ADOBE® DREAMWEAVER® CS5 & CS5.5
API-Referenzhandbuch

Rechtliche Hinweise

Letzte Aktualisierung 22.6.2011

Inhalt

Kapitel 1: Einführung

Im *Adobe Dreamweaver CS5 API-Referenzhandbuch* werden die Anwendungsprogrammierschnittstellen (APIs) beschrieben. Mit APIs können Sie während der Entwicklung von Adobe® Dreamweaver® CS5-Erweiterungen und beim Hinzufügen von Programmcode zu Dreamweaver-Webseiten verschiedene unterstützende Aufgaben ausführen. Eine der wichtigsten APIs ist die JavaScript-API, die Zugriff auf die meisten Kernfunktionen von Dreamweaver bietet. Unter die Kernfunktionen von Dreamweaver fallen generell alle menügesteuerten Aktionen. Hinzu kommen verschiedene Dienstprogramm-APIs für gängige Aufgaben wie das Schreiben und Lesen von Dateien, die Datenübertragung mit HTTP und die Kommunikation mit Fireworks und Flash.

Mit der umfangreichen JavaScript-API können Sie vielfältige kleinere Aufgaben erledigen. Dabei handelt es sich in der Mehrzahl um Arbeitsschritte, die ein Benutzer i. d. R. beim Erstellen oder Bearbeiten von Dreamweaver-Dokumenten ausführt. Diese API-Funktionen sind nach den betreffenden Komponenten der Dreamweaver-Benutzeroberfläche gruppiert. So umfasst die JavaScript-API beispielsweise Arbeitsbereichfunktionen, Dokumentfunktionen, Designfunktionen usw. Mit den API-Funktionen können Sie einige der folgenden Aufgaben und vieles mehr ausführen:

- Öffnen neuer Dokumente
- Abrufen oder Festlegen einer Schriftgröße
- Suchen nach einem Suchstring im HTML-Code
- Sichtbarmachen von Symbolleisten

Erweiterungen

In diesem Buch wird davon ausgegangen, dass Sie Kenntnisse in Dreamweaver, HTML, XML, der JavaScript-Programmierung und eventuell der Programmierung in C haben. Wenn Sie Erweiterungen zum Erstellen von Webanwendungen programmieren, sollten Sie auch mit serverbasierten Skripts auf mindestens einer Plattform vertraut sein, z. B. Active Server Pages (ASP), ASP.NET, PHP: Hypertext Preprocessor (PHP), Adobe ColdFusion oder Java Server Pages (JSP).

Erweitern von Dreamweaver

Weitere Informationen über das Dreamweaver-Framework und die API zur Erstellung von Dreamweaver-Erweiterungen finden Sie im Handbuch *Dreamweaver erweitern*. Im Handbuch *Dreamweaver erweitern* werden die API-Funktionen beschrieben, die Dreamweaver zur Implementierung von Objekten, Menüs, schwebenden Bedienfeldern, Serververhalten usw. aufruft, welche die verschiedenen Funktionen von Dreamweaver umfassen. Mithilfe dieser APIs können Sie dem Produkt Objekte, Menüs, schwebende Bedienfelder oder andere Funktionen hinzufügen. Im Handbuch *Dreamweaver erweitern* wird darüber hinaus erläutert, wie Sie Dreamweaver an Ihre spezifischen Anforderungen anpassen. Durch das Bearbeiten und Hinzufügen von Tags zu verschiedenen HTML- und XML-Dateien können Sie beispielsweise Menüelemente oder Dokumenttypen hinzufügen.

Weitere Ressourcen für das Programmieren von Erweiterungen

Wenn Sie sich mit anderen Entwicklern austauschen möchten, die ebenfalls Erweiterungen erstellen, können Sie der Dreamweaver Extensibility Newsgroup beitreten. Die Website für diese Newsgroup befindet sich unter http://www.adobe.com/cfusion/webforums/forum/categories.cfm?forumid=12&catid=190.

Neue Funktionen in Dreamweaver CS5

Die folgenden neuen Funktionen wurden der JavaScript-API von Dreamweaver CS5 hinzugefügt. Die Überschriften beziehen sich auf die Kapitel und Abschnitte, die die neuen Funktionen enthalten.

Dynamische Dokumente

Die folgenden Funktionen wurden zum Kapitel „Dynamische Dokumente“ hinzugefügt.

Live-Ansichtsfunktionen

„dom.setLiveViewFollowsLinks()“ auf Seite 371

„dom.getLiveViewFollowsLinks()“ auf Seite 372

„dom.isLiveViewBrowsingHomeURI()“ auf Seite 372

„dreamweaver.findSiteForURI()“ auf Seite 373

„dom.browser.isPageNavigationHistoryEnabled()“ auf Seite 374

„dom.browser.enablePageNavigationHistory()“ auf Seite 375

„dom.browser.getPageNavigationHistoryLength()“ auf Seite 375

„dom.browser.getPageNavigationHistoryPosition()“ auf Seite 375

„dom.browser.goToPageNavigationHistoryPosition()“ auf Seite 376

„dom.browser.getPageNavigationHistoryItem()“ auf Seite 376

„dom.browser.setHomePage()“ auf Seite 377

„dom.browser.getHomePage()“ auf Seite 377

Arbeitsbereich

Die folgenden neuen Funktionen wurden zum Kapitel „Arbeitsbereich“ hinzugefügt.

Funktionen für zugehörige Dateien

„dreamweaver.getRelatedFilesFilter()“ auf Seite 198

„dreamweaver.setRelatedFilesFilter()“ auf Seite 198

„dreamweaver.getQuickRelatedFilesFilterStrings()“ auf Seite 198

„dreamweaver.invokeRelatedFilesCustomFilterDialog()“ auf Seite 199

„dreamweaver.getDynamicRelatedFilesDiscoverySetting()“ auf Seite 199

„dreamweaver.setDynamicRelatedFilesDiscoverySetting()“ auf Seite 200

„dreamweaver.refreshRelatedFiles()“ auf Seite 200

„dreamweaver.saveAllRelatedFiles()“ auf Seite 201

„dreamweaver.canSaveAllRelatedFiles()“ auf Seite 201

„document.isRelatedFileViewOpen()“ auf Seite 201

„document.getRelatedFiles()“ auf Seite 202

„document.addRelatedFile()“ auf Seite 203

„document.removeRelatedFile()“ auf Seite 203

„document.getDependentFiles()“ auf Seite 204

... und weitere.

Dokument

Die folgenden neuen Funktionen wurden zum Kapitel „Dokument“ hinzugefügt.

„DWUri.isValidURI()“ auf Seite 286

„DWUri.isAbsolute()“ auf Seite 287

„DWUri.isRelative()“ auf Seite 287

„DWUri.isDirectory()“ auf Seite 287

„DWUri.isHierarchical()“ auf Seite 288

„DWUri.isOfType()“ auf Seite 288

„DWUri.isOfFileType()“ auf Seite 288

... und weitere.

Code

Die folgenden neuen Funktionen wurden zum Kapitel „Code“ hinzugefügt.

„dom.getLiveCodeHighlightsChanges()“ auf Seite 500

„dom.setLiveCodeHighlightsChanges()“ auf Seite 500

... und weitere.

In diesem Handbuch verwendete Konventionen

Typografische Konventionen

In diesem Handbuch werden die folgenden typografischen Konventionen verwendet:

- `Codeschrift` kennzeichnet Codefragmente und API-Literale, z. B. Klassennamen, Methodennamen, Funktionsnamen, Typnamen, Skripts, SQL-Anweisungen, HTML- und XML-Tag-Namen sowie Attributnamen.
- *`Kursive Codeschrift`* kennzeichnet Platzhalterelemente im Code.

- Das Fortsetzungssymbol (¬) weist darauf hin, dass ein langer Code über mehrere Zeilen umbrochen wurde. Da die Zeilenlänge in diesem Handbuch durch die im Format festgelegten Ränder begrenzt ist, muss Code, der eigentlich fortlaufend ist, auf mehrere Zeilen verteilt werden. Löschen Sie beim Kopieren der Codezeilen das Fortsetzungssymbol und geben Sie die Zeilen ohne Umbruch ein.
- Geschweifte Klammern ({ }), die ein Argument einschließen, weisen darauf hin, dass es sich um ein optionales Argument handelt.
- Funktionsnamen mit dem Präfix `dreamweaver.`*`Funktionsname`* können beim Programmieren von Code als `dw.`*`Funktionsname`* abgekürzt werden. In diesem Handbuch wird das ausführliche Präfix `dreamweaver.` bei der Definition der Funktion und im Index verwendet. In vielen Beispielen wird jedoch das kürzere Präfix `dw.` verwendet.

Namengebungskonventionen

Folgende Namengebungskonventionen werden in diesem Handbuch verwendet:

- Sie – die Person, die für das Programmieren von Erweiterungen verantwortlich ist (also der Entwickler).
- Der Benutzer – die Person, die Dreamweaver verwendet.

Kapitel 2: API für Datei-E/A

Adobe® Dreamweaver® CS5 enthält eine gemeinsam genutzte C-Bibliothek mit dem Namen „DWfile“. Mithilfe von DWfile können Autoren von Objekten, Befehlen, Verhalten, Datenübersetzern, schwebenden Bedienfeldern und Eigenschafteninspektoren im lokalen Dateisystem Dateien lesen und schreiben. In diesem Kapitel werden die API für Datei-E/A und deren Verwendung erläutert.

Allgemeine Informationen über das Zusammenwirken von C-Bibliotheken mit dem JavaScript-Interpreter in Dreamweaver finden Sie in *Dreamweaver erweitern* unter „C-Level-Erweiterbarkeit“.

Konfigurationsordner

Unter Microsoft Windows 2000, Windows XP und Mac OS X verfügen die Benutzer über eigene Kopien der Konfigurationsdateien. Wenn Dreamweaver in eine Konfigurationsdatei schreibt, übernimmt Dreamweaver diese Änderung in den Ordner „Configuration“ des Benutzers. Auch beim Lesen einer Konfigurationsdatei sucht Dreamweaver zunächst im Ordner „Configuration“ des Benutzers und erst dann im Ordner „Configuration“ von Dreamweaver. DWfile-Funktionen arbeiten nach dem gleichen Prinzip. Wenn also Ihre Erweiterung eine Datei im Dreamweaver-Ordner „Configuration“ schreibt oder liest, greift sie dabei auch auf den Ordner „Configuration“ des Benutzers zu. Weitere Informationen zu Konfigurationsordnern in Mehrbenutzersystemen finden Sie im Handbuch *Dreamweaver erweitern*.

Funktionen der API für Datei-E/A

Alle Funktionen der API für Datei-E/A sind Methoden des Objekts `DWfile`.

DWfile.copy()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion kopiert die angegebene Datei an einen neuen Ort.

Argumente
originalURL, copyURL

- Das Argument *originalURL* ist die Datei, die Sie kopieren möchten, im URL-Format „file://“.
- Das Argument *copyURL* ist der Speicherort, an dem Sie die kopierte Datei speichern möchten, im URL-Format „file://“.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der Kopiervorgang erfolgreich ist, andernfalls `false`.

Beispiel
Der folgende Code kopiert die Datei „myconfig.cfg“ nach „myconfig_backup.cfg“:

```
var fileURL = "file:///c|/Config/myconfig.cfg";
var newURL ="file:///c|/Config/myconfig_backup.cfg";
DWfile.copy(fileURL, newURL);
```

DWfile.createFolder()

Verfügbarkeit
Dreamweaver 2.

Beschreibung
Diese Funktion erstellt einen Ordner am angegebenen Ort.

Argumente
folderURL

- Das Argument *folderURL* ist der Speicherort des Ordners, den Sie erstellen möchten, im URL-Format „file://“.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der Ordner erfolgreich erstellt wurde, andernfalls `false`.

Beispiel
Der folgende Code erstellt den Ordner „tempFolder“ im Stammverzeichnis von Laufwerk C: und teilt in einem Dialogfeld mit, ob der Vorgang erfolgreich ausgeführt wurde:

```
var folderURL = "file:///c|/tempFolder";
if (DWfile.createFolder(folderURL)){
    alert("Created " + folderURL);
}else{
    alert("Unable to create " + folderURL);
}
```

DWfile.exists()

Verfügbarkeit
Dreamweaver 2.

Beschreibung
Diese Funktion prüft das Vorhandensein einer bestimmten Datei.

Argumente
fileURL

- Das Argument *fileURL* ist die angeforderte Datei im URL-Format „file://“.

Rückgabewerte
Ein boolescher Wert: `true`, wenn die Datei vorhanden ist, andernfalls `false`.

Beispiel
Der folgende Code sucht die Datei „mydata.txt“ und teilt in einer Meldung mit, ob sie vorhanden ist:

```
var fileURL = "file:///c|/temp/mydata.txt";
if (DWfile.exists(fileURL)){
    alert(fileURL + " exists!");
}else{
    alert(fileURL + " does not exist.");
}
```

DWfile.getAttributes()

Verfügbarkeit
Dreamweaver 2.

Beschreibung
Diese Funktion ruft die Attribute der angegebenen Datei bzw. des angegebenen Ordners ab.

Argumente
fileURL

- Das Argument *fileURL* ist die Datei oder der Ordner, für die bzw. den Sie Attribute abrufen möchten, im URL-Format „file://“.

Rückgabewerte
Ein String, der die Attribute der angegebenen Datei bzw. des angegebenen Ordners darstellt. Existiert die Datei oder der Ordner nicht, gibt die Funktion den Wert `null` zurück. Die Attribute werden durch folgende Zeichen im String dargestellt:

- `R` (schreibgeschützt)
- `D` (Ordner)
- `H` (verborgen)
- `S` (Systemdatei oder Ordner)

Beispiel
Der folgende Code fragt die Attribute der Datei „mydata.txt“ ab und zeigt eine Warnmeldung an, falls die Datei schreibgeschützt ist:

```
var fileURL = "file:///c|/temp/mydata.txt";
var str = DWfile.getAttributes(fileURL);
if (str && (str.indexOf("R") != -1)){
    alert(fileURL + " is read only!");
}
```

DWfile.getModificationDate()

Verfügbarkeit
Dreamweaver 2.

Beschreibung
Diese Funktion fragt den Zeitpunkt der letzten Änderung einer Datei ab.

Argumente
fileURL

- Das Argument *fileURL* ist die Datei, deren Zeitpunkt der letzten Änderung Sie überprüfen, im URL-Format „file://“.

Rückgabewerte
Ein String mit einer hexadezimalen Zahl, die die Anzahl der seit dem Ausgangszeitpunkt verstrichenen Zeiteinheiten angibt. Die jeweilige Bedeutung der Zeiteinheiten und des Ausgangszeitpunkts hängt von der verwendeten Plattform ab. So beträgt in Windows eine Zeiteinheit 100 ns und als Ausgangszeitpunkt wird der 1. Januar 1600 verwendet.

Beispiel
Sie sollten die Funktion zweimal aufrufen und die Rückgabewerte vergleichen, da der Rückgabewert dieser Funktion plattformabhängig ist und keine direkt lesbaren Datums- und Uhrzeitwerte darstellt. Der folgende Code fragt das Änderungsdatum von „file1.txt“ und „file2.txt“ ab und zeigt eine Meldung an, in der angegeben wird, welche der beiden Dateien aktueller ist:

```
var file1 = "file:///c|/temp/file1.txt";
var file2 = "file:///c|/temp/file2.txt";
var time1 = DWfile.getModificationDate(file1);
var time2 = DWfile.getModificationDate(file2);
if (time1 == time2){
    alert("file1 and file2 were saved at the same time");
}else if (time1 < time2){
    alert("file1 older that file2");
}else{
    alert("file1 is newer than file2");
}
```

DWfile.getCreationDate()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Diese Funktion fragt den Zeitpunkt ab, zu der die Datei erstellt wurde.

Argumente
fileURL

- Das Argument *fileURL* ist die Datei, deren Zeitpunkt der Erstellung Sie überprüfen, im URL-Format „file://“.

Rückgabewerte
Ein String mit einer hexadezimalen Zahl, die die Anzahl der seit dem Ausgangszeitpunkt verstrichenen Zeiteinheiten angibt. Die jeweilige Bedeutung der Zeiteinheiten und des Ausgangszeitpunkts hängt von der verwendeten Plattform ab. So beträgt in Windows eine Zeiteinheit 100 ns und als Ausgangszeitpunkt wird der 1. Januar 1600 verwendet.

Beispiel
Sie können diese Funktion und die Funktion `DWfile.getModificationDate()` aufrufen, um das Änderungsdatum einer Datei mit ihrem Erstelldatum zu vergleichen:

```
var file1 = "file:///c|/temp/file1.txt";
var time1 = DWfile.getCreationDate(file1);
var time2 = DWfile.getModificationDate(file1);
if (time1 == time2){
    alert("file1 has not been modified since it was created");
}else if (time1 < time2){
    alert("file1 was last modified on " + time2);
}
```

DWfile.getCreationDateObj()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Diese Funktion ruft das JavaScript-Objekt für den Zeitpunkt ab, zu dem die Datei erstellt wurde.

Argumente
fileURL

- Das Argument *fileURL* ist die Datei, deren Zeitpunkt der Erstellung Sie überprüfen, im URL-Format „file://“.

Rückgabewerte
Ein JavaScript `Date`-Objekt für Datum und Uhrzeit der Erstellung einer Datei.

DWfile.getModificationDateObj()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Diese Funktion ruft das JavaScript `Date`-Objekt für den Zeitpunkt der letzten Änderung an einer Datei ab.

Argumente
fileURL

- Das Argument *fileURL* ist die Datei, deren Zeitpunkt der letzten Änderung Sie überprüfen, im URL-Format „file://“.

Rückgabewerte
Ein JavaScript `Date`-Objekt für Datum und Uhrzeit der letzten Änderung an einer Datei.

DWfile.getSize()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Diese Funktion ruft die Größe einer Datei ab.

Argumente
fileURL

- Das Argument *fileURL* ist die Datei, deren Größe Sie überprüfen, im URL-Format „file://“.

Rückgabewerte
Eine Ganzzahl für die tatsächliche Größe (in Byte) einer Datei.

DWfile.listFolder()

Verfügbarkeit
Dreamweaver 2.

Beschreibung
Diese Funktion erstellt eine Liste mit dem Inhalt des angegebenen Ordners.

Argumente
folderURL, {constraint}

- Das Argument *folderURL* ist der Ordner, für den Sie eine Inhaltsliste erstellen möchten, im URL-Format „file://“ gefolgt von einem optionalen Dateimasken-Platzhalter. Gültige Platzhalter sind Sternchen (*) für ein oder mehrere Zeichen und Fragezeichen (?) für ein Zeichen.
- Das Argument *constraint*, falls angegeben, muss entweder `"files"` (nur Dateien zurückgeben) oder `"directories"` (nur Ordner zurückgeben) lauten. Falls das Argument nicht angegeben wird, werden sowohl Dateien als auch Ordner berücksichtigt.

Rückgabewerte
Ein String-Array mit den einzelnen Elementen des Ordners.

Beispiel
Der folgende Code erstellt eine Liste aller Textdateien (TXT) im Ordner „C:/temp“. Die Liste wird in einer Meldung angezeigt:

```
var folderURL = "file:///c|/temp";
var fileMask = "*.txt";
var list = DWfile.listFolder(folderURL + "/" + fileMask, "files");
if (list){
    alert(folderURL + " contains: " + list.join("\n"));
}
```

DWfile.read()

Verfügbarkeit
Dreamweaver 2.

Beschreibung
Diese Funktion liest den Inhalt der angegebenen Datei und schreibt ihn in einen String.

Argumente
fileURL

- Das Argument *fileURL* ist die Datei, die Sie lesen möchten, im URL-Format „file://“.

Rückgabewerte
Ein String, in dem sich der Inhalt der Datei befindet, oder der Wert `null`, wenn der Lesevorgang fehlgeschlagen ist.

Beispiel
Der folgende Code liest die Datei „mydata.txt“ und zeigt bei einem erfolgreichen Lesevorgang eine Meldung mit dem Inhalt der Datei an:

```
var fileURL = "file:///c|/temp/mydata.txt";
var str = DWfile.read(fileURL);
if (str){
    alert(fileURL + " contains: " + str);
}
```

DWfile.remove()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion löscht die angegebene Datei.

Argumente
fileURL

- Das Argument *fileURL* ist die Datei, die Sie entfernen möchten, im URL-Format „file://“.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der Vorgang erfolgreich ist, andernfalls `false`.

Beispiel
Im folgenden Beispiel wird mit `DWfile.getAttributes()` ermittelt, ob eine Datei schreibgeschützt ist, und mit `confirm()` ein Ja/Nein-Dialogfeld in der Benutzeroberfläche angezeigt:

```
function deleteFile(){
    var delAnyway = false;
    var selIndex = document.theForm.menu.selectedIndex;
    var selFile = document.theForm.menu.options[selIndex].value;
    if (DWfile.getAttributes(selFile).indexOf('R') != -1){
        delAnyway = confirm('This file is read-only. Delete anyway?');
        if (delAnyway){
            DWfile.remove(selFile);
        }
    }
}
```

DWfile.setAttributes()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Diese Funktion legt die Attribute auf Systemebene für eine bestimmte Datei fest.

Argumente
fileURL, strAttrs

- Das Argument *fileURL* identifiziert die Datei, für die Sie die Attribute festlegen, im URL-Format „file://“.
- Das Argument *strAttrs* legt die Attribute auf Systemebene für die von *fileURL* bezeichnete Datei fest. In der folgenden Tabelle sind gültige Attributwerte und ihre Bedeutungen aufgeführt:

Attributwert	Beschreibung
`R`	Schreibgeschützt
`W`	Kein Schreibschutz (überschreibt `R`)
`H`	Verborgen
`V`	Sichtbar (überschreibt `H`)

Gültige Werte für den `strAttrs`-String sind `R`, `W`, `H`, `V`, `RH`, `RV`, `WH` und `WV`.

`R` und `W` schließen sich gegenseitig aus und sollten daher nicht gemeinsam verwendet werden. Werden sie in der Kombination eingesetzt, wird `R` außer Kraft gesetzt und der Schreibschutz für die Datei wird aufgehoben (`W`). Auch `H` und `V` schließen sich gegenseitig aus. Wenn Sie sie kombinieren, wird `H` außer Kraft gesetzt und die Datei wird auf sichtbar (`V`) gesetzt.

Legen Sie `H` oder `V` ohne das Lese-/Schreibattribut `R` oder `W` fest, bleibt das aktuelle Lese-/Schreibattribut der Datei unverändert. Wenn Sie `R` oder `W` festlegen, ohne das Attribut `H` oder `V` anzugeben, wird das aktuelle Sichtbarkeitsattribut der Datei nicht geändert.

Rückgabewerte
Keine.

DWfile.write()

Verfügbarkeit
Dreamweaver 2.

Beschreibung
Diese Funktion schreibt den festgelegten String in die angegebene Datei. Wenn die angegebene Datei noch nicht vorhanden ist, wird sie erstellt.

Argumente
`fileURL`, `text`, `{mode}`

- Das Argument *`fileURL`* ist die Datei, in die Sie schreiben, im URL-Format „file:///".

  Hinweis: *Wenn der Pfad Leerzeichen enthält, werden mit dieser Funktion keine Dateien geschrieben.*

- Das Argument *`text`* gibt den zu schreibenden String an.
- Als *`mode`*-Argument (falls angegeben) muss `append` verwendet werden. Falls dieses Argument nicht angegeben wird, überschreibt der String den Dateiinhalt.

Rückgabewerte
Ein boolescher Wert: `true` bei erfolgreichem Schreibvorgang, andernfalls `false`.

Beispiel
Der folgende Code schreibt den String `xxx` in die Datei „mydata.txt" und gibt nach einem erfolgreichen Schreibvorgang eine Meldung aus. Anschließend wird der String `aaa` an den Dateiinhalt angehängt. Wenn dieser Vorgang erfolgreich verläuft, wird eine zweite Meldung ausgegeben. Nach Ausführung des Skripts enthält die Datei „mydata.txt" lediglich den Text `xxxaaa`.

```
var fileURL = "file:///c|/temp/mydata.txt";
if (DWfile.write(fileURL, "xxx")){
    alert("Wrote xxx to " + fileURL);
}
if (DWfile.write(fileURL, "aaa", "append")){
    alert("Appended aaa to " + fileURL);
}
```

Kapitel 3: HTTP-API

Die Einsatzmöglichkeiten der Erweiterungen beschränken sich nicht auf das lokale Dateisystem. Adobe® Dreamweaver® bietet einen Mechanismus, mit dem über HTTP (Hypertext Transfer Protocol) Informationen von einem Webserver abgerufen bzw. an diesen gesendet werden können. In diesem Kapitel werden die HTTP-API und deren Verwendung erläutert.

Funktionsweise der HTTP-API

Alle Funktionen der HTTP-API sind Methoden des Objekts `MMHttp`. Bei den meisten dieser Funktionen wird als Argument eine URL verwendet und die meisten geben ein Objekt zurück. Als Standardanschluss für URL-Argumente wird Anschluss 80 verwendet. Wenn Sie einen anderen Anschluss verwenden möchten, fügen Sie einen Doppelpunkt und die Nummer des Anschlusses an die URL an, wie im folgenden Beispiel gezeigt:

```
MMHttp.getText("http://www.myserver.com:8025");
```

Bei Funktionen, die ein Objekt zurückgeben, verfügt das Objekt über zwei Eigenschaften: `statusCode` und `data`.

Die Eigenschaft `statusCode` gibt den Status der Operation an. Mögliche Werte sind unter anderem:

- 200: Status OK
- 400: Unverständliche Anforderung
- 404: Angeforderte URL nicht gefunden
- 405: Server unterstützt angeforderte Methode nicht
- 500: Unbekannter Serverfehler
- 503: Serverkapazität erreicht

Eine umfassende Liste der Statuscodes für Ihren Server erhalten Sie bei Ihrem Internetdienstanbieter oder Systemadministrator.

Der Wert der Eigenschaft `data` ist von der jeweiligen Funktion abhängig. Die möglichen Werte sind bei den einzelnen Funktionsbeschreibungen angegeben.

Funktionen, die ein Objekt zurückgeben, sind jeweils auch in einer sogenannten Callback-Version vorhanden. Bei Callback-Funktionen können andere Funktionen ausgeführt werden, während der Webserver eine HTTP-Anforderung bearbeitet. Dies ist dann nützlich, wenn Sie in Dreamweaver mehrere HTTP-Anforderungen ausgeben. Die Callback-Version einer Funktion übergibt ihre ID und ihren Rückgabewert direkt an die als ihr erstes Argument definierte Funktion.

Funktionen der HTTP-API

In diesem Abschnitt werden die Funktionen, bei denen es sich um Methoden des Objekts `MMHttp` handelt, ausführlich erläutert.

MMHttp.clearServerScriptsFolder()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Löscht den Ordner „_mmServerScripts“ und alle darin enthaltenen Dateien im Stammordner der aktuellen lokalen Site oder Remote-Site. Der Ordner „_mmServerScripts“ befindet sich im Ordner „Configuration/Connections/Scripts/*Servermodell*/_mmDBScripts“.

Argumente
serverScriptsfolder

- *serverScriptsfolder* ist ein String, der einen bestimmten Ordner relativ zum Ordner „Configuration“ auf dem Anwendungsserver bezeichnet, von dem Sie Serverskripts abrufen und löschen möchten.

Rückgabewerte
Ein Objekt, das die Antwort vom Server repräsentiert. Die Eigenschaft `data` dieses Objekts ist ein String mit dem Inhalt der gelöschten Skripts. Wenn ein Fehler auftritt, zeigt Dreamweaver diesen in der Eigenschaft `statusCode` des zurückgegebenen Objekts an.

Beispiel
Wenn der folgende Code in einer Menübefehlsdatei im Ordner „Configuration/Menus“ steht, entfernt er beim Aufruf über ein Menü alle Dateien aus dem Ordner „_mmServerScripts“:

```
<!-- MENU-LOCATION=NONE -->
<html>
<head>
<TITLE>Clear Server Scripts</TITLE>
<SCRIPT SRC="ClearServerScripts.js"></SCRIPT>
<SCRIPT LANGUAGE="javascript">
</SCRIPT>
<body onLoad="MMHttp.clearServerScriptsFolder()">
</body>
</html>
```

MMHttp.clearTemp()

Beschreibung
Diese Funktion löscht alle Dateien aus dem Ordner „Configuration/Temp“, der sich im Anwendungsordner von Dreamweaver befindet.

Argumente
Keine.

Rückgabewerte
Keine.

Beispiel
Wenn der folgende Code als Datei im Ordner „Configuration/Shutdown“ gespeichert wird, werden beim Beenden von Dreamweaver alle Dateien aus dem Ordner „Configuration/Temp“ gelöscht:

```
<html>
<head>
<title>Clean Up Temp Files on Shutdown</title>
</head>
<body onLoad="MMHttp.clearTemp()">
</body>
</html>
```

MMHttp.getFile()

Beschreibung
Diese Funktion ruft die Datei unter der angegebenen URL ab und speichert sie im Ordner „Configuration/Temp“, der sich im Anwendungsordner von Dreamweaver befindet. Dreamweaver erstellt automatisch Unterordner, durch die die Ordnerstruktur des Servers nachgeahmt wird. Lautet der angegebene Pfad beispielsweise „www.dreamcentral.com/people/index.html“, speichert Dreamweaver die Datei „index.html“ im Unterordner „People“ des Ordners „www.dreamcentral.com“.

Argumente
URL, {prompt}, {saveURL}, {titleBarLabel}

- Das Argument *URL* ist eine absolute URL auf einem Webserver. Wenn http:// in der URL nicht angegeben wird, geht Dreamweaver davon aus, dass HTTP verwendet wird.
- Das optionale Argument *prompt* ist ein boolescher Wert, der angibt, ob der Benutzer zum Speichern der Datei aufgefordert wird. Wenn sich *saveURL* auf einen Speicherort außerhalb des Ordners „Configuration/Temp“ bezieht, wird beim Argument *prompt* der Wert `false` aus Sicherheitsgründen ignoriert.
- Das optionale Argument *saveURL* ist der Speicherort auf der Festplatte des Benutzers, an dem die Datei gespeichert werden soll (im URL-Format „file://“). Wenn für „prompt“ der Wert `true` definiert wurde oder sich *saveURL* auf einen Speicherort außerhalb des Ordners „Configuration/Temp“ bezieht, kann der Benutzer *saveURL* im Dialogfeld „Speichern“ überschreiben.
- Das optionale Argument *titleBarLabel* ist die Bezeichnung, die in der Titelleiste des Dialogfelds „Speichern“ angezeigt werden soll.

Rückgabewerte
Ein Objekt, das die Antwort vom Server repräsentiert. Die Eigenschaft `data` dieses Objekts ist ein String, der den Speicherort der Datei enthält (im URL-Format „file://“). Normalerweise enthält die Eigenschaft `statusCode` des Objekts den vom Server empfangenen Statuscode. Wenn jedoch beim Speichern auf dem lokalen Laufwerk ein Datenträgerfehler aufgetreten ist und der Vorgang nicht erfolgreich ausgeführt werden kann, enthält die Eigenschaft `statusCode` einen Fehlercode in Form eines der folgenden ganzzahligen Werte:

- 1: Unbekannter Fehler
- 2: Datei nicht gefunden
- 3: Ungültiger Pfad
- 4: Zu viele Dateien geöffnet
- 5: Zugriff verweigert

- 6: Ungültiges Datei-Handle
- 7: Aktueller Arbeitsordner kann nicht entfernt werden
- 8: Keine weiteren Ordnereinträge
- 9: Fehler beim Setzen des Dateizeigers
- 10: Hardwarefehler
- 11: Zugriffsverletzung
- 12: Sperrverletzung
- 13: Festplatte voll
- 14: Dateiende erreicht

Beispiel

Mit dem folgenden Code wird zunächst eine HTML-Datei abgerufen. Dann werden alle Dateien im Ordner „Configuration/Temp“ gespeichert und die lokale Kopie der HTML-Datei wird in einem Browser geöffnet:

```
var httpReply = MMHttp.getFile("http://www.dreamcentral.com/people/profiles/scott.html",
false);
if (Boolean == 200){
    var saveLoc = httpReply.data;
    dw.browseDocument(saveLoc);
}
```

MMHttp.getFileCallback()

Beschreibung

Diese Funktion ruft die Datei unter der angegebenen URL ab, speichert die Datei im Ordner „Configuration/Temp“ innerhalb des Anwendungsordners von Dreamweaver und ruft anschließend die angegebene Funktion mit der Anforderungs-ID und dem Antwortergebnis auf. Wenn die Datei lokal gespeichert wird, erstellt Dreamweaver automatisch Unterordner, durch die die Ordnerstruktur des Servers nachgeahmt wird. Lautet der angegebene Pfad beispielsweise „www.dreamcentral.com/people/index.html“, speichert Dreamweaver die Datei „index.html“ im Unterordner „People“ des Ordners „www.dreamcentral.com“.

Argumente

callbackFunction, URL, {prompt}, {saveURL}, {titleBarLabel}

- Das Argument *callbackFunction* ist der Name der JavaScript-Funktion, die nach erfolgter HTTP-Anfrage aufgerufen werden soll.
- Das Argument *URL* ist eine absolute URL auf einem Webserver. Wenn http:// in der URL nicht angegeben wird, geht Dreamweaver davon aus, dass HTTP verwendet wird.
- Das optionale Argument *prompt* ist ein boolescher Wert, der angibt, ob der Benutzer zum Speichern der Datei aufgefordert wird. Wenn sich *saveURL* auf einen Speicherort außerhalb des Ordners „Configuration/Temp“ bezieht, wird beim Argument *prompt* der Wert `false` aus Sicherheitsgründen ignoriert.
- Das optionale Argument *saveURL* ist der Speicherort auf der Festplatte des Benutzers, an dem die Datei gespeichert werden soll (im URL-Format „file://“). Wenn für „prompt“ der Wert `true` definiert wurde oder sich *saveURL* auf einen Speicherort außerhalb des Ordners „Configuration/Temp“ bezieht, kann der Benutzer *saveURL* im Dialogfeld „Speichern“ überschreiben.

- Das optionale Argument *titleBarLabel* ist die Bezeichnung, die in der Titelleiste des Dialogfelds „Speichern“ angezeigt werden soll.

Rückgabewerte
Ein Objekt, das die Antwort vom Server repräsentiert. Die Eigenschaft `data` dieses Objekts ist ein String, der den Speicherort der Datei enthält (im URL-Format „file://“). Normalerweise enthält die Eigenschaft `statusCode` des Objekts den vom Server empfangenen Statuscode. Wenn jedoch beim Speichern auf dem lokalen Laufwerk ein Datenträgerfehler aufgetreten ist, enthält die Eigenschaft `statusCode` einen ganzzahligen Fehlercode. Eine Liste der möglichen Fehlercodes finden Sie unter „MMHttp.getFile()“ auf Seite 16.

MMHttp.getText()

Verfügbarkeit
Dreamweaver UltraDev 4, verbessert in Dreamweaver MX.

Beschreibung
Ruft den Inhalt des Dokuments unter der angegebenen URL ab.

Argumente
URL, {serverScriptsFolder}

- Das Argument *URL* ist eine absolute URL auf einem Webserver. Wenn http:// in der URL nicht angegeben wird, geht Dreamweaver davon aus, dass HTTP verwendet wird.
- Das Argument *serverScriptsFolder* ist ein optionaler String, der einen bestimmten Ordner relativ zum Ordner „Configuration“ auf dem Anwendungsserver bezeichnet, von dem Sie Serverskripts abrufen möchten. Zum Abrufen der Skripts verwendet Dreamweaver das entsprechende Übertragungsprotokoll (z. B. FTP, WebDAV oder Remote-Dateisystem). Dreamweaver kopiert diese Dateien in den Unterordner „_mmServerScripts“ im Stammordner der aktuellen Site.

Wenn ein Fehler auftritt, zeigt Dreamweaver diesen in der Eigenschaft `statusCode` des zurückgegebenen Objekts an.

MMHttp.getTextCallback()

Verfügbarkeit
Dreamweaver UltraDev 4, verbessert in Dreamweaver MX.

Beschreibung
Ruft den Inhalt des Dokuments unter der angegebenen URL ab und übergibt ihn an die angegebene Funktion.

Argumente
callbackFunc, URL, {serverScriptsFolder}

- Das Argument *callbackFunc* ist die JavaScript-Funktion, die nach erfolgter HTTP-Anfrage aufgerufen werden soll.
- Das Argument *URL* ist eine absolute URL auf einem Webserver. Wenn http:// in der URL nicht angegeben wird, geht Dreamweaver davon aus, dass HTTP verwendet wird.

- Das Argument *serverScriptsFolder* ist ein optionaler String, der einen bestimmten Ordner relativ zum Ordner „Configuration“ auf dem Anwendungsserver bezeichnet, von dem Sie Serverskripts abrufen möchten. Zum Abrufen der Skripts verwendet Dreamweaver das entsprechende Übertragungsprotokoll (z. B. FTP, WebDAV oder Remote-Dateisystem). Dreamweaver ruft diese Dateien ab und übergibt sie an die Funktion, die durch *callbackFunc* angegeben wird.

Wenn ein Fehler auftritt, zeigt Dreamweaver MX diesen in der Eigenschaft `statusCode` des zurückgegebenen Objekts an.

MMHttp.postText()

Verfügbarkeit
Dreamweaver UltraDev 4, verbessert in Dreamweaver MX.

Beschreibung
Führt einen HTTP-Sendevorgang durch, um die angegebenen Daten an die angegebene URL zu übertragen. In der Regel handelt es sich dabei um formularkodierten Text, doch kann mit der Funktion jeder beliebige Datentyp übertragen werden, den der Server erwartet.

Argumente
URL, dataToPost, {contentType}, {serverScriptsFolder}

- Das Argument *URL* ist eine absolute URL auf einem Webserver. Wenn http:// in der URL nicht angegeben wird, geht Dreamweaver davon aus, dass HTTP verwendet wird.
- Das Argument *dataToPost* gibt die zu sendenden Daten an. Wenn das dritte Argument `"application/x-www-form-urlencoded"` lautet oder weggelassen wird, muss das Argument *dataToPost* gemäß Abschnitt 8.2.1 der Spezifikation RFC 1866 (zu finden unter www.faqs.org/rfcs/rfc1866.html) formularkodiert sein.
- Das optionale Argument *contentType* bezeichnet den Inhaltstyp der Daten, die durch das post-Verfahren übertragen werden sollen. Bei fehlendem Argument wird als Standardwert `"application/x-www-form-urlencoded"` verwendet.
- Das Argument *serverScriptsFolder* ist ein optionaler String, der einen bestimmten Ordner relativ zum Ordner „Configuration“ auf dem Anwendungsserver bezeichnet, auf den Sie Daten übertragen möchten. Zum Senden der Daten verwendet Dreamweaver das entsprechende Übertragungsprotokoll (z. B. FTP, WebDAV oder Remote-Dateisystem).

Wenn ein Fehler auftritt, zeigt Dreamweaver diesen in der Eigenschaft `statusCode` des zurückgegebenen Objekts an.

Beispiel
Im folgenden Beispiel für einen `MMHttp.postText()`-Funktionsaufruf wird davon ausgegangen, dass ein Entwickler die Datei „myScripts.cfm“ im Ordner „DeployScripts“ abgelegt hat, der sich im Ordner „Configuration“ des lokalen Computers befindet:

```
MMHttp.postText(
    "http://ultraqa8/DeployScripts/myScripts.cfm",
    "arg1=Foo",
    "application/x-www-form-urlencoded",
    "Configuration/DeployScripts/"
)
```

Sobald Dreamweaver diesen Funktionsaufruf ausführt, läuft Folgendes ab:

1. Die Datei „myScripts.cfm“ im Ordner „Configuration/DeployScripts“ des lokalen Computers wird in einen anderen Ordner „DeployScripts“ kopiert, der ein Unterordner des Stammordners der Website „ultraqa8“ ist. Zum Bereitstellen der Dateien verwendet Dreamweaver das bei der Konfiguration der Site angegebene Protokoll.
2. Dreamweaver verwendet das HTTP-Protokoll, um die `arg1=Foo`-Daten an den Webserver zu übertragen.
3. Das Ergebnis der Übertragungsanfrage ist, dass der Webserver von „ultraqa8“ das Skript „myScripts.cfm“ unter Verwendung der `arg1`-Daten ausführt.

MMHttp.postTextCallback()

Verfügbarkeit
Dreamweaver UltraDev 4, verbessert in Dreamweaver MX.

Beschreibung
Führt einen HTTP-Sendevorgang durch, um den Text an die angegebene URL zu übertragen, und übergibt die Antwort vom Server an die angegebene Funktion. In der Regel handelt es sich dabei um formularkodierten Text, doch kann mit der Funktion jeder beliebige Datentyp übertragen werden, den der Server erwartet.

Argumente
callbackFunc, URL, dataToPost, {contentType}, {serverScriptsFolder}

- Das Argument *callbackFunc* ist der Name der JavaScript-Funktion, die nach erfolgter HTTP-Anfrage aufgerufen werden soll.
- Das Argument *URL* ist eine absolute URL auf einem Webserver. Wenn http:// in der URL nicht angegeben wird, geht Dreamweaver davon aus, dass HTTP verwendet wird.
- Das Argument *dataToPost* gibt die zu sendenden Daten an. Wenn das dritte Argument `"application/x-www-form-urlencoded"` lautet oder weggelassen wird, muss das Argument data gemäß Abschnitt 8.2.1 der Spezifikation RFC 1866 (zu finden unter www.faqs.org/rfcs/rfc1866.html) formularkodiert sein.
- Das optionale Argument *contentType* bezeichnet den Inhaltstyp der Daten, die durch das post-Verfahren übertragen werden sollen. Bei fehlendem Argument wird als Standardwert `"application/x-www-form-urlencoded"` verwendet.
- *serverScriptsFolder* ist ein optionaler String. Er bezeichnet einen bestimmten Ordner relativ zum Ordner „Configuration“ auf dem Anwendungsserver, an den Sie Daten senden möchten. Zum Senden der Daten verwendet Dreamweaver das entsprechende Übertragungsprotokoll (z. B. FTP, WebDAV oder Remote-Dateisystem). Dreamweaver ruft diese Daten ab und übergibt sie an die Funktion, die durch *callbackFunc* angegeben wird.

Wenn ein Fehler auftritt, zeigt Dreamweaver diesen in der Eigenschaft `statusCode` des zurückgegebenen Objekts an.

Kapitel 4: API für Design Notes

Mit Adobe® Dreamweaver®, Adobe® Fireworks® und Adobe® Flash® haben Webdesigner und Webentwickler die Möglichkeit, zusätzliche Informationen über Dokumente zu speichern und abzurufen. Diese Informationen sind in Dateien gespeichert, die als „Design Notes“ bezeichnet werden. Zu diesen Informationen zählen Revisionskommentare, Änderungshinweise oder die Quelldateien für GIF- oder JPEG-Dateien.

Weitere Informationen zur Verwendung von Design Notes in Dreamweaver finden Sie in der Dokumentation *Dreamweaver verwenden*.

Funktionsweise von Design Notes

In jeder Design Notes-Datei sind Informationen zu einem einzigen Dokument gespeichert. Wenn mit einem oder mehreren Dokumenten in einem Ordner eine Design Notes-Datei verknüpft ist, erstellt Dreamweaver einen Unterordner mit dem Namen „_notes“, in dem Design Notes-Dateien gespeichert werden können. Der Ordner „_notes“ und die darin enthaltenen Design Notes-Dateien werden im Bedienfeld „Dateien“ nicht angezeigt, jedoch im Finder (Macintosh) bzw. in Windows Explorer. Design Notes-Dateinamen bestehen aus dem Hauptdateinamen und der Erweiterung „.mno“. Die Design Notes-Datei für die Datei „avocado8.gif“ trägt beispielsweise den Namen „avocado8.gif.mno“.

Design Notes-Dateien sind XML-Dateien, in denen Informationen in einer Folge von Schlüssel-Wert-Paaren gespeichert sind. Der Schlüssel beschreibt den gespeicherten Informationstyp und der Wert stellt die eigentlichen Informationen dar. Schlüssel können höchstens 64 Zeichen lang sein.

Im folgenden Beispiel ist die Design Notes-Datei für die Datei „foghorn.gif.mno“ dargestellt:

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<info>
    <infoitem key="FW_source" value="file:///C|sites/dreamcentral/images/sourceFiles/¬
        foghorn.png" />
    <infoitem key="Author" value="Heidi B." />
    <infoitem key="Status" value="Final draft, approved by Jay L." />
</info>
```

JavaScript-API für Design Notes

Alle Funktionen der JavaScript-API für Design Notes sind Methoden des MMNotes-Objekts. MMNotes ist eine gemeinsam genutzte C-Bibliothek, mit deren Hilfe die Autoren von Erweiterungen Design Notes-Dateien lesen und schreiben können. Wie die gemeinsam genutzte Bibliothek DWfile verfügt auch MMNotes über eine JavaScript-API, über die Sie die Funktionen aufrufen können. Die Funktionen werden über Objekte, Befehle, Verhalten, schwebende Bedienfelder, Eigenschafteninspektoren und Datenübersetzer der Bibliothek aufgerufen. Die gemeinsam genutzte Bibliothek MMNotes kann unabhängig von Dreamweaver verwendet werden, selbst wenn Dreamweaver nicht installiert ist.

MMNotes.close()

Beschreibung
Diese Funktion schließt die angegebene Design Notes-Datei und speichert alle Änderungen. Wenn alle Schlüssel-Wert-Paare entfernt wurden, löscht Dreamweaver die Design Notes-Datei. Wenn es sich um die letzte Design Notes-Datei im Ordner „_notes“ handelt, wird auch der Ordner gelöscht.

Hinweis: *Rufen Sie nach Verwendung von Design Notes immer die Funktion* `MMNotes.close()` *auf, damit Dreamweaver das Schreiben in die Datei abschließt.*

Argumente
fileHandle

- Das Argument *fileHandle* ist das von der Funktion `MMNotes.open()` zurückgegebene Datei-Handle.

Rückgabewerte
Keine.

Beispiel
Siehe „MMNotes.set()“ auf Seite 26.

MMNotes.filePathToLocalURL()

Beschreibung
Diese Funktion wandelt den angegebenen lokalen Laufwerkspfad in das URL-Format „file://“ um.

Argumente
drivePath

- Das Argument *drivePath* ist ein String, der den vollständigen Laufwerkspfad enthält.

Rückgabewerte
Ein String mit der URL der angegebenen Datei im URL-Format „file://“.

Beispiel
Beim Aufruf von `MMNotes.filePathToLocalURL('C:\sites\webdev\index.htm')` wird der String `"file:///c|/sites/webdev/index.htm"` zurückgegeben.

MMNotes.get()

Beschreibung
Diese Funktion ruft den Wert des angegebenen Schlüssels in einer Design Notes-Datei ab.

Argumente
fileHandle, keyName

- Das Argument *fileHandle* ist das von `MMNotes.open()` zurückgegebene Datei-Handle.
- Das Argument *keyName* ist ein String, der den Namen des Schlüssels enthält.

Rückgabewerte
Ein String mit dem Wert des Schlüssels.

Beispiel
Siehe „MMNotes.getKeys()“ auf Seite 23.

MMNotes.getKeyCount()

Beschreibung
Diese Funktion ruft die Anzahl der Schlüssel-Wert-Paare in der angegebenen Design Notes-Datei ab.

Argumente
fileHandle

- Das Argument *fileHandle* ist das von der Funktion `MMNotes.open()` zurückgegebene Datei-Handle.

Rückgabewerte
Eine Ganzzahl, die die Anzahl der Schlüssel-Wert-Paare in der Design Notes-Datei angibt.

MMNotes.getKeys()

Beschreibung
Diese Funktion ruft eine Liste sämtlicher Schlüssel in einer Design Notes-Datei ab.

Argumente
fileHandle

- Das Argument *fileHandle* ist das von der Funktion `MMNotes.open()` zurückgegebene Datei-Handle.

Rückgabewerte
Ein String-Array, in dem jeder String den Namen eines Schlüssels enthält.

Beispiel
Der folgende Code kann in einem benutzerdefinierten schwebenden Bedienfeld verwendet werden, um die Design Notes-Informationen für das aktive Dokument anzuzeigen.

