eigenschaft `volume` des Sound Channel-Objekts einzustellen.

Diese Werte entsprechen den Einstellungen im Mac-Kontrollfeld „Ton“. Über diese Eigenschaft kann Skriptcode die Lautstärke direkt ändern oder eine andere Aktion ausführen, wenn der Sound einen bestimmten Pegel erreicht hat.

Ein Beispiel für `soundLevel` in einem fertigen Film finden Sie im Film „Sound Control“ im Ordner „Learning\\Lingo_Examples“ im Director-Anwendungsordner.

Beispiel

Die folgende Anweisung setzt die Variable `oldSound` auf die gleiche Stufe wie den aktuellen Sound:

```
-- Lingo syntax
oldSound = _sound.soundLevel

// JavaScript syntax
var oldSound = _sound.soundLevel;
```

Die folgende Anweisung stellt den Soundpegel auf 5:

```
-- Lingo syntax
_sound.soundLevel = 5

// JavaScript syntax
_sound.soundLevel = 5;
```

Siehe auch

Sound, volume (Windows Media)

soundMixMedia

Syntax

```
-- Lingo syntax
_sound.soundMixMedia

// JavaScript syntax
_sound.soundMixMedia;
```

Beschreibung

Diese Soundeigenschaft bestimmt, ob der Sound von Flash-Darstellern mit in den Drehbuchsoundkanälen vorhandenen Sounds gemischt wird. Lesen/Schreiben.

Für Filme, die mit Director 7 und späteren Versionen erstellt wurden, wird diese Eigenschaft standardmäßig auf `TRUE` gesetzt, für alle anderen Filme auf `FALSE`. Sie kann ebenfalls nur unter Windows verwendet werden.

Wenn diese Eigenschaft `TRUE` ist, wird der Sound von Flash-Darstellern mit in den Drehbuchsoundkanälen vorhandenen Sounds gemischt. Das Mischen und Abspielen von Sounds übernimmt Director für Flash-Darsteller.

Es kann sein, dass Flash-Sounds etwas anders wiedergegeben werden. Um Flash-Sounds so zu hören, wie sie in Flash wiedergegeben werden, setzen Sie diese Eigenschaft auf `FALSE`.

Ist sie auf FALSE gesetzt, werden Flash-Sounds nicht gemischt und müssen zu einem gesonderten Zeitpunkt abgespielt werden.

Beispiel

Die folgenden Anweisungen zeigen den Standardwert der Eigenschaft soundMixMedia in der Erstellungsumgebung an.

```
-- Lingo
put _sound.soundMixMedia
-- 1

// Javascript
trace( _sound.soundMixMedia);
// 1
```

Siehe auch

Sound

soundObjectList

Syntax

```
Mixer.soundObjectList
```

Beschreibung

Diese Audiomixereigenschaft gibt die Liste von Soundobjekten im Mixer zurück.

Beispiele

```
--Lingo syntax
on mouseUp me
put mixer1.soundObjectList.count -- Displays the count of the sound objects in mixer1.
end

// JavaScript syntax
function mouseUp()
{
put (mixer1.soundObjectList.count ); // Displays the count of the sound objects
// in mixer1.


}
```

Siehe auch

Mixer

source

Syntax

```
sprite(whichSprite).camera.backdrop[backdropIndex].source
member(whichCastmember).camera(whichCamera).backdrop
[backdropIndex].source
sprite(whichSprite).camera.overlay[overlayIndex].source
member(whichCastmember).camera(whichCamera).overlay
[overlayIndex].source
```

Beschreibung

Mit dieser 3D-Hintergrund- und -Überlagerungseigenschaft können Sie die Textur ermitteln und festlegen, die als Ausgangsbild für die Überlagerung bzw. den Hintergrund dienen soll.

Beispiel

Die folgende Anweisung setzt die Quelle von Hintergrund 1 auf die Textur „Zeder“:

```
sprite(3).camera.backdrop[1].source =
sprite(3).member.texture("Cedar")
```

Siehe auch

bevelDepth, overlay

sourceFileName

Syntax

```
flashCastMember.sourceFileName
```

Beschreibung

Diese Flash-Darstellereigenschaft gibt den Pfadnamen der FLA-Quelldatei an, die für Launch-and-Edit-Operationen (Aufrufen und Bearbeiten) verwendet werden soll. Diese Eigenschaft kann getestet und eingestellt werden. Der Standardwert ist ein leerer String.

Beispiel

Die folgende Anweisung setzt den `sourceFileName` des Flash-Darstellers "SWF" auf C:\FlashFiles\myFile.fla:

```
-- Lingo
member("SWF").sourceFileName = "C:\FlashFiles\myFile.fla"

// Javascript
member("SWF").sourceFileName = "C:\FlashFiles\myFile.fla";
```

sourceRect

Syntax

```
-- Lingo syntax
windowObjRect.sourceRect

// JavaScript syntax
windowObjRect.sourceRect;
```

Beschreibung

Diese Fenstereigenschaft gibt die Originalbühnenkoordinaten des Films an, der in einem Fenster abgespielt wird. Nur Lesen.

Mit dieser Eigenschaft können Sie ein Fenster wieder auf seine ursprüngliche Position zurücksetzen, nachdem es an eine andere Stelle gezogen wurde oder sein Rechteck eingestellt wurde.

Beispiel

Die folgende Anweisung zeigt die Originalkoordinaten der Bühne `Control_panel` im Nachrichtenfenster an:

```
-- Lingo syntax
put(window("Control_panel").sourceRect)

// JavaScript syntax
put(window("Control_panel").sourceRect);
```

Siehe auch

`Window`

specular (light)

Syntax

```
member(whichCastmember).light(whichLight).specular
```

Beschreibung

Mit dieser 3D-Lichteigenschaft können Sie ermitteln oder festlegen, ob die Glanzlichtfunktion eingeschaltet `(TRUE)` oder ausgeschaltet ist `(FALSE)`, d. h. ob an der Stelle, an der das auf das Modell auftreffende Licht zur Kamera hin reflektiert wird, ein Highlight erscheint. Die Größe eines Glanzlichts richtet sich danach, wie stark das Objekt glänzt. Bei Umgebungslichtern wird diese Eigenschaft ignoriert. Der Standardwert dieser Eigenschaft lautet `TRUE`.

Hinweis: *Wenn diese Eigenschaft ausgeschaltet ist, verbessert dies möglicherweise die Leistung.*

Beispiel

Die folgende Anweisung legt die Eigenschaft specular des Lichts "omni2" auf FALSE fest. Dieses Licht erzeugt keine Highlights. Wenn es sich hierbei um das einzige Licht handelt, das gegenwärtig in der Szene leuchtet, sind auf keinen Shadern in der Szene Glanzlichter zu sehen.

```
-- Lingo
member("3d world").light("omni2").specular = FALSE

// Javascript
member("3d world").getPropRef("light",1).specular = false;
```

Siehe auch

silhouettes, specularLightMap

specular (shader)

Syntax

```
member(whichCastmember).shader(whichShader).specular
```

Beschreibung

Mit dieser 3D-Eigenschaft für den Standard-Shader können Sie die Glanzlichtfarbe eines Shaders ermitteln und festlegen. Diese Eigenschaft bestimmt die Farbe des Glanzlichts, das erzeugt wird, wenn die Glanzlichtfunktion aktiviert ist. Damit diese Eigenschaft eine sichtbare Wirkung hat, muss es in der Szene Lichter geben, bei denen die Eigenschaft "specular" auf TRUE gesetzt ist. Die Glanzlichtfarbe wird durch die Farbe der Lichtquellen, die das Objekt beleuchten, beeinflusst. Wenn die Glanzlichtfarbe weiß, die Farbe eines Lichts aber rot ist, erscheint auf dem Objekt ein rotes Glanzlicht. Der Standardwert dieser Eigenschaft ist „rgb(255, 255, 255)“, d. h. weiß.

Alle Shader können auf die Eigenschaften des #standard-Shaders zugreifen. Darüber hinaus haben die Shader #engraver, #newsprint und #painter noch weitere Eigenschaften, die speziell für den jeweiligen Shader-Typ gelten. Weitere Informationen finden Sie unter newShader.

Beispiel

```
-- Lingo
put member("scene").shader("plutomat").specular
--rgb(11, 11, 11)

// Javascript
trace(member("scene").getPropRef("shader",1).specular
// <rgb(11,11,11)>
```

Siehe auch

silhouettes, specular (light), specularColor, emissive

specularColor

Syntax

```
member(whichCastmember).specularColor
```

Beschreibung

Mit dieser 3D-Darstellereigenschaft können Sie den RGB-Wert der Glanzlichtfarbe des ersten Shaders im Darsteller ermitteln und festlegen. Der Standard-Shader ist immer der erste Shader in der Shader-Palette des Darstellers. Diese und alle anderen 3D-Darstellereigenschaften werden zusammen mit dem Darsteller gespeichert und wiederhergestellt, wenn Sie den Film zum nächsten Mal öffnen. Der Standardwert dieser Eigenschaft ist „rgb(255, 255, 255)“, d. h. weiß.

Beispiel

Die folgende Anweisung setzt die Glanzlichtfarbe des ersten Shaders im Darsteller „Szene“ auf „rgb(255, 0, 0)“. Dies entspricht member("Szene").shader[1].specular = rgb(255, 0, 0). Bei der zweiten Syntax wird der neue Wert jedoch beim Speichern des Films nicht zusammen mit dem Darsteller gespeichert. Nur bei Verwendung von`specularColor` wird der neue Farbwert gespeichert.

```
-- Lingo
member("Scene").specularColor = rgb(255, 0, 0)

// Javascript
member("Scene").specularColor = rgb(255, 0, 0);
```

Siehe auch

`silhouettes`, `specular (light)`, `specular (shader)`

specularLightMap

Syntax

```
member(whichCastmember).shader(whichShader).specularLightMap
member(whichCastmember).model(whichModel).shader.specularLightMap
member(whichCastmember).model(whichModel).shaderList[shaderListIndex].specularLightMap
```

Beschreibung

Mit dieser 3D-Eigenschaft für Standard-Shader können Sie die fünfte Texturebene des angegebenen Standard-Shaders ermitteln oder festlegen. Diese Eigenschaft wird ignoriert, wenn der Modifizierer `toon` auf die Modellressource angewendet wird.

Mögliche Werte:

- `textureModeList[5] = #specular`
- `blendFunctionList[5] = #add`
- `blendFunctionList[1] = #replace`
- `default = void`

Diese Hilfseigenschaft bietet eine einfache Schnittstelle zur Vereinheitlichung des Glanzlicht-Mapping.

Alle Shader können auf die Eigenschaften des `#standard`-Shaders zugreifen. Darüber hinaus haben die Shader `#engraver`, `#newsprint` und `#painter` noch weitere Eigenschaften, die speziell für den jeweiligen Shader-Typ gelten. Weitere Informationen finden Sie unter `newShader`.

Beispiel

Die folgende Anweisung legt die Textur "Oval" als `specularLightMap` des vom Modell `GlassBox` verwendeten Shaders fest:

```
-- Lingo
member("3DPlanet").model("GlassBox").shader.specularLightMap = \
member("3DPlanet").texture("Oval")

// Javascript
member("3DPlanet").getPropRef("model",1).getProp("shader").specularLightMap
=member("3DPlanet").getPropRef("texture",1);
```

Siehe auch

`diffuseLightMap`

spotAngle

Syntax

```
member(whichCastmember).light(whichLight).spotAngle
```

Beschreibung

Mit dieser 3D-Eigenschaft können Sie den Winkel des Lichtprojektionskegels ermitteln oder festlegen. Licht, das außerhalb des für diese Eigenschaft angegebenen Winkels fällt, trägt nicht zur Intensität bei. Diese Eigenschaft nimmt Werte zwischen 0,0 und 180,.00 an und besitzt den Standardwert 90,0. Der von Ihnen angegebene Fließkommawert entspricht der Hälfte dieses Wertes; falls Sie beispielsweise einen 90°-Winkel angeben möchten, übergeben Sie den Wert 45.0.

Beispiel

Die folgende Anweisung setzt die Eigenschaft `spotAngle` des Licht "unidirektional" auf 8. Der Winkel des Lichtprojektionskegels beträgt somit 16°.

```
-- Lingo
member("3d world").light("unidirectional").spotAngle = 8

// Javascript
member("3d world").getPropRef("light",1).spotAngle = 8;
```

spotDecay

Syntax

```
member(whichCastmember).light(whichLight).spotDecay
```

Beschreibung

Mit dieser 3D-Lichteigenschaft können Sie ermitteln und festlegen, ob die Intensität eines Spotlichts mit zunehmender Entfernung von der Kamera abnimmt. Der Standardwert dieser Eigenschaft lautet `FALSE`.

Beispiel

Die folgende Anweisung setzt die Eigenschaft `spotDecay` von Licht 1 auf TRUE. Modelle, die weiter von Licht 1 entfernt sind, erscheinen dunkler als Modelle, die sich näher an Licht 1 befinden.

```
-- Lingo
member("Scene").light[1].spotDecay = TRUE

// Javascript
member("Scene").getPropRef("light",1).spotDecay = true;
```

sprite (Film)

Syntax

```
-- Lingo syntax
_movie.sprite[spriteNameOrNum]

// JavaScript syntax
_movie.sprite[spriteNameOrNum];
```

Beschreibung

Diese Filmeigenschaft bietet indizierten oder benannten Zugriff auf einen Filmdarsteller. Nur Lesen.

Das Argument *spriteNameOrNum* kann eine Zeichenfolge sein, die den Namen des Sprites angibt, oder eine Ganzzahl, die die Nummer des Sprite angibt.

Beispiel

Die folgende Anweisung setzt die Variable `sportSprite` auf das Film-Sprite 5:

```
-- Lingo syntax
sportSprite = _movie.sprite[5]

// JavaScript syntax
var sportSprite = _movie.sprite[5];
```

Siehe auch

Movie

sprite (Sprite-Kanal)

Syntax

```
-- Lingo syntax
spriteChannelObjRef.sprite

// JavaScript syntax
spriteChannelObjRef.sprite;
```

Beschreibung

Diese Sprite-Kanal-Eigenschaft gibt einen Verweis auf das Sprite im aktuellen Bild eines Sprite-Kanals zurück. Nur Lesen.

Beispiel

Die folgende Anweisung setzt die Variable `mySprite` auf das Sprite im Sprite-Kanal namens "Ribbon".

```
-- Lingo syntax
mySprite = channel("Ribbon").sprite

// JavaScript syntax
var mySprite = channel("Ribbon").sprite;
```

Siehe auch

Sprite Channel

spriteNum

Syntax

```
-- Lingo syntax
spriteObjRef.spriteNum

// JavaScript syntax
spriteObjRef.spriteNum;
```

Beschreibung

Diese Sprite-Eigenschaft zeigt die Nummer des Kanals an, in dem sich das Sprite des Verhaltens befindet, und stellt es anderen Verhalten zur Verfügung. Nur Lesen.

Deklarieren Sie die Eigenschaft einfach oben im Verhalten zusammen mit allen anderen Eigenschaften, die ein Verhalten verwenden soll.

Wenn Sie die Prozedur `new()` verwenden, um eine Instanz des Verhaltens zu erstellen, muss die `new()`-Prozedur des Skripts ausdrücklich die Eigenschaft `spriteNum` auf die Sprite-Nummer setzen. Auf diese Weise können Sie feststellen, an welchem Sprite das Skript angebracht ist. Die Sprite-Nummer muss als Argument an die `new()`-Prozedur übergeben werden, wenn die `new()`-Prozedur aufgerufen wird.

Beispiel

In der folgenden Prozedur wird die Eigenschaft `spriteNum` automatisch auf Skriptinstanzen eingestellt, die vom System erstellt werden:

```
-- Lingo syntax
property spriteNum, pMySpriteRef

on mouseDown me
    sprite(spriteNum).member = member("DownPict")
end

// JavaScript syntax
function mouseDown() {
    sprite(this.spriteNum).member = member("DownPict");
}
```

Die folgende Prozedur verwendet aus praktischen Gründen den automatisch in die Eigenschaft `spriteNum` eingefügten Wert, um der neuen Eigenschaftsvariablen `pMySpriteRef` den Sprite-Bezug zuzuordnen:

```
-- Lingo syntax
property spriteNum, pMySpriteRef

on beginSprite me
    pMySpriteRef = sprite(me.spriteNum)
end

// JavaScript syntax
function beginSprite() {
    this.pMySpriteRef = sprite(this.spriteNum);
}
```

Mit dieser Vorgehensweise können Sie den Bezug `pMySpriteRef` später im Skript verwenden, während die Prozedur die Syntax:

```
-- Lingo syntax
currMember = pMySpriteRef.member

// JavaScript syntax
var currMember = pMySpriteRef.member
```

statt der folgenden Syntax verwendet, die etwas länger ist:

```
Lingo syntax
currMember = sprite(spriteNum).member

// JavaScript syntax
var currMember = sprite(this.spriteNum).member
```

Diese alternative Vorgehensweise bietet keine neue Funktionalität, sondern dient nur praktischen Zwecken.

Siehe auch

`new()`, `Sprite`

stage

Syntax

```
-- Lingo syntax
_movie.stage

// JavaScript syntax
_movie.stage;
```

Beschreibung

Diese Filmeigenschaft verweist auf den Hauptfilm. Nur Lesen.

Sie ist nützlich, wenn eine Nachricht von einem Child-Film an den Hauptfilm gesendet wird.

Beispiel

Die folgende Anweisung zeigt die aktuelle Einstellung der Bühne an:

```
-- Lingo syntax
put(_movie.stage.rect)

// JavaScript syntax
put(_movie.stage.rect);
```

Siehe auch

Movie

startAngle

Syntax

```
member(whichCastmember).modelResource(whichModelResource).
startAngle
modelResourceObjectReference.startAngle
```

Beschreibung

Mit dieser 0D-Eigenschaft können Sie bei einer Modellressource vom Typ `#cylinder` oder `#sphere` die Eigenschaft `startAngle` der referenzierten Modellressource als Fließkommazahl zwischen 0.0 und 360.0 abrufen und festlegen. Der Standardwert für diese Eigenschaft lautet 0.0.

Die Eigenschaft `startAngle` bestimmt den anfänglichen Ablenkwinkel der Modellressource und wird zusammen mit der Eigenschaft `endAngle` zum Zeichnen von Kugeln und Zylindern verwendet. Um beispielsweise eine Halbkugel zu zeichnen, setzen Sie `startAngle` auf 0.0 und `endAngle` auf 180.0.

Beispiel

Die folgende Anweisung stellt den Wert `startAngle` der Modellressource „Sphere01“ auf 0,0 ein. Falls der Winkel `endAngle` auf 90 eingestellt ist, erscheint jeweils nur ein Viertel eines Modells, das von dieser Modellressource Gebrauch macht.

```
-- Lingo
put member("3D World").modelResource(1).startAngle
-- 0.0

// Javascript
trace(member("3D World").getPropRef("modelResource",1).startAngle);
// 0
```

Siehe auch

endAngle

startFrame

Syntax

```
-- Lingo syntax
spriteObjRef.startFrame

// JavaScript syntax
spriteObjRef.startFrame;
```

Beschreibung

Diese Sprite-Eigenschaft gibt die Bildnummer des Anfangsbildes in einem Sprite-Einschluss zurück. Nur Lesen.

Diese Eigenschaft ist nützlich, um den Bereich im Drehbuch zu bestimmen, denein bestimmtes Sprite umfasst. Sie steht nur für Bilder zur Verfügung, in denen das Sprite enthalten ist, und kann nicht auf Sprites in anderen Bildern des Films angewendet werden.

Beispiel

Die folgende Anweisung zeigt im Nachrichtenfenster das Anfangsbild des Sprites in Kanal 5 an:

```
-- Lingo syntax
put(sprite(5).startFrame)

// JavaScript syntax
put(sprite(5).startFrame);
```

Siehe auch

endFrame, Sprite

startTime (Sound Channel)

Syntax

```
-- Lingo syntax
soundChannelObjRef.startTime

// JavaScript syntax
soundChannelObjRef.startTime;
```

Beschreibung

Diese Soundkanaleigenschaft gibt die Startzeit des gerade laufenden oder angehaltenen Sounds an, die eingestellt wurde, als der Sound in die Warteschlange eingereiht wurde. Nur Lesen.

Diese Eigenschaft kann nicht eingestellt werden, nachdem der Sound in die Warteschlange eingereiht wurde. Wenn beim Einreihen des Sounds in die Warteschlange kein Wert angegeben wurde, gibt diese Eigenschaft 0 zurück.

Beispiel

Die folgende Anweisung startet das Digitalvideo-Sprite in Kanal 5 100 Ticks nach Beginn des Digitalvideos:

```
-- Lingo syntax
sprite(5).startTime = 100

// JavaScript syntax
sprite(5).startTime = 100;
```

Siehe auch

Sound Channel

startTime (Soundobjekt)

Syntax

```
soundObject.startTime
```

Beschreibung

Diese Soundobjekteigenschaft gibt die Startzeit (in Millisekunden) des aktuellen Soundobjekts zurück.

Beispiele

```
soundObject.startTime = 6890
--Lingo syntax
on mouseUp me
    put soundObjRef.startTime -- Displays the start time for the sound object
-- associated with soundobjectRef.
end

// JavaScript syntax
function mouseUp(){
put (soundObjRef.startTime) ; // Displays the start time for the sound object
// associated with soundobjectRef.
}
```

Siehe auch

endTime (Soundobjekt)

startTimeList

Syntax

```
-- Lingo syntax
dvdObjRef.startTimeList

// JavaScript syntax
dvdObjRef.startTimeList;
```

Beschreibung

Diese DVD-Eigenschaft ist eine Eigenschaftsliste, die die Zeit oder das Kapitel angibt, zu der bzw. mit dem die Wiedergabe starten soll.

Lesen/Schreiben.

Eine `startTimeList` ist eine Eigenschaftsliste, die kapitelbasiert oder zeitbasiert sein kann.

Eine kapitelbasierte `startTimeList` enthält die folgenden Eigenschaften:

- `title`. Gibt den Titel an, der das wiederzugebende Kapitel enthält.
- `chapter`. Gibt das wiederzugebende Kapitel an.

Die folgende `startTimeList` beginnt die Wiedergabe bei Kapitel 2 von Titel 1:

```
[#title:1, #chapter:2]
```

Eine zeitbasierte `startTimeList` enthält die folgenden Eigenschaften:

- `title`. Gibt den Titel an.
- `hours`. Gibt die Stunde an, zu der die Wiedergabe startet.
- `min`. Gibt die Minute an, zu der die Wiedergabe startet.
- `sec`. Gibt die Sekunde an, zu der die Wiedergabe startet.
- `frames`. Gibt die Bilder an, bei denen die Wiedergabe startet.

Die folgende `startTimeList` beginnt die Wiedergabe zu einer bestimmten Zeit in Titel 1:

```
[#title:1, #hours:0, #minutes:45, #seconds:15, #frames:15]
```

Die folgende `startTimeList` listet nur einen Zeitparameter auf:

```
[#title:1, #seconds:15]
```

Die folgende `startTimeList` kann gelöscht werden, indem sie auf "0" festgelegt wird.

Beispiel

Die folgende Anweisung stellt die `startTimeList` des DVD-Objekts auf 0 ein.

```
-- Lingo
sprite(1).startTimeList = 0

// Javascript
sprite(1).startTimeList = 0;
```

Siehe auch

DVD, play() (DVD), stopTimeList

state (3D)

Syntax

```
member(whichCastmember).state
```

Beschreibung

Diese 3D-Eigenschaft gibt den aktuellen Zustand des referenzierten Darstellers beim Streamen und Laden zurück. Diese Eigenschaft bezieht sich entweder auf den ursprünglichen Dateiimport oder die letzte Dateiladeanforderung.

Die Darstellereigenschaft `state` bestimmt, welche 3D-Lingo-Befehle auf den Darsteller angewendet werden können.

Diese Eigenschaft kann einen der folgenden Werte aufweisen:

- `0` – zeigt an, dass der Darsteller gegenwärtig nicht geladen ist und somit keine 3D-Daten verfügbar sind. Auf den Darsteller sollten keine 3D-Lingo-Befehle angewendet werden.
- `1` – zeigt an, dass das Laden von Mediendaten begonnen hat.
- `2` – zeigt an, dass das anfängliche Ladesegment des Darstellers geladen ist. Alle Objekte mit der Stream-Priorität 0 (wird beim Erstellen der Modelldatei festgelegt) werden an dieser Stelle geladen, da sie Teil des anfänglichen Ladesegments sind. Für Objekte mit der Ladepriorität 0 können Sie die meisten 3D-Lingo-Befehle ausführen. Verwenden Sie während dieses Zustands jedoch nicht die Befehle `loadFile` und `resetWorld`.
- `3` – zeigt an, dass gegenwärtig alle zusätzlichen Mediendaten des Darstellers geladen und dekomprimiert werden. An dieser Stelle können die meisten 3D-Lingo-Befehle ausgeführt werden. Verwenden Sie während dieses Zustands jedoch nicht die Befehle `loadFile` und `resetWorld`.
- `4` – zeigt an, dass alle Mediendaten des Darstellers geladen sind und die gesamte Dekomprimierung abgeschlossen ist. Auf den Darsteller können jetzt alle 3D-Lingo-Befehle angewendet werden.
- `-1` – zeigt an, dass beim Streamen der Mediendaten ein undefinierter Fehler aufgetreten ist. Da der Fehler an jeder beliebigen Stelle des Ladeprozesses aufgetreten sein kann, ist der Zustand des Darstellers ungewiss.

Im Allgemeinen sollten Sie für 3D-Darsteller mit einem „state"-Wert unter 3 keine Lingo-Operationen durchführen.

Beispiel

Die folgende Anweisung zeigt, dass der Darsteller `PartyScene` fertig geladen und wiedergabebereit ist und dass beim Laden keine Fehler aufgetreten sind:

```
-- Lingo
put member("PartyScene").state
-- 4

// Javascript
trace(member("PartyScene").state);
// 4
```

state (Flash, SWA)

Syntax

```
-- Lingo syntax
memberObjRef.state

// JavaScript syntax
memberObjRef.state;
```

Beschreibung

Diese Darstellereigenschaft für Shockwave Audio- (SWA-) Streaming-Darsteller und Flash-Filmdarsteller gibt den aktuellen Zustand der gestreamten Datei an. Die Eigenschaften `streamName`, `URL` und `preLoadTime` können nur geändert werden, wenn der SWA-Sound nicht abgespielt wird.

Folgende Eigenschaften der SWA-Datei stellen sinnvolle Rückgabewerte nur dann zur Verfügung, wenn das Streaming der Datei begonnen hat: `cuePointNames`, `cuePointTimes`, `currentTime`, `duration`, `percentPlayed`, `percentStreamed`, `bitRate`, `sampleRate` und `numChannels`.

Für SWA-Streaming-Darsteller sind folgende Werte möglich:

- 0 – Streaming der Besetzung wurde gestoppt.
- 1 – Der Darsteller wird erneut geladen.
- 2 – Vorausladen erfolgreich beendet.
- 3 – Der Darsteller wird abgespielt.
- 4 – Der Darsteller wurde unterbrochen.
- 5 – Streaming des Darstellers beendet.
- 9 – Es ist ein Fehler aufgetreten.
- 10 - Nicht genügend Platz in der CPU.

Für Flash-Filmdarsteller gibt diese Eigenschaft nur dann einen gültigen Wert zurück, wenn der Director-Film läuft. Folgende Werte sind möglich:

- 0 – Der Darsteller befindet sich nicht im Speicher.
- 1 – Die Kopfzeile wird momentan geladen.
- 2 – Laden der Kopfzeile beendet.
- 3 – Es werden momentan die Medien des Darstellers geladen.
- 4 – Die Medien des Darstellers sind vollständig geladen.
- -1 – Es ist ein Fehler aufgetreten.

Diese Eigenschaft kann getestet, aber nicht eingestellt werden.

Beispiel

Die folgende Anweisung gibt eine Warnmeldung aus, wenn für den SWA-Streaming-Darsteller ein Fehler entdeckt wurde:

```
-- Lingo syntax
on mouseDown
    if member("Ella Fitzgerald").state = 9 then
        _player.alert("Sorry, can't find an audio file to stream.")
    end if
end

// JavaScript syntax
function mouseDown() {
    var ellaSt = member("Ella Fitzgerald").state;
    if (ellaSt == 9) {
        _player.alert("Sorry, can't find an audio file to stream.");
    }
}
```

Siehe auch

clearError(), getError() (Flash, SWA)

state (RealMedia)

Syntax

```
-- Lingo syntax
memberOrSpriteObjRef.state

// JavaScript syntax
memberOrSpriteObjRef.state;
```

Beschreibung

Diese Eigenschaft für `RealMedia`-Sprites und -Darsteller gibt den aktuellen Zustand des `RealMedia`-Streams als Ganzzahl im Bereich 1–4 zurück. Jeder „state"-Wert entspricht einer bestimmten Phase im Streamingprozess. Diese Eigenschaft ist während der Wiedergabe dynamisch und kann getestet, aber nicht eingestellt werden.

Der Streamingprozess beginnt, wenn der Abspielkopf den Einschluss des `RealMedia`-Sprites im Drehbuch erreicht, die Methode `play` für ein `RealMedia`-Sprite bzw. einen -Darsteller aufgerufen wird oder ein Benutzer auf die Schaltfläche „Abspielen" im `RealMedia`-Viewer klickt. Bei Aufruf dieser Eigenschaft wird ein numerischer Wert zurückgegeben, der den Zustand des Streamingprozesses für den `RealMedia`-Darsteller angibt. Für jeden Zustand gibt es einen oder mehrere `mediaStatus (RealMedia, Windows Media)`-Eigenschaftswerte; jeder `mediaStatus`-Wert ist nur in jeweils einem Zustand zu beobachten. Beispielsweise treten die `mediaStatus`-Werte `#seeking` und `#buffering` nur dann auf, wenn der `state`-Wert 3 lautet.

Der Wert der Eigenschaft `state` liefert wertvolle Aufschlüsse über die Auswirkung von Lingo-Elementen auf einen Darsteller. Wenn `member.state` kleiner als 2 ist, sind einige Lingo-Eigenschaften u. U. falsch und Lingo-Elemente, die von Eigenschaftsdaten abhängig sind, stimmen ebenfalls nicht. Wenn `member.state` größer oder gleich 2 und kleiner als 4 ist, wird der `RealMedia`-Darsteller zwar nicht angezeigt, aber alle Lingo-Eigenschaften und -Methoden besitzen korrekt definierte Werte und können zur Ausführung von Lingo-Operationen für den Darsteller verwendet werden.

Nach Einleitung des Streamingprozesses durchläuft die Eigenschaft `state` die folgenden Zustände, es sei denn, es tritt ein Fehler (-1) auf, der das Streaming verhindert.

-1 (error) gibt an, dass etwas nicht stimmt. Möglicherweise handelt es sich um einen Fehler, der noch aus dem letzten `RealMedia`-Stream übrig ist. Durch Überprüfung der Eigenschaft `lastError` können Sie u. U. weitere Informationen ausfindig machen. Dieser Zustand entspricht dem Wert `#error` der Eigenschaft `mediaStatus`.

0 (geschlossen) gibt an, dass das Streaming noch nicht begonnen hat, bzw. dass die Darstellereigenschaften sich in einem anfänglichen Zustand befinden oder Kopien einer früheren Wiedergabe des Darstellers sind. Dieser Zustand entspricht dem Wert `#closed` der Eigenschaft `mediaStatus`.

1 (Verbindung wird hergestellt) gibt an, dass das Streaming zwar begonnen hat, aber noch keine Serververbindung hergestellt ist und somit lokal noch nicht genügend Informationen zur Verarbeitung des Darstellers verfügbar sind. Dieser Zustand entspricht dem Wert `#connecting` der Eigenschaft `mediaStatus`.

2 (offen) gibt an, dass die Lingo-Eigenschaften aus dem tatsächlichen Stream aktualisiert wurden. Wenn `state` größer oder gleich 2 ist, sind die Eigenschaften `height`, `width` und `duration` des `RealMedia`-Streams bekannt. Dieser Zustand ist vorübergehend und ändert sich schnell in Zustand 3. Dieser Zustand entspricht dem Wert `#opened` der Eigenschaft `mediaStatus`.

3 (seeking or buffering) gibt an, dass alle Lingo-Eigenschaften des `RealMedia`-Darstellers auf dem neuesten Stand sind, der Darsteller aber noch nicht ganz abspielbereit ist. Auf der Bühne und im `RealMedia`-Viewer erscheint ein schwarzes Rechteck oder das `RealNetworks`-Logo. Wenn dieser Zustand auf eine Neupufferung wegen Netzwerküberlastung zurückzuführen ist, ändert sich der `state`-Wert schnell wieder in `4` (Wiedergabe läuft). Dieser Zustand entspricht dem Wert `#buffering` bzw. `#seeking` der Eigenschaft `mediaStatus`.

4 (playing) gibt an, dass der `RealMedia`-Stream problemlos abgespielt wird (bzw. unterbrochen ist). Bei der normalen Wiedergabe befindet sich der RealMedia-Stream in diesem Zustand. Dieser Zustand entspricht dem Wert `#playing` bzw. `#paused` der Eigenschaft `mediaStatus`.

Beispiel

Die folgenden Beispiele zeigen, dass der Zustand der Streams in Sprite 2 und im Darsteller „Real“ 0 lautet, d. h. dass sie geschlossen sind:

```
-- Lingo syntax
put(sprite(2).state) -- 0
put(member("Real").state) -- 0

// JavaScript syntax
put(sprite(2).state); // 0
put(member("Real").state); // 0
```

Siehe auch

`mediaStatus (RealMedia, Windows Media)`, `percentBuffered`, `lastError`

static

Syntax

```
-- Lingo syntax
memberOrSpriteObjRef.static

// JavaScript syntax
memberOrSpriteObjRef.static;
```

Beschreibung

Diese Darsteller- und Sprite-Eigenschaft steuert die Abspielleistung eines Flash-Film-Sprites abhängig davon, ob der Film eine Animation enthält. Wenn der Film eine Animation enthält (`FALSE`, Standard), zeichnet die Eigenschaft das Sprite in jedem Bild neu. Enthält der Film keine Animation (`TRUE`), zeichnet die Eigenschaft das Sprite nur dann neu, wenn es verschoben oder in seiner Größe verändert wurde.

Diese Eigenschaft kann getestet und eingestellt werden.

Hinweis: *Setzen Sie die `static`-Eigenschaft nur dann auf `TRUE`, wenn sich das Flash-Film-Sprite nicht mit anderen beweglichen Director-Sprites schneidet. Wenn sich ein Flash-Film mit beweglichen Director-Sprites schneidet, wird er eventuell nicht korrekt neu gezeichnet.*

Beispiel

Das folgende Sprite-Skript zeigt die Kanalnummer eines Flash-Film-Sprites im Nachrichtenfenster an und meldet, ob der Flash-Film eine Animation enthält:

```
-- Lingo syntax
property spriteNum

on beginSprite me
    if sprite(spriteNum).static then
        animationType = "does not have animation."
    else
        animationType = "has animation."
    end if
    put("The Flash movie in channel" && spriteNum && animationType)
end

// JavaScript syntax
function beginSprite() {
    var st = sprite(this.spriteNum).static;
    if (st == 1) {
        animationType = "does not have animation.";
    } else {
        animationType = "has animation.";
    }
    trace("The Flash movie in channel " + this.spriteNum + animationType);
}
```

staticQuality

Syntax

```
-- Lingo syntax
spriteObjRef.staticQuality

// JavaScript syntax
spriteObjRef.staticQuality;
```

Beschreibung

Diese QuickTime VR-Sprite-Eigenschaft gibt die verwendete Codec-Qualität an, wenn das Panoramabild statisch ist. Mögliche Werte: `#minQuality`, `#maxQuality` und `#normalQuality`.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Die folgende Anweisung zeigt die statische Qualität `staticQuality` des QuickTime-Sprites im Nachrichtenfenster an.

```
-- Lingo
put sprite(1).staticQuality
-- #normalQuality

// Javascript
trace(sprite(1).staticQuality)
// symbol(normalQuality)
```

status

Syntax

```
-- Lingo syntax
soundChannelObjRef.status

// JavaScript syntax
soundChannelObjRef.status;
```

Beschreibung

Diese Soundkanaleigenschaft gibt den Status eines Soundkanals an. Nur Lesen.

Folgende Werte sind möglich:

Status	Name	Bedeutung
0	Idle	Es werden keine Sounds in die Warteschlange eingereiht oder abgespielt.
1	Loading	Ein Sound in der Warteschlange wird vorausgeladen, aber noch nicht abgespielt.
2	Queued	Der Soundkanal hat einen Sound in der Warteschlange vollständig vorausgeladen, spielt ihn aber noch nicht ab.
3	Playing	Ein Sound wird abgespielt.
4	Paused	Ein Sound ist angehalten.

Beispiel

Die folgende Anweisung zeigt den aktuellen Status von Soundkanal 2 im Nachrichtenfenster an:

```
-- Lingo syntax
put(sound(2).status)

// JavaScript syntax
put(sound(2).status);
```

Siehe auch

Sound Channel

status (Mixer)

Syntax

```
mixer.status (Read-only)
```

Beschreibung

Diese Audioeigenschaft gibt den Status des Mixers zurück. Diese Eigenschaft kann folgende Werte annehmen:

- #playing
- #paused
- #stopping
- #stopped

Wenn sich ein Mixer im Zustand `#stopping` befindet, werden Aufrufe der Funktionen `play` und `stop` vom Soundobjekt und den enthaltenen Mixern ignoriert.

Beispiele

Die folgenden Beispiele starten `mixer1` am Bild `exit`, falls der Mixer zuvor gestoppt wurde.

```
--Lingo syntax
on exitFrame me
     if (mixer1.status = #stopped)then
       mixer1.play()
     end if
end

// JavaScript syntax
function exitFrame() {
if (mixer1.status = #stopped)then
     mixer1.play();
end if
}
```

Siehe auch

Mixer

status (Soundobjekt)

Syntax

```
soundObj.status (Read-only)
```

Beschreibung

Diese Soundobjekteigenschaft gibt den Status des Soundobjekts zurück. Diese Eigenschaft weist einen der folgenden Werte auf:

- `#playing`
- `#paused`
- `#stopping`
- `#stopped`

Wenn sich das Soundobjekt im Zustand `#stopping` befindet, werden Aufrufe wie `play` und `stop` ignoriert.

Beispiel

```
-- Lingo syntax
on exitFrame me
    put soundObjRef.status -- Returns the current status of the sound object.
end

// JavaScript syntax
function exitframe(){
put (soundObjRef.status) ; // Returns the current status of the sound object.

}
```

stillDown

Syntax

```
-- Lingo syntax
_mouse.stillDown

// JavaScript syntax
_mouse.stillDown;
```

Beschreibung

Diese Mauseigenschaft gibt an, ob der Benutzer die Maustaste gedrückt hält (`TRUE`) oder nicht (`FALSE`). Nur Lesen.

Diese Funktion bietet sich in einem `mouseDown`-Skript zum Auslösen bestimmter Ereignisse an, die nur auf die Funktion `mouseUp` folgen.

Mit Skriptcode kann `stillDown` nicht getestet werden, wenn die Eigenschaft innerhalb einer Schleife verwendet wird. Verwenden Sie in Schleifen stattdessen die Funktion `mouseDown`.

Beispiel

Die folgende Anweisung prüft, ob die Maustaste gedrückt wird, und ruft die Prozedur `dragProcedure` auf, wenn dies der Fall ist:

```
-- Lingo syntax
if (_mouse.stillDown) then
    dragProcedure
end if

// JavaScript syntax
if (_mouse.stillDown) {
    dragProcedure();
}
```

Siehe auch

`Mouse`, `mouseDown`, `mouseUp`

stopTime

Syntax

```
sprite(whichSprite).stopTime
the stopTime of sprite whichSprite
```

Beschreibung

Diese Sprite-Eigenschaft bestimmt, wann das angegebene Digitalvideo-Sprite angehalten wird. Der Wert von `stopTime` wird in Ticks gemessen.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Die folgende Anweisung stoppt das Digitalvideo-Sprite in Kanal 5 100 Ticks nach Beginn des Digitalvideos:

```
-- Lingo
sprite(5).stopTime = 100

// Javascript
sprite(5).stopTime = 100;
```

stopTimeList

Syntax

```
-- Lingo syntax
dvdObjRef.stopTimeList

// JavaScript syntax
dvdObjRef.stopTimeList;
```

Beschreibung

Diese DVD-Eigenschaft ist eine Eigenschaftsliste, die die Zeit oder das Kapitel angibt, zu der bzw. mit dem die Wiedergabe beendet werden soll.

Lesen/Schreiben.

Eine `stopTimeList` ist eine Eigenschaftsliste, die kapitelbasiert oder zeitbasiert sein kann.

Eine kapitelbasierte `stopTimeList` enthält die folgenden Eigenschaften:

- `title`. Gibt den Titel an.
- `chapter`. Gibt das Kapitel an. Die Wiedergabe wird im Anschluss an die Wiedergabe dieses Kapitels beendet.

Die folgende `stopTimeList` beendet die Wiedergabe bei Kapitel 4 von Titel 1:

```
[#title:1, #chapter:4]
```

Eine zeitbasierte `stopTimeList` enthält die folgenden Eigenschaften:

- BULLET_ITEM
- `title`. Gibt den Titel an.
- `hours`. Gibt die Stunde an, zu der die Wiedergabe endet.
- `min`. Gibt die Minute an, zu der die Wiedergabe endet.
- `sec`. Gibt die Sekunde an, zu der die Wiedergabe endet.
- `frames`. Gibt die Bilder an, bei denen die Wiedergabe endet.

Die folgende `stopTimeList` beendet die Wiedergabe zu einer bestimmten Zeit in Titel 1:

```
[#title:1, #hours:0, #minutes:55, #seconds:45, #frames:15]
```

Die folgende `stopTimeList` listet nur einen Zeitparameter auf:

```
[#title:1, #seconds:45]
```

Die folgende `stopTimeList` kann gelöscht werden, indem sie auf "0" festgelegt wird.

Siehe auch

DVD, play() (DVD), startTimeList

streamMode

Syntax

```
-- Lingo syntax
memberObjRef.streamMode

// JavaScript syntax
memberObjRef.streamMode;
```

Beschreibung

Diese Flash-Darstellereigenschaft steuert die Art und Weise, wie ein verknüpfter Flash-Filmdarsteller in den Speicher gestreamt wird, wie folgt:

- `#frame` (Standardeinstellung) – Streamt einen Teil des Darstellers immer dann, wenn das Director-Bild vorrückt, während das Sprite auf der Bühne erscheint.
- `#idle` – Streamt einen Teil des Darstellers immer dann, wenn ein Wartezustand eintritt, oder mindestens einmal pro Director-Bild, während das Sprite auf der Bühne erscheint.
- `#manual` – Streamt nur dann einen Teil des Darstellers in den Speicher, wenn der Befehl `stream` für diesen Darsteller ausgeführt wird.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Das folgende `startMovie`-Skript durchsucht die interne Besetzung nach Flash-Filmdarstellern und setzt ihre `streamMode`-Eigenschaften auf `#manual`:

```
-- Lingo syntax
on startMovie
    repeat with i = 1 to castLib(1).member.count
    if member(i, 1).type = #flash then
        member(i, 1).streamMode = #manual
    end if
    end repeat
end

// JavaScript syntax
function startMovie() {
    i = 1;
    while( i < (castLib(whichCast).member.count) + 1)
        var tp = member(i, whichCast).type;
        if (tp == "flash") {
            member(i, 1).streamMode = symbol("manual");
            i++;
        }
    }
}
```

streamName

Syntax

```
-- Lingo syntax
memberObjRef.streamName

// JavaScript syntax
memberObjRef.streamName;
```

Beschreibung

Diese Shockwave Audio- (SWA-) Darstellereigenschaft gibt eine URL oder einen Dateinamen für einen streamenden Darsteller an. Diese Eigenschaft funktioniert genau wie die Darstellereigenschaft `URL`.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Die folgende Anweisung verknüpft die Datei `BigBand` mit einem SWA-Streaming-Darsteller. Die verknüpfte Datei befindet sich auf dem Datenträger `MyDisk` im Ordner „Sounds“.

```
-- Lingo syntax
member("SWAstream").streamName = "MyDisk/sounds/BigBand.swa"

// JavaScript syntax
member("SWAstream").streamName = "MyDisk/sounds/BigBand.swa";
```

streamSize

Syntax

```
-- Lingo syntax
memberObjRef.streamSize

// JavaScript syntax
memberObjRef.streamSize;
```

Beschreibung

Diese Darstellereigenschaft liefert eine Ganzzahl, die die Gesamtzahl von Bytes zum Streamen des angegebenen Darstellers angibt. Die Eigenschaft `streamSize` gibt nur dann einen Wert an, wenn der Director-Film abgespielt wird.

Diese Eigenschaft kann getestet, aber nicht eingestellt werden.

Beispiel

Das folgende Bildskript prüft, ob das Streaming des Flash-Filmdarstellers „Einführung“ beendet ist. Ist dies nicht der Fall, aktualisiert das Skript einen Felddarsteller, der die Anzahl von Bytes, die bis dahin gestreamt sind (Darstellereigenschaft `bytesStreamed`), sowie die Gesamtzahl von Bytes in diesem Darsteller (Darstellereigenschaft `streamSize`) angibt. Das Skript bewirkt, dass der Abspielkopf im aktuellen Bild eine Schleife durchläuft, bis der Film vollständig in den Speicher geladen ist.

```
-- Lingo syntax
on exitFrame
    if member("Intro Movie").percentStreamed < 100 then
        member("Message Line").text = string(member("Intro Movie").bytesStreamed) && "of" \ &&
string(member("Intro Movie").streamSize) && "bytes have downloaded so far."
        _movie.go(_movie.frame)
    end if
end

// JavaScript syntax
function exitFrame() {
    var pctStm = member("Intro Movie").percentStreamed;
    var strSs = new String(member("Intro Movie").streamSize);
    var strIm = new String(member("Intro Movie").bytesStreamed);
    if (pctStm < 100) {
        member("Message Line").text = strStm + " of " + strSS+ " bytes have downloaded so far.";
        _ movie.go(_movie.frame);
    }
}
```

streamSize (3D)

Syntax

```
member(whichCastmember).streamSize
```

Beschreibung

Mit dieser 3D-Eigenschaft können Sie die Größe des herunterzuladenden Datenstroms ermitteln (zwischen „0“ und `maxInteger`). Diese Eigenschaft bezieht sich entweder auf den ursprünglichen Dateiimport oder die letzte Dateiladeanforderung (`loadFile()`).