```
var noteHandle = MMNotes.open(dw.getDocumentDOM().URL);
var theKeys = MMNotes.getKeys(noteHandle);
var noteString = "";
var theValue = "";
for (var i=0; i < theKeys.length; i++){
    theValue = MMNotes.get(noteHandle,theKeys[i]);
    noteString +=0theKeys[i] + " = " theValue + "\n";
}
document.theForm.bigTextField.value = noteString;
// always close noteHandle
MMNotes.close(noteHandle);
```

MMNotes.getSiteRootForFile()

Beschreibung
Diese Funktion ermittelt den Stammordner für die angegebene Design Notes-Datei.

Argumente
fileURL

- Das Argument *fileURL* im URL-Format „file://“ ist der Pfad zu einer lokalen Datei.

Rückgabewerte
Ein String, der den Pfad des lokalen Stammordners der Site enthält (im URL-Format „file://“) oder der leer ist, wenn Dreamweaver nicht installiert ist oder die Design Notes-Datei sich außerhalb einer mit Dreamweaver definierten Site befindet. Diese Funktion sucht nach allen Sites, die in Dreamweaver definiert sind.

MMNotes.getVersionName()

Beschreibung
Diese Funktion ruft den Versionsnamen der gemeinsam genutzten Bibliothek MMNotes ab, der gleichzeitig die implementierende Anwendung angibt.

Argumente
Keine.

Rückgabewerte
Ein String mit dem Namen der Anwendung, von der die gemeinsam genutzte Bibliothek MMNotes implementiert wurde.

Beispiel
Wenn die Funktion `MMNotes.getVersionName()` über einen Befehl, ein Objekt, ein Verhalten, einen Eigenschafteninspektor, ein schwebendes Bedienfeld oder einen Datenübersetzer in Dreamweaver aufgerufen wurde, wird der String `"Dreamweaver"` zurückgegeben. Durch Aufrufen der Funktion `MMNotes.getVersionName()` über Fireworks wird ebenfalls `"Dreamweaver"` zurückgegeben, da Fireworks dieselbe Bibliotheksversion verwendet, die vom Dreamweaver-Entwicklungsteam erstellt wurde.

MMNotes.getVersionNum()

Beschreibung
Diese Funktion ruft die Versionsnummer der gemeinsam genutzten Bibliothek MMNotes ab.

Argumente
Keine.

Rückgabewerte
Ein String, der die Versionsnummer enthält.

MMNotes.localURLToFilePath()

Beschreibung
Diese Funktion wandelt den im URL-Format „file://“ angegebenen Pfad in einen lokalen Laufwerkspfad um.

Argumente
fileURL

- Das Argument *fileURL* im URL-Format „file://“ ist der Pfad zu einer lokalen Datei.

Rückgabewerte
Ein String, der den lokalen Laufwerkspfad für die angegebene Datei enthält.

Beispiel
Beim Aufruf von `MMNotes.localURLToFilePath('file:///MacintoshHD/images/moon.gif')` wird der String `"MacintoshHD:images:moon.gif"` zurückgegeben.

MMNotes.open()

Beschreibung
Diese Funktion öffnet die der angegebenen Datei zugeordnete Design Notes-Datei oder erstellt sie gegebenenfalls.

Argumente
filePath, {bForceCreate}

- Das Argument *filePath* im URL-Format „file://“ ist der Pfad zur Hauptdatei, mit der die Design Notes-Datei verknüpft ist.
- Das Argument *bForceCreate* ist ein boolescher Wert, der angibt, ob die Design Notes-Datei auch dann erstellt werden soll, wenn Design Notes für die Site deaktiviert ist oder wenn *filePath* mit keiner Site verknüpft ist.

Rückgabewerte
Das Datei-Handle für die Design Notes-Datei oder Null (0), wenn die Datei nicht geöffnet oder erstellt wurde.

Beispiel
Siehe „MMNotes.set()“ auf Seite 26.

MMNotes.remove()

Beschreibung
Diese Funktion entfernt den angegebenen Schlüssel (und seinen Wert) aus einer Design Notes-Datei.

Argumente
fileHandle, keyName

- Das Argument *fileHandle* ist das von der Funktion `MMNotes.open()` zurückgegebene Datei-Handle.
- Das Argument *keyName* ist ein String, der den Namen des zu entfernenden Schlüssels enthält.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der Vorgang erfolgreich war, andernfalls `false`.

MMNotes.set()

Beschreibung
Diese Funktion erstellt oder aktualisiert ein Schlüssel-Wert-Paar in einer Design Notes-Datei.

Argumente
fileHandle, keyName, valueString

- Das Argument *fileHandle* ist das von der Funktion `MMNotes.open()` zurückgegebene Datei-Handle.
- Das Argument *keyName* ist ein String, der den Namen des Schlüssels enthält.
- Das Argument *valueString* ist ein String, der den Wert enthält.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der Vorgang erfolgreich war, andernfalls `false`.

Beispiel
Mit dem folgenden Beispielcode wird die mit der Site „dreamcentral“ verknüpfte Design Notes-Datei „peakhike99/index.html“ geöffnet, dieser Datei ein neues Schlüssel-Wert-Paar hinzugefügt, der Wert eines vorhandenen Schlüssels geändert und dann die Design Notes-Datei geschlossen.

```
var noteHandle = MMNotes.open('file:///c|/sites/dreamcentral/peakhike99/
    index.html',true);
if(noteHandle > 0){
    MMNotes.set(noteHandle,"Author","M. G. Miller");
    MMNotes.set(noteHandle,"Last Changed","August 28, 1999");
    MMNotes.close(noteHandle);
}
```

C-API für Design Notes

Neben der JavaScript-API enthält die gemeinsam genutzte Bibliothek MMNotes auch eine C-API, über die andere Anwendungen Design Notes-Dateien erstellen können. Wenn Sie die Bibliothek MMNotes in Dreamweaver verwenden, müssen Sie diese C-Funktionen nicht direkt aufrufen, da sie von den entsprechenden JavaScript-Versionen der Funktionen aufgerufen werden.

In diesem Abschnitt werden die Funktionen sowie deren Argumente und Rückgabewerte beschrieben. Sie finden die Definition der Funktionen und Datentypen in der Datei „MMInfo.h“ im Ordner „Extending/c_files“ des Anwendungsordners von Dreamweaver.

void CloseNotesFile()

Beschreibung
Diese Funktion schließt die angegebene Design Notes-Datei und speichert alle Änderungen. Wenn alle Schlüssel-Wert-Paare aus der Design Notes-Datei entfernt wurden, wird die Datei in Dreamweaver gelöscht. Dreamweaver löscht den Ordner „_notes“, nachdem die letzte Design Notes-Datei gelöscht wurde.

Argumente
noteHandle

- Das Argument *noteHandle* ist das von der Funktion `OpenNotesFile()` zurückgegebene Datei-Handle.

Rückgabewerte
Keine.

BOOL FilePathToLocalURL()

Beschreibung
Diese Funktion wandelt den angegebenen lokalen Laufwerkspfad in das URL-Format „file://“ um.

Argumente
const char* *drivePath*, char* *localURLBuf*, int *localURLMaxLen*

- Das Argument *drivePath* ist ein String, der den vollständigen Laufwerkspfad enthält.
- Das Argument *localURLBuf* ist der Puffer, in dem der URL-String „file://“ gespeichert wird.
- Das Argument *localURLMaxLen* ist die maximale Größe von *localURLBuf*.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der Vorgang erfolgreich war, andernfalls `false`. Das Argument *localURLBuf* erhält den Wert des URL-Strings „file://“.

BOOL GetNote()

Beschreibung
Diese Funktion ruft den Wert des angegebenen Schlüssels in einer Design Notes-Datei ab.

Argumente
FileHandle *noteHandle*, const char *keyName[64]*, char* *valueBuf*, int *valueBufLength*

- Das Argument *noteHandle* ist das von der Funktion `OpenNotesFile()` zurückgegebene Datei-Handle.
- Das Argument *keyName[64]* ist ein String, der den Namen des Schlüssels enthält.
- Das Argument *valueBuf* ist der Puffer, in dem der Wert gespeichert wird.
- Das Argument *valueBufLength* ist die Ganzzahl, die von `GetNoteLength(noteHandle, keyName)` zurückgegeben wird und die maximale Länge des Wertepuffers angibt.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der Vorgang erfolgreich war, andernfalls `false`. Das Argument *valueBuf* erhält den Wert des Schlüssels.

Beispiel
Mit dem folgenden Code wird der Wert des Schlüssels `comments` in der mit der Datei „welcome.html“ verknüpften Design Notes-Datei abgerufen.

```
FileHandle noteHandle = OpenNotesFile("file:///c|/sites/avocado8/iwjs/welcome.html");
if(noteHandle > 0){
    int valueLength = GetNoteLength( noteHandle, "comments");
    char* valueBuffer = new char[valueLength + 1];
    GetNote(noteHandle, "comments", valueBuffer, valueLength + 1);
    printf("Comments: %s",valueBuffer);
    CloseNotesFile(noteHandle);
}
```

int GetNoteLength()

Beschreibung
Diese Funktion ruft die Länge des Werts für den angegebenen Schlüssel ab.

Argumente
FileHandle *noteHandle*, const char *keyName[64]*

- Das Argument *noteHandle* ist das von der Funktion `OpenNotesFile()` zurückgegebene Datei-Handle.
- Das Argument *keyName[64]* ist ein String, der den Namen des Schlüssels enthält.

Rückgabewerte
Eine Ganzzahl, die die Länge des Werts angibt.

Beispiel
Siehe „BOOL GetNote()“ auf Seite 27.

int GetNotesKeyCount()

Beschreibung
Diese Funktion ruft die Anzahl der Schlüssel-Wert-Paare in der angegebenen Design Notes-Datei ab.

Argumente
FileHandle *noteHandle*

- Das Argument *noteHandle* ist das von der Funktion `OpenNotesFile()` zurückgegebene Datei-Handle.

Rückgabewerte
Eine Ganzzahl, die die Anzahl der Schlüssel-Wert-Paare in der Design Notes-Datei angibt.

BOOL GetNotesKeys()

Beschreibung
Diese Funktion ruft eine Liste sämtlicher Schlüssel in einer Design Notes-Datei ab.

Argumente
FileHandle *noteHandle*, char* *keyBufArray[64],* int *keyArrayMaxLen*

- Das Argument *noteHandle* ist das von `OpenNotesFile()` zurückgegebene Datei-Handle.
- Das Argument *keyBufArray[64]* ist das Puffer-Array, in dem die Schlüssel gespeichert werden.
- Das Argument *keyArrayMaxLen* ist die von `GetNotesKeyCount(noteHandle)` zurückgegebene Ganzzahl, die die maximale Anzahl von Elementen im Schlüsselpuffer-Array angibt.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der Vorgang erfolgreich war, andernfalls `false`. Das Argument *keyBufArray* erhält die Schlüsselnamen.

Beispiel
Mit dem folgenden Code werden die Schlüsselnamen und -werte sämtlicher Schlüssel in der Design Notes-Datei ausgegeben, die mit der Datei „welcome.html“ verknüpft ist.

```
typedef char[64] InfoKey;
FileHandle noteHandle = OpenNotesFile("file:///c|/sites/avocado8/iwjs/welcome.html");
if (noteHandle > 0){
    int keyCount = GetNotesKeyCount(noteHandle);
    if (keyCount <= 0)
        return;
    InfoKey* keys = new InfoKey[keyCount];
    BOOL succeeded = GetNotesKeys(noteHandle, keys, keyCount);
    if (succeeded){
        for (int i=0; i < keyCount; i++){
            printf("Key is: %s\n", keys[i]);
            printf("Value is: %s\n\n", GetNote(noteHandle, keys[i]);
        }
    }
    delete []keys;
}
CloseNotesFile(noteHandle);
```

BOOL GetSiteRootForFile()

Beschreibung
Diese Funktion ermittelt den Stammordner für die angegebene Design Notes-Datei.

Argumente
`const char*`*filePath*`, char*`*siteRootBuf*`, int`*siteRootBufMaxLen*`, {InfoPrefs*` *infoPrefs}*

- Das Argument *filePath* im URL-Format „file://“ ist die Datei, deren Stammordner ermittelt werden soll.
- Das Argument *siteRootBuf* ist der Puffer, in dem der Stammordner gespeichert wird.
- Das Argument *siteRootBufMaxLen* ist die maximale Größe des Puffers, auf den *siteRootBuf* verweist.

- Das optionale Argument *infoPrefs* ist ein Verweis auf eine `struct`, in der die Voreinstellungen für die Site gespeichert werden.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der Vorgang erfolgreich war, andernfalls `false`. Das Argument *siteRootBuf* erhält die Adresse des Puffers, in dem der Stammordner gespeichert wird. Wenn Sie das Argument *infoPrefs* angeben, werden auch die Design Notes-Voreinstellungen für die Site zurückgegeben. `InfoPrefs` verfügt über zwei Variablen: `bUseDesignNotes` und `bUploadDesignNotes`, beide vom Typ `BOOL`.

BOOL GetVersionName()

Beschreibung
Diese Funktion ruft den Versionsnamen der gemeinsam genutzten Bibliothek MMNotes ab, der gleichzeitig die implementierende Anwendung angibt.

Argumente
`char* versionNameBuf, int`*versionNameBufMaxLen*

- Das Argument *versionNameBuf* ist der Puffer, in dem der Versionsname gespeichert wird.
- Das Argument *versionNameBufMaxLen* ist die maximale Größe des Puffers, auf den das Argument *versionNameBuf* verweist.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der Vorgang erfolgreich war, andernfalls `false`. Dreamweaver speichert `"Dreamweaver"` im Argument *versionNameBuf*.

BOOL GetVersionNum()

Beschreibung
Diese Funktion ruft die Versionsnummer der gemeinsam genutzten Bibliothek MMNotes ab, sodass Sie ermitteln können, ob bestimmte Funktionen verfügbar sind.

Argumente
`char* versionNumBuf, int`*versionNumBufMaxLen*

- Das Argument *versionNumBuf* ist der Puffer, in dem die Versionsnummer gespeichert wird.
- Das Argument *versionNumBufMaxLen* ist die maximale Größe des Puffers, auf den *versionNumBuf* verweist.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der Vorgang erfolgreich war, andernfalls `false`. Das Argument *versionNumBuf* speichert die Versionsnummer.

BOOL LocalURLToFilePath()

Beschreibung
Diese Funktion wandelt den im URL-Format „file://“ angegebenen Pfad in einen lokalen Laufwerkspfad um.

Argumente
const char* *localURL,* char* *drivePathBuf,* int *drivePathMaxLen*

- Das Argument *localURL* im URL-Format „file:///“ ist der Pfad zu einer lokalen Datei.
- Das Argument *drivePathBuf* ist der Puffer, in dem der lokale Laufwerkspfad gespeichert wird.
- Das Argument *drivePathMaxLen* ist die maximale Größe des Puffers, auf den das Argument *drivePathBuf* verweist.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der Vorgang erfolgreich war, andernfalls `false`. Das Argument *drivePathBuf* erhält den lokalen Laufwerkspfad.

FileHandle OpenNotesFile()

Beschreibung
Diese Funktion öffnet die der angegebenen Datei zugeordnete Design Notes-Datei oder erstellt sie gegebenenfalls.

Argumente
```
const char* localFileURL, {BOOL bForceCreate}
```

- Das Argument *localFileURL* im URL-Format „file:///“ ist ein String, der den Pfad zur Hauptdatei enthält, mit der die Design Notes-Datei verknüpft ist.
- Das Argument *bForceCreate* ist ein boolescher Wert, der angibt, ob die Design Notes-Datei auch dann erstellt werden soll, wenn Design Notes für die Site deaktiviert ist oder der für *localFileURL* angegebene Pfad mit keiner Site verknüpft ist.

FileHandle OpenNotesFilewithOpenFlags()

Beschreibung
Diese Funktion öffnet die der angegebenen Datei zugeordnete Design Notes-Datei oder erstellt sie gegebenenfalls. Die Datei kann im schreibgeschützten Modus geöffnet werden.

Argumente
```
const char* localFileURL, {BOOL bForceCreate}, {BOOL bReadOnly}
```

- Das Argument *localFileURL* im URL-Format „file:///“ ist ein String, der den Pfad zur Hauptdatei enthält, mit der die Design Notes-Datei verknüpft ist.
- Das Argument *bForceCreate* ist ein boolescher Wert, der angibt, ob die Design Notes-Datei auch dann erstellt werden soll, wenn Design Notes für die Site deaktiviert ist oder der Pfad mit keiner Site verknüpft ist. Der Standardwert ist `false`. Dieses Argument ist optional. Es muss jedoch angegeben werden, wenn Sie das dritte Argument angeben.
- Das optionale Argument *bReadOnly* ist ein boolescher Wert, der angibt, ob die Datei im schreibgeschützten Modus geöffnet werden soll. Der Standardwert ist `false`. Das Argument *bReadOnly* ist ab Version 2 der Datei „MMNotes.dll“ verfügbar.

BOOL RemoveNote()

Beschreibung
Diese Funktion entfernt den angegebenen Schlüssel (und seinen Wert) aus einer Design Notes-Datei.

Argumente
`FileHandle`*noteHandle, const char keyName[64]*

- Das Argument *noteHandle* ist das von der Funktion `OpenNotesFile()` zurückgegebene Datei-Handle.
- Das Argument *keyName[64]* ist ein String, der den Namen des zu entfernenden Schlüssels enthält.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der Vorgang erfolgreich war, andernfalls `false`.

BOOL SetNote()

Beschreibung
Diese Funktion erstellt oder aktualisiert ein Schlüssel-Wert-Paar in einer Design Notes-Datei.

Argumente
FileHandle noteHandle, const char *keyName[64],* const char* *value*

- Das Argument *noteHandle* ist das von der Funktion `OpenNotesFile()` zurückgegebene Datei-Handle.
- Das Argument *keyName[64]* ist ein String, der den Namen des Schlüssels enthält.
- Das Argument *value* ist ein String, der den Wert enthält.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der Vorgang erfolgreich war, andernfalls `false`.

Kapitel 5: Fireworks-Integration

FWLaunch ist eine gemeinsam genutzte C-Bibliothek, durch die Autoren von Objekten, Befehlen, Verhalten und Eigenschafteninspektoren mit Adobe® Fireworks® kommunizieren können. Mithilfe von FWLaunch können Sie JavaScript-Code programmieren, um die Fireworks-Benutzeroberfläche zu öffnen und Befehle an Fireworks über die zugehörige JavaScript-API zu senden. Informationen dazu finden Sie im Handbuch *Extending Fireworks*. Allgemeine Informationen über das Zusammenwirken von C-Bibliotheken mit dem JavaScript-Interpreter in Adobe® Dreamweaver® CS5 sowie Details zur C-Level-Erweiterbarkeit finden Sie im Handbuch *Erweitern von Dreamweaver*.

FWLaunch-API

Mit dem FWLaunch-Objekt können Sie Fireworks öffnen, Fireworks-Operationen über die JavaScript-API von Fireworks durchführen und anschließend Werte an Dreamweaver zurückgeben. In diesem Kapitel werden die FWLaunch-Kommunikations-API und deren Verwendung erläutert.

FWLaunch.bringDWToFront()

Verfügbarkeit
Dreamweaver 3, Fireworks 3.

Beschreibung
Diese Funktion stellt Dreamweaver in den Vordergrund.

Argumente
Keine.

Rückgabewerte
Keine.

FWLaunch.bringFWToFront()

Verfügbarkeit
Dreamweaver 3, Fireworks 3.

Beschreibung
Diese Funktion stellt Fireworks in den Vordergrund, falls es ausgeführt wird.

Argumente
Keine.

Rückgabewerte
Keine.

FWLaunch.execJsInFireworks()

Verfügbarkeit
Dreamweaver 3, Fireworks 3.

Beschreibung
Diese Funktion übergibt das angegebene JavaScript oder einen Verweis auf eine JavaScript-Datei zur Ausführung an Fireworks.

Argumente
`javascriptOrFileURL`

- Das Argument `javascriptOrFileURL` im URL-Format „file://“ ist entweder ein String mit explizitem JavaScript-Code oder der Pfad zu einer JavaScript-Datei.

Rückgabewerte
Ein Cookie-Objekt, falls die JavaScript-Daten erfolgreich übergeben wurden, oder ein Fehlercode ungleich null, falls einer der folgenden Fehler aufgetreten ist:

- Ungültige Verwendung: `javascriptOrFileURL` wurde als `null`-Wert oder als leerer String definiert oder der Pfad zu der JS- bzw. JSF-Datei ist ungültig.
- I/O-Dateifehler: Fireworks konnte keine Antwortdatei erstellen, da die Festplatte voll ist.
- Fehlermeldung an Dreamweaver: Benutzer führt keine gültige Version von Dreamweaver (Version 3 oder höher) aus.
- Fehler beim Starten des Fireworks-Prozesses: Die Funktion startet keine gültige Version von Fireworks (Version 3 oder höher erforderlich).
- Die Aktion wurde vom Benutzer abgebrochen.

FWLaunch.getJsResponse()

Verfügbarkeit
Dreamweaver 3, Fireworks 3.

Beschreibung
Diese Funktion bestimmt, ob Fireworks noch immer den von `FWLaunch.execJsInFireworks()` übergebenen JavaScript-Code ausführt, ob das Skript erfolgreich beendet wurde oder ob ein Fehler aufgetreten ist.

Argumente
`progressTrackerCookie`

- Das Argument `progressTrackerCookie` ist das von der Funktion `FWLaunch.execJsInFireworks()` zurückgegebene Cookie-Objekt.

Rückgabewerte
Ein String mit dem Ergebnis des an die Funktion `FWLaunch.execJsInFireworks()` übergebenen Skripts, wenn der Vorgang erfolgreich beendet wurde. Der Wert ist `null`, wenn Fireworks den JavaScript-Code noch ausführt. Ein Fehlercode ungleich Null wird zurückgegeben, wenn einer der folgenden Fehler aufgetreten ist:

- Ungültige Verwendung: Beim Ausführen des Skripts ist ein JavaScript-Fehler aufgetreten.

- I/O-Dateifehler: Fireworks konnte keine Antwortdatei erstellen, da die Festplatte voll ist.
- Fehlermeldung an Dreamweaver: Benutzer führt keine gültige Version von Dreamweaver (Version 3 oder höher) aus.
- Fehler beim Starten des Fireworks-Prozesses: Die Funktion startet keine gültige Version von Fireworks (Version 3 oder höher erforderlich).
- Die Aktion wurde vom Benutzer abgebrochen.

Beispiel

Der folgende Code übergibt den String `"prompt('Please enter your name:')"` an `FWLaunch.execJsInFireworks()` und überprüft das Ergebnis:

```
var progressCookie = FWLaunch.execJsInFireworks("prompt('Please enter your name:')");
var doneFlag = false;
while (!doneFlag){
    // check for completion every 1/2 second
    setTimeout('checkForCompletion()',500);
}
function checkForCompletion(){
    if (progressCookie != null) {
        var response = FWLaunch.getJsResponse(progressCookie);
        if (response != null) {
            if (typeof(response) == "number") {
                // error or user-cancel, time to close the window
                // and let the user know we got an error
                window.close();
                alert("An error occurred.");
            }else{
                // got a valid response!
                alert("Nice to meet you, " + response);
                window.close();
            }
                doneFlag = true;
        }
    }
}
```

FWLaunch.mayLaunchFireworks()

Verfügbarkeit

Dreamweaver 2, Fireworks 2.

Beschreibung

Diese Funktion bestimmt, ob der Start einer Fireworks-Optimierungssitzung möglich ist.

Argumente

Keine.

Rückgabewerte

Ein boolescher Wert, der angibt, ob Windows oder Macintosh als Plattform verwendet wird. Bei Macintosh wird angezeigt, ob bereits eine andere Fireworks-Optimierungssitzung ausgeführt wird.

FWLaunch.optimizeInFireworks()

Verfügbarkeit
Dreamweaver 2, Fireworks 2.

Beschreibung
Diese Funktion startet für das angegebene Bild eine Fireworks-Optimierungssitzung.

Argumente
docURL, imageURL, {targetWidth}, {targetHeight}

- Das Argument *docURL* ist der Pfad zum aktiven Dokument im URL-Format „file://".
- *imageURL* ist der Pfad zum ausgewählten Bild. Wenn der Pfad relativ ausgedrückt ist, ist er relativ zu dem in *docURL* angegebenen Pfad.
- Das optionale Argument *targetWidth* ist die Breite, an die das Bild angepasst werden soll.
- Das optionale Argument *targetHeight* ist die Höhe, an die das Bild angepasst werden soll.

Rückgabewerte
Null, wenn für das ausgewählte Bild eine Fireworks-Optimierungssitzung erfolgreich gestartet wurde. Ein Fehlercode ungleich Null wird zurückgegeben, wenn einer der folgenden Fehler aufgetreten ist:

- Ungültige Verwendung: *docURL*, *imageURL* oder beide wurden als Wert `null` bzw. als leerer String definiert.
- I/O-Dateifehler: Fireworks konnte keine Antwortdatei erstellen, da die Festplatte voll ist.
- Fehlermeldung an Dreamweaver: Benutzer führt keine gültige Version von Dreamweaver (Version 2 oder höher) aus.
- Fehler beim Starten des Fireworks-Prozesses: Die Funktion startet keine gültige Version von Fireworks (Version 2 oder höher erforderlich).
- Die Aktion wurde vom Benutzer abgebrochen.

FWLaunch.validateFireworks()

Verfügbarkeit
Dreamweaver 2, Fireworks 2.

Beschreibung
Diese Funktion sucht auf der Festplatte des Benutzers nach der angegebenen Version von Fireworks.

Argumente
{versionNumber}

- Das Argument *versionNumber* ist ein optionaler Gleitkommawert, der mindestens 2 beträgt. Er stellt die erforderliche Fireworks-Version dar. Bei fehlendem Argument wird der Wert 2 als Standard verwendet.

Rückgabewerte
Ein boolescher Wert, der angibt, ob die betreffende Version von Fireworks gefunden wurde.

Beispiel

Mit dem folgenden Code wird überprüft, ob Fireworks installiert ist:

```
if (FWLaunch.validateFireworks(6.0)){
    alert("Fireworks 6.0 or later is installed.");
}else{
    alert("Fireworks 6.0 is not installed.");
}
```

Einfaches FWLaunch-Kommunikationsbeispiel

Mit dem folgenden Befehl wird Fireworks angewiesen, den Benutzer nach seinem Namen zu fragen und diesen Namen an Dreamweaver zu übergeben:

```
<html>
<head>
<title>Prompt in Fireworks</title>
<meta http-equiv="Content-Type" content="text/html; ¬
charset=iso-8859-1">
<script>

function commandButtons(){
    return new Array("Prompt", "promptInFireworks()", "Cancel", ¬
    "readyToCancel()", "Close","window.close()");
}

var gCancelClicked = false;
var gProgressTrackerCookie = null;

function readyToCancel() {
    gCancelClicked = true;
}

function promptInFireworks() {
    var isFireworks3 = FWLaunch.validateFireworks(3.0);
    if (!isFireworks3) {
        alert("You must have Fireworks 3.0 or later to use this ¬
        command");
    return;
}

    // Tell Fireworks to execute the prompt() method.
    gProgressTrackerCookie = FWLaunch.execJsInFireworks¬
    ("prompt('Please enter your name:')");

    // null means it wasn't launched, a number means an error code
    if (gProgressTrackerCookie == null || ¬
    typeof(gProgressTrackerCookie) == "number") {
        window.close();
        alert("an error occurred");
        gProgressTrackerCookie = null;
    } else {
        // bring Fireworks to the front
        FWLaunch.bringFWToFront();
        // start the checking to see if Fireworks is done yet
        checkOneMoreTime();
    }
```

```
}
function checkOneMoreTime() {
    // Call checkJsResponse() every 1/2 second to see if Fireworks
    // is done yet
    window.setTimeout("checkJsResponse();", 500);
}

function checkJsResponse() {
    var response = null;

    // The user clicked the cancel button, close the window
    if (gCancelClicked) {
        window.close();
        alert("cancel clicked");
    } else {
        // We're still going, ask Fireworks how it's doing
        if (gProgressTrackerCookie != null)
            response = FWLaunch.getJsResponse(gProgressTrackerCookie);

        if (response == null) {
            // still waiting for a response, call us again in 1/2 a
            // second
            checkOneMoreTime();
        } else if (typeof(response) == "number") {
            // if the response was a number, it means an error occurred
            // the user cancelled in Fireworks
            window.close();
            alert("an error occurred.");

        } else {
            // got a valid response!This return value might not
            // always be a useful one, since not all functions in
            // Fireworks return a string, but we know this one does,
            // so we can show the user what we got.
            window.close();
            FWLaunch.bringDWToFront();// bring Dreamweaver to the front
            alert("Nice to meet you, " + response + "!");
        }
    }
}
</script>
</head>
<body>
<form>
<table width="313" nowrap>
<tr>
<td>This command asks Fireworks to execute the prompt() ¬
function. When you click Prompt, Fireworks comes forward and ¬
asks you to enter a value into a dialog box. That value is then ¬
returned to Dreamweaver and displayed in an alert.</td>
</tr>
</table>
</form>
</body>
</html>
```

Kapitel 6: Flash-Integration

Adobe® Dreamweaver® unterstützt die API für Flash-Objekte, die mithilfe der Flash Generator-Vorlagendatei neue Flash-Objekte erstellt. Im Folgenden wird unter „API für Flash-Objekte“ ausführlich erläutert, wie Sie aus Flash Generator-Vorlagen (SWT-Dateien) Flash-Objekte (SWF-Dateien) erstellen.

Informationen zum Hinzufügen von Flash-Inhalt zu Dreamweaver-Objekten oder -Befehlen finden Sie im Handbuch *Dreamweaver erweitern*.

API für Flash-Objekte

Die API für Flash-Objekte dient Entwicklern von Erweiterungen zum Erzeugen von Objekten, die mithilfe von Flash Generator einfache SWF-Dateien erstellen. Mit dieser API können Sie Parameter in einer Flash Generator-Vorlage festlegen und als SWF- bzw. Bilddatei ausgeben. Des Weiteren können Sie mit der API neue Flash-Objekte erstellen sowie vorhandene Flash-Objekte lesen und bearbeiten.

Die SWT-Datei ist eine Flash Generator-Vorlagendatei, in der sich alle erforderlichen Daten für das Erstellen einer Flash-Objektdatei befinden. Mit diesen API-Funktionen können Sie eine SWF-Datei (oder Bilddatei) aus einer SWT-Datei erstellen. Die SWF-Datei entsteht durch Ersetzen der Parameter der SWT-Datei durch echte Werte. Weitere Informationen über Flash finden Sie in der Flash-Dokumentation. Die folgenden Funktionen stellen Methoden des `SWFFile`-Objekts dar.

SWFFile.createFile()

Beschreibung
Diese Funktion generiert eine neue Flash-Objektdatei mit der angegebenen Vorlage und einem Parameter-Array. Außerdem erstellt sie GIF-, PNG-, JPEG- und MOV-Versionen des Titels, sofern für diese Formate Dateinamen angegeben wurden.

Wenn Sie einen optionalen Parameter angeben möchten, der auf andere optionale Parameter folgt, die Sie jedoch nicht verwenden möchten, müssen Sie für die nicht zu verwendenden Parameter leere Strings angeben. Angenommen, Sie möchten eine PNG-Datei erstellen, jedoch keine GIF-Datei. In diesem Fall muss vor dem PNG-Dateinamen ein leerer String stehen.

Argumente
templateFile, templateParams, swfFileName, {gifFileName}, {pngFileName}, {jpgFileName}, {movFileName}, {generatorParams}

- Das Argument *templateFile* ist ein Pfad zu einer Vorlagendatei im URL-Format „file:// “. Hierbei kann es sich um eine SWT-Datei handeln.
- Das Argument *templateParams* ist ein Array aus Name-Wert-Paaren. Die Namen stehen hierbei für die Parameternamen in der SWT-Datei und die Werte für die gewünschten Werte dieser Parameter. Damit Dreamweaver eine SWF-Datei als Flash-Objekt erkennt, muss der erste Parameter `"dwType"` lauten. Der Wert muss ein String sein, der den Namen des Objekttyps repräsentiert, beispielsweise `"Flash Text"`.
- Das Argument *swfFileName* ist der Name einer SWF-Ausgabedatei im URL-Format „file://“ bzw. ein leerer String, der ignoriert wird.

- Das Argument *gifFileName* ist der Name einer GIF-Ausgabedatei im URL-Format „file:///“. Dieses Argument ist optional.
- Das Argument *pngFileName* ist der Name einer PNG-Ausgabedatei im URL-Format „file:///“. Dieses Argument ist optional.
- Das Argument *jpgFileName* ist der Name einer JPEG-Ausgabedatei im URL-Format „file:///“. Dieses Argument ist optional.
- Das Argument *movFileName* ist der Name einer QuickTime-Ausgabedatei im URL-Format „file:///“. Dieses Argument ist optional.
- Das Argument *generatorParams* ist ein String-Array, das optionale Befehlszeilen-Flags für Flash Generator angibt. Dieses Argument ist optional. Die Datenelemente für jedes Flag müssen auf dieses Argument folgen. Die folgende Tabelle enthält eine Beschreibung der gängigsten Flags.

Options-Flag	Daten	Beschreibung	Beispiel
`-defaultsize`	Breite, Höhe	Stellt die Größe des Ausgabebilds auf die angegebene Breite und Höhe ein.	`"-defaultsize", "640", "480"`
`-exactFit`	Keine.	Streckt den Inhalt des Ausgabebilds, um es genau an die festgelegte Ausgabegröße anzupassen.	`"-exactFit"`

Rückgabewerte

Ein String, der einen der folgenden Werte enthält:

- `"noError"` bedeutet, dass der Aufruf erfolgreich war.
- `"invalidTemplateFile"` bedeutet, dass die angegebene Vorlagendatei ungültig war oder nicht gefunden wurde.
- `"invalidOutputFile"` bedeutet, dass mindestens einer der angegebenen Ausgabedateinamen ungültig ist.
- `"invalidData"` bedeutet, dass mindestens eines der Name-Wert-Paare des Arguments *templateParams* ungültig ist.
- `"initGeneratorFailed"` bedeutet, dass Flash Generator nicht initialisiert werden konnte.
- `"outOfMemory"` bedeutet, dass für den Vorgang nicht genügend Speicher zur Verfügung stand.
- `"unknownError"` bedeutet, dass ein unbekannter Fehler aufgetreten ist.

Beispiel

Mit dem folgenden JavaScript wird eine Flash-Objektdatei vom Typ `"myType"` erstellt, die jeden `"text"`-String in der Vorlagendatei durch den String `"Hello World"` ersetzt. Dabei werden eine GIF- und eine SWF-Datei erstellt.

```
var params = new Array;
params[0] = "dwType";
params[1] = "myType";
params[2] = "text";
params[3] = "Hello World";
errorString = SWFFile.createFile( "file:///MyMac/test.swt", ¬
params, "file:///MyMac/test.swf", "file:///MyMac/test.gif");
```

SWFFile.getNaturalSize()

Beschreibung

Diese Funktion gibt die natürliche Größe eines unkomprimierten Flash-Inhalts zurück.

Argumente
fileName

- Das Argument *fileName* ist der Pfad zum Flash-Inhalt im URL-Format „file://“.

Rückgabewerte
Ein Array mit zwei Elementen, die die Breite und Höhe einer unkomprimierten SWF-Datei angeben bzw. den Wert `null`, wenn die Datei keine unkomprimierte SWF-Datei ist.

SWFFile.getObjectType()

Beschreibung
Diese Funktion gibt den Flash-Objekttyp zurück, d. h. den Wert, der im Parameter `dwType` übergeben wurde, als die Datei durch die Funktion `SWFFile.createFile()` erstellt wurde.

Argumente
fileName

- Das Argument *fileName* ist der Pfad zu einer Flash-Objektdatei im URL-Format „file://“. Hierbei handelt es sich normalerweise um eine SWF-Datei.

Rückgabewerte
Ein String, der den Objekttyp repräsentiert, oder `null`, wenn die Datei kein Flash-Objekt ist oder nicht gefunden werden konnte.

Beispiel
Mit dem folgenden Code wird geprüft, ob die Datei „test.swf“ ein Flash-Objekt des Typs `myType` ist:

```
if ( SWFFile.getObjectType("file:///MyMac/test.swf") == "myType" ){
    alert ("This is a myType object.");
}else{
    alert ("This is not a myType object.");
}
```

SWFFile.readFile()

Beschreibung
Mit dieser Funktion wird eine Flash-Objektdatei gelesen.

Argumente
fileName

- Das Argument *fileName* ist der Pfad zu einer Flash-Objektdatei im URL-Format „file://“.

Rückgabewerte
Ein String-Array, bei dem das erste Array-Element den vollständigen Pfad zur SWT-Vorlagendatei angibt. Die folgenden Strings geben die Parameter (Name-Wert-Paare) für das Objekt an. Im Array befindet sich hinter jedem Namen der entsprechende Wert. Das erste Name-Wert-Paar ist `"dwType"` und dahinter steht der zugehörige Wert. Die Funktion gibt den Wert `null` zurück, wenn die Datei nicht gefunden werden kann oder wenn sie keine Flash-Objektdatei ist.

Beispiel
Durch Aufrufen von `var params = SWFFile.readFile("file:///MyMac/test.swf")` werden die folgenden Werte im Parameter-Array zurückgegeben:

```
"file:///MyMac/test.swt"     // template file used to create this .swf file
"dwType"                     // first parameter
"myType"                     // first parameter value
"text"                       // second parameter
"Hello World"                  // second parameter value
```

Funktionen für Flash-Bedienfelder und Flash-Dialogfelder

Mit den folgenden API-Funktionen können Sie Bedienfeldern und Dialogfeldern SWF-Dateien hinzufügen.

dreamweaver.flash.newControl()

Verfügbarkeit
Dreamweaver CS4.

Beschreibung
Mit dieser Funktion können Sie ein Flash-Steuerelement erstellen. Später wird darauf mithilfe des Parameters *`controlID`* Bezug genommen. Das Steuerelement zeigt die mit dem SWF-Pfad angegebene Flash-Datei (.swf) an. Die Position und Größe des Steuerelements ist im Parameter *`defaultGeometry`* angegeben.

Hinweis: *Dreamweaver zeigt Flash-Steuerelemente an, wenn Sie `flash.requestStateChange` aufrufen. Dreamweaver zeigt Steuerelemente von Dialogfeldern an, wenn Sie `newControl` aufrufen. Sie müssen dazu nicht `flash.requestStateChange` aufrufen.*

Argumente
`controlID`, *`controlType`*, *`controlData`*

- Das Argument *`controlID`* ist ein String.
- Das Argument *`controlType`* gibt an, ob das Bedienfeld eine Standarderweiterung ("standard"), eine vertrauenswürdige Standarderweiterung ("trusted") oder eine zusätzliche Erweiterung (jeder andere Wert) ist. Wenn es sich um eine zusätzliche Erweiterung handelt, ist der Wert eine spezielle ID für die Hostanwendung, die den Typ der erforderlichen benutzerdefinierten Integration angibt. Wenn der Anwendung dieser benutzerdefinierte Integrationstyp unbekannt ist, wird ein Fehler zurückgegeben.
- Das Argument *`controlData`* ist ein Objekt. Es folgen einige der wichtigsten Eigenschaften dieses Arguments:

Eigenschaft	Beschreibung	Werte
`controlData.swfUTF8Path`	Speicherort der SWF-Datei. Diese Eigenschaft ist erforderlich und wird als Unicode-String übergeben, da in JavaScript alle Zeichen das Unicode-Format verwenden.	Mögliche Werte für `controlData.windowType`: • `PanelWindow`. In der auf diese Tabelle folgenden Tabelle sind die Spezifikationen für diesen Wert aufgeführt. • `ModalDialogWindow`
`{controlData.scriptPath}`	Pfad zu der JS-Datei mit den Funktionen, die per Aufruf der External-Schnittstelle aus der SWF-Datei ausgeführt werden. Diese Eigenschaft ist optional. Wenn Sie über die External-Schnittstelle einen Rückruf aus der SWF-Datei in JavaScript-Code von Dreamweaver durchführen möchten, können Sie eine JS-Datei mit Funktionen angeben, die dann aus der SWF-Datei aufgerufen werden können. Weitere Informationen finden Sie im Abschnitt zum Aufruf von `dw.flash.executeScript`.	
`controlData.defaultGeometry`	Die Werte von `defaultGeometry` sind als Bildschirmkoordinaten mit Koordinatenursprung in der linken oberen Ecke des Bildschirms angegeben. Diese Eigenschaft ist erforderlich. `Object /*!< default creation geometry, including positioning */ { topleftx: Number, toplefty: Number, width: Number, height: Number }`	

In der folgenden Tabelle sind die Spezifikationen für `PanelWindow` aufgeführt:

Optionen	Typ	Beschreibung
`name`	String	Der auf der Registerkarte angezeigte Name des Bedienfelds. Wenn kein Name angegeben ist, wird „NICHT DEFINIERT“ zugewiesen. Alle Bedienfeldnamen werden in Großbuchstaben dargestellt. Sie können sie nicht in Kleinbuchstaben angeben.
`{controlData.minSize}`	Objekt	***`minSize`*** gilt nur für Steuerelemente des Typs `PanelWindow`. Mit dieser Option wird die Mindestgröße des Bedienfelds festgelegt, die bei Größenänderungen nicht unterschritten werden kann. Diese Option ist optional. Wenn ***`minSize`*** nicht angegeben ist, gelten die in `defaultGeometry` angegebenen Standardwerte für Höhe und Breite und die Größe des Bedienfelds kann nicht geändert werden. `{ width: Number, height: Number }`
`{controlData.maxSize}`	Objekt	***`Die maxSize-Eigenschaft gibt es nur für Steuerelemente des Typs „PanelWindow".`*** Diese Option ist optional. Mit dieser Option wird die Höchstgröße des Bedienfelds festgelegt, die bei Größenänderungen nicht überschritten werden kann. Wenn „maxSize“ nicht angegeben ist, gelten die in `defaultGeometry` angegebenen Standardwerte für Höhe und Breite und die Größe des Bedienfelds kann nicht geändert werden. `{ width: Number, height: Number }`

Optionen	Typ	Beschreibung
`{iconPathNormal}`	String	Pfad zu dem Symbol, das im schwebenden Bedienfeld verwendet wird, wenn das Bedienfeld in den Symbolmodus minimiert wurde. Diese Option ist optional.
`{iconPathRollOver}`	String	Pfad zu dem Symbol, das im schwebenden Bedienfeld verwendet wird, wenn das Bedienfeld in den Symbolmodus minimiert wurde und der Benutzer mit dem Mauszeiger darauf zeigt. Diese Option ist optional.
`{iconPathDisable}`	String	Pfad zu dem Symbol, das im schwebenden Bedienfeld verwendet wird, wenn das Bedienfeld in den Symbolmodus minimiert wurde und deaktiviert ist. Diese Option ist optional.

Rückgabewerte

Einer der folgenden Erfolgs- oder Fehlercodes:

- Der Code `PlugPlugErrorCode_success` gibt an, dass das Steuerelement erfolgreich erstellt wurde.
- Der Code `PlugPlugErrorCode_extensionRegistrationFailed` gibt an, dass das Steuerelement nicht registriert werden konnte.

dreamweaver.flash.requestStateChange()

Verfügbarkeit

Dreamweaver CS4.

Beschreibung

Mit dieser Funktion wird der Status des mit *`uniqueID`* identifizierten schwebenden Bedienfelds für die Erweiterung mit *`extensionID`* geändert.

Argumente

`controlID`, *`stateChange`*, *`stateData`*

- Das Argument *`controlID`* ist ein String.
- Das Argument *`stateChange`* ist ein String mit den folgenden möglichen Werten:

Wert	Beschreibung
`Move`	Änderung des Ursprungs, jedoch nicht der Größe
`Resize`	Neue Größe und ggf. neuer Ursprung
`Show`	Nur Sichtbarkeit, jedoch keine Änderung der Geometrie
`Hide`	Nur Sichtbarkeit, jedoch keine Änderung der Geometrie
`Minimize`	Wie „Hide“, jedoch mit Angabe eines Grunds
`Restore`	Wie „Show“, jedoch mit Angabe eines Grunds
`Open`	Fenster wird erstellt und die entsprechende Erweiterung geladen
`Close`	Enthaltende Erweiterung wird entladen

- Die Werte des Arguments *`stateData`* sind Strings wie in der folgenden Tabelle aufgeführt:

Wert von `stateChange`	Wert von `stateData`
`Move`	`eventData = { topleftx: Number, toplefty: Number }`
`Resize`	`eventData = { width: Number, height: Number }`

Rückgabewerte

Die folgende Tabelle enthält die Rückgabewerte (Strings):

Wert	Beschreibung
`RequestPosted`	Ein Ereignis oder Befehl zum Ausführen der Anforderung wurde in die Warteschlange der Hostanwendung eingefügt.
`RequestComplete`	Die Hostanwendung hat die Anforderung erfolgreich beendet.
`RequestFailed`	Die Hostanwendung hat versucht, die Anforderung auszuführen, dies ist jedoch fehlgeschlagen.
`RequestDenied`	Die Hostanwendung hat die Anforderung abgelehnt, in der Regel weil die angeforderte Aktion nicht unterstützt wird.

Beispiel

```
controlData = {};
controlData.defaultGeometry = {topleftx : 100, toplefty : 100, width : 200, height : 200 };
controlData.minSize = {width : 100; height : 100 };
controlData.maxSize = {width : 300; height : 300 };
var swfPath = dw.getConfigurationPath();
swfPath += '/flash/PhotoAlbum.swf';
controlData.swfUTF8Path = swfPath;
// open the window
flash.requestStateChange("com.adobe.extension.foo","Open",controlData.defaultGeometry);
```

dreamweaver.flash.controlEvent()

Verfügbarkeit

Dreamweaver CS4.

Beschreibung

Diese Funktion dient zum Übergeben von Ereignissen an ein Flash-Steuerelement. Ereignisaufrufe werden als XML-String übergeben, der die Funktion und die benötigten Parameter enthält. Der XML-String enthält die in der SWF-Datei zu startende Funktion.

Argumente

`inControlID`, *`inXMLString`*

- Das Argument *`inControlID`* ist ein String.
- Das Argument *`inXMLString`* ist ein String. Übergeben Sie den folgenden *`inXMLString`*, um die Funktion in der Flash-Datei „flashCallback“ aufzurufen und als Argument den einzelnen String „Hello“ zu übergeben.

```
<invoke name="flashCallback" returntype="xml">
    <arguments>
        <string>Hello</string>
    </arguments>
</invoke>
```

Rückgabewerte
Gibt einen XML-String zurück.

Beispiel
Im folgenden Beispiel wird die Funktion `flashCallback` aus JavaScript aufgerufen. In diesem Beispiel übergeben Sie den Namen der Callback-Funktion und die entsprechenden Argumente als XML-String.

```
var xmlString = '<invoke name="flashCallback" returntype="xml">
<arguments>
<string>Hello</string>
</arguments>
</invoke>';
```

In diesem Beispiel verwenden Sie `dw.flash.controlEvent`, um einen Callback-Aufruf in die Flash-Datei (.swf) durchzuführen:

```
dw.flash.controlEvent('Flickr', xmlString);
```

In dieser Funktion werden folgende Argumente verwendet:

- `Flickr` – ID der Erweiterung, die beim Erstellen des SWF-Steuerelements mit `dw.flash.newControl` übergeben wurde.
- XML-String mit der Callback-Funktion und den Argumenten.

Das folgende Beispiel ist die Implementierung der Funktion `flashcallback` in „flashcallback.mxml“. In diesem Beispiel fügen Sie die Funktion `flashcallback` hinzu. Diese Funktion muss von externen Anwendungen aufgerufen werden.

```
public function initApp():void {
ExternalInterface.addCallback("flashCallback",flashCallback);
}
```

Diese Funktion wird per Callback von außerhalb der Flash-Datei (.swf) aufgerufen.

Hinweis: *Stellen Sie sicher, dass diese Funktion erst aufgerufen wird, nachdem* `ExternalInterface.addCallback ("flashCallback",flashCallback)` *aufgerufen wurde.*

```
public function flashCallback(inputStr:String):String
{
    out.text += inputStr + " got flashCallback!\n";
    return "it worked!";
}
```

dreamweaver.flash.setMenu()

Verfügbarkeit
Dreamweaver CS4.

Beschreibung
Mit dieser Funktion können Sie Ausklappbefehle für Erweiterungen des Typs „PanelWindow“ erstellen.

Argumente

inControlID, *inMenuPosition*, *inMenu*

- *inControlID* ist eine Erweiterungs-ID. Der Aufruf der Funktion wirkt sich auf das Ausklappmenü eines geöffneten Bedienfelds aus, das die Erweiterung enthält. Wenn dieses Argument nicht definiert ist, werden durch den Aufruf die Hauptmenüs der Anwendung beeinflusst.
- *inMenuPosition* ist ein String, der angibt, an welcher Position die angegebenen Befehle eingefügt werden sollen.
  - Wenn dieser String nicht definiert ist, wird ein ganzes Menü ersetzt.
  - Wenn dieser String für ein Bedienfeld bestimmt ist, wird der gesamte benutzerdefinierbare Bereich des Ausklappmenüs ersetzt. (Für die Anwendung sind einige Ausklappmenüeinträge fest reserviert.)
  - Wenn dieser String für eine Anwendung bestimmt ist, wird das gesamte Untermenü „Steuerelemente“ des Menüs „Fenster“ ersetzt.
  - Für den Fall, dass dieser String ein XML-String in einem noch zu bezeichnenden Schema zum Festlegen von Menübereichen ist, steht für zukünftige Kompatibilität das folgende Formular zur Verfügung.
- *inMenu* entspricht `MenuItem`. Dieses Argument gibt eine Liste von Befehlen an, die an der bezeichneten Menüposition eingefügt werden. Dabei werden alle bei einem früheren Aufruf an dieser Position eingefügten Elemente entfernt.

Rückgabewerte

Einer der folgenden Erfolgs- oder Fehlercodes:

- Der Code `PlugPlugErrorCode_success` gibt einen erfolgreichen Abschluss an.
- Der Code `PlugPlugErrorCode_extensionMenuCreationFailed` gibt an, dass die Erstellung des Erweiterungsmenüs fehlgeschlagen ist.
- Der Code `PlugPlugErrorCode_unknown` gibt an, dass die Funktion aus unbekannten Gründen fehlgeschlagen ist.

Beispiel

Das folgende Beispiel wird zum Einrichten des Menüs verwendet:

```
function initializeMenuItem(menuID, menuName,extensionID, submenu)
{
    var menuItem = {};
    menuItem.menuId = menuID; //!< unique menu ID, if NULL menu is disabled
    menuItem.nameUtf8 = menuName; //!< Item title, if "---" item is a separator
    menuItem.extensionId = extensionID; //!< optional extension ID, used for panels only
    menuItem.submenu = submenu; //!< if non-NULL, this is a submenu
    return menuItem;
}
function setupMenu()
{
    var menuItems = new Array();
    menuItems.push(initializeMenuItem('id1','Call .swf
                    ActionScript',undefined,undefined));
    menuItems.push(initializeMenuItem('id0','---',undefined,undefined));
    menuItems.push(initializeMenuItem('id2','Call Dw JavaScript',undefined,undefined));
    dw.flash.setMenu('Flickr',controlID,menuItems);
}
```

Hinweis: *Geben Sie eine Funktion mit dem Namen „`onSelectMenuItem`“ in der JavaScript-Datei an, die in der Eigenschaft `scriptPath` des Objekts festgelegt ist, das an „newControl“ übergeben wurde.*

Bei *onSelectMenuItem* handelt es sich um eine Menüelementprozedur. Sie wird mit der zugehörigen Menü-ID aufgerufen, wenn im Ausklappmenü des schwebenden Bedienfelds ein Befehl ausgewählt wurde.

Im folgenden Beispiel wird die Definition der Callback-Prozedur in „Configuration/flash/Flickr.js“ beschrieben:

```
       function onSelectMenuItem(menuID)
{
    if (menuID == 'id1') {
                              var flashCallbackString = '<invoke name= " flash Callback"
                              returntype="xml">
                              <arguments><string>Hello</string></arguments></invoke>';
                              dw.flash.control Event('Flickr', flashCallbackString);
                                     return("PlugPlugRequestCompleted");
                         } else {
                                 alert ( ' You selected: menuID = ' + menuID);
                                        return ( " PlugPlugRequestCompleted");
                                 }
}
```

dreamweaver.flash.evalScript()

Verfügbarkeit
Dreamweaver CS4.

Beschreibung
Mit dieser Funktion wird eine JavaScript-Funktion für einen der folgenden Zwecke aufgerufen:

- Ausführen einer JavaScript-Funktion, die in der dieser Erweiterung (für CSXS-Erweiterungen) zugeordneten Skriptdatei definiert ist.
- Für nicht CSXS-basierte Erweiterungen die im Parameter *scriptPath* definierte JS-Datei.

Argumente
controlID, *javascript function call*

- Das Argument *controlID* ist die ID der Erweiterung, die den Skriptcode ausführen soll. Diese ID muss mit der ID übereinstimmen, die als erster Parameter für `dw.flash.newControl()` angegeben ist.
- Das Argument für den JavaScript-Funktionsaufruf ermöglicht das Aufrufen von Funktionen mit einer beliebigen Anzahl von Parametern.

Rückgabewerte
Ein boolescher Wert: `true`, wenn die Funktion erfolgreich ausgeführt wurde, andernfalls `false`.

dreamweaver.flash.executeScript()

Verfügbarkeit
Dreamweaver CS4.

Beschreibung

Die Funktion dient zum Ausführen von Funktionen in JS-Dateien. Der ActionScript-Code in der SWF-Datei startet die Funktion `dreamweaver.flash.executeScript()`.

Argumente

`javascript function call`

Hinweis: *Geben Sie einen Pfad zur JS-Datei an, die die aufzurufenden Funktionen enthält.*

Rückgabewerte

Ein XML-String, der in ein ActionScript-Objekt serialisiert wird.

Beispiel

Das folgende Beispiel enthält eine Beispieldatei („Sample.mxml“) und eine JavaScript-Funktion in einer JavaScript-Datei („Sample.js“).

```
private function executeScript():void
    {
        if(ExternalInterface.available)
        {
            out.text += "SwfCalledHost\n";
            var scriptText:String = "helloWorld('scott');\n";
            var resultStr:Object =
                ExternalInterface.call("dw.flash.executeScript",scriptText);
            out.text += "Result: " + resultStr.strResult + '\n';
        }
}
```

Die folgende JavaScript-Datei enthält die JavaScript-Funktion `helloWorld()`, die aus der SWF-Datei aufgerufen wird. Diese Funktion verwendet den Aufruf von `dw.getAppLanguage()`, um einen fünfbuchstabigen Sprachcode zurückzugeben, der in „Sample.js“ eingesetzt wird.

```
function helloWorld(nameStr)
{
    alert('hello ' + nameStr);
    var appLanguage = dw.getAppLanguage();
    var returnStr = '<object><property id="strResult"><string>Language: ' + appLanguage
            + '</string></property></object>'
    alert(returnStr);
    return (returnStr);
}
```

Verwandte Themen

„dreamweaver.flash.newControl()“ auf Seite 42

dreamweaver.flash.controlExists

Verfügbarkeit

Dreamweaver CS4.

Beschreibung

Mit dieser Funktion wird überprüft, ob die Steuerelemente vorhanden sind. `PanelWindow-Steuerelemente werden zwischen den Starts von Dreamweaver gespeichert.`

Argumente

`controlID`

Rückgabewerte

Ein boolescher Wert: `true`, wenn das Steuerelement bereits erstellt wurde, andernfalls `false`.

Kapitel 7: Photoshop-Integration

Adobe® Dreamweaver CS5® ermöglicht eine enge Integration in Adobe® Photoshop®. Benutzer können Photoshop-Bilder in Dreamweaver als Smart Objekte einfügen. Mit Smart Objekten werden Bilder in Dreamweaver automatisch aktualisiert, wenn mit Photoshop Änderungen an den Originalbildern vorgenommen werden.

Funktionsweise von Smart Objekten

Photoshop-Bilder werden als Smart Objekte in Dreamweaver eingefügt. Die Smart Objekte bleiben mit den Photoshop-Originalbildern verknüpft. Wenn Benutzer ein Bild in Photoshop bearbeiten, wird in Dreamweaver ein aktualisiertes Bild angezeigt. Ein Smart Objekt hat einen spezifischen Status, der hauptsächlich auf der Verbindung des Webbilds mit der ursprünglichen Bilddatei beruht. Benutzern wird der Status eines Smart Objekts grafisch dargestellt. Der synchronisierte Zustand des Smart Objekts wird durch ein entsprechendes Symbol angezeigt.

API für Smart Objekte

Mit den Funktionen für Smart Objekte werden Vorgänge der Dreamweaver- und Photoshop-Integration durchgeführt. Mit den Funktionen können Sie folgende Aufgaben ausführen:

- Ermitteln des aktuellen Bildstatus
- Abrufen der Höhe und Breite von Bildern

dreamweaver.assetPalette.canUpdateSmartObjectFromOriginal()

Verfügbarkeit
Dreamweaver CS4.

Beschreibung
Enabler: Mit dieser Funktion wird überprüft, ob im Bedienfeld „Elemente“ ein Smart Objekt ausgewählt ist, auf das der Befehl „Von Original aktualisieren“ angewendet werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn „Von Original aktualisieren“ auf die aktuelle Auswahl angewendet werden kann. Andernfalls `false`.

dreamweaver.assetPalette.updateSmartObjectFromOriginal()

Verfügbarkeit
Dreamweaver CS4.

Beschreibung
Diese Funktion erstellt das ausgewählte Webbild anhand des aktuellen Inhalts der verknüpften ursprünglichen Quelldatei neu.

Argumente
Keine.

Rückgabewerte
Keine.

dreamweaver.getSmartObjectState()

Verfügbarkeit
Dreamweaver CS4.

Beschreibung
Diese Funktion gibt den Status eines Webbilds hinsichtlich der Funktionalität von Smart Objekten zurück.

Argumente
Absolute lokale URL eines Webbilds.

Rückgabewerte
Der Status des Smart Objekts als Zahlenwert:

Zahlenwert	Beschreibung
`-10`	Unbekannter Fehler
`0`	Kein Smart Objekt
`1`	Mit dem Inhalt der ursprünglichen Bilddatei synchronisiert
`100`	Webbild wurde nach letzter Synchronisierung bearbeitet
`200`	Originalbild wurde nach letzter Synchronisierung bearbeitet
`+2`	Abmessungen des Originalbilds unterscheiden sich von den Attributen für Höhe und Breite im HTML-Code
`+4`	Abmessungen des Webbilds unterscheiden sich von den Attributen für Höhe und Breite im HTML-Code
`10`	Zugriff auf die ursprüngliche Bilddatei nicht möglich
`20`	Zugriff auf die Webbilddatei nicht möglich

dreamweaver.getSmartObjectOriginalWidth()

Verfügbarkeit
Dreamweaver CS4.

Beschreibung
Diese Funktion ermittelt die Breite der ursprünglichen Bilddatei eines Smart Objekts in Pixel und gibt diesen Wert zurück.

Argumente
Absolute lokale URL des Webbilds.

Rückgabewerte
Breite der ursprünglichen Bilddatei in Pixel.

dreamweaver.getImageWidth()

Verfügbarkeit
Dreamweaver CS4.

Beschreibung
Diese Funktion ermittelt die Breite eines Bilds in Pixel und gibt diesen Wert zurück.

Argumente
Absolute lokale URL eines Webbilds.

Rückgabewerte
Breite des Bilds in Pixel.

dreamweaver.getImageHeight()

Verfügbarkeit
Dreamweaver CS4.

Beschreibung
Diese Funktion ermittelt die Höhe eines Bilds in Pixel und gibt diesen Wert zurück.

Argumente
Absolute lokale URL eines Webbilds.

Rückgabewerte
Höhe des Bilds in Pixel.

dreamweaver.resolveOriginalAssetFileURLToAbsoluteLocalFilePath()

Verfügbarkeit
Dreamweaver CS4.

Beschreibung
Diese Funktion löst den Dateipfad zu einer ursprünglichen Bilddatei auf (wie in den Design Notes gespeichert). Die Pfadangabe kann leer, relativ zur Site oder absolut sein.

Argumente
Absolute lokale URL oder zur Site relative URL für das Webbild. Diese URL ist erforderlich, um die Site aufzulösen.

Rückgabewerte
Absoluter lokaler Dateipfad.

dreamweaver.canUpdateSmartObjectFromOriginal()

Verfügbarkeit
Dreamweaver CS4.

Beschreibung
Diese Funktion dient zum Beantworten der Frage, ob ein Smart Objekt unter Verwendung der ursprünglichen Bilddatei aktualisiert werden kann.

Argumente
Status des Smart Objekts als Zahlenwert. `ImageManipulatorSettings:GetSmartObjectStatus()` gibt diesen Status zurück.

Rückgabewerte
Ein boolescher Wert: `true`, wenn für den angegebenen Status eine Aktualisierung auf Grundlage des Originalbilds durchgeführt werden kann, andernfalls `false`.

dreamweaver.updateSmartObjectFromOriginal()

Verfügbarkeit
Dreamweaver CS4.

Beschreibung
Diese Funktion aktualisiert ein Webbild auf Grundlage des Inhalts der ursprünglichen Bilddatei.

Argumente
Absolute lokale URL eines Webbilds.

Rückgabewerte
Keine.

Kapitel 8: API für Datenbanken

Mit den Funktionen der API für Datenbanken können Sie Datenbankverbindungen verwalten und auf Informationen zugreifen, die in Datenbanken gespeichert sind. Mit der API für Datenbanken können folgende Aufgaben ausgeführt werden: Verwalten von Datenbankverbindungen und Zugreifen auf Datenbankverbindungen.

Die Funktionen der API für Datenbanken werden während der Entwurfsphase ausgeführt, in der Benutzer ihre Webanwendungen entwerfen, und nicht zur Laufzeit, nachdem die Webanwendungen bereitgestellt wurden.

Sie können diese Funktionen in allen Erweiterungen verwenden. Auch die APIs für Serververhalten, Datenformate und Datenquellen in Adobe® Dreamweaver® CS5 verwenden diese Datenbankfunktionen.

Funktionsweise der API für Datenbanken

Im folgenden Beispiel wird die Serververhaltensfunktion `getDynamicBindings()` für „Recordset.js“ definiert. Dabei wird die Funktion `MMDB.getColumnAndTypeList()` verwendet.

```
function getDynamicBindings(ss)
{
    var serverModel = dw.getDocumentDOM().serverModel.getServerName();
    var bindingsAndTypeArray = new Array();
    var connName=ss.connectionName;
    var statement = ss.source;
    var rsName= ss.rsName;

    // remove SQL comments
    statement = statement.replace(/\/\*[\S\s]*?\*\//g, " ");
    var bIsSimple = ParseSimpleSQL(statement);
    statement = stripCFIFSimple(statement);

    if (bIsSimple) {
    statement = RemoveWhereClause(statement,false);
    } else {
    var pa = new Array();

        if (ss.ParamArray != null) {
        for (var i = 0; i < ss.ParamArray.length; i++) {
            pa[i] = new Array();
            pa[i][0] = ss.ParamArray[i].name;
            pa[i][1] = ss.ParamArray[i].value;
        }
        }

    var statement = replaceParamsWithVals(statement, pa, serverModel);
}
    bindingsAndTypeArray = MMDB.getColumnAndTypeList(connName, statement);
    return bindingsAndTypeArray;
}
```

Funktionen für Datenbankverbindungen

Mithilfe von Datenbankverbindungsfunktionen können Sie Verbindungen herstellen und verwalten, z. B. ADO-, ColdFusion- und JDBC-Verbindungen in Dreamweaver.

Diese Funktionen bilden lediglich eine Schnittstelle mit dem Connection Manager. Über diese Funktionen wird nicht direkt auf die Datenbanken zugegriffen. Informationen zu Funktionen, mit denen auf Datenbanken zugegriffen wird, finden Sie unter „Funktionen für den Datenbankzugriff“ auf Seite 68.

Beim Verwalten der Datenbankverbindungen können Sie den Benutzernamen und das Kennwort abfragen, um verschiedene Aktionen auszuführen, beispielsweise:

- Herstellen einer Verbindung mit einer Datenbank
- Öffnen eines Dialogfelds für die Datenbankverbindung

MMDB.deleteConnection()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Diese Funktion löscht die angegebene Datenbankverbindung.

Argumente
connName

- Das Argument *connName* ist der Name der Datenbankverbindung, wie im Connection Manager angegeben. Dieses Argument gibt den Namen der zu löschenden Datenbankverbindung an.

Rückgabewerte
Keine.

Beispiel
Im folgenden Beispiel wird eine Datenbankverbindung gelöscht:

```
function clickedDelete()
{
    var selectedObj = dw.serverComponents.getSelectedNode();
    if (selectedObj && selectedObj.objectType=="Connection")
    {
        var connRec = MMDB.getConnection(selectedObj.name);
        if (connRec)
        {
            MMDB.deleteConnection(selectedObj.name);
            dw.serverComponents.refresh();
        }
    }
}
```

MMDB.getColdFusionDsnList()

Verfügbarkeit
Dreamweaver UltraDev 4.

Beschreibung
Diese Funktion ruft mithilfe der Funktionen `getRDSUserName()` und `getRDSPassword()` die ColdFusion-DSNs (Data Source Names, Datenquellennamen) vom Server der Site ab.

Argumente
Keine.

Rückgabewerte
Ein Array mit den ColdFusion-DSNs, die auf dem Server der aktuellen Site gespeichert sind.

MMDB.getConnection()

Verfügbarkeit
Dreamweaver UltraDev 4, verbessert in Dreamweaver MX.

Beschreibung
Diese Funktion ruft ein benanntes Verbindungsobjekt ab.

Argumente
name

- Das Argument *name* ist eine Stringvariable, die den Namen der Verbindung angibt, auf die Sie verweisen möchten.

Rückgabewerte
Ein Verweis auf ein benanntes Verbindungsobjekt. Verbindungsobjekte enthalten die folgenden Eigenschaften.

Eigenschaft	Beschreibung
`name`	Verbindungsname
`type`	Wenn `useHTTP` den Wert `false` hat, gibt diese Eigenschaft an, welche DLL-Datei zur Laufzeit für den Aufbau der Verbindung mit einer Datenbank verwendet werden soll.
`string`	Laufzeit-ADO-Verbindungsstring oder JDBC-URL
`dsn`	ColdFusion-DSN
`driver`	Laufzeit-JDBC-Treiber
`username`	Laufzeit-Benutzername
`password`	Laufzeit-Kennwort
`useHTTP`	Ein String, der entweder den Wert `true` oder `false` enthält und dadurch angibt, ob in der Entwurfsphase ein Remote-Treiber (HTTP-Verbindung) verwendet werden soll. Andernfalls wird ein lokaler Treiber (DLL) verwendet.
`includePattern`	Ein regulärer Ausdruck, mit dem die include-Anweisung auf der Seite in der Ansicht „Live Data“ und „Vorschau in Browser“ gesucht wird.

Eigenschaft	Beschreibung
`variables`	Ein Array der Seitenvariablennamen und ihrer Werte, die in der Ansicht „Live Data“ und „Vorschau in Browser“ verwendet werden.
`catalog`	Wird zur Einschränkung der angezeigten Metadaten verwendet (weitere Informationen unter „MMDB.getProcedures()“ auf Seite 72).
`schema`	Wird zur Einschränkung der angezeigten Metadaten verwendet (weitere Informationen unter „MMDB.getProcedures()“ auf Seite 72).
`filename`	Dateiname des Dialogfelds, das zur Erstellung der Verbindung verwendet wurde.

Hinweis: *Bei diesen Eigenschaften handelt es sich um die in Dreamweaver implementierten Standardeigenschaften. Entwickler können eigene Verbindungstypen definieren und diese Standardeigenschaften durch neue Eigenschaften ergänzen oder völlig andere Eigenschaften verwenden.*

MMDB.getConnectionList()

Verfügbarkeit
Dreamweaver UltraDev 1.

Beschreibung
Diese Funktion ruft eine Liste aller im Connection Manager definierten Verbindungsstrings ab.

Argumente
Keine.

Rückgabewerte
Ein String-Array, in dem jeder String der Name einer Verbindung entsprechend der Anzeige im Connection Manager ist.

Beispiel
Ein Aufruf von `MMDB.getConnectionList()` kann die Strings `["EmpDB", "Test", "TestEmp"]` zurückgeben.

MMDB.getConnectionName()

Verfügbarkeit
Dreamweaver UltraDev 1.

Beschreibung
Diese Funktion ruft den Verbindungsnamen ab, der dem angegebenen Verbindungsstring entspricht. Diese Funktion ist nützlich, wenn Sie aus Daten auf der Seite einen Verbindungsnamen in der Benutzeroberfläche neu auswählen möchten.

Wenn ein Verbindungsstring auf zwei verschiedene Treiber verweist, können Sie sowohl den Verbindungsstring als auch den Treiber angeben, der dem zurückzugebenden Verbindungsnamen entspricht. Es können beispielsweise zwei Verbindungen vorliegen.

- Verbindung 1 hat folgende Eigenschaften:

```
ConnectionString="jdbc:inetdae:velcro-qa-5:1433?database=pubs"
DriverName="com.inet.tds.TdsDriver"
```

- Verbindung 2 hat folgende Eigenschaften:

```
ConnectionString="jdbc:inetdae:velcro-qa-5:1433?database=pubs"
DriverName="com.inet.tds.TdsDriver2"
```

Die Verbindungsstrings für Verbindung 1 und Verbindung 2 sind identisch. Verbindung 2 stellt eine Verbindung mit einer neueren Version des Treibers `TdsDriver` her. Sie sollten den Treibernamen an diese Funktion übergeben, um den zurückzugebenden Verbindungsnamen vollständig zu bestimmen.

Argumente
connString, {driverName}

- Das Argument *connString* ist der Verbindungsstring, der den Verbindungsnamen abruft.
- Das optionale Argument *driverName* bestimmt das Argument *connString* näher.

Rückgabewerte
Ein Verbindungsnamensstring, der dem Verbindungsstring entspricht.

Beispiel
Mit dem folgenden Code wird der String `"EmpDB"` zurückgegeben.

```
var connectionName = MMDB.getConnectionName ¬
("dsn=EmpDB;uid=;pwd=");
```

MMDB.getConnectionString()

Verfügbarkeit
Dreamweaver UltraDev 1.

Beschreibung
Diese Funktion ruft den Verbindungsstring ab, der mit der benannten Verbindung verknüpft ist.

Argumente
connName

- Das Argument *connName* ist ein Verbindungsname, der im Connection Manager angegeben wurde. Er gibt den Verbindungsstring an, der in Dreamweaver beim Herstellen einer Datenbankverbindung mit einer Live Data-Quelle verwendet werden soll.

Rückgabewerte
Ein Verbindungsstring, der der benannten Verbindung entspricht.

Beispiel
Mit dem Code `var connectionString = MMDB.getConnectionString ("EmpDB")` werden verschiedene Strings für eine ADO- oder JDBC-Verbindung zurückgegeben.

- Bei einer ADO-Verbindung kann folgender String zurückgegeben werden:

```
"dsn=EmpDB;uid=;pwd=";
```

- Bei einer JDBC-Verbindung kann folgender String zurückgegeben werden:

```
"jdbc:inetdae:192.168.64.49:1433?database=pubs&user=JoeUser&¬
password=joesSecret"
```

MMDB.getDriverName()

Verfügbarkeit
Dreamweaver UltraDev 1.

Beschreibung
Diese Funktion ruft den Treibernamen ab, der mit der angegebenen Verbindung verknüpft ist. Nur eine JDBC-Verbindung hat einen Treibernamen.

Argumente
connName

- Das Argument *connName* ist ein Verbindungsname, der im Connection Manager angegeben wurde. Er gibt den Verbindungsstring an, der in Dreamweaver beim Herstellen einer Datenbankverbindung mit einer Live Data-Quelle verwendet werden soll.

Rückgabewerte
Ein String, der den Treibernamen enthält.

Beispiel
Die Anweisung `MMDB.getDriverName ("EmpDB");` kann den folgenden String zurückgeben:

```
"jdbc/oracle/driver/JdbcOracle"
```

MMDB.getLocalDsnList()

Verfügbarkeit
Dreamweaver UltraDev 4.

Beschreibung
Diese Funktion ruft ODBC-DSNs ab, die im System des Benutzers definiert sind.

Argumente
Keine.

Rückgabewerte
Ein Array, das die im System des Benutzers definierten ODBC-DSNs enthält.

MMDB.getPassword()

Verfügbarkeit
Dreamweaver UltraDev 1.

Beschreibung
Diese Funktion ruft das Kennwort für die angegebene Verbindung ab.

Argumente
connName

- Das Argument *connName* ist ein Verbindungsname, der im Connection Manager angegeben wurde. Er gibt den Verbindungsstring an, der in Dreamweaver beim Herstellen einer Datenbankverbindung mit einer Live Data-Quelle verwendet werden soll.

Rückgabewerte
Ein mit dem Verbindungsnamen verknüpfter Kennwortstring.

Beispiel
Die Anweisung `MMDB.getPassword ("EmpDB");` kann beispielsweise `"joessecret"` zurückgeben.

MMDB.getRDSPassword()

Verfügbarkeit
Dreamweaver UltraDev 4.

Beschreibung
Diese Funktion ruft das RDS-Kennwort (Remote Development Services) ab (zur Verwendung mit ColdFusion-Verbindungen).

Argumente
Keine.

Rückgabewerte
Ein String, der das RDS-Kennwort enthält.

MMDB.getRDSUserName()

Verfügbarkeit
Dreamweaver UltraDev 4.

Beschreibung
Diese Funktion ruft den RDS-Benutzernamen (zur Verwendung mit ColdFusion-Verbindungen) ab.

Argumente
Keine.

Rückgabewerte
Ein String, der den RDS-Benutzernamen enthält.

MMDB.getRemoteDsnList()

Verfügbarkeit
Dreamweaver UltraDev 4, verbessert in Dreamweaver MX.

Beschreibung
Diese Funktion ruft die ODBC-DSNs vom Server der Site ab. Die Funktionen `getRDSUserName()` und `getRDSPassword()` werden verwendet, wenn ColdFusion das Servermodell der aktuellen Site ist. Mit dieser Funktion haben Entwickler die Möglichkeit, einen URL-Parameterstring anzugeben, der an die von `MMDB.getRemoteDsnList()` generierte URL für die Remote-Verbindung angefügt wird. Wenn der Entwickler einen Parameterstring angibt, übergibt diese Funktion den String an die HTTP-Verbindungsskripts.

Argumente
{urlParams}

- Das optionale Argument *urlParams* ist ein String, der eine Liste von Ausdrücken im Format *Name=Wert* enthält, die jeweils durch Und-Zeichen (&) getrennt sind. Die Werte dürfen nicht in Anführungszeichen gesetzt werden. Einige Zeichen, z. B. das Leerzeichen im Wert `Hello World`, müssen kodiert werden. Beispiel für ein gültiges Argument, das an `MMDB.getRemoteDsnList()` übergeben werden kann: a=1&b=Hello%20World

Rückgabewerte
Ein Array mit den ODBC-DSNs, die auf dem Server der aktuellen Site definiert sind.

MMDB.getRuntimeConnectionType()

Verfügbarkeit
Dreamweaver UltraDev 1.

Beschreibung
Diese Funktion gibt den Laufzeitverbindungstyp des angegebenen Verbindungsnamens zurück.

Argumente
connName

- Das Argument *connName* ist ein Verbindungsname, der im Connection Manager angegeben wurde. Er gibt den Verbindungsstring an, der in Dreamweaver beim Herstellen einer Datenbankverbindung mit einer Live Data-Quelle verwendet werden soll.

Rückgabewerte
Ein String, der dem Verbindungstyp entspricht. Diese Funktion kann einen der folgenden Werte zurückgeben: `"ADO"`, `"ADODSN"`, `"JDBC"` oder `"CFDSN"`.

Beispiel
Mit dem folgenden Code wird bei einer ADO-Verbindung der String `"ADO"` zurückgegeben.

```
var connectionType = MMDB.getRuntimeConnectionType ("EmpDB")
```

MMDB.getUserName()

Verfügbarkeit
Dreamweaver UltraDev 1.

Beschreibung
Diese Funktion gibt einen Benutzernamen für die angegebene Verbindung zurück.

Argumente
connName

- Das Argument *connName* ist ein Verbindungsname, der im Connection Manager angegeben wurde. Er gibt den Verbindungsstring an, der in Dreamweaver beim Herstellen einer Datenbankverbindung mit einer Live Data-Quelle verwendet werden soll.

Rückgabewerte
Ein mit dem Verbindungsnamen verknüpfter Benutzernamenstring.

Beispiel
Die Anweisung `MMDB.getUserName ("EmpDB");` kann beispielsweise `"amit"` zurückgeben.

MMDB.hasConnectionWithName()

Verfügbarkeit
Dreamweaver UltraDev 4.

Beschreibung
Diese Funktion stellt fest, ob eine Verbindung mit dem angegebenen Namen vorhanden ist.

Argumente
name

- Das Argument *name* ist der Verbindungsname.

Rückgabewerte
Ein boolescher Wert: `true`, wenn eine Verbindung mit dem angegebenen Namen vorhanden ist, andernfalls `false`.

MMDB.needToPromptForRdsInfo()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Diese Funktion legt fest, ob das Dialogfeld „RDS-Anmeldeinformationen“ in Dreamweaver geöffnet werden soll.

Argumente
bForce

- Das Argument *bForce* ist ein boolescher Wert. Der Wert `true` gibt an, dass ein Benutzer, der zuvor das Dialogfeld „RDS-Anmeldeinformationen“ über die Schaltfläche „Abbrechen“ geschlossen hat, weiterhin zur Eingabe der RDS-Anmeldeinformationen aufgefordert werden muss.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der Benutzer zur Eingabe der RDS-Anmeldeinformationen aufgefordert werden muss, andernfalls `false`.

MMDB.needToRefreshColdFusionDsnList()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Diese Funktion weist den Connection Manager an, den Cache-Speicher zu löschen und die ColdFusion-Datenquellenliste vom Anwendungsserver abzurufen, wenn ein Benutzer die Liste das nächste Mal anfordert.

Argumente
Keine.

Rückgabewerte
Keine.

MMDB.popupConnection()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Diese Funktion zeigt ein Verbindungsdialogfeld an. Die Funktion hat die folgenden drei Signaturen:

- Wenn die Argumentliste nur aus dem Argument *dialogFileName* (String) besteht, bewirkt die Funktion `popupConnection()`, dass das Verbindungsdialogfeld angezeigt wird, sodass Sie eine neue Verbindung definieren können.
- Wenn die Argumentliste nur das Argument *connRec* (Verbindungsverweis) enthält, bewirkt die Funktion `popupConnection()`, dass das Verbindungsdialogfeld im Bearbeitungsmodus angezeigt wird, sodass Sie die benannte Verbindung bearbeiten können. In diesem Modus ist das Textfeld „Name“ abgeblendet.
- Wenn die Argumentliste aus dem Argument *connRec* und dem booleschen Wert *bDuplicate* besteht, bewirkt die Funktion `popupConnection()`, dass das Verbindungsdialogfeld im Duplizierungsmodus angezeigt wird. In diesem Modus ist das Textfeld „Name“ leer. Die restlichen Eigenschaften werden kopiert, um ein Duplikat der Verbindung zu definieren.

Argumente
dialogFileName oder *connRec* oder *connRec, bDuplicate*

- Das Argument *dialogFileName* ist ein String, der den Namen einer HTML-Datei enthält, die im Ordner „Configuration/Connections/*Servermodell*“ gespeichert ist. Mit dieser HTML-Datei wird das Dialogfeld definiert, das zur Erstellung einer Verbindung verwendet wird. In dieser Datei müssen drei JavaScript-API-Funktionen implementiert sein: `findConnection()`, `inspectConnection()` und `applyConnection()`. In der Regel erstellen Sie eine JavaScript-Datei, die diese Funktionen implementiert, und anschließend fügen Sie diese Datei in die HTML-Datei ein. (Weitere Informationen zum Herstellen von Verbindungen finden Sie unter „API für Datenbankverbindungen“ auf Seite 81.)
- Das Argument *connRec* ist ein Verweis auf ein vorhandenes Verbindungsobjekt.
- Das Argument *bDuplicate* ist ein boolescher Wert.

Rückgabewerte
Keine. Das definierte Verbindungsdialogfeld wird angezeigt.

MMDB.setRDSPassword()

Verfügbarkeit
Dreamweaver UltraDev 4.

Beschreibung
Diese Funktion legt das RDS-Kennwort fest.

Argumente
password

- Das Argument *password* ist ein String, der das RDS-Kennwort enthält.

Rückgabewerte
Keine.

MMDB.setRDSUserName()

Verfügbarkeit
Dreamweaver UltraDev 4.

Beschreibung
Diese Funktion legt den RDS-Benutzernamen fest.

Argumente
username

- Das Argument *username* ist ein gültiger RDS-Benutzername.

Rückgabewerte
Keine.

MMDB.showColdFusionAdmin()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Diese Funktion zeigt das Dialogfeld „ColdFusion-Administrator“ an.

Argumente
Keine.

Rückgabewerte
Keine. Das Dialogfeld „ColdFusion Administrator“ wird angezeigt.

MMDB.showConnectionMgrDialog()

Verfügbarkeit
Dreamweaver UltraDev 1.

Beschreibung
Diese Funktion zeigt das Dialogfeld „Connection Manager“ an.

Argumente
Keine.

Rückgabewerte
Keine. Das Dialogfeld „Connection Manager“ wird angezeigt.

MMDB.showOdbcDialog()

Verfügbarkeit
Dreamweaver UltraDev 4 (nur Windows).

Beschreibung
Diese Funktion zeigt das Dialogfeld „System-ODBC-Administration“ oder „ODBC-Datenquellen-Administrator“ an.

Argumente
Keine.

Rückgabewerte
Keine. Das Dialogfeld „System-ODBC-Administration“ oder „ODBC-Datenquellen-Administrator“ wird angezeigt.

MMDB.showRdsUserDialog()

Verfügbarkeit
Dreamweaver UltraDev 4.

Beschreibung
Diese Funktion zeigt das Dialogfeld zur Eingabe des RDS-Benutzernamens und Kennworts an.

Argumente
username, password

- Das Argument *username* ist der anfänglich verwendete Benutzername.
- Das Argument *password* ist das anfänglich verwendete Kennwort.

Rückgabewerte
Ein Objekt, das die neuen Werte in den Eigenschaften `username` und `password` enthält. Wenn eine der beiden Eigenschaften nicht definiert ist, hat der Benutzer das Dialogfeld über den Befehl „Abbrechen“ geschlossen.

MMDB.showRestrictDialog()

Verfügbarkeit
Dreamweaver UltraDev 4.

Beschreibung
Diese Funktion zeigt das Dialogfeld „Beschränken“ an.

Argumente
catalog, schema

- Das Argument *catalog* ist der anfänglich verwendete Katalogwert.
- Das Argument *schema* ist der anfänglich verwendete Schemawert.

Rückgabewerte
Ein Objekt, das die neuen Werte in den Eigenschaften `catalog` und `schema` enthält. Wenn eine der beiden Eigenschaften nicht definiert ist, hat der Benutzer das Dialogfeld über den Befehl „Abbrechen“ geschlossen.

MMDB.testConnection()

Verfügbarkeit
Dreamweaver UltraDev 4.

Beschreibung
Diese Funktion prüft die Verbindungseinstellungen. Sie zeigt ein modales Dialogfeld an, in dem die Ergebnisse dargestellt sind.

Argumente

serverPropertiesArray

Diese Funktion erwartet ein einzelnes Argument, ein Array-Objekt, das Werte aus der folgenden Liste enthält, die auf das aktuelle Servermodell zutreffen. Geben Sie für Eigenschaften, die auf die zu prüfende Verbindung nicht zutreffen, einen leeren String (`""`) an.

- Das Argument *type* gibt an, welche DLL-Datei für den Verbindungsaufbau mit einer Datenbank in der Entwurfsphase zum Prüfen von Verbindungseinstellungen verwendet werden soll, wenn *useHTTP* den Wert `false` hat.
- Das Argument *string* ist der ADO-Verbindungsstring oder die JDBC-URL.
- Das Argument *dsn* ist der Datenquellenname (DSN).
- Das Argument *driver* ist der JDBC-Treiber.
- Das Argument *username* ist der Benutzername.
- Das Argument *password* ist das Kennwort.
- Das Argument *useHTTP* ist ein boolescher Wert. Mit dem Wert `true` wird angegeben, dass in Dreamweaver während der Entwurfsphase eine HTTP-Verbindung verwendet werden soll. Andernfalls wird eine DLL-Datei verwendet.

Rückgabewerte

Ein boolescher Wert: `true`, wenn der Verbindungstest erfolgreich war, andernfalls `false`.

Funktionen für den Datenbankzugriff

Mithilfe der Funktionen für den Datenbankzugriff können Sie Daten in Datenbanken abrufen.

Beim Zugriff auf Datenbankinformationen können Sie beispielsweise Metadaten abrufen, mit denen das Schema oder die Struktur einer Datenbank definiert werden. Zu diesen Metadaten gehören Informationen zu Tabellen, Spalten, gespeicherten Prozeduren und Ansichten. Sie können zudem die Ergebnisse der Ausführung einer Datenbankabfrage oder gespeicherten Prozedur anzeigen. Für den Zugriff auf eine Datenbank über diese API werden SQL-Anweisungen (Structured Query Language) verwendet.

Eine Sammlung der Funktionen, über die Datenbankverbindungen verwaltet werden, finden Sie unter „Funktionen für Datenbankverbindungen“ auf Seite 56.

In der folgenden Aufstellung werden einige der Argumente beschrieben, die mit den verfügbaren Funktionen häufig verwendet werden.

- Die meisten Funktionen für den Datenbankzugriff verwenden einen Verbindungsnamen als Argument. Eine Liste der gültigen Verbindungsnamen finden Sie im Connection Manager. Sie können jedoch auch mit der Funktion `MMDB.getConnectionList()` eine Liste aller Verbindungsnamen abrufen.

- Für gespeicherte Prozeduren müssen häufig Parameter angegeben werden. Die Parameterwerte von Datenbankfunktionen können auf zwei Arten festgelegt werden. Eine Möglichkeit besteht darin, ein Array von Parameterwerten (*paramValuesArray)* anzugeben. Wenn Sie nur Parameterwerte festlegen, müssen die Werte in der Reihenfolge angegeben werden, in der die gespeicherte Prozedur sie benötigt. Zweitens können Sie Parameterwerte definieren, um ein Array von Parameternamen (*paramNameArray*) bereitzustellen. Mithilfe der Funktion `MMDB.getSPParamsAsString()` können Sie die Parameter der gespeicherten Prozedur abrufen. Wenn Sie Parameternamen festlegen, müssen die in *paramValuesArray* angegebenen Werte in derselben Reihenfolge angegeben werden, in der auch die Parameternamen in *paramNameArray* festgelegt wurden.

MMDB.getColumnAndTypeList()

Verfügbarkeit
Dreamweaver UltraDev 1.

Beschreibung
Diese Funktion ruft eine Liste der Spalten und der zugehörigen Typen aus einer ausgeführten SQL-Anweisung `SELECT` ab.

Argumente
connName, statement

- Das Argument *connName* ist ein Verbindungsname, der im Connection Manager angegeben wurde. Er gibt den Verbindungsstring an, der in Dreamweaver beim Herstellen einer Datenbankverbindung mit einer Live Data-Quelle verwendet werden soll.
- Das Argument *statement* ist die auszuführende SQL-Anweisung `SELECT`.

Rückgabewerte
Ein String-Array, das eine Liste der Spalten (und der zugehörigen Typen) angibt, die der `SELECT`-Anweisung entsprechen. Andernfalls ein Fehler, wenn die SQL-Anweisung ungültig war oder die Verbindung nicht hergestellt werden konnte.

Beispiel
Der Code `var columnArray = MMDB.getColumnAndTypeList("EmpDB","Select * from Employees")` gibt das folgende String-Array zurück:

```
columnArray[0] = "EmpName"
columnArray[1] = "varchar"
columnArray[2] = "EmpFirstName"
columnArray[3] = "varchar"
columnArray[4] = "Age"
columnArray[5] = "integer"
```

MMDB.getColumnList()

Verfügbarkeit
Dreamweaver UltraDev 1.

Beschreibung
Diese Funktion ruft eine Liste der Spalten aus einer ausgeführten SQL-Anweisung `SELECT` ab.

Argumente
connName, statement

- Das Argument *connName* ist ein Verbindungsname, der im Connection Manager angegeben wurde. Er gibt den Verbindungsstring an, der in Dreamweaver beim Herstellen einer Datenbankverbindung mit einer Live Data-Quelle verwendet werden soll.
- Das Argument *statement* ist die auszuführende SQL-Anweisung `SELECT`.

Rückgabewerte
Ein String-Array, das eine Liste der Spalten angibt, die der `SELECT`-Anweisung entsprechen. Andernfalls ein Fehler, wenn die SQL-Anweisung ungültig war oder die Verbindung nicht hergestellt werden konnte.

Beispiel
Der Code `var columnArray = MMDB.getColumnList("EmpDB","Select * from Employees")` gibt das folgende String-Array zurück:

```
columnArray[0] = "EmpName"
columnArray[1] = "EmpFirstName"
columnArray[2] = "Age"
```

MMDB.getColumns()

Verfügbarkeit
Dreamweaver MX, Argumente aktualisiert in Dreamweaver MX 2004.

Beschreibung
Diese Funktion gibt ein Array von Objekten zurück, die die in der Tabelle angegebenen Spalten beschreiben.

Argumente
connName, tableName

- Das Argument *connName* ist der Verbindungsname. Dieser Wert gibt den Verbindungsstring an, der beim Herstellen einer Datenbankverbindung mit einer Live Data-Quelle verwendet werden soll.
- Das Argument *tableName* ist die abzufragende Tabelle.

Rückgabewerte
Ein Array von Objekten, wobei für jede Spalte ein Objekt verwendet wird. Jedes Objekt definiert die folgenden Eigenschaften für die jeweils zugehörige Spalte.

Name der Eigenschaft	Beschreibung
`name`	Name der Spalte (z. B. `price`)
`datatype`	Datentyp der Spalte (z. B. `small money`)
`definedsize`	Definierte Größe der Spalte (z. B. `8`)
`nullable`	Gibt an, ob die Spalte `null`-Werte enthalten kann.

Beispiel
Im folgenden Beispiel wird mithilfe von `MMDB.getColumns()` der QuickInfo-Textwert festgelegt.

```
var columnNameObjs = MMDB.getColumns(connName,tableName);
var databaseType = MMDB.getDatabaseType(connName);
    for (i = 0; i < columnNameObjs.length; i++)
    {
        var columnObj = columnNameObjs[i];
        var columnName = columnObj.name;
        var typename = columnObj.datatype;
        if (dwscripts.isNumber(typename))
        {
            // it already is a num
            typename = dwscripts.getDBColumnTypeAsString(typename, databaseType);
        }
        var tooltiptext = typename;
    }
```

MMDB.getColumnsOfTable()

Verfügbarkeit
Dreamweaver UltraDev 1.

Beschreibung
Diese Funktion ruft eine Liste aller Spalten in der angegebenen Tabelle ab.

Argumente
connName, tableName

- Das Argument *connName* ist ein Verbindungsname, der im Connection Manager angegeben wurde. Er gibt den Verbindungsstring an, der in Dreamweaver beim Herstellen einer Datenbankverbindung mit einer Live Data-Quelle verwendet werden soll.
- Das Argument *tableName* ist der Name einer Tabelle in der Datenbank, die über *connName* angegeben wurde.

Rückgabewerte
Ein String-Array, in dem jeder String der Name einer Spalte in der Tabelle ist.

Beispiel
Die Anweisung `MMDB.getColumnsOfTable ("EmpDB","Employees");` gibt die folgenden Strings zurück:

```
["EmpID", "FirstName", "LastName"]
```

MMDB.getPrimaryKeys()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Diese Funktion gibt die Namen der Spalten zurück, die gemeinsam den Primärschlüssel der benannten Tabelle bilden. Ein Primärschlüssel ist der eindeutige Bezeichner einer Datenbankzeile und besteht aus mindestens einer Spalte.

Argumente
connName, tableName

- Das Argument *connName* ist ein Verbindungsname, der im Connection Manager angegeben wurde. Er gibt den Verbindungsstring an, der in Dreamweaver beim Herstellen einer Datenbankverbindung mit einer Live Data-Quelle verwendet werden soll.
- Das Argument *tableName* ist der Name der Tabelle, aus der Sie die Spalten abrufen möchten, die den Primärschlüssel der Tabelle bilden.

Rückgabewerte
Ein String-Array. Das Array enthält jeweils einen String für alle Spalten, die den Primärschlüssel bilden.

Beispiel
Im folgenden Beispiel wird der Primärschlüssel für die angegebene Tabelle zurückgegeben.