Beispiel

Die folgende Anweisung zeigt, dass die letzte Dateiladeanforderung für den Darsteller „Szene“ 325300 Bytes umfasste:

```
put member("Scene").streamSize
-- 325300
```

Siehe auch

`bytesStreamed (3D)`, `percentStreamed (3D)`, `state (3D)`, `preLoad (3D)`

strokeColor

Syntax

```
-- Lingo syntax
memberObjRef.strokeColor

// JavaScript syntax
memberObjRef.strokeColor;
```

Beschreibung
Diese Vektorformdarstellereigenschaft zeigt die Farbe des Umrandungsstriches der Form als RGB-Wert an.

Ein Beispiel für `strokeColor` in einem fertigen Film finden Sie im Film „Vector Shapes“ im Ordner „Learning\\Lingo_Examples“ im Director-Anwendungsordner.

Beispiel
Die folgende Anweisung setzt `strokeColor` für den Darsteller „Linie“ auf Rot:

```
-- Lingo syntax
member("line").strokeColor = color(255, 0, 0)

// JavaScript syntax
member("line").strokeColor = color(255, 0, 0);
```

Siehe auch
`color()`, `fillColor`, `endColor`, `backgroundColor`

strokeWidth

Syntax

```
-- Lingo syntax
memberObjRef.strokeWidth

// JavaScript syntax
memberObjRef.strokeWidth;
```

Beschreibung
Diese Vektorformdarstellereigenschaft gibt die Breite des Umrandungsstriches der Form in Pixel an.

Der Wert ist eine Fließkommazahl zwischen 0 und 100 und kann getestet und eingestellt werden.

Ein Beispiel für `strokeWidth` in einem fertigen Film finden Sie im Film „Vector Shapes“ im Ordner „Learning\\Lingo_Examples“ im Director-Anwendungsordner.

Beispiel
Der folgende Code setzt `strokeWidth` für den Darsteller "Linie" auf 10 Pixel:

```
-- Lingo syntax
member("line").strokeWidth = 10

// JavaScript syntax
member("line").strokeWidth = 10;
```

style

Syntax

```
member(whichCastmember).model(whichModel).toon.style
member(whichCastmember).model(whichModel).shader.style
member(whichCastmember).shader(whichShader).style
```

Beschreibung

Diese 3D-Eigenschaft für den Modifizierer `toon` und den Shader `painter` bestimmt, wie der Modifizierer `toon` und der Shader `painter` Farbe auf ein Modell anwenden. Mögliche Werte:

- `#toon`: abrupte Übergänge zwischen Farben.
- `#gradient`: glatte Übergänge zwischen Farben (Standardeinstellung).
- `#blackAndWhite`: zweifarbig (schwarz und weiß).

Die vom Modifizierer `toon` und dem Shader `painter`verwendete Anzahl von Farben wird mit der Eigenschaft `colorSteps` des Modifizierers bzw. Shaders festgelegt.

Beispiel

Die folgende Anweisung setzt die Eigenschaft style des auf das Modell "Teekanne" angewendeten Modifizierers "toon" auf `#blackAndWhite`. Das Modell wird schwarzweiß (zweifarbig) wiedergegeben.

```
member("Shapes").model("Teapot").toon.style = #blackAndWhite
```

subdivision

Syntax

```
member(whichCastmember).model(whichModel).sds.subdivision
```

Beschreibung

Mit dieser 3D-Eigenschaft für den Modifizierer `sds` können Sie den Oberflächenunterteilungsmodus ermitteln und festlegen. Mögliche Werte:

- Bei Angabe von `#uniform` wird das Gitternetz einheitlich hochskaliert; jede Fläche wird gleich oft unterteilt.
- Bei Angabe von `#adaptive` werden nur dann zusätzliche Details hinzugefügt, wenn größere Ausrichtungsänderungen stattfinden, und selbst dann nur zu den gegenwärtig sichtbaren Bereichen des Gitternetzes.

Der Modifizierer `sds` kann nicht zusammen mit den Modifizierern `inker` und `toon` eingesetzt werden. Sie sollten außerdem vorsichtig sein, wenn Sie den Modifizierer `sds` in Verbindung mit dem Modifizierer `lod` verwenden. Weitere Informationen enthält der Eintrag zum Modifizierer `sds`.

Beispiel

Die folgende Anweisung setzt die Eigenschaft subdivision des Modifizierers `sds` des Modells "Baby" auf `#adaptive`. Die Geometrie von "Baby“ wird nicht einheitlich modifiziert.

```
-- Lingo
member("Scene").model(1).sds.subdivision = #adaptive

// Javascript
Member("Scene").getPropRef("model",1).sds.subdivision = symbol("adaptive");
```

Siehe auch

`sds (Modifizierer)`, `error`, `enabled (sds)`, `depth (3D)`, `tension`

subPicture

Syntax

```
-- Lingo syntax
dvdObjRef.subPicture

// JavaScript syntax
dvdObjRef.subPicture;
```

Beschreibung

Diese DVD-Eigenschaft bestimmt ggf. die aktuelle Untergrafik („subPicture“). Lesen/Schreiben.

Der Wert von `subPicture` ist eine Ganzzahl. Der Wert „0“ deaktiviert `subPicture`.

Siehe auch

DVD

subPictureCount

Syntax

```
-- Lingo syntax
dvdObjRef.subPictureCount

// JavaScript syntax
dvdObjRef.subPictureCount;
```

Beschreibung

Diese DVD-Eigenschaft gibt die Anzahl der verfügbaren Untergrafiken zurück. Nur Lesen.

Siehe auch

DVD

suspendUpdates

Syntax

```
sprite(which3dSprite).suspendUpdates
```

Beschreibung

Wenn diese 3D-Sprite-Eigenschaft auf `TRUE` gesetzt ist, wird das Sprite bei einer normalen Bildschirmaktualisierung nicht neu gezeichnet. Dadurch lässt sich die Wiedergabeleistung des Films steigern. Bestimmte Bildschirmaktualisierungen wirken sich aber nach wie vor auf das Sprite aus, z. B. wenn ein anderes Fenster über das Sprite gezogen wird. Wenn die Eigenschaft `suspendUpdates` auf `FALSE` gesetzt ist, wird das Sprite ganz normal aktualisiert.

Die Eigenschaft `suspendUpdates` muss auf `FALSE` gesetzt sein, während Elemente im 3D-Sprite animiert werden.

Beispiel

Die folgende Anweisung unterbricht die Aktualisierungen von Sprite(3).

```
-- Lingo
sprite(3).suspendUpdates = FALSE

// Javascript
sprite(3).suspendUpdates = false;
```

switchColorDepth

Syntax

```
-- Lingo syntax
_player.switchColorDepth

// JavaScript syntax
_player.switchColorDepth;
```

Beschreibung

Diese Player-Eigenschaft bestimmt, ob Director den Bildschirm, auf dem sich die Bühne befindet, auf die Farbtiefe des ladenden Filmes setzt (`TRUE`) oder die Farbtiefe des Bildschirms beibehält, wenn ein Film geladen wird (`FALSE`, Standard). Lesen/Schreiben.

Wenn `switchColorDepth` auf `TRUE` gesetzt ist, passiert nichts, bis ein neuer Film geladen ist.

Es ist empfehlenswert, die Farbtiefe des Bildschirms auf die des Films einzustellen.

- Wenn die Farbtiefe des Bildschirms niedriger eingestellt ist als die des Films, sollte sie auf die Farbtiefe des Films gesetzt werden, da auf diese Weise das beabsichtigte Aussehen des Films beibehalten wird (vorausgesetzt, der Bildschirm kann die Farbtiefe des Films wiedergeben).
- Ist die Farbtiefe des Bildschirms höher als die des Films, sollte sie reduziert werden, damit der Film bei der Wiedergabe so wenig Speicher belegt wie möglich, Darsteller effizienter geladen werden und Animationen schneller erfolgen können.

Der Wert dieser Einstellung kann auch mithilfe der Option „Bildschirm an Filmfarbtiefe anpassen“ im Dialogfeld „Allgemeine Voreinstellungen“ festgelegt werden.

Beispiel

Die folgende Anweisung setzt die Variable "switcher" auf die aktuelle Einstellung von `switchColorDepth`:

```
-- Lingo syntax
switcher = _player.switchColorDepth

// JavaScript syntax
var switcher = _player.switchColorDepth;
```

Die folgende Anweisung prüft, ob die aktuelle Farbtiefe 8 Bit ist, und schaltet die Eigenschaft `switchColorDepth` ein, wenn dies der Fall ist:

```
-- Lingo syntax
if (_system.colorDepth = 8) then
    _player.switchColorDepth = TRUE
end if

// JavaScript syntax
if (_system.colorDepth == 8) {
    _player.switchColorDepth = true;
}
```

Siehe auch

colorDepth, Player

systemTrayIcon

Syntax

```
-- Lingo syntax
_movie.displayTemplate.systemTrayIcon
windowObjRef.systemTrayIcon

// JavaScript syntax
_movie.displayTemplate.systemTrayIcon;
windowObjRef.systemTrayIcon;
```

Beschreibung

Diese Film- und Fenstereigenschaft (nur Microsoft Windows) bestimmt, ob ein Fenster ein Symbol im Infobereich der Taskleiste eines Benutzerdesktops besitzt. Lesen/Schreiben.

Wenn `systemTrayIcon` den Wert `TRUE` hat, wird ein Fenstersymbol im Infobereich platziert.

Wenn `systemTrayIcon` den Wert `FALSE` hat, wird kein Symbol im Infobereich angezeigt.

Beispiel

Die folgende Anweisung zeigt das Infobereichsymbol an, wenn der Film wiedergegeben wird.

```
-- Lingo
_movie.displayTemplate.systemTrayIcon = TRUE

// Javascript
_movie.displayTemplate.systemTrayIcon = true;
```

Siehe auch

displayTemplate, Movie, systemTrayTooltip, Window

systemTrayTooltip

Syntax

```
-- Lingo syntax
_movie.displayTemplate.systemTrayTooltip
windowObjRef.systemTrayTooltip

// JavaScript syntax
_movie.displayTemplate.systemTrayTooltip;
windowObjRef.systemTrayTooltip;
```

Beschreibung

Diese Film- und Fenstereigenschaft (nur Microsoft Windows) bestimmt die Zeichenfolge, die im QuickInfo-Popup des Infobereichsymbols angezeigt wird. Lesen/Schreiben.

Diese Eigenschaft ist nur anwendbar, wenn die Eigenschaft `systemTrayIcon` auf `TRUE` eingestellt ist. Wenn `systemTrayIcon` den Wert `TRUE` hat, wird die QuickInfo angezeigt, wenn ein Benutzer den Mauszeiger über das Infobereichsymbol bewegt.

Der Standardwert von `systemTrayTooltip` ist der Fenstertitel.

Siehe auch

`displayTemplate`, `Movie`, `systemTrayIcon`, `Window`

tabCount

Syntax

```
chunkExpression.tabCount
```

Beschreibung

Diese Textdarstellereigenschaft gibt an, wie viele eindeutige Tabulatoren der angegebene Chunk-Ausdruck des Textdarstellers enthält.

Der Wert ist eine Ganzzahl größer oder gleich 0 und kann getestet, aber nicht eingestellt werden.

Beispiel

Die folgende Anweisung zeigt die Anzahl der Registerkarten in einem Textfeld an.

```
-- Lingo
put member(1).tabCount
-- 3

// Javascript
trace(member(1).tabCount);
// 3
```

tabs

Syntax

```
member(whichTextMember).tabs
```

Beschreibung

Diese Textdarstellereigenschaft enthält eine Liste aller Tabulatoreinstellungen im Textdarsteller.

Jedes Listenelement enthält Informationen zur jeweiligen Tabulatoreinstellung für den Textdarsteller. Mögliche Eigenschaften in der Liste:

`#type`	Kann `#left`, `#center`, `#right` oder `#decimal` sein.
`#position`	Ganzzahl, die die Tabulatorposition in Punkten angibt.

Sie können diese Eigenschaft lesen und einstellen. Bei Einstellung von `tabs` ist die Eigenschaft `type` optional. Wenn `type` nicht angegeben ist, wird der Tabulatortyp standardmäßig auf `#left` gesetzt.

Beispiel

Die folgende Anweisung ruft alle Tabulatoreinstellungen des Textdarstellers „Vorspann“ ab und zeigt sie im Nachrichtenfenster an:

```
-- Lingo
put member("Intro credits").tabs
-- [[#type: #left, #position: 36], [#type: #Decimal, #position: 141], [#type: #right,
#position: 216]]

// Javascript
trace(member("Intro credits").tabs)
// < [[#type: #left, #position: 36], [#type: #Decimal, #position: 141], [#type: #right,
#position: 216]]>
```

target

Syntax

```
timeoutObject.target
```

Beschreibung

Diese Timeout-Objekteigenschaft gibt das Child-Objekt an, an das das angegebene *timeoutObject* Timeout-Ereignisse sendet. Timeout-Objekte, deren Zieleigenschaft `VOID` ist, senden ihre Ereignisse an eine Prozedur in einem Filmskript.

Diese Eigenschaft bietet sich zum Debugging von Verhalten und Parent-Skripts an, die Timeout-Objekte verwenden.

Beispiel

Die folgende Anweisung zeigt im Nachrichtenfenster den Namen des Child-Objekts an, das Timeout-Ereignisse vom Timeout-Objekt `timerOne` erhält:

```
-- Lingo
put timeout("timerOne").target

// Javascript
trace(timeout("timerOne").target)
```

Siehe auch

name (timeout), timeout(), timeoutHandler, timeoutList

targetFrameRate

Syntax

```
sprite(which3dSprite).targetFrameRate
```

Beschreibung

Diese 3D-Sprite-Eigenschaft ermittelt die bevorzugte Anzahl von Bildern pro Sekunde (BpS), die beim Rendern eines 3D-Sprites verwendet werden soll. Der Standardwert ist 30 Bilder pro Sekunde. Die Eigenschaft `targetFrameRate` ist nur dann sinnvoll, wenn die Eigenschaft `useTargetFrameRate` auf `TRUE` gesetzt ist. Ist die Eigenschaft `useTargetFrameRate` auf `TRUE` gesetzt, wird die Polygonanzahl der Modelle im Sprite entsprechend der angegebenen Zielbildrate reduziert.

Beispiel

Die folgenden Anweisungen setzen die Eigenschaft `targetFrameRate` von Sprite 3 auf 45 und sorgen dafür, dass die Bildrate verwendet wird (`useTargetFrameRate` = TRUE):

```
-- Lingo
sprite(3).targetFrameRate = 45
sprite(3).useTargetFrameRate = TRUE

// Javascript
sprite(3).targetFrameRate = 45;
sprite(3).useTargetFrameRate = true;
```

Siehe auch

useTargetFrameRate

tension

Syntax

```
member(whichCastmember).model(whichModel).sds.tension
```

Beschreibung

Mit dieser 3D-Oberflächenunterteilungseigenschaft können Sie angeben, inwieweit die neu generierte Oberfläche der ursprünglichen Oberfläche entspricht (Prozentsatz als Fließkommawert zwischen 0.0 und 100.0). Je höher der Wert, desto mehr entspricht die unterteilte Oberfläche der ursprünglichen Oberfläche. Der Standardwert ist 65.0.

Beispiel

Die folgende Anweisung setzt die Eigenschaft „tension“ des Modifizierers „sds“ für das Modell „Baby“ auf 35. Wenn die Einstellung „error“ des Modifizierers „sds“ niedrig und die Einstellung „depth“ hoch ist, ergibt sich eine starke Auswirkung auf die Geometrie von „Baby“.

```
-- Lingo
member("scene").model("Baby").sds.tension = 35

// Javascript
member("scene").getPropRef("model",1).sds.tension = 35;
```

Siehe auch

sds (Modifizierer), error, depth (3D)

text

Syntax

```
-- Lingo syntax
memberObjRef.text

// JavaScript syntax
memberObjRef.text;
```

Beschreibung

Diese Textdarstellereigenschaft bestimmt den String in dem in *whichCastMember* angegebenen Textdarsteller.

Die Darstellereigenschaft `text` eignet sich zur Anzeige von Nachrichten und zur Aufzeichnung von Benutzereingaben.

Diese Eigenschaft kann getestet und eingestellt werden.

Wenn Sie mit Lingo den gesamten Text eines Darstellers ändern, wird die gesamte Formatierung entfernt, die Sie auf einzelne Wörter oder Zeilen angewendet haben. Nach Änderung der Darstellereigenschaft `text` wird erneut die globale Formatierung verwendet. Um nur bestimmte Teile des Textes zu ändern, referenzieren Sie die gewünschten Zeilen, Wörter oder Textelemente.

Wenn der Film als Applet wiedergegeben wird, lautet der Wert dieser Eigenschaft `""` (leerer String) für einen Felddarsteller, dessen Text noch nicht gestreamt wurde.

Ein Beispiel für `text` in einem fertigen Film finden Sie im Film „Forms and Post“ im Ordner Learning\\Lingo_Examples im Director-Anwendungsordner.

Beispiel

Die folgende Anweisung setzt den Satz „Danke schön“ in den leeren Darsteller „Antwort“:

```
--Lingo syntax
if (member("Response").text = EMPTY) then
    member("Response").text = "Thank You."
end if

// JavaScript syntax
if (member("Response").text == " ") {
    member("Response").text = "Thank You.";
}
```

Die folgende Anweisung setzt den Inhalt des Darstellers "Hinweis" auf "Sie haben die richtige Entscheidung getroffen!".

```
--Lingo syntax
member("Notice").text = "You have made the right decision!"

// JavaScript syntax
member("Notice").text = "You have made the right decision!";
```

Siehe auch

selEnd, selStart

texture

Syntax

```
member(whichCastmember).texture(whichTexture)
member(whichCastmember).texture[index]
member(whichCastmember).shader(whichShader).texture
member(whichCastmember).model(whichModel).shader.texture
member(whichCastmember).model(whichModel).shaderList.texture
member(whichCastmember).model(whichModel).shaderList[index].texture
member(whichCastmember).modelResource(whichParticleSystemModelResource).texture
```

Beschreibung

Diese 3D-Element- und -Shader-Eigenschaft gibt ein Grafikobjekt an, anhand dessen ein Shader das Aussehen der Modelloberfläche definiert. Die Grafik wird durch den Shader um die Geometrie des Modells gewickelt.

Die sichtbare Komponente eines Shaders besteht aus bis zu acht Texturebenen. Diese acht Texturebenen werden wiederum in Director aus Bitmapdarstellern oder Grafikobjekten erstellt oder zusammen mit Modellen aus 3D-Modellierungsprogrammen importiert.

Texturen werden mit den Befehlen `newTexture()` und `deleteTexture()` erstellt und gelöscht.

Texturen werden in der Texturpalette des 3D-Darstellers gespeichert. Auf sie kann nach Namen (*whichTexture*) oder Palettenindex (*textureIndex*) Bezug genommen werden. Eine Textur kann von beliebig vielen Shadern benutzt werden. Texturänderungen erscheinen in allen Shadern, die die jeweilige Textur verwenden.

Es gibt drei Arten von Texturen:

`#fromCastmember`: Die Textur wird mit dem Befehl `newTexture()` aus einem Bitmapdarsteller erstellt.

`#fromImageObject`: Die Textur wird mit dem Befehl `newTexture()` aus einem Lingo-Grafikobjekt erstellt.

`#importedFromFile`: Die Textur wird zusammen mit einem Modell aus einem 3D-Modellierungsprogramm importiert.

Weitere Informationen zu Textureigenschaften finden Sie in den Themen des Director-Benutzerhandbuchs im Director-Hilfefenster.

Die Textur eines Partikelsystems ist eine Eigenschaft der Modellressource vom Typ `#particle`.

Beispiel

Die folgende Anweisung setzt die Eigenschaft texture des Shaders `WallSurface` auf die Textur `BluePaint`:

```
-- Lingo
member("Room").shader("WallSurface").texture = member("Room").texture("BluePaint")

// Javascript
member("Room").getPropRef("shader",1).texture = member("Room").getPropRef("texture",1);
```

Siehe auch

`newTexture`, `deleteTexture`

textureCoordinateList

Syntax

```
member(whichCastmember).modelResource(whichmodelResource).
textureCoordinateList
modelResourceObjectReference.textureCoordinateList
```

Beschreibung

Mit dieser 3D-Eigenschaft können Sie bei einer Modellressource vom Typ `#mesh` oder bei einem Modell, an dem der Modifizierer `meshDeform` angebracht ist, die Eigenschaft `textureCoordinateList` der Modellressource ermitteln und festlegen.

Die Eigenschaft `textureCoordinateList` ist eine Liste mit Unterlisten, die Positionen in einem Bild angeben, welche beim Texture Mapping eines Dreiecks verwendet werden. Jede Unterliste besteht aus zwei Werten, die eine Position in einer Texture Map angeben. Die Werte müssen zwischen 0.0 und 1.0 liegen, damit sie auf Texture Maps mit willkürlicher Größe zugeschnitten werden können. Der Standardwert ist eine leere Liste.

Manipulieren Sie `modelResource.face[index].textureCoordinates` oder `model.meshdeform.mesh[index].face[index]`, um die Zuordnung zwischen Texturkoordinaten (`textureCoordinates`) und den Ecken einer Gitternetzfläche zu ändern.

Beispiel

```
-- Lingo
put member(5).modelResource("mesh square").textureCoordinateList
--[ [0.1, 0.1], [0.2, 0.1], [0.3, 0.1], [0.1, 0.2], [0.2, 0.2], [0.3, 0.2], [0.1, 0.3], [0.2,
0.3], [0.3, 0.3] ]

// Javascript
trace(member(5).getPropRef("modelResource",1).textureCoordinateList;
// <[ [0.1, 0.1], [0.2, 0.1], [0.3, 0.1], [0.1, 0.2], [0.2, 0.2], [0.3, 0.2], [0.1, 0.3], [0.2,
0.3], [0.3, 0.3] ]>
```

Siehe auch

`face`, `texture`, `meshDeform (Modifizierer)`

textureCoordinates

Syntax

```
member(whichCastmember).modelResource(whichModelResource).face[faceIndex].textureCoordinates
modelResourceObject.face[faceIndex].textureCoordinates
```

Beschreibung

Diese 3D-Eigenschaft gibt an, welche Elemente in der `textureCoordinateList` für die *faceIndex*-Seite verwendet werden sollen. Diese Eigenschaft muss eine Liste aus drei Ganzzahlen sein, die Indizes in der `textureCoordinateList` angeben, welche den Texturkoordinaten (`textureCoordinates`) für die einzelnen Ecken der Gitternetzfläche entsprechen.

Siehe auch

`face`, `textureCoordinateList`

textureLayer

Syntax

```
member(whichCastmember).model(whichModel).meshDeform.mesh[index].textureLayer.count
member(whichCastmember).model(whichModel).meshdeform.mesh[index].texturelayer.add()
member(whichCastmember).model(whichModel).meshdeform.mesh[index].texturelayer[index].texture
CoordinateList.
```

Beschreibung

Mit diesen 3D-Eigenschaften für den Modifizierer `meshdeform` können Sie Informationen über die Texturebenen eines bestimmten Gitternetzes abrufen und festlegen.

Für einen Shader kann es bis zu acht Texturebenen geben; jede Ebene kann zwar nur eine Textur enthalten, doch eine Textur kann für mehrere Ebenen verwendet werden. Texturebenen sind Ebenen mit von Shadern verwendeten Texturen.

Mit den folgenden Eigenschaften können Sie auf Texturebenen zugreifen und sie manipulieren:

`meshdeform.mesh[index].texturelayer.count` gibt die Anzahl von Texturebenen für das angegebene Gitternetz zurück.

`model.meshdeform.mesh[index].texturelayer.add()` fügt eine leere Texturebene zum angegebenen Gitternetz hinzu.

Mit `model.meshdeform.mesh[index].texturelayer[index].texturecoordinatelist` können Sie eine Liste von `textureCoordinates` für eine bestimmte Ebene des angegebenen Gitternetzes festlegen oder abrufen. Mit dieser Eigenschaft können Sie außerdem wie folgt Texturkoordinaten zwischen Texturebenen kopieren:

```
model.meshdeform.texturelayer[a].texturecoordinatelist =
model.meshdeform.texturelayer[b].texturecoordinatelist
```

Siehe auch

`meshDeform (Modifizierer)`, `mesh (Eigenschaft)`, `textureCoordinateList`, `add (3D-Textur)`, `count`, `texture`, `textureModeList`

textureList

Syntax

```
member(whichMember).model(whichModel).shader(whichShader).textureList
member(whichMember).model(whichModel).shader(whichShader).textureList[index]
```

Beschreibung

Diese 3D-Shader-Eigenschaft bestimmt die Liste der auf den Shader angewendeten Texturen. Ein Shader kann bis zu acht Texturebenen verwenden. Beim Testen gibt diese Eigenschaft eine lineare Liste mit Texturobjekten zurück. Wenn diese Eigenschaft ohne Angabe eines Index eingestellt wird, gibt sie ein Texturobjekt an, das auf alle Ebenen angewendet werden soll. Wenn Sie die Eigenschaft `textureList` auf `VOID` setzen, wird die Texturfunktion für alle Ebenen deaktiviert. Der Standardwert ist `VOID`.

Um das Texturobjekt für eine bestimmte Texturebene zu testen oder festzulegen, geben Sie einen Indexwert an.

Beispiel

Die folgende Anweisung setzt die dritte Texturebene des Shaders `WallSurface` auf die Textur `BluePaint` im Darsteller „Raum“:

```
-- Lingo
member(3).model("Car").shader("WallSurface").textureList[3] =
member("Room").texture("BluePaint")

// Javascript
member(3).getPropRef("model",1).getPropRef("shader",1).textureList[3] =
member("Room").getPropRef("texture",1);
```

Siehe auch

`textureModeList`

textureMember

Syntax

```
member(whichCastmember).textureMember
```

Beschreibung

Diese 3D-Darstellereigenschaft gibt den Namen des Bitmapdarstellers an, der als Quelle für die Standardtextur des 3D-Darstellers verwendet werden soll.

Die 3D-Darstellereigenschaft `textureType` muss auf `#member` gesetzt sein, damit die Eigenschaft `textureMember` in Kraft tritt.

Beispiel

Die folgende Anweisung setzt die Eigenschaft `textureMember` des Darstellers `YardScene` auf "Zaun". Wenn die Eigenschaft `textureType` von `YardScene` auf `#member` gesetzt ist, wird der Darsteller „Zaun“ als Quellbitmap für die Standardtextur in `YardScene` verwendet.

```
-- Lingo
member("YardScene").textureMember = "Fence"

// Javascript
member("YardScene").textureMember = "Fence";
```

Siehe auch

textureType

textureMode

Syntax

```
member(whichCastmember).shader(whichShader).textureMode
member(whichCastmember).model(whichModel).shader.textureMode
member(whichCastmember).model(whichModel).shaderList{[index]}.textureMode
```

Beschreibung

Diese 3D-Eigenschaft für Shader vom Typ `#standard` gibt an, wie die erste Texturebene der Modelloberfläche zugeordnet wird. Mit der Eigenschaft `textureModeList` können Sie Texturen für andere Ebenen als die erste angeben. Diese Eigenschaft wird ignoriert, wenn der Modifizierer `#toon` auf die Modellressource angewendet wird.

Mögliche Werte für diese Eigenschaft: `#none`, `#wrapPlanar`, `#wrapCylindrical`, `#wrapSpherical`, `#reflection`, `#diffuseLight` und `#specularLight`. Eine Beschreibung dieser Elemente finden Sie unter textureModeList.

Beispiel

Die folgende Anweisung setzt den Wert der Eigenschaft `textureMode` der ersten Texturebene des vom Modell "Ball" verwendeten Shaders auf `#wrapSpherical`:

```
-- Lingo
member("scene").model("Ball").shader.textureMode = #wrapSpherical

// Javascript
member("scene").getPropRef("model",1).getProp("shader").textureMode =
symbol("wrapSpherical");
```

Siehe auch

textureModeList

textureModeList

Syntax

```
member(whichCastmember).shader(whichShader).textureModeList
member(whichCastmember).shader(whichShader).
textureModeList[textureLayerIndex]
member(whichCastmember).model(whichModel).shader.textureModeList
member(whichCastmember).model(whichModel).shader.
textureModeList[textureLayerndex]
```

Beschreibung

Mit dieser 3D-Eigenschaft für Standard-Shader können Sie ändern, wie eine `textureLayer` zur Modelloberfläche zugeordnet (gemappt) wird. Diese Eigenschaft wird ignoriert, wenn der Modifizierer `#toon` auf die Modellressource angewendet wird. Mögliche Werte:

- `#none` verwendet die ursprünglich für die Modellressource definierten Texturkoordinaten. Diese Einstellung deaktiviert `wrapTransform and wrapTransformList[textureLayerIndex]`.
- `#wrapPlanar` projiziert die Textur so auf die Modelloberfläche, als würde sie von einem Overhead-Projektor projiziert. Die `wrapTransformList[textureLayerIndex]` des Shaders wird auf den Mapping-Raum angewendet, bevor die Texturkoordinaten im Modellraum generiert werden. Bei einer Identitäts-`wrapTransformList[textureLayerIndex]` (Standard) erfolgt die Ausrichtung des Planar Mapping so, dass die Textur entlang der Z-Achse extrudiert wird und die Oberseite der Textur entlang der Y-Achse verläuft.
- `#wrapCylindrical` wickelt die Textur so um die Oberfläche, als befände sich die Oberfläche in der Mitte der Textur und die Textur würde dann um die Oberfläche gerollt, damit ein Zylinder entsteht. Bevor die Texturkoordinaten im Modellraum erzeugt werden, wird `wrapTransformList[textureLayerIndex]` auf den Mapping-Raum angewendet. Bei einer Identitäts-`wrapTransformList[textureLayerIndex]` (Standard) erfolgt die Ausrichtung des Cylindrical Mapping so, dass die Textur beginnend mit der linken Kante von der -Y-Achse zur +X-Achse hin um die Z-Achse gewickelt wird. Die Oberseite der Textur in Richtung der positiven Z-Achse.
- `#wrapSpherical` wickelt die Textur so um die Oberfläche, als befände sich die Oberfläche in der Mitte der Textur und als würden anschließend alle vier Ecken der Textur nach oben gezogen, bis sie oben aufeinanderstoßen. Bevor die Texturkoordinaten im Modellraum erzeugt werden, wird `wrapTransformList[textureLayerIndex]` auf den Mapping-Raum angewendet. Bei einer Identitäts-`wrapTransformList[textureLayerIndex]` erfolgt das Spherical Mapping am Ursprung des Modellraums und ist so ausgerichtet, dass die Textur beginnend mit der linken Kante von der -Y-Achse zur +X-Achse hin um die Z-Achse gewickelt wird. Die Oberseite der Textur in Richtung der positiven Z-Achse.
- `#reflection` ist ähnlich wie `#wrapSpherical`, hierbei werden jedoch die neuen Texturkoordinaten von einer festen Ausrichtung aus kontinuierlich auf die Oberfläche projiziert. Wenn sich das Modell dreht, drehen sich die Texturkoordinaten nicht mit. Das Reflection Mapping simuliert Licht, das von der Umgebung auf ein Objekt reflektiert wird. Diese Einstellung deaktiviert `wrapTransform`.
- `#diffuseLight` generiert Texturkoordinatenwerte für das Streulicht-Mapping (einen pro Scheitelpunkt) und speichert die Ergebnisse im referenzierten Gitternetz. Diese Einstellung deaktiviert `wrapTransform`.
- `#specularLight` generiert Texturkoordinatenwerte für das Glanzlicht-Mapping (einen pro Scheitelpunkt) und speichert die Ergebnisse im referenzierten Gitternetz. Diese Einstellung deaktiviert `wrapTransform`.

Beispiel

In diesem Beispiel wird ein Shader eingerichtet, der eine glänzende Kugel simuliert. Die erste `textureLayer` des Shaders verwendet Spherical Mapping („#wrapSpherical“) und die dritte Texturebene `textureLayer` Reflection-Mapping (`#refection`). Der dritte Eintrag in der Texturliste des Shaders (`textureList[3]`) wird von der Umgebung auf alle Modelle reflektiert, die diesen Shader benutzen.

```
-- Lingo
member("scene").shader("GardenBall).textureList[1] =
member("scene").texture("FlatShinyBall")
member("scene").shader("GardenBall").textureModeList[1] = #wrapSpherical
member("scene").shader("GardenBall").textureList[3] =
member("scene").texture("GardenEnvironment")
member("scene").shader("GardenBall").textureModeList[3] = #reflection

// Javascript
member("scene").getPropRef("shader",1).textureList[1] =
member("scene").getPropRef("texture",1);
member("scene").getPropRef("shader",1).textureModeList[1] = symbol("wrapSpherical");
member("scene").getPropRef("shader",1).textureList[3] =
member("scene").getPropRef("texture",1);
member("scene").getPropRef("shader",1).textureModeList[3] = symbol("reflection");
```

Siehe auch

textureTransformList, wrapTransform

textureRenderFormat

Syntax

```
getRenderServices().textureRenderFormat
```

Beschreibung

Mit dieser 3D-Eigenschaft für `rendererServices` können Sie das Standardbitformat für alle Texturen in allen 3D-Darstellern ermitteln und festlegen. Verwenden der Textur einer TexturMit der Eigenschaft `RenderFormat` lässt sich diese Einstellung für bestimmte Texturen außer Kraft setzen. Kleinere Bitformate (d.h. 16-Bit-Varianten wie `#rgba5551`) belegen weniger Videospeicher im Hardwarebeschleuniger, sodass Sie mehr Texturen nutzen können, bevor Sie auf den Software-Renderer umschalten müssen. Größere Bitformate (d.h. 32-Bit-Varianten wie `#rgba8888`) sehen in der Regel besser aus. Damit Sie die Alphatransparenz in einer Textur verwenden können, darf das letzte Bit nicht Null sein. Um glatte Transparenzverläufe zu erzielen, muss die Präzision des Alphakanals über 1 Bit liegen.

Pixelformate bestehen aus vier Ziffern – jede Ziffer gibt den Genauigkeitsgrad für Rot, Grün, Blau und Alpha an. Der von Ihnen gewählte Wert bestimmt die Farbtreue (Genauigkeit des Alphakanals) und den vom Hardwaretexturpuffer belegten Speicherplatz. Sie können einen Wert wählen, der die Farbtreue verbessert, oder einen Wert, der es erlaubt, mehr Texturen auf die Videokarte zu laden. Beachten Sie hierbei, dass 16-Bit-Texturen nur halb so viel Speicherplatz beanspruchen wie 32-Bit-Texturen. Wenn ein Film versucht, mehr Texturen zu verwenden als jeweils auf der Karte gespeichert werden können, schaltet Director auf Software-Rendering (`#software`) um.

Mögliche Werte für `textureRenderFormat`:

- `#rgba8888`: 32-Bit-Farbmodus mit je 8 Bits für Rot, Grün, Blau und Alpha.
- `#rgba8880`: wie oben, aber ohne Alphawert.
- `#rgba5650`: 16-Bit-Farbmodus ohne Alpha; 5 Bits für Rot, 6 für Grün, 5 für Blau.
- `#rgba5550`: 16-Bit-Farbmodus ohne Alpha; je 5 Bits für Rot, Grün und Blau, ohne Alphamessung.
- `#rgba5551`: 16-Bit-Farbmodus mit je 5 Bits für Rot, Grün und Blau und 1 Bit für Alpha.
- `#rgba4444`: 16-Bit-Farbmodus mit je 4 Bits für Rot, Grün, Blau und Alpha.

Standardwert: `#rgba5551`.

Beispiel

Die folgende Anweisung setzt die globale Eigenschaft `textureRenderFormat` für den 3D-Darsteller auf `#rgba8888`. Jede Textur dieses Films wird in 32-Bit-Farbe gerendert, falls es sich nicht um eine Textur handelt.Die Eigenschaft `renderFormat` wird auf einen anderen Wert als `#default` gesetzt.

```
-- Lingo
getRendererServices().textureRenderFormat = #rgba8888

// Javascript
getRendererServices().textureRenderFormat = symbol("rgba8888");
```

Siehe auch

`renderer`, `preferred3dRenderer`, `renderFormat`, `getRendererServices()`

textureRepeat

Syntax

```
member(whichCastmember).shader(whichShader).textureRepeat
member(whichCastmember).model(whichModel).shader.textureRepeat
member(whichCastmember).model(whichModel).shaderList{[index]}.textureRepeat
```

Beschreibung

Diese 3D-Eigenschaft für Shader vom Typ `#standard` steuert das Texture-Clamping-Verhalten der ersten Texturebene des Shaders. Mit der Eigenschaft `textureRepeatList` können Sie diese Eigenschaft für andere Texturebenen als die erste steuern.

Wenn `textureRepeat` auf `TRUE` gesetzt wird und der Wert der x- und/oder y-Komponente von `shaderReference.textureTransform.scale` unter 1 liegt, wird die Textur über die Modelloberfläche hinweg kachelartig wiederholt.

Wenn `textureRepeat` auf `FALSE` gesetzt ist, wird die Textur nicht wie eine Kachel verlegt. Wenn der Wert der x- und/oder y-Komponente von `shaderReference.textureTransform.scale` kleiner als 1 ist, erscheinen alle Bereiche des Modells, die nicht von der Textur bedeckt sind, schwarz. Wenn der Wert der x- und/oder y-Komponente von `shaderReference.textureTransform.scale` größer als 1 ist, wird die Textur dort abgeschnitten, wo sie über den Texturkoordinatenbereich hinaus ragt.

Der Standardwert dieser Eigenschaft lautet `TRUE`. Diese Eigenschaft gibt bei Verwendung des Software-Renderers (`#software`) immer den Wert `TRUE` zurück.

Beispiel

Die folgende Anweisung setzt die Eigenschaft `textureRepeat` des ersten vom Modell `gbCyl3` verwendeten Shaders auf TRUE. Die erste Textur in diesem Shader wird kachelartig wiederholt, wenn der Wert der x- oder y-Komponente der zugehörigen Eigenschaft `textureTransform` oder `textureTransformList` kleiner als 1 ist.

```
-- Lingo
member("scene").model("gbCyl3").shader.textureRepeat = TRUE

// Javascript
member("scene").getPropRef("model",1).getProp("shader").textureRepeat = true;
```

Siehe auch

textureTransform, textureTransformList

textureRepeatList

Syntax

```
shaderReference.textureRepeatList[textureLayerIndex]
member(whichCastmember).shader(whichShader).textureRepeatList[textureLayerIndex]
member(whichCastmember).shader[shaderListIndex].textureRepeatList[textureLayerIndex]
member(whichCastmember).model(whichModel).shader.textureRepeatList[textureLayerIndex]
member(whichCastmember).model(whichModel).shaderList[shaderListIndex].
textureRepeatList[textureLayerIndex]
```

Beschreibung

Mit dieser 3D-Eigenschaft für Standard-Shader können Sie das Texture-Clamping-Verhalten einer beliebigen Texturebene ermitteln und festlegen. Wenn diese Eigenschaft auf `TRUE` gesetzt ist (Standardeinstellung), kann die Textur in `textureLayerIndex` über Modelloberflächen hinweg mehrmals kachelartig wiederholt werden. Um dies zu erreichen, setzen Sie `shaderReference.textureTransform[textureLayerIndex].scale` in der x- oder y-Dimension auf einen Wert kleiner als 1. Ist die Eigenschaft "textureRepeatList" auf "FALSE" gesetzt, wird die Textur auf einen kleineren Teil der Modelloberflächen angewendet und nicht kachelartig über diese Oberflächen hinweg verlegt, sofern der Wert von `shaderReference.textureTransform[textureLayerIndex].scale` in der x- oder y-Dimension kleiner als 1 ist. Dies kann man sich als Schrumpfen des Quellbildes im Rahmen des Originalbilds und Auffüllen der Lücken mit Schwarz vorstellen. Wenn analog dazu `shaderReference.textureTransform[textureLayerIndex].scale` in der x- oder y-Dimension auf einen Wert größer als 1 gesetzt ist, wird das Bild abgeschnitten, da der Texturrand über den Texturkoordinatenbereich hinaus ragt.

Beispiel

Der folgende Code bedeckt eine Kugel komplett mit der Granittextur `textureMap`, die viermal über die Oberfläche hinweg wiederholt wird, und einem Logo, das ein Viertel der Oberfläche belegt:

```
-- Lingo
m = member(2).model("mySphere")
f = member(2).newTexture("granite", #fromCastmember, member("granite"))
g = member(2).newTexture("logo", #fromCastmember, member("logo"))
s = member(2).newShader("s", #standard)
s.textureList[1] = g
s.textureList[2] = f
s.textureRepeatList[2] = false
s.textureRepeatList[1] = true
s.textureTransformList[1].scale(0.5,0.5,1.0)
s.textureTransformList[2].scale(0.5,0.5,1.0)
s.textureModeList[2] = #wrapPlanar
s.blendFunctionList[2] = #add
m.shaderList = s

// Javascript
m = member(2).getPropRef("model","1");
f = member(2).newTexture("granite", symbol("fromCastmember"), member("granite"));
g = member(2).newTexture("logo", symbol("fromCastmember"), member("logo"));
s = member(2).newShader("s", symbol("standard"));
s.textureList[1] = g;
s.textureList[2] = f;
s.textureRepeatList[2] = false;
s.textureRepeatList[1] = true;
s.textureTransformList[1].scale(0.5,0.5,1.0);
s.textureTransformList[2].scale(0.5,0.5,1.0);
s.textureModeList[2] = symbol("wrapPlanar");
s.blendFunctionList[2] = symbol("add");
m.shaderList = s;
```

textureTransform

Syntax

```
member(whichCastmember).shader(whichShader).textureTransform
member(whichCastmember).model(whichModel).shader.textureTransform
member(whichCastmember).model(whichModel).shaderList{[index]}.textureTransform
```

Beschreibung

Über diese 3D-Eigenschaft für Shader vom Typ `#standard` können Sie auf eine Transformation zugreifen, die die Texturkoordinatenzuordnung der ersten Texturebene des Shaders modifiziert. Manipulieren Sie diese Transformation, um die Textur vor Übertragung auf die Modelloberfläche kachelartig zu wiederholen, zu drehen oder zu verschieben. Die Textur selbst bleibt unverändert; die Transformation ändert lediglich die Art und Weise, auf die der Shader die Textur anwendet. Die Eigenschaft `textureTransform` wird unabhängig von der Eigenschaft `textureMode` auf alle Texturkoordinaten angewendet. Dies ist die letzte Modifizierung der Texturkoordinaten, bevor sie an den Renderer gesendet werden. Die Eigenschaft `textureTransform` ist eine Matrix, die sich auf die Textur im `textureImage`-Raum auswirkt.Der `TextureImage`-Raum existiert nur auf der X,Y-Ebene.

Um das Bild zweimal entlang seiner horizontalen Achse zu wiederholen, verwenden Sie `shaderReference.textureTransform.scale(0.5, 1.0, 1.0)`. Die Skalierung auf der Z-Achse wird ignoriert.

Um das Bild um `point(xOffset,yOffset)` zu verschieben, verwenden Sie `shaderReference.textureTransform.translate(xOffset,yOffset,0.0)`. Eine Verschiebung um Ganzzahlen hat keine Wirkung, wenn die Shader-Eigenschaft `textureRepeat` auf `TRUE` gesetzt ist, da die Breite und Höhe der Textur in diesem Fall einen Wert zwischen0.0 und1.0 aufweist.

Um eine Drehung auf eine Texturebene anzuwenden, verwenden Sie `shaderReference.textureTransform.rotate(0,0,angle)`. Drehungen um die Z-Achse erfolgen um den 2D-Bildpunkt (0,0), wodurch die obere linke Ecke der Textur gemappt wird. Drehungen um die X- und Y-Achse werden ignoriert.

Wie bei der Transformation eines Modells werden `textureTransform`-Änderungen auf alle betroffenen Ebenen übertragen. Wenn die Textur nicht um point(0,0), sondern um point(xOffset,yOffset) gedreht werden soll, führen Sie zuerst eine Translation zu point(0 - xOffset, 0 - yOffset), dann eine Drehung und schließlich eine Translation zu point(xOffset, yOffset) durch. Die Eigenschaft `textureTransform` ist der Shader-Eigenschaft `wrapTransform` sehr ähnlich, wird aber im 2D-Bildraum angewendet, nicht im 3D-Weltraum. Aus diesem Grund sind nur Drehungen um die Z-Achse und Translationen und Skalierungen auf der X- und Y-Achse möglich. Die Transformation wird unabhängig von der Einstellung `shaderReference.textureMode` angewendet. Im Gegensatz dazu hat `wrapTransform` nur dann eine Wirkung, wenn `textureMode` auf `#wrapPlanar`, `#wrapCylindrical` oder `#wrapSpherical` gesetzt ist.