```
var connName      = componentRec.parent.parent.parent.name;
var tableName     = componentRec.name;
var primaryKeys    = MMDB.getPrimaryKeys(connName,tableName);
```

MMDB.getProcedures()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Diese Funktion gibt ein Array der Prozedurobjekte zurück, die mit einer benannten Verbindung verknüpft sind.

Argumente
connName

- Das Argument *connName* ist ein Verbindungsname, der im Connection Manager angegeben wurde. Er gibt den Verbindungsstring an, der in Dreamweaver beim Herstellen einer Datenbankverbindung mit einer Live Data-Quelle verwendet werden soll.

Rückgabewerte
Ein Array der Prozedurobjekte, in dem jedes Prozedurobjekt die folgenden drei Eigenschaften aufweist.

Name der Eigenschaft	Beschreibung
`schema`	Name des mit dem Objekt verknüpften Schemas. Diese Eigenschaft gibt den Benutzer an, der mit der gespeicherten Prozedur in der SQL-Datenbank verknüpft ist, auf die mit der Funktion `getProcedures()` zugegriffen wird. Die von dieser Funktion aufgerufene Datenbank hängt vom Typ der Verbindung ab. • Bei ODBC-Verbindungen wird die Datenbank durch die ODBC-Datenquelle definiert. Der DSN wird durch die Eigenschaft `dsn` im Verbindungsobjekt (connName) angegeben, das Sie an die Funktion `getProcedures()` übergeben. • Bei OLE-Datenbankverbindungen wird die Datenbank durch den Verbindungsstring definiert.
`catalog`	Name des mit dem Objekt verknüpften Katalogs (Eigentümerqualifizierer). Der Wert der Eigenschaft `catalog` wird durch ein Attribut des OLE-Datenbanktreibers definiert. Mit diesem Treiberattribut wird die Standardeigenschaft `user.database` definiert, die verwendet wird, wenn im OLE-Verbindungsstring keine Datenbank angegeben ist.
`procedure`	Name der Prozedur.

Hinweis: *Wenn Sie eine Datensatzgruppe ändern, wird eine Verbindung mit der Datenbank hergestellt. Dann werden alle Tabellen in der Datenbank abgerufen. Wenn die Datenbank viele Tabellen enthält, nimmt das Abrufen der Tabellen auf bestimmten Systemen möglicherweise viel Zeit in Anspruch. Wenn die Datenbank ein Schema oder einen Katalog enthält, können Sie mit dem Schema bzw. dem Katalog die Anzahl der Datenbankelemente beschränken, die in Dreamweaver während der Entwurfsphase abgerufen werden. Zunächst müssen Sie in der Datenbankanwendung ein Schema oder einen Katalog erstellen, bevor Sie das Schema oder den Katalog in Dreamweaver zuweisen können. Weitere Informationen erhalten Sie in der Dokumentation zu Ihrer Datenbank oder von Ihrem Systemadministrator.*

Beispiel

Mit folgendem Code wird eine Liste der Prozeduren abgerufen:

```
var procObjects = MMDB.getProcedures(connectionName);
for (i = 0; i < procObjects.length; i++)
{
    var thisProcedure = procObjects[i]
    thisSchema =Trim(thisProcedure.schema)
    if (thisSchema.length == 0)
    {
    thisSchema = Trim(thisProcedure.catalog)
    }
    if (thisSchema.length > 0)
    {
    thisSchema += "."
    }

    var procName = String(thisSchema + thisProcedure.procedure);
 }
```

MMDB.getSPColumnList()

Verfügbarkeit

Dreamweaver UltraDev 1.

Beschreibung
Diese Funktion ruft eine Liste der Ergebnissatzspalten ab, die durch einen Aufruf der angegebenen gespeicherten Prozedur generiert wurden.

Argumente
connName, statement, paramValuesArray

- Das Argument *connName* ist ein Verbindungsname, der im Connection Manager angegeben wurde. Er gibt den Verbindungsstring an, der in Dreamweaver beim Herstellen einer Datenbankverbindung mit einer Live Data-Quelle verwendet werden soll.
- Das Argument *statement* ist der Name der gespeicherten Prozedur, die den Ergebnissatz zurückgibt, wenn sie ausgeführt wird.
- Das Argument *paramValuesArray* ist ein Array, das eine Liste von Testwerten der Entwurfsphasenparameter enthält. Geben Sie die Parameterwerte in der Reihenfolge an, in der sie von der gespeicherten Prozedur erwartet werden. Mithilfe der Funktion `MMDB.getSPParamsAsString()` können Sie die Parameter der gespeicherten Prozedur abrufen.

Rückgabewerte
Ein String-Array, das die Liste der Spalten enthält. Diese Funktion gibt einen Fehler zurück, wenn die SQL-Anweisung oder der Verbindungsstring ungültig ist.

Beispiel
Mit dem folgenden Code kann eine Liste der Ergebnissatzspalten zurückgegeben werden, die von der ausgeführten gespeicherten Prozedur `getNewEmployeesMakingAtLeast` generiert wurden.

```
var paramValueArray = new Array("2/1/2000", "50000")
var columnArray = MMDB.getSPColumnList("EmpDB", ¬
"getNewEmployeesMakingAtLeast", paramValueArray)
The following values return:
columnArray[0] = "EmpID", columnArray[1] = "LastName", ¬
columnArray[2] ="startDate", columnArray[3] = "salary"
```

MMDB.getSPColumnListNamedParams()

Verfügbarkeit
Dreamweaver UltraDev 1.

Beschreibung
Diese Funktion ruft eine Liste der Ergebnissatzspalten ab, die durch einen Aufruf der angegebenen gespeicherten Prozedur generiert wurden.

Argumente
connName, statement, paramNameArray, paramValuesArray

- Das Argument *connName* ist ein Verbindungsname, der im Connection Manager angegeben wurde. Er gibt den Verbindungsstring an, der in Dreamweaver beim Herstellen einer Datenbankverbindung mit einer Live Data-Quelle verwendet werden soll.
- Das Argument *statement* ist der Name der gespeicherten Prozedur, die den Ergebnissatz zurückgibt, wenn sie ausgeführt wird.

- Das Argument *paramNameArray* ist ein Array, das eine Liste mit Parameternamen enthält. Mithilfe der Funktion `MMDB.getSPParamsAsString()` können Sie die Parameter der gespeicherten Prozedur abrufen.
- Das Argument *paramValuesArray* ist ein Array, das eine Liste von Testwerten der Entwurfsphasenparameter enthält. Sie können angeben, ob beim Ausführen der Prozedur Parameter erforderlich sind. Wenn Sie in *paramNameArray* Parameternamen festgelegt haben, geben Sie die Parameternamen in derselben Reihenfolge an, in der sie in *paramNameArray* aufgeführt sind. Wenn Sie keine Parameter für *paramNameArray* angegeben haben, legen Sie die Werte in der Reihenfolge fest, in der sie von der gespeicherten Prozedur erwartet werden.

Rückgabewerte
Ein String-Array, das die Liste der Spalten enthält. Diese Funktion gibt einen Fehler zurück, wenn die SQL-Anweisung oder der Verbindungsstring ungültig ist.

Beispiel
Mit dem folgenden Code kann eine Liste der Ergebnissatzspalten zurückgegeben werden, die von der ausgeführten gespeicherten Prozedur `getNewEmployeesMakingAtLeast` generiert wurden.

```
var paramNameArray = new Array("startDate", "salary")
var paramValueArray = new Array("2/1/2000", "50000")
var columnArray = MMDB.getSPColumnListNamedParams("EmpDB", ¬
"getNewEmployeesMakingAtLeast", paramNameArray, paramValueArray)
```

Die folgenden Werte werden zurückgegeben:

```
columnArray[0] = "EmpID", columnArray[1] = "LastName",¬
columnArray[2] ="startDate", columnArray[3] = "salary"
```

MMDB.getSPParameters()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Diese Funktion gibt ein Array der Parameterobjekte einer benannten Prozedur zurück.

Argumente
connName, procName

- Das Argument *connName* ist ein Verbindungsname, der im Connection Manager angegeben wurde. Er gibt den Verbindungsstring an, der in Dreamweaver beim Herstellen einer Datenbankverbindung mit einer Live Data-Quelle verwendet werden soll.
- Das Argument *procName* ist der Name der Prozedur.

Rückgabewerte
Ein Array von Parameterobjekten, die jeweils die folgende Gruppe von Eigenschaften angeben.

Name der Eigenschaft	Beschreibung
`name`	Name des Parameters (z. B. `@@lolimit`)
`datatype`	Datentyp des Parameters (z. B. `smallmoney`)
`direction`	Richtung des Parameters: `1` – Der Parameter wird nur für Eingaben verwendet. `2` – Der Parameter wird nur für Ausgaben verwendet. In diesem Fall wird der Parameter durch Verweis übergeben. Die Methode weist dann dem Parameter einen Wert zu. Sie können den Wert nach der Rückgabe durch die Methode verwenden. `3` – Der Parameter wird für Ein- und Ausgaben verwendet. `4` – Der Parameter enthält einen Rückgabewert.

Beispiel

Im folgenden Beispiel werden die Parameterobjekte für die angegebene Prozedur abgerufen und für jedes Objekt wird anhand der entsprechenden Eigenschaft eine QuickInfo erstellt.

```
var paramNameObjs = MMDB.getSPParameters(connName,procName);
for (i = 0; i < paramNameObjs.length; i++)
{
    var paramObj = paramNameObjs[i];
    var tooltiptext = paramObj.datatype;
    tooltiptext+=" ";
    tooltiptext+=GetDirString(paramObj.directiontype);
}
```

MMDB.getSPParamsAsString()

Verfügbarkeit

Dreamweaver UltraDev 1.

Beschreibung

Diese Funktion ruft einen String ab, der Kommas als Trennzeichen verwendet und die Liste der Parameter enthält, die die gespeicherte Prozedur erwartet.

Argumente

connName, procName

- Das Argument *connName* ist ein Verbindungsname, der im Connection Manager angegeben wurde. Er gibt den Verbindungsstring an, der in Dreamweaver beim Herstellen einer Datenbankverbindung mit einer Live Data-Quelle verwendet werden soll.
- Das Argument *procName* ist der Name der gespeicherten Prozedur.

Rückgabewerte

Ein String, der Kommas als Trennzeichen verwendet und die Liste der Parameter enthält, die für die gespeicherte Prozedur erforderlich sind. Der Name, die Richtung und der Datentyp der Parameter sind enthalten und jeweils durch einen Strichpunkt (;) getrennt.

Beispiel
Der Code `MMDB.getSPParamsAsString ("EmpDB","getNewEmployeesMakingAtLeast")` kann einen String der Form `startDate;direction:in;datatype:date, salary;direction:in;datatype:integer` zurückgeben.

In diesem Beispiel hat die gespeicherte Prozedur `getNewEmployeesMakingAtLeast` zwei Parameter: `startDate` und `Salary`. Bei `startDate` lautet die Richtung `in` und der Datentyp `date`. Bei `salary` lautet die Richtung `in` und der Datentyp `integer`.

MMDB.getTables()

Verfügbarkeit
Dreamweaver UltraDev 1.

Beschreibung
Diese Funktion ruft eine Liste aller Tabellen ab, die für die angegebene Datenbank definiert sind. Jedes Tabellenobjekt hat drei Eigenschaften: `table`, `schema` und `catalog`.

Argumente
connName

- Das Argument *connName* ist ein Verbindungsname, der im Connection Manager angegeben wurde. Er gibt den Verbindungsstring an, der in Dreamweaver beim Herstellen einer Datenbankverbindung mit einer Live Data-Quelle verwendet werden soll.

Rückgabewerte
Ein Array von Objekten, in dem jedes Tabellenobjekt drei Eigenschaften hat: `table`, `schema` und `catalog`. Die Eigenschaft `table` ist der Name der Tabelle. Der Wert für `schema` ist der Name des Schemas, das die Tabelle enthält. Die Eigenschaft `catalog` ist der Katalog, der die Tabelle enthält.

Beispiel
Die Anweisung `MMDB.getTables ("EmpDB");` kann beispielsweise ein Array mit zwei Objekten generieren. Die Eigenschaften des ersten Objekts entsprechen möglicherweise dem folgenden Beispiel:

```
object1[table:"Employees", schema:"personnel", catalog:"syscat"]
```

Die Eigenschaften des zweiten Objekts entsprechen möglicherweise dem folgenden Beispiel:

```
object2[table:"Departments", schema:"demo", catalog:"syscat2"]
```

MMDB.getViews()

Verfügbarkeit
Dreamweaver UltraDev 4.

Beschreibung
Diese Funktion ruft eine Liste aller Ansichten ab, die für die angegebene Datenbank definiert sind. Jedes Ansichtsobjekt hat die Eigenschaften `catalog`, `schema` und `view`.

Argumente
connName

- Das Argument *connName* ist ein Verbindungsname, der im Connection Manager angegeben wurde. Er gibt den Verbindungsstring an, der in Dreamweaver beim Herstellen einer Datenbankverbindung mit einer Live Data-Quelle verwendet werden soll.

Rückgabewerte
Ein Array von Ansichtsobjekten, in dem jedes Objekt drei Eigenschaften hat: `catalog`, `schema` und `view`. Verwenden Sie `catalog` und `schema` zum Einschränken oder Filtern der Anzahl der Ansichten, die sich auf einen bestimmten Schema- oder Katalognamen beziehen, der im Rahmen der Verbindungsinformationen definiert wurde.

Beispiel
Im folgenden Beispiel werden die Ansichten für den Verbindungswert `CONN_LIST.getValue()` zurückgegeben.

```
var viewObjects = MMDB.getViews(CONN_LIST.getValue())
for (i = 0; i < viewObjects.length; i++)
{
    thisView = viewObjects[i]
    thisSchema = Trim(thisView.schema)
    if (thisSchema.length == 0)
    {
        thisSchema = Trim(thisView.catalog)
    }
    if (thisSchema.length > 0)
    {
        thisSchema += "."
    }
    views.push(String(thisSchema + thisView.view))
}
```

MMDB.showResultset()

Verfügbarkeit
Dreamweaver UltraDev 1.

Beschreibung
Diese Funktion zeigt ein Dialogfeld mit den Ergebnissen aus der Ausführung der angegebenen SQL-Anweisung an. Im Dialogfeld wird ein tabellarisches Raster angezeigt, dessen Kopfzeile die Spalteninformationen für den Ergebnissatz angibt. Wenn der Verbindungsstring oder die SQL-Anweisung ungültig ist, wird ein Fehler angezeigt. Diese Funktion überprüft die SQL-Anweisung auf Gültigkeit.

Argumente
connName, SQLstatement

- Das Argument *connName* ist ein Verbindungsname, der im Connection Manager angegeben wurde. Er gibt den Verbindungsstring an, der in Dreamweaver beim Herstellen einer Datenbankverbindung mit einer Live Data-Quelle verwendet werden soll.
- Das Argument *SQLstatement* ist die SQL-Anweisung `SELECT`.

Rückgabewerte
Keine. Diese Funktion gibt einen Fehler zurück, wenn die SQL-Anweisung oder der Verbindungsstring ungültig ist.

Beispiel
Mit dem folgenden Code werden die Ergebnisse der ausgeführten SQL-Anweisung angezeigt.

```
MMDB.showResultset("EmpDB","Select EmpName,EmpFirstName,Age ¬
from Employees")
```

MMDB.showSPResultset()

Verfügbarkeit
Dreamweaver UltraDev 1.

Beschreibung
Diese Funktion zeigt ein Dialogfeld mit den Ergebnissen aus der Ausführung der angegebenen gespeicherten Prozedur an. Im Dialogfeld wird ein tabellarisches Raster angezeigt, dessen Kopfzeile die Spalteninformationen für den Ergebnissatz angibt. Wenn der Verbindungsstring oder die gespeicherte Prozedur ungültig ist, wird ein Fehler angezeigt. Diese Funktion überprüft die gespeicherte Prozedur auf Gültigkeit.

Argumente
connName, procName, paramValuesArray

- Das Argument *connName* ist ein Verbindungsname, der im Connection Manager angegeben wurde. Er gibt den Verbindungsstring an, der in Dreamweaver beim Herstellen einer Datenbankverbindung mit einer Live Data-Quelle verwendet werden soll.
- Das Argument *procName* ist der Name der auszuführenden gespeicherten Prozedur.
- Das Argument *paramValuesArray* ist ein Array, das eine Liste von Testwerten der Entwurfsphasenparameter enthält. Geben Sie die Parameterwerte in der Reihenfolge an, in der sie von der gespeicherten Prozedur erwartet werden. Mithilfe der Funktion `MMDB.getSPParamsAsString()` können Sie die Parameter der gespeicherten Prozedur abrufen.

Rückgabewerte
Diese Funktion gibt einen Fehler zurück, wenn die SQL-Anweisung oder der Verbindungsstring ungültig ist. Andernfalls wird kein Wert zurückgegeben.

Beispiel
Mit dem folgenden Code werden die Ergebnisse der ausgeführten gespeicherten Prozedur angezeigt.

```
var paramValueArray = new Array("2/1/2000", "50000")
MMDB.showSPResultset("EmpDB", "getNewEmployeesMakingAtLeast", ¬
paramValueArray)
```

MMDB.showSPResultsetNamedParams()

Verfügbarkeit
Dreamweaver UltraDev 1.

Beschreibung

Diese Funktion zeigt ein Dialogfeld mit den Ergebnissen der angegebenen gespeicherten Prozedur an. Im Dialogfeld wird ein tabellarisches Raster angezeigt, dessen Kopfzeile die Spalteninformationen für den Ergebnissatz angibt. Wenn der Verbindungsstring oder die gespeicherte Prozedur ungültig ist, wird ein Fehler angezeigt. Diese Funktion überprüft die gespeicherte Prozedur auf Gültigkeit. Diese Funktion unterscheidet sich von `MMDB.showSPResultset()`, da Sie Parameterwerte nach Namen festlegen können und nicht in der Reihenfolge angeben müssen, in der sie von der gespeicherten Prozedur erwartet werden.

Argumente

connName, procName, paramNameArray, paramValuesArray

- Das Argument *connName* ist ein Verbindungsname, der im Connection Manager angegeben wurde. Er gibt den Verbindungsstring an, der in Dreamweaver beim Herstellen einer Datenbankverbindung mit einer Live Data-Quelle verwendet werden soll.
- Das Argument *procName* ist der Name der gespeicherten Prozedur, die den Ergebnissatz zurückgibt, wenn sie ausgeführt wird.
- Das Argument *paramNameArray* ist ein Array, das eine Liste mit Parameternamen enthält. Mithilfe der Funktion `MMDB.getSPParamsAsString()` können Sie die Parameter der gespeicherten Prozedur abrufen.
- Das Argument *paramValuesArray* ist ein Array, das eine Liste von Testwerten der Entwurfsphasenparameter enthält.

Rückgabewerte

Diese Funktion gibt einen Fehler zurück, wenn die SQL-Anweisung oder der Verbindungsstring ungültig ist. Andernfalls wird kein Wert zurückgegeben.

Beispiel

Mit dem folgenden Code werden die Ergebnisse der ausgeführten gespeicherten Prozedur angezeigt.

```
var paramNameArray = new Array("startDate", "salary")
var paramValueArray = new Array("2/1/2000", "50000")
MMDB.showSPResultsetNamedParams("EmpDB","getNewEmployees¬
MakingAtLeast", paramNameArray, paramValueArray)
```

Kapitel 9: API für Datenbankverbindungen

Als Entwickler können Sie neue Verbindungstypen und die entsprechenden Dialogfelder für neue oder vorhandene Servermodelle für Adobe® Dreamweaver® erstellen. Anschließend kann ein Benutzer ein Verbindungsobjekt anlegen, wenn er eine Site einrichtet, um Seiten zu erstellen. Zum Erstellen des Verbindungsobjekts muss der Benutzer zunächst die spezifische Verbindung auswählen, die Sie erstellt haben.

Auswählen eines neuen Verbindungstyps

Der Benutzer kann den neuen Verbindungstyp wie folgt auswählen:

- Er kann im Bedienfeld „Anwendung“ auf die Schaltfläche mit dem Pluszeichen (+) klicken und die Option „Datensatzgruppe“ auswählen. Im Dialogfeld „Datensatzgruppe“ kann er dann die Einträge des Popupmenüs „Verbindung“ anzeigen.
- Er kann im Bedienfeld „Datenbanken“ auf der Registerkarte „Datenbank“ auf die Schaltfläche mit dem Pluszeichen (+) klicken und „Data Source Name (DSN)“ auswählen.

Erstellen eines neuen Verbindungstyps

Mit den folgenden Schritten wird der Vorgang zum Erstellen eines neuen Verbindungstyps erläutert.

1 Erstellen Sie das Layout für das Verbindungsdialogfeld.

Erstellen Sie eine HTML-Datei, in der die Benutzeroberfläche für das Verbindungsdialogfeld festgelegt wird. Speichern Sie die Datei unter dem Namen der Verbindung (z. B. „myConnection.htm“). Informationen zum Erstellen von Dialogfeldern finden Sie im Handbuch *Erste Schritte mit Dreamweaver*.

Stellen Sie sicher, dass die HTML-Datei einen Verweis auf die JavaScript-Implementierungsdatei enthält, die Sie in Schritt 2 definieren. Erstellen Sie eine JavaScript-Datei, die mindestens die folgenden Elemente implementiert:

```
<head>
    <script SRC="../myConnectionImpl.js"></script>
</head>
```

Speichern Sie diese HTML-Datei, die das Verbindungsdialogfeld definiert, im Ordner Configuration/Connections/*Servermodell/Plattform* (*Plattform* steht entweder für Windows oder für Macintosh).

Das standardmäßige ADO-Verbindungsdialogfeld für ein ASP-JavaScript-Dokument unter Windows wird beispielsweise unter dem Namen „Connection_ado_conn_string.htm“ im Ordner „ASP_Js/Win“ gespeichert.

Hinweis: *Zur Laufzeit erstellt Dreamweaver dynamisch eine Liste der Verbindungstypen, die dem Benutzer aus der Auswahl der Dialogfelder im Ordner „ASP_Js/Win“ zur Verfügung stehen.*

Im Ordner „Configuration/ServerModel“ befinden sich HTML-Dateien, die die einzelnen Servermodelle definieren. Jede dieser HTML-Dateien enthält die Funktion `getServerModelFolderName()`, die den Namen des Ordners zurückgibt, der mit dem jeweiligen Servermodell verknüpft ist. Im folgenden Beispiel ist die Funktion für den ASP-JavaScript-Dokumenttyp dargestellt:

```
function getServerModelFolderName()
{
    return "ASP_JS";
}
```

Anhand der Datei „MMDocumentTypes.xml“ im Ordner „Configuration/DocumentTypes“ können Sie die Zuordnung der Servermodelle zu den Dokumenttypen ermitteln.

2 Erstellen Sie eine JavaScript-Datei, die mindestens die folgenden Elemente implementiert:

Element	Beschreibung	Beispiele
Eine Gruppe von Variablen	Jede Variable definiert eine bestimmte Verbindungseigenschaft.	Verbindungstyp, DSN (Data Source Name) usw.
Eine Gruppe von Schaltflächen	Jede Schaltfläche wird im Verbindungsdialogfeld angezeigt.	„Testen“, „Hilfe“ usw. („OK“ und „Abbrechen“ werden automatisch eingefügt.)
Verbindungsfunktionen	Alle diese Funktionen definieren die API für Datenbankverbindungen.	• `findConnection()` • applyConnection() • inspectConnection()

Sie können einen beliebigen Namen für diese Implementierungsdatei auswählen, sie muss jedoch die Erweiterung „.js“ haben (z. B. „myConnectionImpl.js“). Sie können die Datei lokal oder auf einem Remote-Computer speichern. Sie können die Datei auch in einem Unterordner im Ordner „Configuration/Connections“ speichern.

Hinweis: *Die in Schritt 1 („Erstellen Sie das Layout für das Verbindungsdialogfeld“) definierte HTML-Datei muss diese Implementierungsdatei für den Verbindungstyp enthalten.*

Wenn Sie die in der Standarddatei „connection_includefile.edml“ festgelegten Verbindungsparameter nicht anpassen möchten, kann mit diesen beiden Schritten ein neues Verbindungsdialogfeld erstellt werden.

Hinweis: *Der Titel des Dialogfelds in der Benutzeroberfläche steht im Tag* `title`*, das im HTML-Dokument angegeben wird.*

Mithilfe der im nächsten Abschnitt aufgeführten Funktionen können Sie ein Verbindungsdialogfeld erstellen. Neben der Implementierung der Aufrufe zum Generieren von Include-Dateien für den Benutzer können Sie Ihren Verbindungstyp im Abschnitt für die Servermodelle in der XML-Verbindungsdatei registrieren.

Informationen zur API für Datenbankverbindungen im Zusammenhang mit dem Erstellen einer neuen Verbindung finden Sie unter „Funktionen für Datenbankverbindungen“ auf Seite 56.

API für Verbindungen

Um einen neuen Verbindungstyp zu erstellen, einschließlich des Dialogfelds für die Benutzeroberfläche, müssen Sie die folgenden drei Funktionen implementieren: `findConnection()`, `inspectConnection()` und `applyConnection()`. Sie programmieren die drei Funktionen und fügen sie in die JavaScript-Implementierungsdatei ein, die mit dem neuen Verbindungstyp verknüpft ist (siehe Schritt 2, „Erstellen Sie eine JavaScript-Datei, die mindestens die folgenden Elemente implementiert“).

Die Funktion `applyConnection()` gibt eine HTML-Quelle in einer Include-Datei zurück. Beispiele für die HTML-Quelle finden Sie unter „Generierte Include-Datei“ auf Seite 85. Die Funktion `findConnection()` extrahiert die Eigenschaften der HTML-Quelle. Sie können die Funktion `findConnection()` implementieren, um mithilfe der Suchmuster aus XML-Dateien Informationen zu extrahieren, die von `applyConnection()` zurückgegeben werden. Beispiele für diese Implementierung finden Sie in den folgenden beiden JavaScript-Dateien:

- Die Datei „connection_ado_conn_string.js“ befindet sich im Ordner „Configuration/Connections/ASP_Js“.
- Die Datei „connection_common.js“ befindet sich im Ordner „Configuration/Connections/Shared“.

Wenn der Benutzer eine Site aufruft, öffnet Dreamweaver jede Datei im Ordner „Connections“ und übergibt deren Inhalt an die Funktion `findConnection()`. Wenn der Inhalt einer Datei den Kriterien für eine gültige Verbindung entspricht, gibt die Funktion `findConnection()` ein Verbindungsobjekt zurück. Alle Verbindungsobjekte werden dann in Dreamweaver im Bedienfeld „Database Explorer“ angezeigt.

Wenn der Benutzer ein Verbindungsdialogfeld öffnet, um eine neue Verbindung zu erstellen oder eine bestehende zu kopieren bzw. zu bearbeiten, ruft Dreamweaver die Funktion `inspectConnection()` auf und gibt das gleiche Verbindungsobjekt zurück, das von `findConnection()` erstellt wurde. Auf diese Weise kann das Dialogfeld mit Verbindungsinformationen gefüllt werden.

Wenn der Benutzer in einem Verbindungsdialogfeld auf „OK“ klickt, ruft Dreamweaver die Funktion `applyConnection()` auf, um die HTML-Datei zu erstellen, die dann in die Include-Datei im Ordner „Configuration/Connections“ eingefügt wird. Die Funktion `applyConnection()` gibt einen leeren String zurück, der angibt, dass eines der Felder einen Fehler enthält und das Dialogfeld noch nicht geschlossen werden sollte. Die Dateinamenerweiterung der Include-Datei entspricht dem aktuellen Servermodell.

Wenn der Benutzer einer Seite ein Serververhalten hinzufügt, das die Verbindung verwendet (z. B. eine Datensatzgruppe oder eine gespeicherte Prozedur), wird eine Anweisung in die Seite eingefügt, die die Include-Datei der Verbindung enthält.

findConnection()

Verfügbarkeit
Dreamweaver UltraDev 4.

Beschreibung
Dreamweaver ruft diese Funktion auf, um eine Verbindung in der ausgewählten HTML-Quelldatei zu erkennen und die Verbindungsparameter zu analysieren. Wenn der Inhalt dieser Quelldatei mit den Kriterien für eine gültige Verbindung übereinstimmt, gibt `findConnection()` ein Verbindungsobjekt zurück. Andernfalls wird der Wert `null` zurückgegeben.

Argumente
`htmlSource`

Das Argument *htmlSource* ist die HTML-Quelldatei für eine Verbindung.

Rückgabewerte
Ein Verbindungsobjekt, das Werte für eine bestimmte Kombination der in der folgenden Tabelle aufgeführten Eigenschaften enthält. Die Eigenschaften, für die diese Funktion einen Wert zurückgibt, hängen vom jeweiligen Dokumenttyp ab.

Eigenschaft	Beschreibung
`name`	Name der Verbindung
`type`	Wenn `useHTTP` den Wert `false` hat, gibt diese Eigenschaft an, welche DLL für die Verbindung mit der Datenbank zur Laufzeit verwendet werden soll.
`string`	Verbindungsstring zur Laufzeit. Für ADO handelt es sich um einen String mit Verbindungsparametern, für JDBC um eine Verbindungs-URL.
`dsn`	Data Source Name, der für ODBC- oder ColdFusion-Verbindungen zur Laufzeit verwendet wird.
`driver`	Name eines zur Laufzeit verwendeten JDBC-Treibers
`username`	Name des Benutzers für die Laufzeitverbindung
`password`	Kennwort für die Laufzeitverbindung
`designtimeString`	Entwurfsphasen-Verbindungsstring (siehe `string`)
`designtimeDsn`	Entwurfsphasen-Data Source Name (siehe `dsn`)
`designtimeDriver`	Name eines während der Entwurfsphase verwendeten JDBC-Treibers
`designtimeUsername`	Name des Benutzers für die Entwurfsphasenverbindung
`designtimePassword`	Kennwort für die Entwurfsphasenverbindung
`designtimeType`	Entwurfsphasen-Verbindungstyp
`usesDesigntimeInfo`	Wenn `false`, werden in der Entwurfsphase Laufzeiteigenschaften verwendet, andernfalls werden Entwurfsphaseneigenschaften verwendet.
`useHTTP`	String mit dem Wert `true` oder `false`.`true` gibt an, dass in der Entwurfsphase die HTTP-Verbindung verwendet werden soll. `false` gibt an, dass die DLL verwendet werden soll.
`includePattern`	Ein regulärer Ausdruck, mit dem die include-Anweisung auf der Seite in der Ansicht „Live Data“ und „Vorschau in Browser“ gesucht wird.
`variables`	Objekt mit einer Eigenschaft für jede Seitenvariable, die auf den entsprechenden Wert gesetzt wird. Dieses Objekt wird in den Ansichten „Live Data“ und „Vorschau in Browser“ verwendet.
`catalog`	String mit einem Datenbankbezeichner, der den Umfang der angezeigten Metadaten einschränkt.
`schema`	String mit einem Datenbankbezeichner, der den Umfang der angezeigten Metadaten einschränkt.
`filename`	Name des Dialogfelds, über das die Verbindung erstellt wird.

Wenn keine Verbindung in htmlSource gefunden wurde, wird der Wert `null` zurückgegeben.

Hinweis: *Entwickler können benutzerdefinierte Eigenschaften (z. B. Metadaten) in die HTML-Quelle einfügen, die von* `applyConnection()` *zusammen mit den Standardeigenschaften zurückgeben werden.*

inspectConnection()

Verfügbarkeit
Dreamweaver UltraDev 4.

Beschreibung
Dreamweaver ruft diese Funktion auf, um die Dialogfelddaten zum Definieren der Verbindung zu initialisieren, wenn der Benutzer eine vorhandene Verbindung bearbeitet. Auf diese Weise können im Dialogfeld die richtigen Verbindungsinformationen angezeigt werden.

Argumente
parameters

Beim Argument *parameters* handelt es sich um das gleiche Objekt, das die Funktion `findConnection()` zurückgibt.

Rückgabewerte
Keine.

applyConnection()

Verfügbarkeit
Dreamweaver UltraDev 4.

Beschreibung
Dreamweaver ruft diese Funktion auf, wenn der Benutzer im Verbindungsdialogfeld auf „OK“ klickt. Die Funktion `applyConnection()` generiert den HTML-Quellcode für eine Verbindung. Dreamweaver schreibt den HTML-Code in die Include-Datei *Verbindungsname.erw* im Ordner „Configuration/Connections“, wobei *Verbindungsname* der Name Ihrer Verbindung (siehe „Erstellen eines neuen Verbindungstyps“ auf Seite 81) und „.erw“ die Standarderweiterung für das Servermodell ist.

Argumente
Keine.

Rückgabewerte
Der HTML-Quellcode für eine Verbindung. Dreamweaver schließt auch das Verbindungsdialogfeld. Wenn eine Feldüberprüfung einen Fehler ergibt, zeigt `applyConnection()` eine Fehlermeldung an und gibt einen leeren String zurück, um anzugeben, dass das Dialogfeld nicht geschlossen werden soll.

Generierte Include-Datei

In der von `applyConnection()` generierten Include-Datei werden alle Eigenschaften einer Verbindung deklariert. Der Dateiname der Include-Datei ist der Verbindungsname. Die Datei hat die Dateinamenerweiterung, die für das mit der aktuellen Site verknüpfte Servermodell definiert ist.

Hinweis: *Verbindungen werden gemeinsam verwendet. Setzen Sie daher den Wert für* `allowMultiple` *auf* `false`*. Dadurch wird sichergestellt, dass die Verbindungsdatei nur einmal in das Dokument eingefügt wird. Zudem wird gewährleistet, dass das Serverskript auch in der Seite bleibt, wenn andere Serververhalten es verwenden.*

In den folgenden Abschnitten finden Sie einige Beispiele für Include-Dateien, die von `applyConnection()` für verschiedene Standardservermodelle generiert werden.

Hinweis: *Um ein Dateiformat für eine Include-Datei zu erstellen, definieren Sie eine neue EDML-Zuordnungsdatei, z. B. „connection_includefile.edml“ (siehe dazu „Definitionsdatei für den Verbindungstyp“ auf Seite 86).*

ASP JavaScript

Die ASP- und JavaScript-Include-Datei sollte den Namen „myConnection1.asp“ haben, wobei „myConnection1“ der Name der Verbindung ist. Das folgende Beispiel ist eine Include-Datei für einen ADO-Verbindungsstring:

```
<%
    // Filename="Connection_ado_conn_string.htm"
    // Type="ADO"
    // HTTP="true"
    // Catalog=""
    // Schema=""
    var MM_MyConnection1_STRING = "dsn=pubs";
%>
```

Die Serververhaltendatei schließt diese Verbindung mit ein, indem sie die relative Include-Anweisung verwendet (siehe folgendes Beispiel).

```
<!--#include file="../Connections/MyConnection1.asp"-->
```

ColdFusion

Bei Verwendung von UltraDev 4 ColdFusion ruft Dreamweaver mithilfe einer ColdFusion Include-Datei eine Liste von Datenquellen ab.

Hinweis: *Bei Dreamweaver-ColdFusion ignoriert Dreamweaver normalerweise alle Include-Dateien und ruft die Liste der Datenquellen stattdessen über RDS von ColdFusion ab.*

Die Include-Datei für UltraDev 4 ColdFusion sollte den Namen „myConnection1.cfm“ haben, wobei „myConnection1“ der Name der Verbindung ist. Im folgenden Beispiel ist die Include-Datei für eine ColdFusion-Verbindung mit einer Produkttabelle dargestellt:

```
<!-- FileName="Connection_cf_dsn.htm" "dsn=products" -->
<!-- Type="ADO" -->
<!-- Catalog="" -->
<!-- Schema="" -->
<!-- HTTP="false" -->
<CFSET MM_MyConnection1_DSN        = "products">
<CFSET MM_MyConnection1_USERNAME   = "">
<CFSET MM_Product_USERNAME         = "">
<CFSET MM_MyConnection1_PASSWORD     = "">
```

Die Serververhaltendatei schließt diese Verbindung mit ein, indem sie die Anweisung `cfinclude` verwendet (siehe folgendes Beispiel).

```
<cfinclude template="Connections/MyConnection1.cfm">
```

Definitionsdatei für den Verbindungstyp

Für jedes Servermodell gibt es eine Datei „connection_includefile.edml“, die den Verbindungstyp definiert und die in der Include-Datei definierten Eigenschaften den entsprechenden Elementen in der Benutzeroberfläche von Dreamweaver zuordnet.

Dreamweaver enthält Standarddefinitionsdateien, d. h. eine Datei für jedes der vordefinierten Servermodelle, die in der folgenden Tabelle aufgeführt sind.

Servermodell	Unterordner im Ordner „Configuration/Connections“
ASP JavaScript	ASP_Js
ASP.NET CSharp	ASP.NET_Csharp
ASP.NET VBScript	ASP.NET_VB
ASP VBScript	ASP_Vbs
ColdFusion	ColdFusion
Java Server Pages	JSP
PHP MySql	PHP_MySql

Dreamweaver ermittelt Verbindungsblöcke mit den Parametern `quickSearch` und `searchPattern` und erstellt Verbindungsblöcke mit dem Parameter `insertText`. Weitere Informationen zu EDML-Tags und EDML-Attributen sowie zu Suchmustern für reguläre Ausdrücke finden Sie unter „Serververhalten“ im Handbuch *Dreamweaver erweitern*.

Hinweis: *Wenn Sie das Format Ihrer Include-Datei ändern oder eine Include-Datei für ein neues Servermodell definieren, müssen Sie die Verbindungsparameter der Dreamweaver-Benutzeroberfläche sowie den Ansichten „Live Data“ und „Vorschau in Browser“ zuordnen. Im folgenden Beispiel für eine EDML-Datei, die mit dem ASP-JS-Standardservermodell verknüpft ist, wurden alle Variablen der Verbindungsseite den entsprechenden Werten zugeordnet, bevor die Seite an den Server gesendet wurde. Weitere Informationen über EDML und Suchmuster für reguläre Ausdrücke finden Sie unter „Serververhalten“ im Handbuch „Dreamweaver erweitern“.*

```
<participant name="connection_includefile" version="5.0">
    <quickSearch>
        <![CDATA[// HTTP=]]></quickSearch>
    <insertText location="">
<![CDATA[<%
// FileName="@@filename@@"
// Type="@@type@@" @@designtimeString@@
// DesigntimeType="@@designtimeType@@"
// HTTP="@@http@@"
// Catalog="@@catalog@@"
// Schema="@@schema@@"
var MM_@@cname@@_STRING = @@string@@
%>
]]>
    </insertText>
    <searchPatterns whereToSearch="directive">
        <searchPattern paramNames="filename">
            <![CDATA[/\/\/\s*FileName="([^"]*)"/]]></searchPattern>
        <searchPattern paramNames="type,designtimeString">
            <![CDATA[/\/\/\s+Type="(\w*)"([^\r\n]*)/]]></searchPattern>
        <searchPattern paramNames="designtimeType" isOptional="true">
            <![CDATA[/\/\/\s*DesigntimeType="(\w*)"/]]></searchPattern>
        <searchPattern paramNames="http">
            <![CDATA[/\/\/\s*HTTP="(\w+)"/]]></searchPattern>
        <searchPattern paramNames="catalog">
            <![CDATA[/\/\/\s*Catalog="(\w*)"/]]></searchPattern>
        <searchPattern paramNames="schema">
            <![CDATA[/\/\/\s*Schema="(\w*)"/]]></searchPattern>
        <searchPattern paramNames="cname,string">
            <![CDATA[/var\s+MM_(\w*)_STRING\s*=\s*([^\r\n]+)/]]></searchPattern>
    </searchPatterns>
</participant>
```

Token in einer EDML-Datei (z. B. `@@filename@@` in diesem Beispiel) ordnen Werte in der Include-Datei den jeweiligen Eigenschaften eines Verbindungsobjekts zu. Die Eigenschaften von Verbindungsobjekten legen Sie in der JavaScript-Implementierungsdatei fest.

Alle im Lieferumfang von Dreamweaver enthaltenen Dialogfelder verwenden die Zuordnungsdatei „connection_includefile.edml“. Damit Dreamweaver diese Datei finden kann, wird der Dateiname in der JavaScript-Implementierungsdatei festgelegt, wie im folgenden Beispiel dargestellt ist.

```
var PARTICIPANT_FILE = "connection_includefile";
```

Beim Erstellen eines benutzerdefinierten Verbindungstyps können Sie in Ihren Dialogfeldern eine beliebige Zuordnungsdatei angeben. Wenn Sie eine Zuordnungsdatei erstellen, können Sie für Ihre EDML-Datei einen von „connection_includefile“ abweichenden Namen verwenden. Wenn Sie einen anderen Namen verwenden, müssen Sie diesen auch in der JavaScript-Implementierungsdatei definieren, wenn Sie den Wert für die Variable `PARTICIPANT_FILE` angeben, wie im folgenden Beispiel dargestellt ist.

```
var PARTICIPANT_FILE = "myConnection_mappingfile";
```

Kapitel 10: API zur Integration der Quellcodeverwaltung

Mit der API zur Integration der Quellcodeverwaltung können Sie gemeinsam genutzte Bibliotheken programmieren. Mit diesen API-Funktionen können Sie das Ein- und Auschecken in Adobe® Dreamweaver® unter Verwendung von Quellcode-Verwaltungssystemen (z. B. Sourcesafe oder CVS) erweitern.

Ihre Bibliotheken müssen eine Mindestanzahl erforderlicher API-Funktionen unterstützen, damit Dreamweaver in ein Quellcode-Verwaltungssystem integriert werden kann. Außerdem müssen sich Ihre Bibliotheken im Ordner „Programme/Adobe/Adobe Dreamweaver CS5/Configuration/SourceControl“ befinden.

Beim Start von Dreamweaver werden die einzelnen Bibliotheken geladen. Dreamweaver ruft `GetProcAddress()` für jede API-Funktion auf, um zu ermitteln, welche Funktionen die Bibliothek unterstützt. Wenn keine Adresse vorhanden ist, nimmt Dreamweaver an, dass die API nicht von dieser Bibliothek unterstützt wird. Wenn die Adresse vorhanden ist, verwendet Dreamweaver die Bibliothek-Version der Funktion, um die Funktionalität zu unterstützen. Wenn ein Dreamweaver-Benutzer eine Site definiert oder bearbeitet und anschließend die Registerkarte „Web Server SCS“ auswählt, werden die Auswahlmöglichkeiten angezeigt, die den DLLs entsprechen. Diese Auswahlmöglichkeiten werden zusätzlich zu den Standardelementen auf der Registerkarte angezeigt. Die DLLs werden aus dem Ordner „Programme/Adobe/Adobe Dreamweaver CS5/Configuration/SourceControl“ geladen.

Wenn Sie ein Menü „Site“ > „Source Control“ (Quellcodeverwaltung) erstellen möchten, dem Sie benutzerdefinierte Elemente hinzufügen können, verwenden Sie den folgenden Code. Fügen Sie den Code dem Menü „Site“ in der Datei „menus.xml“ hinzu:

```
<menu name="Source Control" id="DWMenu_MainSite_Site_Source¬
Control"><menuitem dynamic name="None"file="Menus/MM/¬
File_SCSItems.htm" id="DWMenu_MainSite_Site_NewFeatures_¬
Default" />
</menu>
```

Funktionsweise der Integration der Quellcodeverwaltung in Dreamweaver

Wenn ein Dreamweaver-Benutzer eine Serververbindungs-, Dateiübertragungs- oder Design Notes-Funktion auswählt, ruft Dreamweaver die DLL-Version der entsprechenden API-Funktion auf (`Connect()`, `Disconnect()`, `Get()`, `Put()`, `Checkin()`, `Checkout()`, `Undocheckout()` und `Synchronize()`). Die Anforderung wird von der DLL abgewickelt; dazu gehören beispielsweise auch die Anzeige von Dialogfeldern, in denen Informationen erfasst werden, sowie Benutzerinteraktionen mit der DLL. Die DLL zeigt außerdem Informationen und Fehlermeldungen an.

Das Quellcodeverwaltungssystem kann optional Design Notes und das Ein- und Auschecken unterstützen. In Dreamweaver aktiviert der Benutzer Design Notes für Quellcode-Verwaltungssysteme, indem er im Dialogfeld „Site-Definition“ auf der Registerkarte „Erweitert“ die Kategorie „Design Notes“ auswählt und dann das entsprechende Kontrollkästchen aktiviert. So werden Design Notes auch über FTP und LANs aktiviert. Wenn Design Notes vom Quellcode-Verwaltungssystem nicht unterstützt werden und der Benutzer diese Funktion dennoch verwenden möchte, werden Design Notes-Dateien (MNO-Dateien) von Dreamweaver übertragen, damit die Design Notes erhalten bleiben (wie auch über FTP und LANs).

Das Ein- und Auschecken wird anders behandelt als die Design Notes-Funktion: Wenn das Quellcode-Verwaltungssystem Ein- und Auschecken unterstützt, kann der Benutzer die Verwendung nicht vom Dialogfeld „Site-Definition“ aus außer Kraft setzen. Wenn der Benutzer versucht, das Quellcode-Verwaltungssystem außer Kraft zu setzen, wird eine Fehlermeldung angezeigt.

Hinzufügen von Quellcode-Verwaltungssystemfunktionen

Sie können Dreamweaver Quellcode-Verwaltungssystemfunktionen hinzufügen, indem Sie einen `GetNewFeatures`-Handler programmieren, der eine Gruppe von Menüelementen und entsprechenden C-Funktionen zurückgibt. Wenn Sie beispielsweise eine Sourcesafe-Bibliothek programmieren und Dreamweaver-Benutzern ermöglichen möchten, den Verlauf einer Datei anzuzeigen, können Sie einen Handler `GetNewFeatures` schreiben, der das Menüelement „Verlauf“ und den C-Funktionsnamen `history` zurückgibt. Wenn der Benutzer dann in Windows mit der rechten Maustaste auf eine Datei klickt, wird das Menüelement „Verlauf“ im Kontextmenü aufgeführt. Wenn ein Benutzer das Menüobjekt „Verlauf“ auswählt, ruft Dreamweaver die entsprechende Funktion auf und die ausgewählten Dateien werden an die DLL übergeben. Die DLL zeigt das Dialogfeld „Verlauf“ an, damit der Benutzer damit genauso arbeiten kann wie bei Sourcesafe.

Erforderliche Funktionen der API zur Integration der Quellcodeverwaltung

Die API zur Integration der Quellcodeverwaltung verfügt über erforderliche und optionale Funktionen. Bei den hier aufgeführten Funktionen handelt es sich um erforderliche Funktionen.

bool SCS_GetAgentInfo()

Beschreibung
Diese Funktion fordert die DLL auf, ihren Namen und eine Beschreibung zurückzugeben. Diese Informationen werden im Dialogfeld „Site-Definition“ angezeigt. Der Name wird im Popupmenü „Zugriff“ angezeigt (z. B. Sourcesafe, WebDav, Perforce) und die Beschreibung direkt unter dem Popupmenü.

Argumente
*char name[32], char version[32], char description[256], const char *dwAppVersion*

- Das Argument *name* ist der Name des Quellcode-Verwaltungssystems. Der Name wird im Dialogfeld „Site-Definition“ auf der Registerkarte „Erweitert“ in der Kategorie „Remote-Informationen“ im Popupmenü zum Auswählen eines Quellcode-Verwaltungssystems angezeigt. Der Name kann aus maximal 32 Zeichen bestehen.
- Das Argument *version* ist ein String, der die Version der DLL angibt. Die Version wird im Dialogfeld „Site-Definition“ auf der Registerkarte „Erweitert“ in der Kategorie „Remote-Informationen“ angezeigt. Die Version kann aus maximal 32 Zeichen bestehen.
- Das Argument *description* ist ein String, der die Beschreibung des Quellcode-Verwaltungssystems enthält. Die Beschreibung wird im Dialogfeld „Site-Definition“ auf der Registerkarte „Erweitert“ in der Kategorie „Remote-Informationen“ angezeigt. Die Beschreibung kann aus maximal 256 Zeichen bestehen.

- Das Argument *dwAppVersion* ist ein String, der die Dreamweaver-Version angibt, die die DLL aufruft. Die DLL kann anhand dieses Strings die Version und Sprache von Dreamweaver bestimmen.

Rückgabewerte
Ein boolescher Wert: `true`, wenn erfolgreich, andernfalls `false`.

bool SCS_Connect()

Beschreibung
Diese Funktion verbindet den Benutzer mit dem Quellcode-Verwaltungssystem. Wenn die DLL nicht über Anmeldungsinformationen verfügt, muss die DLL ein Dialogfeld anzeigen, in dem der Benutzer aufgefordert wird, die Informationen einzugeben. Sie muss außerdem die Daten für die spätere Verwendung speichern.

Argumente
*void **connectionData, const char siteName[64]*

- Das Argument *connectionData* ist ein Handle zu den Daten, die beim Aufrufen anderer API-Funktionen von Dreamweaver an den Agenten übergeben werden sollen.
- Das Argument *siteName* ist ein String, der auf den Namen der Site verweist. Der Sitename kann aus maximal 64 Zeichen bestehen.

Rückgabewerte
Ein boolescher Wert: `true`, wenn erfolgreich, andernfalls `false`.

bool SCS_Disconnect()

Beschreibung
Diese Funktion trennt die Verbindung des Benutzers zum Quellcode-Verwaltungssystem.

Argumente
*void *connectionData*

- Das Argument *connectionData* ist ein Zeiger auf die Daten des Agenten, die während des Aufrufs von `Connect()` an Dreamweaver übergeben wurden.

Rückgabewerte
Ein boolescher Wert: `true`, wenn erfolgreich, andernfalls `false`.

bool SCS_IsConnected()

Beschreibung
Diese Funktion bestimmt den Status der Verbindung.

Argumente
*void *connectionData*

- Das Argument *connectionData* ist ein Zeiger auf die Daten des Agenten, die während des Aufrufs von `Connect()` an Dreamweaver übergeben wurden.

Rückgabewerte
Ein boolescher Wert: `true`, wenn erfolgreich, andernfalls `false`.

int SCS_GetRootFolderLength()

Beschreibung
Diese Funktion gibt die Länge des Stammordnernamens zurück.

Argumente
*void *connectionData*

- Das Argument *connectionData* ist ein Zeiger auf die Daten des Agenten, die während des Aufrufs von `Connect()` an Dreamweaver übergeben wurden.

Rückgabewerte
Eine Ganzzahl, die die Länge des Stammordnernamens angibt. Ist der Rückgabewert der Funktion `< 0`, wird dies von Dreamweaver als Fehler interpretiert, und das Programm versucht, die Fehlermeldung von der DLL abzurufen, sofern dies unterstützt wird.

bool SCS_GetRootFolder()

Beschreibung
Diese Funktion gibt den Namen des Stammordners zurück.

Argumente
*void *connectionData, char remotePath[], const int folderLen*

- Das Argument *connectionData* ist ein Zeiger auf die Daten des Agenten, die während des Aufrufs von `Connect()` an Dreamweaver übergeben wurden.
- *remotePath* ist ein Puffer, in dem der vollständige Remote-Pfad des Stammordners gespeichert wird.
- Das Argument *folderLen* ist eine Ganzzahl, die die Länge von *remotePath* angibt. Dies ist der von *GetRootFolderLength* zurückgegebene Wert.

Rückgabewerte
Ein boolescher Wert: `true`, wenn erfolgreich, andernfalls `false`.

int SCS_GetFolderListLength()

Beschreibung
Diese Funktion gibt die Anzahl der Elemente im übergebenen Ordner zurück.

Argumente
*void *connectionData, const char *remotePath*

- Das Argument *connectionData* ist ein Zeiger auf die Daten des Agenten, die während des Aufrufs von `Connect()` an Dreamweaver übergeben wurden.
- Das Argument *remotePath* ist der vollständige Pfad und Name des Remote-Ordners, den die DLL auf die Anzahl der Elemente prüft.

Rückgabewerte
Eine Ganzzahl, die die Anzahl der Elemente im aktuellen Ordner angibt. Ist der Rückgabewert der Funktion `< 0`, wird dies von Dreamweaver als Fehler interpretiert, und das Programm versucht, die Fehlermeldung von der DLL abzurufen, sofern dies unterstützt wird.

bool SCS_GetFolderList()

Beschreibung
Diese Funktion gibt eine Liste der Dateien und Ordner im übergebenen Ordner zurück, einschließlich zugehöriger Informationen wie Größe, Datum der letzten Änderung und der Angabe, ob es sich um einen Ordner oder eine Datei handelt.

Argumente
*void *connectionData, const char *remotePath, itemInfo itemList[ ], const int numItems*

- Das Argument *connectionData* ist ein Zeiger auf die Daten des Agenten, die während des Aufrufs von `Connect()` an Dreamweaver übergeben wurden.
- Das Argument *remotePath* ist der Pfadname des Remote-Ordners, den die DLL auf die Anzahl der Elemente prüft.
- Das Argument *itemList* ist eine bereits zugewiesene Liste mit `itemInfo`-Strukturen:

name	char[256]	Name der Datei oder des Ordners
`isFolder`	bool	`true`, wenn Ordner; `false`, wenn Datei
`month`	int	Komponente Monat von Änderungsdatum 1–12
`day`	int	Komponente Tag von Änderungsdatum 1–31
`year`	int	Komponente Jahr von Änderungsdatum 1900+
`hour`	int	Komponente Stunde von Änderungsdatum 0–23
`minutes`	int	Komponente Minute von Änderungsdatum 0–59
`seconds`	int	Komponente Sekunde von Änderungsdatum 0–59
`type`	char[256]	Dateityp (falls nicht durch die DLL festgelegt, bestimmt Dreamweaver den Typ wie bisher anhand der Dateierweiterung)
`size`	int	In Byte

- Das Argument *numItems* ist die Anzahl der für `itemList` zugewiesenen Elemente (zurückgegeben von `GetFolderListLength`).

Rückgabewerte
Ein boolescher Wert: `true`, wenn erfolgreich, andernfalls `false`.

bool SCS_Get()

Beschreibung
Diese Funktion ruft eine Liste mit Dateien oder Ordnern ab und speichert sie lokal.

Argumente
*void *connectionData, const char *remotePathList[], const char *localPathList[], const int numItems*

- Das Argument *connectionData* ist ein Zeiger auf die Daten des Agenten, die während des Aufrufs von `Connect()` an Dreamweaver übergeben wurden.
- Das Argument *remotePathList* ist eine Liste der abzurufenden Remote-Dateien oder Remote-Ordner. Die Liste enthält die vollständigen Pfade und Namen.
- Das Argument *localPathList* ist eine gespiegelte Liste der Namen lokaler Dateien oder Ordnerpfade.
- Das Argument *numItems* ist die Anzahl der Elemente in jeder Liste.

Rückgabewerte
Ein boolescher Wert: `true`, wenn erfolgreich, andernfalls `false`.

bool SCS_Put()

Beschreibung
Diese Funktion stellt eine Liste mit lokalen Dateien oder Ordnern im Quellcode-Verwaltungssystem bereit.

Argumente
*void *connectionData, const char *localPathList[], const char *remotePathList[], const int numItems*

- Das Argument *connectionData* ist ein Zeiger auf die Daten des Agenten, die während des Aufrufs von `Connect()` an Dreamweaver übergeben wurden.
- Das Argument *localPathList* ist die Liste mit Namen der lokalen Dateien oder Pfaden zu lokalen Ordnern, die im Quellcode-Verwaltungssystem bereitgestellt werden sollen.
- Das Argument *remotePathList* ist eine gespiegelte Liste mit Namen von Remote-Dateien oder Pfaden zu Remote-Ordnern.
- Das Argument *numItems* ist die Anzahl der Elemente in jeder Liste.

Rückgabewerte
Ein boolescher Wert: `true`, wenn erfolgreich, andernfalls `false`.

bool SCS_NewFolder()

Beschreibung
Mit dieser Funktion wird ein neuer Ordner erstellt.

Argumente
*void *connectionData, const char *remotePath*

- Das Argument *connectionData* ist ein Zeiger auf die Daten des Agenten, die während des Aufrufs von `Connect()` an Dreamweaver übergeben wurden.
- Das Argument *remotePath* ist der vollständige Pfadname des Remote-Ordners, der von der DLL erstellt wird.

Rückgabewerte
Ein boolescher Wert: `true`, wenn erfolgreich, andernfalls `false`.

bool SCS_Delete()

Beschreibung
Diese Funktion löscht eine Liste mit Dateien oder Ordnern aus dem Quellcode-Verwaltungssystem.

Argumente
*void *connectionData, const char *remotePathList[], const int numItems*

- Das Argument *connectionData* ist ein Zeiger auf die Daten des Agenten, die während des Aufrufs von `Connect()` an Dreamweaver übergeben wurden.
- Das Argument *remotePathList* ist eine Liste der Namen zu löschender Remote-Dateien bzw. der Pfade zu löschender Remote-Ordner.
- Das Argument *numItems* ist die Anzahl der Elemente in *remotePathList*.

Rückgabewerte
Ein boolescher Wert: `true`, wenn erfolgreich, andernfalls `false`.

bool SCS_Rename()

Beschreibung
Diese Funktion bewirkt, dass eine Datei oder ein Ordner umbenannt oder verschoben wird, je nach den für *oldRemotePath* und *newRemotePath* angegebenen Werten. Ist beispielsweise *oldRemotePath* gleich `"$/folder1/file1"` und *newRemotePath* gleich `"$/folder1/renamefile1"`, wird „file1“ in „renamefile1“ umbenannt und im gleichen Ordner wie bisher („folder1“) abgelegt.

Ist *oldRemotePath* gleich `"$/folder1/file1"` und *newRemotePath* gleich `"$/folder1/subfolder1/file1"`, wird „file1“ in den Ordner „subfolder1“ verschoben.

Um herauszufinden, ob es sich bei dem Aufruf dieser Funktion um eine Verschiebung oder um eine Umbenennung handelt, überprüfen Sie die übergeordneten Pfade der beiden Eingabewerte. Wenn sie identisch sind, handelt es sich bei dem Vorgang um eine Umbenennung.

Argumente
*void *connectionData, const char *oldRemotePath, const char *newRemotePath*

- Das Argument *connectionData* ist ein Zeiger auf die Daten des Agenten, die während des Aufrufs von `Connect()` an Dreamweaver übergeben wurden.

- Das Argument *oldRemotePath* ist der Pfad des umzubenennenden Remote-Ordners bzw. der umzubenennenden Remote-Datei.
- Das Argument *newRemotePath* ist der Remote-Pfad des neuen Namens für die Datei oder den Ordner.

Rückgabewerte
Ein boolescher Wert: `true`, wenn erfolgreich, andernfalls `false`.

bool SCS_ItemExists()

Beschreibung
Diese Funktion bestimmt, ob eine Datei bzw. ein Ordner auf dem Server vorhanden ist.

Argumente
*void *connectionData, const char *remotePath*

- Das Argument *connectionData* ist ein Zeiger auf die Daten des Agenten, die während des Aufrufs von `Connect()` an Dreamweaver übergeben wurden.
- Das Argument *remotePath* ist der Pfad zu einer Remote-Datei bzw. einem Remote-Ordner.

Rückgabewerte
Ein boolescher Wert: `true`, wenn erfolgreich, andernfalls `false`.

Optionale Funktionen der API zur Integration der Quellcodeverwaltung

Die API zur Integration der Quellcodeverwaltung verfügt über erforderliche und optionale Funktionen. Bei den hier aufgeführten Funktionen handelt es sich um optionale Funktionen.

bool SCS_GetConnectionInfo()

Beschreibung
Diese Funktion zeigt ein Dialogfeld an, in dem der Benutzer die Verbindungsinformationen für diese Site ändern oder festlegen kann. Eine Verbindung wird nicht hergestellt. Diese Funktion wird aufgerufen, wenn der Benutzer im Dialogfeld „Site-Definition“ auf der Registerkarte „Erweitert“ in der Kategorie „Remote-Informationen“ auf die Schaltfläche „Einstellungen“ klickt.

Argumente
*void **connectionData, const char siteName[64]*

- Das Argument *connectionData* ist ein Handle zu den Daten, die beim Aufrufen anderer API-Funktionen von Dreamweaver an den Agenten übergeben werden sollen.
- Das Argument *siteName* ist ein String, der auf den Namen der Site verweist. Der Name darf höchstens 64 Zeichen enthalten.

Rückgabewerte
Ein boolescher Wert: `true`, wenn erfolgreich, andernfalls `false`.

bool SCS_SiteDeleted()

Beschreibung
Diese Funktion teilt der DLL mit, dass die Site gelöscht wurde oder nicht mehr mit diesem Quellcode-Verwaltungssystem verbunden ist. Dies bedeutet, dass das Quellcode-Verwaltungssystem die entsprechenden Daten für diese Site löschen kann.

Argumente
const char siteName[64]

- Das Argument *siteName* ist ein String, der auf den Namen der Site verweist. Der Name darf höchstens 64 Zeichen enthalten.

Rückgabewerte
Ein boolescher Wert: `true`, wenn erfolgreich, andernfalls `false`.

bool SCS_SiteRenamed()

Beschreibung
Diese Funktion benachrichtigt die DLL, wenn der Benutzer die Site umbenannt hat, damit die entsprechenden Informationen zur Site in der DLL aktualisiert werden können.

Argumente
const char oldSiteName[64], const char newSiteName[64]

- Das Argument *oldSiteName* ist ein String, der auf den ursprünglichen Namen der Site verweist. Dies ist der Name der Site vor der Umbenennung. Der Name darf höchstens 64 Zeichen enthalten.
- Das Argument *newSiteName* ist ein String, der auf den neuen Namen der Site verweist. Dies ist der Name der Site nach der Umbenennung. Der Name darf höchstens 64 Zeichen enthalten.

Rückgabewerte
Ein boolescher Wert: `true`, wenn erfolgreich, andernfalls `false`.

int SCS_GetNumNewFeatures()

Beschreibung
Diese Funktion gibt die Zahl der neuen Funktionen zurück, die Dreamweaver hinzugefügt werden (z. B. Dateiverlauf, Unterschiede usw.).

Argumente
Keine.

Rückgabewerte
Eine Ganzzahl, die die Anzahl der neuen Funktionen angibt, die Dreamweaver hinzugefügt werden. Ist der Rückgabewert der Funktion `< 0`, wird dies von Dreamweaver als Fehler interpretiert, und das Programm versucht, die Fehlermeldung von der DLL abzurufen, sofern dies unterstützt wird.

bool SCS_GetNewFeatures()

Beschreibung
Diese Funktion gibt eine Liste mit Menüelementen zurück, die in das Hauptmenü und in die Kontextmenüs von Dreamweaver eingefügt werden sollen. Beispielsweise können mit der Sourcesafe-DLL Verlauf und Dateiunterschiede in das Hauptmenü eingefügt werden.

Argumente
char menuItemList[][32], scFunction functionList[], scFunction enablerList[], const int numNewFeatures

- Das Argument *menuItemList* ist eine Liste mit Strings, die von der DLL ausgefüllt wird. Sie gibt die Menüelemente an, die dem Hauptmenü und den Kontextmenüs hinzugefügt werden sollen. Jeder String kann höchstens 32 Zeichen enthalten.
- Das Argument *functionList* wird von der DLL gefüllt. Es legt fest, welche Routinen in der DLL aufgerufen werden, wenn der Benutzer das entsprechende Menüelement auswählt.
- Das Argument *enablerList* wird von der DLL gefüllt. Es legt fest, welche Routinen in der DLL aufgerufen werden, wenn Dreamweaver bestimmen muss, ob das entsprechende Menüelement aktiviert ist.
- Das Argument *numNewFeatures* ist die Anzahl der Elemente, die von der DLL hinzugefügt werden. Dieser Wert wird vom Aufruf `GetNumNewFeatures()` abgerufen.

  Die folgende Funktionssignatur definiert die Funktionen und Enabler, die mit den Argumenten `functionlist` und `enablerList` an den Aufruf von `SCS_GetNewFeatures()` übergeben werden.

```
bool (*scFunction)(void *connectionData, const char *remotePathList[],
    const char *localPathList[], const int numItems)
```

Rückgabewerte
Ein boolescher Wert: `true`, wenn erfolgreich, andernfalls `false`.

bool SCS_GetCheckoutName()

Beschreibung
Diese Funktion gibt den Auschecknamen des aktuellen Benutzers zurück. Falls diese Funktion vom Quellcode-Verwaltungssystem nicht unterstützt wird, aber vom Benutzer aktiviert wurde, wird die interne Dreamweaver-Funktion „Einchecken/Auschecken“ verwendet, mit der LCK-Dateien in das Quellcode-Verwaltungssystem und aus dem Quellcode-Verwaltungssystem übertragen werden.

Argumente
*void *connectionData, char checkOutName[64], char emailAddress[64]*

- Das Argument *connectionData* ist ein Zeiger auf die Daten des Agenten, die während des Aufrufs von `Connect()` an Dreamweaver übergeben wurden.
- Das Argument *checkOutName* ist der Name des aktuellen Benutzers.

- Das Argument *emailAddress* ist die E-Mail-Adresse des aktuellen Benutzers.

Rückgabewerte
Ein boolescher Wert: `true`, wenn erfolgreich, andernfalls `false`.

bool SCS_Checkin()

Beschreibung
Diese Funktion checkt eine Liste mit lokalen Dateien oder Ordnern in das Quellcode-Verwaltungssystem ein. Die DLL ist verantwortlich für das Einstellen des Schreibschutzes für die Datei. Falls diese Funktion vom Quellcode-Verwaltungssystem nicht unterstützt wird, aber vom Benutzer aktiviert wurde, wird die interne Dreamweaver-Funktion „Einchecken/Auschecken" verwendet, mit der LCK-Dateien in das Quellcode-Verwaltungssystem und aus dem Quellcode-Verwaltungssystem übertragen werden.

Argumente
*void *connectionData, const char *localPathList[], const char *remotePathList[], bool successList[], const int numItems*

- Das Argument *connectionData* ist ein Zeiger auf die Daten des Agenten, die während des Aufrufs von `Connect()` an Dreamweaver übergeben wurden.
- Das Argument *localPathList* ist eine Liste der Namen einzucheckender lokaler Dateien oder Ordnerpfade.
- Das Argument *remotePathList* ist eine gespiegelte Liste mit Namen von Remote-Dateien oder Pfaden zu Remote-Ordnern.
- Das Argument *successList* ist eine Liste mit booleschen Werten, die von der DLL angegeben werden, um Dreamweaver mitzuteilen, welche der zugehörigen Dateien erfolgreich eingecheckt wurden.
- Das Argument *numItems* ist die Anzahl der Elemente in jeder Liste.

Rückgabewerte
Ein boolescher Wert: `true`, wenn erfolgreich, andernfalls `false`.

bool SCS_Checkout()

Beschreibung
Diese Funktion checkt eine Liste lokaler Dateien oder Ordner aus dem Quellcode-Verwaltungssystem aus. Die DLL ist verantwortlich dafür, die Schreibberechtigungen für die Datei zu gewähren. Falls diese Funktion vom Quellcode-Verwaltungssystem nicht unterstützt wird, aber vom Benutzer aktiviert wurde, wird die interne Dreamweaver-Funktion „Einchecken/Auschecken" verwendet, mit der LCK-Dateien in das Quellcode-Verwaltungssystem und aus dem Quellcode-Verwaltungssystem übertragen werden.

Argumente
*void *connectionData, const char *remotePathList[], const char *localPathList[], bool successList[], const int numItems*

- Das Argument *connectionData* ist ein Zeiger auf die Daten des Agenten, die während des Aufrufs von `Connect()` an Dreamweaver übergeben wurden.
- Das Argument *remotePathList* ist eine Liste der Namen auszucheckender Remote-Dateien bzw. der Pfade auszucheckender Remote-Ordner.
- Das Argument *localPathList* ist eine gespiegelte Liste der Namen lokaler Dateien oder Ordnerpfade.

- Das Argument *successList* ist eine Liste mit booleschen Werten, die von der DLL angegeben werden, um Dreamweaver mitzuteilen, welche der zugehörigen Dateien erfolgreich ausgecheckt wurden.
- Das Argument *numItems* ist die Anzahl der Elemente in jeder Liste.

Rückgabewerte
Ein boolescher Wert: `true`, wenn erfolgreich, andernfalls `false`.

bool SCS_UndoCheckout()

Beschreibung
Diese Funktion macht den Auscheck-Status einer Liste mit Dateien oder Ordnern rückgängig. Die DLL ist verantwortlich für das Einstellen des Schreibschutzes für die Datei. Falls diese Funktion vom Quellcode-Verwaltungssystem nicht unterstützt wird, aber vom Benutzer aktiviert wurde, wird die interne Dreamweaver-Funktion „Einchecken/Auschecken“ verwendet, mit der LCK-Dateien in das Quellcode-Verwaltungssystem und aus dem Quellcode-Verwaltungssystem übertragen werden.

Argumente
*void *connectionData, const char *remotePathList[], const char *localPathList[], bool successList[], const int numItems*

- Das Argument *connectionData* ist ein Zeiger auf die Daten des Agenten, die während des Aufrufs von `Connect()` an Dreamweaver übergeben wurden.
- Das Argument *remotePathList* ist eine Liste der Namen von Remote-Dateien oder Pfadnamen von Remote-Ordnern, für die das Auschecken rückgängig gemacht werden soll.
- Das Argument *localPathList* ist eine gespiegelte Liste der Namen lokaler Dateien oder Ordnerpfade.
- Das Argument *successList* ist eine Liste mit booleschen Werten, die von der DLL angegeben werden, um Dreamweaver mitzuteilen, für welche der zugehörigen Dateien das Auschecken erfolgreich rückgängig gemacht wurde.
- Das Argument *numItems* ist die Anzahl der Elemente in jeder Liste.

Rückgabewerte
Ein boolescher Wert: `true`, wenn erfolgreich, andernfalls `false`.

int SCS_GetNumCheckedOut()

Beschreibung
Diese Funktion gibt die Anzahl der Benutzer zurück, die eine Datei ausgecheckt haben.

Argumente
*void *connectionData, const char *remotePath*

- Das Argument *connectionData* ist ein Zeiger auf die Daten des Agenten, die während des Aufrufs von `Connect()` an Dreamweaver übergeben wurden.
- Das Argument *remotePath* ist der Pfad der Remote-Datei oder des Remote-Ordners, der daraufhin überprüft werden soll, wie viele Benutzer die Datei ausgecheckt haben.

Rückgabewerte
Eine Ganzzahl, die angibt, wie viele Personen die Datei ausgecheckt haben. Ist der Rückgabewert der Funktion `< 0`, wird dies von Dreamweaver als Fehler interpretiert, und das Programm versucht, die Fehlermeldung von der DLL abzurufen, sofern dies unterstützt wird.

bool SCS_GetFileCheckoutList()

Beschreibung
Diese Funktion gibt eine Liste der Benutzer zurück, die eine Datei ausgecheckt haben. Wenn die Liste leer ist, hat kein Benutzer die Datei ausgecheckt.

Argumente
*void *connectionData, const char *remotePath, char checkOutList[][64], char emailAddressList[][64], const int numCheckedOut*

- Das Argument *connectionData* ist ein Zeiger auf die Daten des Agenten, die während des Aufrufs von `Connect()` an Dreamweaver übergeben wurden.
- Das Argument *remotePath* ist der Pfadname der Remote-Datei oder des Remote-Ordners, der daraufhin überprüft werden soll, wie viele Benutzer die Datei ausgecheckt haben.
- Das Argument *checkOutList* ist eine Liste mit Strings, die den Benutzern entsprechen, die die Datei ausgecheckt haben. Die Benutzerstrings können jeweils maximal 64 Zeichen lang sein.
- Das Argument *emailAddressList* ist eine Liste mit Strings, die den E-Mail-Adressen der Benutzer entsprechen. Die Strings für die E-Mail-Adressen können jeweils maximal 64 Zeichen lang sein.
- Das Argument *numCheckedOut* ist die Anzahl der Personen, die die Datei ausgecheckt haben. Diese Zahl wird von `GetNumCheckedOut()` zurückgegeben.

Rückgabewerte
Ein boolescher Wert: `true`, wenn erfolgreich, andernfalls `false`.

int SCS_GetErrorMessageLength()

Beschreibung
Diese Funktion gibt die Länge der aktuellen internen Fehlermeldung der DLL zurück. Dies weist den Puffer zu, der an die Funktion `GetErrorMessage()` übergeben wird. Diese Funktion sollte nur dann aufgerufen werden, wenn eine API-Funktion `false` oder `<0` zurückgibt, wodurch ein Fehler dieser API-Funktion angegeben wird.

Argumente
*void *connectionData*

- Das Argument *connectionData* ist ein Zeiger auf die Daten des Agenten, die während des Aufrufs von `Connect()` an Dreamweaver übergeben wurden.

Rückgabewerte
Eine Ganzzahl, die die Länge der Fehlermeldung angibt.

bool SCS_GetErrorMessage()

Beschreibung
Diese Funktion gibt die letzte Fehlermeldung zurück. Wenn Sie `getErrorMessage()` implementieren, ruft Dreamweaver die Funktion jedes Mal auf, wenn eine Ihrer API-Funktionen den Wert `false` zurückgibt.

Wenn eine Routine `-1` oder `false` zurückgibt, wird dadurch angegeben, dass eine Fehlermeldung verfügbar sein sollte.

Argumente
*void *connectionData, char errorMsg[], const int *msgLength*

- Das Argument *connectionData* ist ein Zeiger auf die Daten des Agenten, die während des Aufrufs von `Connect()` an Dreamweaver übergeben wurden.
- Das Argument *errorMsg* ist ein bereits zugewiesener String, der von der DLL mit der Fehlermeldung gefüllt wird.
- Das Argument *msgLength* ist die Länge des Puffers, der durch das *errorMsg[]*-Argument dargestellt wird.

Rückgabewerte
Ein boolescher Wert: `true`, wenn erfolgreich, andernfalls `false`.

int SCS_GetNoteCount()

Beschreibung
Diese Funktion gibt die Anzahl der Design Note-Schlüssel für den angegebenen Pfad der Remote-Datei bzw. des Remote-Ordners zurück. Wenn dies vom Quellcode-Verwaltungssystem nicht unterstützt wird, ruft Dreamweaver diese Informationen aus der zugehörigen MNO-Datei ab.

Argumente
*void *connectionData, const char *remotePath*

- Das Argument *connectionData* ist ein Zeiger auf die Daten des Agenten, die während des Aufrufs von `Connect()` an Dreamweaver übergeben wurden.
- Das Argument *remotePath* ist der Pfadname der Remote-Datei oder des Remote-Ordners, den die DLL auf die Anzahl der angefügten Design Notes überprüft.

Rückgabewerte
Eine Ganzzahl, die die Anzahl der Design Notes angibt, die zu dieser Datei gehören. Ist der Rückgabewert der Funktion `< 0`, wird dies von Dreamweaver als Fehler interpretiert, und das Programm versucht, die Fehlermeldung von der DLL abzurufen, sofern dies unterstützt wird.

int SCS_GetMaxNoteLength()

Beschreibung
Diese Funktion gibt die Länge der größten Design Note für die angegebene Datei bzw. den angegebenen Ordner zurück. Wenn dies vom Quellcode-Verwaltungssystem nicht unterstützt wird, ruft Dreamweaver diese Informationen aus der zugehörigen MNO-Datei ab.

Argumente
*void *connectionData, const char *remotePath*

- Das Argument *connectionData* ist ein Zeiger auf die Daten des Agenten, die während des Aufrufs von `Connect()` an Dreamweaver übergeben wurden.
- Das Argument *remotePath* ist der Pfadname der Remote-Datei oder des Remote-Ordners, den die DLL auf die maximale Design Note-Länge überprüft.

Rückgabewerte
Eine Ganzzahl, die die Größe der längsten Design Note angibt, die zu dieser Datei gehört. Ist der Rückgabewert der Funktion `< 0`, wird dies von Dreamweaver als Fehler interpretiert, und das Programm versucht, die Fehlermeldung von der DLL abzurufen, sofern dies unterstützt wird.

bool SCS_GetDesignNotes()

Beschreibung
Diese Funktion ruft Schlüssel-Wert-Paare aus den Metadaten für die angegebene Datei bzw. den angegebenen Ordner ab. Wenn dies vom Quellcode-Verwaltungssystem nicht unterstützt wird, ruft Dreamweaver diese Informationen aus der zugehörigen MNO-Datei ab.

Argumente
*void *connectionData, const char *remotePath, char keyList[][64], char *valueList[], bool showColumnList[], const int noteCount, const int noteLength*

- Das Argument *connectionData* ist ein Zeiger auf die Daten des Agenten, die während des Aufrufs von `Connect()` an Dreamweaver übergeben wurden.
- Das Argument *remotePath* ist der Pfadname der Remote-Datei oder des Remote-Ordners, den die DLL auf die Anzahl der Elemente überprüft.
- Das Argument *keyList* ist eine Liste der Design Note-Schlüssel, z. B. `"Status"`.
- Das Argument *valueList* ist eine Liste der Design Note-Werte, die den Design Note-Schlüsseln entsprechen, z. B. `"Awaiting Signoff"`.
- Das Argument *showColumnList* ist eine Liste boolescher Werte, die den Design Note-Schlüsseln entsprechen. Damit wird angegeben, ob Dreamweaver den Schlüssel im Bedienfeld „Dateien“ als Spalte anzeigen kann.
- Das Argument *noteCount* ist die Anzahl der Design Notes, die an eine Datei oder einen Ordner angefügt sind. Dieser Wert wird vom Aufruf `GetNoteCount()` zurückgegeben.
- Das Argument *noteLength* ist die maximale Länge einer Design Note. Dies ist der vom Aufruf `GetMaxNoteLength()` zurückgegebene Wert.

Rückgabewerte
Ein boolescher Wert: `true`, wenn erfolgreich, andernfalls `false`.

bool SCS_SetDesignNotes()

Beschreibung
Diese Funktion speichert die Schlüssel-Wert-Paare in den Metadaten für die angegebene Datei bzw. für den angegebenen Ordner. Hierdurch wird das Set von Metadaten für die Datei ersetzt. Wenn dies vom Quellcode-Verwaltungssystem nicht unterstützt wird, speichert Dreamweaver Design Notes in MNO-Dateien.

Argumente
*void *connectionData, const char *remotePath, const char keyList[][64], const char *valueList[], bool showColumnList[], const int noteCount, const int noteLength*

- Das Argument *connectionData* ist ein Zeiger auf die Daten des Agenten, die während des Aufrufs von `Connect()` an Dreamweaver übergeben wurden.
- Das Argument *remotePath* ist der Pfadname der Remote-Datei oder des Remote-Ordners, den die DLL auf die Anzahl der Elemente überprüft.
- Das Argument *keyList* ist eine Liste der Design Note-Schlüssel, z. B. `"Status"`.
- Das Argument *valueList* ist eine Liste der Design Note-Werte, die den Design Note-Schlüsseln entsprechen, z. B. `"Awaiting Signoff"`.
- Das Argument *showColumnList* ist eine Liste boolescher Werte, die den Design Note-Schlüsseln entsprechen. Damit wird angegeben, ob Dreamweaver den Schlüssel im Bedienfeld „Dateien“ als Spalte anzeigen kann.
- Das Argument *noteCount* ist die Anzahl der Design Notes, die an eine Datei oder einen Ordner angefügt sind. Anhand dieser Zahl kann die DLL die Größe der angegebenen Listen ermitteln. Wenn *noteCount* gleich `0` ist, werden alle Design Notes aus der Datei entfernt.
- Das Argument *noteLength* ist die Länge der größten Design Note für die angegebene Datei bzw. den angegebenen Ordner.

Rückgabewerte
Ein boolescher Wert: `true`, wenn erfolgreich, andernfalls `false`.

bool SCS_IsRemoteNewer()

Beschreibung
Diese Funktion prüft jeden angegebenen Remote-Pfad, um festzustellen, ob die Remote-Version neuer ist. Wenn dies vom Quellcode-Verwaltungssystem nicht unterstützt wird, verwendet Dreamweaver seinen internen `isRemoteNewer`-Algorithmus.

Argumente
*void *connectionData, const char *remotePathList[], constchar*localPathList[], int remoteIsNewerList[], const int numItems*

- Das Argument *connectionData* ist ein Zeiger auf die Daten des Agenten, die während des Aufrufs von `Connect()` an Dreamweaver übergeben wurden.
- Das Argument *remotePathList* ist eine Liste der Namen von Remote-Dateien oder Pfadnamen von Remote-Ordnern, die auf einen neueren Status überprüft werden sollen.
- Das Argument *localPathList* ist eine gespiegelte Liste der Namen lokaler Dateien oder Ordnerpfade.

- Das Argument *remoteIsNewerList* ist eine Liste mit Ganzzahlen, die von der DLL ausgefüllt wird und Dreamweaver mitteilt, welche der entsprechenden Dateien auf der Remote-Seite neuer sind. Die folgenden Werte sind gültig: 1 gibt an, dass die Remote-Version neuer ist. –1 gibt an, dass die lokale Version neuer ist. 0 gibt an, dass die beiden Versionen identisch sind.
- Das Argument *numItems* ist die Anzahl der Elemente in jeder Liste.

Rückgabewerte
Ein boolescher Wert: `true`, wenn erfolgreich, andernfalls `false`.

Enabler

Wenn die optionalen Enabler vom Quellcode-Verwaltungssystem nicht unterstützt werden oder wenn die Anwendung nicht mit dem Server verbunden ist, bestimmt Dreamweaver, wann die Menüobjekte aktiviert werden, und zwar anhand der vorliegenden Informationen über die Remote-Dateien.

bool SCS_canConnect()

Beschreibung
Diese Funktion gibt einen Wert zurück, aus dem hervorgeht, ob das Menüelement „Verbindung herstellen“ aktiviert werden soll.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn erfolgreich, andernfalls `false`.

bool SCS_canGet()

Beschreibung
Diese Funktion gibt einen Wert zurück, aus dem hervorgeht, ob das Menüelement „Abrufen“ aktiviert werden soll.

Argumente
*void *connectionData, const char *remotePathList[], const char *localPathList[], const int numItems*

- Das Argument *connectionData* ist ein Zeiger auf die Daten des Agenten, die während des Aufrufs von `Connect()` an Dreamweaver übergeben wurden.
- Das Argument *remotePathList* ist eine Liste der Namen abzurufender Remote-Dateien bzw. der Pfade abzurufender Remote-Ordner.
- Das Argument *localPathList* ist eine gespiegelte Liste der Namen lokaler Dateien oder Ordnerpfade.
- Das Argument *numItems* ist die Anzahl der Elemente in jeder Liste.

Rückgabewerte
Ein boolescher Wert: `true`, wenn erfolgreich, andernfalls `false`.

bool SCS_canCheckout()

Beschreibung
Diese Funktion gibt einen Wert zurück, aus dem hervorgeht, ob das Menüelement „Auschecken“ aktiviert werden soll.

Argumente
*void *connectionData, const char *remotePathList[], const char *localPathList[], const int numItems*

- Das Argument *connectionData* ist ein Zeiger auf die Daten des Agenten, die während des Aufrufs von `Connect()` an Dreamweaver übergeben wurden.
- Das Argument *remotePathList* ist eine Liste der Namen auszucheckender Remote-Dateien bzw. der Pfade auszucheckender Remote-Ordner.
- Das Argument *localPathList* ist eine gespiegelte Liste der Namen lokaler Dateien oder Ordnerpfade.
- Das Argument *numItems* ist die Anzahl der Elemente in jeder Liste.

Rückgabewerte
Ein boolescher Wert: `true`, wenn erfolgreich, andernfalls `false`.

bool SCS_canPut()

Beschreibung
Diese Funktion gibt einen Wert zurück, aus dem hervorgeht, ob das Menüelement „Bereitstellen“ aktiviert werden soll.

Argumente
*void *connectionData, const char *localPathList[], const char *remotePathList[], const int numItems*

- Das Argument *connectionData* ist ein Zeiger auf die Daten des Agenten, die während des Aufrufs von `Connect()` an Dreamweaver übergeben wurden.
- Das Argument *localPathList* ist eine Liste mit Namen der lokalen Dateien oder Pfaden zu lokalen Ordnern, die im Quellcode-Verwaltungssystem bereitgestellt werden sollen.
- Das Argument *localPathList* ist eine gespiegelte Liste mit den Namen der Remote-Dateien oder Pfadnamen der Remote-Ordner, die im Quellcode-Verwaltungssystem bereitgestellt werden sollen.
- Das Argument *numItems* ist die Anzahl der Elemente in jeder Liste.

Rückgabewerte
Ein boolescher Wert: `true`, wenn erfolgreich, andernfalls `false`.

bool SCS_canCheckin()

Beschreibung
Diese Funktion gibt einen Wert zurück, aus dem hervorgeht, ob das Menüelement „Einchecken“ aktiviert werden soll.

Argumente
*void *connectionData, const char *localPathList[], const char *remotePathList[], const int numItems*

- Das Argument *connectionData* ist ein Zeiger auf die Daten des Agenten, die während des Aufrufs von `Connect()` an Dreamweaver übergeben wurden.
- Das Argument *localPathList* ist eine Liste der Namen einzucheckender lokaler Dateien oder Ordnerpfade.
- Das Argument *remotePathList* ist eine gespiegelte Liste mit Namen von Remote-Dateien oder Pfaden zu Remote-Ordnern.
- Das Argument *numItems* ist die Anzahl der Elemente in jeder Liste.

Rückgabewerte
Ein boolescher Wert: `true`, wenn erfolgreich, andernfalls `false`.

bool SCS_CanUndoCheckout()

Beschreibung
Diese Funktion gibt einen Wert zurück, aus dem hervorgeht, ob das Menüelement „Auschecken rückgängig“ aktiviert werden soll.

Argumente
*void *connectionData, const char *remotePathList[], const char *localPathList[], const int numItems*

- Das Argument *connectionData* ist ein Zeiger auf die Daten des Agenten, die während des Aufrufs von `Connect()` an Dreamweaver übergeben wurden.
- Das Argument *remotePathList* ist eine Liste der Namen auszucheckender Remote-Dateien bzw. der Pfade auszucheckender Remote-Ordner.
- Das Argument *localPathList* ist eine Liste mit Namen der lokalen Dateien oder Pfaden zu lokalen Ordnern, die im Quellcode-Verwaltungssystem bereitgestellt werden sollen.
- Das Argument *numItems* ist die Anzahl der Elemente in jeder Liste.

Rückgabewerte
Ein boolescher Wert: `true`, wenn erfolgreich, andernfalls `false`.

bool SCS_canNewFolder()

Beschreibung
Diese Funktion gibt einen Wert zurück, aus dem hervorgeht, ob das Menüelement „Neuer Ordner“ aktiviert werden soll.

Argumente
*void *connectionData, const char *remotePath*

- Das Argument *connectionData* ist ein Zeiger auf die Daten des Agenten, die während des Aufrufs von `Connect()` an Dreamweaver übergeben wurden.
- Das Argument *remotePath* ist eine Liste mit Namen der Remote-Dateien oder Pfaden zu Remote-Ordnern, die der Benutzer ausgewählt hat, um anzugeben, wo der neue Ordner erstellt wird.

Rückgabewerte
Ein boolescher Wert: `true`, wenn erfolgreich, andernfalls `false`.

bool SCS_canDelete()

Beschreibung
Diese Funktion gibt einen Wert zurück, aus dem hervorgeht, ob das Menüelement „Löschen“ aktiviert werden soll.

Argumente
*void *connectionData, const char *remotePathList[], const int numItems*

- Das Argument *connectionData* ist ein Zeiger auf die Daten des Agenten, die während des Aufrufs von `Connect()` an Dreamweaver übergeben wurden.
- Das Argument *remotePathList* ist eine Liste der Namen zu löschender Remote-Dateien bzw. der Pfade zu löschender Remote-Ordner.
- Das Argument *numItems* ist die Anzahl der Elemente in jeder Liste.

Rückgabewerte
Ein boolescher Wert: `true`, wenn erfolgreich, andernfalls `false`.

bool SCS_canRename()

Beschreibung
Diese Funktion gibt einen Wert zurück, aus dem hervorgeht, ob das Menüelement „Umbenennen“ aktiviert werden soll.

Argumente
*void *connectionData, const char *remotePath*

- Das Argument *connectionData* ist ein Zeiger auf die Daten des Agenten, die während des Aufrufs von `Connect()` an Dreamweaver übergeben wurden.
- Das Argument *remotePathList* ist eine Liste der Namen der Remote-Dateien oder Pfade der Remote-Ordner, die umbenannt werden können.

Rückgabewerte
Ein boolescher Wert: `true`, wenn erfolgreich, andernfalls `false`.

bool SCS_BeforeGet()

Beschreibung
Dreamweaver ruft diese Funktion auf, bevor eine oder mehrere Dateien abgerufen oder ausgecheckt werden. Mit dieser Funktion kann Ihre DLL einen Vorgang ausführen und beispielsweise einer Gruppe von Dateien einen Auscheck-Kommentar hinzufügen.

Argumente
**connectionData*

- Das Argument **connectionData* ist ein Zeiger auf die Daten des Agenten, die während des Aufrufs von `Connect()` an Dreamweaver übergeben wurden.

Rückgabewerte
Ein boolescher Wert: `true`, wenn erfolgreich, andernfalls `false`.

Beispiel
Um eine Gruppe von Dateien abzurufen, führt Dreamweaver in der folgenden Reihenfolge Aufrufe an die DLL durch:

```
SCS_BeforeGet(connectionData);
SCS_Get(connectionData,remotePathList1,localPathList1,successList1);
SCS_Get(connectionData,remotePathList2,localPathList2,successList2);
SCS_Get(connectionData,remotePathList3,localPathList3,successList3);
SCS_AfterGet(connectionData);
```

bool SCS_BeforePut()

Beschreibung
Dreamweaver ruft diese Funktion auf, bevor eine oder mehrere Dateien bereitgestellt oder eingecheckt werden. Mit dieser Funktion kann Ihre DLL einen Vorgang ausführen und beispielsweise einer Gruppe von Dateien einen Eincheck-Kommentar hinzufügen.

Argumente
**connectionData*

- Das Argument **connectionData* ist ein Zeiger auf die Daten des Agenten, die während des Aufrufs von `Connect()` an Dreamweaver übergeben wurden.

Rückgabewerte
Ein boolescher Wert: `true`, wenn erfolgreich, andernfalls `false`.

Beispiel
Um eine Gruppe von Dateien abzurufen, führt Dreamweaver in der folgenden Reihenfolge Aufrufe an die DLL durch:

```
SCS_BeforePut(connectionData);
SCS_Put(connectionData,localPathList1,remotePathList1,successList1);
SCS_Put(connectionData,localPathList2,remotePathList2,successList2);
SCS_Put(connectionData,localPathList3,remotePathList3,successList3);
SCS_AfterPut(connectionData);
```

bool SCS_AfterGet()

Beschreibung
Dreamweaver ruft diese Funktion auf, nachdem eine oder mehrere Dateien abgerufen oder ausgecheckt wurden. Mit dieser Funktion kann Ihre DLL nach dem Abrufen oder Auschecken per Stapelverarbeitung einen beliebigen Vorgang ausführen und beispielsweise ein Dialogfeld mit einer Zusammenfassung erstellen.

Argumente
**connectionData*

- Das Argument **connectionData* ist ein Zeiger auf die Daten des Agenten, die während des Aufrufs von `Connect()` an Dreamweaver übergeben wurden.

Rückgabewerte
Ein boolescher Wert: `true`, wenn erfolgreich, andernfalls `false`.

Beispiel
Siehe „bool SCS_BeforeGet()“ auf Seite 108.

bool SCS_AfterPut()

Beschreibung
Dreamweaver ruft diese Funktion auf, nachdem eine oder mehrere Dateien bereitgestellt oder eingecheckt wurden. Mit dieser Funktion kann Ihre DLL nach dem Bereitstellen oder Einchecken per Stapelverarbeitung einen beliebigen Vorgang ausführen und beispielsweise ein Dialogfeld mit einer Zusammenfassung erstellen.

Argumente
**connectionData*

- Das Argument **connectionData* ist ein Zeiger auf die Daten des Agenten, die während des Aufrufs von `Connect()` an Dreamweaver übergeben wurden.

Rückgabewerte
Ein boolescher Wert: `true`, wenn erfolgreich, andernfalls `false`.

Beispiel
Siehe „bool SCS_BeforePut()“ auf Seite 109.

Kapitel 11: Anwendung

Mit den Anwendungsfunktionen werden Vorgänge durchgeführt, die die Interaktion zwischen Adobe® Dreamweaver® und anderen Anwendungen betreffen, oder von einzelnen Dokumenten unabhängige Dreamweaver-Vorgänge, z. B. Festlegen von Voreinstellungen, Beenden von Dreamweaver usw.

Funktionen für externe Anwendungen

Funktionen für externe Anwendungen betreffen Vorgänge in Anwendungen, z. B. Adobe® Flash®, sowie in den Browsern und externen Editoren, die in den Voreinstellungen für „Vorschau in Browser“ und „Externe Editoren“ definiert werden. Mithilfe dieser Funktionen können Sie Informationen zu diesen externen Anwendungen abrufen und Dateien in diesen Anwendungen öffnen.

dreamweaver.browseDocument()

Verfügbarkeit
Dreamweaver 2, in Version 3 und 4 verbessert.

Beschreibung
Öffnet die betreffende URL im angegebenen Browser.

Argumente
`fileName, {browser}`

- Das Argument *`fileName`* ist der Name der zu öffnenden Datei, der als absolute URL angegeben wird.
- Das Argument *`browser`* gibt einen Browser an. Bei diesem Argument kann es sich um den Namen eines Browsers handeln, der in den Voreinstellungen für „Vorschau in Browser“ definiert wurde, oder um `primary` oder `secondary`. Wenn kein Argument angegeben ist, wird die URL im Primärbrowser des Benutzers geöffnet.

Hinweis: *Einige Browser können die Datei nicht finden, wenn die URL einen Anker enthält, z. B. „Configuration/ExtensionHelp/browseHelp.htm#helpyou“.*

Rückgabewerte
Keine.

Beispiel
Die folgende Funktion öffnet mithilfe der Funktion `dreamweaver.browseDocument()` die Homepage von Adobe in einem Browser:

```
function goToadobe(){
    dreamweaver.browseDocument('http://www.adobe.com/');
}
```

In Dreamweaver 4 kann dieser Vorgang mit dem folgenden Code erweitert werden, um das Dokument in Microsoft Internet Explorer zu öffnen.