Beispiel

Die folgende Anweisung zeigt die `textureTransform` der ersten Textur im ersten vom Modell `gbCyl3` verwendeten Shader an:

```
-- Lingo
put member("Scene").model("gbCyl3").shader.textureTransform
-- transform(1.0000, 0.0000, 0.0000,0.0000, 0.0000, 1.0000, 0.0000, 0.0000, 0.0000, 0.0000,
1.0000, 0.0000, 0.0000, 0.0000, 0.0000, 1.0000)

// Javascript
trace(member("Scene").getPropRef("model",1).getProp("shader").textureTransform);
// <transform(1.0000, 0.0000, 0.0000,0.0000, 0.0000, 1.0000, 0.0000, 0.0000, 0.0000, 0.0000,
1.0000, 0.0000, 0.0000, 0.0000, 0.0000, 1.0000)>
```

Die folgende Anweisung halbiert die Höhe und Breite der ersten vom Shader `gbCyl3` verwendeten Textur. Wenn die Eigenschaft `textureRepeat` von `gbCyl3` auf TRUE gesetzt ist, werden vier Kopien der Textur kachelartig über den Shader hinweg verlegt.

```
-- Lingo
member("Scene").shader("gbCyl3").textureTransform.scale = vector(0.5, 0.5, 1)

// Javascript
member("Scene").getPropRef("shader",1).textureTransform.scale = vector(0.5,0.5,1);
```

Die folgende Anweisung dreht die erste vom Shader `gbCyl3` verwendete Textur um 90° von vector(0, 0, 0) aus:

```
-- Lingo
member("Scene").shader("gbCyl3").textureTransform.rotation = vector(0, 0, 90)

// Javascript
member("Scene").getPropRef("shader",1).textureTransform.rotation = vector(0,0,90);
```

textureTransformList

Syntax

```
shaderReference .textureTransformList[textureLayerIndex]
member(whichCastmember).shader(ShaderName).textureTransformList[textureLayerIndex]
member(whichCastmember).shader[shaderListIndex].textureTransformList[textureLayerIndex]
member(whichCastmember).model(modelName).shader.textureTransformList[textureLayerIndex]
member(whichCastmember).model(modelName).shaderList[shaderListIndex].
textureTransformList[textureLayerIndex]
```

Beschreibung

Über diese 3D-Eigenschaft für Standard-Shader können Sie auf eine Transformation zugreifen, die die Texturkoordinatenzuordnung einer Texturebene modifiziert. Manipulieren Sie diese Transformation, um die Textur vor Übertragung auf die Modelloberfläche kachelartig zu wiederholen, zu drehen oder zu verschieben. Die Textur selbst bleibt unverändert; die Transformation ändert lediglich die Art und Weise, auf die der Shader die Textur anwendet.

Um die Grafik zweimal entlang ihrer horizontalen Achse zu wiederholen, verwenden Sie `textureTransformList[whichTextureLayer].scale(0.5, 1.0, 1.0)`. Skalierungen auf der Z-Achse werden ignoriert, da für Texturen 2D-Grafiken verwendet werden. Vermeiden Sie den Skalierungswert 0.0 (selbst auf der Z-Achse), da hierdurch der gesamte Textureffekt negiert wird.

Um das Bild um point(xOffset,yOffset) zu verschieben, verwenden Sie `textureTransformList[whichTextureLayer].translate(xOffset,yOffset,0.0)`. Eine Verschiebung um Ganzzahlen hat keine Wirkung, wenn die Eigenschaft `textureRepeat` der Texturebene auf `TRUE` gesetzt ist, da die Breite und Höhe der Textur in diesem Fall einen Wert zwischen 0.0 und 1.0 aufweist.

Um eine Drehung auf eine Texturebene anzuwenden, verwenden Sie `textureTransformList[whichTextureLayer].rotate(0,0,angle)`. Drehungen um die Z-Achse erfolgen um den 2D-Bildpunkt (0,0), wodurch die obere linke Ecke der Textur gemappt wird. Drehungen um die X- und Y-Achse werden ignoriert, da für Texturen 2D-Grafiken verwendet werden.

Wie bei der Transformation eines Modells werden `textureTransform`-Änderungen auf alle betroffenen Ebenen übertragen. Wenn das Bild nicht um point(0,0), sondern um point(xOffset,yOffset) gedreht werden soll, führen Sie zuerst eine Translation zu point(0 - xOffset, 0 - yOffset), dann eine Drehung und schließlich eine Translation zu point(xOffset, yOffset) durch.

Die Eigenschaft `textureTransformList` ist der Shader-Eigenschaft `wrapTransformList` sehr ähnlich,

wird aber im 2D-Bildraum angewendet, nicht im 3D-Weltraum. Aus diesem Grund sind nur Drehungen um die Z-Achse und Translationen und Skalierungen auf der X- und Y-Achse möglich.

Die Transformation wird unabhängig von der Einstellung `shaderReference.textureModeList[index]` angewendet. Im Gegensatz dazu hat `wrapTransform` nur dann eine Wirkung, wenn `textureMode` auf `#wrapPlanar`, `#wrapCylindrical` oder `#wrapSpherical` gesetzt ist.

Beispiel

Die folgende Anweisung zeigt die `textureTransform` der dritten Textur im ersten vom Modell `gbCyl3` verwendeten Shader:

```
-- Lingo
put member("scene").model("gbCyl3").shader.textureTransformList[3]
-- transform(1.0000, 0.0000, 0.0000, 0.0000, 0.0000, 1.0000, 0.0000, 0.0000, 0.0000, 0.0000,
1.0000, 0.0000, 0.0000, 0.0000, 0.0000, 1.0000)

// Javascript
trace(member("scene").getPropRef("model",1).getProp("shader").textureTransformList[3])
//< transform(1.0000, 0.0000, 0.0000, 0.0000, 0.0000, 1.0000, 0.0000, 0.0000, 0.0000, 0.0000,
1.0000, 0.0000, 0.0000, 0.0000, 0.0000, 1.0000)>
```

Die folgende Anweisung halbiert die Höhe und Breite der fünften vom Shader `gbCyl3` verwendeten Textur. Wenn der Wert `textureRepeatList[5]` von `gbCyl3` auf TRUE gesetzt ist, werden vier Kopien der Textur kachelartig über den Shader hinweg verlegt.

```
-- Lingo
member("scene").shader("gbCyl3").textureTransformList[5].scale = vector(0.5, 0.5, 1)

// Javascript
member("scene").getPropRef("shader","1").textureTransformList[5].scale = vector(0.5, 0.5, 1);
```

Die folgende Anweisung dreht die vierte vom Shader `gbCyl3` verwendete Textur um 90° von vector(0, 0, 0) aus:

```
-- Lingo
member("scene").shader("gbCyl3").textureTransformList[4].rotation = vector(0, 0, 90)

// Javascript
member("scene").getPropRef("shader","1 ").textureTransformList[4].rotation = vector(0, 0, 90);
```

Die folgenden Anweisungen drehen die dritte vom Shader `gbCyl3` verwendete Textur um 90° um seinen Mittelpunkt, sofern es sich bei `textureList[3]` um eine 128x128 große Textur handelt:

```
-- Lingo
s = member("scene").shader("gbCyl3")
s.textureTransformList[3].translate(-64,-64,0)
s.textureTransformList[3].rotate(0,0,90)
s.textureTransformList[3].translate(64,64,0)

// Javascript
s = member("scene").getPropRef("shader","1");
s.textureTransformList[3].translate(-64,-64,0);
s.textureTransformList[3].rotate(0,0,90);
s.textureTransformList[3].translate(64,64,0);
```

textureType

Syntax

```
member(whichCastmember).textureType
```

Beschreibung

Mit dieser 3D-Textureigenschaft können Sie den Texturtyp für die Standardtextur ermitteln und festlegen. Mögliche Werte:

- `#none` bedeutet, dass es keinen Texturtyp gibt.
- `#default` verwendet die Textur aus dem ursprünglichen Shader.

- `#member` verwendet die Grafik aus dem angegebenen Darsteller als Textur.

Der Standardwert dieser Eigenschaft lautet `#default`. Sie müssen für diese Eigenschaft den Wert `#member` angeben, um die Eigenschaft `textureMember` benutzen zu können.

Beispiel

Die folgende Anweisung setzt die Eigenschaft `textureType` des Darstellers "Szene" auf `#member`:

```
member("Scene").textureType = #member
```

Auf diese Weise kann über die Eigenschaft `textureMember` ein Bitmapdarsteller als Quelle für die Standardtextur verwendet werden. Der Bitmapdarsteller heißt „Gras“.

```
member("Scene").textureMember = "grass"
```

Siehe auch

`textureMember`

thumbNail

Syntax

```
-- Lingo syntax
memberObjRef.thumbNail

// JavaScript syntax
memberObjRef.thumbNail;
```

Beschreibung

Diese Darstellereigenschaft enthält die zur Vorschau eines Darstellers im Besetzungsfenster verwendete Grafik. Lesen/Schreiben; nur während der Erstellung.

Die Grafik kann für jeden Darsteller benutzerdefiniert eingestellt werden.

Beispiel

In der folgenden Anweisung sehen Sie, wie Sie einen Platzhalterdarsteller zur Anzeige eines anderen Piktogramms auf der Bühne einsetzen. Der Platzhalterdarsteller wird auf der Bühne platziert und das Bild dieses Darstellers auf das Piktogramm von Darsteller 10 eingestellt. Auf diese Weise kann eine verkleinerte Grafik angezeigt werden, ohne sie dafür skalieren oder anderweitig ändern zu müssen:

```
-- Lingo syntax
member("Placeholder").picture = member(10).thumbNail

// JavaScript syntax
member("Placeholder").picture = member(10).thumbNail;
```

Siehe auch

`Member`

tilt

Syntax

```
-- Lingo syntax
spriteObjRef.tilt

// JavaScript syntax
spriteObjRef.tilt;
```

Beschreibung

Diese QuickTime VR-Sprite-Eigenschaft gibt die aktuelle Neigung des QuickTime VR-Films in Grad an.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Die folgende Anweisung zeigt, wie die Neigung des aktuellen QuickTime-Films, der sich in sprite(3) befindet, abgerufen werden kann.

```
-- Lingo
putsprite(3).tilt

// Javascript
trace(sprite(3).tilt);
```

time (Timeout-Objekt)

Syntax

```
timeoutObject.time
```

Beschreibung

Diese Timeout-Objekteigenschaft gibt die Systemzeit in Millisekunden an, zu der vom jeweiligen *timeoutObject* das nächste Timeout-Ereignis gesendet wird.

Hierbei handelt es sich nicht um die Zeit, die bis zum nächsten Ereignis verstreicht, sondern die absolute Uhrzeit des nächsten Timeout-Ereignisses.

Beispiel

Die folgende Prozedur ermittelt, wann das nächste timeout-Ereignis vom Timeout-Objekt Update gesendet wird. Hierzu wird die Differenz zwischen der Eigenschaft time und dem aktuellen Millisekundenwert berechnet und das Ergebnis im Feld "Zeit bis" angezeigt:

```
-- Lingo
on prepareFrame
    msBeforeUpdate = timeout("Update").time - the milliseconds
    secondsBeforeUpdate = msBeforeUpdate / 1000
    minutesBeforeUpdate = secondsBeforeUpdate / 60
    member("Time Until").text = string(minutesBeforeUpdate) && "minutes before next \
    timeout"
end

// Javascript
Function prepareFrame()
{
    var msBeforeUpdate = timeout("Update").time - _system.milliseconds;
    var secondsBeforeUpdate = msBeforeUpdate / 1000;
    var minutesBeforeUpdate = secondsBeforeUpdate / 60;
    member("Time Until").text = string(minutesBeforeUpdate) + " minutes before next timeout";
}
```

Siehe auch

milliseconds, period, persistent, target, timeout(), timeoutHandler

timeoutHandler

Syntax

```
timeoutObject.timeoutHandler
```

Beschreibung

Diese Systemeigenschaft gibt den Namen der Prozedur an, an die das *timeoutObject* Timeout-Nachrichten sendet. Ihr Wert ist ein Symbol wie `#timeExpiredHandler`. `TimeoutHandler` ist immer eine Prozedur im`target`-Objekt des Timeout-Objekts oder in einem Filmskript, falls für das Timeout-Objekt kein `target` angegeben wird.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Die folgende Anweisung zeigt die `timeoutHandler`-Prozedur des Timeout-Objekts "Quiz-Zeitgeber" im Nachrichtenfenster an:

```
-- Lingo
put timeout("Quiz Timer").timeoutHandler

// Javascript
trace(timeout("Quiz Timer").timeoutHandler);
```

Siehe auch

target, timeout(), timeoutList

timeoutList

Syntax

```
-- Lingo syntax
_movie.timeoutList

// JavaScript syntax
_movie.timeoutList;
```

Beschreibung

Diese Filmeigenschaft gibt eine lineare Liste mit allen gegenwärtig aktiven Timeout-Objekten zurück. Nur Lesen.

Mit der `forget()`-Methode können Sie ein Timeout-Objekt löschen.

Timeout-Objekte werden mit der `new()`-Methode zur `timeoutList` hinzugefügt.

Beispiel

Die folgende Anweisung löscht das dritte Timeout-Objekt aus der Timeout-Liste:

```
-- Lingo syntax
_movie.timeoutList[3].forget()

// JavaScript syntax
_movie.timeoutList[3].forget();
```

Siehe auch

`forget() (Fenster)`, `Movie`, `new()`, `forget() (Timeout)`, `timeout()`

timeScale

Syntax

```
member(whichCastMember).timeScale
the timeScale of member whichCastMember
```

Beschreibung

Diese Darstellereigenschaft gibt die Zeiteinheit in Sekunden an, auf der die Bilder eines Digitalvideos beruhen. Beispielsweise ist eine Zeiteinheit in einem QuickTime-Digitalvideo 1/600 Sekunde.

Diese Eigenschaft kann getestet, aber nicht eingestellt werden.

Beispiel

Die folgende Anweisung zeigt die Zeitskala eines QuickTime-Digitalvideos in sprite(3) an.

```
-- Lingo
put(sprite(3).timeScale)

// Javascript
trace(sprite(3).timeScale)
```

Siehe auch

digitalVideoTimeScale

title (DVD)

Syntax

```
-- Lingo syntax
dvdObjRef.title

// JavaScript syntax
dvdObjRef.title;
```

Beschreibung

Diese DVD-Eigenschaft gibt den aktuellen Titel an. Lesen/Schreiben.

Diese Eigenschaft gibt eine Ganzzahl zurück, die die Nummer des aktuellen Titels angibt.

Beispiel

Die folgende Anweisung gibt den aktuelle Titel zurück:

```
-- Lingo syntax
trace (member(1).title)-- 1

// JavaScript syntax
trace (member(1).title);// 1
```

Siehe auch

DVD

title (Fenster)

Syntax

```
-- Lingo syntax
windowObjRef.title

// JavaScript syntax
windowObjRef.title;
```

Beschreibung

Diese Fenstereigenschaft weist einem Fenster einen Titel zu. Lesen/Schreiben.

Beispiel

Die folgenden Anweisungen weisen dem fünften Fenster den Titel „Planets“ zu:

```
-- Lingo syntax
_player.windowList[5].title = "Planets"

// JavaScript syntax
_player.windowList[5].title = "Planets";
```

Siehe auch

Window

titlebarOptions

Syntax

```
-- Lingo syntax
windowObjRef.titlebarOptions

// JavaScript syntax
windowObjRef.titlebarOptions;
```

Beschreibung

Diese Fenstereigenschaft gibt eine Liste von Eigenschaften an, in der die Titelleistenoptionen eines Fensters gespeichert sind. Lesen/Schreiben.

Die Eigenschaftsliste enthält die folgenden Eigenschaften:

Eigenschaft	Beschreibung
`#icon`	Gibt das Darstellersymbol an, das in der Titelleiste verwendet werden soll. Diese Eigenschaft ist nur verfügbar, wenn die Titelleiste eingeblendet ist (die Eigenschaft `#visible` auf `TRUE` festgelegt ist).
`#visible`	Gibt an, ob die Titelleiste sichtbar ist: Bei Festlegung auf `FALSE` ist die Titelleiste nicht sichtbar und alle anderen Titelleisten- und Fenstereigenschaften bleiben unverändert. Bei Festlegung auf `TRUE` ist die Titelleiste sichtbar und das Fenster erhält den Zustand aller anderen Titelleisten- und Fenstereigenschaften. Der Standardwert ist `TRUE`.
`#closebox`	Gibt an, ob in der rechten oberen Ecke des Fensters eine Schaltfläche zum Schließen angezeigt wird. Bei Festlegung auf `TRUE` wird eine Schaltfläche zum Schließen angezeigt. Bei Festlegung auf `FALSE` wird keine Schaltfläche zum Schließen angezeigt. Der Standardwert ist `TRUE`.
`#minimizebox`	Gibt an, ob in der rechten oberen Ecke des Fensters eine Schaltfläche zum Minimieren angezeigt wird. Bei Festlegung auf `TRUE` wird eine Schaltfläche zum Minimieren angezeigt. Bei Festlegung auf `FALSE` wird keine Schaltfläche zum Minimieren angezeigt. Der Standardwert ist `TRUE`.
`#maximizebox`	Gibt an, ob in der rechten oberen Ecke des Fensters eine Schaltfläche zum Maximieren angezeigt wird. Bei Festlegung auf `TRUE` wird eine Schaltfläche zum Maximieren angezeigt. Bei Festlegung auf `FALSE` wird keine Schaltfläche zum Maximieren angezeigt. Der Standardwert ist `TRUE`.
`#sideTitlebar`	(Nur Mac) Gibt an, ob die Titelleiste an der Seite des Fensters angezeigt wird. Bei Festlegung auf `TRUE` wird die Titelleiste an der Seite des Fensters angezeigt. Bei Festlegung auf `FALSE` wird die Titelleiste nicht an der Seite des Fensters angezeigt. Der Standardwert ist `FALSE`.

Auf diese Eigenschaften kann ebenfalls unter Verwendung der Eigenschaft `displayTemplate` des Movie-Objekts zugegriffen werden.

Beispiel

Die folgende Anweisung zeigt die verfügbaren Titelleistenoptionen für das Fenster namens „Elements“ im Nachrichtenfenster an:

```
-- Lingo syntax
trace(window("Elements").titlebarOptions)

// JavaScript syntax
trace(window("Elements").titlebarOptions);
```

Die folgende Anweisung legt die Eigenschaft icon auf den Bitmapdarsteller `smallIcon` fest:

```
-- Lingo syntax
window("Elements").titlebarOptions.icon = member("smallIcon")

// JavaScript syntax
window("Elements").titlebarOptions.icon = member("smallIcon");
```

Siehe auch

appearanceOptions, displayTemplate, Window

titleCount

Syntax

```
-- Lingo syntax
dvdObjRef.titleCount

// JavaScript syntax
dvdObjRef.titleCount;
```

Beschreibung

Diese DVD-Eigenschaft gibt die Anzahl der verfügbaren Titel zurück. Nur Lesen.

Die Anzahl der verfügbaren Titel liegt zwischen 1 und 99.

Beispiel

Die folgende Anweisung gibt die Titelanzahl für das DVD-Objekt zurück.

```
-- Lingo
put sprite(1).titleCount

// Javascript
trace(sprite(1).titleCount)
```

Siehe auch

DVD

toolXtraList

Syntax

```
-- Lingo syntax
_player.toolXtraList

// JavaScript syntax
_player.toolXtraList;
```

Beschreibung

Diese Player-Eigenschaft gibt eine lineare Liste aller im Director-Player verfügbaren Xtra-Werkzeugerweiterungen zurück. Nur Lesen.

Beispiel

Die folgende Anweisung zeigt alle verfügbaren Xtra-Werkzeugerweiterungen im Nachrichtenfenster an.

```
-- Lingo syntax
put(_player.toolXtraList)

// JavaScript syntax
put(_player.toolXtraList);
```

Siehe auch

`mediaXtraList`, `Player`, `scriptingXtraList`, `transitionXtraList`, `xtraList (Player)`

toon (Modifizierer)

Syntax

```
member(whichCastmember).model(whichModel).toon.toonModifierProperty
```

Beschreibung

Nachdem Sie den 3D-Modifizierer `#toon` zu einem Modell hinzugefügt haben, können Sie die Eigenschaften dieses Modifizierers `#toon` abrufen und festlegen.

Der Modifizierer „#toon“ zeichnet ein Modell mit einer Handvoll Farben, was zu einer cartoonartigen Wiedergabe der Modelloberfläche führt. Bei Verwendung des Modifizierers `#toon` werden die Shader-Eigenschaften `texture`, `reflectionMap`, `diffuseLightMap`, `specularLightMap` und `glossMap` des Modells ignoriert.

Bei Verwendung des Modifizierers `#toon` in Verbindung mit dem Modifizierer `#inker` ergibt sich beim Rendering ein kumulativer Effekt, der je nachdem, welcher Modifizierer zuerst angewandt wurde, variiert. Die von der Eigenschaft `modifier` zurückgegebene Modifiziererliste enthält entweder `#inker` oder `#toon` (je nachdem, welcher Modifizierer zuerst hinzugefügt wurde), aber nicht beide. Der Modifizierer „#toon“ kann nicht zusammen mit dem Modifizierer `#sds` verwendet werden.

Der Modifizierer `#toon` weist folgende Eigenschaften auf:

Hinweis: *Weitere Informationen zu den folgenden Eigenschaften finden Sie in den einzelnen Beschreibungen.*

- Mit `style` können Sie den Farbverlaufsstil ermitteln und festlegen. Mögliche Werte:

  `#toon`: abrupte Übergänge zwischen verfügbaren Farben.

  `#gradient`: weiche Übergänge zwischen verfügbaren Farben.

  `#blackAndWhite`: abrupter Übergang zwischen Schwarz und Weiß.

- Mit `colorSteps` können Sie die Anzahl der für Beleuchtungsberechnungen verwendeten Farben ermitteln und festlegen. Dieser Wert wird auf die nächste Potenz von 2 abgerundet. Zulässige Werte: 2, 4, 8 und 16. Standardwert: 2.
- Mit `shadowPercentage` können Sie den für die Beleuchtung definierten Prozentsatz von Farben (`colorSteps`), die zur Wiedergabe des Schattenabschnitts der Modelloberfläche verwendet werden, ermitteln und festlegen. Mögliche Werte: 0 bis 100. Standardwert: 50.
- Mit `shadowStrength` können Sie ermitteln und festlegen, wie dunkel der Schattenabschnitt der Modelloberfläche wiedergegeben werden soll. Mögliche Werte: jede beliebige nicht negative Fließkommazahl. Der Standardwert ist 1.0.

- Mit `highlightPercentage` können Sie den für die Beleuchtung definierten Prozentsatz von Farben (`colorSteps`), die zur Wiedergabe des Highlight-Abschnitts der Modelloberfläche verwendet werden, ermitteln und festlegen. Mögliche Werte: 0 bis 100. Standardwert: 50.
- Mit `highlightStrength` können Sie ermitteln und festlegen, wie hell der Highlight-Abschnitt der Modelloberfläche wiedergegeben werden soll. Mögliche Werte: jede beliebige nicht negative Fließkommazahl. Der Standardwert ist 1.0.
- Mit `lineColor` können Sie die Farbe der von diesem Modifizierer gezeichneten Linien abrufen und einstellen. Mögliche Werte: jedes beliebige Lingo-Farbobjekt. Standardwert: `rgb (0, 0, 0)` (schwarz).
- Mit `creases` können Sie abrufen und einstellen, ob in Falten Linien gezeichnet werden. Dies ist ein Boolescher Wert; der Standardwert ist `TRUE`.
- Mit `creaseAngle` können Sie ermitteln und festlegen, wie empfindlich die Linienzeichnungsfunktion des Modifizierers "toon" auf Falten anspricht, wenn `creases` auf `TRUE` gesetzt ist.
- Mit `boundary` können Sie ermitteln und festlegen, ob um die Grenzen der Oberfläche herum Linien gezeichnet werden. Dies ist ein Boolescher Wert; der Standardwert ist `TRUE`.
- Mit `lineOffset` können Sie ermitteln und festlegen, wo relativ zur schattierten Oberfläche und der Kamera Linien gezeichnet werden. Bei negativen Werten werden die Linien zur Kamera hin verschoben, bei positiven Werten von der Kamera weg. Mögliche Werte sind Fließkommawerte von -100,0 bis 100,0. Der Standardwert ist 2,0.
- Mit `useLineOffset` können Sie ermitteln und festlegen, ob `lineOffset` aktiviert ist oder nicht. Dies ist ein Boolescher Wert; der Standardwert ist `FALSE`.
- Mit `silhouettes` können Sie ermitteln und festlegen, ob zum Definieren der Kanten entlang der Modellgrenze Linien gezeichnet werden, durch die die Form des Modells hervorgehoben wird. Dies ist ein Boolescher Wert; der Standardwert ist `TRUE`.

Siehe auch

`addModifier`, `modifiers`, `sds (Modifizierer)`, `inker (Modifizierer)`

top

Syntax

```
-- Lingo syntax
spriteObjRef.top

// JavaScript syntax
spriteObjRef.top;
```

Beschreibung

Diese Sprite-Eigenschaft gibt die obere vertikale Koordinate des Begrenzungsrechtecks des Sprites zurück oder legt sie fest. Die Koordinate wirdin Pixel von der linken oberen Ecke der Bühne aus angegeben. Lesen/Schreiben.

Beispiel

Die folgende Anweisung prüft, ob der obere Rand von Sprite 3 über den oberen Rand der Bühne hinausragt, und ruft die Prozedur offtopedge auf, wenn dies der Fall ist:

```
-- Lingo syntax
if (sprite(3).top < 0) then
    offTopEdge()
end if

// JavaScript syntax
if (sprite(3).top < 0) {
    offTopEdge();
}
```

Siehe auch

bottom, height, left, locH, locV, right, Sprite, width

topSpacing

Syntax

```
chunkExpression.topSpacing
```

Beschreibung

Mit dieser Textdarstellereigenschaft können Sie zusätzlichen Abstand am oberen Rand eines jeden Absatzes im *chunkExpression*-Teil des Textdarstellers angeben.

Der Wert selbst ist eine Ganzzahl; Werte unter 0 bedeuten einen kleineren Abstand zwischen Absätzen und Werte über 0 einen größeren.

Der Standardwert ist 0. Dadurch werden Absätze durch den Standardabstand getrennt.

Beispiel

Die folgende Anweisung setzt die Eigenschaft `topSpacing` für den zweiten Absatz im Textdarsteller "`myText`" auf 20:

```
-- Lingo
member(1).paragraph[2].topSpacing = 20

// Javascript
member(1).getPropRef("paragraph",2).topSpacing = 20;
```

Siehe auch

bottomSpacing

traceLoad

Syntax

```
-- Lingo syntax
_movie.traceLoad

// JavaScript syntax
_movie.traceLoad;
```

Beschreibung
Diese Filmeigenschaft gibt den Umfang der Informationen an, die beim Laden von Darstellern über sie angezeigt werden. Lesen/Schreiben.

`traceLoad` kann folgende gültige Werte annehmen:

- 0 – Zeigt keine Informationen an (Standard).
- 1 – Zeigt die Darstellernamen an.
- 2 – Zeigt die Darstellernamen, die Nummer des aktuellen Bildes, den Filmnamen und den sogenannten „File Seek Offset“ an (die relative Menge, die zum Laden der Media auf dem Datenträger verschoben werden musste).

Beispiel
Die folgende Anweisung bewirkt, dass die Namen der Darsteller beim Laden angezeigt werden:

```
-- Lingo syntax
_movie.traceLoad = 1

// JavaScript syntax
_movie.traceLoad = 1;
```

Siehe auch
`Movie`

traceLogFile

Syntax

```
-- Lingo syntax
_movie.traceLogFile

// JavaScript syntax
_movie.traceLogFile;
```

Beschreibung
Diese Filmeigenschaft gibt den Namen der Datei an, in die die Nachrichtenfensteranzeige geschrieben werden soll. Lesen/Schreiben.

Sie können die Datei schließen, indem Sie die Eigenschaft `traceLogFile` auf `EMPTY` (Lingo) oder auf eine leere Zeichenfolge " " (JavaScript-Syntax) setzen. Jede Ausgabe, die im Nachrichtenfenster erscheint, wird in diese Datei geschrieben. Diese Eigenschaft ist sowohl beim Abspielen eines Films in einem Projektor als auch beim Erstellen des Films zum Debugging geeignet.

Beispiel
Die folgende Anweisung schreibt den Inhalt des Nachrichtenfensters in die Datei „Nachrichten.txt“ im selben Ordner wie der aktuelle Film:

```
-- Lingo syntax
_movie.traceLogFile = _movie.path & "Messages.txt"

// JavaScript syntax
_movie.traceLogFile = _movie.path + "Messages.txt";
```

Die folgende Anweisung schließt die Datei, in die die Nachrichtenfensteranzeige geschrieben wird:

```
-- Lingo syntax
_movie.traceLogFile = ""

// JavaScript syntax
_movie.traceLogFile = "";
```

Siehe auch

Movie

traceScript

Syntax

```
-- Lingo syntax
_movie.traceScript

// JavaScript syntax
_movie.traceScript;
```

Beschreibung

Diese Filmeigenschaft gibt an, ob die Verfolgungsfunktion des Films eingeschaltet ist (`TRUE`) oder nicht (`FALSE`).

Lesen/Schreiben.

Wenn `traceScript` aktiv ist, erscheinen alle ausgeführten Skriptzeilen im Nachrichtenfenster.

Beispiel

Die folgende Anweisung aktiviert die Eigenschaft `traceScript`.

```
-- Lingo syntax
_movie.traceScript = TRUE

// JavaScript syntax
_movie.traceScript = true;
```

Siehe auch

Movie

trackCount (Darsteller)

Syntax

```
-- Lingo syntax
memberObjRef.trackCount()

// JavaScript syntax
memberObjRef.trackCount();
```

Beschreibung
Diese Digitalvideo-Darstellereigenschaft gibt die Anzahl von Spuren im angegebenen Digitalvideo-Darsteller an.

Diese Eigenschaft kann getestet, aber nicht eingestellt werden.

Beispiel
Die folgende Anweisung bestimmt die Anzahl von Spuren im Digitalvideo-Darsteller "Jazz-Chronik“ und zeigt das Ergebnis im Nachrichtenfenster an:

```
-- Lingo syntax
put(member("Jazz Chronicle").trackCount())

// JavaScript syntax
trace(member("Jazz Chronicle").trackCount());
```

trackCount (Sprite)

Syntax

```
-- Lingo syntax
spriteObjRef.trackCount()

// JavaScript syntax
spriteObjRef.trackCount();
```

Beschreibung
Diese Digitalvideo-Sprite-Eigenschaft gibt die Anzahl von Spuren im angegebenen Digitalvideo-Sprite zurück.

Diese Eigenschaft kann getestet, aber nicht eingestellt werden.

Beispiel
Die folgende Anweisung bestimmt die Anzahl von Spuren im Digitalvideo-Sprite, das Kanal 10 zugewiesen ist, und zeigt das Ergebnis im Nachrichtenfenster an:

```
-- Lingo syntax
put(sprite(10).trackCount())

// JavaScript syntax
trace(sprite(10).trackCount());
```

trackEnabled

Syntax

```
-- Lingo syntax
spriteObjRef.trackEnabled(whichTrack)

// JavaScript syntax
spriteObjRef.trackEnabled(whichTrack);
```

Beschreibung

Diese Digitalvideo-Sprite-Eigenschaft zeigt den Status einer bestimmten Spur in einem Digitalvideo an. Diese Eigenschaft ist `TRUE`, wenn die Spur aktiviert ist und abgespielt wird. Diese Eigenschaft ist `FALSE`, wenn die Spur deaktiviert ist und nicht abgespielt oder aktualisiert wird.

Diese Eigenschaft kann nicht eingestellt werden. Verwenden Sie stattdessen die Eigenschaft `setTrackEnabled`.

Beispiel

Die folgende Anweisung prüft, ob Spur 2 von Digitalvideo-Sprite 1 aktiviert ist:

```
-- Lingo syntax
put(sprite(1).trackEnabled(2))

// JavaScript syntax
put(sprite(1).trackEnabled(2));
```

Siehe auch

`setTrackEnabled()`

trackInfo

Syntax

```
put member(1).trackInfo
put sprite(1).trackInfo
```

Beschreibung

MP4Media/FLV-Darsteller- und Sprite-Eigenschaft, welche die Liste aller Video- und Audiospuren im angegebenen Videodarsteller zurückgeben. Diese Eigenschaft lässt sich auslesen, aber nicht verändern.

Die Liste der Spuren liegt in folgendem Format vor.

```
[[#trackNum: #trackType]…]
#trackNum is 0, 1, 2…
#trackType is #video or #audio
Output: [[#tracknum:0 , #trackType:#video],[#tracknum:1, #trackType: #audio]]
```

Hinweis: *Für MP4-Video-Darsteller wird nur eine Videospur unterstützt. Videodarsteller können jedoch beliebig viele Tonspuren besitzen.*

Beispiele

```
-- Lingo syntax
put member(1).trackInfo -- [[#tracknum:0 , #trackType:#video],[#tracknum:1,
-- #trackType:#audio]]
put sprite(1).trackInfo -- [[#tracknum:0 , #trackType:#video],[#tracknum:1,
-- #trackType:#audio]]

// JavaScript syntax
put member(1).trackInfo; // [[#tracknum:0 , #trackType:#video],[#tracknum:1,
// #trackType:#audio]]
put sprite(1).trackInfo; // [[#tracknum:0 , #trackType:#video],[#tracknum:1,
// #trackType:#audio]]
```

trackNextKeyTime

Syntax

```
-- Lingo syntax
spriteObjRef.trackNextKeyTime(whichTrack)

// JavaScript syntax
spriteObjRef.trackNextKeyTime(whichTrack);
```

Beschreibung

Diese Digitalvideo-Sprite-Eigenschaft gibt die Zeit des Schlüsselbildes an, das der aktuellen Zeit in der angegebenen Digitalvideospur folgt.

Diese Eigenschaft kann getestet, aber nicht eingestellt werden.

Beispiel

Die folgende Anweisung bestimmt die Zeit des Schlüsselbildes, das der aktuellen Zeit in Spur 5 des Digitalvideos folgt, das Sprite-Kanal 15 zugeordnet ist, und zeigt das Ergebnis im Nachrichtenfenster an:

```
-- Lingo syntax
put(sprite(15).trackNextKeyTime(5))

// JavaScript syntax
put(sprite(15).trackNextKeyTime(5));
```

trackNextSampleTime

Syntax

```
-- Lingo syntax
spriteObjRef.trackNextSampleTime(whichTrack)

// JavaScript syntax
spriteObjRef.trackNextSampleTime(whichTrack);
```

Beschreibung

Diese Digitalvideo-Sprite-Eigenschaft gibt die Zeit des nächsten Samples an, das auf die aktuelle Zeit im Digitalvideo folgt. Mit dieser Eigenschaft können Sie leicht Textspuren im Digitalvideo finden.

Diese Eigenschaft kann getestet, aber nicht eingestellt werden.

Beispiel

Die folgende Anweisung bestimmt die Zeit des nächsten Samples, das auf die aktuelle Zeit in Spur 5 des Digitalvideos folgt, welches Sprite 15 zugeordnet ist:

```
-- Lingo syntax
put(sprite(15).trackNextSampleTime(5))

// JavaScript syntax
put(sprite(15).trackNextSampleTime(5));
```

trackPreviousKeyTime

Syntax

```
-- Lingo syntax
spriteObjRef.trackPreviousKeyTime(whichTrack)

// JavaScript syntax
spriteObjRef.trackPreviousKeyTime(whichTrack);
```

Beschreibung

Diese Digitalvideo-Sprite-Eigenschaft gibt die Zeit des Schlüsselbildes an, das der aktuellen Zeit vorausgeht.

Diese Eigenschaft kann getestet, aber nicht eingestellt werden.

Beispiel

Die folgende Anweisung bestimmt die Zeit des Schlüsselbildes in Spur 5, das der aktuellen Zeit im Digitalvideo-Sprite vorausgeht, welches Kanal 15 zugeordnet ist, und zeigt das Ergebnis im Nachrichtenfenster an:

```
-- Lingo syntax
put(sprite(15).trackPreviousKeyTime(5))

// JavaScript syntax
put(sprite(15).trackPreviousKeyTime(5));
```

trackPreviousSampleTime

Syntax

```
-- Lingo syntax
spriteObjRef.trackPreviousSampleTime(whichTrack)

// JavaScript syntax
spriteObjRef.trackPreviousSampleTime(whichTrack);
```

Beschreibung

Diese Digitalvideo-Sprite-Eigenschaft zeigt die Zeit des Samples an, das der aktuellen Zeit im Digitalvideo vorausgeht. Mit dieser Eigenschaft können Sie leicht Textspuren im Digitalvideo finden.

Diese Eigenschaft kann getestet, aber nicht eingestellt werden.

Beispiel

Die folgende Anweisung bestimmt die Zeit des Samples in Spur 5, das der aktuellen Zeit im Digitalvideo-Sprite vorausgeht, welches Kanal 15 zugeordnet ist, und zeigt das Ergebnis im Nachrichtenfenster an:

```
-- Lingo syntax
put(sprite(15).trackPreviousSampleTime(5))

// JavaScript syntax
put(sprite(15).trackPreviousSampleTime(5));
```

trackStartTime (Darsteller)

Syntax

```
-- Lingo syntax
memberObjRef.trackStartTime(whichTrack)

// JavaScript syntax
memberObjRef.trackStartTime(whichTrack);
```

Beschreibung

Diese Digitalvideo-Darsteller-Eigenschaft gibt die Startzeit der angegebenen Spur eines bestimmten Digitalvideo-Darstellers zurück.

Diese Eigenschaft kann getestet, aber nicht eingestellt werden.

Beispiel

Die folgende Anweisung bestimmt die Startzeit von Spur 5 im Digitalvideo-Darsteller "Jazz-Chronik“ und zeigt das Ergebnis im Nachrichtenfenster an:

```
-- Lingo syntax
put(member("Jazz Chronicle").trackStartTime(5))

// JavaScript syntax
put(member("Jazz Chronicle").trackStartTime(5));
```

trackStartTime (Sprite)

Syntax

```
-- Lingo syntax
spriteObjRef.trackStartTime(whichTrack)

// JavaScript syntax
spriteObjRef.trackStartTime(whichTrack);
```

Beschreibung

Diese Digitalvideo-Sprite-Eigenschaft gibt die Startzeit eines Digitalvideofilms im angegebenen Sprite-Kanal an. Der Wert von `trackStartTime` wird in Ticks gemessen.

Diese Eigenschaft kann getestet, aber nicht eingestellt werden.

Beispiel

Die folgende Anweisung zeigt im Nachrichtenfenster an, wann die Wiedergabe von Spur 5 in Sprite-Kanal 10 beginnt. Die Startzeit liegt 120 Ticks (2 Sekunden) nach Beginn der Spur.

```
-- Lingo syntax
put(sprite(10).trackStartTime(5))

// JavaScript syntax
put(sprite(10).trackStartTime(5))
```

Siehe auch

duration (Darsteller), playRate (QuickTime, AVI, MP4, FLV), currentTime (QuickTime, AVI)

trackStopTime (Darsteller)

Syntax

```
-- Lingo syntax
memberObjRef.trackStopTime(whichTrack)

// JavaScript syntax
memberObjRef.trackStopTime(whichTrack);
```

Beschreibung

Diese Digitalvideo-Darsteller-Eigenschaft gibt die Endzeit der angegebenen Spur eines bestimmten Digitalvideo-Darstellers an. Sie kann getestet, aber nicht eingestellt werden.

Beispiel

Die folgende Anweisung bestimmt die Endzeit von Spur 5 im Digitalvideo-Darsteller "Jazz-Chronik“ und zeigt das Ergebnis im Nachrichtenfenster an:

```
-- Lingo syntax
put(member("Jazz Chronicle").trackStopTime(5))

// JavaScript syntax
put(member("Jazz Chronicle").trackStopTime(5));
```

trackStopTime (Sprite)

Syntax

```
-- Lingo syntax
spriteObjRef.trackStopTime(whichTrack)

// JavaScript syntax
spriteObjRef.trackStopTime(whichTrack);
```

Beschreibung

Diese Digitalvideo-Sprite-Eigenschaft gibt die Endzeit der angegebenen Spur eines bestimmten Digitalvideo-Sprites zurück.

Wenn ein Digitalvideofilm abgespielt wird, kennzeichnet `trackStopTime` die Stelle, an der die Wiedergabe unterbrochen bzw. die Schleife durchlaufen wird, sofern die Eigenschaft `loop` eingeschaltet ist.

Diese Eigenschaft kann getestet, aber nicht eingestellt werden.

Beispiel

Die folgende Anweisung bestimmt die Endzeit von Spur 5 im Digitalvideo, das Sprite 6 zugeordnet ist, und zeigt das Ergebnis im Nachrichtenfenster an:

```
-- Lingo syntax
put(sprite(6).trackStopTime(5))

// JavaScript syntax
put(sprite(6).trackStopTime(5));
```

Siehe auch

playRate (QuickTime, AVI, MP4, FLV), currentTime (QuickTime, AVI), trackStartTime (Darsteller)

trackText

Syntax

```
-- Lingo syntax
spriteObjRef.trackText(whichTrack)

// JavaScript syntax
spriteObjRef.trackText(whichTrack);
```

Beschreibung

Diese Digitalvideo-Sprite-Eigenschaft zeigt den Text an, der sich zum gegenwärtigen Zeitpunkt in der angegebenen Spur des Digitalvideos befindet. Das Ergebnis ist ein bis zu maximal 32.000 Zeichen langer String. Die Eigenschaft kann nur für Textspuren verwendet werden.

Diese Eigenschaft kann getestet, aber nicht eingestellt werden.

Beispiel

Die folgende Anweisung ordnet dem Felddarsteller "Archive“ den Text aus Spur 5 des Digitalvideos zu, das gegenwärtig Sprite 20 zugeordnet ist:

```
-- Lingo syntax
member("Archives").text = string(sprite(20).trackText(5))

// JavaScript syntax
member("Archives").text = sprite(20).trackText(5).toString();
```

trackType (Darsteller)

Syntax

```
-- Lingo syntax
memberObjRef.trackType(whichTrack)

// JavaScript syntax
memberObjRef.trackType(whichTrack);
```

Beschreibung

Diese Digitalvideo-Darsteller-Eigenschaft gibt an, welche Medientypen sich in der angegebenen Spur eines bestimmten Darstellers befinden. Mögliche Werte sind `#video`, `#sound`, `#text` und `#music`.

Diese Eigenschaft kann getestet, aber nicht eingestellt werden.

Beispiel

Die folgende Prozedur prüft, ob Spur 5 des Digitalvideo-Darstellers „Aktuelle Nachrichten“ eine Textspur ist, und führt anschließend die Prozedur `textFormat` aus, wenn dies der Fall ist:

```
-- Lingo syntax
on checkForText
    if member("Today's News").trackType(5) = #text then
        textFormat
    end if
end

// JavaScript syntax
function checkForText() {
    var tt = member("Today's News").trackType(5);
    if (tt == "text") {
        textFormat();
    }
}
```

trackType (Sprite)

Syntax

```
-- Lingo syntax
spriteObjRef.trackType(whichTrack)

// JavaScript syntax
spriteObjRef.trackType(whichTrack);
```

Beschreibung

Diese Digitalvideo-Sprite-Eigenschaft gibt den Medientyp in einer bestimmten Spur des angegebenen Sprites zurück. Mögliche Werte sind `#video`, `#sound`, `#text` und `#music`.

Diese Eigenschaft kann getestet, aber nicht eingestellt werden.

Beispiel

Folgende Prozedur prüft, ob Spur 5 des Digitalvideo-Sprites, das Kanal 10 zugeordnet ist, eine Textspur ist, und führt die Prozedur `textFormat` aus, wenn dies der Fall ist:

```
-- Lingo syntax
on checkForText
    if sprite(10).trackType(5) = #text then
        textFormat
    end if
end

// JavaScript syntax
function checkForText() {
    var tt = sprite(10).trackType(5);
    if (tt == "text") {
        textFormat();
    }
}
```

trails

Syntax

```
sprite(whichSprite).trails
the trails of sprite whichSprite
```

Beschreibung

Diese Sprite-Eigenschaft schaltet für das in *whichSprite* angegebene Sprite den Spurenfarbeffekt ein (1 oder `TRUE`) oder aus (0 oder `FALSE`). Der von Lingo eingestellte Wert ist nur dann über das aktuelle Sprite hinaus gültig, wenn es sich bei diesem um einmit Skript erstelltes Sprite handelt.

Um Spuren zu löschen, animieren Sie ein anderes Sprite über diese Pixel hinweg, oder verwenden Sie einen Übergang.

Beispiel

Die folgende Anweisung schaltet die Spuren für Sprite 7 ein:

```
-- Lingo
sprite(7).trails = 1

// Javascript
sprite(7).trails = 1;
```

Siehe auch

`directToStage`

transform (Eigenschaft)

Syntax

```
member(whichCastmember).node(whichNode).transform
member(whichCastmember).node(whichNode).transform.transformProperty
member(whichCastmember).model(whichModel).bonesPlayer.bone[boneID].transform
member(whichCastmember).model(whichModel).bonesPlayer.bone[boneID].transform.transformProperty
```

Beschreibung

Mit dieser 3D-Eigenschaft bzw. diesem 3D-Befehl können Sie die Transformation, die mit einer bestimmten Node bzw. einem bestimmten Knochen in einem Modell mit dem Modifizierer `bonesPlayer` verknüpft ist, ermitteln und festlegen. Als Befehl ermöglicht `transform` den Zugriff auf die verschiedenen Befehle und Eigenschaften des Transformationsobjekts. Ein Node kann ein Kamera-, Gruppen-, Licht- oder Modellobjekt sein.

Bei Node-Objekten wird diese Eigenschaft standardmäßig auf die Einheitsmatrix gesetzt. Die Transformation eines Node definiert die Position, Drehung und Skalierung des Nodes im Verhältnis zu seinem Parent-Objekt. Wenn das Parent-Objekt eines Nodes das Gruppenobjekt „World“ ist, besitzt die Node-Eigenschaft `transform` den gleichen Wert, der durch den Befehl `getWorldTransform()` zurückgegeben wird.

Bei Knochen in Modellen, die den Modifizierer `bonesPlayer` verwenden, wird diese Eigenschaft standardmäßig auf die Transformation gesetzt, die dem Knochen bei Erstellung der Modelldatei zugeordnet wurde. Die Transformation eines Knochen stellt die Drehung des Knochens bezogen auf seinen Parent-Knochen und seine Position bezogen auf seine ursprüngliche Gelenkposition dar. Die ursprüngliche Gelenkposition wird bei Erstellung der Modelldatei festgelegt.