```
function goToadobe(){
    var prevBrowsers = dw.getBrowserList();
    var theBrowser = "";
    for (var i=1; i < prevBrowsers.length; i+2){
        if (prevBrowsers[i].indexOf('Iexplore.exe') != -1){
            theBrowser = prevBrowsers[i];
            break;
        }
    }
    dw.browseDocument('http://www.adobe.com/',theBrowser);
}
```

Weitere Informationen zur Funktion `dreamweaver.getBrowserList()` finden Sie unter „dreamweaver.getBrowserList()“ auf Seite 112.

dreamweaver.getBrowserList()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Ruft eine Liste aller Browser im Untermenü „Datei“ > „Vorschau in Browser“ ab.

Argumente
Keine.

Rückgabewerte
Ein Array mit jeweils zwei Strings für jeden Browser in der Liste. Der jeweils erste String enthält den Namen des Browsers und der zweite String den Speicherort auf dem Computer des Benutzers im URL-Format „file://“. Wenn das Untermenü keine Browser enthält, wird kein Wert zurückgegeben.

dreamweaver.getExtensionEditorList()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Ruft für die angegebene Datei eine Liste der Editoren aus den Voreinstellungen für „Externe Editoren“ ab.

Argumente
fileURL

- Das Argument *fileURL* kann eine vollständige Angabe im URL-Format „file://“, ein Dateiname oder eine Dateierweiterung (einschließlich Punkt) sein.

Rückgabewerte
Ein Array mit jeweils zwei Strings für die einzelnen Editoren in der Liste. Der jeweils erste String enthält den Namen des Editors und der zweite String den Speicherort auf dem Computer des Benutzers im URL-Format „file://“. Wenn unter „Voreinstellungen“ kein Editor definiert ist, wird ein Array mit einem leeren String zurückgegeben.

Beispiel
Beim Aufruf der Funktion `dreamweaver.getExtensionEditorList(".gif")` wird beispielsweise ein Array mit den folgenden Strings zurückgegeben:

- `"Fireworks 3"`
- `"file:///C|/Programme/Adobe/Fireworks 3/Fireworks 3.exe"`

dreamweaver.getExternalTextEditor()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Ruft den Namen des derzeit konfigurierten externen Texteditors ab.

Argumente
Keine.

Rückgabewerte
Ein String mit dem Namen des Texteditors, der für die Anzeige in der Benutzeroberfläche besser geeignet ist als der vollständige Pfad.

dreamweaver.getFlashPath()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Ruft den vollständigen Pfad zur Anwendung Flash MX im Format einer Datei-URL ab.

Argumente
Keine.

Rückgabewerte
Ein Array mit zwei Elementen. Element [0] ist ein String mit dem Namen des Flash MX-Editors. Element [1] ist ein String, der im URL-Format „file://“ den Pfad zur Flash-Anwendung auf dem lokalen Computer enthält. Wenn Flash nicht installiert ist, wird kein Wert zurückgegeben.

Beispiel
Im folgenden Beispiel wird durch Aufrufen der Funktion `dw.getFlashPath()` der Pfad zur Flash-Anwendung abgerufen und dann im URL-Format „file://“ an die Funktion `dw.openWithApp()` übergeben, um das Dokument in Flash zu öffnen.

```
var myDoc = dreamweaver.getDocumentDOM();

if (dreamweaver.validateFlash()) {
    var flashArray = dreamweaver.getFlashPath();
    dreamweaver.openWithApp(myDoc.myForm.swfFilePath, flashArray[1]);
}
```

dreamweaver.getPrimaryBrowser()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Ruft den Pfad zum Primärbrowser ab.

Argumente
Keine.

Rückgabewerte
Ein String mit dem Pfad des Primärbrowsers auf dem Computer des Benutzers im URL-Format „file://“. Wenn kein Primärbrowser definiert wurde, wird kein Wert zurückgegeben.

dreamweaver.getPrimaryExtensionEditor()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Ruft den primären Editor für die angegebene Datei ab.

Argumente
fileURL

- Das Argument *fileURL* ist der Pfad der zu öffnenden Datei im URL-Format „file://“.

Rückgabewerte
Ein Array mit zwei Strings. Der erste String enthält den Namen des Editors und der zweite String den Speicherort auf dem Computer des Benutzers im URL-Format „file://“. Wenn kein primärer Editor definiert wurde, wird ein Array mit einem leeren String zurückgegeben.

dreamweaver.getSecondaryBrowser()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Ruft den Pfad zum Sekundärbrowser ab.

Argumente
Keine.

Rückgabewerte
Ein String mit dem Pfad des Sekundärbrowsers auf dem Computer des Benutzers im URL-Format „file://“. Wenn kein Sekundärbrowser definiert wurde, wird kein Wert zurückgegeben.

dreamweaver.openHelpURL()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Öffnet die angegebene Hilfedatei in der Hilfeansicht des Betriebssystems.

Die Hilfe wird in Dreamweaver in der Hilfeansicht des Betriebssystems und nicht in einem Browser angezeigt. Die Hilfeseiten liegen im HTML-Format vor, sind jedoch für die Windows-HTML-Hilfe oder den Help Viewer von Mac OS 9 und OS X ausgelegt.

Die Hilfe setzt sich aus den folgenden vier Dateitypen zusammen. Weitere Informationen zu Hilfedateien finden Sie in der Dokumentation Ihres Betriebssystems.

- Hilfebuch

  Das Hilfebuch besteht aus HTML-Hilfedateien, Grafiken und Indizes. Unter Windows wird das Hilfebuch als Datei mit der Dateinamenerweiterung „.chm“ bereitgestellt. Auf einem Macintosh-Computer ist das Hilfebuch ein Ordner.

  Die Dateien befinden sich im Dreamweaver-Ordner „Help“.

- Datei „help.xml“

  Die Datei „help.xml“ ordnet den Abschnitten im Hilfebuch Buch-IDs zu. Im folgenden XML-Codebeispiel wird die Buch-ID für die Dreamweaver-Hilfe den Dateinamen zugeordnet, die Hilfedateien für Windows und für Macintosh enthalten.

  ```
  <?xml version = "1.0" ?> <help-books><book-id id="DW_Using" win-
  mapping="UsingDreamweaver.chm" mac-mapping="Dreamweaver Help"/> </help-books>
  ```

  Jeder `book-id`-Eintrag hat folgende Attribute:

  - Das Attribut `id` ist die Buch-ID, die in den Dateien „help.map“ und „HelpDoc.js“ verwendet wird.
  - Das Attribut `win-mapping` ist der Name des Windows-Hilfebuchs, in diesem Beispiel `„UsingDreamweaver.chm"`.
  - Das Attribut `mac-mapping` ist der Name des Macintosh-Hilfebuchs, in diesem Beispiel `„Dreamweaver Help"`.

- Datei „help.map“

  Die Datei „help.map“ ordnet den jeweiligen Abschnitten in der Hilfe Inhalts-IDs zu. Dreamweaver sucht mithilfe der Datei „help.map“ bestimmte Hilfeeinträge, wenn die Hilfe intern aufgerufen wird.

- Datei „helpDoc.js“

Mit der Datei „helpDoc.js“ können Sie Variablennamen zuordnen, die Sie anstelle der Buch-ID und des Seitenstrings verwenden können. Die Datei „helpDoc.js“ ordnet einer HTML-Seite in einem bestimmten Hilfebuch eine Hilfeinhalts-ID zu. Dreamweaver ruft die Hilfe unter Verwendung der Datei „helpDoc.js“ über JavaScript auf.

Argumente
bookID

- Das obligatorische Argument `bookID` hat das Format `ID:page`.

  Der `ID`-Teil gibt die Buch-ID `bookID` in der Datei „help.xml“ an, mit der die Datei mit dem anzuzeigenden Hilfeinhalt benannt wird. Durch `page` wird die anzuzeigende Seite angegeben. Auf die Seiten wird in der Datei „help.map“ verwiesen.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der Vorgang erfolgreich ist, `false`, wenn die angegebene Datei in Dreamweaver in der Hilfeansicht nicht geöffnet werden kann.

Beispiel
```
openHelpURL("DW_Using:index.htm");
```

dreamweaver.openWithApp()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Öffnet die betreffende Datei in der angegebenen Anwendung.

Argumente
fileURL, appURL

- Das Argument *fileURL* ist der Pfad der zu öffnenden Datei im URL-Format „file://“.
- Das Argument *appURL* ist der Pfad der Anwendung, in der die Datei geöffnet werden soll, im URL-Format „file://“.

Rückgabewerte
Keine.

dreamweaver.openWithBrowseDialog()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Öffnet das Dialogfeld „Externen Editor auswählen“. In diesem Dialogfeld kann der Benutzer die Anwendung auswählen, in der die angegebene Datei geöffnet werden soll.

Argumente
fileURL

- Das Argument *fileURL* ist der Pfad der zu öffnenden Datei im URL-Format „file://".

Rückgabewerte
Keine.

dreamweaver.openWithExternalTextEditor()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Öffnet das aktuelle Dokument in dem externen Texteditor, der im Dialogfeld „Voreinstellungen" unter „Externe Editoren" definiert wurde.

Argumente
Keine.

Rückgabewerte
Keine.

dreamweaver.openWithImageEditor()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Öffnet die betreffende Datei im angegebenen Bildeditor.

Hinweis: *Wenn Fireworks als Bildeditor festgelegt ist, wird eine spezielle Funktion zur Integration von Adobe Fireworks gestartet, die Informationen an das aktive Dokument zurückgibt. Um Fehlermeldungen zu vermeiden, wenn kein Dokument aktiv ist, rufen Sie diese Funktion nicht über das Bedienfeld „Dateien" auf.*

Argumente
fileURL, appURL

- Das Argument *fileURL* ist der Pfad der zu öffnenden Datei im URL-Format „file://".
- Das Argument *appURL* ist der Pfad der Anwendung, in der die Datei geöffnet werden soll, im URL-Format „file://".

Rückgabewerte
Keine.

dreamweaver.validateFlash()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Ermittelt, ob Flash MX (oder eine spätere Version) auf dem lokalen Computer installiert ist.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn Flash MX (oder eine spätere Version) auf dem lokalen Computer installiert ist, andernfalls `false`.

dom.insertFiles()

Verfügbarkeit
Dreamweaver CS3.

Beschreibung
Fügt eine oder mehrere Dateien an der aktuellen Einfügemarke oder anstelle der aktuellen Markierung in das aktuelle Dokument ein und fordert den Benutzer gegebenenfalls zur Eingabe von Parametern auf.

Argumente
strFiles

- Das Argument *`strFiles`* ist ein String, der die Pfade und Namen der einzufügenden Dateien angibt. An diese Funktion können mehrere Dateinamen übergeben werden.

Rückgabewerte
Keine.

dreamweaver.activateApp()

Verfügbarkeit
Dreamweaver CS3.

Beschreibung
Bringt die angegebene Anwendung in den Vordergrund, wodurch sie zur aktiven Anwendung wird.

Argumente
applicationID

- Das Argument *applicationID* ist ein String, der die zu aktivierende Anwendung angibt, z. B. `dreamweaver`.

Rückgabewerte
Keine.

dreamweaver.printDocument()

Verfügbarkeit
Dreamweaver CS3.

Beschreibung
Führt die Entsprechung des Dreamweaver-Befehls „Datei“ > „Code drucken“ für die angeforderte Datei aus.

Argumente
fileName

- Das Argument `fileName` ist ein String, der den Namen der zu druckenden Datei als URL angibt.

Rückgabewerte
Keine.

dreamweaver.revealDocument()

Verfügbarkeit
Dreamweaver CS3.

Beschreibung
Setzt den Betriebssystemfokus auf Dreamweaver und bringt die angegebene Datei in den Vordergrund, wenn diese in Dreamweaver geöffnet ist.

Argumente
`fileName`

- Das Argument *fileName* ist ein String, der den Namen der anzuzeigenden Datei als URL angibt.

Rückgabewerte
Keine.

dreamweaver.launchApp()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Startet die angegebene Anwendung mit optionalen Befehlszeilenargumenten.

Argumente

`fileURL`

- Das Argument *fileURL* ist der im URL-Format `file://` angegebene Pfad für die Anwendung.

`optionalArgs`

- Das Argument *optionalArgs* ist ein String, mit dem Befehlszeilenargumente an die angegebene Anwendung übergeben werden können.

Rückgabewerte

Keine.

Beispiel

```
 // Launches the notepad application to edit filefoo.txt file.
dreamweaver.launchApp("file:///c:/windows/system32/notepad.exe",  "c:\temp\foo.txt");
// Launches myapp with some command line arguments.
dreamweaver.launchApp("file:///c:/bin/myapp.exe",  "-chrome false –print c:\temp\foo.txt");
```

Globale Anwendungsfunktionen

Globale Anwendungsfunktionen wirken sich auf die gesamte Anwendung aus. Mit diesen Funktionen kann unter anderem das Dialogfeld „Voreinstellungen“ geöffnet und geschlossen werden.

dreamweaver.beep()

Verfügbarkeit

Dreamweaver MX.

Beschreibung

Erstellt einen Systemwarnton.

Argumente

Keine.

Rückgabewerte

Keine.

Beispiel

Im folgenden Beispiel wird die Aufmerksamkeit des Benutzers mithilfe von `dw.beep()` auf eine von der Funktion `alert()` angezeigte Meldung gelenkt.

```
beep(){
    if(confirm("Is your order complete?")
    {
        dreamweaver.beep();
        alert("Click OK to submit your order");
    }
}
```

dreamweaver.getShowDialogsOnInsert()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob in der Kategorie „Allgemein“ des Dialogfelds „Voreinstellungen“ die Option „Beim Einfügen von Objekten Dialogfeld anzeigen“ aktiviert ist.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert, der angibt, ob die Option aktiviert ist.

dreamweaver.quitApplication()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Mit dieser Funktion wird Dreamweaver beendet, nachdem das aufrufende Skript ausgeführt wurde.

Argumente
Keine.

Rückgabewerte
Keine.

dreamweaver.showAboutBox()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Öffnet das Dialogfeld „Über Dreamweaver“.

Argumente
Keine.

Rückgabewerte
Keine.

dreamweaver.showDynamicDataDialog()

Verfügbarkeit
Dreamweaver UltraDev 1.

Beschreibung
Zeigt das Dialogfeld „Dynamische Daten“ oder „Dynamischer Text“ an und wartet darauf, dass der Benutzer das Dialogfeld schließt. Wenn der Benutzer auf „OK“ klickt, gibt die Funktion `showDynamicDataDialog()` einen String zurück, der in das Dokument des Benutzers eingefügt wird. (Dieser String wurde von der API-Datenquellenfunktion `generateDynamicDataRef()` zurückgegeben und an die API-Datenformatfunktion `formatDynamicDataRef()` übergeben. Der Rückgabewert von `formatDynamicDataRef()` entspricht dem von der Funktion `showDynamicDataDialog()` zurückgegebenen Wert.)

Argumente
source, {title}

- Das Argument *source* ist ein String mit Quellcode, der das dynamische Datenobjekt angibt. Es ist der gleiche String, der bei einem früheren Aufruf dieser Funktion zurückgegeben wurde. Die Funktion verwendet den Inhalt des *source*-Arguments, um alle Steuerelemente in Dialogfeldern zu initialisieren, damit sie genauso angezeigt werden wie zu dem Zeitpunkt, als der Benutzer zum Erstellen dieses Strings auf „OK“ geklickt hat.

  Dreamweaver übergibt diesen String an die Funktion `inspectDynamicDataRef()`, um zu ermitteln, ob der String einem Knoten in der Struktur entspricht. Wenn der String einem Knoten entspricht, wird dieser ausgewählt, wenn das Dialogfeld angezeigt wird. Sie können auch einen leeren String übergeben. Dadurch wird das Dialogfeld nicht initialisiert. So wird ein Dialogfeld bei der Erstellung eines neuen Elements beispielsweise nicht initialisiert.
- Das optionale Argument „title“ ist ein String mit dem Text, der in der Titelleiste des Dialogfelds angezeigt werden soll. Wenn dieses Argument nicht angegeben ist, wird in der Titelleiste „Dynamische Daten“ angezeigt.

Rückgabewerte
Ein String, der das dynamische Datenobjekt angibt, wenn der Benutzer auf „OK“ klickt.

dreamweaver.showPasteSpecialDialog()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion zeigt das Dialogfeld „Inhalte einfügen“ an. Wenn der Benutzer auf „OK“ klickt, führt die Funktion `showPasteSpecialDialog()` den Einfügevorgang aus.

Argumente
Keine.

Rückgabewerte
Keine.

Beispiel
```
dw.showPasteSpecialDialog();
```

dreamweaver.showPreferencesDialog()

Verfügbarkeit
Dreamweaver 3. In Dreamweaver 8 wurde das Argument *strCategory* hinzugefügt. Aktualisiert in CS4.

Beschreibung
Diese Funktion öffnet das Dialogfeld „Voreinstellungen“.

Argumente
{strCategory}

- Damit die entsprechende Kategorie im Dialogfeld „Voreinstellungen“ geöffnet wird, muss für das Argument *strCategory* einer der folgenden Strings angegeben werden: `general`, `accessibility`, `"html colors"` (für die Kategorie „Farbe für Code“), `"html format"` (für die Kategorie „Codeformat“), `"code hints"`, `"html rewriting"` (für die Kategorie „Codeumschreibung“), `copyPaste`, `"css styles"`, `"file compare"`, `"external editors"` (für die Kategorie „Dateitypen/Editoren“), `fonts`, `highlighting`, `"invisible elements"`, `layers`, `"new document"`, `floaters` (für die Kategorie „Bedienfelder“), `browsers` (für die Kategorie „Vorschau in Browser“), `"site ftp"` (für die Kategorie „Site“), `"status bar"` und `validator`. Wenn das Argument nicht als gültiger Name für den Bereich erkannt wird, wird das Dialogfeld mit dem Bereich geöffnet, der zuletzt aktiv war. Dies trifft auch zu, wenn das Argument nicht angegeben wird.

Rückgabewerte
Keine.

Beispiel
Im folgenden Beispiel wird das Dialogfeld „Voreinstellungen“ geöffnet und die Kategorie „Farbe für Code“ ausgewählt.

```
dw.showPreferencesDialog("html colors");
```

dreamweaver.showTagChooser()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Schaltet die Sichtbarkeit des Dialogfelds „Tag-Auswahl“ um, in dem Benutzer in der Codeansicht Tags einfügen können. Die Funktion zeigt das Dialogfeld „Tag-Auswahl“ im Vordergrund vor allen anderen Dreamweaver-Fenstern an. Wenn das Dialogfeld nicht sichtbar ist, wird es von der Funktion geöffnet, in den Vordergrund gebracht und aktiviert. Wenn die Tag-Auswahl sichtbar ist, blendet die Funktion das Dialogfeld aus.

Argumente
Keine.

Rückgabewerte
Keine.

dw.registerIdleHandler()

Verfügbarkeit
Dreamweaver CS3.

Beschreibung
Diese Funktion registriert eine JavaScript-Funktion, die während der Leerlaufzeiten regelmäßig aufgerufen werden soll.

Argumente
id, *idleFunction*, *interval*

- Das Argument `id` ist ein eindeutiger String, mit dem der zu registrierende Leerlauf-Programmtask angegeben wird. Damit die Eindeutigkeit gewährleistet ist, stellen Sie der ID einen eindeutigen Bezeichner voran. Um beispielsweise einen Warnton festzulegen, der alle 5 Sekunden ausgegeben wird, sollten Sie nicht den Task `"beep"` aufrufen, da ein anderer Benutzer möglicherweise einen Task mit demselben Namen erstellt hat. Ein besserer Name ist z. B. `"acme_beep_task"`, womit sowohl Kontextinformationen angegeben werden als auch die Eindeutigkeit des Strings gewährleistet ist.
- Das Argument `idleFunction` ist die JavaScript-Funktion, die während der Leerlaufzeiten aufgerufen werden soll.
- Das Argument `interval` ist die Anzahl der Sekunden zwischen den Aufrufen von *idleFunction*, sofern Leerlaufzeiten vorliegen.

Rückgabewerte
Ein boolescher Wert, der angibt, ob der Leerlauf-Task erfolgreich registriert wurde.

Beispiel
Im folgenden Beispiel gibt das System alle 5 Sekunden ein akustisches Signal aus:

```
dw.registerIdleHandler("acme_beep_task", function() { dw.beep();}, 5);
```

dw.revokeIdleHandler()

Verfügbarkeit
Dreamweaver CS3.

Beschreibung
Diese Funktion entfernt einen Leerlauf-Task, der zuvor durch die Funktion `registerIdleHandler()` gestartet wurde. Auf diese Weise ist es möglich, zuvor registrierte Leerlauf-Tasks zu entfernen. Wenn ein Leerlauf-Task aktiv bleiben soll, bis die Anwendung beendet wird, muss diese Funktion nicht aufgerufen werden. In diesem Fall wird der Leerlauf-Task vor dem Beenden der Anwendung automatisch entfernt.

Argumente
id

- Das id-Argument ist ein eindeutiger String zum Angeben des registrierten Leerlauf-Tasks, der entfernt werden soll. Dies ist dieselbe ID, die anfänglich zum Registrieren des Tasks verwendet wurde.

Rückgabewerte
Ein boolescher Wert, der angibt, ob der Leerlauf-Task erfolgreich entfernt wurde.

Beispiel
Im folgenden Beispiel wird der Leerlauf-Task `"dw_beep_task"` aus der Warteschlange der Leerlauf-Tasks entfernt:

```
dw.revokeIdleHandler("acme_beep_task");
```

Bridge-Kommunikationsfunktionen

Die Bridge-Kommunikationsfunktionen ermöglichen den Datenaustausch zwischen Dreamweaver und der Bridge-Anwendung. Ein Ziel dieser Kommunikation besteht darin, dem Benutzer in Dreamweaver das Navigieren zu Dateien in Bridge zu erleichtern.

BridgeTalk.bringToFront()

Verfügbarkeit
Dreamweaver CS3.

Beschreibung
Bringt die angegebene Anwendung als Prozess in den Vordergrund, indem die Funktion `BridgeTalk::bringToFront()` aufgerufen wird.

Argumente
applicationID

- Das Argument *applicationID* ist ein String (z. B. `bridge` oder `dreamweaver`), der die zu aktivierende Anwendung angibt.

Rückgabewerte
Keine.

Beispiel
In diesem Beispiel wird in Dreamweaver die Funktion `browseInBridge()` implementiert. Zunächst wird eine BridgeTalk-Instanz erstellt, anschließend werden die beiden wichtigsten Eigenschaften festgelegt: `target` und `body`. `<target>` ist die Zielanwendung. In diesem Fall ist dies die Bridge-Anwendung. Der entsprechende Bezeichner ist `bridge`. `<body>` ist die zu sendende Nachricht. Normalerweise ist `<body>` ein Skript, das von der Zielanwendung interpretiert werden kann und nach dem Empfang ausgeführt wird. Die Funktion `send()` wird aufgerufen, um die Nachricht `<body>` an das Ziel `<target>` zu senden.

```
if (!JSBridge.isRunning('bridge'))
{
var bt = new BridgeTalk;
var scriptSavePath = browsePath.replace(/['"\\]/g, "\$&");
var script = "app.document.thumbnail = new Thumbnail(decodeURI('" + scriptSavePath + "'));";
// Send the script to bridge and give it 10 sec to launch before assuming an error.
bt.target = "bridge";
bt.body = script;
result = bt.send(10);
}
if (result)
BridgeTalk.bringToFront('bridge');
```

Bridgetalk.send()

Verfügbarkeit
Dreamweaver CS3.

Beschreibung
Ermöglicht den Datenaustausch mit Adobe Bridge.

Argumente
timeout

- Das Argument *timeout* ist ein optionales Attribut zum Festlegen des Zeitüberschreitungsintervalls in Sekunden.

Rückgabewerte
Ein boolescher Wert, der angibt, ob die Kommunikation mit der Bridge-Anwendung erfolgreich war (`true`) oder nicht (`false`).

Beispiel

```
result = bridgeTalk.send(10);
```

BridgeTalk.suppressStartupScreen()

Verfügbarkeit
Dreamweaver CS3.

Beschreibung
Sucht die Startoptionen für `-nostartupscreen`, um festzulegen, ob die modalen Fenster nach dem Start unterdrückt werden sollen.

Rückgabewerte
Ein boolescher Wert, der angibt, ob Startfenster unterdrückt werden sollen.

dw.browseInBridge()

Verfügbarkeit
Dreamweaver CS3.

Beschreibung
Ermöglicht die Suche von Dateien in Bridge über Dreamweaver. Mit der Funktion `dw.browseInBridge()` wird die Bridge-Anwendung gestartet. Wenn Bridge bereits ausgeführt wird, wechselt `dw.browseInBridge` zur Bridge-Anwendung.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert, der angibt, ob das Browsing-Skript erfolgreich an die Bridge-Anwendung gesendet wurde (`true`) oder nicht (`false`).

Kapitel 12: Arbeitsbereich

Mit den API-Funktionen für den Arbeitsbereich werden Elemente im Adobe® Dreamweaver®-Arbeitsbereich erstellt oder bearbeitet. Mit diesen Funktionen werden u. a. folgende Aufgaben ausgeführt:

- Erneutes Ausführen der im Bedienfeld „Verlauf“ angezeigten Schritte
- Ablegen eines Objekts auf der Einfügeleiste
- Navigieren mit Tastaturfunktionen
- Erneutes Laden von Menüs
- Bearbeiten eigenständiger oder integrierter Ergebnisfenster
- Festlegen von Optionen
- Positionieren von Symbolleisten
- Ermitteln oder Festlegen des Fokus

Verlaufsfunktionen

Verlaufsfunktionen bewirken das Rückgängigmachen, Wiederholen, Aufzeichnen und Wiedergeben von Schritten, die im Bedienfeld „Verlauf“ angezeigt werden. Als Schritt wird jede wiederholbare Veränderung am Dokument oder an einer Auswahl im Dokument bezeichnet. Die Methoden des Objekts `dreamweaver.historyPalette` gelten für die Auswahl im Bedienfeld „Verlauf“ und nicht für die im aktuellen Dokument ausgewählten Elemente.

dom.redo()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Stellt den zuletzt im Dokument rückgängig gemachten Schritt wieder her.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dom.canRedo()“ auf Seite 513.

dom.undo()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Macht den zuvor ausgeführten Schritt rückgängig.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dom.canUndo()“ auf Seite 516.

dreamweaver.getRedoText()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Ruft den Text ab, der mit der Bearbeitungsaktion verknüpft ist, die wiederhergestellt wird, wenn der Benutzer den Befehl „Bearbeiten“ > „Wiederherstellen“ auswählt oder Strg+Y (Windows) bzw. Befehl+Y (Macintosh) drückt.

Argumente
Keine.

Rückgabewerte
Ein String mit dem Text für den wiederherzustellenden Bearbeitungsvorgang.

Beispiel
Wenn durch die letzte Aktion des Benutzers der ausgewählte Text fett formatiert wurde, wird beim Aufrufen der Funktion `dreamweaver.getRedoText()` der Wert `"Repeat Apply Bold"` zurückgegeben.

dreamweaver.getUndoText()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Ruft den Text ab, der mit der Bearbeitungsaktion verknüpft ist, die rückgängig gemacht wird, wenn der Benutzer den Befehl „Bearbeiten“ >„Rückgängig“ auswählt oder Strg+Z (Windows) bzw. Befehl+Z (Macintosh) drückt.

Argumente
Keine.

Rückgabewerte
Ein String mit dem Text für den rückgängig zu machenden Bearbeitungsvorgang.

Beispiel
Wenn durch die letzte Aktion des Benutzers ein CSS-Stil (Cascading Stylesheet) auf ausgewählten Text angewendet wurde, wird beim Aufrufen der Funktion `dreamweaver.getUndoText()` der Wert `"Undo Apply <span>"` zurückgegeben.

dreamweaver.playRecordedCommand()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Gibt den aufgezeichneten Befehl im aktiven Dokument wieder.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dreamweaver.canPlayRecordedCommand()“ auf Seite 520.

dreamweaver.redo()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Stellt den im aktiven Dokumentfenster, Dialogfeld, schwebenden Bedienfeld oder Bedienfeld „Dateien“ zuletzt rückgängig gemachten Schritt wieder her.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dreamweaver.canRedo()“ auf Seite 521.

dreamweaver.startRecording()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Beginnt die Aufzeichnung von Schritten im aktiven Dokument; der zuvor aufgezeichnete Befehl wird sofort verworfen.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dreamweaver.isRecording()“ auf Seite 528 (muss den Wert `false` zurückgeben).

dreamweaver.stopRecording()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Stoppt die Aufzeichnung, ohne den Benutzer dazu aufzufordern.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dreamweaver.isRecording()“ auf Seite 528 (muss den Wert `true` zurückgeben).

dreamweaver.undo()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Macht den zuvor im aktiven Dokumentfenster, Dialogfeld, schwebenden Bedienfeld oder Bedienfeld „Dateien“ durchgeführten Schritt rückgängig.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dom.canUndo()“ auf Seite 516.

dreamweaver.historyPalette.clearSteps()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Entfernt alle Schritte aus dem Bedienfeld „Verlauf“ und deaktiviert die Menüeinträge „Rückgängig“ und „Wiederherstellen“.

Argumente
Keine.

Rückgabewerte
Keine.

dreamweaver.historyPalette.copySteps()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Kopiert die angegebenen Verlaufsschritte in die Zwischenablage. Der Benutzer wird vor eventuell unbeabsichtigten Folgen gewarnt, wenn die betreffenden Schritte eine nicht wiederholbare Aktion beinhalten.

Argumente
arrayOfIndices

- Das Argument *arrayOfIndices* ist ein Array von Positionsindizes im Bedienfeld „Verlauf“.

Rückgabewerte
Ein String mit JavaScript-Code, der den angegebenen Verlaufsschritten entspricht.

Beispiel
Im folgenden Beispiel werden die ersten vier Schritte im Bedienfeld „Verlauf“ kopiert:

```
dreamweaver.historyPalette.copySteps([0,1,2,3]);
```

dreamweaver.historyPalette.getSelectedSteps()

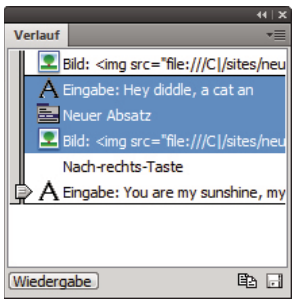
Verfügbarkeit
Dreamweaver 3.

Beschreibung
Bestimmt, welcher Teil des Bedienfelds „Verlauf“ ausgewählt ist.

Argumente
Keine.

Rückgabewerte
Ein Array mit den Positionsindizes aller ausgewählten Schritte. Die erste Position ist Position 0 (null).

Beispiel
Wenn im Bedienfeld „Verlauf“ der zweite, der dritte und der vierte Schritt ausgewählt sind, wie in der folgenden Abbildung gezeigt, wird beim Aufrufen der Funktion `dreamweaver.historyPalette.getSelectedSteps()` der Wert `[1,2,3]` zurückgegeben.



dreamweaver.historyPalette.getStepCount()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Ruft die Anzahl der Schritte im Bedienfeld „Verlauf“ ab.

Argumente
Keine.

Rückgabewerte
Eine Ganzzahl, die die Anzahl der Schritte angibt, die sich derzeit im Bedienfeld „Verlauf“ befinden.

dreamweaver.historyPalette.getStepsAsJavaScript()

Verfügbarkeit
Dreamweaver 3.

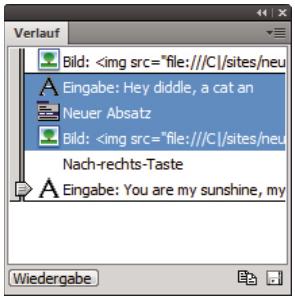
Beschreibung
Ruft den JavaScript-Code ab, der den ausgewählten Verlaufsschritten entspricht.

Argumente
arrayOfIndices

- Das Argument *arrayOfIndices* ist ein Array von Positionsindizes im Bedienfeld „Verlauf“.

Rückgabewerte
Ein String mit JavaScript-Code, der den angegebenen Verlaufsschritten entspricht.

Beispiel
Wenn im Bedienfeld „Verlauf“ die im folgenden Beispiel angezeigten drei Schritte ausgewählt sind, wird beim Aufrufen der Funktion
`dreamweaver.historyPalette.getStepsAsJavaScript(dw.historyPalette.getSelectedSteps())`
folgender Code zurückgegeben: `"dw.getDocumentDOM().insertText('Hey diddle diddle, a cat and a fiddle, the cow jumped over the moon.');\ ndw.getDocumentDOM().newBlock();\n dw.getDocumentDOM().insertHTML('<img src=\ "../wdw99/50browsers/images/sun.gif\">', true);\n":`



dreamweaver.historyPalette.getUndoState()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Ruft den aktuellen Rückgängig-Status ab.

Argumente
Keine.

Rückgabewerte
Die Position der Rückgängig-Markierung im Bedienfeld „Verlauf“.

dreamweaver.historyPalette.replaySteps()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Gibt die angegebenen Verlaufsschritte im aktiven Dokument wieder. Der Benutzer wird vor eventuell unbeabsichtigten Folgen gewarnt, wenn die betreffenden Schritte eine nicht wiederholbare Aktion beinhalten.

Argumente
arrayOfIndices

- Das Argument *arrayOfIndices* ist ein Array von Positionsindizes im Bedienfeld „Verlauf“.

Rückgabewerte
Ein String mit JavaScript-Code, der den angegebenen Verlaufsschritten entspricht.

Beispiel
Beim Aufruf der Funktion `dreamweaver.historyPalette.replaySteps([0,2,3])` werden der erste, der dritte und der vierte Schritt im Bedienfeld „Verlauf“ wiedergegeben.

dreamweaver.historyPalette.saveAsCommand()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Öffnet das Dialogfeld „Als Befehl speichern“, in dem der Benutzer die angegebenen Schritte als Befehl speichern kann. Der Benutzer wird vor eventuell unbeabsichtigten Folgen gewarnt, wenn die betreffenden Schritte eine nicht wiederholbare Aktion beinhalten.

Argumente
arrayOfIndices

- Das Argument *arrayOfIndices* ist ein Array von Positionsindizes im Bedienfeld „Verlauf“.

Rückgabewerte
Ein String mit JavaScript-Code, der den angegebenen Verlaufsschritten entspricht.

Beispiel
Im folgenden Beispiel werden der vierte, der sechste und der achte Schritt im Bedienfeld „Verlauf“ als Befehl gespeichert:

```
dreamweaver.historyPalette.saveAsCommand([3,5,7]);
```

dreamweaver.historyPalette.setSelectedSteps()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Wählt die angegebenen Schritte im Bedienfeld „Verlauf“ aus.

Argumente
arrayOfIndices

- Die Funktion *arrayOfIndices* ist ein Array von Positionsindizes im Bedienfeld „Verlauf“. Wenn kein Argument angegeben wird, werden alle Schritte deaktiviert.

Rückgabewerte
Keine.

Beispiel
Im folgenden Beispiel werden der erste, der zweite und der dritte Schritt im Bedienfeld „Verlauf“ ausgewählt:

```
dreamweaver.historyPalette.setSelectedSteps([0,1,2]);
```

dreamweaver.historyPalette.setUndoState()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Durchläuft die korrekte Anzahl von Rückgängig- bzw. Wiederherstellen-Schritten, um den angegebenen Rückgängig-Status zu erreichen.

Argumente
undoState

- Das Argument *undoState* ist das von der Funktion `dreamweaver.historyPalette.getUndoState()` zurückgegebene Objekt.

Rückgabewerte
Keine.

Funktionen zum Einfügen von Objekten

Funktionen zum Einfügen von Objekten beziehen sich auf Vorgänge für die Objekte in der Einfügeleiste bzw. im Menü „Einfügen“.

dreamweaver.objectPalette.getMenuDefault()

Verfügbarkeit
Dreamweaver MX 2004.

Beschreibung
Ruft den ID-String des Standardelements für das zugehörige Menü ab.

Argumente
menuId

- Das Argument *menuId* ist der String, durch den das Menü in der Datei „insertbar.xml“ definiert wird.

Rückgabewerte
Ein Stringwert, der die ID des Standardelements definiert.

Beispiel
Im folgenden Beispiel wird das aktuelle Standardobjekt für das Menü „Media“ der Variablen `defID` zugewiesen:

```
var defId = dw.objectPalette.getMenuDefault("DW_Media");
```

dreamweaver.objectPalette.setMenuDefault()

Verfügbarkeit
Dreamweaver MX 2004.

Beschreibung
Legt das Standardobjekt für ein Popupmenü fest. Das Symbol für das Standardobjekt bezieht sich auf das angegebene Popupmenü in der Einfügeleiste. Der Benutzer kann auf das Standardobjekt klicken, um dieses Objekt einzufügen, oder auf den Pfeil neben dem Standardobjekt, um das Popupmenü zu öffnen und die anderen Objekte in diesem Menü anzuzeigen. Dreamweaver legt die neue Standardeinstellung für das Menü fest, wenn der Benutzer das nächste Mal Dreamweaver öffnet oder den Befehl „Erweiterungen neu laden“ verwendet.

Argumente
menuId, defaultId

- Das Argument *menuId* ist der String, durch den das Menü in der Datei „insertbar.xml“ definiert wird.
- Das Argument *defaultId* ist der String, durch den das neue Standardobjekt im Feld „insertbar.xml“ definiert wird.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der neue Standard erfolgreich festgelegt wurde, andernfalls `false`.

Beispiel
Im folgenden Beispiel wird das Flash-Objekt als Standardobjekt für das Menü „Media“ festgelegt:

```
dw.objectPalette.setMenuDefault("DW_Media", "DW_Flash");
```

dreamweaver.reloadObjects()

Verfügbarkeit
Dreamweaver MX 2004.

Beschreibung
Lädt alle Objekte in der Einfügeleiste neu. Das gleiche Ergebnis erzielen Sie, wenn Sie in der Einfügeleiste bei gedrückter Strg-Taste mit der linken Maustaste auf das Kategorienmenü klicken und die Option „Erweiterungen neu laden“ auswählen.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn die Objekte erfolgreich geladen wurden, andernfalls `false`.

dom.convertActiveContent()

Verfügbarkeit
Dreamweaver CS3.

Beschreibung
Konvertiert den gesamten aktiven Inhalt im angegebenen Dokument.

Argumente
forceUpdate

- Das Argument *`forceUpdate`* ist ein boolescher Wert, der angibt, ob die Voreinstellungen des Benutzers überschrieben werden sollen (`true`) oder nicht. Dieses Argument ist optional.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der gesamte aktive Inhalt erfolgreich konvertiert wurde. Gibt `false` zurück, wenn ein Teil des aktiven Inhalts, der konvertiert werden sollte, nicht konvertiert wurde (z. B. Objekt-Tags in einem gesperrten Bereich einer Vorlageninstanz).

Beispiel

```
if( !dom.convertActiveContent(true) ) {
alert(dw.loadString("ActiveContent/notAllConverted"));
}
```

dom.convertNextActiveContent()

Verfügbarkeit
Dreamweaver CS3.

Beschreibung
Gibt an, dass für das nächste Objekt-Tag, das eingefügt wird (für den Rest der aktuellen Bearbeitungsaktion, die rückgängig gemacht werden kann), ein Skript erstellt wird. Diese Funktion ermöglicht es Ihnen, mit einer Erweiterung eines Drittanbieters ein geeignetes Skript für den jeweiligen aktiven Inhalt zu generieren.

Argumente
Keine.

Rückgabewerte
Keine.

Beispiel
```
dom.convertNextActiveContent();
dom.insertHTML("<object classid=\"clsid:D27CDB6E-AE6D-11cf-96B8-444553540000\" codebase=\
"http://download.Macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=7,0,19,0\"
width=\"100\" height=\"22\"><param name=\"movie\" value=\"button1.swf\" /><param name=\
"quality\" value=\"high\" /><embed src=\"button1.swf\" quality=\"high\" pluginspage=\
"http://www.Macromedia.com/go/getflashplayer\" type=\"application/
x-shockwave-flash\"width=\"100\" height=\"22\"></embed></object>\");
```

Tastaturfunktionen

Mit Tastaturfunktionen werden die Pfeiltasten und die Rücktaste sowie die Entf-, Bild-auf- und Bild-ab-Taste simuliert. Abgesehen von allgemeinen Pfeil- und Steuertastenfunktionen, wie `arrowLeft()` und `backspaceKey()`, stehen in Dreamweaver auch Funktionen zur Verfügung, mit denen zum nächsten oder zum vorherigen Wort bzw. Absatz gesprungen werden kann. Außerdem kann zum Anfang oder zum Ende der Zeile bzw. des Dokuments gesprungen werden.

dom.arrowDown()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Bewegt die Einfügemarke um die angegebene Anzahl von Schritten nach unten.

Argumente
{nTimes}, {bShiftIsDown}

- Das Argument *nTimes* gibt die Anzahl der Schritte an, um die die Einfügemarke nach unten bewegt werden soll. Bei fehlendem Argument gilt der Standardwert `1`.
- Das Argument *bShiftIsDown* ist ein boolescher Wert, der angibt, ob die Auswahl erweitert werden soll. Bei fehlendem Argument gilt der Standardwert `false`.

Rückgabewerte
Keine.

dom.arrowLeft()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Bewegt die Einfügemarke um die angegebene Anzahl von Schritten nach links.

Argumente
{nTimes}, {bShiftIsDown}

- Das optionale Argument *nTimes* gibt die Anzahl der Schritte an, um die die Einfügemarke nach links bewegt werden soll. Bei fehlendem Argument gilt der Standardwert `1`.
- Das optionale Argument *bShiftIsDown* ist ein boolescher Wert, der angibt, ob die Auswahl erweitert werden soll. Bei fehlendem Argument gilt der Standardwert `false`.

Rückgabewerte
Keine.

dom.arrowRight()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Bewegt die Einfügemarke um die angegebene Anzahl von Schritten nach rechts.

Argumente
{nTimes}, {bShiftIsDown}

- Das optionale Argument *nTimes* gibt die Anzahl der Schritte an, um die die Einfügemarke nach rechts bewegt werden soll. Bei fehlendem Argument gilt der Standardwert `1`.
- Das optionale Argument *bShiftIsDown* ist ein boolescher Wert, der angibt, ob die Auswahl erweitert werden soll. Bei fehlendem Argument gilt der Standardwert `false`.

Rückgabewerte
Keine.

dom.arrowUp()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion bewegt die Einfügemarke um die angegebene Anzahl von Schritten nach oben.

Argumente
{nTimes}, {bShiftIsDown}

- Das optionale Argument *nTimes* gibt die Anzahl der Schritte an, um die die Einfügemarke nach oben bewegt werden soll. Bei fehlendem Argument gilt der Standardwert `1`.
- Das optionale Argument *bShiftIsDown* ist ein boolescher Wert, der angibt, ob die Auswahl erweitert werden soll. Bei fehlendem Argument gilt der Standardwert `false`.

Rückgabewerte
Keine.

dom.backspaceKey()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion entspricht dem Drücken der Rücktaste (so oft wie angegeben). Das Ergebnis hängt davon ab, ob eine Auswahl oder nur eine Einfügemarke vorhanden ist.

Argumente
{nTimes}

- Das optionale Argument *nTimes* gibt an, wie oft die Rücktaste gedrückt werden muss. Bei fehlendem Argument gilt der Standardwert `1`.

Rückgabewerte
Keine.

dom.deleteKey()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion entspricht dem Drücken der Entf-Taste (so oft wie angegeben). Das Ergebnis hängt davon ab, ob eine Auswahl oder nur eine Einfügemarke vorhanden ist.

Argumente
{nTimes}

- Das optionale Argument *nTimes* gibt an, wie oft die Entf-Taste gedrückt werden muss. Bei fehlendem Argument gilt der Standardwert `1`.

Rückgabewerte
Keine.

dom.endOfDocument()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Bewegt die Einfügemarke zum Ende des Dokuments (d. h. hinter den letzten sichtbaren Inhalt im Dokumentfenster bzw. hinter das abschließende `HTML`-Tag im Codeinspektor, je nachdem, welches Fenster sich im Fokus befindet).

Argumente
{bShiftIsDown}

- Das optionale Argument *bShiftIsDown* ist ein boolescher Wert, der angibt, ob die Auswahl erweitert werden soll. Bei fehlendem Argument gilt der Standardwert `false`.

Rückgabewerte
Keine.

dom.endOfLine()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Bewegt die Einfügemarke an das Ende der Zeile.

Argumente
{bShiftIsDown}

- Das optionale Argument *bShiftIsDown* ist ein boolescher Wert, der angibt, ob die Auswahl erweitert werden soll. Bei fehlendem Argument gilt der Standardwert `false`.

Rückgabewerte
Keine.

dom.nextParagraph()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Bewegt die Einfügemarke an den Anfang des nächsten Absatzes oder überspringt mehrere Absätze, falls *nTimes* größer als 1 ist.

Argumente
{nTimes}, {bShiftIsDown}

- Das optionale Argument *nTimes* gibt die Anzahl der Absätze an, um die die Einfügemarke weiter bewegt werden soll. Bei fehlendem Argument gilt der Standardwert `1`.
- Das Argument *bShiftIsDown* ist ein boolescher Wert, der angibt, ob die Auswahl erweitert werden soll. Bei fehlendem Argument gilt der Standardwert `false`.

Rückgabewerte
Keine.

dom.nextWord()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Bewegt die Einfügemarke an den Anfang des nächsten Worts oder überspringt mehrere Wörter, falls *nTimes* größer als 1 ist.

Argumente
{nTimes}, {bShiftIsDown}

- Das optionale Argument *nTimes* gibt die Anzahl der Wörter an, um die die Einfügemarke weiter bewegt werden soll. Bei fehlendem Argument gilt der Standardwert `1`.
- Das optionale Argument *bShiftIsDown* ist ein boolescher Wert, der angibt, ob die Auswahl erweitert werden soll. Bei fehlendem Argument gilt der Standardwert `false`.

Rückgabewerte
Keine.

dom.pageDown()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Bewegt die Einfügemarke um eine Seite nach unten (entspricht der Bild-ab-Taste).

Argumente
{nTimes}, {bShiftIsDown}

- Das optionale Argument *nTimes* gibt die Anzahl der Seiten an, um die die Einfügemarke nach unten bewegt werden soll. Bei fehlendem Argument gilt der Standardwert `1`.
- Das optionale Argument *bShiftIsDown* ist ein boolescher Wert, der angibt, ob die Auswahl erweitert werden soll. Bei fehlendem Argument gilt der Standardwert `false`.

Rückgabewerte
Keine.

dom.pageUp()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Bewegt die Einfügemarke um eine Seite nach oben (entspricht der Bild-auf-Taste).

Argumente
{nTimes}, {bShiftIsDown}

- Das optionale Argument *nTimes* gibt die Anzahl der Seiten an, um die die Einfügemarke nach oben bewegt werden soll. Bei fehlendem Argument gilt der Standardwert `1`.
- Das optionale Argument *bShiftIsDown* ist ein boolescher Wert, der angibt, ob die Auswahl erweitert werden soll. Bei fehlendem Argument gilt der Standardwert `false`.

Rückgabewerte
Keine.

dom.previousParagraph()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Bewegt die Einfügemarke an den Anfang des vorherigen Absatzes oder überspringt mehrere Absätze, falls *nTimes* größer als 1 ist.

Argumente
{nTimes}, {bShiftIsDown}

- Das optionale Argument *nTimes* gibt die Anzahl der Absätze an, um die die Einfügemarke zurück bewegt werden soll. Bei fehlendem Argument gilt der Standardwert `1`.
- Das optionale Argument *bShiftIsDown* ist ein boolescher Wert, der angibt, ob die Auswahl erweitert werden soll. Bei fehlendem Argument gilt der Standardwert `false`.

Rückgabewerte
Keine.

dom.previousWord()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Bewegt die Einfügemarke an den Anfang des vorherigen Worts oder überspringt mehrere Wörter, falls *nTimes* größer als 1 ist.

Argumente
{nTimes}, {bShiftIsDown}

- Das optionale Argument *nTimes* gibt die Anzahl der Wörter an, um die die Einfügemarke zurück bewegt werden soll. Bei fehlendem Argument gilt der Standardwert `1`.
- Das optionale Argument *bShiftIsDown* ist ein boolescher Wert, der angibt, ob die Auswahl erweitert werden soll. Bei fehlendem Argument gilt der Standardwert `false`.

Rückgabewerte
Keine.

dom.startOfDocument()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Bewegt die Einfügemarke an den Anfang des Dokuments (d. h. vor den ersten sichtbaren Inhalt im Dokumentfenster bzw. vor das öffnende HTML-Tag im Codeinspektor, je nachdem, welches Fenster sich im Fokus befindet).

Argumente
{bShiftIsDown}

- Das optionale Argument *bShiftIsDown* ist ein boolescher Wert, der angibt, ob die Auswahl erweitert werden soll. Bei fehlendem Argument gilt der Standardwert `false`.

Rückgabewerte
Keine.

dom.startOfLine()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Bewegt die Einfügemarke an den Anfang der Zeile.

Argumente
{bShiftIsDown}

- Das optionale Argument *bShiftIsDown* ist ein boolescher Wert, der angibt, ob die Auswahl erweitert werden soll. Bei fehlendem Argument gilt der Standardwert `false`.

Rückgabewerte
Keine.

dreamweaver.mapKeyCodeToChar()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Konvertiert einen Tastencode, der aus dem Feld `keyCode` des Ereignisobjekts abgerufen wurde, in ein Zeichen. Überprüfen Sie, ob es sich bei dem Tastencode um eine Sondertaste handelt, wie z. B. um POS 1, BILD-AUF usw. Bezieht sich der Tastencode nicht auf eine Sondertaste, kann diese Methode verwendet werden, um den Code in einen für die Anzeige geeigneten Zeichencode umzuwandeln.

Argumente
keyCode

- Das Argument *keyCode* ist der Tastencode, der in ein Zeichen umgewandelt werden soll.

Rückgabewerte
Gibt den Zeichencode zurück, wenn die Zuordnung erfolgreich war. Andernfalls wird 0 zurückgegeben.

Menüfunktionen

Mithilfe von Menüfunktionen lassen sich die Menüs von Dreamweaver optimieren und neu laden. Die Funktionen `dreamweaver.getMenuNeedsUpdating()` und `dreamweaver.notifyMenuUpdated()` sind speziell dafür konzipiert, unnötige Aktualisierungen der in Dreamweaver integrierten dynamischen Menüs zu vermeiden. Weitere Informationen siehe „dreamweaver.getMenuNeedsUpdating()“ auf Seite 146 und „dreamweaver.notifyMenuUpdated()“ auf Seite 147.

dreamweaver.getMenuNeedsUpdating()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob das betreffende Menü aktualisiert werden muss.

Argumente
menuId

- Das Argument *menuId* ist ein String mit dem Wert des Attributs `id` für das Menüelement, wie in der Datei „menus.xml“ angegeben.

Rückgabewerte
Ein boolescher Wert, der angibt, ob das Menü aktualisiert werden muss. Diese Funktion gibt nur dann den Wert `false` zurück, wenn `dreamweaver.notifyMenuUpdated()` mit diesem *menuId*-Argument aufgerufen und der Rückgabewert von *menuListFunction* nicht geändert wurde. Weitere Informationen siehe „dreamweaver.notifyMenuUpdated()“ auf Seite 147.

dreamweaver.notifyMenuUpdated()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Gibt eine Meldung an Dreamweaver aus, wenn das angegebene Menü aktualisiert werden muss.

Argumente
menuId, menuListFunction

- Das Argument *menuId* ist ein String mit dem Wert des Attributs `id` für das Menüelement, wie in der Datei „menus.xml“ angegeben.
- Das Argument *menuListFunction* muss einer der folgenden Strings sein: `"dw.cssStylePalette.getStyles()"`, `"dw.getDocumentDOM().getFrameNames()"`, `"dw.getDocumentDOM().getEditableRegionList"`, `"dw.getBrowserList()"`, `"dw.getRecentFileList()"`, `"dw.getTranslatorList()"`, `"dw.getFontList()"`, `"dw.getDocumentList()"`, `"dw.htmlStylePalette.getStyles()"` oder `"site.getSites()"`.

Rückgabewerte
Keine.

dreamweaver.reloadMenus()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Lädt die gesamte Menüstruktur aus der Datei „menus.xml“ im Ordner „Configuration/Menus“ neu.

Argumente
Keine.

Rückgabewerte
Keine.

Ergebnisfenster-Funktionen

Die Ergebnisfenster-Funktionen ermöglichen Ihnen die Interaktion mit den integrierten Bedienfeldern in der Bedienfeldgruppe „Ergebnisse“. Ferner können Sie ein separates Fenster erstellen, in dem Spalten mit formatierten Daten angezeigt werden.

Integrierte Bedienfeldgruppe „Ergebnisse“

Mit diesen Funktionen wird eine Ausgabe in der Bedienfeldgruppe „Ergebnisse“ erzeugt. In der Bedienfeldgruppe „Ergebnisse“ werden Registerkarten für Suchvorgänge, Quellcodeprüfungen, Site-Berichte, Browserkompatibilitätsprüfungen, Server-Debugging, FTP-Protokolle und Hyperlink-Überprüfungen angezeigt.

Spezielle untergeordnete Bedienfelder

Die folgenden untergeordneten Bedienfelder sind integrierte Ergebnisfenster, die grundsätzlich in der Dreamweaver-Benutzeroberfläche zur Verfügung stehen und auf die Sie direkt zugreifen können.

- `dreamweaver.resultsPalette.siteReports`
- `dreamweaver.resultsPalette.validator`
- `dreamweaver.resultsPalette.bcc`

Da es sich bei diesen Bedienfeldern um Ergebnisfenster handelt, können Sie die folgenden, für separate Ergebnisfenster definierten Methoden verwenden:

- `getItem()`
- `getItemCount()`
- `getSelectedItem()`
- `setSelectedItem()`

Weitere Informationen über die Verwendung der `resWin`-Methoden finden Sie unter „Separates Ergebnisfenster“ auf Seite 153.

Aktives untergeordnetes Bedienfeld

Die folgenden allgemeinen API-Funktionen gelten für das jeweils aktive untergeordnete Bedienfeld. Bei einigen untergeordneten Bedienfeldern werden manche dieser Funktionen möglicherweise ignoriert. Wenn das aktive untergeordnete Bedienfeld die Funktion nicht unterstützt, hat der Aufruf dieser Funktion keinerlei Auswirkungen.

dreamweaver.showResults()

Verfügbarkeit

Dreamweaver MX 2004.

Beschreibung

Öffnet das angegebene schwebende Ergebnisbedienfeld und wählt das Element aus.

Hinweis: *Diese Funktion wird nur für die Bedienfelder „Überprüfung“, „Browserkompatibilität“ und „Site-Berichte“ der Bedienfeldgruppe „Ergebnisse“ unterstützt.*

Argumente
floaterName, floaterIndex

- Das Argument *floaterName* ist ein String, der angibt, welches schwebende Ergebnisbedienfeld geöffnet werden soll. Gültige Werte sind `'validation'` oder `'reports'`.
- Das Argument *floaterIndex* ist eine Zahl oder ein String. Mit einer Zahl geben Sie den Index eines Elements an, das im Bedienfeld „Ergebnisse“ ausgewählt werden soll. Mit einem String geben Sie die URL eines Dokuments an. Wenn Sie eine URL angeben, wählt die Funktion das erste sichtbare Element für das Dokument aus.

Rückgabewerte
Keine.

Beispiel
Im folgenden Beispiel wird am Offset der aktuellen Auswahl im Dokument eine Fehlerprüfung vorgenommen. Werden Fehler gefunden, werden sie im angegebenen Fenster (`floaterName`) im Bedienfeld „Ergebnisse“ angezeigt. Andernfalls wird das Fenster „Browserkompatibilität“ im Bedienfeld „Ergebnisse“ geöffnet und das erste sichtbare Element für das aktuelle Dokument angezeigt.

```
var offset = dw.getDocumentDOM().source.getSelection()[0];
var errors = dw.getDocumentDOM().source.getValidationErrorsForOffset(offset);
if ( errors && errors.length > 0 )
    dw.showResults( errors[0].floaterName, errors[0].floaterIndex );
else
    dw.showResults('bcc', dw.getDocumentDOM().URL);
```

dreamweaver.resultsPalette.siteReports.addResultItem()

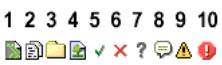
Verfügbarkeit
Dreamweaver 4.

Beschreibung
Fügt dem Bedienfeld „Site-Bericht“ einen neuen Ergebniseintrag hinzu. Dabei dienen die Daten in der Datei als Grundlage, die von der Funktion `processfile()` verarbeitet werden.

Diese Funktion steht nur über die Callback-Funktion `processFile()` eines Site-Berichts zur Verfügung. Ausführliche Informationen zu Site-Berichten finden Sie in *Dreamweaver erweitern* unter „Berichte“.

Argumente
strFilePath, strIcon, strDisplay, strDesc, {iLineNo}, {iStartSel}, {iEndSel}

- Das Argument *strFilePath* ist der vollständige URL-Pfadname der zu verarbeitenden Datei.
- Das Argument *strIcon* ist der Pfad des zu verwendenden Symbols. Um ein integriertes Symbol anzuzeigen, geben Sie anstelle des vollständigen Pfadnamens für das Symbol einen Wert von 1 bis 10 ein. (Geben Sie 0 ein, wenn kein Symbol angezeigt werden soll.) Im Folgenden sind die Symbole aufgeführt, die den Werten 1 bis 10 entsprechen:



- Das Argument *strDisplay* ist der String, der in der ersten Spalte des Ergebnisfensters angezeigt werden soll (normalerweise der Dateiname).

- Das Argument *strDesc* ist die Beschreibung für den Eintrag.
- Das Argument *iLineNo* entspricht der Anzahl der Zeilen in der Datei (optional).
- Das Argument *iStartSel* ist der Beginn des Offsets in der Datei. (Dieses Argument ist optional. Wenn es verwendet wird, muss auch das Argument *iEndSel* verwendet werden.)
- Das Argument *iEndSel* ist das Ende des Offsets in der Datei (erforderlich, wenn *iStartSel* verwendet wurde).

Rückgabewerte
Keine.

dreamweaver.resultsPalette.clear()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Löscht den Inhalt des aktiven Bedienfelds.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dreamweaver.resultsPalette.canClear()“ auf Seite 529.

dreamweaver.resultsPalette.Copy()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Sendet eine kopierte Meldung an das aktive Fenster (dies wird häufig für das Fenster für die FTP-Protokollierung verwendet).

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dreamweaver.resultsPalette.canCopy()“ auf Seite 529.

dreamweaver.resultsPalette.cut()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Sendet eine ausgeschnittene Meldung an das aktive Fenster (dies wird häufig für das Fenster für die FTP-Protokollierung verwendet).

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dreamweaver.resultsPalette.canCut()“ auf Seite 530.

dreamweaver.resultsPalette.Paste()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Sendet eine eingefügte Meldung an das aktive Fenster (dies wird häufig für das Fenster für die FTP-Protokollierung verwendet).

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dreamweaver.resultsPalette.canPaste()“ auf Seite 530.

dreamweaver.resultsPalette.openInBrowser

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Sendet einen Bericht an den Standardbrowser („Site-Berichte“, „Browserkompatibilität“, „Überprüfung“ und „Hyperlink-Prüfer“).

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dreamweaver.resultsPalette.canOpenInBrowser()“ auf Seite 530.

dreamweaver.resultsPalette.openInEditor()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Springt bei bestimmten Berichten („Site-Berichte“, „Browserkompatibilität“, „Überprüfung“ und „Hyperlink-Prüfer“) zur ausgewählten Zeile und öffnet das Dokument im Editor.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dreamweaver.resultsPalette.canOpenInEditor()“ auf Seite 531.

dreamweaver.resultsPalette.save()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Öffnet das Dialogfeld „Speichern“ für ein Fenster, das die Funktion „Speichern“ unterstützt („Site-Berichte“, „Browserkompatibilität“, „Überprüfung“ und „Hyperlink-Prüfer“).

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dreamweaver.resultsPalette.canSave()“ auf Seite 531.

dreamweaver.resultsPalette.selectAll()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Sendet den Befehl „Alles auswählen“ an das aktive Fenster.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dreamweaver.resultsPalette.canSelectAll()“ auf Seite 531.

Separates Ergebnisfenster

Die Funktion `dreamweaver.createResultsWindow` erstellt ein Ergebnisfenster.

dreamweaver.createResultsWindow()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Erstellt ein neues Ergebnisfenster und gibt eine JavaScript-Objektreferenz an dieses Fenster zurück.

Argumente
strName, arrColumns

- Das Argument *strName* ist der String für den Fenstertitel.
- Das Argument *arrColumns* ist ein Array von Spaltennamen, die für das Listensteuerelement verwendet werden.

Rückgabewerte
Eine Objektreferenz, die an das erstellte Fenster übergeben wird.

resWin.addItem()

Verfügbarkeit
Dreamweaver 4.

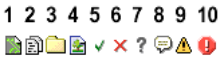
Beschreibung
Fügt dem Ergebnisfenster ein neues Element hinzu.

Hinweis: *Verwenden Sie diese Funktion nur bei eigenständigen Ergebnisfenstern, die mit „dreamweaver.createResultsWindow()“ auf Seite 153 erstellt wurden. Die Funktion* `resWin.addItem()` *kann nicht mit den integrierten Ergebnisfenstern („Überprüfung“, „Browserkompatibilität“ und „Site-Berichte“) verwendet werden.*

Argumente

resultWindowObj, strIcon, strDesc, itemData, iStartSel, iEndSel, colNdata

- Das Argument *resultWindowObj* ist das von der Funktion `createResultsWindow()` zurückgegebene Objekt.
- Das Argument *strIcon* ist ein String, der den Pfad zu dem gewünschten Symbol angibt. Um ein integriertes Symbol anzuzeigen, geben Sie anstelle des vollständigen Pfadnamens einen Wert von 1 bis 10 an. Geben Sie 0 (null) an, wenn kein Symbol angezeigt werden soll. Im Folgenden sind die Symbole aufgeführt, die den Werten 1 bis 10 entsprechen:



- Das Argument *strDesc* ist eine ausführliche Beschreibung des Elements. Geben Sie **0** an, wenn keine Beschreibung vorhanden ist.
- Das Argument *itemData* ist ein String, in dem Sie bestimmte Daten für das hinzuzufügende Element (z. B. die Zeilennummer im Dokument) speichern können.
- Das Argument *iStartSel* ist der Beginn des Auswahl-Offsets in der Datei. Geben Sie den Wert `null` an, wenn Sie keinen Offset angeben möchten.
- Das Argument *iEndSel* ist das Ende des Auswahl-Offsets in der Datei. Geben Sie den Wert `null` an, wenn Sie keinen Offset angeben möchten.
- Das Argument *colNdata* ist ein String-Array, das die Daten für die einzelnen Spalten angibt (d. h. für 3 Spalten ist ein Array von 3 Strings anzugeben).

Rückgabewerte

Ein boolescher Wert: `true`, wenn das Element erfolgreich hinzugefügt wurde, andernfalls `false`.

Beispiel

Im folgenden Beispiel wird ein Ergebnisfenster mit dem Namen `resWin` und den Spaltenüberschriften „Frodo“, „Sam“ und „Gollum“ erstellt. Durch den Aufruf der Funktion `resWin.addItem()` wird ein Ordnersymbol hinzugefügt. Dann werden die drei Strings `msg1`, `msg2` und `msg3` in die drei für das Fenster definierten Spalten eingefügt.

```
var resWin = dw.createResultsWindow("Test Window", ["Frodo", "Sam", "Gollum"]);
resWin.addItem(resWin, "3", "Description", null, null, null, ["msg1", "msg2", "msg3"]);
```

resWin.getItem()

Verfügbarkeit

Dreamweaver 4.

Beschreibung

Ruft ein Array mit Elementen ab, das die Namen der Befehle enthält, mit denen die jeweiligen Elemente hinzugefügt wurden, sowie die Strings, die an die Funktion `addItem()` übergeben wurden.

Argumente
itemIndex

- Das Argument *itemIndex* ist der Index des Elements, dessen Daten abgerufen werden sollen.

Rückgabewerte
Ein String-Array. Das erste Feld im Array ist der Name des Befehls, mit dem das Element hinzugefügt wurde. Die weiteren Felder enthalten die Strings, die an die Funktion `addItem()` übergeben wurden.

resWin.getItemCount()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Ruft die Anzahl der Elemente in der Liste ab.

Argumente
Keine.

Rückgabewerte
Die Anzahl der Elemente in der Liste.

resWin.getSelectedItem()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Ruft den Index des ausgewählten Elements ab.

Argumente
Keine.

Rückgabewerte
Der Index des aktuell ausgewählten Elements.

resWin.setButtons()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Definiert die im Argument *arrButtons* angegebenen Schaltflächen.

Argumente
cmdDoc, arrButtons

- Das Argument *cmdDoc* bezeichnet ein Dokumentobjekt, das den Befehl repräsentiert, der die Funktion aufruft. Befehle sollten das Schlüsselwort *this* verwenden.
- Das Argument *arrButtons* ist ein Array mit Strings, die den Schaltflächentext sowie den JavaScript-Code angeben, der ausgeführt werden soll, wenn der Benutzer auf die Schaltfläche klickt. Dies entspricht der Arbeitsweise der Funktion `commandButtons()` für Befehle. Im Fenster können nur zwei Schaltflächen definiert werden.

Rückgabewerte
Keine.

resWin.setCallbackCommands()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Teilt dem Ergebnisfenster mit, bei welchen Befehlen die Methode `processFile()` aufgerufen werden soll. Wenn diese Funktion nicht aufgerufen wird, wird stattdessen der Befehl aufgerufen, mit dem das Ergebnisfenster erstellt wurde.

Argumente
arrCmdNames

- Das Argument *arrCmdNames* ist ein Array von Befehlen, bei denen die Funktion `processFile()` aufgerufen werden soll.

Rückgabewerte
Keine.

resWin.setColumnWidths()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Stellt die Breite der einzelnen Spalten ein.

Argumente
arrWidth

- Das Argument *arrWidth* ist ein Array von Ganzzahlen, das die Breite der einzelnen Spalten im Steuerelement definiert.

Rückgabewerte
Keine.

resWin.setFileList()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Übergibt eine Liste mit Dateien und/oder Ordnern an das Ergebnisfenster, um einen Befehlssatz zur Verarbeitung aufzurufen.

Argumente
arrFilePaths, bRecursive

- Das Argument *arrFilePaths* ist ein Array von Pfaden zu Dateien oder Ordnern, das wiederholt durchlaufen werden soll.
- Das Argument *bRecursive* ist ein boolescher Wert, der angibt, ob die Wiederholung rekursiv (`true`) oder nicht rekursiv (`false`) erfolgen soll.

Rückgabewerte
Keine.

resWin.setSelectedItem()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Weist dem ausgewählten Element das mit *itemIndex* angegebene Element zu.

Argumente
itemIndex

- Der Index des in der Liste auszuwählenden Elements.

Rückgabewerte
Der Index des zuvor ausgewählten Elements.

resWin.setTitle()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Stellt den Titel des Fensters ein.

Argumente
strTitle

- Das Argument *strTitle* ist der neue Name des schwebenden Bedienfelds.

Rückgabewerte
Keine.

resWin.startProcessing()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Startet die Verarbeitung der Datei.

Argumente
Keine.

Rückgabewerte
Keine.

resWin.stopProcessing()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Stoppt die Verarbeitung der Datei.

Argumente
Keine.

Rückgabewerte
Keine.

Server-Debugging

Dreamweaver kann Dateien von Adobe ColdFusion anfordern und die Antwort im integrierten Browser anzeigen. Wenn die Antwort vom Server zurückgegeben wird, durchsucht Dreamweaver die Antwort nach einem XML-Paket mit einer bekannten Signatur. Wenn Dreamweaver XML-Code mit dieser Signatur findet, wird der XML-Code verarbeitet und der Inhalt in einer Strukturansicht angezeigt. Diese Strukturansicht enthält Informationen über die folgenden Elemente:

- Alle Vorlagen, benutzerdefinierten Tags und Include-Dateien, die für die Erstellung der angezeigten CFM-Seite verwendet werden
- Ausnahmen
- SQL-Abfragen
- Objektabfragen
- Variablen

- Verfolgungsprotokolle

Im Bedienfeld „Serverdebug“ können zudem Debugdaten von anderen Servermodellen angezeigt werden. Verwenden Sie die Funktion `dreamweaver.resultsPalette.debugWindow.addDebugContextData()`, um Dreamweaver für das Debuggen anderer Servermodelle einzurichten.

dreamweaver.resultsPalette.debugWindow.addDebugContextData()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Interpretiert eine benutzerdefinierte XML-Datei, die von dem im Dialogfeld „Site-Definition“ angegebenen Server zurückgegeben wird. Der Inhalt der XML-Datei wird in einer Strukturansicht im Bedienfeld „Serverdebug“ angezeigt. Sie können also im Bedienfeld „Serverdebug“ den vom Server erstellten Inhalt verschiedener Servermodelle auswerten.

Argumente
treedata

- Das Argument *treedata* enthält den vom Server zurückgegebenen XML-String. Für den XML-String sollte die folgende Formatierung verwendet werden:

`server debug node`	Stammknoten für die XML-Debugdaten
`debugnode`	Entspricht den einzelnen Knoten
`context`	Name des Elements, das in der Kontextliste angezeigt wird
`icon`	Das für den Strukturknoten zu verwendende Symbol
`name`	Der anzuzeigende Name
`value`	Der anzuzeigende Wert
`timestamp`	Gilt nur für den Kontextknoten

Die folgenden Strings sind optional:

`jumptoline`	Hyperlink zu einer bestimmten Zeilennummer
`template`	Name der Vorlagendatei in der URL
`path`	Pfad der Datei aus Sicht des Servers
`line number`	Zeilennummer innerhalb der Datei
`start position`	Startzeichen-Offset innerhalb der Zeile
`end position`	Endzeichen-Offset innerhalb der Zeile

Beispiel:

```
<serverdebuginfo>
    <context>
        <template><![CDATA[/ooo/master.cfm]]></template>
        <path><![CDATA[C:\server\wwwroot\ooo\master.cfm]]></path>
        <timestamp><![CDATA[0:0:0.0]]></timestamp>
    </context>
    <debugnode>
        <name><![CDATA[CGI]]></name>
        <icon><![CDATA[ServerDebugOutput/ColdFusion/CGIVariables.gif]]></icon>
        <debugnode>
            <name><![CDATA[Pubs.name.sourceURL]]></name>
            <icon><![CDATA[ServerDebugOutput/ColdFusion/Variable.gif]]></icon>
            <value><![CDATA[jdbc:Macromedia:sqlserver:
                //name.Macromedia.com:1111;databaseName=Pubs]]></value>
        </debugnode>
    </debugnode>
    <debugnode>
        <name><![CDATA[Element Snippet is undefined in class
coldfusion.compiler.TagInfoNotFoundException]]></name>
        <icon><![CDATA[ServerDebugOutput/ColdFusion/Exception.gif]]></icon>
        <jumptoline linenumber="3" startposition="2" endposition="20">
            <template><![CDATA[/ooo/master.cfm]]></template>
            <path><![CDATA[C:\Neo\wwwroot\ooo\master.cfm]]></path>
        </jumptoline>
    </debugnode>
</serverdebuginfo>
```

Rückgabewerte
Keine.

Umschaltfunktionen

Mit Umschaltfunktionen lassen sich verschiedene Optionen abrufen und festlegen, die entweder aktiviert oder deaktiviert sind.

dom.getEditNoFramesContent()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion ruft den aktuellen Status der Option „Modifizieren“ > „Frameset“ > „NoFrames-Inhalt bearbeiten“ ab.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der `NOFRAMES`-Inhalt die aktive Ansicht darstellt, andernfalls `false`.

dom.getHideAllVisualAids()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Diese Funktion bestimmt, ob die visuellen Hilfsmittel ausgeblendet sind.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn die Option „Alle visuellen Hilfsmittel ausblenden“ aktiviert ist, andernfalls `false`.

dom.getPreventLayerOverlaps()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion ruft den aktuellen Status der Option „Ebenenüberlappungen verhindern“ ab.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn die Option „Ebenenüberlappungen verhindern“ aktiviert ist, andernfalls `false`.

dom.getShowAutoIndent()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Diese Funktion bestimmt, ob der automatische Einzug in der Codeansicht des Dokumentfensters aktiviert ist.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der automatische Einzug aktiviert ist, andernfalls `false`.

dom.getShowFrameBorders()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion ruft den aktuellen Status der Option „Ansicht“ > „Visuelle Hilfsmittel“ > „Frame-Rahmen ab“.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn die Frame-Rahmen sichtbar sind, andernfalls `false`.

dom.getShowGrid()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion ruft den aktuellen Status der Option „Ansicht“ > „Raster“ > „Raster anzeigen“ ab.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn das Raster sichtbar ist, andernfalls `false`.

dom.getShowHeadView()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion ruft den aktuellen Status der Option „Ansicht“ > „Head-Inhalt“ ab.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der Head-Inhalt sichtbar ist, andernfalls `false`.

dom.getShowInvalidHTML()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Diese Funktion bestimmt, ob ungültiger HTML-Code derzeit in der Codeansicht des Dokumentfensters hervorgehoben wird.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn ungültiger HTML-Code hervorgehoben wird, andernfalls `false`.

dom.getShowImageMaps()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion ruft den aktuellen Status der Option „Ansicht“ > „Visuelle Hilfsmittel“ > „Imagemaps“ ab.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn die Imagemaps sichtbar sind, andernfalls `false`.

dom.getShowInvisibleElements()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion ruft den aktuellen Status der Option „Ansicht“ > „Visuelle Hilfsmittel“ > „Unsichtbare Elemente“ ab.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn Markierungen für unsichtbare Elemente angezeigt werden, andernfalls `false`.

dom.getShowLayerBorders()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion ruft den aktuellen Status der Option „Ansicht“ > „Visuelle Hilfsmittel“ > „Ebenenrahmen“ ab.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn die Ebenenrahmen sichtbar sind, andernfalls `false`.

dom.getShowLineNumbers()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Diese Funktion bestimmt, ob Zeilennummern in der Codeansicht angezeigt werden.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn die Zeilennummern angezeigt werden, andernfalls `false`.

dom.getShowRulers()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion ruft den aktuellen Status der Option „Ansicht“ > „Lineale“ > „Zeigen“ ab.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn die Lineale sichtbar sind, andernfalls `false`.

dom.getShowSyntaxColoring()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Diese Funktion bestimmt, ob die Syntaxfarbcodierung in der Codeansicht des Dokumentfensters aktiviert ist.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn die Syntaxfarbcodierung aktiviert ist, andernfalls `false`.

dom.getShowTableBorders()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion ruft den aktuellen Status der Option „Ansicht“ > „Visuelle Hilfsmittel“ > „Tabellenrahmen“ ab.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn die Tabellenrahmen sichtbar sind, andernfalls `false`.

dom.getShowToolbar()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Diese Funktion bestimmt, ob die Symbolleiste angezeigt wird.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn die Symbolleiste angezeigt wird, andernfalls `false`.

dom.getShowTracingImage()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion ruft den aktuellen Status der Option „Ansicht“ > „Tracing-Bild“> „Zeigen“ ab.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn die Option aktiviert ist, andernfalls `false`.

dom.getShowWordWrap()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Diese Funktion bestimmt, ob der Zeilenumbruch in der Codeansicht des Dokumentfensters aktiviert ist.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der Zeilenumbruch aktiviert ist, andernfalls `false`.

dom.getSnapToGrid()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion ruft den aktuellen Status der Option „Ansicht“ > „Raster“ > „Am Raster ausrichten“ ab.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn die Ausrichtung am Raster aktiviert ist, andernfalls `false`.

dom.setEditNoFramesContent()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion aktiviert bzw. deaktiviert die Option „Modifizieren“ > „Frameset“ > „NoFrames-Inhalt bearbeiten“.

Argumente
bEditNoFrames

- Das Argument *bEditNoFrames* ist ein boolescher Wert: `true` aktiviert die Option „NoFrames-Inhalt bearbeiten“, `false` deaktiviert sie.

Rückgabewerte
Keine.

Enabler
Siehe „dom.canEditNoFramesContent()“ auf Seite 510.

dom.setHideAllVisualAids()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Diese Funktion deaktiviert die Anzeige aller Rahmen, Imagemaps und unsichtbaren Elemente, unabhängig von den individuellen Einstellungen im Menü „Ansicht“.

Argumente
bSet

- Das Argument *bSet* ist ein boolescher Wert: `true` blendet visuelle Hilfsmittel aus, `false` blendet sie ein.

Rückgabewerte
Keine.

dom.setPreventLayerOverlaps()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion aktiviert bzw. deaktiviert die Option „Ebenenüberlappungen verhindern“.

Argumente
bPreventLayerOverlaps

- Das Argument *bPreventLayerOverlaps*ist ein boolescher Wert: `true` aktiviert die Option „Ebenenüberlappungen verhindern“, `false` deaktiviert sie.

Rückgabewerte
Keine.

dom.setShowFrameBorders()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion aktiviert bzw. deaktiviert die Option „Ansicht“ > „Visuelle Hilfsmittel“ > „Frame-Rahmen“.

Argumente
bShowFrameBorders

- Das Argument *bShowFrameBorders* ist ein boolescher Wert: `true` schaltet die Frame-Rahmen ein, `false` schaltet sie aus.

Rückgabewerte
Keine.

dom.setShowGrid()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion aktiviert bzw. deaktiviert die Option „Ansicht“ > „Raster“ > „Raster anzeigen“.

Argumente
bShowGrid

- Das Argument *bShowGrid* ist ein boolescher Wert: `true` aktiviert die Option „Raster anzeigen“, `false` deaktiviert sie.

Rückgabewerte
Keine.

dom.setShowHeadView()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion aktiviert bzw. deaktiviert die Option „Ansicht“ > „Head-Inhalt“.

Argumente
bShowHead

- Das Argument *bShowHead* ist ein boolescher Wert: `true` aktiviert die Option „Head-Inhalt“, `false` deaktiviert sie.

Rückgabewerte
Keine.

dom.setShowInvalidHTML()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Mit dieser Funktion wird die Hervorhebung des ungültigen HTML-Codes in der Codeansicht des Dokumentfensters aktiviert bzw. deaktiviert.

Diese Funktion bestimmt, ob ungültiger HTML-Code derzeit hervorgehoben wird.

Argumente
bShow

- Das Argument *bShow* ist ein boolescher Wert: `true` bedeutet, dass ungültiger HTML-Code hervorgehoben wird, und `false`, dass er nicht hervorgehoben wird.

Rückgabewerte
Keine.

dom.setShowImageMaps()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion aktiviert bzw. deaktiviert die Option „Ansicht“ > „Visuelle Hilfsmittel“ > „Imagemaps“.

Argumente
bShowImageMaps

- Das Argument *bShowImageMaps* ist ein boolescher Wert: `true` aktiviert die Option „Imagemaps“, `false` deaktiviert sie.

Rückgabewerte
Keine.

dom.setShowInvisibleElements()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion aktiviert bzw. deaktiviert die Option „Ansicht“ > „Visuelle Hilfsmittel“ > „Unsichtbare Elemente“.

Argumente
bViewInvisibleElements

- Das Argument *bViewInvisibleElements* ist ein boolescher Wert: `true` aktiviert die Option „Unsichtbare Elemente“, `false` deaktiviert sie.

Rückgabewerte
Keine.

dom.setShowLayerBorders()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion aktiviert bzw. deaktiviert die Option „Ansicht“ > „Visuelle Hilfsmittel“ > „Ebenenrahmen“.

Argumente
bShowLayerBorders

- Das Argument *bShowLayerBorders* ist ein boolescher Wert: `true` aktiviert die Option „Ebenenrahmen“, `false` deaktiviert sie.

Rückgabewerte
Keine.

dom.setShowLineNumbers()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Diese Funktion blendet die Zeilennummern in der Codeansicht des Dokumentfensters ein oder aus.

Argumente
bShow

- Das Argument *bShow* ist ein boolescher Wert: `true` blendet die Zeilennummern ein, `false` blendet sie aus.

Rückgabewerte
Keine.

dom.setShowRulers()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion aktiviert bzw. deaktiviert die Option „Ansicht“ > „Lineale“ > „Zeigen“.

Argumente
bShowRulers

- Das Argument *bShowRulers* ist ein boolescher Wert: `true` aktiviert die Option „Zeigen“, `false` deaktiviert sie.

Rückgabewerte
Keine.

dom.setShowSyntaxColoring()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Mit dieser Funktion wird die Syntaxfarbcodierung in der Codeansicht des Dokumentfensters aktiviert bzw. deaktiviert.

Argumente
bShow

- Das Argument *bShow* ist ein boolescher Wert: `true` schaltet die Syntaxfarbcodierung ein, `false` schaltet sie aus.

Rückgabewerte
Keine.

dom.setShowTableBorders()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion aktiviert bzw. deaktiviert die Option „Ansicht“ > „Visuelle Hilfsmittel“ > „Tabellenrahmen“.

Argumente
bShowTableBorders

- Das Argument *bShowTableBorders* ist ein boolescher Wert: `true` aktiviert die Option „Tabellenrahmen“, `false` deaktiviert sie.

Rückgabewerte
Keine.

dom.setShowToolbar()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Diese Funktion blendet die Symbolleiste ein oder aus.

Argumente
bShow

- Das Argument *bShow* ist ein boolescher Wert: `true` blendet die Symbolleiste ein, `false` blendet sie aus.

Rückgabewerte
Keine.

dom.setShowTracingImage()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion aktiviert bzw. deaktiviert die Option „Ansicht“ > „Tracing-Bild“ > „Zeigen“.

Argumente
bShowTracingImage

- Das Argument *bShowTracingImage* ist ein boolescher Wert: `true` aktiviert die Option „Zeigen“, `false` deaktiviert sie.

Rückgabewerte
Keine.

dom.setShowWordWrap()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Diese Funktion aktiviert bzw. deaktiviert den Zeilenumbruch in der Codeansicht des Dokumentfensters.

Argumente
bShow

- Das Argument *bShow* ist ein boolescher Wert: `true` aktiviert den Zeilenumbruch, `false` deaktiviert ihn.

Rückgabewerte
Keine.

dom.setSnapToGrid()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion aktiviert bzw. deaktiviert die Option „Ansicht“ > „Raster“ > „Am Raster ausrichten“.

Argumente
bSnapToGrid

- Das Argument *bSnapToGrid* ist ein boolescher Wert: `true` aktiviert die Ausrichtung am Raster, `false` deaktiviert sie.

Rückgabewerte
Keine.

dreamweaver.getHideAllFloaters()

Verfügbarkeit
Dreamweaver 3, aktualisiert in CS4.

Beschreibung
Diese Funktion bestimmt, ob alle angedockten oder schwebenden Bedienfelder ausgeblendet sind. Der Sichtbarkeitsstatus der Einfügeleiste wird dabei nicht berücksichtigt. Die folgenden Komponenten werden nicht als ausgeblendet betrachtet:

- Geschlossene Bedienfelder
- Reduzierte Registerkartengruppen
- Reduzierte Bedienfelder

Hinweis: *Im Gegensatz zur Einfügeleiste wird das Bedienfeld „Einfügen“ im Ergebnis berücksichtigt.*

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn alle Bedienfelder ausgeblendet sind, andernfalls `false`.

dreamweaver.getShowStatusBar()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion ruft den aktuellen Status der Option „Ansicht“ > „Statusleiste“ ab.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn die Statusleiste sichtbar ist, andernfalls `false`.

dreamweaver.htmlInspector.getShowAutoIndent()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Diese Funktion bestimmt, ob im Codeinspektor die Option „Automatischer Einzug“ aktiviert ist.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der automatische Einzug aktiviert ist, andernfalls `false`.

dreamweaver.htmlInspector.getShowInvalidHTML()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Diese Funktion bestimmt, ob im Codeinspektor derzeit ungültiger HTML-Code hervorgehoben wird.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn ungültiger HTML-Code hervorgehoben wird, andernfalls `false`.

dreamweaver.htmlInspector.getShowLineNumbers()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Diese Funktion bestimmt, ob im Codeinspektor Zeilennummern angezeigt werden.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn Zeilennummern angezeigt werden, andernfalls `false`.

dreamweaver.htmlInspector.getShowSyntaxColoring()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Diese Funktion bestimmt, ob im Codeinspektor die Syntaxfarbcodierung aktiviert ist.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn die Syntaxfarbcodierung aktiviert ist, andernfalls `false`.

dreamweaver.htmlInspector.getShowWordWrap()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Diese Funktion bestimmt, ob im Codeinspektor der Zeilenumbruch aktiviert ist.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der Zeilenumbruch aktiviert ist, andernfalls `false`.

dreamweaver.htmlInspector.setShowAutoIndent()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Diese Funktion aktiviert bzw. deaktiviert im Codeinspektor die Option „Automatischer Einzug“.

Argumente
bShow

- Das Argument *bShow* ist ein boolescher Wert: `true` aktiviert den automatischen Einzug, `false` deaktiviert ihn.

Rückgabewerte
Keine.

dreamweaver.htmlInspector.setShowInvalidHTML()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Mit dieser Funktion wird die Hervorhebung des ungültigen HTML-Codes im Codeinspektor aktiviert bzw. deaktiviert.

Argumente
bShow

- Das Argument *bShow* ist ein boolescher Wert: `true` bedeutet, dass ungültiger HTML-Code hervorgehoben wird, und `false`, dass er nicht hervorgehoben wird.

Rückgabewerte
Keine.

dreamweaver.htmlInspector.setShowLineNumbers()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Diese Funktion blendet die Zeilennummern im Codeinspektor ein oder aus.

Argumente
bShow

- Das Argument *bShow* ist ein boolescher Wert: `true` blendet die Zeilennummern ein, `false` blendet sie aus.

Rückgabewerte
Keine.

dreamweaver.htmlInspector.setShowSyntaxColoring()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Mit dieser Funktion wird die Syntaxfarbcodierung im Codeinspektor aktiviert bzw. deaktiviert.

Argumente
bShow

- Das Argument *bShow* ist ein boolescher Wert: `true` schaltet die Syntaxfarbcodierung ein, `false` schaltet sie aus.

Rückgabewerte
Keine.

dreamweaver.htmlInspector.setShowWordWrap()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Diese Funktion aktiviert bzw. deaktiviert den Zeilenumbruch im Codeinspektor.

Argumente
bShow

- Das Argument *bShow* ist ein boolescher Wert: `true` aktiviert den Zeilenumbruch, `false` deaktiviert ihn.

Rückgabewerte
Keine.

dreamweaver.setHideAllFloaters()

Verfügbarkeit
Dreamweaver 3, aktualisiert in CS4.

Beschreibung
Diese Funktion blendet alle Bedienfelder ein oder aus. Die Einfügeleiste wird hierbei nicht berücksichtigt.

Argumente

bShowFloatingPalettes

- Das Argument *bShowFloatingPalettes* ist ein boolescher Wert: `true` blendet alle Bedienfelder aus, `false` blendet sie ein. Wenn eines der Bedienfelder sichtbar ist, werden mit `false` die übrigen Bedienfelder eingeblendet. Wenn alle Bedienfelder bereits sichtbar sind, bleibt die Übergabe von `false` wirkungslos.

Hinweis: *Mit diesem Befehl werden nur dann Bedienfelder ausgeblendet, wenn alle Bedienfelder sichtbar sind. Wenn eines der Bedienfelder sichtbar ist, bleibt die Übergabe von „true" daher wirkungslos.*

Rückgabewerte

Keine.

dreamweaver.setShowStatusBar()

Verfügbarkeit

Dreamweaver 3.

Beschreibung

Diese Funktion aktiviert bzw. deaktiviert die Option „Ansicht" > „Statusleiste".

Argumente

bShowStatusBar

- Das Argument *bShowStatusBar* ist ein boolescher Wert: `true` aktiviert die Statusleiste, `false` deaktiviert sie.

Rückgabewerte

Keine.

site.getShowToolTips()

Verfügbarkeit

Dreamweaver 3.

Beschreibung

Diese Funktion ruft den aktuellen Status der Option „QuickInfo" ab.

Argumente

Keine.

Rückgabewerte

Ein boolescher Wert: `true`, wenn QuickInfos im Bedienfeld „Dateien" angezeigt werden, andernfalls `false`.

site.setShowToolTips()

Verfügbarkeit

Dreamweaver 3.

Beschreibung
Diese Funktion aktiviert bzw. deaktiviert die Option „QuickInfo“.

Argumente
bShowToolTips

- Das Argument *bShowToolTips* ist ein boolescher Wert: `true` aktiviert die Option „QuickInfo“, `false` deaktiviert sie.

Rückgabewerte
Keine.

Symbolleistenfunktionen

Mit den folgenden JavaScript-Funktionen können Sie die Sichtbarkeit von Symbolleisten und deren Beschriftungen abrufen und festlegen, die Beschriftungen von Symbolleistenelementen im aktuellen Fenster abrufen, Symbolleisten positionieren und Symbolleisten-IDs abrufen. Weitere Informationen zum Erstellen und Bearbeiten von Symbolleisten finden Sie im Hilfemodul „Dreamweaver erweitern“ unter „Symbolleisten“.

dom.getShowToolbarIconLabels()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Diese Funktion bestimmt, ob Beschriftungen von Schaltflächen im aktuellen Dokumentfenster sichtbar sind. Dreamweaver zeigt grundsätzlich Beschriftungen von Steuerelementen an, bei denen es sich nicht um Schaltflächen handelt, sofern die Beschriftungen definiert sind.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn Beschriftungen von Schaltflächen im aktuellen Dokumentfenster angezeigt werden, andernfalls `false`.

Beispiel
Im folgenden Beispiel werden die Beschriftungen von Schaltflächen sichtbar gemacht:

```
var dom = dw.getDocumentDom();
if (dom.getShowToolbarIconLabels() == false)
{
    dom.setShowToolbarIconLabels(true);
}
```

dom.getToolbarIdArray()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Diese Funktion gibt ein Array der IDs aller Symbolleisten in der Anwendung zurück. Mit `dom.getToolbarIdArray()` können Sie alle Symbolleisten ausblenden. Anschließend können Sie sie neu anordnen und nur eine bestimmte Auswahl einblenden.

Argumente
Keine.

Rückgabewerte
Ein Array aller Symbolleisten-IDs.

Beispiel
Im folgenden Beispiel wird das Array der Symbolleisten-IDs in der Variablen `tb_ids` gespeichert:

```
var tb_ids = new Array();
tb_ids = dom.getToolbarIdArray();
```

dom.getToolbarItemValue()

Verfügbarkeit
Dreamweaver MX 2004.

Beschreibung
Ruft den Wert des angegebenen Symbolleistenelements ab.

Argumente
toolbarID, itemID

- Das Argument *toolbarID* ist ein String, der die ID der Symbolleiste mit dem Element angibt, für das Sie einen Wert abrufen möchten.
- Das Argument *itemID* ist ein String, der die ID des Elements angibt, für das Sie einen Wert abrufen möchten.

Rückgabewerte
Ein String, der den Wert des Symbolleistenelements darstellt.

Beispiel
Das folgende Beispiel für `receiveArguments()` befindet sich in einem Symbolleistenbefehl, der das Verhalten eines Textfelds für die Größe steuert. Der Wert des Größenfelds wird als Argument abgerufen. Anschließend wird der Wert des Einheitenfelds gelesen, um einen gültigen Wert für die Funktion `font-size` der CSS-Eigenschaft zu ermitteln:

```
receiveArguments(newSize){
var dom = dw.getDocumentDOM();
if (newSize != ""){
    dom.applyFontMarkupAsStyle('font-size', newSize +
    dom.getToolbarItemValue("DW_Toolbar_Text","DW_Text_Units"));
    }
else{
    dom.removeFontMarkupAsStyle('font-size');
    }
}
```

dom.getToolbarLabel()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Diese Funktion ruft die Beschriftung der angegebenen Symbolleiste ab. Sie können `dom.getToolbarLabel()` auf Menüs anwenden, die Symbolleisten ein- oder ausblenden.

Argumente
toolbar_id

- Das Argument *`toolbar_id`* ist die ID der Symbolleiste, d. h. der Wert des ID-Attributs für das Symbolleisten-Tag in der Datei „toolbars.xml“.

Rückgabewerte
Der Namenstring *label*, der dem `toolbar`-Tag als Attribut zugeordnet ist.

Beispiel
Im folgenden Beispiel wird die Beschriftung für `myEditbar` in der Variablen `label` gespeichert:

```
var label = dom.getToolbarLabel("myEditbar");
```

dom.getToolbarVisibility()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Diese Funktion gibt einen booleschen Wert zurück, der angibt, ob die durch *`toolbar_id`* angegebene Symbolleiste angezeigt wird.

Argumente
toolbar_id

- Das Argument *`toolbar_id`* ist der ID-String, der der Symbolleiste zugeordnet ist.

Rückgabewerte
Ein boolescher Wert: `true`, wenn die Symbolleiste angezeigt wird; `false`, wenn die Symbolleiste nicht angezeigt wird oder nicht vorhanden ist.

Beispiel
Im folgenden Beispiel wird geprüft, ob die Symbolleiste `myEditbar` im Dokumentfenster angezeigt wird. Anschließend wird der Wert in der Variablen `retval` gespeichert:

```
var retval = dom.getToolbarVisibility("myEditbar");
return retval;
```

dom.setToolbarItemAttribute()

Verfügbarkeit
Dreamweaver MX 2004.

Beschreibung
Ändert einen Attributwert für die drei Bildattribute oder das tooltip-Attribut eines Symbolleistenelements.

Argumente
toolbarID, toolbarItemId, attrName, attrValue

- Das Argument *toolbarID* ist ein String, der die ID der Symbolleiste angibt.
- Das Argument *toolbarItemId* ist ein String, der die ID des Symbolleistenelements angibt.
- Das Argument *attrName* ist ein String, der den Namen des festzulegenden Attributs angibt. Gültige Werte sind `'image'`, `'overImage'`, `'disabledImage'` und `'tooltip'`.
- Das Argument *attrValue* ist ein String, der den festzulegenden Wert angibt.

Rückgabewerte
Keine.

Beispiel
Im folgenden Beispiel wird `dom.setToolbarItemAttribute()` dreimal aufgerufen, um die Attribute `image`, `imageOver` und `tooltip` für das Symbolleistenelement `MyButton` auf der Symbolleiste mit der ID `DW_Toolbar_Main` festzulegen.