Zusammen mit der Eigenschaft `transform` von Node-Objekten können Sie die folgenden Transformationsbefehle und -eigenschaften verwenden:

Hinweis: *Dieser Abschnitt enthält lediglich eine Übersicht. Weitere Informationen finden Sie in den einzelnen Einträgen.*

- `preScale` führt vor den aktuellen Positions-, Drehungs- und Skalierungsoffsets der Transformation eine Skalierung durch.
- `preTranslate` führt vor den aktuellen Positions-, Drehungs- und Skalierungsoffsets der Transformation eine Translation durch.
- `preRotate` führt vor den aktuellen Positions-, Drehungs- und Skalierungsoffsets der Transformation eine Drehung durch.
- `scale` (Befehl) führt nach den aktuellen Positions-, Drehungs- und Skalierungsoffsets der Transformation eine Skalierung durch.
- Mit `scale` (Transformation) können Sie den Skalierungsfaktor der Transformation ermitteln und festlegen.
- `translate` führt nach den aktuellen Positions-, Drehungs- und Skalierungsoffsets der Transformation eine Translation durch.
- `rotate` führt nach den aktuellen Positions-, Drehungs- und Skalierungsoffsets der Transformation eine Drehung durch.
- Mit `position` (Transformation) können Sie den Positionsoffset der Transformation ermitteln und festlegen.
- Mit `rotation`(Transformation) können Sie den Drehungsoffset der Transformation ermitteln und festlegen.

Wenn Sie die Eigenschaft `transform` eines Knochens in einem Modell ändern möchten, müssen Sie eine Kopie der ursprüngliche Knochentransformation speichern, die gespeicherte Kopie mit den oben beschriebenen Befehlen und Eigenschaften ändern und anschließend die Eigenschaft `transform` des Knochens zurücksetzen, sodass sie der geänderten Transformation entspricht. Beispiel:

```
t = member("character").model("biped").bonesPlayer.bone[38].transform.duplicate()
t.translate(25,0,-3)
member("character").model("biped").bonesPlayer.bone[38].transform = t
```

Parameter

Keiner

Beispiel

Der folgende Lingo-Code zeigt die Transformation des Modells „Box“, gefolgt von ihren Positions- und Rotationseigenschaften:

```
put member("3d world").model("box").transform
-- transform(1.000000,0.000000,0.000000,0.000000, 0.000000,1.000000,0.000000,0.000000,
0.000000,0.000000,1.000000,0.000000, -94.144844,119.012825,0.000000,1.000000)
put member("3d world").model("box").transform.position
-- vector(-94.1448, 119.0128, 0.0000)
put member("3d world").model("box").transform.rotation
--vector(0.0000, 0.0000, 0.0000)
```

Siehe auch

`interpolateTo()`, `scale (Transformation)`, `rotation (Transformation)`, `position (Transformation)`, `bone`, `worldTransform`, `preRotate`, `preScale()`, `preTranslate()`

transitionType

Syntax

```
member(whichCastMember).transitionType
the transitionType of member whichCastMember
```

Beschreibung

Diese Übergangsdarsteller-Eigenschaft bestimmt den Übergangstyp, der als Nummer angegeben wird. Die möglichen Werte sind die gleichen wie die Codes, die im Befehl `puppetTransition` für Übergänge verwendet werden.

Beispiel

Die folgende Anweisung setzt den Typ für den Übergangsdarsteller 3 auf 51, was dem Übergang „Pixelweise auflösen“ entspricht:

```
member(3).transitionType = 51
```

transitionXtraList

Syntax

```
-- Lingo syntax
_player.transitionXtraList

// JavaScript syntax
_player.transitionXtraList;
```

Beschreibung

Diese Player-Eigenschaft gibt eine lineare Liste aller im Director-Player verfügbaren Xtra-Übergangserweiterungen zurück. Nur Lesen.

Beispiel

Die folgende Anweisung zeigt alle verfügbaren Xtra-Übergangserweiterungen im Nachrichtenfenster an.

```
-- Lingo syntax
put(_player.transitionXtraList)

// JavaScript syntax
put(_player.transitionXtraList);
```

Siehe auch

mediaXtraList, Player, scriptingXtraList, toolXtraList, xtraList (Player)

translation

Syntax

```
-- Lingo syntax
memberOrSpriteObjRef.translation

// JavaScript syntax
memberOrSpriteObjRef.translation;
```

Beschreibung

Diese QuickTime-Darsteller- und Sprite-Eigenschaft steuert den Versatz der Grafik des QuickTime-Sprites innerhalb des Sprite-Begrenzungsrechtecks.

Dieser Versatz wird relativ zur Standardposition des Sprites ausgedrückt, die durch die Eigenschaft `center` definiert ist. Wenn `center` auf `TRUE` gesetzt ist, wird das Sprite relativ zur Mitte seines Begrenzungsrechtecks versetzt; wenn `center` auf `FALSE` gesetzt ist, wird das Sprite relativ zur linken oberen Ecke des Begrenzungsrechtecks versetzt.

Der Versatz, der in Pixel angegeben wird (positive oder negative Ganzzahl), wird als Director-Liste definiert: [*xTrans*, *yTrans*]. Der Parameter *xTrans* gibt den horizontalen Versatz von der Standardposition des Sprites an. Der Parameter *yTrans* gibt den vertikalen Versatz an. Die Standardeinstellung ist [0,0].

Wenn die Sprite-Eigenschaft `crop` auf `TRUE` gesetzt ist, kann die Eigenschaft `translation` zur Maskierung von Teilen eines QuickTime-Films verwendet werden. Hierzu werden diese Teile außerhalb des Begrenzungsrechtecks verschoben. Wenn die Eigenschaft `crop` auf `FALSE` gesetzt ist, wird die Eigenschaft `translation` ignoriert und das immer in der linken oberen Ecke des Sprite-Rechtecks positioniert.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Beim folgenden Bildskript wird davon ausgegangen, dass die Eigenschaft center des Darstellers eines 320 Pixel breiten QuickTime-Sprites in Kanal 5 auf FALSE und die Eigenschaft crop auf TRUE gesetzt ist. Der Abspielkopf verbleibt so lange im aktuellen Bild, bis sich der horizontale Versetzungspunkt des Films in Schritten von jeweils 10 Pixel an den rechten Rand des Sprites bewegt hat. Dies entspricht dem Effekt „Nach rechts wischen“, d. h. das Sprite bewegt sich nach rechts und verschwindet. Wenn das Sprite nicht mehr sichtbar ist, geht der Abspielkopf zum nächsten Bild über.

```
-- Lingo syntax
on exitFrame
    horizontalPosition = sprite(5).translation[1]
    if horizontalPosition < 320 then
        sprite(5).translation = sprite(5).translation + [10, 0]
        _movie.go(_movie.frame)
    end if
end

// JavaScript syntax
function exitFrame() {
    var horizontalPosition = sprite(5).translation[1];
    if (horizontalPosition < 320 ) {
        sprite(5).translation = sprite(5).translation + list(10, 0);
        _movie.go(_movie.frame);
    }
}
```

transparent

Syntax

```
member(whichCastmember).shader(whichShader).transparent
member(whichCastmember).model(whichModel).shader.transparent
member(whichCastmember).model(whichModel).shaderList[shaderListIndex].transparent
```

Beschreibung

Mit dieser 3D-Eigenschaft für Standard-Shader können Sie ermitteln oder festlegen, ob ein Modell unter Verwendung von Alphawerten gemischt(`TRUE`) oder als undurchsichtig (opak) gerendert wird (`FALSE`). Der Standardwert dieser Eigenschaft lautet `TRUE` (Alphamischung).

Die Funktionalität von `shader.blend` hängt von dieser Eigenschaft ab.

Alle Shader können auf die Eigenschaften des `#standard`-Shaders zugreifen. Darüber hinaus haben die Shader `#engraver`, `#newsprint` und `#painter` noch weitere Eigenschaften, die speziell für den jeweiligen Shader-Typ gelten. Weitere Informationen finden Sie unter dem Eintrag `newShader`

Beispiel

Die folgende Anweisung bewirkt, dass das Modell "Pluto“ undurchsichtig wird. Die für den Shader des Modells festgelegte blend-Einstellung hat keine Wirkung.

```
-- Lingo
member("scene").model("Pluto").shader.transparent = FALSE

// Javascript
member("scene").getPropRef("model",1).getProp("shader").transparent = false;
```

Siehe auch

`blendFactor`, `blend (3D)`

triggerCallback

Syntax

```
-- Lingo syntax
spriteObjRef.triggerCallback

// JavaScript syntax
spriteObjRef.triggerCallback;
```

Beschreibung

Diese QuickTime VR-Sprite-Eigenschaft enthält den Namen der Prozedur, die ausgeführt wird, wenn der Benutzer in einem QuickTime VR-Film auf einen Hotspot klickt. Der Prozedur werden zwei Argumente übergeben: der Parameter `me` und die ID des Hotspots, den der Benutzer geklickt hat.

Der von der Prozedur zurückgegebene Wert bestimmt, wie der Film den Hotspot verarbeitet. Gibt die Prozedur `#continue` zurück, verarbeitet das QuickTime VR-Sprite den Hotspot weiterhin normal. Gibt die Prozedur `#cancel` zurück, wird das Standardverhalten des Hotspots aufgehoben.

Setzen Sie die Eigenschaft auf 0, um den Rückruf zu löschen.

Das QuickTime VR-Sprite erhält die Nachricht zuerst.

Um einen Leistungsabfall zu vermeiden, sollten Sie die Eigenschaft `triggerCallback` nur bei Bedarf verwenden.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Die folgende Anweisung setzt die Rückrufprozedur für ein QuickTime VR-Sprite auf die Prozedur `MyHotSpotCallback`, wenn der Abspielkopf zum ersten Mal in den Sprite-Einschluss eintritt. Bei jedem Auslösen des Hotspots wird die Prozedur `MyHotSpotCallback` ausgeführt. Wenn der Abspielkopf den Sprite-Einschluss verlässt, wird der Rückruf abgebrochen.

```
-- Lingo syntax
property pMySpriteNum, spriteNum

on beginSprite(me)
    pMySpriteNum = spriteNum
    sprite(pMySpriteNum).triggerCallback = #MyHotSpotCallback
end

on MyHotSpotCallback(me, hotSpotID)
    put "Hotspot" && hotSpotID && "was just triggered"
end

on endSprite me
    sprite(pMySpriteNum).triggerCallback = 0
end

// JavaScript syntax
function beginSprite() {
    pMySpriteNum = this.spriteNum;
    sprite(this.pMySpriteNum).triggerCallback = symbol("MyHotSpotCallback");
}

function MyHotSpotCallback(hotSpotID) {
    trace("Hotspot " + hotSpotID + " was just triggered");
}

function endSprite() {
    sprite(pMySpriteNum).triggerCallback = 0;
}
```

trimWhiteSpace

Syntax

```
-- Lingo syntax
memberObjRef.trimWhiteSpace

// JavaScript syntax
memberObjRef.trimWhiteSpace;
```

Beschreibung

Diese Darstellereigenschaft bestimmt, ob die weißen Pixel um einen Bitmapdarsteller herum entfernt und beibehalten werden. Diese Eigenschaft wird beim Importieren des Darstellers eingestellt. Sie kann per Lingo oder über die Registerkarte „Bitmap“ im Eigenschafteninspektor geändert werden.

Beispiel

Die folgende Anweisung bewirkt, dass die weißen Bereiche im Textdarsteller zugeschnitten werden.

```
-- Lingo
member(1).trimWhiteSpace;

// Javascript
member(1).trimWhiteSpace;
```

tunnelDepth

Syntax

```
member(whichTextmember).tunnelDepth
member(whichCastMember).modelResource(whichExtruderModelResource).tunnelDepth
```

Beschreibung

Mit dieser 3D-Eigenschaft für Extrudermodellressourcen und Textdarsteller Mit dieser Eigenschaft können Sie die Extrusionstiefe (Abstand zwischen der Vorder- und Rückseite) einer 3D-Textmodellressource ermitteln und festlegen. Mögliche Werte: Fließkommazahlen zwischen 100,0 und 50,0.

Weitere Informationen zur Arbeit mit Extrudermodellressourcen und Textdarstellern enthält der Eintrag zu `extrudeToMember`.

Beispiel

In diesem Beispiel ist der Darsteller "Logo" ein Textdarsteller. Die folgende Anweisung setzt die Eigenschaft `tunnelDepth` von „Logo“ auf 5. Wenn „Logo“ im 3D-Modus angezeigt wird, sind seine Buchstaben nicht sehr tief.

```
-- Lingo
member("logo").tunnelDepth = 5

// Javascript
member("logo").tunnelDepth = 5;
```

In diesem Beispiel ist die Modellressource des Modells „Slogan“ extrudierter Text. Die folgende Anweisung setzt `tunnelDepth` der Modellressource von „Slogan“ auf 1000, sodass die Buchstaben von „Slogan“ sehr tief erscheinen.

```
-- Lingo
member("scene").model("Slogan").resource.tunnelDepth = 1000

// Javascript
member("scene").getPropRef("model",1).getProp("resource").tunnelDepth = 1000;
```

Siehe auch

`extrude3D`

tweened

Syntax

```
sprite(whichSprite).tweened
the tweened of sprite whichSprite
```

Beschreibung

Diese Sprite-Eigenschaft bestimmt, ob nur das erste Bild eines neuen Sprites als Schlüsselbild erstellt wird (`TRUE`) oder ob alle Bilder des neuen Sprites als Schlüsselbilder erstellt werden (`FALSE`).

Diese Eigenschaft hat keinen Einfluss auf die Wiedergabe und ist nur für die Drehbuchaufzeichnung nützlich.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Bei der folgenden Anweisung wird für neu erstellte Sprites in Kanal 25 nur das erste Bild des Sprite-Einschlusses als Schlüsselbild eingerichtet:

```
-- Lingo
sprite(25).tweened = 1

// Javascript
sprite(25).tweened = 1;
```

tweenMode

Syntax

```
member(whichCastmember).modelResource(whichModelResource).tweenMode
modelResourceObjectReference.tweenMode
```

Beschreibung

Mit dieser 3D-Partikeleigenschaft können Sie ermitteln und festlegen, ob die Farbe eines Partikels nach Geschwindigkeit oder Alter variiert. Die Eigenschaft `tweenMode` kann folgende Werte aufweisen:

- `#velocity` ändert die Farbe des Partikels zwischen `colorRange.start` und `colorRange.end` ausgehend von seiner Geschwindigkeit.
- `#age` ändert die Farbe des Partikels durch lineare Interpolation zwischen `colorRange.start` und `colorRange.end` über die Lebenszeit des Partikels hinweg. Dies ist die Standardeinstellung für diese Eigenschaft.

Beispiel

In diesem Beispiel ist `ThermoSystem` eine Modellressource vom Typ `#particle`. Die folgende Anweisung setzt die `ThermoSystem`-Eigenschaft `tweenMode` auf `#velocity`, damit nur die schnelleren Partikel die durch `colorRange`.end angegebene Farbe erreichen, die langsameren jedoch nicht:

```
-- Lingo
member(8).modelResource("thermoSystem").tweenMode = #velocitytype

// Javascript
member(8).getPropRef("modelResource",1).tweenMode =symbol("velocitytype");
```

type (light)

Syntax

```
member(whichCastmember).light(whichLight).type
```

Beschreibung

Diese 3D-Lichteigenschaft gibt die Art des referenzierten Lichts an. Mögliche Werte:

- `#ambient`: Lichtquellen dieser Art beleuchten alle Oberflächen gleichmäßig. Die Stärke von Umgebungslichtern wird nicht durch die Entfernung zur Lichtquelle beeinflusst.
- `#directional` : Lichtquellen dieser Art scheinen in eine bestimmte Richtung, sind aber nicht so gebündelt wie Lichtquellen vom Typ `#spot`. Die Stärke von gerichteten Lichtquellen nimmt mit zunehmender Entfernung von der Lichtquelle ab.
- `#point`: Lichtquellen dieser Art scheinen von einer bestimmten Stelle in der 3D-Welt aus in alle Richtungen. Die Wirkung ist der einer nackten Glühbirne sehr ähnlich. Die Stärke von Punktlichtern nimmt mit zunehmender Entfernung von der Lichtquelle ab.
- `#spot`: Lichtquellen dieser Art scheinen von einem bestimmten Punkt aus innerhalb des durch die Vorwärtsrichtung des Lichts und die Eigenschaft `spotAngle` definierten Kegels. Die Stärke von Spotlichtern nimmt mit zunehmender Entfernung von der Lichtquelle ab, und zwar in Abhängigkeit von den Werten in der Lichteigenschaft `attenuation`.

Beispiel

Die folgende Anweisung zeigt die Eigenschaft „type“ des Lichts `MainLight` an:

```
-- Lingo
put member("3D").motion("MainLight").type
-- #spot

// Javascript
trace(member("3D").getPropRef("motion",1).type);
// symbol("spot")
```

Siehe auch

`spotAngle`, `attenuation`

type (Darsteller)

Syntax

```
-- Lingo syntax
memberObjRef.type

// JavaScript syntax
memberObjRef.type;
```

Beschreibung

Diese Darstellereigenschaft zeigt den Typ eines Darstellers an. Nur Lesen.

Die Eigenschaft `type` kann einen der folgenden Werte besitzen:

#animgif	#palette
#bitmap	#physics
#button	#picture
#cursor	#QuickTimeMedia
#digitalVideo	#realMedia
#DVD	#script
#empty	#shape
#field	#shockwave3D
#filmLoop	#sound
#flash	#swa
#flashcomponent	#text
#font	#transition
#havok	#vectorShape
#movie	#windowsMedia
#ole	#MP4Asset

Diese Liste enthält die Darstellertypen, die in Director und den mitgelieferten Xtra-Erweiterungen verfügbar sind. Für benutzerdefinierte Darsteller können Sie auch benutzerdefinierte Darstellertypen definieren.

Bei Filmen, die in Director 6 und 6.5 erstellt wurden, gibt die Darstellereigenschaft type für Felddarsteller `#field` und für Textdarsteller `#richText` zurück. In allen Versionen ab Director 7 wird für Felddarsteller `#field` und für Textdarsteller `#text` zurückgegeben.

Beispiel

Die folgende Prozedur prüft, ob der Darsteller "Aktuelle Nachrichten“ ein Felddarsteller ist, und zeigt eine Warnmeldung an, wenn dies nicht der Fall ist:

```
-- Lingo syntax
on checkFormat
    if (member("Today's News").type <> #field) then
        _player.alert("Sorry, this cast member must be a field.")
    end if
end

// JavaScript syntax
function checkFormat() {
    if (member("Today's News").type != "field") {
        _player.alert("Sorry, this cast member must be a field.");
    }
}
```

Siehe auch

Member

type (Modellressource)

Syntax

```
member(whichCastmember).modelResource(whichModelResource).type
```

Beschreibung

Diese 3D-Modellressourceneigenschaft gibt den Ressourcentyp der referenzierten Modellressource an. Mögliche Werte:

- `#box` bedeutet, dass diese Modellressource eine primitive Boxressource ist, die mit dem Befehl `newModelResource` erstellt wurde.
- `#cylinder` bedeutet, dass diese Modellressource eine primitive Zylinderressource ist, die mit dem Befehl `newModelResource` erstellt wurde.
- `#extruder` bedeutet, dass diese Modellressource eine primitive Textextruderressource ist, die mit dem Befehl `extrude3d` erstellt wurde.
- `#mesh` bedeutet, dass diese Modellressource eine primitive Gitternetzgeneratorressource ist, die mit dem Befehl `newMesh` erstellt wurde.
- `#particle`bedeutet, dass diese Modellressource eine primitive Partikelsystemressource ist, die mit dem Befehl `newModelResource` erstellt wurde.
- `#plane` bedeutet, dass diese Modellressource eine primitive Ebenenressource ist, die mit dem Befehl `newModelResource` erstellt wurde.
- `#sphere` bedeutet, dass diese Modellressource eine primitive Kugelressource ist, die mit dem Befehl `newModelResource` erstellt wurde.
- `#fromFile` bedeutet, dass diese Modellressource außerhalb von Director erstellt und aus einer externen Datei bzw. aus einem Darsteller geladen wurde.

Beispiel

Die folgende Anweisung zeigt die Eigenschaft type der Modellressource "Helix" an:

```
put member("helix models").modelResource("Helix").type
-- #fromFile
```

Siehe auch

`newModelResource`, `newMesh`, `extrude3D`

type (motion)

Syntax

```
member(whichCastmember).motion(whichMotion).type
```

Beschreibung

Diese 3D-Bewegungseigenschaft gibt den Bewegungstyp des referenzierten Bewegungsobjekts an. Mögliche Werte:

- `#bonesPlayer` gibt an, dass diese Bewegung eine knochenbasierte Animation ist und zur Wiedergabe der Modifizierer `#bonesPlayer` verwendet werden muss.

- `#keyFramePlayer` gibt an, dass diese Bewegung eine schlüsselbildbasierte Animation ist und zur Wiedergabe der Modifizierer `#keyFramePlayer` verwendet werden muss.
- `#none` bedeutet, dass diese Bewegung keine zugeordnete Bewegung hat und mit dem Modifizierer `#bonesPlayer` oder `#keyFramePlayer` wiedergegeben werden kann. Das in jedem 3D-Darsteller zu findende Standardbewegungsobjekt weist diesen Typ auf.

Beispiel

Die folgende Anweisung zeigt die Eigenschaft type der Bewegung "Laufen" an:

```
put member("scene").motion("Run").type
-- #bonesPlayer
```

Die folgende Anweisung zeigt die Eigenschaft type der Bewegung `DefaultMotion` an.

```
put member("scene").motion("DefaultMotion").type
-- #none
```

Siehe auch

`bonesPlayer (Modifizierer)`, `keyframePlayer (Modifizierer)`

type (shader)

Syntax

```
member(whichCastmember).shader(whichShader).type
```

Beschreibung

Diese 3D-Shader-Eigenschaft gibt den Shader-Typ des referenzierten Shader-Objekts an. Mögliche Werte:

- `#standard` gibt an, dass es sich um einen Standard-Shader handelt.
- `#painter` gibt an, dass es sich um einen Maler-Shader handelt.
- `#newsprint` gibt an, dass es sich um einen Zeitungs-Shader handelt.
- `#engraver` gibt an, dass es sich um einen Gravierer-Shader handelt.

Beispiel

Die folgende Anweisung zeigt, dass der vom Modell „Box2“ verwendete Shader ein Maler-Shader (Typ #painter) ist:

```
put member("Scene").model("box2").shader.type
-- #painter
```

Siehe auch

`newShader`

type (sprite)

Syntax

```
sprite(whichSprite).type
the type of sprite whichSprite
```

Beschreibung

Diese Sprite-Eigenschaft löscht Sprite-Kanäle während der Drehbuchaufzeichnung, indem sie den Wert der Sprite-Eigenschaft `type` für diesen Kanal auf 0 setzt.

Hinweis: *Sie sollten den Darsteller eines Sprites nur gegen einen Darsteller des gleichen Typs austauschen, damit beim Darstellerwechsel die Sprite-Eigenschaften nicht geändert werden.*

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Die folgende Anweisung löscht Sprite-Kanal 1, wenn sie während der Drehbuchaufzeichnung ausgeführt wird:

```
sprite(1).type = 0
```

Siehe auch

`beginRecording()`, `updateFrame()`

type (texture)

Syntax

```
member(whichCastmember).shader(whichShader).type
```

Beschreibung

Diese 3D-Textureigenschaft gibt den Texturtyp des referenzierten Texturobjekts an. Mögliche Werte:

- `#fromCastMember` gibt an, dass diese Textur mit dem Befehl `newTexture` aus einem Director-Darsteller erstellt wurde, der die Eigenschaft `image` unterstützt.
- `#fromImageObject` gibt an, dass diese Textur mit dem Befehl `newTexture` aus einem Grafikobjekt erstellt wurde.
- `#importedFromFile` gibt an, dass diese Textur außerhalb von Director erstellt und beim Importieren der Datei bzw. beim Laden des Darstellers in die Anwendung integriert wurde.

Beispiel

Die folgende Anweisung zeigt, dass die vom Shader für das Modell "Pluto" verwendete Textur aus einem Bildobjekt erstellt wurde:

```
put member("scene").model("Pluto").shader.texture.type
-- #fromImageObject
```

Siehe auch

`newTexture`

type (Fenster)

Syntax

```
-- Lingo syntax
windowObjRef.type

// JavaScript syntax
windowObjRef.type;
```

Beschreibung

Diese Fenstereigenschaft gibt den Fenstertyp an. Lesen/Schreiben.

Wenn die Eigenschaft `type` festgelegt ist, werden alle zu dem neuen Fenster gehörenden Eigenschaften entsprechend eingestellt.

Diese Eigenschaft kann einen der folgenden Werte aufweisen:

Eigenschaft	Beschreibung
`#document`	Gibt an, dass das Fenster mit einer Standardtitelleiste, einer Schaltfläche zum Schließen sowie Schaltflächen zum Minimieren und Maximieren angezeigt wird. Diese Fenstertypen können verschoben werden.
`#tool`	Gibt an, dass das Fenster mit einer kürzeren Titelleiste und nur einer kleinen Schaltfläche zum Schließen in der oberen rechten Ecke angezeigt wird. Diese Fenstertypen empfangen keine Aktivierungs- oder Deaktivierungsereignisse mehr, da `#tool`-Fenster immer aktiv sind. Diese Fenstertypen überlagern sich immer gegenseitig und werden immer vor `#document`-Fenstern angezeigt.
`#dialog`	Gibt an, dass das Fenster mit einer Standardtitelleiste, einer Schaltfläche zum Schließen und ohne Symbol angezeigt wird. Diese Fenstertypen sind modal und werden immer vor allen anderen Fenstern angezeigt.

Auf diese Eigenschaften kann ebenfalls unter Verwendung der Eigenschaft `displayTemplate` des Movie-Objekts zugegriffen werden.

Fensterverhalten sind außerdem von den Werten der Eigenschaft `type` und der Eigenschaft `dockingEnabled` des Movie-Objekts abhängig.

- Hat `dockingEnabled` den Wert `TRUE` und ist `type` auf `#document` festgelegt, sieht das MIAW aus wie ein Dokumentfenster in Director und verhält sich auch so. Das Fenster wird im Anzeigebereich angezeigt und kann an die Bühnen-, Drehbuch- und Besetzungsfenster sowie an Medien-Editoren und Nachrichtenfenster angedockt werden. Das Fenster kann aber mit keinem dieser Fenster gruppiert werden.
- Hat `dockingEnabled` den Wert `TRUE` und ist `type` auf `#tool` festgelegt, sieht das MIAW aus wie ein Werkzeugfenster in Director und verhält sich auch so. Das Fenster kann mit allen Werkzeugfenstern gruppiert werden, mit Ausnahme des Eigenschafteninspektors und der Werkzeugpalette.
- Hat `dockingEnabled` den Wert `TRUE` und ist `type` auf `#fullscreen` oder `#dialog` festgelegt, wird der Typ ignoriert und das Fenster ist ein Erstellungsfenster.

Beispiel

Die folgende Anweisung legt den Typ des Fensters namens "Planets" auf `#tool` fest.

```
-- Lingo syntax
window("Planets").type = #tool

// JavaScript syntax
window("Planets").type = "tool";
```

Siehe auch

appearanceOptions, displayTemplate, dockingEnabled, titlebarOptions, Window

updateLock

Syntax

```
-- Lingo syntax
_movie.updateLock

// JavaScript syntax
_movie.updateLock;
```

Beschreibung

Diese Filmeigenschaft bestimmt, ob die Bühne während der Drehbuchaufzeichnung aktualisiert werden soll (FALSE) oder nicht (TRUE). Lesen/Schreiben.

Sie können die Bühnenanzeige während der Drehbuchaufzeichnung konstant lassen, indem Sie die Filmeigenschaft updateLock auf TRUE setzen, bevor das Drehbuch durch Skriptcode aktualisiert wird. Wenn updateLock auf FALSE gesetzt ist, wird die Bühne beim Eintreten in ein Bild aktualisiert, um das neue Bild anzuzeigen.

Sie können mit updateLock auch eine unbeabsichtigte Drehbuchaktualisierung verhindern, wenn Sie ein Bild verlassen, um z. B. Eigenschaften in einem anderen Bild zu untersuchen.

Obwohl mit dieser Eigenschaft während der Laufzeit vorgenommene Bildänderungen maskiert werden können, sollten Sie bedenken, dass Änderungen an Felddarstellern sofort unmittelbar nach Änderung ihres Inhalts angezeigt werden, im Gegensatz zu Positions- oder Darstelleränderungen in anderen Sprites, die erst aktualisiert werden, wenn diese Eigenschaft ausgeschaltet wird.

Beispiel

Die folgende Anweisung zeigt den Wert der Eigenschaft updateLock im Nachrichtenfenster an.

```
-- Lingo
put _movie.updateLock

// Javascript
trace(_movie.updateLock);
```

Siehe auch

Movie

updateMovieEnabled

Syntax

```
the updateMovieEnabled
```

Beschreibung

Diese Filmeigenschaft bestimmt, ob die im aktuellen Film vorgenommenen Änderungen automatisch gespeichert werden, wenn der Film in einen anderen Film verzweigt (TRUE) oder nicht (FALSE, Standard).

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel
Die folgende Anweisung weist Director an, die Änderungen im aktuellen Film jedes Mal dann zu speichern, wenn der Film in einen anderen Film verzweigt:

```
-- Lingo
_movie.updateMovieEnabled = TRUE

// Javascript
_movie.updateMovieEnabled = true;
```

URL

Syntax

```
-- Lingo syntax
memberObjRef.URL

// JavaScript syntax
memberObjRef.URL;
```

Beschreibung
Diese Darstellereigenschaft gibt die URL für Shockwave Audio- (SWA-) und Flash-Filmdarsteller an.

Für Flash-Filmdarsteller ist diese Eigenschaft gleichbedeutend mit der Darstellereigenschaft `pathName`.

Die Eigenschaft `URL` kann getestet und eingestellt werden. Für SWA-Darsteller kann sie nur eingestellt werden, wenn der SWA-Streaming-Darsteller gestoppt wurde.

Beispiel
Die folgende Anweisung macht eine Datei auf einem Internetserver zur URL für den SWA-Darsteller „Benny Goodman“:

```
-- Lingo syntax
on mouseDown
    member("Benny Goodman").URL = "http://audio.adobe.com/samples/classic.swa"
end

// JavaScript syntax
function mouseDown() {
    member("Benny Goodman").URL = "http://audio.adobe.com/samples/classic.swa"
}
```

useAlpha

Syntax

```
-- Lingo syntax
memberObjRef.useAlpha
imageObjRef.useAlpha

// JavaScript syntax
memberObjRef.useAlpha;
imageObjRef.useAlpha;
```

Beschreibung

Diese Bitmapdarsteller- und Grafikobjekteigenschaft bestimmt bei 32-Bit-Darstellern und -Bildobjekten mit Alphakanal-Informationen, ob Director beim Zeichnen der Grafik auf der Bühne die Alphakanal-Informationen verwendet (`TRUE`) oder ignoriert (`FALSE`).

Beispiel

Durch die folgende Anweisung wird der Alphakanal des Darstellers „Vordergrund“ ein- und ausgeschaltet:

```
-- Lingo syntax
member("foreground").useAlpha = not(member("foreground").useAlpha)

// JavaScript syntax
switch(member("foreground").useAlpha) {
    case 0:
        member("foreground").useAlpha = 1;
        break;
    case 1:
        member("foreground").useAlpha = 0;
        break;
    }
}
```

useDiffuseWithTexture

Syntax

```
member(whichCastmember).shader(whichShader).useDiffuseWithTexture
```

Beschreibung

Mit dieser 3D-Eigenschaft für Standard-Shader können Sie ermitteln und festlegen, ob die Streulichtfarbe zum Modulieren der Textur verwendet wird (`TRUE`) oder nicht (`FALSE`).

Mit dem Wert `TRUE` funktioniert diese Eigenschaft zusammen mit den Eigenschaften `blendFunction` und `blendConstant`: Wenn `blendFunction` auf `#blend` eingestellt ist, wird die Streulichtfarbe mit der Texturfarbe gemischt, um die endgültige Farbe zu bestimmmen. Ist `blendFunction` beispielsweise auf `#blend` und `blendConstant` auf 100.0 gesetzt, wird als endgültige Farbe die Texturfarbe verwendet. Ist `blendConstant` auf 0.0 gesetzt, wird als endgültige Farbe die Streulichtfarbe verwendet. Ist `blendConstant` auf 10.0 gesetzt, setzt sich die endgültige Farbe zu 10% aus der Texturfarbe und zu 90% aus der Streulichtfarbe zusammen.

Der Standardwert dieser Eigenschaft lautet `FALSE`.

Alle Shader können auf die Eigenschaften des `#standard`-Shaders zugreifen. Darüber hinaus haben die Shader `#engraver`, `#newsprint` und `#painter` noch weitere Eigenschaften, die speziell für den jeweiligen Shader-Typ gelten. Weitere Informationen hierzu finden Sie unter `newShader`.

Beispiel

In diesem Beispiel enthält die Eigenschaft `shaderList` des Modells `MysteryBox` sechs Shader. Jeder Shader verfügt über eine Texturliste mit bis zu acht Texturen. Die Eigenschaft `diffuseColor` des Darstellers (Ebene2) ist auf "rgb(255, 0, 0)" gesetzt. Bei allen sechs Shadern ist die Eigenschaft `blendFunction` auf `#blend` und die Eigenschaft `blendConstant` auf "80" gesetzt. Die folgende Anweisung setzt die Eigenschaft `useDiffuseWithTexture` aller von `MysteryBox` verwendeten Shader auf "TRUE". Etwas Rot wird in die Modelloberfläche einfließen. Diese Eigenschaft wird durch die Einstellungen für die Eigenschaften `blendFunction`, `blendFunctionList`, `blendSource`, `blendSourceList`, `blendConstant` und `blendConstantList` beeinflusst.

```
-- Lingo
member("Level2").model("MysteryBox").shaderlist.useDiffuseWithTexture = TRUE

// Javascript
member("Level2").getPropRef("model",1).getProp("shaderlist").useDiffuseWithTexture = true;
```

Siehe auch

`blendFunction`, `blendConstant`

useFastQuads

Syntax

```
-- Lingo syntax
_movie.useFastQuads

// JavaScript syntax
_movie.useFastQuads;
```

Beschreibung

Diese Filmeigenschaft bestimmt, ob schnellere (`TRUE`) oder langsamere (`FALSE`, Standard) Quad-Berechnungsoperationen verwendet werden sollen. Lesen/Schreiben.

Bei Festlegung auf `TRUE` verwendet Director eine schnellere, aber ungenauere Methode zur Berechnung von Quad-Operationen. Schnelle Quad-Berechnungen empfehlen sich für einfache Dreh- und Neigungseffekte bei Sprites.

Bei Festlegung auf `FALSE` verwendet Director die langsamere Quad-Standardberechnungsmethode, optisch ansprechendere Ergebnisse liefert, wenn Quads für Verzerrungen und andere willkürliche Effekte benutzt werden.

Für einfache Dreh- und Neigungsoperationen bei Sprites wird immer die schnelle Quad-Berechnungsmethode verwendet, und zwar unabhängig von dieser Einstellung. Diese einfachen Operationen lassen sich durch Einstellen von `useFastQuads` auf `TRUE` nicht beschleunigen.

Beispiel

Die folgende Anweisung bewirkt, dass Director für alle Quad-Operationen im Film den schnelleren Quad-Berechnungscode verwendet:

```
- Lingo syntax
_movie.useFastQuads = TRUE

// JavaScript syntax
_movie.useFastQuads = true;
```

Siehe auch

Movie

useHypertextStyles

Syntax

```
-- Lingo syntax
memberObjRef.useHypertextStyles

// JavaScript syntax
memberObjRef.useHypertextStyles;
```

Beschreibung

Diese Textdarstellereigenschaft steuert die Anzeige von Hypertext-Links im Textdarsteller.

Wenn `useHypertextStyles` auf `TRUE` gesetzt ist, werden alle Links automatisch blau und unterstrichen angezeigt, und der Cursor ändert sich über einem Link in einen Zeigefinger.

Um die automatische Formatierung und Cursoränderung zu deaktivieren, setzen Sie diese Eigenschaft auf `FALSE`.

Beispiel

Das folgende Verhalten schaltet die Formatierung von Links im Textdarsteller `myText`:

```
--Lingo syntax
on mouseUp
    member("myText").usehypertextStyles = not(member("myText").usehypertextStyles)
end

// JavaScript syntax
function mouseUp() {
    member("myText").usehypertextStyles = !(member("myText").usehypertextStyles)
}
```

useLineOffset

Syntax

```
member(whichCastmember).model(whichModel).toon.useLineOffset
member(whichCastmember).model(whichModel).inker.useLineOffset
```

Beschreibung

Diese 3D-Eigenschaft für die Modifizierer `toon` und `inker` gibt an, ob die Eigenschaft `lineOffset` vom Modifizierer beim Zeichnen von Linien auf der Modelloberfläche verwendet wird.

Der Standardwert dieser Eigenschaft lautet `FALSE`.

Beispiel

Die folgende Anweisung setzt die Eigenschaft `useLineOffset` des auf das Modell "Teekanne" angewendeten Modifizierers toon auf FALSE. Die Eigenschaft `lineOffset` des Modifizierers toon hat keine Wirkung.

```
-- Lingo syntax
member("tp").model("Teapot").toon.useLineOffset = FALSE

// JavaScript syntax
member("tp").getPropRef("model",1).getProp("toon").useLineOffset = false;
```

Siehe auch

`lineOffset`

userData

Syntax

```
member(whichCastmember).model(whichModel).userData
member(whichCastmember).light(whichLight).userData
member(whichCastmember).camera(whichCamera).userData
member(whichCastmember).group(whichCamera).userData
```

Beschreibung

Diese 3D-Eigenschaft gibt die `userData`-Eigenschaftsliste eines Modells, einer Gruppe, einer Kamera oder eines Lichts zurück. Bei einem außerhalb von Director erstellten Objekt ist der Standardwert dieser Eigenschaft eine Liste aller Eigenschaften, die der Eigenschaft `userData` des Modells im 3D-Modellierungsprogramm zugeordnet wurden. Bei einem in Director erstellten Objekt ist der Standardwert dieser Eigenschaft eine leere Eigenschaftsliste ([:]), es sei denn, das Objekt wurde mit einem Klonbefehl erstellt. Wenn zur Erstellung des Objekts ein Klonbefehl verwendet wurde, weist die Eigenschaft `userData` des neuen Objekts einen Wert auf, der dem des ursprünglichen Quellobjekts entspricht.

Um die Elemente in dieser Liste zu ändern, müssen Sie die Befehle `addProp` und `deleteProp` verwenden.

Beispiel

Die folgende Anweisung zeigt die Eigenschaft `userData` des Modells "neueKarosserie" an:

```
put member("Car").model("New Body").userData
-- [#driver: "Bob", #damage: 34]
```

Die folgende Anweisung für die Eigenschaft `#health` mit dem Wert 100 zur `userData`-Eigenschaftsliste des Modells „Player“ hinzu:

```
member("scene").model("Player").userData.addProp(#health,100)
```

userName

Syntax

```
-- Lingo syntax
_player.userName

// JavaScript syntax
_player.userName;
```

Beschreibung

Diese Eigenschaft ist ein String, der den bei der Installation von Director eingegebenen Benutzernamen enthält.
Schreibgeschützt

Diese Eigenschaft ist nur in der Authoring-Umgebung verfügbar. Sie kann in einem MIAW (Film in einem Fenster)-Werkzeug verwendet werden, dass die persönlichen Angaben eines Benutzers anzeigt.

Beispiel

Die folgende Prozedur stellt den Benutzernamen und die Seriennummer in ein Anzeigefeld, wenn das Fenster geöffnet wird. (Ein Filmskript im MIAW eignet sich besonders für diese Prozedur.)

```
-- Lingo syntax
on prepareMovie
    displayString = _player.userName & RETURN & _player.organizationName & RETURN &
_player.serialNumber
    member("User Info").text = displayString
end

// JavaScript syntax
function prepareMovie() {
    var displayString = _player.userName + "\n" + _player.organizationName+ "\n" +
_player.serialNumber;
    member("User Info").text = displayString;
}
```

Siehe auch

Player

userName (RealMedia)

Syntax

```
-- Lingo syntax
memberOrSpriteObjRef.userName

// JavaScript syntax
memberOrSpriteObjRef.userName;
```

Beschreibung

Mit dieser Eigenschaft für `RealMedia`-Sprites und -Darsteller können Sie den Benutzernamen festlegen, der zum Zugriff auf einen geschützten `RealMedia`-Stream erforderlich ist. Aus Sicherheitsgründen können Sie mit dieser Eigenschaft keinen bereits festgelegten Benutzernamen abrufen. Wenn bereits ein Benutzername festgelegt ist, gibt diese Eigenschaft den String `"********"` zurück. Der Standardwert dieser Eigenschaft ist eine leere Liste, d. h. es wurde noch kein Benutzername festgelegt.

Beispiel

Die folgenden Beispiele zeigen, dass für den `RealMedia`-Stream im Darsteller „Real“ bzw. in Sprite 2 ein Benutzername festgelegt wurde:

```
-- Lingo syntax
put(sprite(2).userName) -- "********"
put(member("Real").userName) -- "********"

// JavaScript syntax
put(sprite(2).userName); // "********"
put(member("Real").userName); // "********"
```

Die folgenden Beispiele zeigen, dass für den `RealMedia`-Stream im Darsteller „Real“ bzw. in Sprite 2 noch kein Benutzername festgelegt wurde.

```
-- Lingo syntax
put(sprite(2).userName) -- ""
put(member("Real").userName) -- ""

// JavaScript syntax
put(sprite(2).userName); // ""
put(member("Real").userName); // ""
```

In den folgenden Beispielen wird der Benutzername für den `RealMedia`-Stream im Darsteller „Real“ und in Sprite 2 auf „Marcelle“ gesetzt.

```
-- Lingo syntax
member("Real").userName = "Marcelle"
sprite(2).userName = "Marcelle"

// JavaScript syntax
member("Real").userName = "Marcelle";
sprite(2).userName = "Marcelle";
```

Siehe auch

password

useTargetFrameRate

Syntax

```
sprite(which3dSprite).useTargetFrameRate
```

Beschreibung

Diese 3D-Sprite-Eigenschaft bestimmt, ob die Eigenschaft `targetFrameRate` des Sprites verwendet wird. Ist die Eigenschaft `useTargetFrameRate` auf `TRUE` gesetzt, wird die Polygonanzahl der Modelle im Sprite entsprechend der angegebenen Zielbildrate reduziert.

Beispiel

Die folgenden Anweisungen setzen die Eigenschaft `targetFrameRate` von Sprite 3 auf 45 und sorgen dafür, dass die Bildrate verwendet wird (`useTargetFrameRate` = TRUE):

```
-- Lingo syntax
sprite(3).targetFrameRate = 45
sprite(3).useTargetFrameRate = TRUE

// JavaScript syntax
sprite(3).targetFrameRate = 45;
sprite(3).useTargetFrameRate = true;
```

Siehe auch

`targetFrameRate`

vertex

Syntax

```
-- Lingo syntax
memberObjRef.vertex[whichVertexPosition]

// JavaScript syntax
memberObjRef.vertex[whichVertexPosition];
```

Beschreibung

Dieser Chunk-Ausdruck ermöglicht den direkten Zugriff auf bestimmte Teile der Scheitelpunktliste eines Vektorformdarstellers.

Anhand dieses Chunk-Ausdrucks können Sie verhindern, dass unterschiedliche Chunks der Scheitelpunktliste geparst werden. Sie können mit diesem Chunk-Ausdruck Werte in der Scheitelpunktliste testen und festlegen.

Beispiel

Anhand des folgenden Codes können Sie die Anzahl von Scheitelpunkten in einem Darsteller ermitteln:

```
-- Lingo syntax
put(member("Archie").vertex.count) -- 2

// JavaScript syntax
put(member("Archie").vertex.count); // 2
```

Mit dem folgenden Code können Sie den zweiten Scheitelpunkt für den Darsteller abrufen:

```
-- Lingo syntax
put(member("Archie").vertex[2]) -- point(66.0000, -5.0000)

// JavaScript syntax
put(member("Archie").vertex[2]); // point(66.0000, -5.0000)
```

Sie können auch den Wert in einem Anfasser einstellen:

```
-- Lingo syntax
member("Archie").vertex[2].handle1 = point(-63.0000, -16.0000)

// JavaScript syntax
member("Archie").vertex[2].handle1 = point(-63.0000, -16.0000);
```

Siehe auch

vertexList

vertexList

Syntax

```
-- Lingo syntax
memberObjRef.vertexList

// JavaScript syntax
memberObjRef.vertexList;
```

Beschreibung

Diese Darstellereigenschaft gibt eine lineare Liste zurück, die Eigenschaftslisten enthält (eine für jeden Scheitelpunkt einer Vektorform). Die Eigenschaftsliste enthält die Position des Scheitelpunkts und des Anfassers. Wenn die Position (0,0) lautet, gibt es keine Anfasser.

Jeder Scheitelpunkt kann zwei Anfasser besitzen, die die Kurve zwischen diesem Scheitelpunkt und den angrenzenden Scheitelpunkten bestimmen. In `vertexList` werden die Koordinaten der Anfasser für einen Scheitelpunkt relativ zu diesem Scheitelpunkt – und nicht absolut im Koordinatensystem der Form – gemessen. Befindet sich der erste Anfasser eines Scheitelpunkts 10 Pixel links vor diesem Scheitelpunkt, wird seine Position als (-10, 0) gespeichert. Wenn die Position eines Scheitelpunkts mit Lingo geändert wird, bewegen sich die Anfasser zusammen mit dem Scheitelpunkt und müssen daher nicht aktualisiert werden (es sei denn, der Benutzer möchte die Position oder Größe des Anfassers ändern).

Beachten Sie beim Modifizieren dieser Eigenschaft, dass Sie den Listeninhalt zurücksetzen müssen, wenn Sie die Werte ändern. Der Grund: Wenn Sie eine Variable auf den Wert einer Eigenschaft setzen, stellen Sie eine Kopie der Liste – und nicht die Liste selbst – in die Variable. Um eine Änderung durchzuführen, verwenden Sie folgende Syntax:

```
- Get the current property contents
currVertList = member(1).vertexList
-- Add 25 pixels to the horizontal and vertical positions of the first vertex in the list
currVertList[1] .vertex = currVertList[1] .vertex + point(25, 25)
-- Reset the actual property to the newly computed position
member(1).vertexList = currVertList
```

Beispiel

Die folgende Anweisung zeigt den `vertextList`-Wert für eine gebogene Linie mit zwei Scheitelpunkten an:

```
-- Lingo syntax
put(member("Archie").vertexList)-- [[#vertex: point(-66.0000, 37.0000), #handle1: point(-
70.0000, -36.0000), #handle2: point(-62.0000, 110.0000)], [#vertex: point(66.0000, -5.0000),
#handle1: point(121.0000, 56.0000), #handle2: point(11.0000, -66.0000)]]

// JavaScript syntax
put(member("Archie").vertexList);//[[#vertex: point(-66.0000, 37.0000),#handle1: point(-
70.0000, -36.0000),#handle2: point(-62.0000, 110.0000)], [#vertex: point(66.0000, -
5.0000),#handle1: point(121.0000, 56.0000), #handle2: point(11.0000, -66.0000)]]
```

Siehe auch

addVertex(), count(), deleteVertex(), moveVertex(), moveVertexHandle(), originMode, vertex

vertexList (mesh generator)

Syntax

```
member(whichCastmember).modelResource(whichModelResource).vertexList
```

Beschreibung

Mit dieser 3D-Eigenschaft können Sie bei einer Modellressource vom Typ `#mesh` die Eigenschaft `vertexList` der Modellressource ermitteln und festlegen.

Die `vertexList` ist eine lineare Liste aller im Gitternetz verwendeten Scheitelpunkte. Ein einzelner Scheitelpunkt kann von zahlreichen Gitternetzflächen benutzt werden. Für diese Eigenschaft können Sie zwar eine beliebige lange Liste angeben, doch es werden nur so viele Einträge gespeichert, wie bei Erstellung der `#mesh`-Modellressource mit dem Befehl `newMesh()` angegeben wurden.

Beispiel

Die folgende Anweisung definiert die `vertexList` der Modellressource „Dreieck“:

```
member("Shapes").modelResource("Triangle").vertexList = [vector(0,0,0), vector(20,0,0),
vector(20, 20, 0)]
```

Siehe auch

newMesh, face, vertices

vertexList (mesh deform)

Syntax

```
member(whichCastmember).model(whichModel).meshDeform.mesh[index].vertexList
```

Beschreibung

Mit dieser 3D-Eigenschaft können Sie bei einem Modell, an dem der Modifizierer `#meshDeform` angebracht ist, die Eigenschaft `vertexList` für das angegebene Gitternetz im referenzierten Modell ermitteln und festlegen.

Die `vertexList` ist eine lineare Liste aller im Gitternetz verwendeten Scheitelpunkte. Ein einzelner Scheitelpunkt kann von zahlreichen Gitternetzflächen benutzt werden.

Wenn ein Modell zusätzlich zum Modifizierer `#meshDeform` den Modifizierer `#sds` oder `#lod` verwendet, wird der Wert dieser Eigenschaft durch den Modifizierer `#sds` oder `#lod` beeinflusst.

Beispiel

Die folgende Anweisung zeigt die `vertexList` des Modifizierer `#meshDeform` für das erste Gitternetz im Modell „Dreieck“ an:

```
put member("Shapes").model("Triangle").meshDeform.mesh[1].vertexList
-- [vector(0,0,0), vector(20,0,0), vector(20, 20, 0)]
```

Siehe auch

face, vertices, mesh (Eigenschaft)

vertices

Syntax

```
member(whichCastmember).modelResource(whichModelResource).face[faceIndex].vertices
```

Beschreibung

Mit dieser 3D-Flächeneigenschaft können Sie bei einer Modellressource vom Typ `#mesh` ermitteln und festlegen, welche Scheitelpunkte aus der `vertexList` der Ressource für die durch *faceIndex* angegebene Gitternetzfläche verwendet werden.

Diese Eigenschaft ist eine lineare Liste aus drei Integerwerten, die den Indexpositionen der drei Scheitelpunkte, aus denen sich die angegebene Fläche zusammensetzt, in der Eigenschaft `vertexList` des Gitternetzes entsprechen.

Die Scheitelpunkte müssen in der Liste gegen den Uhrzeigersinn angegeben werden, damit eine nach außen zeigende Oberflächennormale entsteht.

Wenn Sie diese Eigenschaft ändern oder den Befehl `generateNormals()` verwenden, müssen Sie den Befehl `build()` zur Neuerstellung des Gitternetzes aufrufen.

Beispiel

Das folgende Beispiel zeigt zuerst die `vertexList`der Gitternetzmodellressource `SimpleSquare` und dann die Eigenschaft vertices der zweiten Fläche dieses Gitternetzes an:

```
put member("3D").modelResource("SimpleSquare").vertexList
-- [vector( 0.0000, 0.0000, 0.0000), vector( 0.0000, 5.0000, 0.0000), vector( 5.0000, 0.0000,
0.0000), vector( 5.0000, 5.0000, 0.0000)]
put member("3D").modelResource("SimpleSquare").face[1].vertices
-- [3, 4, 1]
```

Siehe auch

face, vertexList (mesh deform), generateNormals()

video (QuickTime, AVI)

Syntax

```
member(whichCastMember).video
the video of member whichCastMember
```

Beschreibung

Diese Digitalvideo-Darsteller-Eigenschaft bestimmt, ob die Grafik eines bestimmten Digitalvideo-Darsteller abgespielt wird (`TRUE` oder 1) oder nicht (`FALSE` oder 0).

Hiervon wird nur das visuelle Element des Digitalvideo-Darstellers betroffen. Wenn beispielsweise die Darstellereigenschaft `video` auf `FALSE` gesetzt ist, wird die Tonspur des Digitalvideos (sofern vorhanden) weiterhin abgespielt.

Beispiel

Die folgende Anweisung schaltet das mit dem Darsteller "Interview“ verbundene Video aus:

```
-- Lingo syntax
member("Interview").video = FALSE

// JavaScript syntax
member("Interview").video = false;
```

Siehe auch

`setTrackEnabled()`, `trackEnabled`

video (MP4Media/FLV)

Syntax

```
member(1).video = true
sprite(1).video = true
```

Beschreibung

MP4Media/FLV-Darsteller- und Sprite-Eigenschaft, die festlegt, ob der angegebene MP4Media/FLV-Darsteller abgespielt wird (True bzw. 1) oder nicht (False bzw. 0).

Hiervon wird nur das visuelle Element des Digitalvideo-Darstellers betroffen. Die Tonspur des Digitalvideos wird weiterhin abgespielt.

Beispiele

Das folgende Beispiel schaltet das mit dem Darsteller `Interview` verbundene Video aus:

```
-- Lingo syntax
member("Interview").video = FALSE

// JavaScript syntax
member("Interview").video = false;
```

video (RealMedia, Windows Media)

Syntax

```
-- Lingo syntax
memberOrSpriteObjRef.video

// JavaScript syntax
memberOrSpriteObjRef.video;
```

Beschreibung

Mit dieser `RealMedia`- und Windows Media-Eigenschaft können Sie festlegen und ermitteln, ob das Sprite bzw. der Darsteller Video rendert (`TRUE` bzw. 1) oder nur die Sounds wiedergibt (`FALSE` bzw. 0). Lesen/Schreiben.

Andere Ganzzahlen als 1 oder 0 werden als `TRUE` interpretiert.

Mit dieser Eigenschaft können Sie die Videoanzeige beim Abspielen der Audiokomponente eines `RealMedia`- oder Windows Media-Darstellers unterdrücken bzw. die Videoanzeige während der Wiedergabe ein- und ausschalten.

Beispiel

Die folgenden Beispiele zeigen, dass die Eigenschaft video für Sprite 2 und den Darsteller "Real" auf TRUE gesetzt wird.

```
-- Lingo syntax
put(sprite(2).video) -- 1
put(member("Real").video) -- 1

// JavaScript syntax
put(sprite(2).video); // 1
put(member("Real").video); // 1
```

In den folgenden Beispielen wird die Eigenschaft video für das `RealMedia`-Videoelement von Sprite 2 und des Darstellers „Real“ auf FALSE gesetzt:

```
-- Lingo syntax
sprite(2).video = FALSE
member("Real").video = FALSE

// JavaScript syntax
sprite(2).video = 0;
member("Real").video = 0;
```

videoFormat

Syntax

```
-- Lingo syntax
dvdObjRef.videoFormat

// JavaScript syntax
dvdObjRef.videoFormat;
```

Beschreibung

Diese DVD-Eigenschaft gibt ein Symbol zurück, das das Videoformat angibt. Nur Lesen.

Folgende Symbole sind möglich:

Symbol	Beschreibung
#MPEG1	Das Videoformat ist MPEG-1.
#MPEG2	Das Videoformat ist MPEG-2.
#unknown	Das Videoformat ist unbekannt.

Beispiel

Die folgende Anweisung gibt das Videoformat des DVD-Objekt-Sprites zurück.

```
-- Lingo syntax
put sprite(1).videoFormat

// JavaScript syntax
trace(sprite(1).videoFormat)
```

Siehe auch

DVD

videoForWindowsPresent

Syntax

```
the videoForWindowsPresent
```

Beschreibung

Diese Systemeigenschaft gibt an, ob AVI-Software auf dem Computer vorhanden ist.

Diese Eigenschaft kann getestet, aber nicht eingestellt werden.

Beispiel

Die folgende Anweisung prüft, ob Video für Windows fehlt. Wenn nicht, verzweigt der Abspielkopf zur Markierung "Szene wechseln“.

```
if the videoForWindowsPresent= FALSE then go to "Alternate Scene"
```

Siehe auch

QuickTimeVersion()

viewH

Syntax

```
-- Lingo syntax
memberOrSpriteObjRef.viewH

// JavaScript syntax
memberOrSpriteObjRef.viewH;
```

Beschreibung

Diese Darsteller- und Sprite-Eigenschaft steuert die horizontale Koordinate des Ansichtspunkts eines Flash-Films und einer Vektorform und gibt diese in Pixeleinheiten an. Die Werte können Fließkommazahlen sein. Der Standardwert ist 0.

Der Ansichtspunkt eines Flash-Films wird relativ zu seinem Ursprungspunkt eingestellt.

Bei Einstellung eines positiven Wertes für `viewH` wird der Film im Sprite nach links versetzt, bei einem negativen Wert nach rechts. Die Änderung der Eigenschaft `viewH` kann somit zum Beschneiden des Films führen oder den Film ganz aus der Ansicht entfernen.

Diese Eigenschaft kann getestet und eingestellt werden.

Hinweis: *Diese Eigenschaft muss auf den Standardwert eingestellt sein, wenn die Eigenschaft* `scaleMode` *auf* `#autoSize` *gesetzt ist. Andernfalls wird das Sprite nicht richtig angezeigt.*

Beispiel

Die folgende Prozedur akzeptiert einen Sprite-Bezug als Parameter und verschiebt die Ansicht eines Flash-Film-Sprites im Begrenzungsrechteck des Sprites von links nach rechts:

```
-- Lingo syntax
on panRight whichSprite
    repeat with i = 120 down to -120
        sprite(whichSprite).viewH = i
        _movie.updateStage()
    end repeat
end

// JavaScript syntax
function panRight(whichSprite) {
    var i = 120;
    while(i > -121) {
        sprite(whichSprite).viewH = i;
        _movie.updateStage();
        i--;
    }
}
```

Siehe auch

`scaleMode`, `viewV`, `viewPoint`, `viewScale`

viewPoint

Syntax

```
-- Lingo syntax
memberOrSpriteObjRef.viewPoint

// JavaScript syntax
memberOrSpriteObjRef.viewPoint;
```

Beschreibung

Diese Darsteller- und Sprite-Eigenschaft steuert den Punkt in einem Flash-Film oder einer Vektorform, der in der Mitte des Begrenzungsrechtecks des Sprites angezeigt wird. Der Punkt wird in Pixeleinheiten angegeben. Die Werte sind Ganzzahlen.

Wenn Sie den Ansichtspunkt eines Darstellers ändern, ändern Sie nur die Ansicht des Films im Sprite-Begrenzungsrechteck, nicht die Position des Sprites auf der Bühne. Der Ansichtspunkt ist die Koordinate innerhalb eines Darstellers, die in der Mitte des Sprite-Begrenzungsrechtecks angezeigt und immer relativ zum Ausgangspunkt des Films ausgedrückt wird (wie in den Eigenschaften `originPoint`, `originH` und `originV` eingestellt). Wenn Sie beispielsweise den Ansichtspunkt eines Flash-Films auf point (100,100) einstellen, ist die Mitte des Sprites im Flash-Film der Punkt, der sich 100 Flash-Film-Pixeleinheiten rechts vom und 100 Flash-Film-Pixeleinheiten unterhalb des Ursprungspunktes befindet, unabhängig davon, wohin Sie den Ursprungspunkt verschoben haben.

Die Eigenschaft `viewPoint` wird als Director-Punktwert angegeben, Die Einstellung des Ansichtspunkts für einen Flash-Film mit der Eigenschaft `viewPoint` hat die gleiche Wirkung, als wenn Sie die Eigenschaften `viewH` und `viewV` separat einstellen. Wenn Sie beispielsweise die Eigenschaft `viewPoint` auf (50,75) setzen, erzielen Sie das gleiche Ergebnis wie bei Einstellung der Eigenschaft `viewH` auf 50 und `viewV` auf 75.

Director-Punktwerte, die für die Eigenschaft `viewPoint` angegeben werden, sind auf Ganzzahlen beschränkt. Im Gegensatz dazu können `viewH` und `viewV` als Fließkommazahlen angegeben werden. Beim Testen der Eigenschaft `viewPoint` werden Punktwerte auf Ganzzahlen gekürzt. Grundsätzlich gilt: Verwenden Sie die Eigenschaften `viewH` und `viewV` aus Gründen der Genauigkeit und die Eigenschaft `originPoint`, weil sie praktisch und schnell ist.

Diese Eigenschaft kann getestet und eingestellt werden. Der Standardwert ist „point(0,0)“.

Hinweis: *Diese Eigenschaft muss auf den Standardwert eingestellt sein, wenn die Eigenschaft* `scaleMode` *auf* `#autoSize` *gesetzt ist. Andernfalls wird das Sprite nicht richtig angezeigt.*

Beispiel

Die folgende Prozedur bewirkt, dass ein bestimmtes Flash-Film-Sprite in Schritten von jeweils fünf Flash-Film-Pixeleinheiten nach unten und nach rechts verschoben wird:

```
-- Lingo syntax
on panAcross(whichSprite)
    repeat with i = 1 to 10
        sprite(whichSprite).viewPoint = sprite(whichSprite).viewPoint + point(i * -5, i * -5)
        _movie.updateStage()
    end repeat
end

// JavaScript syntax
function panAcross(whichSprite) {
    var i = 1;
    while(i < 11) {
        sprite(whichSprite).viewPoint = sprite(whichSprite).viewPoint + point(i * -5, i * -5);
        _movie.updateStage();
        i++
    }
}
```

Siehe auch

`scaleMode`, `viewV`, `viewH`, `viewScale`

viewScale

Syntax

```
-- Lingo syntax
memberOrSpriteObjRef.viewScale

// JavaScript syntax
memberOrSpriteObjRef.viewScale;
```

Beschreibung

Diese Darsteller- und Sprite-Eigenschaft bestimmt, um welchen Faktor die Ansicht eines Flash-Films oder Vektorform-Sprites im Sprite-Begrenzungsrechteck insgesamt skaliert werden soll. Der Skalierungsfaktor wird mithilfe einer Fließkommazahl als Prozentwert angegeben. Der Standardwert ist 100.

Das Sprite-Rechteck selbst wird nicht skaliert, nur die Ansicht des Darstellers im Rechteck. Die Einstellung der Eigenschaft `viewScale` für das Sprite ist vergleichbar mit der Auswahl eines Kameraobjektivs. Je mehr sich der `viewScale`-Wert verkleinert, desto größer wird die scheinbare Größe des Films innerhalb des Sprites, und umgekehrt. Wenn Sie z. B. die Eigenschaft `viewScale` auf 200% setzen, ist im Sprite doppelt so viel wie vorher zu sehen, und der Darsteller im Sprite erscheint halb so groß wie in der Originalgröße.

Der Hauptunterschied zwischen den Eigenschaften `viewScale` und `scale` liegt darin, dass bei `viewScale` die Skalierung immer von der Mitte des Sprite-Begrenzungsrechtecks ausgeht, während bei `scale` die Skalierung von dem Punkt ausgeht, der in der Flash-Filmeigenschaft `originMode` angegeben ist.

Diese Eigenschaft kann getestet und eingestellt werden.

Hinweis: *Diese Eigenschaft muss auf den Standardwert eingestellt sein, wenn die Eigenschaft `scaleMode` auf `#autoSize` gesetzt ist. Andernfalls wird das Sprite nicht richtig angezeigt.*

Beispiel

Das folgende Sprite-Skript richtet ein Flash-Film-Sprite ein und verdoppelt die Ansichtsskalierung:

```
-- Lingo syntax
property spriteNum

on beginSprite me
    sprite(spriteNum).viewScale = 200
end

// JavaScript syntax
function beginSprite() {
    sprite(this.spriteNum).viewScale = 200;
}
```

Siehe auch

scaleMode, viewV, viewPoint, viewH

viewV

Syntax

```
-- Lingo syntax
memberOrSpriteObjRef.viewV

// JavaScript syntax
memberOrSpriteObjRef.viewV;
```

Beschreibung

Diese Darsteller- und Sprite-Eigenschaft steuert die vertikale Koordinate des Ansichtspunkts eines Flash-Films und einer Vektorform. Die Koordinate wird in Pixeleinheiten angegeben. Die Werte können Fließkommazahlen sein. Der Standardwert ist 0.

Der Ansichtspunkt eines Flash-Films wird relativ zu seinem Ursprungspunkt eingestellt.

Bei Einstellung eines positiven Wertes für `viewV` wird der Film im Sprite nach oben verschoben, bei einem negativen Wert nach unten. Die Änderung der Eigenschaft `viewV` kann somit zum Beschneiden des Films führen oder den Film ganz aus der Ansicht entfernen.

Diese Eigenschaft kann getestet und eingestellt werden.

Hinweis: *Diese Eigenschaft muss auf den Standardwert eingestellt sein, wenn die Eigenschaft* `scaleMode` *auf* `#autoSize` *gesetzt ist. Andernfalls wird das Sprite nicht richtig angezeigt.*

Beispiel

Die folgende Prozedur akzeptiert ein Sprite-Bezug als Parameter und verschiebt die Ansicht eines Flash-Film-Sprites im Sprite-Begrenzungsrechteck von oben nach unten.

```
-- Lingo syntax
on panDown(whichSprite)
    repeat with i = 120 down to -120
        sprite(whichSprite).viewV = i
        _movie.updateStage()
    end repeat
end

// JavaScript syntax
function panDown(whichSprite) {
    var i = 120;
    while(i > -121) {
        sprite(whichSprite).viewV = i;
        _movie.updateStage();
        i--;
    }
}
```

Siehe auch

`scaleMode`, `viewPoint`, `viewH`

visible

Syntax

```
-- Lingo syntax
windowObjRef.visible

// JavaScript syntax
windowObjRef.visible;
```

Beschreibung

Diese Fenstereigenschaft bestimmt, ob ein Fenster sichtbar ist (`TRUE`) oder nicht (`FALSE`). Lesen/Schreiben.

Beispiel

Die folgende Anweisung macht das Fenster "Steuerpult“ sichtbar:

```
-- Lingo syntax
window("Control_Panel").visible = TRUE

// JavaScript syntax
window("Control_Panel").visible = true;
```

Siehe auch

`Window`

visible (Sprite)

Syntax

```
sprite(whichSprite).visible
the visible of sprite whichSprite
```

Beschreibung

Diese Sprite-Eigenschaft bestimmt, ob das in *whichSprite* angegebene Sprite sichtbar ist (`TRUE`) oder nicht (`FALSE`). Diese Eigenschaft betrifft alle Sprites im Kanal, unabhängig von ihrer Position im Drehbuch.

Hinweis: *Wenn Sie die Eigenschaft `visible` für einen Sprite-Kanal auf `FALSE` setzen, wird das Sprite unsichtbar, und es werden keine mausspezifischen Ereignisse an diesen Kanal gesendet. Die Ereignisse `beginSprite`, `endSprite`, `prepareFrame`, `enterFrame` und `exitFrame` werden weiterhin gesendet und sind nicht von der Sichtbarkeitseinstellung des Sprites („visible“) betroffen. Sie können jedoch im Drehbuch auf die Schaltfläche „Ton aus“ für diesen Kanal klicken, um die Eigenschaft `visible` property to `FALSE`auf FALSE zu setzen und zu verhindern, dass Ereignisse an diesen Kanal gesendet werden. Die „Ton aus“-Einstellung deaktiviert den Kanal, während die Einstellung der Sprite-Eigenschaft `visible` auf `FALSE` sich nur auf die grafische Eigenschaft auswirkt.*

Diese Eigenschaft kann getestet und eingestellt werden. Wenn diese Eigenschaft auf `FALSE` gesetzt ist, wird sie nach Ende des Sprites nicht automatisch auf `TRUE` zurückgesetzt. Sie müssen die Sprite-Eigenschaft `visible` explizit auf `TRUE` setzen, damit andere Darsteller, die diesen Kanal benutzen, sichtbar werden.

Beispiel

Die folgende Anweisung macht Sprite 8 sichtbar:

```
sprite(8).visible = TRUE
```

visibility

Syntax

```
member(whichCastmember).model(whichModel).visibility
modelObjectReference.visibility
```

Beschreibung

Mit dieser 3D-Eigenschaft können Sie die Eigenschaft `visibility` des referenzierten Modells ermitteln und festlegen. Diese Eigenschaft bestimmt, wie die Geometrie des Modells gezeichnet wird. Mögliche Werte:

- `#none`: Es werden keine Polygone gezeichnet, das Modell ist unsichtbar.
- `#front`: Nur die Polygone, die zur Kamera hin zeigen, werden gezeichnet. Diese Methode wird „Back Face Culling“ genannt und optimiert die Rendergeschwindigkeit. Dies ist die Standardeinstellung für diese Eigenschaft.
- `#back`: Nur die Polygone, die von der Kamera weg zeigen, werden gezeichnet. Verwenden Sie diese Einstellung, wenn Sie die Innenseite eines Modells zeichnen möchten, oder für Modelle, die nicht richtig gezeichnet werden, möglicherweise weil sie aus einem Dateiformat importiert wurden, das bei der Berechnung von Normalen einen anderen „Handiness“-Wert verwendete.
- `#both`: Bei allen Polygonen werden beide Seiten gezeichnet. Verwenden Sie diese Einstellung, wenn Sie die Ebene unabhängig von der Blickrichtung sehen möchten, und für Modelle, die nicht richtig gezeichnet werden.

Beispiel

Die folgende Anweisung zeigt, dass die Eigenschaft `visibility` des Modells `Monster02` auf `#none` gesetzt ist. Das Modell ist unsichtbar.

```
-- Lingo syntax
put member("3D").model("Monster02").visibility
-- #none

// JavaScript syntax
trace(member("3D").getPropRef("model",1).visibility)
// symbol("none")
```

volume (DVD)

Syntax

```
-- Lingo syntax
dvdObjRef.volume

// JavaScript syntax
dvdObjRef.volume;
```

Beschreibung

Diese DVD-Eigenschaft bestimmt die aktuelle DVD-Lautstärke. Lesen/Schreiben.

Die Lautstärke muss eine Ganzzahl zwischen 0 (Ton aus) und 100 (maximale Lautstärke) sein.

Unter Windows ist die Lautstärkeskala logarithmisch. Auf dem Mac ist sie linear.

Beispiel

Die folgende Anweisung stellt die Lautstärke des DVD-Darstellers ein.

```
-- Lingo syntax
member(1).volume = 20

// JavaScript syntax
member(1).volume = 20;
```

Siehe auch

DVD

volume (Darsteller)

Syntax

```
-- Lingo syntax
memberObjRef.volume

// JavaScript syntax
memberObjRef.volume;
```

Beschreibung

Diese Shockwave Audio- (SWA-) Darstellereigenschaft bestimmt die Lautstärke des angegebenen SWA-Streaming-Darstellers. Gültige Werte: 0 bis 255.

Diese Eigenschaft kann getestet und eingestellt werden.

Beispiel

Die folgende Anweisung stellt die Lautstärke eines SWA-Streaming-Darstellers auf die Hälfte der möglichen Lautstärke ein:

```
-- Lingo syntax
member("SWAfile").volume = 128

// JavaScript syntax
member("SWAfile").volume = 128;
```

volume (Mixer)

Syntax

```
mixer.volume
```

Beschreibung

Diese Audiomixereigenschaft bestimmt die Lautstärke eines Mixers. Der Wert 255 bedeutet volle Lautstärke und 0 bedeutet Stummschaltung.

Beispiele

```
--Lingo syntax
on mouseUp me
    mixerRef.volume = 120
end

// JavaScript syntax
function mouseUp(){
mixerRef.volume = 120;
}
```

Siehe auch

mute (Mixer), unmute (Mixer), Mixer

volume (MP4Media/FLV)

Syntax

```
sprite(1).volume
```

Beschreibung

MP4Media/FLV-Sprite-Eigenschaft, welche die Lautstärke eines MP4Media/FLV-Sprites festlegt.

Der Wert dieser Eigenschaft ist eine Ganzzahl zwischen 0 (Ton aus) und 255 (laut).

Hinweis: *Vor dem Gebrauch dieser Methode sollte die Eigenschaft „Audio" auf „True" gesetzt werden.*

Beispiele

Diese Beispiele stellen die Lautstärke für Sprite 7 auf 200 ein:

```
-- Lingo syntax
sprite(7).volume = 200

// JavaScript syntax
sprite(7).volume = 200;
```

volume (Soundkanal)

Syntax

```
-- Lingo syntax
soundChannelObjRef.volume

// JavaScript syntax
soundChannelObjRef.volume;
```

Beschreibung

Diese Soundkanaleigenschaft bestimmt die Lautstärke eines Soundkanals. Lesen/Schreiben.

Soundkanäle sind mit 1, 2, 3 usw. bis 8 nummeriert. Kanal 1 und 2 erscheinen im Drehbuch.

Der Wert der Eigenschaft `volume` liegt zwischen 0 (Ton aus) und 255 (maximale Lautstärke). Der Wert 255 ist die Höchstlautstärke für den Computer, die durch die Eigenschaft `soundLevel` des Sound-Objekts gesteuert wird. Niedrigere Werte werden im Verhältnis zu diesem Höchstwert skaliert. Mit dieser Eigenschaft können verschiedene Kanäle voneinander unabhängige Einstellungen im zulässigen Wertebereich haben.

Je niedriger der Wert der Soundeigenschaft `volume` ist, desto höher ist die Wahrscheinlichkeit, dass Sie Statik oder andere Geräusche hören. Die Verwendung von `soundLevel` kann u.U. den Geräuschpegel reduzieren, bietet aber weniger Steuerungsmöglichkeiten.

Ein Beispiel für `volume` in einem fertigen Film finden Sie im Film „Sound Control“ im Ordner „Learning/Lingo_Examples“ im Director-Anwendungsordner.

Beispiel

Die folgende Anweisung stellt die Lautstärke für Soundkanal 2 auf 130, was einer mittleren Einstellung entspricht:

```
-- Lingo syntax
sound(2).volume = 130

// JavaScript syntax
sound(2).volume = 130;
```

Siehe auch

`Sound Channel`, `soundLevel`

volume (Soundobjekt)

Syntax

```
soundObject.volume
```

Beschreibung

Diese Soundobjekteigenschaft gibt die Lautstärke (von 0 bis 255) des Soundobjekts aus oder legt diese fest. Der Wert 255 bedeutet volle Lautstärke und 0 bedeutet Stummschaltung.

Diese Eigenschaft kann ausgelesen und beschrieben werden.

Beispiele

```
soundObject.volume = 120
--Lingo syntax
on mouseUp me
    put soundObjRef.volume -- Displays the volume of the sound object
-- associated with soundobjectRef.
end

// JavaScript syntax
function mouseUp(){
put (soundObjRef.volume) ; // Displays the volume of the sound object
// associated with soundobjectRef.
}
```

Siehe auch

`mute (Soundobjekt)`, `unmute (Soundobjekt)`

volume (Sprite)

Syntax

```
-- Lingo syntax
spriteObjRef.volume

// JavaScript syntax
spriteObjRef.volume;
```

Beschreibung

Diese Sprite-Eigenschaft steuert die Lautstärke eines durch seinen Namen oder seine Nummer angegebenen Digitalvideo-Film- oder Windows Media-Darstellers. Die Werte können zwischen 0 und 256 liegen. Durch Werte kleiner oder gleich 0 wird der Sound abgeschaltet. Werte über 256 sind laut und klingen stark verzerrt.

Beispiel

Die folgende Anweisung stellt die Lautstärke des in Sprite-Kanal 7 ablaufenden QuickTime-Films auf 256, was der maximalen Lautstärke entspricht:

```
-- Lingo syntax
sprite(7).volume = 256

// JavaScript syntax
sprite(7).volume = 256;
```

Siehe auch

soundLevel

volume (Windows Media)

Syntax

```
-- Lingo syntax
windowsMediaObjRef.volume

// JavaScript syntax
windowsMediaObjRef.volume;
```

Beschreibung

Diese Windows Media-Sprite-Eigenschaft bestimmt die Lautstärke eines Windows Media-Sprites.

Der Wert dieser Eigenschaft ist eine Ganzzahl zwischen 0 (Ton aus) und 7 (laut).

Diese Eigenschaft kann auch in Director mithilfe des Menüs „Steuerung Lautstärke“ festgelegt werden.

Beispiel

Die folgende Anweisung legt die Lautstärke von Sprite 7 auf 2 fest:

```
-- Lingo syntax
sprite(7).volume = 2

// JavaScript syntax
sprite(7).volume = 2;
```

Siehe auch

Windows Media

warpMode

Syntax

```
-- Lingo syntax
spriteObjRef.warpMode

// JavaScript syntax
spriteObjRef.warpMode;
```

Beschreibung

Diese QuickTime VR-Sprite-Eigenschaft gibt den Typ der Verwerfung in einer Panoramaansicht an.

Mögliche Werte: `#full`, `#partial` und `#none`.

Diese Eigenschaft kann getestet und eingestellt werden. Wenn die Werte für den statischen Modus und den Bewegungsmodus beim Testen unterschiedlich sind, wird der Wert der Eigenschaft auf den für den aktuellen Modus eingestellten Wert gesetzt. Nach der Einstellung bestimmt diese Eigenschaft die Verwerfung für beide Modi, den statischen wie auch den Bewegungsmodus.

Beispiel

Die folgende Anweisung setzt den Verwerfungsmodus `warpMode` für Sprite 1 auf `#full`:

```
-- Lingo syntax
sprite(1).warpMode = #full

// JavaScript syntax
sprite(1).warpMode = symbol("full");
```

width

Syntax

```
-- Lingo syntax
memberObjRef.width
imageObjRef.width
spriteObjRef.width

// JavaScript syntax
memberObjRef.width;
imageObjRef.width;
spriteObjRef.width;
```

Beschreibung

Diese Darsteller-, Grafik- und Sprite-Eigenschaft bestimmt die Breite (in Pixel) von Vektorform-, Flash-, animierten GIF-, `RealMedia`-, Windows Media-, Bitmap- und Formdarstellern. Nur Lesen für Darsteller und Grafikobjekte. Lesen/Schreiben für Sprites.

Diese Eigenschaft hat keine Auswirkungen auf Feld- und Schaltflächendarsteller.

Beispiel

Die folgende Anweisung weist die Breite von Darsteller 50 der Variablen `theHeight` zu:

```
-- Lingo syntax
theHeight = member(50).width

// JavaScript syntax
var theHeight = member(50).width;
```

Die folgende Anweisung stellt die Breite von Sprite 10 auf 26 Pixel ein:

```
-- Lingo syntax
sprite(10).width = 26

// JavaScript syntax
sprite(10).width = 26;
```

Die folgende Anweisung ordnet der Variablen `howWide` die Breite von Sprite-Nummer i + 1 zu:

```
-- Lingo syntax
howWide = sprite(i + 1).width

// JavaScript syntax
var howWide = sprite(i + 1).width;
```

Siehe auch

`height`, `image (Grafik)`, `Member`, `Sprite`

width (3D)

Syntax

```
member(whichCastmember).modelResource(whichModelResource).width
modelResourceObjectReference.width
```

Beschreibung

Mit dieser 3D-Eigenschaft können Sie die Breite der Ebene für eine Modellressource vom Typ `#box` oder `#plane` ermitteln und festlegen. Diese Eigenschaft muss größer als 0.0 sein und hat den Standardwert 1.0. Für Objekte vom Typ `#box` beträgt der Standardwert für `width` 50.0. Für Objekte vom Typ `#plane` ist der Standardwert 1.0. Der Wert `width` wird entlang der X-Achse gemessen.

Beispiel

Die folgende Anweisung setzt die Breite der Modellressource "Grasebene“ auf 250.0:

```
-- Lingo syntax
member("3D World").modelResource("Grass plane").width = 250.0

// JavaScript syntax
Member("3D World").getPropRef("modelResource",1).width = 250.0;
```

width (MP4Media/FLV)

Syntax

```
put member(1).width
Sprite(1).width = 50
```

Beschreibung

Darsteller- und Sprite-Eigenschaft, die für MP4Media/FLV-Darsteller deren Breite in Pixel des Originalvideos des Autors entspricht, das auf der Bühne angezeigt wird. Diese Eigenschaft kann für Darsteller nur ausgelesen werden und nur für Sprites beschrieben werden.

Beispiele

In folgendem Beispiel wird die Höhe des MP4Media/FLV-Darstellers der Variable `theHeight` zugewiesen:

```
-- Lingo syntax
theHeight = member(50).width

// JavaScript syntax
var theHeight = member(50).width;
```

Dabei wird die Breite von Sprite 10 auf 26 Pixel eingestellt:

```
-- Lingo syntax
sprite(10).width = 26

// JavaScript syntax
sprite(10).width = 26;
```

widthVertices

Syntax

```
member(whichCastmember).modelResource(whichModelResource).
widthVertices
modelResourceObjectReference.widthVertices
```

Beschreibung

Mit dieser 3D-Eigenschaft können Sie die Anzahl von Scheitelpunkten (Ganzzahl) auf der X-Achse einer Modellressource vom Typ `#box` oder `#plane` ermitteln und festlegen. Diese Eigenschaft muss größer oder gleich dem Standardwert 2 sein.

Beispiel

Die folgende Anweisung setzt die Eigenschaft `widthVertices` der Modellressource "Turm" auf 10. Zur Definition der Geometrie der Modellressource entlang der X-Achse werden 18 Polygone (2 * (10-1) Dreiecke) verwendet.

```
member("3D World").modelResource("Tower").widthVertices = 10
```

wind

Syntax

```
member(whichCastmember).modelResource(whichModelResource).wind
modelResourceObjectReference.wind
```

Beschreibung

Mit dieser 3D-Eigenschaft können Sie die Eigenschaft `wind` einer Modellressource vom Typ `#particle` als Vektor ermitteln und festlegen.

Die Eigenschaft `wind` definiert die Windrichtung und -stärke, die in jedem Simulationsschritt auf alle Partikel ausgeübt wird. Der Standardwert dieser Eigenschaft ist vector(0, 0, 0), d. h. es wird kein Wind verwendet.

Beispiel

```
-- Lingo syntax
put member("3D").modelResource("fog bank").wind
-- vector(10.5,0,0)

// JavaScript syntax
trace(member("3D").getPropRef("modelResource","1").wind)
// vector(10.5,0,0)
```

window

Syntax

```
-- Lingo syntax
_player.window[windowNameOrNum]

// JavaScript syntax
_player.window[windowNameOrNum];
```

Beschreibung

Diese Player- und Filmeigenschaft bietet indizierten oder benannten Zugriff auf die vom Director-Player erstellten Window-Objekte. Nur Lesen.

Das Argument *windowNameOrNum* ist entweder eine Zeichenfolge, die den Namen des Fensters angibt, auf das zugegriffen werden soll, oder eine Ganzzahl, die die Indexposition des Fensters angibt, auf das zugegriffen werden soll.

Die Funktionalität dieser Eigenschaft ist mit der der Top-Level-Methode `window()` identisch.

Beispiel

Die folgende Anweisung legt die Variable `myWindow` auf das dritte Window-Objekt fest:

```
-- Lingo syntax
myWindow = _player.window[3]

// JavaScript syntax
var myWindow = _player.window[3];
```

Siehe auch

Player, window()

windowBehind

Syntax

```
-- Lingo syntax
windowObjRef.windowBehind

// JavaScript syntax
windowObjRef.windowBehind;
```

Beschreibung

Diese Fenstereigenschaft gibt einen Verweis auf das Fenster zurück, das hinter allen anderen Fenstern liegt. Nur Lesen.

Beispiel

Die folgenden Anweisungen legen die Variable `backWindow` auf das Fenster fest, das hinter allen anderen Fenstern liegt, und verschieben dann das Fenster in den Vordergrund:

```
-- Lingo syntax
backWindow = _player.windowList[5].windowBehind
backWindow.moveToFront()

// JavaScript syntax
var backWindow = _player.windowList[5].windowBehind;
backWindow.moveToFront();
```

Siehe auch

moveToBack(), moveToFront(), Window, windowInFront, windowList

windowInFront

Syntax

```
-- Lingo syntax
windowObjRef.windowInFront

// JavaScript syntax
windowObjRef.windowInFront;
```

Beschreibung

Diese Fenstereigenschaft gibt einen Verweis auf das Fenster zurück, das vor allen anderen Fenstern liegt. Schreibgeschützt.

Beispiel

Die folgenden Anweisungen legen die Variable `frontWindow` auf das Fenster fest, das vor allen anderen Fenstern liegt, und verschieben dann das Fenster in den Hintergrund:

```
-- Lingo syntax
frontWindow = _player.windowList[5].windowInFront
frontWindow.moveToBack()

// JavaScript syntax
var frontWindow = _player.windowList[5].windowInFront
frontWindow.moveToBack();
```

Siehe auch

moveToBack(), moveToFront(), Window, windowBehind, windowList

windowList

Syntax

```
-- Lingo syntax
_player.windowList

// JavaScript syntax
_player.windowList;
```

Beschreibung

Diese Player-Eigenschaft zeigt eine Liste mit Verweisen auf alle bekannten Filmfenster an. Nur Lesen.

Die Bühne wird ebenfalls als Fenster angesehen.

Beispiel

Die folgende Anweisung zeigt eine Liste aller bekannten Filmfenster im Nachrichtenfenster an:

```
-- Lingo syntax
trace(_player.windowList)

// JavaScript syntax
trace(_player.windowList);
```

Siehe auch

Player

wordWrap

Syntax

```
-- Lingo syntax
memberObjRef.wordWrap

// JavaScript syntax
memberObjRef.wordWrap;
```

Beschreibung

Diese Darstellereigenschaft bestimmt, ob ein Zeilenumbruch zulässig ist (TRUE) oder nicht (FALSE).

Beispiel

Die folgende Anweisung schaltet den Zeilenumbruch für den Felddarsteller „Rokujo“ aus:

```
--Lingo syntax
member("Rokujo").wordWrap = FALSE

// JavaScript syntax
member("Rokujo").wordWrap = false;
```

worldPosition

Syntax

```
member(whichCastmember).model(whichModel).worldPosition
member(whichCastmember).light(whichLight).worldPosition
member(whichCastmember).camera(whichCamera).worldPosition
member(whichCastmember).group(whichGroup).worldPosition
```

Beschreibung

Mit dieser 3D-Eigenschaft können Sie die Position des angegebenen Node in Weltkoordinaten ermitteln, aber nicht festlegen. Ein Node kann ein Modell, eine Gruppe, eine Kamera oder ein Licht sein. Diese Eigenschaft hat die gleiche Wirkung wie der Befehl `getWorldTransform().position`. Die Position eines Node wird anhand eines Vektorobjekts dargestellt.

Beispiel

Die folgende Anweisung zeigt, dass die Position des Modells „Mars“ in Weltkoordinaten „vector (-1333.2097, 0.0000, -211.0973)“ ist.

```
--Lingo syntax
put member("scene").model("Mars").worldPosition
-- vector(-1333.2097, 0.0000, -211.0973)

// JavaScript syntax
trace(member("scene").getProp("model",1).worldPosition)
// vector(-1333.2097, 0.0000, -211.0973)
```

Siehe auch

getWorldTransform(), position (Transformation)

worldTransform

Syntax

```
member(whichMember).model(whichModel).bonesPlayer.bone[index].worldTransform
```

Beschreibung

Mit dieser 3D-Eigenschaft für den Modifizierer „bonesPlayer“ können Sie die weltbezogene Transformation eines bestimmten Knochens ermitteln. Die Eigenschaft `transform` gibt im Gegensatz dazu die parentbezogene Transformation des Knochens zurück. Die Eigenschaft `worldTransform` kann nur bei Modellen mit dem Modifizierer „bonesPlayer“ verwendet werden.

Beispiel

Die folgende Anweisung speichert die weltbezogene Transformation eines Knochens in der Variablen `finalTransform`:

```
--Lingo syntax
finalTransform = member("3D").model("biped").bonesPlayer.bone[3].worldTransform

// JavaScript syntax
finalTransform =
member("3D").getPropRef("model",1).getProp("bonesPlayer").bone[3].worldTransform;
```

Siehe auch

`bone`, `getWorldTransform()`, `transform (Eigenschaft)`

wrapTransform

Syntax

```
member( whichCastmember ).shader( ShaderName ).wrapTransform
member( whichCastmember ).shader[ ShaderIndex ].wrapTransform
member( whichCastmember ).model[modelName].shader.wrapTransform
member( whichCastmember ).model.shaderlist[ shaderListIndex ].wrapTransform
```

Beschreibung

Über diese 3D-Eigenschaft für Standard-Shader können Sie auf eine Transformation zugreifen, die die Texturkoordinatenzuordnung der `texture` des Shaders modifiziert. Durch Drehen dieser Transformation können Sie die Projektion der Textur auf eine Modelloberfläche ändern. Die Textur selbst bleibt unverändert; die Transformation ändert lediglich die Ausrichtung, in der der Shader die Textur anwendet.

Hinweis: *Dieser Befehl hat nur dann eine Wirkung, wenn die* `textureModeList` *auf* `#planar`, `#spherical` *oder* `#cylindrical` *gesetzt ist.*

Beispiel

Die folgenden Anweisungen setzen die Eigenschaft `transformMode` des Shaders "Shad2" auf `#wrapCylindrical` und drehen dann diese zylindrische Projektion um 90° um die x-Achse, sodass die zylindrische Zuordnung um die *y*-Achse statt um die *z*-Achse gewickelt wird.

```
--Lingo syntax
s = member("Scene").shader("shad2")
s.textureMode= #wrapCylindrical
s.wrapTransform.rotate(90.0, 0.0, 0.0)

// JavaScript syntax
var s = member("Scene").getPropRef("shader",1);
s.textureMode = symbol("wrapCylindrical");
s.wrapTransform.rotate(90.0,0.0,0.0);
```

wrapTransformList

Syntax

```
member( whichCastmember ).shader( ShaderName ).wrapTransformList[ textureLayerIndex ]
member( whichCastmember ).shader[ shaderListIndex ].wrapTransformList[ textureLayerIndex ]
member( whichCastmember ).model( modelName ).shader.wrapTransformList[ textureLayerIndex ]
member( whichCastmember ).model( modelName ).shaderList[ shaderListIndex ]. wrapTransformList[
textureLayerIndex ]
```

Beschreibung

Über diese 3D-Eigenschaft für Standard-Shader können Sie auf eine Transformation zugreifen, die die Texturkoordinatenzuordnung einer Texturebene modifiziert. Durch Drehen dieser Transformation können Sie die Projektion der Textur auf eine Modelloberfläche ändern. Die Textur selbst bleibt unverändert; die Transformation ändert lediglich die Ausrichtung, in der der Shader die Textur anwendet.

Hinweis: *`wrapTransformList[textureLayerIndex]` hat nur dann eine Wirkung, wenn `textureModeList[textureLayerIndex]` auf `#planar`, `#spherical` oder `#cylindrical` gesetzt ist.*

Beispiel

Im folgenden Beispiel legt Zeile 2 die Eigenschaft `transformMode` der dritten Texturebene des Shaders "Shad2" auf `#wrapCylindrical` fest. In Zeile 3 wird diese zylindrische Projektion um 90° um die x-Achse gedreht, sodass die zylindrische Zuordnung um die *y*-Achse statt um die *z*-Achse gewickelt wird.

```
--Lingo syntax
s = member("Scene").shader("shad2")
s.textureModeList[3] = #wrapCylindrical
s.wrapTransformList[3].rotate(90.0, 0.0, 0.0)

// JavaScript syntax
var s = member("Scene").getPropRef("shader",1);
s.textureModeList[3] = symbol("wrapCylindrical");
s.wrapTransformList[3].rotate(90.0, 0.0, 0.0);
```

Siehe auch

`newShader`, `textureModeList`

x (Vektor)

Syntax

```
member(whichCastmember).vector.x
member(whichCastmember).vector[1]
```

Beschreibung

Mit dieser 3D-Eigenschaft können Sie die *x*-Komponente eines Vektors ermitteln und festlegen.

Beispiel

Die folgende Anweisung zeigt die *x*-Komponente eines Vektors an:

```
vec = vector(20, 30, 40)
put vec.x
-- 20.0000
```

xAxis

x5796 | Lingo_3d_x_axis_help

Syntax

```
member(whichCastmember).transform.xAxis
```

Beschreibung

Mit dieser 3D-Transformationseigenschaft können Sie den Vektor, der die kanonische *x*-Achse der Transformation im Transformationsraum darstellt, ermitteln, aber nicht festlegen.

Beispiel

In der ersten Zeile dieses Beispiels wird die Transformation des Modells `ModCylinder` auf die Einheitsmatrix gesetzt. Die nächsten beiden Zeilen zeigen, dass die *x*-Achse von `ModCylinder` der Vektor (1.0000, 0.0000, 0.0000) ist, d. h. dass die *x*-Achse von `ModCylinder`an der *x*-Achse der Welt ausgerichtet ist. In der nächsten Zeile wird `ModCylinder` um 90° um die Y-Achse gedreht. Dadurch werden auch die Achsen von `ModCylinder` gedreht. Die letzten zwei Zeilen zeigen, dass die *x*-Achse von `ModCylinder` jetzt der Vektor (0.0000, 0.0000, -1.0000) ist, d. h. dass die *x*-Achse von `ModCylinder`nun an der negativen *z*-Achse der Welt ausgerichtet ist.

```
-- Lingo syntax
member("Engine").model("ModCylinder").transform.identity()
put member("Engine").model("ModCylinder").transform.xAxis
-- vector( 1.0000, 0.0000, 0.0000 )
member("Engine").model("ModCylinder").rotate(0, 90, 0)
put member("Engine").model("ModCylinder").transform.xAxis
-- vector( 0.0000, 0.0000, -1.0000 )

// JavaScript syntax
member("Engine").getPropRef("model",1).transform.identity();
trace(member("Engine").getPropRef("model",1).transform.xAxis)
// vector( 1.0000, 0.0000, 0.0000 )
member("Engine").getPropRef("model",1).rotate(0, 90, 0);
trace(member("Engine").getPropRef("model",1).transform.xAxis)
// vector( 0.0000, 0.0000, -1.0000 )
```

xtra

Syntax

```
-- Lingo syntax
_player.xtra[xtraNameOrNum]

// JavaScript syntax
_player.xtra[xtraNameOrNum];
```

Beschreibung

Diese Player-Eigenschaft bietet indizierten oder benannten Zugriff auf die im Director-Player verfügbaren Xtra-Erweiterungen. Nur Lesen.

Das Argument *xtraNameOrNum* ist entweder eine Zeichenfolge, die den Namen der Xtra-Erweiterung angibt, auf die zugegriffen werden soll, oder eine Ganzzahl, die die Indexposition der Xtra-Erweiterung angibt, auf die zugegriffen werden soll.

Die Funktionalität dieser Eigenschaft ist mit der der Top-Level-Methode `xtra()` identisch.

Beispiel

Die folgende Anweisung legt die Variable `myXtra` auf die Xtra-Spracherweiterung fest:

```
-- Lingo syntax
myXtra = _player.xtra["SpeechXtra"]

// JavaScript syntax
var myXtra = _player.xtra["SpeechXtra"];
```

Siehe auch

`Player`, `xtra()`

xtraList (Film)

Syntax

```
-- Lingo syntax
_movie.xtraList

// JavaScript syntax
_movie.xtraList;
```

Beschreibung

Diese Filmeigenschaft zeigt eine lineare Eigenschaftsliste aller Xtra-Erweiterungen im Dialogfeld „Filme/Xtras“ an, die zum Film hinzugefügt wurden. Nur Lesen.

`xtraList` kann zwei Eigenschaften enthalten

- `#filename` – Gibt den Dateinamen der Xtra-Erweiterung auf der aktuellen Plattform an. Eine Liste ohne einen `#filename`-Eintrag ist möglich, z. B. wenn die Xtra-Erweiterung nur auf einer Plattform vorhanden ist.
- `#packageurl` – Gibt den Speicherort des durch`#packagefiles` angegebenen Downloadpakets als URL an.
- `#packagefiles` – Wird nur eingestellt, wenn die Xtra-Erweiterung zum Herunterladen markiert ist. Der Wert dieser Eigenschaft ist eine weitere Liste, die eine Eigenschaftsliste für jede Datei im Download-Paket für die aktuelle Plattform enthält. Die Eigenschaften in dieser untergeordneten Eigenschaftsliste sind `#name` und `#version`, und die darin enthaltenen Informationen sind mit denen in `xtraList (Player)` identisch.

Beispiel

Die folgende Anweisung zeigt die `xtraList` im Nachrichtenfenster an:

```
-- Lingo syntax
put(_movie.xtraList)

// JavaScript syntax
put(_movie.xtraList);
```

Siehe auch

Movie, xtraList (Player)

xtraList (Player)

Syntax

```
-- Lingo syntax
_player.xtraList

// JavaScript syntax
_player.xtraList;
```

Beschreibung

Diese Player-Eigenschaft zeigt eine lineare Eigenschaftsliste aller verfügbaren Xtra-Erweiterungen und deren Dateiversionen an. Nur Lesen.

Diese Eigenschaft ist nützlich, wenn die Funktionalität eines Films von einer bestimmten Version einer Xtra-Erweiterung abhängt.

Es gibt zwei mögliche Eigenschaften, die in einer `xtraList` vorkommen können:

- `#filename` – Gibt den Dateinamen der Xtra-Erweiterung auf der aktuellen Plattform an. Eine Liste ohne einen `#filename`-Eintrag ist möglich, z. B. wenn die Xtra-Erweiterung nur auf einer Plattform vorhanden ist.
- `#version` – Gibt an, dass dies die gleiche Versionsnummer ist, die auch im Dialogfeld "Eigenschaften" (Windows) oder im Dialogfenster „Information" (Macintosh) erscheint, wenn die Datei auf dem Desktop ausgewählt wird. Eine Xtra-Erweiterung muss nicht unbedingt über eine Versionsnummer verfügen.

Beispiel

Die folgende Anweisung zeigt alle für den Director-Player verfügbaren Xtra-Erweiterungen im Nachrichtenfenster an.

```
-- Lingo syntax
trace(_player.xtraList)

// JavaScript syntax
trace(_player.xtraList);
```

Siehe auch

mediaXtraList, Player, scriptingXtraList, toolXtraList, transitionXtraList

y (Vektor)

Syntax

```
member(whichCastmember).vector.y
member(whichCastmember).vector[2]
```

Beschreibung

Mit dieser 3D-Eigenschaft können Sie die *y*-Komponente eines Vektors ermitteln und festlegen.

Beispiel

Die folgende Anweisung zeigt die y-Komponente eines Vektors an:

```
vec = vector(20, 30, 40)
put vec.y
-- 30.0000
```

yAxis

Syntax

```
member(whichCastmember).transform.yAxis
```

Beschreibung

Mit dieser 3D-Transformationseigenschaft können Sie den Vektor, der die kanonische *y*-Achse der Transformation im Transformationsraum darstellt, ermitteln, aber nicht festlegen.

Beispiel

In der ersten Zeile dieses Beispiels wird die Transformation des Modells `ModCylinder` auf die Einheitsmatrix gesetzt. Die nächsten zwei Zeilen zeigen, dass die Y-Achse von `ModCylinder` der Vektor (0.0000, 1.0000, 0.0000) ist, d. h. dass die *y*-Achse von `ModCylinder`an der *y*-Achse der Welt ausgerichtet ist. In der nächsten Zeile wird `ModCylinder` um 90° um die *x*-Achse gedreht. Dadurch werden auch die Achsen von `ModCylinder` gedreht. Die letzten zwei Zeilen zeigen, dass die *y*-Achse von `ModCylinder` jetzt der Vektor (0.0000, 0.0000, 1.0000) ist, d. h. dass die *y*-Achse von `ModCylinder` nun an der positiven *z*-Achse der Welt ausgerichtet ist.

```
member("Engine").model("ModCylinder").transform.identity()
put member("Engine").model("ModCylinder").transform.yAxis
-- vector( 0.0000, 1.0000, 0.0000 )
member("Engine").model("ModCylinder").rotate(90, 0, 0)
put member("Engine").model("ModCylinder").transform.yAxis
-- vector( 0.0000, 0.0000, 1.0000 )
```

yon

Syntax

```
member(whichCastmember).camera(whichCamera).yon
```

Beschreibung

Mit dieser 3D-Eigenschaft können Sie die Entfernung von der Kamera ermitteln oder festlegen, bei der der Sichtkegel entlang der Z-Achse der Kamera abgeschnitten wird. Objekte, die sich in einer größeren Entfernung als `yon` befinden, werden nicht gezeichnet.

Der Standardwert dieser Eigenschaft lautet 3.40282346638529e38.

Beispiel

Die folgende Anweisung setzt die Eigenschaft yon von Kamera 1 auf 50000:

```
-- Lingo syntax
member("3d world").camera[1].yon = 50000

// JavaScript syntax
member("3d world").getPropRef("camera",1).yon = 50000
```

Siehe auch

`hither`

z (Vektor)

Syntax

```
member(whichCastmember).vector.z
member(whichCastmember).vector[3]
```

Beschreibung

Mit dieser 3D-Eigenschaft können Sie die *z*-Komponente eines Vektors ermitteln und festlegen.

Beispiel

Die folgende Anweisung zeigt die z-Komponente eines Vektors an:

```
vec = vector(20, 30, 40)
put vec.z
-- 40.0000
```

zAxis

Syntax

```
member(whichCastmember).transform.zAxis
```

Beschreibung

Mit dieser 3D-Transformationseigenschaft können Sie den Vektor, der die kanonische *z*-Achse der Transformation im Transformationsraum darstellt, ermitteln, aber nicht festlegen.

Beispiel

In der ersten Zeile dieses Beispiels wird die Transformation des Modells `ModCylinder` auf die Einheitsmatrix gesetzt. Die nächsten zwei Zeilen zeigen, dass die *z*-Achse von `ModCylinder` der Vektor (0.0000, 0.0000, 1.0000) ist, d. h. dass die *z*-Achse von `ModCylinder` an der *z*-Achse der Welt ausgerichtet ist. In der nächsten Zeile wird `ModCylinder` um 90° um die *y*-Achse gedreht. Dadurch werden auch die Achsen von `ModCylinder` gedreht. Die letzten zwei Zeilen zeigen, dass die *z*-Achse von `ModCylinder` jetzt der Vektor (1.0000, 0.0000, 0.0000) ist, d. h. dass die *z*-Achse von `ModCylinder` an der *x*-Achse der Welt ausgerichtet ist.

```
-- Lingo syntax
member("Engine").model("ModCylinder").transform.identity()
put member("Engine").model("ModCylinder").transform.zAxis
-- vector( 1.0000, 0.0000, 0.0000 )
member("Engine").model("ModCylinder").rotate(0, 90, 0)
put member("Engine").model("ModCylinder").transform.zAxis
-- vector( 0.0000, 0.0000, -1.0000 )

// JavaScript syntax
member("Engine").getPropRef("model",1).transform.identity();
trace(member("Engine").getPropRef("model",1).transform.zAxis)
//vector( 1.0000, 0.0000, 0.0000 )
member("Engine").getPropRef("model",1).rotate(0, 90, 0);
trace(member("Engine").getPropRef("model",1).transform.zAxis)
// vector( 0.0000, 0.0000, -1.0000 )
```

Kapitel 15: Physics-Engine

Das Physics-Xtra ist ein Hochleistungswerkzeug, das Entwickler beim Erstellen von 3D-Welten unterstützt, in denen Objekte interagieren. Das Xtra führt Berechnungen durch, um die Ergebnisse von Kollisionen zu ermitteln, und berücksichtigt dabei Objekteigenschaften wie Masse, Geschwindigkeit und Drehung. Sie haben die Möglichkeit, Kräfte anzulegen und Objekte durch Beschränkungen miteinander zu verknüpfen. Als Beschränkungen stehen Verbindungen mit sechs Freiheitsgraden, lineare Verbindungen, Winkelverbindungen und Federverbindungen zur Verfügung.

Zusätzlich werden Geländearten und Raycasting unterstützt. Ein Gelände ist einer hügeligen Ebene vergleichbar, die in zwei Dimensionen unbegrenzt ist und entlang einer dritten Dimension eine Erhebung definiert. Raycasting bezeichnet den Mechanismus der Kollisionserkennung mit Strahlen. Raycasting kann mit allen Arten von Festkörpern und Geländen durchgeführt werden.

Mit diesem Xtra können sich Entwickler ganz auf den Spielablauf und die Benutzerinteraktion konzentrieren und müssen sich nicht mit der Erstellung einer Echtzeitphysik-Engine mithilfe von Lingo-Skripts aufhalten.

Das Physics-Xtra (dynamiks) ist eine vollständig integrierte Festkörpersimulations-Engine für Adobe® Director®. Das Dynamik-Xtra wird unter Windows- und Macintosh-Plattformen unterstützt.

Die Physik-Engine für Director bietet die nachfolgend aufgeführten Leistungsmerkmale. Diese werden im weiteren Verlauf des Kapitels detailliert beschrieben.

Eigenschaften der Physikwelt

angularDamping

Syntax

```
world.angularDamping
```

Zugriff: Abrufen/Festlegen

Typ: Fließkommazahl

Standardwert: 0.000

Beschreibung

Diese Eigenschaft steht für den Dämpfungsgrad, um den die Winkelkräfte reduziert werden.

Beispiel

```
--Lingo Syntax
member("PhysicsWorld").angularDamping = 10.0
//Javascript syntax
member("PhysicsWorld").angularDamping = 10.0;
```

Siehe auch

linearDamping

contactTolerance

Syntax

```
world.contactTolerance
```

Zugriff: Abrufen/Festlegen

Typ: Fließkommazahl

Beschreibung

Die Durchdringungstiefe zwischen Festkörpern für die zu erkennende Kollision.

Hinweis: *Um beim Simulieren einer Kollision zwischen Körpern die erwarteten Ergebnisse zu erzielen, sollten Sie den Wert für die Kontakttoleranz anpassen.*

Beispiel

Die folgende Anweisung legt die Durchdringungstiefe zwischen Festkörpern für die zu erkennende Kollision fest.

```
--Lingo Syntax
member("PhysicsWorld").contactTolerance = 0.01
//JavaScript Syntax
member("PhysicsWorld").contactTolerance = 0.01;
```

Hinweis:

- Im Idealfall beträgt die Kontakttoleranz 2% der Festkörpermaße.
- Wird die Kontakttoleranz nicht für jeden Festkörper festgelegt, wird der für die Welt angegebene Wert verwendet.
- Höhere Kontakttoleranzwerte führen zu numerischen Fehlern und Instabilität.

friction

Syntax

```
world.friction
```

Zugriff: Abrufen/Festlegen

Typ: Fließkommazahl

Wert 0 - 1

Beschreibung

Diese Eigenschaft stellt den Standardwert für den Reibungskoeffizienten für alle Festkörper in der Welt dar. Die „friction“-Eigenschaft nimmt Werte von 0-1 ein.

Beispiel

```
--Lingo Syntax
member("PhysicsWorld").friction = 0.5
//Javascript syntax
member("PhysicsWorld").friction = 0.5;
```

Siehe auch

restitution

gravity

Syntax

```
world.gravity
```

Zugriff: Abrufen/Festlegen

Typ: vector

Beschreibung

Diese Eigenschaft steht für die Erdbeschleunigung, die bei allen Objekten in der Physikwelt auftritt.

Beispiel

```
--Lingo Syntax
member("PhysicsWorld").gravity = vector(0,-9.81,0)
//Javascript syntax
member("PhysicsWorld").gravity = vector(0,-9.81,0);
```

IsInitialized

Syntax

```
world.isInitialized
```

Zugriff: Abrufen

Typ: Boolesch

Beschreibung

Der aktuelle Initialisierungsstatus der Welt. Wenn die init()-Methode für die Welt erfolgreich ausgeführt wurde, ist diese Eigenschaft auf 1 eingestellt.

Beispiel

```
--Lingo Syntax
If not member("PhysicsWorld").isInitalized then
-- Initialize the physics world here
end if
//JavaScript Syntax
If (member("PhysicsWorld").isInitialized == 0) {
    // Initialize World here
}
```

Siehe auch

init()

linearDamping

Syntax

```
world.linearDamping
```

Zugriff: Abrufen/Festlegen

Typ: Fließkommazahl

Standardwert: 0.000

Beschreibung
Diese Eigenschaft steht für den Verlust an linearer Geschwindigkeit am Ende eines Zeitschritts.

Beispiel
```
--Lingo Syntax
member("PhysicsWorld").linearDamping = 10.0
//Javascript syntax
member("PhysicsWorld").linearDamping = 10.0;
```

Siehe auch
angularDamping

restitution

Syntax
```
world.restitution
```

Zugriff: Abrufen/Festlegen

Typ: Fließkommazahl

Wert 0 - 1

Beschreibung
Diese Eigenschaft stellt den Standardwert für die Stoßzahl für alle Festkörper in der Welt dar. Die „restitution"-Eigenschaft nimmt Werte von 0-1 ein.

Beispiel
```
--Lingo Syntax
member("PhysicsWorld").restitution = 0.5
//Javascript syntax
member("PhysicsWorld").restitution = 0.5;
```

Siehe auch
friction

scalingFactor

Syntax
```
world.scalingFactor
```

Zugriff: Abrufen

Typ: vector

Beschreibung
Diese Eigenschaft steht für den Maßstabsfaktor für die Umwandlung, wenn die 3D-Welt und die Physikwelt unterschiedliche Maßeinheiten aufweisen. „scalingFactor" ist ein Vektor, der den Maßstabsfaktor für Länge, Masse und Zeit akzeptiert.

Hinweis: *Der Wert für diese Eigenschaft kann in Director 11 nicht eingestellt werden, da diese Funktion erst in künftigen Versionen zur Verfügung steht.*

Beispiel

```
--Lingo Syntax
put member("PhysicsWorld").scalingFactor
//Javascript syntax
put(member("PhysicsWorld').scalingFactor)
```

Siehe auch

`init()`

sleepMode

Syntax

```
world.sleepMode
```

Zugriff: Abrufen/Festlegen

Werte: #energy, #linearvelocity

Standardwert: #energy

Beschreibung

Diese Eigenschaft ruft den Ruhezustand für Festkörper ab bzw. legt diesen fest.

#energy Die kinetische Energie des Festkörpers dient dazu, seinen Ruhezustand zu ermitteln.

#linearvelocity Die lineare Geschwindigkeit des Festkörpers dient dazu, seinen Ruhezustand zu ermitteln.

Beispiel

```
--Lingo Syntax
member("PhysicsWorld").sleepMode = #energy
//Javascript syntax
member("PhysicsWorld").sleepMode = symbol("energy");
```

Siehe auch

`sleepThreshold`

sleepThreshold

Syntax

```
world.sleepThreshold
```

Zugriff: Abrufen/Festlegen

Typ: Fließkommazahl

Standardwert: 0.000

Beschreibung

Diese Eigenschaft bestimmt den Schwellenwert, unterhalb dessen Festkörperobjekte inaktiv werden. Dieser Wert ist entweder die kinetische Energie oder die linear Geschwindigkeit des Festkörpers, je nachdem, welcher Wert für die Eigenschaft „sleepMode“ festgelegt wurde. Diese Eigenschaft kann nur auf dynamische Objekte angewandt werden.

Beispiel

```
--Lingo Syntax
member("PhysicsWorld").sleepThreshold = 10.0
//Javascript syntax
member("PhysicsWorld").sleepThreshold = 10.0;
```

Siehe auch

sleepMode

subSteps

Syntax

```
world.subSteps
```

Zugriff: Abrufen/Festlegen

Typ: Ganzzahl

Beschreibung

Gibt die Anzahl der Unterschritte für jeden simulate-Aufruf an.

Beispiel

```
--Lingo Syntax
member("PhysicsWorld").subSteps = 5
//Javascript syntax
member("PhysicsWorld").subSteps = 5;
```

Siehe auch

init(),timeStep,timeStepMode

timeStep

Syntax

```
world.timeStep
```

Zugriff: Abrufen/Festlegen

Typ: Fließkommazahl

Beschreibung

Diese Eigenschaft wird nur im Zeitschrittmodus „equal“ verwendet und legt die Schrittdauer für die Physikwelt fest.

Beispiel

```
--Lingo Syntax
member("PhysicsWorld"). timeStep = 0.0167
//Javascript syntax
member("PhysicsWorld"). timeStep = 0.0167;
```

Siehe auch

init(),subSteps,timeStepMode

timeStepMode

Syntax

```
world.timeStepMode
```

Zugriff: Abrufen/Festlegen

Beschreibung

Diese Eigenschaft gibt den aktuellen Zeitschrittmodus zurück, der im init()-Aufruf an die Welt festgelegt ist. Der Zeitschrittmodus kann „#automatic“ oder „#equal“ lauten.

#equal – Durchläuft die Physikwelt in Zeitschritten, die der Parameter timeStep des init-Aufrufs festlegt.

#automatic - Durchläuft die Physikwelt gemäß der tatsächlich verstrichenen Zeit.

Beispiel

Die folgende Anweisung stellt den aktuellen Zeitschrittmodus für die Physikwelt auf „equal“ ein:

```
--Lingo Syntax
member("PhysicsWorld").timeStepMode = #equal
//JavaScript Syntax
member("PhysicsWorld").timeStepMode = symbol("equal");
```

Siehe auch

init(),timeStep,subSteps

Methoden der Physikwelt

destroy()

Syntax

```
<void>world.destroy()
```

Beschreibung

Diese Funktion stoppt die Simulation für die Welt und gibt alle Ressourcen in der Welt frei. Um die Simulation erneut zu starten, muss init() für die Welt aufgerufen werden.

Parameter

Keine

Beispiel:

```
--Lingo Syntax
member("PhysicsWorld").destroy()
//Javascript syntax
member("PhysicsWorld").destroy();
```

Hinweis: *Wenn kein destroy()-Aufruf erfolgt und der Film in der Authoring-Umgebung von Director wiedergegeben wird, tritt beim Erstellen von Physikobjekten ein Fehler auf.*

Siehe auch

`init()`

getSimulationTime()

Syntax

```
<float>world.getSimulationTime()
```

Beschreibung

Gibt die verstrichene Simulationsdauer der Physikwelt zurück.

Parameter

Keine

Beispiel

```
--Lingo Syntax
nFloatTime = member("PhysicsWorld").getSimulationTime()
//Javascript syntax
var nFloatTime = member("PhysicsWorld").getSimulationTime();
```

Siehe auch

`simulate()`

init()

Syntax

```
world.init(string 3dmembername, vector scalingFactor, symbol timeStepMode, float timeStep, int
substepCount).
world.init(3dmemberref, vector scalingFactor, symbol timeStepMode,float timeStep, int
substepCount).
```

Beschreibung

Diese Funktion initialisiert die Physikwelt mit dem angegebenen 3D-Darsteller. Jeder Physikdarsteller sollte mit einem eindeutigen 3D-Darsteller verknüpft sein, der durch 3dmembername angegeben wird.

- Sie müssen die init()-Funktion mit world.destroy() beenden, bevor Sie eine weitere init()-Funktion aufrufen.
- Verwenden Sie die Methode resetworld() in einer Physiksimulation nicht zwischen den Methoden init() und destroy().

- Es wird empfohlen, Physik nicht auf Modelle mit Schlüsselbildanimationen oder Animationen des Typs „Knochen-Player“ anzuwenden.

Hinweis: *Ein Fehler vom Typ „-2“ wird ausgelöst, wenn die Welt nicht initialisiert ist, bevor auf eine Methode oder Eigenschaft eines Physikobjekts zugegriffen wird.*

Parameter

Parameter	Beschreibung
3dmembername	Erforderlich. String/Darstellerreferenz zur Angabe eines eindeutigen 3D-Darstellers
scalingFactor	Erforderlich. Vektor zur Angabe des Maßstabsfaktors der 3D-Welt in der Physikwelt
timeStep	Wert in Millisekunden
timeStepMode	Erforderlich. #equal – Durchläuft die Physikwelt in Zeitschritten, die der Parameter timeStep des init-Aufrufs festlegt. #automatic - Durchläuft die Physikwelt gemäß der tatsächlich verstrichenen Zeit.
subStepcount	Erforderlich. Ganzzahliger Wert, der aus Gründen der Genauigkeit berechnet wird.

Beispiel:

```
--Lingo Syntax
member("PhysicsWorld").init("3dmembername",vector(1.0,1.0,1.0),#equal,0.33,5)
member("PhysicsWorld").init(member("3dWorld"),vector(1.0,1.0,1.0),#equal,0.33,5)
member("PhysicsWorld").init("3dmembername",vector(1.0,1.0,1.0),#automatic,0,5)
member("PhysicsWorld").init(member("3dWorld"),vector(1.0,1.0,1.0),#automatic,0,5)
//Javascript syntax
member("PhysicsWorld").init("3dmembername",vector(1.0,1.0,1.0),symbol("equal"),0.167,5);
member("PhysicsWorld").init(member("3dWorld"),vector(1.0,1.0,1.0),symbol("equal"),0.33, 5);
member("PhysicsWorld").init("3dmembername",vector(1.0,1.0,1.0),symbol("automatic"),0,5);
member("PhysicsWorld").init(member("3dWorld"),vector(1.0,1.0,1.0),symbol("automatic"),0, 5);
```

Siehe auch

`destroy()`, `subSteps`, `timeStep`, `timeStepMode`

PauseSimulation()

Syntax

```
world.PauseSimulation()
```

Beschreibung

Diese Funktion unterbricht die Physiksimulation für diese Physikwelt. Simulate()-Aufrufe werden ignoriert, bis ResumeSimulation () aufgerufen wurde.

Parameter

Keine

Beispiel

```
--Lingo Syntax
member("PhysicsWorld").PauseSimulation()
//Javascript syntax
member("PhysicsWorld").PauseSimulation();
```

Siehe auch

ResumeSimulation ()

ResumeSimulation ()

Syntax

```
<void>world.ResumeSimulation ()
```

Beschreibung

Diese Funktion setzt die Simulation der Physikwelt fort, die durch den world.PauseSimulation()-Aufruf unterbrochen wurde.

Parameter

Keine

Beispiel

```
--Lingo Syntax
member("PhysicsWorld").ResumeSimulation ()
//Javascript syntax
member("PhysicsWorld").ResumeSimulation ();
```

Siehe auch

PauseSimulation()

simulate()

Syntax

```
world.simulate()
```

Beschreibung

Der Aufruf dieser Funktion durchläuft die Physikwelt gemäß bestimmter Zeitschritte. Das Zeitintervall hängt dabei von dem im init()-Aufruf festgelegten timeStepMode-Wert ab.

Für timeStepMode können die beiden folgenden Werte angegeben werden:

#equal Der simulate()-Aufruf durchläuft die Physikwelt in Zeitschritten, die der Parameter timeStep des init()-Aufrufs festlegt.

#automatic Der simulate()-Aufruf durchläuft die Physikwelt gemäß der tatsächlich verstrichenen Zeit.

- Sämtliche Änderungen an der Physikwelt werden erst wiedergespiegelt, wenn die Methode simulate() aufgerufen wird. Wenn Sie beispielsweise einen Impuls mithilfe der Methode applyimpulse() auf einen Festkörper anwenden, ändert sich der Wert von linearVelocity erst nach dem Aufruf von simulate().

- Sie sollten die Methode simulate() in jedem exitFrame() aufrufen, um eine ordnungsgemäße Simulation sicherzustellen.

Hinweis: *Ein Fehler vom Typ „-3“ wird angezeigt, wenn der Aufruf von „simulate()“ fehlschlägt.*

Parameter

Keine

Beispiel

```
--Lingo Syntax
member("PhysicsWorld").simulate()
//Javascript syntax
member("PhysicsWorld").simulate();
```

Siehe auch

`init()`, `subSteps`, `timeStep`, `timeStepMode`

Funktionen für die Verwaltung von Festkörperobjekten

Hinweis: *Achten Sie darauf, dass sich die Namen für den Festkörper und das Gelände unterscheiden. Bei Verwendung desselben Namens für Festkörper und Gelände wird der Fehler „-4“ zurückgegeben, der auf einen ungültigen Parameter hinweist.*

addProxyTemplate()

Syntax

```
physicsWorld.addProxyTemplate(3dmember, proxyTemplate, string proxyname)
```

Beschreibung

Fügt die mithilfe der Methode `createproxyTemplate()` erstellte Proxy-Vorlage dem angegebenen 3D-Darsteller hinzu.

Parameter

Parameter	Beschreibung
3dmember	Erforderlich. Eine Referenz auf einen 3D-Darsteller.
proxyTemplate	Erforderlich. Die Proxy-Vorlage wird mithilfe von „createProxyTemplate()“ erstellt.
proxyName	Erforderlich. Der Name in dem der Proxy gespeichert werden muss.

Ein Rückgabewert von 1 bedeutet Erfolg, der Wert 0 bedeutet einen Fehler.

Diese Methode fügt die mithilfe der Methode „createproxyTemplate()“ erstellte Proxy-Vorlage dem angegebenen 3D-Darsteller hinzu.

Eine neue Modellressource vom Typ „#physicsmesh“ wird dem 3D-Darsteller hinzugefügt.

Hinweis: *Die Methode „addProxyTemplate()“ speichert die Proxy-Vorlage „proxyTemplate“ als neue Modellressource „#physiksmesh“ im angegebenen 3D-Welt-Darsteller. Um den neu hinzugefügten Proxy zu speichern, muss entweder die Methode „saveWorld()“ oder die Methode „saveW3d()“ aufgerufen wird.*

Beispiel

```
--Lingo syntax
member("PhysicsWorld").addProxyTemplate(member("3dWorld") , ProxyTemplate,
"saved_proxyTemplateName")
--creates a modelresource of type #physicsmesh in the 3d world specified.
--modelresource("saved_proxyTemplateName").meshtype gives the type of physicsmesh . i.e
#concave and #convex.
--Use saveWorld() or saveW3d() methods to save the proxy added to the 3d world.
-- member("3dWorld").saveW3d(the moviepath & "proxyWorld.w3d")
// JavaScript syntax
member("PhysicsWorld").addProxyTemplate(member("3dWorld") ,
ProxyTemplate,"saved_proxyTemplateName");
```

Hinweis: *Die beste Vorgehensweise ist es, die Proxy-Vorlage mithilfe des Nachrichtenfensters zu speichern.*

Siehe auch

`createRigidBodyFromProxy()`, `createProxyTemplate()`, `loadProxyTemplate()`

createRigidBody()

Syntax

```
<RigidBody> world.createRigidBody(string rigidbodyname, string
3Dmodelname,symbolBodyProxy,symbol bodyType,symbol flipNormals)
```

Beschreibung

Diese Methode erstellt einen Festköper mit der angegebenen Proxyform.

Der Modifizierer „#meshdeform“ muss dem Modell vor dem Erzeugen eines Festkörpers hinzugefügt werden. Ein Fehler vom Typ „-21“ wird angezeigt, wenn der Modifizierer „#meshdeform“ nicht dem 3D-Modell hinzugefügt wird.

Hinweis: *Informationen bezüglich des Erstellens dynamischer, konkaver Starrkörper finden Sie unter „CreateRigidBodyFromProxy()“.*

Parameter

Parameter	Beschreibung
3dModelName	Erforderlich. String, der den Namen des 3D-Modells angibt, in dem der Festkörper erstellt wird
rigidBodyName	Erforderlich. String, der den Namen des Festkörpers angibt

Parameter	Beschreibung

symbolBodyProxy	Erforderlich. Symbol mit folgenden möglichen Werten:
	`#box`
	`#Sphere`
	`#convexShape`
	`#concaveShape`

bodyType	Erforderlich. #static – wenn ein statisches oder stationäres Objekt erstellt werden muss. ***Hinweis:*** *Wenn Sie ein statisches Objekt durch Ändern seiner Position verschieben, ändert sich zwar die Position des Festkörpers, die Position des 3D-Modells bleibt jedoch unverändert.* #dynamic – wenn ein verschiebbares Objekt erstellt werden muss.
flipNormals	Optional. Dieser Parameter kehrt die Flächennormalen um. Bei Objekten mit dem Proxy „#concaveshape“ werden Kollisionen in positiver Richtung der Flächennormalen erkannt. Die Durchdringung wird von der gegenüberliegenden Seite betrachtet. Dieses Verhalten können Sie mithilfe von „FlipNormals“ umkehren. Dieser Parameter kann nur für die Proxys „#convexshape“ und „#concaveshape“ verwendet werden. Bei anderen Proxys wird ein Fehler zurückgegeben.

Es wird ein Verweis auf den Festkörper zurückgegeben. Schlägt die Erstellung fehl, wird „void“ zurückgegeben.

Hinweis: *Ein Festkörper mit dem Proxy „#concaveshape“ muss als „#static“ definiert sein. Wenn Sie versuchen, ein konkaves dynamisches Objekt zu erstellen, wird der Fehler „-28“ zurückgegeben. Weitere Informationen hierzu finden Sie unter createRigidBodyFromProxy(). zum Erstellen dynamischer, konkaver Starrkörper.*

Beispiel

```
--Lingo syntax
objRigidBody = member("PhysicsWorld").createRigidBody("RigidBodyA","ModelA",#sphere,#static)
objRigidBody = member("PhysicsWorld").createRigidBody("RigidBodyA","ModelA",#box,#dynamic)
objRigidBody =
member("PhysicsWorld").createRigidBody("RigidBodyA","ModelA",#convexshape,#dynamic)
objRigidBody
=member("PhysicsWorld").createRigidBody("RigidBodyA","ModelA",#concaveshape,#static,#flipNor
mals) --The last parameter,#flipNormals,is optional.
//JavaScript Syntax
var objRigidBody =
member("PhysicsWorld").createRigidBody("RigidBodyA","ModelA",symbol("sphere"),symbol("static
"));
var objRigidBody =
member("PhysicsWorld").createRigidBody("RigidBodyA","ModelA",symbol("box"),symbol("dynamic")
);
var objRigidBody =
member("PhysicsWorld").createRigidBody("RigidBodyA","ModelA",symbol("convexshape"),symbol("d
ynamic'));
var objRigidBody =
member("PhysicsWorld").createRigidBody("RigidBodyA","ModelA",symbol("concaveshape"),symbol("
static"),symbol("flipNormals"));//The last parameter, symbol("flipNormals"),is optional.
```

Siehe auch

deleteRigidBody()

createRigidBodyFromProxy()

Syntax

```
world.createRigidBodyFromProxy(string rigidbodyname, string 3Dmodelname,symbol
bodyType,ProxyTemplate,symbol flipNormals)
```

Beschreibung

Diese Methode erstellt einen Festköper mit der angegebenen Proxyform.

Fügen Sie dem Modell den Modifizierer `#meshdeform` hinzu, bevor Sie einen Starrkörper erstellen. Ein Starrkörper lässt sich aus einem Proxy mithilfe folgender Prozeduren erzeugen:

- Erstellen Sie eine Proxy-Vorlage und verwenden Sie diese direkt, um einen Starrkörper zur Laufzeit zu erzeugen.
- Erstellen Sie eine Proxy-Vorlage der Modellressource und speichern Sie diese unter 3D-Welt (W3D) zur Bearbeitungszeit. Verwenden Sie diese Vorlage anschließend dazu, den eigentlichen Starrkörper zur Laufzeit zu erzeugen. Dieses Vorgehen wird zum Erstellen konkaver und konvexer Starrkörper empfohlen. Weitere Proxytypen werden von dieser Methode nicht unterstützt.

  Die Skalierungs- und Modelltransformationen werden automatisch während des Erstellens des Starrkörpers angewendet. Die Proxy-Vorlage lässt sich erneut verwenden, wenn Starrkörper erstellt werden, die von der gleichen Modellressource abgeleitet wurden.

  Das Erstellen und Speichern eines 3D-Darstellers während der Bearbeitungszeit zur späteren Verwendung bedeutet einen Leistungsgewinn.

  Der Fehler „-30“ wird ausgelöst, wenn die Proxy-Vorlage ungültig ist oder es wird beim Erstellen eines dynamischen, konkaven Starrkörpers die Proxy-Vorlage „#concave“ übergeben.

Parameter

Parameter	Beschreibung
3dModelName	Erforderlich. String, der den Namen des 3D-Modells angibt, in dem der Festkörper erstellt wird
rigidBodyName	Erforderlich. String, der den Namen des Festkörpers angibt

Parameter	Beschreibung
bodyType	Erforderlich. #static – wenn ein statisches oder stationäres Objekt erstellt werden muss. ***Hinweis:*** *Wenn Sie ein statisches Objekt durch Ändern seiner Position verschieben, ändert sich zwar die Position des Festkörpers, die Position des 3D-Modells bleibt jedoch unverändert.* #dynamic – wenn ein verschiebbares Objekt erstellt werden muss.
ProxyTemplate	Erforderlich. Eine Proxy-Vorlage-Objekt, dass über die Methode „createProxyTemplate() “ ermittelt wurde. ***Hinweis:*** *Weitere Informationen zum Erstellen und Speichern von Proxys finden Sie bei den Methoden „createProxyTemplate()“, „addProxyTemplate()“ und „loadProxyTemplate()“.*
flipNormals	Optional. Dieser Parameter kehrt die Flächennormalen um. Bei Objekten mit dem Proxy „#concaveshape“ werden Kollisionen in positiver Richtung der Flächennormalen erkannt. Die Durchdringung wird von der gegenüberliegenden Seite betrachtet. Dieses Verhalten können Sie mithilfe von „FlipNormals“ umkehren. Dieser Parameter kann nur für die Proxys „#convexshape“ und „#concaveshape“ verwendet werden. Bei anderen Proxys wird ein Fehler zurückgegeben.

Es wird ein Verweis auf den Festkörper zurückgegeben. Schlägt die Erstellung fehl, wird „void“ zurückgegeben.

Hinweis: *Das Angeben von „#concave“ in der Proxy-Vorlage eines dynamischen, konkaven Starrkörpers führt zu einem Fehler. Die Funktionen „createProxyTemplate()“ und „addProxyTemplate()“ sollten im Idealfall im Nachrichtenfenster verwendet werden und nicht Teil des Laufzeitcodes sein. Mit diesem Ansatz wird die Zusatzbelastung der konvexen Dekomposition dynamischer Starrkörper gemindert. Auch das Speichern des Proxys in einer separaten 3D-Welt eignet sich dazu, einer übermäßigen Nutzung der Modellressourcen vorzubeugen.*

Beispiel

```
--Lingo syntax
--During Author time save the proxy template to a 3D world using steps 1 to 3
--Step1: Create a proxy template from a model resource
ProxyTemplate = member("PhysicsWorld").createProxyTemplate(modelResource Reference
,#convexDecomposed,[#concavity:2, #depth:8,#mergeVolume:2])
--Step2: Add the proxy template to the 3D world
member("PhysicsWorld").addProxyTemplate(member("3DproxyWorld"), ProxyTemplate, "proxyName")
--Step3: Save the 3D world
member("3DproxyWorld").saveWorld()
--At runtime load the saved proxy template and create the rigid body.
--Load the proxy template from the saved 3D world
Saved_proxy = member("PhysicsWorld").loadProxyTemplate("proxyName",member("3DproxyWorld"))
--Create the rigid body from proxy template
objRB = member("PhysicsWorld").createRigidBodyFromProxy("RigidBodyA","ModelA", #dynamic,
Saved_proxy)
//Javascript syntax
Var objRB = member("PhysicsWorld").createRigidBodyFromProxy("RigidBodyA"," ModelA",
symbol("dynamic"), Saved_proxy) ;
```

Siehe auch

`createProxyTemplate()`,`loadProxyTemplate()`,`addProxyTemplate()`

createProxyTemplate()

Syntax

```
physicsWorld.createProxyTemplate( ModelResourceRef, #proxyTemplateSymbol, decomposition parameter list)
```

Beschreibung

Diese Methode erstellt einen Festköper mit der angegebenen Proxyform.

Parameter

Parameter	Beschreibung
ModelResourceRef	Erforderlich. Eine Referenz des Modells, für das ein dynamischer, konkaver proxy erstellt werden soll.
proxyTemplateSymbol	Erforderlich. Das Symbol nimmt einen der folgenden Werte an: `#convexDecomposed` - Erstellt eine dynamische, konkave Proxy-Vorlage mithilfe einer Dekompositionsfunktion mit konvexem Gitternetz. `#concave` - Erstellt eine statische, konkave Proxy-Vorlage. `#convex` - Erstellt eine konvexe Gitternetz-Proxy-Vorlage.
Konvexe Dekomposition für dynamische, konkave Starrkörper	Optional. Property list `#depth` - Standardwert: 3, Bereich: 0-10. `#concavity` - Standardwert: 10. Bereich: 0-100. `#mergeVolume` - Standardwert: 10. Bereich: 0-100.

Eine Referenz des erstellten Proxys für die angegebene Modellressource. Schlägt die Erstellung fehl, wird „void“ zurückgegeben.

Hinweis: *Ein -31-Fehler wird erzeugt, wenn die verwendete Modellressource ungültig ist. Ein „-29“-Fehler wird ausgelöst, wenn ein Duplikat der Modellressource vorhanden ist.*

Der zurückgegebene Proxy ist eine Eigenschaftsliste, welche die Eigenschaften des aus dem Modell erzeugten Proxys enthält.

numMeshes Anzahl konvexer Gitternetze, die zum Erstellen der konkaven Form nötig sind.

vertexList Eine Liste mit den Scheitelpunkten der einzelnen Gitternetzpunkte.

indexList Eine Liste mit den Indizes der einzelnen Gitternetzpunkte.

meshType Symbol returning `#concave` or `#convex`.

proxyTemplate Symbol	proxyTemplate.meshType	Starrkörper, die mit der Proxy-Vorlage „proxyTemplate“ erzeugt werden können.
#convexdecomposed	#convex	Konkav-dynamisch, konkav-statisch
#concave	#concave	Konkav-statisch
#convex	#convex	Konvex-statisch, konvex-dynamisch

Director 3D-Grundelemente des Modellressourcentyps, die nicht „#mesh“ lauten, können nicht zum Erstellen von Proxyvorlagen mithilfe dieser Methode verwendet werden. Ein „-31“-Fehler wird ausgelöst.

Beispiel

```
--Lingo syntax
ProxyTemplate = member("PhysicsWorld").createProxyTemplate(modelResource Reference
,#convexDecomposed,[#concavity:4, #depth:6,#mergeVolume:4])
// JavaScript syntax
var ProxyTemplate = member("PhysicsWorld").
createProxyTemplate(member("3dWorld").getPropRef("modelResource",2)
,symbol("convexDecomposed"),propList(symbol("concavity"),2, symbol("depth"),8,
symbol("mergeVolume"),2));
```

Siehe auch

createRigidBodyFromProxy(), loadProxyTemplate(), addProxyTemplate()

deleteRigidBody()

Syntax

```
<void>world.deleteRigidBody(string rigidbodyname)
<void>world.deleteRigidBody(RigidBodyref r)
```

Beschreibung

Entfernt den Festkörper aus der Physikwelt.

Hinweis: *Stellen Sie sicher, dass Sie das mit dem Festkörper verknüpfte 3D-Modell nicht vor dem Festkörper löschen.*

Parameter

Einer der folgenden Parameter muss angegeben werden:

Parameter	Beschreibung
Rigidbodyname	String, der den Namen des Festkörpers angibt
Rigibodyref	Verweis auf den Festkörper, der mit createrigidbody() erstellt wurde

Beispiel

```
--Lingo Syntax
objRB = member("PhysicsWorld").createRigidBody("RigidBodyA","ModelA",#sphere,#static)
member("PhysicsWorld").deleteRigidBody("RigidBodyA")
```

oder

```
objRB = member("PhysicsWorld").createRigidBody("RigidBodyA","ModelA",#sphere,#static)
member("PhysicsWorld").deleteRigidBody(objRB);
//Javascript Syntax
var objRB = member("PhysicsWorld").createRigidBody("RigidBodyA",
"ModelA",symbol("sphere"),symbol("static"));
member("PhysicsWorld").deleteRigidBody("RigidBodyA");
```

oder

```
var objRB = member("PhysicsWorld").createRigidBody("RigidBodyA",
"ModelA",symbol("sphere"),symbol("static"));
member("PhysicsWorld").deleteRigidBody(objRB);
```

Siehe auch

createRigidBody()

getRigidBody()

Syntax

```
<RigidBody>world.getRigidBody(string rigidBodyName)
```

Beschreibung

Gibt den durch den Festkörpernamen bezeichneten Festkörper zurück. Der Rückgabewert lautet „void“, wenn der angegebene Festkörper nicht vorhanden ist.

Parameter

Parameter	Beschreibung
Rigidbodyname	Erforderlich. String, der den Namen des Festkörpers angibt

Gibt den Verweis auf den Festkörper zurück.

Beispiel

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
```

getRigidBodies()

Syntax

```
<list of RigidBodies>world.getRigidBodies()
```

Beschreibung

Gibt eine Liste von Festkörperobjekten zurück, die derzeit in der Physikwelt vorhanden sind.

Parameter

Keine

Gibt eine Liste mit Bezügen auf die Festkörperobjekte der Physikwelt zurück.

Beispiel

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBodies()
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBodies();
```

getSleepingBodies()

Syntax

```
<list of RigidBodies> world.getSleepingBodies()
```

Beschreibung

Gibt eine Liste inaktiver Festkörperobjekte zurück.

Parameter

Keine

Gibt eine Liste mit Bezügen auf die inaktiven Festkörperobjekte der Physikwelt zurück.

Beispiel

```
--Lingo Syntax
objRB = member("PhysicsWorld").getSleepingBodies()
//Javascript Syntax
var objRB = member("PhysicsWorld").getSleepingBodies ();
```

loadProxyTemplate()

Syntax

```
physicsWorld.loadProxyTemplate( string proxyname, 3dmember)
```

Beschreibung

Diese Methode lädt die Proxy-Vorlage, die mit den Methoden `createproxyTemplate()` und`addProxyTemplate()` erstellt wurde.

Parameter

Parameter	Beschreibung
proxyName	Erforderlich. Der Name, in dem der Proxy gespeichert werden muss.
3dmember	Erforderlich. Eine Referenz auf den 3D-Darsteller.

Liefert eine Referenz auf die Modellressource `#physicsmesh` zurück..

Beispiel

```
--Lingo syntax
Saved_proxyTemplate  =
member("PhysicsWorld").loadProxyTemplate("saved_proxyTemplateName",member("3dWorld"))
//JavaScript syntax
var Saved_proxyTemplate  =
member("PhysicsWorld").loadProxyTemplate("saved_proxyTemplateName",member("3dWorld"));
```

Siehe auch

createRigidBodyFromProxy(), createProxyTemplate(), addProxyTemplate()

Festkörpereigenschaften

angularDamping

Syntax

```
r.angularDamping
```

Zugriff: Abrufen/Festlegen

Typ: Fließkommazahl

Beschreibung

Ruft den Winkeldämpfungsgrad für den Körper ab bzw. legt ihn fest. Dies ist die für Winkelbewegungen geltende Entsprechung der linearen Dämpfung.

Beispiel

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
put objRB.angularDamping
objRB.angularDamping = 0.5
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
put(objRB.angularDamping);
objRB.angularDamping = 0.5;
```

Siehe auch

linearDamping

angularMomemtum

Syntax

```
r.angularMomemtum
```

Zugriff: Abrufen/Festlegen

Typ: vector

Beschreibung

Gibt den Winkelimpuls für den Festkörper zurück bzw. legt ihn fest.

Beispiel

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
put objRB.linearVelocity
objRB.angularMomentum = vector(-100,0,0)
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
put(objRB.linearVelocity);
objRB.angularMomentum =  vector(-100,0,0);
```

Siehe auch

linearMomemtum

angularVelocity

Syntax

```
r.angularVelocity
```

Zugriff: Abrufen/Festlegen

Typ: vector

Beschreibung

Gibt den Winkelgeschwindigkeitsvektor für den Festkörper zurück bzw. legt ihn fest.

Beispiel

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
put objRB. angularVelocity
objRB.angularVelocity = vector(-100,0,0)
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
put(objRB. angularVelocity);
objRB.angularVelocity =  = vector(-100,0,0);
```

Siehe auch

linearVelocity

axisAffinity

Syntax

```
r.axisAffinity
```

Zugriff: Abrufen/Festlegen

Typ: Boolesch

Beschreibung

Diese Eigenschaft dynamischer, konkaver Starrkörper steuert die Affinität eines Starrkörpers zu einer Achse, wenn eine Kraft auf den Körper einwirkt.

Starrkörper sind stets affin zur Weltachse. Wenn Sie die Eigenschaft `axisAffinity` eines Starrkörpers auf „False“ setzen, wird diese Affinität zur Weltachse kompensiert.

Hinweis: *Der Standardwert der Eigenschaft `axisAffinity` ist „True“, weil die Einstellung „False“ mehr Rechenleistung beansprucht.*

Beispiel

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
put objRB.axisAffinity
objRB.axisAffinity = TRUE

//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
put(objRB.axisAffinity);
objRB.axisAffinity = true;
```

centerOfMass

Syntax

```
r.centerOfMass
```

Zugriff: Abrufen/Festlegen

Typ: vector

Beschreibung

Diese Eigenschaft gibt den Schwerpunkt des Festkörpers in Festkörperkoordinaten an.

Beispiel

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
put objRB.centerOfMass
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
put(objRB.centerOfMass);
```

Siehe auch

`mass`

contactTolerance

Syntax

```
r.contacttolerance
```

Zugriff: Abrufen/Festlegen

Typ: Fließkommazahl

Beschreibung

Legt die Durchdringungstiefe für den Festkörper bei einer Kollision mit einem anderen Körper fest.

Beispiel

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
objRB.contactTolerance = 0.01
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
objRB.contactTolerance = 0.01;
```

friction

Syntax

```
r.friction
```

Zugriff: Abrufen/Festlegen

Typ: Fließkommazahl

Beschreibung

Gibt den Reibungskoeffizienten für den Festkörper zurück bzw. legt ihn fest. Die „friction"-Eigenschaft nimmt Werte von 0-1 ein.

Beispiel

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
put objRB.friction
objRB.friction= 0.5
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody ("RigidBodyA");
put(objRB.friction);
objRB.friction= 0.5;
```

Siehe auch

restitution

isPinned

Syntax

```
r.ispinned
```

Zugriff: Abrufen/Festlegen

Typ: Boolesch

Beschreibung

Ruft ab, ob der Körper fixiert ist, bzw. legt die Fixierung fest. Bei der Einstellung TRUE ist der Körper in seiner aktuellen Ausrichtung an seiner Position fixiert. Externe Kräfte ändern weder die Position noch die Ausrichtung des Körpers.

Beispiel

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
put objRB.isPinned
objRB.isPinned = TRUE
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
put(objRB.isPinned);
objRB.isPinned = true;
```

isSleeping

Syntax

```
r.isSleeping
```

Zugriff: Abrufen/Festlegen

Typ: Boolesch

Beschreibung

Gibt den inaktiven Zustand eines Festkörpers zurück bzw. legt ihn fest.

Hinweis: *Der Ruhezustand eines Festkörpers hängt von seiner Form ab. Bei einem Festkörper, der mit einem Box-Proxy erstellt wurde, dauert die Inaktivität beispielsweise länger als bei einer Kugel.*

Beispiel

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
put objRB.isSleeping
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody ("RigidBodyA");
put(objRB. isSleeping)
```

linearDamping

Syntax

```
r.linearDamping
```

Zugriff: Abrufen/Festlegen

Typ: Fließkommazahl

Beschreibung

Ruft den aktuellen Wert der linearen Dämpfung für den Körper ab bzw. legt ihn fest. Dies ist der Dämpfungsgrad, um den der Körper bei linearen Bewegungen in der Welt verlangsamt wird.

Beispiel

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
put objRB.linearDamping
objRB.linearDamping = 0.5
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
put(objRB.linearDamping);
objRB.linearDamping = 0.5;
```

Siehe auch

angularDamping

linearMomemtum

Syntax

```
r.linearMomemtum
```

Zugriff: Abrufen/Festlegen

Typ: vector

Beschreibung

Gibt den linearen Impuls für den Festkörper zurück bzw. legt ihn fest.

Beispiel

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
put objRB.linearMomentum
objRB.linearMomentum = vector(-100,0,0)
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
put(objRB.linearMomentum);
objRB.linearMomentum =  vector(-100,0,0);
```

Siehe auch

angularMomemtum

linearVelocity

Syntax

```
r.linearVelocity
```

Zugriff: Abrufen/Festlegen

Typ: vector

Beschreibung

Gibt den linearen Geschwindigkeitsvektor für den Festkörper zurück bzw. legt ihn fest.

Beispiel

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
put objRB.linearVelocity
objRB.linearVelocity = vector(-100,0,0)
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
put(objRB. linearVelocity);
objRB.linearVelocity = vector(-100,0,0);
```

Siehe auch

angularVelocity

mass

Syntax

```
r.mass
```

Zugriff: Abrufen/Festlegen

Typ: Fließkommazahl

Beschreibung

Legt die Masse des Festkörpers fest bzw. ruft sie ab. Der Standardwert für die Masse ist Null. Legen Sie einen Wert größer Null für diese Eigenschaft fest. Wenn Sie keinen Wert angeben, führt dies zu Ungenauigkeiten bei der Simulation.

Beispiel

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
put objRB.mass
objRB.mass = objRB.mass + 20
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
put(objRB.mass);
objRB.mass = objRB.mass + 20;
```

Siehe auch

centerOfMass

model

Syntax

```
r.model
```

Zugriff: Abrufen

Beschreibung

Gibt das 3D-Modell zurück, das mit dem betreffenden Festkörper verknüpft ist.

Beispiel

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
objModel = objRB.model
put objModel.name
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
var objModel = objRB.model
put(objModel.name)
```

name

Syntax

```
r.name
```

Zugriff: Abrufen

Typ: String

Beschreibung

Gibt den Namen des Festkörpers zurück.

Beispiel

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
put objRB.name
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
put(objRB.name);
```

orientation

Syntax

```
r.orientation
```

Zugriff: Abrufen/Festlegen

Typ: Liste

Beschreibung

Ruft die Ausrichtung des Festkörpers in der Physikwelt ab bzw. legt sie fest. Die Ausrichtungseigenschaft enthält eine Liste mit dem Richtungsvektor sowie dem Winkel zwischen den Achsen des Festkörpers und der Weltachse.

Beispiel

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
objRB.orientation = [vector(10,0,0),45]
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
objRB.orientation = list(vector(10,0,0),45);
```

Hinweis: *Wenn Sie die Position oder Ausrichtung eines statischen Festkörpers ändern, wird die Modellposition dadurch nicht geändert. Die Modellposition oder -ausrichtung muss explizit geändert werden.*

position

Syntax

```
r.position
```

Zugriff: Abrufen/Festlegen

Typ: vector

Beschreibung

Ruft die Position des Festkörpers in der Physikwelt ab bzw. legt sie fest. Die Position stellt einen Vektor(x,y,z) in Weltkoordinaten dar.

Beispiel

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
objRB.position = vector(0,0,0)
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
objRB.position = vector(0,0,0);
```

Hinweis: *Wenn Sie die Position oder Ausrichtung eines statischen Festkörpers ändern, wird die Modellposition dadurch nicht geändert. Die Modellposition oder -ausrichtung muss explizit geändert werden.*

Eigenschaften

Syntax

```
r.properties
```

Zugriff: Abrufen

Typ: Eigenschaftsliste

Beschreibung

Gibt eine Eigenschaftsliste mit den Eigenschaften des betreffenden Festkörpers zurück. rb.properties gibt eine Liste aller Eigenschaften aus, die auf dem Typ der Form „rigidbody“ basieren.

rb.shape	rb.properties
#box	[#length, #width, #height, #center]
#sphere	[#radius, #center]
#convex	[#numvertices, #numfaces, #vertexlist, #face
#concave	[#numvertices, #numfaces, #vertexlist, #face]

Beispiel

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
put objRB.properties
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
put(objRB.properties);
```

restitution

Syntax

```
r.restitution
```

Zugriff: Abrufen/Festlegen

Typ: Fließkommazahl

Beschreibung

Gibt die Stoßzahl für den Festkörper zurück bzw. legt sie fest. Die „restitution"-Eigenschaft nimmt Werte von 0-1 ein.

Beispiel

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
put objRB.restitution
objRB.restitution = 0.5
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
put(objRB.restitution);
objRB.restitution = 0.5;
```

Form

Syntax

```
r.shape
```

Zugriff: Abrufen

Werte: #sphere, #box, #convexshape oder #concaveshape

Beschreibung

Gibt die Form des Festkörpers zurück.

Beispiel

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
    put objRB.shape
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
put(objRB.shape);
```

sleepMode

Syntax

```
r.sleepMode
```

Zugriff: Abrufen/Festlegen

Beschreibung

Diese Eigenschaft ruft den Ruhezustand für einen Festkörper ab bzw. legt diesen fest. Mögliche Werte sind „#energy“ oder „#linearvelocity“. Der Standardwert ist „#energy“.

#energy Die kinetische Energie des Festkörpers dient dazu, seinen Ruhezustand zu ermitteln.

#linearvelocity Die lineare Geschwindigkeit des Festkörpers dient dazu, seinen Ruhezustand zu ermitteln.

Beispiel

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
objRB.sleepMode = #energy
//Javascript syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
objRB.sleepMode = symbol("energy");
```

Siehe auch

sleepThreshold

sleepThreshold

Syntax

```
r.sleepThreshold
```

Zugriff: Abrufen/Festlegen

Typ: Fließkommazahl

Beschreibung

Steht für den Schwellenwert, unterhalb dessen der Festkörper inaktiv wird. Dieser Wert ist entweder die kinetische Energie oder die linear Geschwindigkeit des Festkörpers, je nachdem, welcher Wert für die Eigenschaft sleepMode festgelegt wurde. Diese Eigenschaft kann nur auf dynamische Objekte angewandt werden. Der Standardwert lautet 0.000.

Beispiel

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
objRB.sleepMode = #energy
objRB.sleepThreshold = 10.0
//Javascript syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
objRB.sleepMode = symbol("energy");
objRB.sleepThreshold = 10.0;
```

Siehe auch

sleepMode

type

Syntax

```
r.type
```

Zugriff: Abrufen

Beschreibung

Gibt den Festkörpertyp zurück. Dieser kann „#static“ oder „#dynamic“ lauten.

Beispiel

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
put objRB.type
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody ("RigidBodyA");
put(objRB.type);
```

userData

Syntax

```
r.userData
```

Zugriff: Abrufen/Festlegen

Typ: Objekt

Beschreibung

Diese Eigenschaft ermöglicht Benutzern, benutzerdefinierte Daten abzurufen bzw. festzulegen und mit dem betreffenden Festkörper zu verknüpfen.

Beispiel

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
--Assigning Integer Value
objRB.userData = 0.5
--Assigning String Value
objRB.userData = "My User data Value"
--Assigning Reference Value
objRB.userData = member("PhysicsWorld').getRigidBody("RigidBodyB")
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
--Assigning Integer Value
objRB.userData = 0.5;
--Assigning String Value
objRB.userData = "My User data Value";
--Assigning Reference Value
objRB.userData = member("PhysicsWorld').getRigidBody("RigidBodyB");
```

Festkörpermethoden

applyForce()

Syntax

```
<int>r.applyForce(vector vForce, vector vposition)
```

Beschreibung

Legt die angegebene Kraft an der Position an, welche der Positionsvektor in Festkörperkoordinaten festlegt. Wird die Position nicht festgelegt, wird die Kraft am Schwerpunkt des Festkörpers angelegt.

Parameter

Parameter	Beschreibung
vForce	Erforderlich. Vektorwert, welcher die Größe und Richtung der angelegten Kraft angibt.
vPosition	Optional. Vektor, der den Punkt angibt, an welchem die Kraft angelegt wird.

Beispiel

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
objRB.applyForce(vector(-100,0,0),vector(0,0,0))
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
objRB.applyForce(vector(-100,0,0),vector(0,0,0));
```

applyLinearImpulse()

Syntax

```
< int >r.applyLinearImpulse(vector vImpulse, vector vposition)
```

Beschreibung

Legt den angegebenen linearen Impuls an der Position an, welche der Positionsvektor in Festkörperkoordinaten festlegt. Wird die Position nicht festgelegt, wird der Impuls am Schwerpunkt des Festkörpers angelegt.

Parameter

Parameter	Beschreibung
vImpulse	Erforderlich. Vektorwert, welcher die Größe und Richtung der angelegten Kraft angibt.
vPosition	Optional. Vektor, der den Punkt angibt, an welchem die Kraft angelegt wird.

Beispiel

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
objRB.applyLinearImpulse(vector(-100,0,0),vector(0,0,0))
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
objRB.applyLinearImpulse(vector(-100,0,0),vector(0,0,0));
```

applyAngularImpulse()

x5848 | Lingo_r_linearimpulse

Syntax

```
< int >r.applyAngularImpulse(vector vImpulse)
```

Beschreibung

Legt den festgelegten Winkelimpuls am Schwerpunkt an.

Parameter

Parameter	Beschreibung
vImpulse	Erforderlich. Vektor, der den angelegten Winkelimpuls angibt.

Beispiel

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
objRB.applyAngularImpulse(vector(-100,100,0))
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
objRB.applyAngularImpulse (vector(-100,0,0));
```

applyTorque()

Syntax

```
< int >r.applyTorque(vector vTorque)
```

Beschreibung

Diese Funktion legt das angegebene Drehmoment am Schwerpunkt des betreffenden Festkörpers an.

Parameter

Parameter	Beschreibung
vTorque	Erforderlich. Vektor, der das angelegte Drehmoment angibt.

Beispiel

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
objRB.applyTorque(vector(-100,100,0))
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody("RigidBodyA");
objRB.applyTorque(vector(-100,0,0));
```

attemptMoveTo()

Syntax

```
< int >r.attemptMoveTo(vector vposition)
```

Beschreibung
Diese Funktion versucht, den Festkörper an die angegebenen Position zu verschieben, sofern keine Kollision entsteht. Wird an der angegebenen Position eine Kollision festgestellt, wird das Objekt nicht verschoben.

Parameter

Parameter	Beschreibung
vPosition	Erforderlich. Vektor, welcher den Punkt angibt, an den das Objekt verschoben werden soll.

Beispiel

```
--Lingo Syntax
objRB = member("PhysicsWorld").getRigidBody("RigidBodyA")
objRB.attemptMoveTo(vector(-100,0,0))
//Javascript Syntax
var objRB = member("PhysicsWorld").getRigidBody ("RigidBodyA");
objRB.atemptMoveTo(vector(-100,0,0));
```

Beschränkungsmethoden

ConstraintDesc

Syntax

```
ConstraintDesc(sname,ObjectA,ObjectB,vPointA,vPointB,fStiffness,fDamping)
```

vPointA und vPointB müssen auf den oder innerhalb der Festkörper liegen. Sie werden in Objektkoordinaten angegeben. Das Verhalten ist undefiniert, wenn die Punkte nicht auf den Festkörpern liegen.

In linearen Verbindung wird die Ausrichtung von Objekt B im Abhängigkeit zu Objekt A durch den Parameter `orientation` in der Methode `createLinearJoint` festgelegt.

Beschreibung
Es muss ein Beschränkungsdeskriptorobjekt angelegt werden, bevor eine Feder-, Winkel- oder lineare Verbindung erstellt wird.

Hinweis: *Ein Fehler vom Typ „-5“ wird angezeigt, wenn Sie versuchen, einen Beschränkungsdeskriptor mit einem vorhandenen Namen zu erstellen.*

Parameter
Alle Beschränkungen werden mithilfe eines Beschränkungsdeskriptors erstellt. Dabei handelt es sich um einen Typ mit folgendem Inhalt:

Parameter	Beschreibung
string Name	Name der Beschränkung
ObjectA	Festkörper A, mit dem die Beschränkung verbunden wird
ObjectB	Festkörper B, mit dem die Beschränkung verbunden wird

Parameter	Beschreibung
vector PoCA	Punkt, an welchem die Beschränkung an Festkörper A angebracht wird
vector PoCB	Punkt, an welchem die Beschränkung an Festkörper B angebracht wird
float Stiffness	Wert der Steifigkeit der Beschränkung
float damping	Wert der Dämpfung der Beschränkung

Beispiel

```
--Lingo Syntax
-- Spring constraint Descriptor between two rigid bodies' b1 and b2.
SpringDesc =
ConstraintDesc("SpringDesc",b1,b2,vector(6.0,0.0,0.0),vector(8.0,0.0,0.0),500,1)
-- Linear Joint constraint Descriptor between a rigid body b1 and point in world.
LJointDesc =
ConstraintDesc("LJoinDesc",b1,void,vector(6.0,0.0,0.0),vector(8.0,0.0,0.0),500,1)
//JavaScript Syntax
//Spring constraint Descriptor between two rigid bodies' b1 and b2.
var SpringDesc =
ConstraintDesc("SpringDesc",b1,b2,vector(6.0,0.0,0.0),vector(8.0,0.0,0.0),500,1);
```

Hinweis:

- Um eine Beschränkung zwischen einem Punkt in der Welt und einem Festkörper zu erstellen, legen Sie „void“ als Wert von „ObjectB“ fest.
- Ein Beschränkungsdeskriptor kann nicht zwischen einem Punkt und einem statischen Festkörper, zwischen zwei statischen Festkörpern oder zwischen zwei Punkten in der Welt erstellt werden.

Siehe auch

`createAngularJoint()`,`createLinearJoint()`,`createSpring()`,`createD6Joint`

createAngularJoint()

Syntax

```
<AngularJoint> world.createAngularJoint(ConstraintDesc desc , float length)
```

Beschreibung

Diese Funktion erstellt eine Winkelverbindung. Winkelverbindungen sind in winkliger Richtung frei beweglich. Ihre linearen Bewegungen sind dagegen beschränkt. Die Länge, die als Beschränkung eingehalten werden muss, wird als zweiter Parameter angegeben.

Parameter

Parameter	Beschreibung
ConstraintDesc	Erforderlich. Referenz auf das Beschränkungsdeskriptorobjekt.
restlength	Erforderlich. Fließkommawert, welcher die Ruhelänge der Verbindung angibt.

Beispiel

```
--Lingo Syntax
-- Angular Joint constraint Descriptor between two rigid bodies' b1 and b2.
AJointDesc =
ConstraintDesc("AJointDesc",b1,b2,vector(6.0,0.0,0.0),vector(8.0,0.0,0.0),500,1)
--Create a angular joint with a rest length of 5.
member("PhysicsWorld").createAngularJoint(AJointDesc, 5.0)
//JavaScript Syntax
//Angular Joint constraint Descriptor between two rigid bodies' b1 and b2.
var AJointDesc =
ConstraintDesc("AJointDesc",b1,b2,vector(6.0,0.0,0.0),vector(8.0,0.0,0.0),500,1);
//Create a angular Joint with a rest length of 5.
member("PhysicsWorld").createAngularJoint (AJointDesc,5.0);
```

Siehe auch

ConstraintDesc, createLinearJoint(), createSpring(), createD6Joint

createLinearJoint()

Syntax

```
<LinearJoint> world.createLinearJoint(ConstraintDesc desc , list orientation)
```

Beschreibung

Diese Funktion erstellt eine lineare Verbindung. Lineare Verbindungen sind in linearer Richtung frei beweglich. Ihre Winkelbewegungen sind dagegen beschränkt. Die Winkel, die als Beschränkung eingehalten werden müssen, werden als zweiter Parameter angegeben.

Parameter

Parameter	Beschreibung
ConstraintDesc	Erforderlich. Referenz auf das Beschränkungsdeskriptorobjekt.
Orientation	Erforderlich. Listenwert, welcher die Ausrichtung der Verbindung angibt.

Beispiel

```
--Lingo Syntax
-- Angular Joint constraint Descriptor between two rigid bodies' b1 and b2.
LJointDesc =
ConstraintDesc("LJointDesc",b1,b2,vector(6.0,0.0,0.0),vector(8.0,0.0,0.0),500,1)
--Create a linear joint with angular constraint in x,y axis at an angle of 45.
member("PhysicsWorld").createLinearJoint(LJointDesc,[vector(1,1,0),45])
//JavaScript Syntax
//Linear Joint constraint Descriptor between two rigid bodies' b1 and b2.
var LJointDesc =
ConstraintDesc("LJointDesc",b1,b2,vector(6.0,0.0,0.0),vector(8.0,0.0,0.0),500,1);
//Create a linear joint with angular constraint in x,y axis at an angle of 45.
member("PhysicsWorld").createLinearJoint (LJointDesc, list(vector(1,1,0),45));
```

Siehe auch

createAngularJoint() , ConstraintDesc, createSpring(), createD6Joint

createSpring()

Syntax

```
<Spring> world.createSpring( ConstraintDesc desc, symbol forceExertionMode , float restLength)
```

Beschreibung

Erstellt eine Feder mit den im Beschränkungsdeskriptor angegebenen Details.

Parameter

Parameter	Beschreibung
ConstraintDesc	Erforderlich. Referenz auf das Beschränkungsdeskriptorobjekt.
forceexertionmode	Hat einen der folgenden Symbolwerte: #kDuringCompression – Die Rückstellkraft wird nur angelegt, wenn die tatsächliche Länge kleiner als die Ruhelänge zwischen Objekt A und Objekt B ist. #kDuringExpansion – Die Rückstellkraft wird nur angelegt, wenn die tatsächliche Länge größer als die Ruhelänge zwischen Objekt A und Objekt B ist. #kBoth – Die Rückstellkraft wird in beiden oben genannten Fällen angelegt.
restlength	Erforderlich. Fließkommawert, welcher die Ruhelänge der Feder angibt.

Beispiel

```
--Lingo Syntax
-- Spring constraint Descriptor between two rigid bodies' b1 and b2.
SpringDesc =
ConstraintDesc("SpringDesc",b1,b2,vector(6.0,0.0,0.0),vector(8.0,0.0,0.0),500,1)
--Create a spring which has expansion and compression with a rest length of 5.
member("PhysicsWorld").createSpring(SpringDesc,#kboth,5.0)
//JavaScript Syntax
//Spring constraint Descriptor between two rigid bodies' b1 and b2.
SpringDesc =
ConstraintDesc("SpringDesc",b1,b2,vector(6.0,0.0,0.0),vector(8.0,0.0,0.0),500,1);
//Create a spring which has expansion and compression with a rest length of 5.
member("PhysicsWorld").createSpring(SpringDesc,symbol("kboth"),5.0);
```

Siehe auch

createAngularJoint(),createLinearJoint(),ConstraintDesc,createD6Joint

deleteConstraint()

Syntax

```
<int>deleteConstraint(string constraintname)
<int>deleteConstraint(Constraint constraintref)
```

Beschreibung

Diese Funktion löscht die angegebene Beschränkung anhand des Beschränkungsnamens.

Parameter

Parameter	Beschreibung
ConstraintName	Erforderlich. Zeichenfolgenname der Beschränkung

oder

Parameter	Beschreibung
Constraintref	Erforderlich. Referenz auf das Beschränkungsobjekt.

Beispiel

```
--Lingo Syntax
member("PhysicsWorld").deleteConstraint("ConstraintDesc")
//JavaScript Syntax
member("PhysicsWorld").deleteConstraint("ConstraintDesc");
```

deleteSpring()

Syntax

```
<int>deleteSpring(string springname)
<int>deleteSpring(Spring spring)
```

Beschreibung

Diese Funktion löscht die angegebene Feder anhand des Federnamens.

Parameter

Parameter	Beschreibung
Springname	Erforderlich. String, welcher den Namen den erstellten Feder angibt.

Beispiel

```
--Lingo Syntax
-- Spring constraint descriptor between two rigid bodies' b1 and b2.
SpringDesc =
ConstraintDesc("SpringDesc",b1,b2,vector(6.0,0.0,0.0),vector(8.0,0.0,0.0),500,1)
--Create a spring which has expansion and compression with a rest length of 5.
member("PhysicsWorld").createSpring(SpringDesc,#kboth,5.0)
--Deletes the spring
member("PhysicsWorld").deleteSpring("SpringDesc")
//JavaScript Syntax
//Spring constraint descriptor between two rigid bodies' b1 and b2.
SpringDesc =
ConstraintDesc("SpringDesc",b1,b2,vector(6.0,0.0,0.0),vector(8.0,0.0,0.0),500,1);
//Create a spring which has expansion and compression with a rest length of 5.
member("PhysicsWorld").createSpring(SpringDesc,symbol("kboth"),5.0);
//Deletes the spring
member("PhysicsWorld").deleteSpring("SpringDesc");
```

getConstraint()

Syntax

```
<AngularJoint / LinearJoint>getConstraint(string constraintname)
```

Beschreibung

Diese Funktion gibt die Beschränkung mit dem angegebenen Namen zurück.

Parameter

Parameter	Beschreibung
ConstraintName	Erforderlich. Zeichenfolgenname der Beschränkung

Gibt den Verweis auf die Verbindung zurück.

Beispiel

```
--Lingo Syntax
objJoint = member("PhysicsWorld").getConstraint("ConstraintDesc")
//JavaScript Syntax
var objJoint = member("PhysicsWorld").getConstraint("ConstraintDesc");
```

Siehe auch

ConstraintDesc

getConstraints()

Syntax

```
<list of Constraints> world.getConstraints(symbol constraintType)
```

Beschreibung

Diese Funktion gibt alle Beschränkung vom angegebenen Typ zurück. Der Typ kann „#linearjoint“, „#angularjoint“ oder „#d6joint“ lauten.

Parameter

Parameter	Beschreibung
constraintType	Optional. Symbol, welches den Beschränkungstyp angibt. Wird dieser Parameter nicht angegeben, werden alle in der Physikwelt vorhandenen Beschränkungen (außer Federn) zurückgegeben.

Beispiel

```
--Lingo Syntax
--Returns all the constraints in the world.
lstJoint = member("PhysicsWorld").getConstraints()
--Returns all the Linear joint constraints in the world.
lstJoint = member("PhysicsWorld").getConstraints(#linearjoint)
//JavaScript Syntax
var lstJoint = member("PhysicsWorld").getConstraints(symbol("angularjoint"));
```

Siehe auch

ConstraintDesc

getSpring()

Syntax

```
<Spring > world.getSpring(string springname)
```

Beschreibung

Gibt das Federobjekt mit dem angegebenen Namen zurück.

Parameter

Parameter	Beschreibung
Springname	Erforderlich. String, welcher den Namen den erstellten Feder angibt.

Beispiel

```
--Lingo Syntax
objSpring = member("PhysicsWorld").getSpring("SpringName")
//JavaScript Syntax
var objSpring = member("PhysicsWorld").getSpring("SpringName");
```

Siehe auch

ConstraintDesc

getSprings()

Syntax

```
<list of Springs> world.getSprings();
```

Beschreibung

Diese Funktion gibt eine Liste aller in der Physikwelt vorhandenen Federn zurück.

Standardmäßig ist die Kollision zwischen allen in der Welt vorhandenen Körpern aktiviert. Mit diesen Funktionen kann die Kollision zwischen Körperpaaren aktiviert bzw. deaktiviert werden.

Parameter

Keine

Beispiel

```
--Lingo Syntax
lSprings= member("PhysicsWorld").getSprings()
//JavaScript Syntax
var lSprings = member("PhysicsWorld").getSprings();
```

Beschränkungseigenschaften

constraintType

Syntax

```
c.constraintType
```

Zugriff: Abrufen

Beschreibung

Gibt den Beschränkungstyp zurück. Der Typ kann „#linearjoint“, „#angularjoint“ oder „#d#6joint“ lauten.

Beispiel

```
--Lingo Syntax
objJoint = member("PhysicsWorld").getConstraint("lJoint01")
put objJoint.constraintType
//JavaScript Syntax.
objJoint = member("PhysicsWorld").getConstraint("lJoint01");
put(objJoint.constraintType);
```

damping

Syntax

```
c.damping
```

Zugriff: Abrufen/Festlegen

Typ: Fließkommazahl

Beschreibung

Gibt die Dämpfung der Beschränkung zurück.

Beispiel

```
--Lingo Syntax
objJoint = member("PhysicsWorld").getSpring("lJoint01")
put objJoint.damping
//JavaScript Syntax.
objJoint = member("PhysicsWorld").getSpring("lJoint01");
put(objJoint.damping);
```

name

Syntax

```
c.name
```

Zugriff: Abrufen

Typ: String

Beschreibung
Gibt den Namen der Beschränkung zurück.

Beispiel
```
--Lingo Syntax
objConstraint = member("PhysicsWorld").getConstraint("ConstraintA")
put objConstraint.name
//Javascript Syntax
var objConstraint = member("PhysicsWorld").getConstraint("ConstraintA");
put(objConstraint.name);
```

objectA

Syntax
```
s.objectA
```

Zugriff: Abrufen

Typ: Festkörper

Beschreibung
Gibt eine Referenz auf den mit dem Gelenk verbundenen Starrkörper „A“ zurück.

Beispiel
```
--Lingo Syntax
--Change one end of the Joint to already created rigidbody - rigidbodyX
objJoint = member("PhysicsWorld").getConstraint("lJoint01")
put objJoint.objectA
//JavaScript Syntax.
--Change one end of the Joint to already created rigidbody - rigidbodyX
objJoint = member("PhysicsWorld").getConstraint("lJoint01");
put(objJoint.objectA);
```

objectB

Syntax
```
s.objectB
```

Zugriff: Abrufen

Typ: Festkörper

Beschreibung
Gibt eine Referenz auf den mit der Feder verbundenen Starrkörper „A“ zurück.

Beispiel

```
--Lingo Syntax
--Change one end of the Joint to already created rigidbody - rigidbodyX
objJoint = member("PhysicsWorld").getConstraint ("lJoint01")
put objJoint.objectB
--Change one end of the Joint to a worldPoint
objJoint = member("PhysicsWorld").getConstraint("lJoint01")
put objJoint.objectB
//JavaScript Syntax.
--Change one end of the joint to already created rigidbody - rigidbodyX
objJoint = member("PhysicsWorld").getConstraint("lJoint01");
put(objJoint.objectB);
```

pointA

Syntax

```
s.pointA
```

Zugriff: Abrufen

Typ: vector

Beschreibung

Gibt den Punkt zurück, an welchem die Verbindung an Objekt A angebracht ist.

Beispiel

```
--Lingo Syntax
objJoint = member("PhysicsWorld").getConstraint("lJoint01")
put objJoint.pointA
//JavaScript Syntax.
objJoint = member("PhysicsWorld").getConstraint("lJoint01");
put(objJoint.pointA);
```

pointB

Syntax

```
s.pointB
```

Zugriff: Abrufen

Typ: vector

Beschreibung

Gibt den Punkt zurück, an welchem die Verbindung an Objekt B angebracht ist.

Beispiel

```
--Lingo Syntax
objJoint = member("PhysicsWorld").getConstraint("lJoint01")
put objJoint.pointB
//JavaScript Syntax.
objJoint = member("PhysicsWorld").getConstraint("lJoint01");
put(objJoint.pointB);
```

Eigenschaften

Syntax

```
c.properties
```

Zugriff: Abrufen/Festlegen

Typ: Eigenschaftsliste

Beschreibung

Gibt eine Eigenschaftsliste zurück, welche die Eigenschaften der Verbindung enthält. Bei linearen Verbindung wird beispielsweise die Ausrichtung, bei Winkelverbindungen die Länge zurückgegeben.

Beispiel

```
--Lingo Syntax
objJoint = member("PhysicsWorld").getConstraint("lJoint01")
put objJoint.properties
//JavaScript Syntax.
objJoint = member("PhysicsWorld").getConstraint("lJoint01");
put(objJoint.properties);
```

stiffness

Syntax

```
s.stiffness
```

Zugriff: Abrufen/Festlegen

Typ: Fließkommazahl

Beschreibung

Gibt die Steifigkeit der Beschränkung zurück.

Beispiel

```
--Lingo Syntax
objJoint = member("PhysicsWorld").getSpring("lJoint01")
put objJoint.stiffness
//JavaScript Syntax.
objJoint = member("PhysicsWorld").getSpring("lJoint01");
put(objJoint.stiffness);
```

Federeigenschaften

damping

Syntax

```
s.damping
```

Zugriff: Abrufen/Festlegen

Typ: Fließkommazahl

Beschreibung
Steht für die Dämpfung der Feder.

Beispiel
```
--Lingo Syntax
objSpring = member("PhysicsWorld").getSpring("SpringDesc")
put objSpring.damping
//JavaScript Syntax
var objSpring = member("PhysicsWorld").getSpring("SpringDesc");
put(objSpring.damping);
```

flags

Syntax
```
s.flags
```

Zugriff: Abrufen/Festlegen

Beschreibung
Steht für die Komprimierungs-/Dehnungskräfte, die an der Feder anliegen. Gibt entweder „#kDuringCompression“, „#kDuringExpansion“ oder „#kBoth“ zurück.

Beispiel
```
--Lingo Syntax
objSpring = member("PhysicsWorld").getSpring("SpringDesc")
put objSpring.flags
//JavaScript Syntax
var objSpring = member("PhysicsWorld").getSpring("SpringDesc");
put(objSpring.flags);
```

name

Syntax
```
s.name
```

Zugriff: Abrufen

Typ: String

Beschreibung
Gibt den Namen der Feder zurück.

Beispiel
```
--Lingo Syntax
objSpring = member("PhysicsWorld").getSpring("SpringA")
put objSpring.name
//Javascript Syntax
var objSpring = member("PhysicsWorld").getSpring("SpringA");
put(objSpring.name);
```

objectA

Syntax

```
s.objectA
```

Zugriff: Abrufen/Festlegen

Typ: Festkörper

Beschreibung

Gibt den Namen des Festkörpers „A“ zurück, der an der Feder angebracht ist.

Beispiel

```
--Lingo Syntax
--Change one end of the spring to already created rigidbody - rigidbodyX
objSpring = member("PhysicsWorld").getSpring("Spring01")
put objSpring.objectA
objSpring.objectA = rigidbodyX
--Change one end of the spring to a worldPoint
objSpring = member("PhysicsWorld").getSpring("Spring01")
put objSpring.objectA
objSpring.objectA = void
//JavaScript Syntax.
--Change one end of the spring to already created rigidbody - rigidbodyX
objSpring = member("PhysicsWorld").getSpring("Spring01");
put(objSpring.objectA);
objSpring.objectA = rigidbodyX;
```

objectB

Syntax

```
s.objectB
```

Zugriff: Abrufen/Festlegen

Typ: Festkörper

Beschreibung

Gibt den Namen des Festkörpers „B“ zurück, der an der Feder angebracht ist.

Beispiel

```
--Lingo Syntax
--Change one end of the spring to already created rigidbody - rigidbodyX
objSpring = member("PhysicsWorld").getSpring("Spring01")
put objSpring.objectB
objSpring.objectB = rigidbodyX

--Change one end of the spring to a worldPoint
objSpring = member("PhysicsWorld").getSpring("Spring01")
put objSpring.objectB
objSpring.objectB = void
//JavaScript Syntax.
--Change one end of the spring to already created rigidbody - rigidbodyX
objSpring = member("PhysicsWorld").getSpring("Spring01");
put(objSpring.objectB);
objSpring.objectB = rigidbodyX;
```

pointA

Syntax

```
s.pointA
```

Zugriff: Abrufen/Festlegen

Typ: vector

Beschreibung

Gibt den Punkt zurück, an welchem die Feder an Objekt A angebracht ist.

Beispiel

```
--Lingo Syntax
objSpring = member("PhysicsWorld").getSpring("Spring01")
put objSpring.pointA
//JavaScript Syntax.
objSpring = member("PhysicsWorld").getSpring("Spring01");
put(objSpring.pointA);
```

pointB

Syntax

```
s.pointB
```

Zugriff: Abrufen/Festlegen

Typ: vector

Beschreibung

Gibt den Punkt zurück, an welchem die Feder an Objekt B angebracht ist.

Beispiel

```
--Lingo Syntax
objSpring = member("PhysicsWorld").getSpring("Spring01")
put objSpring.pointB
//JavaScript Syntax.
objSpring = member("PhysicsWorld").getSpring("Spring01");
put(objSpring.pointB);
```

restLength

Syntax

```
s.restLength
```

Zugriff: Abrufen/Festlegen

Typ: Fließkommazahl

Beschreibung

Steht für die Länge der Feder im Ruhezustand.

Beispiel

```
--Lingo Syntax
objSpring = member("PhysicsWorld").getSpring("SpringDesc")
put objSpring.restLength
//JavaScript Syntax
var objSpring = member("PhysicsWorld").getSpring("SpringDesc");
put(objSpring.restLength);
```

stiffness

Syntax

```
s.stiffNess
```

Zugriff: Abrufen/Festlegen

Typ: Fließkommazahl

Beschreibung

Steht für die Steifigkeit der Feder.

Beispiel

```
--Lingo Syntax
objSpring = member("PhysicsWorld").getSpring("SpringDesc")
put objSpring.stiffness
//JavaScript Syntax
var objSpring = member("PhysicsWorld").getSpring("SpringDesc");
put(objSpring.stiffness);
```

Kollisions- und Kollisionsrückrufmethoden

disableCollision()

Syntax

```
<int>world .disableCollision(RigidBody a, RigidBody b)
<int>world .disableCollision(string rigidbodyname1, string rigidbodyname2)
```

Beschreibung

Haben A und B den Wert „void“, wird die Kollision zwischen allen in der Welt vorhandenen Festkörpern deaktiviert.

Hat B den Wert „void“, deaktiviert diese Methode die Kollision zwischen Festkörper A und allen Festkörpern.

Werden A und B angegeben, deaktiviert diese Methode die Kollision zwischen diesen beiden Festkörpern, falls sie zuvor aktiviert war.

Hinweis: *„disableCollision( )“ deaktiviert alle Rückrufe. Durch die Verwendung von enableCollision( ) werden jedoch nicht automatisch alle Rückrufe aktiviert. Zur Aktivierung der Rückrufe müssen Sie enableCollisionCallback( ) separat aufrufen.*

Hinweis:

Parameter

Hängt von der Anzahl der Festkörper ab, für die Kollisionen deaktiviert werden müssen.

Beispiel

```
--Lingo Syntax
--All collisions are disabled.
member("PhysicsWorld").disableCollision()
--Collisions disabled for rigid body A
member("PhysicsWorld").disableCollision(rigidbodyA)
--Collisions disabled for rigidbody A and rigidbody B
member("PhysicsWorld").disableCollision(rigidbodyA,rigidbodyB)
//JavaScript Syntax
//All collisions are disabled.
member("PhysicsWorld").disableCollision();
//Collisions disabled for rigid body A
member("PhysicsWorld").disableCollision(rigidbodyA);
//Collisions disabled for rigidbody A and rigidbody B
member("PhysicsWorld").disableCollision(rigidbodyA,rigidbodyB);
```

Siehe auch

enableCollision()

disableCollisionCallback()

Syntax

```
<int>world .disableCollisionCallback(RigidBody a, RigidBody b)
<int>world . disableCollisionCallback (string rigidbodyname1, string rigidbodyname2)
```

Beschreibung

Wenn keine Parameter übergeben werden, werden Kollisionsrückrufberichte für alle Festkörper deaktiviert, die vor diesem Aufruf erstellt wurden.

Werden Parameter an diese Methode übergeben, werden Kollisionsrückrufe nur für den bzw. die angegebenen Festkörper deaktiviert.

Parameter

Hängt von der Anzahl der Festkörper ab, für die Kollisionsrückrufe deaktiviert werden müssen.

Beispiel

```
--Lingo Syntax
--All collisions callbacks are disabled for rigid bodies created before this call.
member("PhysicsWorld").disableCollisionCallback()
--Collisions callback disabled for rigid body A with all other bodies.
member("PhysicsWorld").disableCollisionCallback(rigidbodyA)
--Collisions  callback disabled between rigidbody A and rigidbody B.
member("PhysicsWorld").disableCollisionCallback(rigidbodyA,rigidbodyB)
//JavaScript Syntax
//All collision callbacks are disabled.
member("PhysicsWorld").disableCollisionCallback();
//Collision callbacks disabled for rigid body A
member("PhysicsWorld").disableCollisionCallback(rigidbodyA);
//Collision callbacks disabled between rigidbody A and rigidbody B
member("PhysicsWorld").disableCollisionCallback(rigidbodyA,rigidbodyB);
```

Siehe auch

enableCollisionCallback(),getCollisionCallbackDisabledPairs(),getCollisionDisabledPairs(),registerCollisionCallback(),removeCollisionCallback()

enableCollision()

Syntax

```
<int>world .enableCollision(RigidBody a, RigidBody b)
<int>world .enableCollision(string rigidbodyname1, string rigidbodyname2)
```

Beschreibung

Werden keine Parameter übergeben, werden alle Kollisionen aktiviert. Andernfalls werden Kollisionen nur für den bzw. die angegebenen Festkörper aktiviert.

Parameter

Hängt von der Anzahl der Festkörper ab, für die Kollisionen aktiviert werden müssen.

Beispiel

```
--Lingo Syntax
--All collisions are enabled.
member("PhysicsWorld").enableCollision()
--Collisions enabled for rigid body A with all other bodies.
member("PhysicsWorld").enableCollision(rigidbodyA)
--Collisions enabled between rigidbody A and rigidbody B.
member("PhysicsWorld").enableCollision(rigidbodyA,rigidbodyB)
//JavaScript Syntax
//All collisions are enabled.
member("PhysicsWorld").enableCollision();
//Collisions enabled for rigid body A
member("PhysicsWorld").enableCollision(rigidbodyA);
//Collisions enabled for rigidbody A and rigidbody B
member("PhysicsWorld").enableCollision(rigidbodyA,rigidbodyB);
```

Siehe auch

disableCollision()

enableCollisionCallback()

Syntax

```
<int>world .enableCollisionCallback(RigidBody a, RigidBody b)
<int>world . enableCollisionCallback(string rigidbodyname1, string rigidbodyname2)
```

Beschreibung

Wenn keine Parameter übergeben werden, werden Kollisionsrückrufberichte für alle Festkörper aktiviert, die vor diesem Aufruf erstellt wurden.

Werden Parameter an diese Methode übergeben, werden Kollisionsrückrufe nur für den bzw. die angegebenen Festkörper aktiviert.

Parameter

Hängt von der Anzahl der Festkörper ab, für die Kollisionsrückrufe aktiviert werden müssen.

Beispiel

```
--Lingo Syntax
--All collisions callbacks are enabled for rigid bodies created before this call.
member("PhysicsWorld").enableCollisioncallback()
--Collisions callback enabled for rigid body A with all other bodies.
member("PhysicsWorld").enableCollisioncallback(rigidbodyA)
--Collisions  callback enabled between rigidbody A and rigidbody B.
member("PhysicsWorld").enableCollisioncallback(rigidbodyA,rigidbodyB)
//JavaScript Syntax
//All collision callbacks are enabled.
member("PhysicsWorld").enableCollisioncallback();
//Collision callbacks enabled for rigid body A
member("PhysicsWorld").enableCollisioncallback(rigidbodyA);
//Collision callbacks enabled for rigidbody A and rigidbody B
member("PhysicsWorld").enableCollisioncallback(rigidbodyA,rigidbodyB);
```

Siehe auch

disableCollisionCallback(),getCollisionDisabledPairs(),getCollisionCallbackDisabledPairs(),registerCollisionCallback(),removeCollisionCallback()

getCollisionCallbackDisabledPairs()

Syntax

```
<[[RigidBody,RigidBody]…]>world .getCollisionCallbackDisabledPairs()
```

Beschreibung

Diese Methode gibt die Liste aller Festkörperpaare zurück, für welche Kollisionsrückrufe deaktiviert wurden.

Parameter

Keine

Beispiel

```
--Lingo Syntax
lstPairs = member("PhysicsWorld").getCollisionCallbackDisabledPairs()
lstPair1 = lstPairs[1]
//JavaScript Syntax
var lstPairs = member("PhysicsWorld").getCollisionCallbackDisabledPairs();
var lstPair1 = lstPairs(0);
```

Siehe auch

enableCollisionCallback(),disableCollisionCallback(),getCollisionDisabledPairs(),registerCollisionCallback(),removeCollisionCallback()

getCollisionDisabledPairs()

Syntax

```
<[[RigidBody,RigidBody]…]>world .getCollisionDisabledPairs()
```

Beschreibung

Diese Methode gibt die Liste aller Festkörperpaare zurück, für welche die Kollision deaktiviert wurde.

Parameter

Keine

Beispiel

```
--Lingo Syntax
lstPairs = member("PhysicsWorld").getCollisionDisabledPairs()
lstPair1 = lstPairs[1]
//JavaScript Syntax
var lstPairs = member("PhysicsWorld").getCollisionDisabledPairs();
var lstPair1 = lstPairs(0);
```

Siehe auch

enableCollisionCallback(), disableCollisionCallback(), getCollisionCallbackDisabledPairs(), registerCollisionCallback(), removeCollisionCallback()

registerCollisionCallback()

Syntax

```
<int>world .registerCollisionCallback( #collisionCallback,<script reference>)
```

Beschreibung

Diese Funktion registriert die Rückruffunktion, die aufgerufen werden muss, um die Kollisionen zu melden. Es darf nur eine Kollisionsrückrufprozedur für die Physikszene geben.

Die Art des an der Kollision beteiligten Starrkörpers lässt sich mithilfe der Methode `ILK(<rigidBody_handler>)` ermitteln.

Die Informationen über die Kollisionen werden in einem Kollisionsbericht an diese Funktion übergeben, der folgende Angaben enthält:

Parameter

Parameter	Beschreibung
CollisionCallback	Erforderlich. Gibt den Namen der Kollisionsrückrufprozedur an.
Script reference	Optional. Hierbei handelt es sich um ein Verhalten oder eine Instanz eines Parent-Skripts. Wird dieser Parameter nicht angegeben, wird davon ausgegangen, dass die Rückruffunktion in einem Filmskript definiert ist.

Aufbau eines Kollisionsberichts:

ObjectA	An der Kollision beteiligtes Objekt A
ObjectB	An der Kollision beteiligtes Objekt B
list (vector<float,float,float>)	Die Kontaktpunkte zwischen Objekt A und Objekt B
list (vector<float,float,float>)	Die Kontaktnormalen für jeden der oben angegebenen Punkte.

Hinweis: *Wenn innerhalb der Kollisionsrückrufprozedur ein Skript Laufzeitfehler verursacht (weil z. B. eine Eigenschaft nicht gefunden wird), werden die Fehlermeldungen nicht angezeigt und die Prozedurausführung wird an dem Punkt abgebrochen, an welchem der Fehler auftrat. Alle nachfolgenden Anweisungen werden nicht ausgeführt.*

Nehmen Sie möglichst keine Änderungen an der 3D-Szene innerhalb der Kollisionsrückrufprozedur wie resetworld(), deleteModel(), eleterigidBody() usw. vor.

Beispiel

```
--Lingo Syntax
member("PhysicsWorld").registercollisionCallback(#collisioncallback)
--Movie Script
On collisionCallback collisionReport
CollisionA = collisionReport[1]
R1 = CollisionA.objectA
R2 = CollisionA.objectB
lstPoints = CollisionA.contactPoints
lstNormals = CollisionA.contactNormals
end
//JavaScript Syntax
member("PhysicsWorld").registerCallback(symbol("collisioncallback"));
//Movie Script
function collisionCallback(collisionreport)
{
var CollisionA = collisionReport(0);
var R1 = CollisionA.objectA;
var R2 = CollisionA.objectB;
var lstPoints = CollisionA.contactPoints;
var lstNormals = CollisionA.contactNormals;
}
```

Siehe auch

enableCollisionCallback(), disableCollisionCallback(), getCollisionDisabledPairs(), getCollisionDisabledPairs(), removeCollisionCallback()

removeCollisionCallback()

Syntax

```
<int>world .removeCollisionCallback()
```

Beschreibung

Diese Funktion entfernt den zu einem früheren Zeitpunkt registrierten Rückruf.

Parameter

Keine

Beispiel

```
--Lingo Syntax
member("PhysicsWorld").removeCollisionCallback()
//JavaScript Syntax
member("PhysicsWorld").removeCollisionCallback();
```

Siehe auch

enableCollisionCallback(), disableCollisionCallback(), getCollisionDisabledPairs(), getCollisionCallbackDisabledPairs(), registerCollisionCallback()

Geländemethoden

Hinweis: *Achten Sie darauf, dass sich die Namen für den Festkörper und das Gelände unterscheiden. Bei Verwendung desselben Namens für Festkörper und Gelände wird der Fehler „-4“ zurückgegeben, der auf einen ungültigen Parameter hinweist.*

createTerrain

Syntax

```
createTerrain(terrainName, terrainDesc, position, orientation, rowScale, columnScale,
heightScale)
```

Beschreibung

Diese Methode erstellt das Gelände als statischen Festkörper und verwendet dabei die angegebenen Höhen-Map-Informationen.

Hinweis: *Beim Erstellen eines Gitternetzes und einer Höhen-Map mithilfe eines Höhen-Map-Bilds können Sie die Leistung verbessern, indem Sie das Gitternetz in kleinere Teile unterteilen und diese später wieder zusammensetzen.*

Parameter

Parameter	Beschreibung
terrainName	Erforderlich. String, welcher den Namen des Geländes darstellt.
terrainDesc	Erforderlich. Referenz auf den Geländedeskriptor.
position	Erforderlich. Vektor, welcher die Position des Geländes festlegt.
orientation	Erforderlich. Liste der Achsen und Winkel (z. B.: [vector(1,0,0), 45]), mit denen die Ausrichtung des Geländes beschrieben wird.
rowScale	Erforderlich. Fließkommawert, welcher den Zeilenskalierungsfaktor angibt.
columnScale	Erforderlich. Fließkommawert, welcher den Spaltenskalierungsfaktor angibt.
heightScale	Erforderlich. Fließkommawert, welcher den Höhenskalierungsfaktor angibt.

Beispiel

```
--Lingo Syntax
objTerrain =
member("PhysicsWorld").createTerrain("myterrain",terrainDesc,position,orientation,1,1,1)
//JavaScript Syntax
var objTerrain=
member("PhysicsWorld").createTerrain("myterrain",terrainDesc,position,orientation,1,1,1);
```

Siehe auch

createTerrainDesc,terrainDesc

createTerrainDesc

Syntax

```
createTerrainDesc(elevationMatrix,friction,restitution)
```

Beschreibung

Vor der Erstellung eines Geländes muss ein Geländedeskriptorobjekt erstellt werden.

Parameter

Parameter	Beschreibung	Werte:
ElevationMatrix	Erforderlich. Lingo-Matrix, welche die Höhen-Map-Informationen des Geländes darstellt.	
Friction	Erforderlich. Fließkommawert, welcher den Reibungskoeffizienten angibt.	0-1
Restitution	Erforderlich. Fließkommawert, welcher die Stoßzahl angibt.	0-1

Beispiel

```
--Lingo Syntax
tDesc = member("PhysicsWorld").createTerrainDesc(myMatrix,0.4,0.5)
//JavaScript Syntax
//Spring constraint Descriptor between two rigid bodies' b1 and b2.
var tDesc = member("PhysicsWorld").createTerrainDesc(myMatrix,0.4,0.5);
```

Siehe auch

createTerrain,terrainDesc

deleteTerrain

Syntax

```
world.deleteterrain(String terrainName)
world.deleteterrain(terrain refTerrain)
```

Beschreibung

Diese Methode löscht das Gelände aus der Physikszene. Die Methode akzeptiert den Geländenamen oder -verweis als Parameter.

Parameter

Parameter	Beschreibung
terrainName	Erforderlich. String, welcher den Namen des erstellten Geländes angibt.

oder

Parameter	Beschreibung
refTerrain	Erforderlich. Referenz auf das erstellte Geländeobjekt.

Beispiel

```
--Lingo Syntax
member("PhysicsWorld").deleteTerrain("myterrain")
//JavaScript Syntax
member("PhysicsWorld").deleteTerrain("myTerrain");
```

getTerrain

Syntax

```
<terrain refTerrain> world.getterrain(String "myterrain")
```

Beschreibung

Diese Methode gibt das Geländeobjekt mit dem angegebenen Namen zurück.

Parameter

Parameter	Beschreibung
myTerrain	: Erforderlich. String, welcher den Namen des Geländes angibt.

Beispiel

```
--Lingo Syntax
objTerrain = member("PhysicsWorld").getTerrain("myTerrain")
//JavaScript Syntax
var objTerrain = member("PhysicsWorld").getTerrain("myTerrain");
```

Siehe auch

createTerrainDesc, terrainDesc

getTerrains

Syntax

```
<list lstTerrain> world.getterrains()
```

Beschreibung

Diese Methode gibt eine Liste der in der Physikwelt vorhandenen Geländeobjekte zurück.

Parameter

Keine Parameter.

Beispiel

```
--Lingo Syntax
lstTerrain = member("PhysicsWorld").getTerrains()
//JavaScript Syntax
var lstTerrain = member("PhysicsWorld").getTerrains();
```

Geländeeigenschaften

columnScale

Syntax
```
t.columnScale
```

Zugriff: Abrufen

Beschreibung
Steht für den Spaltenskalierungsfaktor des Geländes.

Beispiel
```
--Lingo Syntax
objTerrain = Member("PhysicsWorld").getTerrain("myTerrain")
put objTerrain.columnScale
//JavaScript Syntax
Var objTerrain = Member("PhysicsWorld").getTerrain("myTerrain");
Put(objTerrain.columnScale);
```

contactTolerance

Syntax
```
t.contactTolerance
```

Zugriff: Abrufen/Festlegen

Typ: Fließkommazahl

Beschreibung
Dies ist die Durchdringungstiefe zwischen dem Gelände und einem anderen Festkörper/Gelände für die zu erkennende Kollision.

Beispiel
```
--Lingo Syntax
objTerrain = Member("PhysicsWorld").getTerrain("myTerrain")
put objTerrain.contactTolerance
//JavaScript Syntax
Var objTerrain = Member("PhysicsWorld").getTerrain("myTerrain");
Put(objTerrain.contactTolerance);
```

heightScale

Syntax
```
t.heightScale
```

Zugriff: Abrufen

Beschreibung
Steht für den Höhenskalierungsfaktor des Geländes.

Beispiel
```
--Lingo Syntax
objTerrain = Member("PhysicsWorld").getTerrain("myTerrain")
put objTerrain.heightScale
//JavaScript Syntax
Var objTerrain = Member("PhysicsWorld").getTerrain("myTerrain");
Put(objTerrain.heightScale);
```

name

Syntax
```
t.name
```

Zugriff: Abrufen

Typ: String

Beschreibung
Gibt den Namen des Geländes zurück.

Beispiel
```
--Lingo Syntax
put objTerrain.name
//Javascript Syntax
put(objTerrain.name);
```

orientation

Syntax
```
t.orientation
```

Zugriff: Abrufen

Beschreibung
Liste der Achsen und Winkel (z. B.: [vector(1,0,0), 45]), welche die Ausrichtung des Geländes beschreibt.

Beispiel
```
--Lingo Syntax
objTerrain = Member("PhysicsWorld").getTerrain("myTerrain")
put objTerrain.orientation
//JavaScript Syntax
Var objTerrain = Member("PhysicsWorld").getTerrain("myTerrain");
Put(objTerrain.orientation);
```

position

Syntax

```
t.position
```

Zugriff: Abrufen

Beschreibung

Vektor, welcher die Position des Geländes darstellt.

Beispiel

```
--Lingo Syntax
objTerrain = Member("PhysicsWorld").getTerrain("myTerrain")
put objTerrain.position
//JavaScript Syntax
Var objTerrain = Member("PhysicsWorld").getTerrain("myTerrain");
Put(objTerrain.position);
```

rowScale

Syntax

```
t.rowScale
```

Zugriff: Abrufen

Steht für den Zeilenskalierungsfaktor des Geländes.

Beispiel

```
--Lingo Syntax
objTerrain = Member("PhysicsWorld").getTerrain("myTerrain")
put objTerrain.rowScale
//JavaScript Syntax
Var objTerrain = Member("PhysicsWorld").getTerrain("myTerrain");
Put(objTerrain.rowScale);
```

terrainDesc

Syntax

```
t.terrainDesc
```

Zugriff: Abrufen

Beschreibung

Steht für das Geländedeskriptorobjekt.

Der Geländedeskriptor verfügt über folgende Attribute:

- terraindesc.elevationmatrix
- terraindesc.friction
- terraindesc.restitution
- terraindesc.numrows

- terraindesc.numcolumns

Hinweis: *„numrows“ und „numcolumns“ entsprechen den Zeilen und Spalten der Erhebungsmatrix.*

Beispiel

```
--Lingo Syntax
objTerrain = Member("PhysicsWorld").getTerrain("myTerrain")
put objTerrain.terrainDesc
//JavaScript Syntax
Var objTerrain = Member("PhysicsWorld").getTerrain("myTerrain");
Put(objTerrain.terrainDesc);
```

Siehe auch

createTerrainDesc,createTerrain

Methoden für 6-DOF-Verbindungen

createD6Joint

Die Reihenfolge, in der Festkörper als Eingabe einer 6DOF-Verbindung bereitgestellt werden, ist in Bezug auf die festgelegte Verbindungsachse und die Bewegungswerte von Bedeutung. Wenn die Reihenfolge beispielsweise „rb1,rb2“ und die Bewegungsposition „vector(10,0,0)“ lautet, ist die Richtung, in der sich der Festkörper bewegt, um die Position einzunehmen, genau entgegengesetzt der Richtung, in der er sich bewegt hätte, lautete die Reihenfolge „rb2,rb1“.

createD6Joint

Syntax

```
<d6joint> world.createD6Joint(jointName, RigidBody rb1 ,Rigidbody rb2, vector globalanchor)
```

Beschreibung

Diese Methode erstellt eine 6-DOF-Verbindung (sechs Freiheitsgrade) zwischen zwei Festkörpern oder zwischen einem Punkt in der Welt und einem Festkörper.

Hinweis: *D6-Verbindungen sind auch eine Art von Beschränkung wie lineare und Winkelverbindungen. Der Beschränkungstyp für eine D6-Verbindung lautet „#d6joint“.*

Zum Löschen der D6-Verbindung verwenden Sie deleteConstraint().

Die zusätzlichen, für D6-Verbindungen spezifischen Eigenschaften werden im Abschnitt „6-DOF-Eigenschaften“ beschrieben.

Parameter

Parameter	Beschreibung
jointName	Erforderlich. String, welcher den Namen der Verbindung angibt.
rb1	Erforderlich. Referenz auf den Festkörper, der Teil der Verbindung ist. Muss eine Verbindung zwischen einem Punkt und einem Festkörper erstellt werden, sollte dieser Parameter den Wert „void“ haben.
rb2	Erforderlich. Referenz auf den Festkörper, der Teil der Verbindung ist.
globalAnchor	Erforderlich. Vektor, welcher den Ankerpunkt der Verbindung angibt.

Beispiel

```
--Lingo Syntax
--Creates a joint between rigid bodies rb1 and rb2.
ObjJoint = member("PhysicsWorld").createD6Joint("myJoint",rb1,rb2,vector(1,1,1))
-- Creates a joint between the anchor point and rb2.
objJoint = member("PhysicsWorld").createD6Joint("myJoint",void,rb2,vector(1,1,1))
//JavaScript Syntax
//Creates a joint between rigid bodies rb1 and rb2.
var objJoint = member("PhysicsWorld").createD6Joint("myJoint",rb1,rb2,vector(1,1,1));
```

Siehe auch

`createAngularJoint()`, `createLinearJoint()`, `createSpring()`, `ConstraintDesc`, `globalAnchor`

6-DOF-Eigenschaften

axisDrive

Syntax

```
d6joint.axisDrive = [type, spring, damping, forceLimit]
```

Zugriff: Abrufen/Festlegen

Beschreibung

Listenwert, welcher die lineare Bewegung der Verbindung angibt, welche die Verbindung entlang der Verbindungsachse zur festgelegten Position bringt.

Die Liste enthält die folgenden Attribute:

type #position oder #velocity

stiffness Die während der Bewegung anzulegende Federkraft (> 0)

damping die Dämpfung der Feder (< 0)

forceLimit Die Kraft bzw. das Drehmoment für das Erreichen der angegebenen Position oder Geschwindigkeit (> 0)

Beispiel

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
objJoint.axisDrive = [#position,100,0.1,100]
//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
objJoint.axisDrive = [#position,100,0.1,100];
```

Siehe auch

biNormalDrive, normalDrive, swingDrive, twistDrive, driveAngularVelocity, driveLinearVelocity, driveOrientation, drivePosition

axisMotion

Syntax

```
D6joint.axisMotion
```

Zugriff: Abrufen/Festlegen

Beschreibung

Symbol, das den linearen Freiheitsgrad entlang der Verbindungsachse festlegt.

Diese Eigenschaft kann folgende Werte aufweisen:

#Locked Es ist keine Bewegung entlang dieses Freiheitsgrads möglich.

#Limited Die Bewegung entlang dieses Freiheitsgrads ist eingeschränkt.

#Free Die Bewegung entlang dieses Freiheitsgrads ist uneingeschränkt möglich.

Beispiel

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
put objJoint.axisMotion
//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
Put(objJoint.axisMotion);
```

Siehe auch

biNormalMotion, normalMotion, swing1Motion, swing2Motion, twistMotion

biNormalDrive

Syntax

```
d6joint.binormalDrive = [type, spring, damping, forceLimit]
```

Zugriff: Abrufen/Festlegen

Beschreibung

Listenwert, welcher die lineare Bewegung der Verbindung angibt, welche die Verbindung entlang der Binormalachse der Verbindung zur festgelegten Position bringt.

Die Liste enthält die folgenden Attribute:

type #position oder #velocity

stiffness Die während der Bewegung anzulegende Federkraft (> 0)

damping die Dämpfung der Feder (< 0)

forceLimit Die Kraft bzw. das Drehmoment für das Erreichen der angegebenen Position oder Geschwindigkeit (> 0)

Beispiel

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
objJoint.binormalDrive = [#position,100,0.1,100]
//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
objJoint.binormalDrive = [#position,100,0.1,100];
```

Siehe auch

normalDrive, swingDrive, twistDrive, driveAngularVelocity, driveLinearVelocity, driveOrientation, drivePosition, axisDrive

biNormalMotion

Syntax

```
D6joint.binormalMotion
```

Zugriff: Abrufen/Festlegen

Beschreibung

Symbol, das den linearen Freiheitsgrad entlang der Binormalachse der Verbindung festlegt.

Diese Eigenschaft kann folgende Werte aufweisen:

#Locked Es ist keine Bewegung entlang dieses Freiheitsgrads möglich.

#Limited Die Bewegung entlang dieses Freiheitsgrads ist eingeschränkt.

#Free Die Bewegung entlang dieses Freiheitsgrads ist uneingeschränkt möglich.

Beispiel

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
put objJoint.biNormalMotion
//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
Put(objJoint.biNormalMotion);
```

Siehe auch

normalMotion, swing1Motion,swing2Motion, twistMotion, axisMotion

constraintType

Syntax

```
d6joint.constraintType
```

Zugriff: Abrufen

Beschreibung

D6-Verbindungen sind auch eine Art von Beschränkung wie lineare und Winkelverbindungen. Der Beschränkungstyp für eine D6-Verbindung lautet „#d6joint“.

Beispiel

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
put objJoint.constraintType
//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
put(objJoint.constraintType);
```

driveAngularVelocity

Syntax

```
d6joint.driveAngularVelocity = vector velocity
```

Zugriff: Abrufen/Festlegen

Beschreibung

Vektorwert, welcher die vorgesehene Winkelgeschwindigkeit angibt, wenn #velocity als Bewegungstyp für swingDrive oder twistDrive angegeben wird.

Beispiel

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
objJoint.driveAngularVelocity = vector(20,10,0)
//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
objJoint.driveAngularVelocity = vector(20,10,0);
```

Siehe auch

normalDrive, swingDrive, twistDrive, driveLinearVelocity, driveOrientation, drivePosition, axisDrive, biNormalDrive

driveLinearVelocity

Syntax

```
d6joint.driveLinearVelocity = vector velocity
```

Zugriff: Abrufen/Festlegen

Beschreibung
Vektorwert, welcher die vorgesehene lineare Geschwindigkeit angibt, wenn #velocity als Bewegungstyp für axisDrive, normalDrive oder binormalDrive angegeben wird.

Beispiel
```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
objJoint.driveLinearVelocity = vector(20,10,0)
//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
objJoint.driveLinearVelocity = vector(20,10,0);
```

Siehe auch
normalDrive, swingDrive, twistDrive, driveAngularVelocity, driveOrientation, drivePosition, axisDrive, biNormalDrive

driveOrientation

Syntax
```
d6joint.driveOrientation = list orientation
```

Zugriff: Abrufen/Festlegen

Beschreibung
Listenwert, welcher die Zielausrichtung angibt, wenn #position als Bewegungstyp für swingDrive oder twistDrive angegeben wird.

Beispiel
```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
objJoint.driveOrientation = [vector(20,10,0),45]
//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
objJoint.driveOrientation = [vector(20,10,0),45];
```

Siehe auch
normalDrive, swingDrive, twistDrive, driveLinearVelocity, driveAngularVelocity, drivePosition, axisDrive, biNormalDrive

drivePosition

Syntax
```
d6joint.drivePosition = vector position
```

Zugriff: Abrufen/Festlegen

Beschreibung
Vektorwert, welcher die Zielposition angibt, wenn #velocity als Bewegungstyp für axisDrive, normalDrive oder binormalDrive angegeben wird.

Beispiel

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
objJoint.drivePosition = vector(20,10,0)
//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
objJoint.drivePosition = vector(20,10,0);
```

Siehe auch

normalDrive, swingDrive,twistDrive, driveLinearVelocity, driveAngularVelocity, driveOrientation, axisDrive, biNormalDrive

globalAnchor

Syntax

```
d6joint.globalAnchor = vector position
```

Zugriff: Abrufen/Festlegen

Beschreibung

Vektorwert, welcher den Ankerpunkt für die Verbindung angibt.

Beispiel

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
objJoint.globalAnchor = vector(20,10,0)
//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
objJoint.globalAnchor = vector(20,10,0);
```

Siehe auch

localAnchorA, localAnchorB

linearLimit

Syntax

```
D6joint.linearlimit = [limitvalue, stiffness, damping, restitution]
```

Zugriff: Abrufen/Festlegen

Beschreibung

Liste, welche das Verbindungsverhalten bei eingeschränkter linearer Bewegung angibt. Bei Erreichen des Grenzwerts pendelt der Festkörper. Dies gilt für alle linearen Bewegungen.

Beispiel

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
objJoint.linearLimit =  [2, 100, 0.01, 0]
//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
objJoint.linearLimit = [2, 100, 0.01, 0];
```

Hinweis: *Legen Sie zuerst die Grenzwerte fest und stellen Sie dann die Bewegungsparameter [#free,#limited usw.] ein, damit die Grenzwerte in Kraft treten.*

Siehe auch

swing1Limit, swing2Limit, twistLimit

localAnchorA

Syntax

```
D6joint.localAnchorA
```

Zugriff: Abrufen/Festlegen

Beschreibung

Befestigungspunkt der Verbindung im Raum von Festkörper A.

Die Werte der Eigenschaften localAxisA, localAxisB, localNormalA, localNormalB, localAnchorA und localAnchorB stehen erst zur Verfügung, wenn Sie die Werte explizit für diese Eigenschaften festgelegt haben. Wenn Sie den Wert nicht festgelegen, wird ein Void-Vektor zurückgegeben.

Hinweis: *Um eine stabile Verbindung zu erstellen, geben Sie gleiche Werte für localAnchorA und localAnchorB an, damit beide Eigenschaften im Raum der 3D-Welt demselben Punkt entsprechen.*

Beispiel

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
objJoint.localAnchorA = vector(0,5,0)
//JavaScript Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint");
objJoint.localAnchorA = vector(0,5,0);
```

Siehe auch

localAnchorB, globalAnchor

localAnchorB

Syntax

```
D6joint.localAnchorB
```

Zugriff: Abrufen/Festlegen

Beschreibung

Befestigungspunkt der Verbindung im Raum von Festkörper B.

Die Werte der Eigenschaften localAxisA, localAxisB, localNormalA, localNormalB, localAnchorA und localAnchorB stehen erst zur Verfügung, wenn Sie die Werte explizit für diese Eigenschaften festgelegt haben. Wenn Sie den Wert nicht festgelegen, wird ein Void-Vektor zurückgegeben.

Beispiel

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
objJoint.localAnchorB = vector(0,-5,0)
//JavaScript Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint");
objJoint.localAnchorB = vector(0,-5,0);
```

Siehe auch

`localAnchorA`, `globalAnchor`

localAxisA

Syntax

```
D6joint.localAxisA
```

Zugriff: Abrufen/Festlegen

Beschreibung

Dies ist die Primärachse der Verbindung bezogen auf den lokalen Koordinatenraum von objectA der D6-Verbindung. localAxisA und localNormalA sollten im rechten Winkel zueinander liegen.

Bei einem statischen Festkörper gelten die für ihn festgelegten lokalen Achsen als global, da sie nicht entlang der Verbindungsachsen ausgerichtet werden können.

Die Werte der Eigenschaften `localAxisA`, `localAxisB`, `localNormalA`, `localNormalB`, `localAnchorA`, and`localAnchorB` stehen erst zur Verfügung, wenn Sie die Werte explizit für diese Eigenschaften festgelegt haben. Wenn Sie den Wert nicht festgelegen, wird ein Void-Vektor zurückgegeben.

- Die Primärachse der Verbindung ist die Achse, um welche die Drehung erfolgt und entlang derer die Bewegung AxisMotion verläuft.
- Die Normalachse der Verbindung ist die Achse, um welche der Schwenkvorgang swing1 erfolgt und entlang derer die Bewegung NormalMotion verläuft.
- Die Binormalachse der Verbindung ist die Achse, um welche der Schwenkvorgang swing2 erfolgt und entlang derer die Bewegung BinormalMotion verläuft.

Sie können die Verbindungsachsen jederzeit ändern, indem Sie die folgenden Eigenschaften entsprechend festlegen:

- localAxisA
- localNormalA
- localAxisB
- localNormalB

Beispiel

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
--If ObjectA's orientation is vector(0,0,0)
objJoint.localAxisA = vector(0,1,0)
objJoint.localNormalA = vector(1,0,0)
-- If the ObjectB's orientation is vector(90,0,0)
objJoint.localAxisB = vector(0,0,-1)
objJoint.localNormalB = vector(1,0,0)
//JavaScript Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
//If ObjectA's orientation is vector(0,0,0)
objJoint.localAxisA = vector(0,1,0);
objJoint.localNormalA = vector(1,0,0);
// If the ObjectB's orientation is vector(90,0,0)
objJoint.localAxisB = vector(0,0,-1);
objJoint.localNormalB = vector(1,0,0);
```

Siehe auch

localAxisB, localNormalA, localNormalB

localAxisB

Syntax

```
D6joint.localAxisB
```

Zugriff: Abrufen/Festlegen

Beschreibung

Dies ist die Primärachse der Verbindung bezogen auf den lokalen Koordinatenraum von objectB der D6-Verbindung. localAxisB und localNormalB sollten im rechten Winkel zueinander liegen. Dies ist die Achse, um welche die Drehung und entlang derer die Bewegung AxisMotion definiert sind.

Die Werte der Eigenschaften localAxisA, localAxisB, localNormalA, localNormalB, localAnchorA und localAnchorB stehen erst zur Verfügung, wenn Sie die Werte explizit für diese Eigenschaften festgelegt haben. Wenn Sie den Wert nicht festgelegen, wird ein Void-Vektor zurückgegeben.

Beispiel

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
--If ObjectA's orientation is vector(0,0,0)
objJoint.localAxisA = vector(0,1,0)
objJoint.localNormalA = vector(1,0,0)
-- If the ObjectB's orientation is vector(90,0,0)
objJoint.localAxisB = vector(0,0,-1)
objJoint.localNormalB = vector(1,0,0)
//JavaScript Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
//If ObjectA's orientation is vector(0,0,0)
objJoint.localAxisA = vector(0,1,0);
objJoint.localNormalA = vector(1,0,0);
// If the ObjectB's orientation is vector(90,0,0)
objJoint.localAxisB = vector(0,0,-1);
objJoint.localNormalB = vector(1,0,0);
```

Siehe auch

localAxisA, localNormalA, localNormalB

localNormalA

Syntax

```
D6joint.localNormalA
```

Zugriff: Abrufen/Festlegen

Beschreibung

Dies ist die Normalachse der Verbindung bezogen auf den lokalen Koordinatenraum von objectA der D6-Verbindung. localAxisA und localNormalA sollten im rechten Winkel zueinander liegen. Dies ist die Achse, um welche die Drehung „swing1“ und entlang derer die Bewegung NormalMotion definiert sind.

Bei einem statischen Festkörper gelten die für ihn festgelegten lokalen Achsen als global, da sie nicht entlang der Verbindungsachsen ausgerichtet werden können.

Die Werte der Eigenschaften localAxisA, localAxisB, localNormalA, localNormalB, localAnchorA und localAnchorB stehen erst zur Verfügung, wenn Sie die Werte explizit für diese Eigenschaften festgelegt haben. Wenn Sie den Wert nicht festgelegen, wird ein Void-Vektor zurückgegeben.

Beispiel

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
--If ObjectA's orientation is vector(0,0,0)
objJoint.localAxisA = vector(0,1,0)
objJoint.localNormalA = vector(1,0,0)
-- If the ObjectB's orientation is vector(90,0,0)
objJoint.localAxisB = vector(0,0,-1)
objJoint.localNormalB = vector(1,0,0)
//JavaScript Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
//If ObjectA's orientation is vector(0,0,0)
objJoint.localAxisA = vector(0,1,0);
objJoint.localNormalA = vector(1,0,0);
// If the ObjectB's orientation is vector(90,0,0)
objJoint.localAxisB = vector(0,0,-1);
objJoint.localNormalB = vector(1,0,0);
```

Siehe auch

localAxisB, localAxisA, localNormalB

localNormalB

Syntax

```
D6joint.localNormalB
```

Zugriff: Abrufen/Festlegen

Beschreibung

Dies ist die Normalachse der Verbindung bezogen auf den lokalen Koordinatenraum von objectB der D6-Verbindung. localAxisB und localNormalB sollten im rechten Winkel zueinander liegen. Dies ist die Achse, um welche die Drehung „swing1“ und entlang derer die Bewegung „NormalMotion“ definiert sind.

Die Werte der Eigenschaften localAxisA, localAxisB, localNormalA, localNormalB, localAnchorA und localAnchorB stehen erst zur Verfügung, wenn Sie die Werte explizit für diese Eigenschaften festgelegt haben. Wenn Sie den Wert nicht festgelegen, wird ein Void-Vektor zurückgegeben.

Beispiel

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
--If ObjectA's orientation is vector(0,0,0)
objJoint.localAxisA = vector(0,1,0)
objJoint.localNormalA = vector(1,0,0)
-- If the ObjectB's orientation is vector(90,0,0)
objJoint.localAxisB = vector(0,0,-1)
objJoint.localNormalB = vector(1,0,0)
//JavaScript Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
//If ObjectA's orientation is vector(0,0,0)
objJoint.localAxisA = vector(0,1,0);
objJoint.localNormalA = vector(1,0,0);
// If the ObjectB's orientation is vector(90,0,0)
objJoint.localAxisB = vector(0,0,-1);
objJoint.localNormalB = vector(1,0,0);
```

Siehe auch

localAxisA, localAxisB, localNormalA

name

Syntax

```
d6joint.name
```

Zugriff: Abrufen

Typ: String

Beschreibung

Gibt den Namen der 6-DOF-Verbindung zurück.

Beispiel

```
--Lingo Syntax
objConstraint = member("PhysicsWorld").getConstraint("d6joint")
put objConstraint.name
//Javascript Syntax
var objConstraint = member("PhysicsWorld").getConstraint("d6joint");
put(objConstraint.name);
```

normalDrive

Syntax

```
d6joint.normalDrive = [type, spring, damping, forceLimit]
```

Zugriff: Abrufen/Festlegen

Beschreibung

Listenwert, welcher die lineare Bewegung der Verbindung angibt, welche die Verbindung entlang der Normalachse der Verbindung zur festgelegten Position bringt.

Die Liste enthält die folgenden Attribute:

type #position oder #velocity

Hinweis: *Wenn Sie #position wählen, wird der Wert für forceLimit ignoriert und nur der Wert für spring berücksichtigt. Gleichermaßen gilt: Wenn Sie #velocity wählen, wird der Wert für spring ignoriert und nur der Wert für forceLimit berücksichtigt.*

stiffness Die während der Bewegung anzulegende Feder (> 0)

damping Die Dämpfung der Feder (> 0)

forceLimit Die Kraft bzw. das Drehmoment für das Erreichen der angegebenen Position oder Geschwindigkeit (> 0)

Beispiel

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
objJoint.normalDrive = [#position,100,0.1,100]
//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
objJoint.normalDrive = [#position,100,0.1,100];
```

Siehe auch

swingDrive, twistDrive, driveAngularVelocity, driveLinearVelocity, driveOrientation, drivePosition, axisDrive, biNormalDrive

normalMotion

Syntax

```
D6joint.normalMotion
```

Zugriff: Abrufen/Festlegen

Beschreibung

Symbol, das den linearen Freiheitsgrad entlang der Normalachse der Verbindung festlegt.

Diese Eigenschaft kann folgende Werte aufweisen:

#Locked Es ist keine Bewegung entlang dieses Freiheitsgrads möglich.

#Limited Die Bewegung entlang dieses Freiheitsgrads ist eingeschränkt.

#Free Die Bewegung entlang dieses Freiheitsgrads ist uneingeschränkt möglich.

Beispiel

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
put objJoint.normalMotion
//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
Put(objJoint.normalMotion);
```

Siehe auch

swing1Motion, swing2Motion, twistMotion, axisMotion, biNormalMotion

objectA

Syntax

```
d6joint.objectA
```

Zugriff: Abrufen

Typ: Festkörper

Beschreibung

Gibt den Namen des Festkörpers „A“ zurück, der an der Verbindung angebracht ist.

Beispiel

```
--Lingo Syntax
objJoint = member("PhysicsWorld").getConstraint("Joint01")
put objJoint.objectA
//JavaScript Syntax.
objJoint = member("PhysicsWorld").getConstraint("Joint01");
put(objJoint.objectA);
```

Siehe auch

objectB

objectB

Syntax

```
d6joint.objectB
```

Zugriff: Abrufen

Typ: Festkörper

Beschreibung

Gibt den Namen des Festkörpers „B“ zurück, der an der Feder angebracht ist.

Beispiel

```
--Lingo Syntax
objJoint = member("PhysicsWorld").getConstraint("Joint01")
put objJoint.objectB
//JavaScript Syntax.
objJoint = member("PhysicsWorld").getConstraint("Joint01");
put(objJoint.objectB);
```

Siehe auch

objectA

swing1Limit

Syntax

```
D6joint.swing1limit = [limitvalue, stiffness, damping, restitution]
```

Zugriff: Abrufen/Festlegen

Beschreibung

Liste, welche das Verbindungsverhalten bei eingeschränkter Winkelbewegung um die Normalachse der Verbindung angibt. Bei Erreichen des Grenzwerts pendelt der Festkörper.

Der mögliche Wertebereich für den Grenzwert liegt zwischen 3,14 und -3,14 (pi).

Beispiel

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
objJoint.swing1limit =  [3.14*0.5, 100, 0.01, 0]
//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
objJoint.swing1limit = [3.14*0.5, 100, 0.01, 0];
```

Hinweis: *Legen Sie zuerst die Grenzwerte fest und stellen Sie dann die Bewegungsparameter [#free,#limited usw.] ein, damit die Grenzwerte in Kraft treten.*

Siehe auch

swing2Limit,`twistLimit`, linearLimit

swing1Motion

Syntax

```
D6joint.swing1Motion
```

Zugriff: Abrufen/Festlegen

Beschreibung

Symbol, das den Winkelfreiheitsgrad um die Normalachse der Verbindung festlegt.

Diese Eigenschaft kann folgende Werte aufweisen:

#Locked Es ist keine Bewegung entlang dieses Freiheitsgrads möglich.

#Limited Die Bewegung entlang dieses Freiheitsgrads ist eingeschränkt.

#Free Die Bewegung entlang dieses Freiheitsgrads ist uneingeschränkt möglich.

Beispiel

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
put objJoint.swing1Motion
//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
Put(objJoint.swing1Motion);
```

Siehe auch

normalMotion, swing2Motion, twistMotion, axisMotion, biNormalMotion

swing2Limit

Syntax

```
D6joint.swing2limit = [limitvalue, stiffness, damping, restitution]
```

Zugriff: Abrufen/Festlegen

Beschreibung

Liste, welche das Verbindungsverhalten bei eingeschränkter Winkelbewegung um die Binormalachse der Verbindung angibt. Bei Erreichen des Grenzwerts pendelt der Festkörper.

Der mögliche Wertebereich für den Grenzwert liegt zwischen 3,14 und -3,14 (pi).

Beispiel

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
objJoint.swing2limit =  [3.14*0.5, 100, 0.01, 0]
//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
objJoint.swing2limit = [3.14*0.5, 100, 0.01, 0];
```

Hinweis: *Legen Sie zuerst die Grenzwerte fest und stellen Sie dann die Bewegungsparameter [#free,#limited usw.] ein, damit die Grenzwerte in Kraft treten.*

Siehe auch

swing1Limit, twistLimit, linearLimit

swing2Motion

Syntax

```
D6joint.swing2Motion
```

Zugriff: Abrufen/Festlegen

Beschreibung

Symbol, das den Winkelfreiheitsgrad um die Binormalachse der Verbindung festlegt.

Diese Eigenschaft kann folgende Werte aufweisen:

#Locked Es ist keine Bewegung entlang dieses Freiheitsgrads möglich.

#Limited Die Bewegung entlang dieses Freiheitsgrads ist eingeschränkt.

#Free Die Bewegung entlang dieses Freiheitsgrads ist uneingeschränkt möglich.

Beispiel

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
put objJoint.swing2Motion
//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
Put(objJoint.swing2Motion);
```

Siehe auch

normalMotion, swing1Motion, twistMotion, axisMotion, biNormalMotion

swingDrive

Syntax

```
d6joint.swingDrive = [type, spring, damping, forceLimit]
```

Zugriff: Abrufen/Festlegen

Beschreibung

Listenwert, welcher die Winkelbewegung der Verbindung angibt, welche die Verbindung um die Normal- und Binormalachsen der Verbindung in die festgelegte Ausrichtung bringt.

Die Liste enthält die folgenden Attribute:

type #position oder #velocity

stiffness Die während der Bewegung anzulegende Feder (> 0)

damping Die Dämpfung der Feder (> 0)

forceLimit Die Kraft bzw. das Drehmoment für das Erreichen der angegebenen Position oder Geschwindigkeit (> 0)

Beispiel

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
objJoint.swingDrive = [#position,100,0.1,100]
//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
objJoint.swingDrive = [#position,100,0.1,100];
```

Siehe auch

normalDrive, twistDrive, driveAngularVelocity, driveLinearVelocity, driveOrientation, drivePosition, axisDrive, biNormalDrive

twistDrive

Syntax

```
d6joint.twistDrive = [type, spring, damping, forceLimit]
```

Zugriff: Abrufen/Festlegen

Beschreibung

Listenwert, welcher die Winkelbewegung der Verbindung angibt, welche die Verbindung um die Verbindungsachse in die festgelegte Ausrichtung bringt.

Die Liste enthält die folgenden Attribute:

type #position oder #velocity

stiffness Die während der Bewegung anzulegende Feder (> 0)

damping Die Dämpfung der Feder (> 0)

forceLimit Die Kraft bzw. das Drehmoment für das Erreichen der angegebenen Position oder Geschwindigkeit (> 0)

Beispiel

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
objJoint.twistDrive = [#position,100,0.1,100]
//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
objJoint.twistDrive = [#position,100,0.1,100];
```

Siehe auch

normalDrive, swingDrive, driveAngularVelocity, driveLinearVelocity, driveOrientation, drivePosition, axisDrive, biNormalDrive

twistLimit

Syntax

```
D6joint.twistLimit = [limitvalue, stiffness, damping, restitution]
```

Zugriff: Abrufen/Festlegen

Beschreibung

Liste, welche das Verbindungsverhalten bei eingeschränkter Winkelbewegung entlang der Verbindungsachse angibt. Bei Erreichen des Grenzwerts pendelt der Festkörper.

Der mögliche Wertebereich für den Grenzwert liegt zwischen 3,14 und -3,14 (pi).

Beispiel

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
objJoint.twistLimit =  [3.14*0.5, 100, 0.01, 0]
//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
objJoint.twistLimit = [3.14*0.5, 100, 0.01, 0];
```

Hinweis: *Legen Sie zuerst die Grenzwerte fest und stellen Sie dann die Bewegungsparameter [#free,#limited usw.] ein, damit die Grenzwerte in Kraft treten.*

Siehe auch

swing2Limit, swing1Limit, linearLimit

twistMotion

Syntax

```
D6joint.twistMotion
```

Zugriff: Abrufen/Festlegen

Beschreibung

Symbol, das den Winkelfreiheitsgrad um die Primärachse der Verbindung festlegt.

Diese Eigenschaft kann folgende Werte aufweisen:

#Locked Es ist keine Bewegung entlang dieses Freiheitsgrads möglich.

#Limited Die Bewegung entlang dieses Freiheitsgrads ist eingeschränkt.

#Free Die Bewegung entlang dieses Freiheitsgrads ist uneingeschränkt möglich.

Beispiel

```
--Lingo Syntax
objJoint = Member("PhysicsWorld").getConstraint("D6Joint")
put objJoint.twistMotion
//JavaScript Syntax
var objJoint = Member("PhysicsWorld").getConstraint ("D6Joint");
Put(objJoint.twistMotion);
```

Siehe auch

normalMotion, swing2Motion, swing1Motion, axisMotion, biNormalMotion

RayCasting-Methoden

rayCastAll

Syntax

```
<list <list>> world.rayCastAll(vector origin, vector direction, sorted list)
```

Beschreibung

Diese Methode gibt die Referenzen aller Festkörper oder Gelände zurück, die entlang des vom festgelegten Ursprung ausgehenden Strahls in der angegebenen Richtung gefunden werden. Zudem gibt die Methode den Kontaktpunkt, die Kontaktnormale und den Abstand vom Ursprung des Strahls zurück.

Parameter

Parameter	Beschreibung
origin	Erforderlich. Vektor, welcher den Ursprung des Raycastings angibt.
direction	Erforderlich. Vektor, welcher die Richtung des Raycastings angibt.
Sortiert	Optional. Gibt eine nach Abstand sortierte Liste mit Starrkörpern zurück.

Diese Methode gibt eine Liste zurück, die eine Liste mit folgenden Informationen enthält:

- Festkörper-/Geländereferenz
- Kontaktpunkt
- Kontaktnormale
- Abstand von Festkörper oder Gelände zum Ursprung des Strahls

Beispiel

```
--Lingo Syntax
lstraycast = member("PhysicsWorld").rayCastAll (vector(10,0,0),vector(0,0,1), [#sorted:#distance])
repeat with i = 1 to lstraycast.count
    raycstEntry = lstraycast[i]
    put "Name:" & raycstEntry[1].name
    put "Contact Point:" & raycstEntry[2]
    put "Contact Normal:" & raycstEntry[3]
    put "Distance:" & raycstEntry[4]
end repeat
//JavaScript Syntax
var lstraycast = member("PhysicsWorld").rayCastAll (vector(10,0,0),vector(0,0,1),
propList(symbol("sorted"), symbol("distance"));
for(i = 1; i <= lstraycast.count ; i++)
{
    raycstEntry = lstraycast[i];
    put("Name:" + raycstEntry[1].name);
    put"Contact Point:" & raycstEntry[2]);
    put("Contact Normal:" & raycstEntry[3]);
    put("Distance:" & raycstEntry[4]);
}
```

Siehe auch

rayCastClosest

rayCastClosest

Syntax

```
<list> world.rayCastClosest(vector origin, vector direction)
```

Beschreibung

Diese Methode gibt die Referenz des am nächsten liegenden Festkörpers oder Geländes zurück, das entlang des vom festgelegten Ursprung ausgehenden Strahls in der angegebenen Richtung gefunden wird. Zudem gibt die Methode den Kontaktpunkt, die Kontaktnormale und den Abstand vom Ursprung des Strahls zurück.

Parameter

Parameter	Beschreibung
origin	Erforderlich. Vektor, welcher den Ursprung des Raycastings angibt.
direction	Erforderlich. Vektor, welcher die Richtung des Raycastings angibt.

Diese Methode gibt eine Liste zurück, die folgende Informationen enthält:

- Festkörper-/Geländereferenz
- Kontaktpunkt
- Kontaktnormale
- Abstand von Festkörper oder Gelände zum Ursprung des Strahls

Beispiel

```
--Lingo Syntax
lstraycast = member("PhysicsWorld").rayCastClosest(vector(10,0,0),vector(0,0,1))
put "Name:"  & lstraycast[1].name
put "Contact Point:"  & lstraycast[2]
put "Contact Normal:"  & lstraycast[3]
put "Distance:"  & lstraycast[4]
//JavaScript Syntax
var lstraycast = member("PhysicsWorld").rayCastClosest(vector(10,0,0),vector(0,0,1));
put("Name:"  + lstraycast[0].name);
put"Contact Point:"  & lstraycast[1]);
put("Contact Normal:"  & lstraycast[2]);
put("Distance:"  & lstraycast[3]);
```

Siehe auch

rayCastAll

Fehlercodes

In der nachfolgenden Tabelle werden die folgenden Fehlercodes und ihre Ursachen beschrieben.

Fehlercode	Ursache des Fehlers
-1	Unbekannter Fehler Beim Erstellen eines Festkörpers wird eventuell den Fehler „-1“ gemeldet, wenn die Kamera und das Modell weit voneinander entfernt sind. Dies liegt daran, dass die Gitternetzanzahl aufgrund der Optimierung Null wird.
-2	Welt nicht initialisiert
-3	simulate-Funktion schlug fehl
-4	Ungültiger Parameter
-5	Doppelte Beschränkung
-6	Ungültige Beschränkung/Beschränkung ist nicht vorhanden.
-8	Ungültiger Festkörper/Festkörper ist nicht vorhanden.
-11	Element nicht gefunden. Es wird ein Physikobjekt verwendet, das gelöscht wurde.
-18	Ungültiges 3D-Modell
-19	3D-Fehler
-20	Nicht genügend Speicher
-21	Gitternetzdeformation wurde für das Modell nicht hinzugefügt.

Fehlercode	**Ursache des Fehlers**
-28	Tritt auf, wenn ein Festkörper mit dem Proxy „#concaveshape“ als „#dynamic“ festgelegt wird
-29	Duplikat der Modellressource.
-30	Wenn die Proxy-Vorlage ungültig ist.
-31	Wenn die Modellressource ungültig ist.

Index

B

C

D

E

G

M

N

O

P

S

T

U