```
var dom = dw.getDocumentDOM();
dom.setToolbarItemAttribute('DW_Toolbar_Main', 'MyButton', 'image',
    'Toolbars/imgs/newimage.gif');
dom.setToolbarItemAttribute('DW_Toolbar_Main', 'MyButton', 'imageOver',
    'Toolbars/imgs/newimageOver.gif');
dom.setToolbarItemAttribute('DW_Toolbar_Main', 'MyButton', 'tooltip', 'One fine button');
```

dom.setShowToolbarIconLabels()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Diese Funktion weist Dreamweaver an, die Beschriftungen von Schaltflächen, für die es Beschriftungen gibt, anzuzeigen. Dreamweaver zeigt grundsätzlich Beschriftungen von Steuerelementen an, bei denen es sich nicht um Schaltflächen handelt, sofern die Beschriftungen definiert sind.

Argumente
bShow

- Das Argument *bShow* ist ein boolescher Wert: `true` zeigt die Beschriftungen von Schaltflächen an, `false` zeigt sie nicht an.

Rückgabewerte
Keine.

Beispiel
Im folgenden Beispiel wird Dreamweaver angewiesen, die Beschriftungen für die Schaltflächen auf den Symbolleisten anzuzeigen:

```
dom.setShowToolbarIconLabels(true);
```

dom.setToolbarPosition()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Diese Funktion verschiebt die angegebene Symbolleiste an die angegebene Position.

Hinweis: *Die aktuelle Position einer Symbolleiste kann nicht bestimmt werden.*

Argumente
toolbar_id, position, relative_to

- Das Argument *`toolbar_id`* ist die ID der Symbolleiste, d. h. der Wert des ID-Attributs für das Symbolleisten-Tag in der Datei „toolbars.xml“.
- Das Argument *position* gibt an, an welcher Stelle Dreamweaver die Symbolleiste im Verhältnis zu anderen Symbolleisten positionieren soll. Die möglichen Werte für *position* sind in der folgenden Liste beschrieben:
  - `top` ist die Standardposition. Die Symbolleiste wird oberhalb des Dokumentfensters angezeigt.
  - `below` bewirkt, dass die Symbolleiste am Anfang der Zeile angezeigt wird, die sich direkt unterhalb der in *`relative_to`* angegebenen Symbolleiste befindet. Wenn die in *`relative_to`* definierte Symbolleiste nicht gefunden wird, gibt Dreamweaver einen Fehler aus.
  - `floating` bewirkt, dass die Symbolleiste über dem Dokument schwebt. Dreamweaver platziert die Symbolleiste automatisch so, dass sie gegenüber anderen schwebenden Symbolleisten versetzt ist. Auf dem Macintosh wird `floating` ebenso wie `top` interpretiert.
  - *`relative_to="toolbar_id"`* ist erforderlich, wenn der Wert für *position* gleich `below` ist. Andernfalls wird der Parameter ignoriert. Gibt die ID der Symbolleiste an, unterhalb der diese Symbolleiste positioniert werden soll.

Rückgabewerte
Keine.

Beispiel
Im folgenden Beispiel wird die Symbolleiste „myEditbar“ unterhalb der Symbolleiste „myPicturebar“ positioniert:

```
dom.setToolbarPosition("myEditbar", "below", "myPicturebar");
```

dom.setToolbarVisibility()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Diese Funktion blendet die angegebene Symbolleiste ein oder aus.

Argumente
toolbar_id, bShow

- Das Argument *toolbar_id* ist die ID der Symbolleiste, d. h. der Wert des ID-Attributs für das Symbolleisten-Tag in der Datei „toolbars.xml“.
- Das Argument *bShow* ist ein boolescher Wert, der angibt, ob die Symbolleiste ein- oder ausgeblendet werden soll. Hat *bshow* den Wert `true`, blendet `dom.setToolbarVisibility()` die Symbolleiste ein. Hat *bShow* den Wert `false`, blendet `dom.setToolbarVisibility()` die Symbolleiste aus.

Rückgabewerte
Keine.

Beispiel
Im folgenden Beispiel wird geprüft, ob die Symbolleiste „myEditbar“ im Dokumentfenster angezeigt wird. Ist dies nicht der Fall, wird sie eingeblendet:

```
var dom = dw.getDocumentDOM();
if(dom != null && dom.getToolbarVisibility("myEditbar") == false)
{
    dom.setToolbarVisibility("myEditbar", true);
{
```

dreamweaver.reloadToolbars()

Verfügbarkeit
Dreamweaver CS4.

Beschreibung
Mit dieser Funktion werden alle JavaScript-Symbolleisten im Ordner „Configuration/Toolbars“ neu geladen.

Argumente

{*resetToDefault*}

- Das Argument *resetToDefault* ist ein boolescher Wert, der angibt, ob die Standardsichtbarkeit und -position der einzelnen Symbolleisten aus der Datei „toolbars.xml“ gelesen wird. Wenn dieser Wert `false` ist oder nicht angegeben wird, bleiben Position und Sichtbarkeit der einzelnen Symbolleisten beim Neuladen erhalten. Dieses Argument ist optional.

Rückgabewerte

Keine.

Fensterfunktionen

Fensterfunktionen beeinflussen das Dokumentfenster und die schwebenden Bedienfelder. Diese Funktionen können schwebende Bedienfelder ein- und ausblenden, bestimmen, welcher Bereich des Dokumentfensters aktiviert ist, und das aktive Dokument festlegen. Informationen zu Funktionen, die speziell das Bedienfeld „Dateien“ betreffen, finden Sie unter „Site-Funktionen“ auf Seite 226.

Hinweis: *Einige der in diesem Abschnitt aufgeführten Funktionen können nur unter Windows verwendet werden. Ob dies der Fall ist, können Sie jeweils der Beschreibung einer Funktion entnehmen.*

dom.getFocus()

Verfügbarkeit

Dreamweaver 3.

Beschreibung

Diese Funktion bestimmt, auf welchem Teil der Anwendung sich der Fokus momentan befindet.

Argumente

Keine.

Rückgabewerte

Einer der folgenden Strings:

- Der String `"head"`, wenn der Bereich `HEAD` aktiv ist.
- Der String `"body"`, wenn der Bereich `BODY` oder `NOFRAMES` aktiv ist.
- Der String `"frameset"`, wenn ein Frameset oder einer seiner Frames ausgewählt ist.
- Der String `"none"`, wenn der Fokus sich nicht auf dem Dokument befindet (sondern beispielsweise auf dem Eigenschafteninspektor oder einem anderen schwebenden Bedienfeld).

dom.getView()

Verfügbarkeit

Dreamweaver 4, aktualisiert in CS4.

Beschreibung
Diese Funktion bestimmt, welche Ansicht sichtbar ist.

Argumente
Keine.

Rückgabewerte
`"design"`, `"code"`, `"split"` oder `"split code"`, je nachdem, welche Ansicht sichtbar ist.

dom.getWindowTitle()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion ruft den Titel des Fensters ab, das das Dokument enthält.

Argumente
Keine.

Rückgabewerte
Ein String mit dem Text, der zwischen den `TITLE`-Tags des Dokuments enthalten ist, bzw. kein Rückgabewert, wenn sich das Dokument nicht in einem geöffneten Fenster befindet.

dom.setView()

Verfügbarkeit
Dreamweaver 4, aktualisiert in CS4.

Beschreibung
Diese Funktion blendet die Entwurfsansicht bzw. die Codeansicht ein oder aus, damit nur die Entwurfsansicht, nur die Codeansicht oder eine geteilte Ansicht angezeigt wird.

Argumente
`viewString`

- Das Argument *`viewString`* ist die gewünschte Ansicht und muss einen der folgenden Werte haben: `"design"`, `"code"`, `"split"` oder `"split code"`.

Rückgabewerte
Keine.

dreamweaver.bringAttentionToFloater()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Bringt das angegebene Bedienfeld oder den angegebenen Inspektor in den Vordergrund und macht den Benutzer auf dieses Bedienfeld bzw. diesen Inspektor durch Blinken aufmerksam. Diese Funktion unterscheidet sich geringfügig von `dw.toggleFloater()`.

Argumente
floaterName

- Das Argument *floaterName* ist der Name des Fensters, Bedienfelds oder Inspektors.

Rückgabewerte
Keine.

Beispiel
Im folgenden Beispiel wird das Bedienfeld „Elemente“ geöffnet und durch Blinken hervorgehoben:

```
dw.bringAttentionToFloater("library");
```

dreamweaver.cascade()

Verfügbarkeit
Dreamweaver MX (nur Windows), Dreamweaver 8 (um Macintosh-Unterstützung erweitert).

Beschreibung
Die Dokumentfenster werden überlappend angeordnet. Dabei wird ausgehend von der linken oberen Ecke jedes weitere Fenster etwas nach unten und rechts versetzt neben dem vorherigen Fenster positioniert.

Argumente
Keine.

Rückgabewerte
Keine.

Beispiel
Im folgenden Beispiel werden die geöffneten Dokumente überlappend angeordnet:

```
dw.cascade()
```

dreamweaver.getActiveWindow()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion ruft das Dokument im aktiven Fenster ab.

Argumente
Keine.

Rückgabewerte
Das dem Dokument im aktiven Fenster entsprechende Dokumentobjekt bzw., falls sich das Dokument in einem Frame befindet, das dem Frameset entsprechende Dokumentobjekt.

dreamweaver.getDocumentList()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion ruft eine Liste aller geöffneten Dokumente ab.

Argumente
Keine.

Rückgabewerte
Ein Array von Dokumentobjekten, die jeweils einem geöffneten Dokumentfenster entsprechen. Wenn ein Dokumentfenster ein Frameset enthält, bezieht sich das Dokumentobjekt auf das Frameset und nicht auf den Inhalt der Frames.

dreamweaver.getFloatersVisible()

Verfügbarkeit

Beschreibung
Bestimmt, ob alle angedockten oder schwebenden Bedienfelder sichtbar sind. Folgendes wird ignoriert:

- Sichtbarkeitsstatus der Einfügeleiste
- Geschlossene Bedienfelder

Hinweis: *Im Gegensatz zur Einfügeleiste wird das Bedienfeld „Einfügen“ im Ergebnis berücksichtigt.*

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn Bedienfelder sichtbar sind, andernfalls `false`.

dreamweaver.getFloaterVisibility()

Verfügbarkeit
Dreamweaver 3, aktualisiert in CS4.

Beschreibung
Diese Funktion prüft, ob das angegebene Bedienfeld bzw. der angegebene Inspektor sichtbar ist.

Argumente

floaterName

- Das Argument *floaterName* ist der Name eines schwebenden Bedienfelds. Wenn *floaterName* mit keinem der integrierten Bedienfelder übereinstimmt, sucht Dreamweaver im Ordner „Configuration/Floaters“ nach einer Datei mit dem Namen „floaterName.htm“. Hierbei steht *floaterName* für den Namen eines schwebenden Bedienfelds.

  In der folgenden Liste sind die rechts neben den Bedienfeldnamen stehenden Strings die *floaterName*-Werte für die in Dreamweaver integrierten Bedienfelder:

Elemente = "assets"
Verhalten = "behaviors"
Bindungen = "data bindings"
Codeinspektor = "html"
Komponenten = "server components"
CSS-Stile = "css styles"
Datenbanken = "databases"
Frames = "frames"
Verlauf = "history"
Einfügen = "objects"
Ebenen = "layers"
Bibliothek = "library"
Hyperlink-Prüfer (Ergebnisse) = "linkchecker"
Eigenschaften = "properties"
Referenz = "reference"
Berichte (Ergebnisse) = "reports"
Suchen (Ergebnisse) = "search"
Auswahlinspektor = "selection inspector"
Serververhalten = "server behaviors"
Site = "site"
Site-Dateien = "site files"
Codefragmente = "snippets"
Browserkompatibilitätsprüfung = "bcc"
Überprüfung (Ergebnisse) = "validation"

Rückgabewerte

Ein boolescher Wert: `true`, wenn das schwebende Bedienfeld sichtbar ist und sich im Vordergrund befindet, `false`, falls dies nicht zutrifft oder kein schwebendes Bedienfeld mit dem Namen *floaterName* gefunden werden kann.

dreamweaver.getFocus()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Diese Funktion bestimmt, auf welchem Teil der Anwendung sich der Fokus momentan befindet.

Argumente
bAllowFloaters

- Das Argument `bAllowFloaters` ist ein boolescher Wert: `true`, wenn die Funktion den Namen des schwebenden Bedienfelds zurückgeben soll (sofern sich der Fokus auf einem schwebenden Bedienfeld befindet), andernfalls `false`.

Rückgabewerte
Einer der folgenden Strings:

- Der String `"document"`, wenn sich der Fokus auf dem Dokumentfenster befindet.
- Der String `"site"`, wenn sich der Fokus auf dem Bedienfeld „Dateien“ befindet.
- Der String `"textView"`, wenn sich der Fokus auf der Textansicht befindet.
- Der String `"html"`, wenn sich der Fokus auf dem Codeinspektor befindet.
- Der String `floaterName`, wenn `bAllowFloaters` den Wert `true` hat und sich der Fokus auf einem schwebenden Bedienfeld befindet. `floaterName` kann folgende Werte haben: `"objects"`, `"properties"`, `"launcher"`, `"library"`, `"css styles"`, `"html styles"`, `"behaviors"`, `"timelines"`, `"layers"`, `"frames"`, `"templates"` oder `"history"`.
- (Macintosh) Der String `"none"`, wenn weder das Bedienfeld „Dateien“ noch ein Dokumentfenster geöffnet ist.

dreamweaver.getPrimaryView()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Diese Funktion bestimmt, welche Ansicht als primäre Ansicht im Vordergrund angezeigt wird.

Argumente
Keine.

Rückgabewerte
Der String `"design"` oder `"code"`, je nachdem, welche Ansicht angezeigt wird oder sich in einer geteilten Ansicht oben befindet.

dreamweaver.getSnapDistance()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Diese Funktion gibt den Abstand für die Einrastfunktion in Pixel zurück.

Argumente
Keine.

Rückgabewerte
Eine Ganzzahl, die den Abstand für die Einrastfunktion in Pixel angibt. Der Standardwert lautet 10 Pixel. 0 bedeutet, dass die Einrastfunktion deaktiviert ist.

dreamweaver.minimizeRestoreAll()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Diese Funktion bewirkt, dass alle Fenster in Dreamweaver minimiert (auf Symbolgröße verkleinert) oder wiederhergestellt werden.

Argumente
bMinimize

- Das Argument *bMinimize* ist ein boolescher Wert: `true`, wenn Fenster minimiert werden sollen, `false`, wenn minimierte Fenster wiederhergestellt werden sollen.

Rückgabewerte
Keine.

dreamweaver.setActiveWindow()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion aktiviert das Fenster, das das angegebene Dokument enthält.

Argumente
documentObject, {bActivateFrame}

- Das Argument *documentObject* ist das Objekt am Stamm der DOM-Struktur eines Dokuments (der von `dreamweaver.getDocumentDOM()` zurückgegebene Wert).

- Das optionale Argument *bActivateFrame* ist nur gültig, wenn sich *documentObject* innerhalb eines Framesets befindet. Das Argument *bActivateFrame* ist ein boolescher Wert: `true`, um den Frame mit dem Dokument sowie das Fenster mit dem Frameset zu aktivieren, andernfalls `false`.

Rückgabewerte
Keine.

dreamweaver.setFloaterVisibility()

Verfügbarkeit
Dreamweaver 3, aktualisiert in CS4.

Beschreibung
Diese Funktion legt fest, ob ein bestimmtes schwebendes Bedienfeld oder ein Inspektor sichtbar gemacht werden soll.

Argumente
`floaterName`, *`bIsVisible`*

- Das Argument *`floaterName`* ist der Name eines schwebenden Bedienfelds. Wenn *`floaterName`* nicht einer der Namen für integrierte Bedienfelder ist, wird im Ordner „Configuration/Floaters“ nach einer Datei mit dem Namen *`floaterName.htm`* gesucht. Sollte kein schwebendes Bedienfeld mit dem Namen *`floaterName`* gefunden werden, ist diese Funktion wirkungslos.

  In der folgenden Liste sind die rechts neben den Bedienfeldnamen stehenden Strings die *`floaterName`*-Werte für die in Dreamweaver integrierten Bedienfelder:

Elemente = "`assets`"
Verhalten = "`behaviors`"
Bindungen = "`data sources`"
Codeinspektor = "`html`"
Komponenten = "`server components`"
CSS-Stile = "`css styles`"
Datenbanken = "`databases`"
Frames = "`frames`"
Verlauf = "`history`"
HTML-Stile = "`html styles`"
Einfügen = "`objects`"
Ebenen = "`layers`"
Bibliothek = "`library`"
Hyperlink-Prüfer (Ergebnisse) = "`linkchecker`"
Eigenschaften = "`properties`"
Referenz = "`reference`"
Berichte (Ergebnisse) = "`reports`"

Suchen (Ergebnisse) = "`search`"
Serververhalten = "`server behaviors`"
Site = "`site`"
Site-Dateien = "`site files`"
Codefragmente = "`snippets`"
Tag-Inspektor = "`tag inspector`"
Browserkompatibilitätsprüfung = "`bcc`"
Vorlagen = "`templates`"
Überprüfung (Ergebnisse) = "`validation`"

- Das Argument *`bIsVisible`* ist ein boolescher Wert, der angibt, ob das schwebende Bedienfeld sichtbar gemacht werden soll.

Rückgabewerte
Keine.

dreamweaver.setPrimaryView()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Diese Funktion zeigt die angegebene Ansicht oben im Dokumentfenster an.

Argumente
viewString

- Das Argument *viewString* ist die Ansicht, die oben im Dokumentfenster angezeigt werden soll. Es kann sich um einen der folgenden Werte handeln": `"design"` oder `"code"`

Rückgabewerte
Keine.

dreamweaver.setSnapDistance()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Diese Funktion legt den Abstand für die Einrastfunktion in Pixel fest.

Argumente
snapDistance

- Das Argument *snapDistance* ist eine Ganzzahl, die den Abstand für die Einrastfunktion in Pixel angibt. Der Standardwert ist 10 Pixel. Geben Sie 0 an, um die Einrastfunktion zu deaktivieren.

Rückgabewerte
Keine.

dreamweaver.showProperties()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion zeigt den Eigenschafteninspektor an und übergibt ihm den Fokus.

Argumente
Keine.

Rückgabewerte
Keine.

dreamweaver.tileHorizontally()

Verfügbarkeit
Dreamweaver MX (nur Windows), Dreamweaver 8 (um Macintosh-Unterstützung erweitert).

Beschreibung
Ordnet die Dokumentfenster untereinander an, wobei die einzelnen Fenster aneinander grenzen, ohne dass sich die Dokumente überlappen. Dieser Vorgang entspricht einer vertikalen Aufteilung des Arbeitsbereichs.

Argumente
Keine.

Rückgabewerte
Keine.

Beispiel
Im folgenden Beispiel werden die geöffneten Dokumente untereinander angeordnet:

```
dw.tileHorizontally()
```

dreamweaver.tileVertically()

Verfügbarkeit
Dreamweaver MX (nur Windows), Dreamweaver 8 (um Macintosh-Unterstützung erweitert).

Beschreibung
Ordnet die Dokumentfenster nebeneinander an, wobei die einzelnen Fenster aneinander grenzen, ohne dass sich die Dokumente überlappen. Dieser Vorgang entspricht einer horizontalen Aufteilung des Arbeitsbereichs.

Argumente
Keine.

Rückgabewerte
Keine.

Beispiel
Im folgenden Beispiel werden die geöffneten Dokumente nebeneinander angeordnet:

```
dw.tileVertically()
```

dreamweaver.toggleFloater()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion bewirkt, dass das angegebene schwebende Bedienfeld bzw. der Inspektor eingeblendet, ausgeblendet oder in den Vordergrund gestellt wird.

Hinweis: *Diese Funktion hat nur in der Datei „menus.xml" Bedeutung. Um schwebende Bedienfelder anzuzeigen, in den Vordergrund zu stellen oder auszublenden, verwenden Sie* `dw.setFloaterVisibility()`.

Argumente
floaterName

- Das Argument *floaterName* ist der Name des Fensters. Wenn der Name des schwebenden Bedienfelds `reference` lautet, wird der Status „sichtbar/unsichtbar" des Bedienfelds „Referenz" durch die Auswahl aktualisiert, die der Benutzer in der Codeansicht vornimmt. Für alle anderen Bedienfelder wird die Auswahl kontinuierlich protokolliert. Beim Bedienfeld „Referenz" wird die Auswahl in der Codeansicht jedoch nur protokolliert, wenn der Benutzer diese Funktion explizit aufruft.

Rückgabewerte
Keine.

dreamweaver.updateReference()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Diese Funktion aktualisiert das schwebende Bedienfeld „Referenz“. Wenn das schwebende Bedienfeld „Referenz“ nicht sichtbar ist, wird es durch `dw.updateReference()` angezeigt und aktualisiert.

Argumente
Keine.

Rückgabewerte
Keine.

Funktionen für die Informationsleiste

Die Informationsleiste dient zum Anzeigen von Fehlermeldungen, ohne den Arbeitsablauf zu unterbrechen. Die folgenden Funktionen für die Informationsleiste werden verwendet, um die Informationsleiste mit den Fehlermeldungen anzuzeigen oder auszublenden.

dom.showInfoBar()

Verfügbarkeit
Dreamweaver CS4.

Beschreibung
Diese Funktion zeigt die Informationsleiste sowie die als Parameter übergebene Meldung an. Wenn die Informationsleiste bereits angezeigt wird, wird die bisherige Meldung durch die übergebene ersetzt. Wenn keine Meldung übergeben wird, tritt ein JavaScript-Fehler auf.

Argumente
Meldung.

Rückgabewerte
Keine.

dom.hideInfoBar()

Verfügbarkeit
Dreamweaver CS4.

Beschreibung
Mit dieser Funktion wird die Informationsleiste ausgeblendet.

Argumente
Keine.

Rückgabewerte
Keine.

Funktionen für zugehörige Dateien

Die Funktionen für zugehörige Dateien erleichtern Programmierern das Bearbeiten ihres Quellcodes, indem problemlos auf aktiv verwendete zugehörige und unterstützende Dateien zugegriffen werden kann.

dreamweaver.getRelatedFiles()

Verfügbarkeit
Dreamweaver CS4.

Beschreibung
Diese Funktion ruft eine Liste aller zugehörigen Dateien ab. Die zugehörigen Dateien können untergeordnete Dokumente, HTML-Quelldateien und generierte Quelldateien sein.

Argumente
Ein boolescher Wert, der die Anzeigenamen des übergeordneten Dokuments und der generierten Quelldateien festlegt.

- Verwenden Sie den Wert `true`, wenn im Menü HTML-Quelldateien und generierte Quelldateien angezeigt werden sollen.
- Verwenden Sie den Wert `false`, wenn im Menü die tatsächlichen Namen der zugehörigen Dateien angezeigt werden sollen.

Rückgabewerte
Ein Array von Strings, das alle analysierten zugehörigen Dateien als absolute lokale URLs enthält.

dreamweaver.openRelatedFile()

Verfügbarkeit
Dreamweaver CS4.

Beschreibung
Zeigt die ausgewählte zugehörige Datei in der Codeansicht des aktuellen Dokuments an.

Argumente
Ein String, der die absolute lokale URL der Datei enthält.

Rückgabewerte
Keine.

dreamweaver.getActiveRelatedFilePath()

Verfügbarkeit
Dreamweaver CS4.

Beschreibung
Diese Funktion ruft den vollständigen Pfad der aktuellen geöffneten Datei ab.

Argumente
Keine.

Rückgabewerte
Ein String, der die absolute lokale URL der zugehörigen Datei enthält.

dreamweaver.getRelatedFilesFilter()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Diese Funktion wird verwendet, um den Dateinamenfilter auf die zugehörigen Dateien anzuwenden.

Argumente
Keine.

Rückgabewerte
Ein `DWFilenameFilter`-Objekt für den Filter, der derzeit auf die zugehörigen Dateien angewendet wird. Ein leeres Filterobjekt gibt an, dass in der Leiste „Zugehörige Dateien“ alle Dateien angezeigt werden.

`DWFilenameFilter`-Objekte sind in Dreamweaver CS5 neu hinzugekommen. Mit diesen Objekten können die in der Leiste „Zugehörige Dateien“ angezeigten Dateien beschränkt werden.

dreamweaver.setRelatedFilesFilter()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Diese Funktion wird verwendet, um den auf die Leiste „Zugehörige Dateien“ anzuwendenden Filter festzulegen.

Argumente
Ein String oder ein DWFilenameFilter-Objekt. Beispiel: "`.js`", "`.php`", "`*.js`", "`*.php.js`", "`*.css`", "", "`b*.js`", "`b*.*`", "`*.js;.css`". Bei einem leeren String werden in der Leiste „Zugehörige Dateien“ alle Dateien angezeigt.

Rückgabewerte
Keine.

dreamweaver.getQuickRelatedFilesFilterStrings()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Diese Funktion wird verwendet, um ein Array von Stringobjekten abzurufen, die die Dateierweiterungen der in der Leiste „Zugehörige Dateien“ angezeigten Dateien darstellen.

Argumente
Keine.

Rückgabewerte
Ein Array von Stringobjekten, die die Dateierweiterungen der in der Leiste „Zugehörige Dateien“ angezeigten Dateien darstellen. Beispiel: {"`.js`", "`.php`", "`.css`"}.

dreamweaver.invokeRelatedFilesCustomFilterDialog()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Diese Funktion wird verwendet, um das benutzerdefinierte Dialogfeld zum Filtern der zugehörigen Dateien aufzurufen. Der Benutzer kann in diesem Dialogfeld einen Filter anwenden. Zum Ermitteln des neuen Filters soll die Funktion `getRelatedFilesFilter` aufgerufen werden.

Argumente
Keine.

Rückgabewerte
Keine.

dreamweaver.getDynamicRelatedFilesDiscoverySetting()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Die Funktion „Dynamisch zugehörige Dateien“ erweitert die Funktionalität der Funktion „Zugehörige Dateien“ dadurch, dass Sie die zugehörigen Dateien dynamischer Seiten auf der Leiste „Zugehörige Dateien“ anzeigen können.

Diese Funktion wird verwendet, um die Einstellung für die Erkennung dynamisch zugehöriger Dateien abzurufen.

Argumente
Keine.

Rückgabewerte
Ein Stringobjekt, das den Erkennungsmechanismus angibt. Enthält einen der folgenden Werte:

Wert	Beschreibung
automatic	Aktiviert die automatische Erkennung zugehöriger Dateien.
manual	Aktiviert die manuelle Erkennung zugehöriger Dateien. In diesem Fall muss der Benutzer die Verweise auf die zugehörigen Dateien manuell auflösen.
disabled	Deaktiviert die automatische Erkennung zugehöriger Dateien.

dreamweaver.setDynamicRelatedFilesDiscoverySetting()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Diese Funktion wird verwendet, um die Einstellung für die Erkennung dynamisch zugehöriger Dateien festzulegen.

Argumente
Ein Stringobjekt, das den Erkennungsmechanismus angibt. Enthält einen der folgenden Werte:

Wert	Beschreibung
automatic	Aktiviert die automatische Erkennung zugehöriger Dateien.
manual	Aktiviert die manuelle Erkennung zugehöriger Dateien. In diesem Fall muss der Benutzer die Verweise auf die zugehörigen Dateien manuell auflösen.
disabled	Deaktiviert die automatische Erkennung zugehöriger Dateien.

Rückgabewerte
Keine.

dreamweaver.refreshRelatedFiles()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Diese Funktion wird verwendet, um die in der Leiste „Zugehörige Dateien“ angezeigten zugehörigen Dateien für das aktuelle Dokument zu aktualisieren.

Argumente
Keine.

Rückgabewerte
Keine.

dreamweaver.saveAllRelatedFiles()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Diese Funktion wird verwendet, um alle geänderten zugehörigen Dateien für das aktuelle Dokument zu speichern. Beim Aufrufen der Funktion werden das aktuelle Dokument und alle geänderten zugehörigen Dateien für das aktuelle Dokument gespeichert. Wenn es sich um ein neu erstelltes Dokument handelt, wird das Dialogfeld „Speichern unter“ angezeigt.

Argumente
Keine.

Rückgabewerte
Keine.

dreamweaver.canSaveAllRelatedFiles()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Mit dieser Funktion wird ermittelt, ob eine der zugehörigen Dateien des aktuellen Dokuments geändert wurde und gespeichert werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn eine der zugehörigen Dateien geändert wurde und gespeichert werden kann.

document.isRelatedFileViewOpen()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Mit dieser Funktion wird ermittelt, ob im Codeansichtsfenster derzeit der Quellcode des Dokuments angezeigt wird.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn im Codeansichtsfenster der Quellcode des Dokuments angezeigt wird. Wenn im Codeansichtsfenster der Live-Code einer zugehörigen Datei angezeigt wird, wird `false` zurückgegeben.

document.getRelatedFiles()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Mit dieser Funktion wird die Liste der zugehörigen Dateien des entsprechenden Dokuments abgerufen.

Argumente
filtered

Ein boolescher Wert: Bei `true` wird der in der Leiste „Zugehörige Dateien“ ausgewählte Filter auf das Ergebnis angewendet. Verwenden Sie `false`, um alle zugehörigen Dateien des Dokuments abzurufen. Dieses Argument ist optional. Der Standardwert ist `false`.

type

Ein optionaler String, der den Typ der zugehörigen Dateien angibt. Dabei kann es sich um einen leeren String oder einen String mit einem der folgenden Werte handeln:

- `SOURCE_HTML` - Ruft das HTML-Quelldokument ab, d. h. das oberste Dokument.
- `GENERATED_HTML` - Ruft das live erzeugte HTML-Dokument ab. Diese Option ist nur gültig, wenn die Funktion „Live-Code“ aktiviert ist.
- `CHILD_DOC` - Ruft die Liste der zugehörigen Dokumente mit statischer Pfadangabe ab.
- `PROCESSED_CHILD_DOC` - Ruft die Liste der zugehörigen Dokumente ab, die vom Server verarbeitet werden. Diese Option ist nur gültig, wenn dynamisch zugehörige Dateien erkannt wurden und das vom Server erzeugte Ergebnis auf untergeordnete Dokumente verweist, die nicht bereits beim Öffnen des Dokuments als zugehörige Dokumente mit statischem Pfad gefunden wurden.
- `LIVE_VIEW_CHILD` - Ruft die Liste der zugehörigen untergeordneten Live-Ansicht-Dokumente ab. Diese Option ist nur gültig, wenn das Dokument in der Live-Ansicht geöffnet ist und die vom Server erzeugte Quelle auf untergeordnete Dokumente verweist, die nicht bereits beim Öffnen des Dokuments als zugehörige Dokumente mit statischem Pfad gefunden wurden.
- `LIVE_VIEW_XHR_CHILD` - Ruft die Liste der Live-Ansicht-Dokumente für Ressourcenverweise ab. Diese Option ist nur gültig, wenn das Dokument in der Live-Ansicht geöffnet ist.
- `DYNAMIC_PATH_CHILD_DOC` - Ruft die Liste der zugehörigen Dateien mit dynamischer Pfadangabe ab. Diese Option ist nur gültig, wenn dynamisch zugehörige Dateien erkannt und während der Erkennung untergeordnete dynamisch zugehörige Dokumente gefunden wurden.
- `USER_DEFINED_CHILD_DOC` - Ruft die Liste der benutzerdefinierten zugehörigen Dateien mit dynamischer Pfadangabe ab. Diese Option ist nur gültig, wenn nach dem Aufrufen der Funktion `addRelatedFile()` mit einer Erweiterung benutzerdefinierte zugehörige Dateien eingefügt wurden. Weitere Informationen zur Funktion `addRelatedFile()` finden Sie unter „document.addRelatedFile()“ auf Seite 203.
- `ALL_TYPES` - Ruft alle zugehörigen Dateien ab. Dies ist der Standardwert.

Rückgabewerte
Ein Array von Objekten für zugehörige Dateien. Jedes Objekt hat folgende Eigenschaften:

- `uri` - Ein `DWUri`-Objekt, das den URI des zugehörigen Dokuments darstellt.

- `type` - Ein String mit einem der im Abschnitt „Argumente“ aufgeführten Typen oder bei unbekanntem Typ der Wert „UNKNOWN_TYPE“.
- `document` - Das Dokumentobjekt des zugehörigen Dokuments. Wenn es kein zugehöriges Dokument gibt, hat diese Eigenschaft den Wert „NULL“.
- `isChildDocType` - Ein boolescher Wert. Der Wert ist „true“, wenn das zugehörige Dokument ein untergeordnetes Dokument (nicht die Quelle oder das erzeugte Quelldokument) ist.
- `isSelectedDoc` - Ein boolescher Wert. Der Wert ist „true“, wenn das Dokument ausgewählt ist.

document.addRelatedFile()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Mit dieser Funktion wird eine in einer Erweiterung definierte zugehörige Datei hinzugefügt.

Achten Sie als Entwickler von Erweiterungen darauf, die Funktion `refreshRelatedFiles()` nach dem Aufrufen von `addRelatedFile()` aufzurufen. Weitere Informationen zur Funktion `refreshRelatedFiles()` finden Sie unter „dreamweaver.refreshRelatedFiles()“ auf Seite 200.

Argumente
- `uri` - Ein `DWUri`-Objekt, das den URI der zugehörigen Datei darstellt. Dieses Argument ist erforderlich.
- `persistent` - Ein boolescher Wert. Der Wert ist „true“, wenn das Objekt der zugehörigen Datei beim erneuten Durchsuchen des Dokuments auf oberster Ebene erhalten bleibt. Dieses Argument ist optional.
- type - Ein optionaler String, der in der Funktion „document.getRelatedFiles()“ angegeben wird. Wenn dieser Parameter beim Aufruf nicht angegeben wird, wird als Typ der zugehörigen Datei der Wert `USER_DEFINED_CHILD_DOC` eingefügt.

Rückgabewerte
Keine.

document.removeRelatedFile()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Mit dieser Funktion wird eine in einer Erweiterung definierte zugehörige Datei entfernt.

Mit dieser Funktion können nur Objekte für zugehörige Dokumente des Typs `CHILD_DOC` entfernt werden. Objekte für zugehörige Dokumente des Typs `SOURCE_HTML` oder `GENERATED_HTML` können dagegen mit dieser Funktion nicht entfernt werden.

Argumente
`uri` - Ein `DWUri`-Objekt, das den URI der zugehörigen Datei darstellt. Dieses Argument ist erforderlich.

Rückgabewerte
Keine.

document.getDependentFiles()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Diese Funktion wird verwendet, um die abhängigen Dateien des Dokumentobjekts abzurufen.

Die Liste der abhängigen Dateien ist eine Aufstellung der Dateien, die zum Server übertragen werden, wenn sie in der Browservorschau oder in der Live-Ansicht geändert werden bzw. wenn das Dokument vom Server oder aus der Versionskontrolle abgerufen wird.

Argumente
Keine.

Rückgabewerte
Ein Array von `DWUri`-Objekten, das die Liste der abhängigen Dateien darstellt.

DWFilenameFilter-Referenz

Die Objekte des Typs `DWFilenameFilter` sind in Dreamweaver CS5 neu hinzugekommen und werden verwendet, um die in der Leiste „Zugehörige Dateien“ angezeigten Dateien zu beschränken. Zum Bearbeiten der Filter in der Leiste „Zugehörige Dateien“ können Entwickler von Erweiterungen ein neues DWFilenameFilter-Objekt erstellen und das Verhalten des Filters mithilfe der in den folgenden Abschnitten beschriebenen Funktionen ändern.

Beispiel:

```
var filter = new DWFilenameFilter;
filter.setExpression('*.js');
dw.setRelatedFilesFilter(filter);
```

Entwickler von Erweiterungen können auch den aktuellen auf die Leiste „Zugehörige Dateien“ angewendeten Filter bearbeiten, indem sie die Funktion `dreamweaver.getRelatedFilesFilter()` aufrufen (siehe „dreamweaver.getRelatedFilesFilter()“ auf Seite 198) und dann das Verhalten des Filters mithilfe der anderen Funktionen ändern.

Beispiel:

```
var filter = dw.getRelatedFilesFilter();
filter.addExtensionToExclusionExpression('.js');
dw.setRelatedFilesFilter(filter);
```

DWFilenameFilter.isValidFilterExpression()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Mit dieser Funktion wird ermittelt, ob ein Ausdruck ein gültiger Filterausdruck ist.

Argumente
`expression` - Ein String, der einen Filterausdruck darstellt.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der Ausdruck gültig ist.

DWFilenameFilter.isEmpty()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Mit dieser Funktion wird ermittelt, ob ein Filterobjekt leer ist.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn das Filterobjekt leer ist.

DWFilenameFilter.doesExcludeExtension()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Mit dieser Funktion wird ermittelt, ob eine Erweiterung durch das Filterobjekt ausgeschlossen wird.

Filter verwenden einen Ausschlusstest, um zu ermitteln, ob eine im Ausschlussausdruck angegebene Dateierweiterung ausgeschlossen ist. Verwenden Sie diese Funktion, um zu ermitteln, ob eine bestimmte Dateierweiterung zuvor mithilfe der Funktion `DWFilenameFilter.addExtensionToExclusionExpression()` zur Ausschlussliste hinzugefügt wurde. Weitere Informationen finden Sie unter „DWFilenameFilter.addExtensionToExclusionExpression()“ auf Seite 208.

Argumente
`extension` - Ein String, der eine Dateierweiterung darstellt.

Rückgabewerte
Ein boolescher Wert: `true`, wenn die Erweiterung durch den Filter ausgeschlossen wird.

DWFilenameFilter.isAdvancedFilter()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Mit dieser Funktion wird ermittelt, ob es sich bei einem Filterobjekt um einen erweiterten Filter handelt.

Ein Filter wird als erweiterter Filter eingestuft, wenn der Benutzer den Menüeintrag „Erweitert...“ im Menü „Filter“ auswählt.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der Filter ein erweiterter Filter ist.

DWFilenameFilter.willMatchAnyFile()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Mit dieser Funktion wird ermittelt, ob ein Filterobjekt beliebige Dateien zurückgibt. Ein solcher Filter bewirkt, dass in der Leiste „Zugehörige Dateien“ alle Dateien angezeigt werden.

Um beliebige Dateien zurückzugeben, muss ein Filterobjekt entweder leer sein oder es ist ein erweiterter Filter mit dem Wert „`*.*`“ als Filterausdruck.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der Filter beliebige Dateien zurückgibt.

DWFilenameFilter.getExpression()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Mit dieser Funktion wird der Filterausdruck abgerufen.

Argumente
Keine.

Rückgabewerte
Ein String, der den Filterausdruck darstellt.

Wenn der Filter beliebige Dateien zurückgibt, wird ein leerer String zurückgegeben.

DWFilenameFilter.setExpression()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Mit dieser Funktion wird der Filterausdruck festgelegt.

Argumente
`expression` - Ein String, der einen Filterausdruck darstellt.

Bei diesem Argument handelt es sich um einen oder mehrere Platzhalter-Filterausdrücke. Mehrere Ausdrücke sind jeweils durch ein Semikolon getrennt. Beispiel: "`*.css;help*.js`".

Wenn für das Filterobjekt ein leerer String festgelegt wird, gibt der Filter beliebige Dateien zurück.

Rückgabewerte
Keine.

DWFilenameFilter.getExcludedExtensions()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Mit dieser Funktion wird die Liste aller ausgeschlossenen Dateierweiterungen abgerufen.

Argumente
Keine.

Rückgabewerte
Ein Array von Stringobjekten mit den Dateierweiterungen, die durch den Filter ausgeschlossen werden sollen. Siehe auch „DWFilenameFilter.addExtensionToExclusionExpression()“ auf Seite 208.

Beispiel: {`".php", ".css", ".engine"`}

DWFilenameFilter.getExclusionExpression()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Mit dieser Funktion wird der Ausschlussausdruck abgerufen.

Argumente
Keine.

Rückgabewerte
Ein String mit durch Semikola getrennten Werten, der die Liste der auszuschließenden Dateierweiterungen darstellt.

Beispiel: `".php;.css;.engine"`

DWFilenameFilter.getAdvancedExpression()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Mit dieser Funktion wird der erweiterte Ausdruck abgerufen.

Bei einem Filter kann es sich entweder um einen erweiterten Filter mit einer Aufzählung von Platzhalter-Einschlussfiltern handeln oder um einen einfachen Filter mit einer Liste auszuschließender Dateierweiterungen.

Argumente
Keine.

Rückgabewerte
Ein String mit durch Semikola getrennten Werten, der die Liste der Filter im erweiterten Filterausdruck darstellt.

Beispiel: `"*.css;help*.js"`

DWFilenameFilter.addExtensionToExclusionExpression()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Mit dieser Funktion wird der Ausschlussliste des Filterobjekts eine Dateierweiterung hinzugefügt.

Argumente
`extension` - Ein String oder ein `DWUri`-Objekt.

Dieses Argument ist ein Wert für den Dateinamen oder die Dateierweiterung, der dem Filter hinzugefügt werden soll.

Das Argument kann eine Dateierweiterung, ein vollständig qualifizierter lokaler Dateiname, eine als String angegebene Remote-URL oder ein gültiges `DWUri`-Objekt sein.

Rückgabewerte
Ein String mit durch Semikola getrennten Werten, der die Liste der Filter im erweiterten Filterausdruck darstellt.

Beispiel: `"*.css;help*.js"`

DWFilenameFilter.removeExtensionFromExclusionExpression()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Mit dieser Funktion wird eine Dateierweiterung aus der Ausschlussliste des Filterobjekts entfernt.

Argumente
`extension` - Ein String oder ein `DWUri`-Objekt.

Dieses Argument ist ein Wert für den Dateinamen oder die Dateierweiterung, der aus dem Filter entfernt werden soll.

Das Argument kann eine Dateierweiterung, ein vollständig qualifizierter lokaler Dateiname, eine als String angegebene Remote-URL oder ein gültiges `DWUri`-Objekt sein.

Rückgabewerte
Keine.

DWFilenameFilter.empty()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Mit dieser Funktion wird der Filterausdruck zurückgesetzt.

Ein leerer Filter gibt alle Dateien zurück.

Argumente
Keine.

Rückgabewerte
Keine.

Funktionen für die vertikal geteilte Ansicht

Mit den Funktionen für die vertikal geteilte Ansicht können entweder die Code- und die Entwurfsansicht oder die Codeansicht und der Layoutmodus nebeneinander angezeigt werden. Die Funktionen ermöglichen den Benutzern auch, zwischen der horizontalen und vertikalen Ausrichtung der geteilten Ansicht oder Codeteilung zu wechseln.

dreamweaver.getSplitViewOrientation()

Verfügbarkeit
Dreamweaver CS4.

Beschreibung
Diese Funktion ermittelt die aktuelle Ausrichtung der geteilten Ansicht. Die Ausrichtung kann auch abgerufen werden, wenn die Ansicht nicht geteilt ist. Dabei gibt der Rückgabewert die Ausrichtung für den Fall an, dass die Ansicht zur geteilten Ansicht oder zur Codeteilung wechselt.

Argumente
Keine.

Rückgabewerte
Ein Stringwert, der die Ausrichtung festlegt. Gibt je nach aktueller Ausrichtung den Wert `vertical` bzw. `horizontal` zurück.

dreamweaver.setSplitViewOrientation()

Verfügbarkeit
Dreamweaver CS4.

Beschreibung
Diese Funktion ändert die aktuelle Ausrichtung der geteilten Ansicht. Die Ausrichtung kann auch geändert werden, wenn die Ansicht nicht geteilt ist. In diesem Fall gibt das Argument die Ausrichtung für den nächsten Wechsel der Ansicht zur geteilten Ansicht oder zur Codeteilung an.

Argumente
Ein Stringwert, der die Ausrichtung angibt. Verwenden Sie zum Angeben der Ausrichtung die Werte `vertical` oder `horizontal`. Dieses Argument ist erforderlich.

Rückgabewerte
Ein boolescher Wert: `true`, wenn erfolgreich, andernfalls `false`.

dreamweaver.getPrimaryView()

Verfügbarkeit
Dreamweaver CS4.

Beschreibung
Diese Funktion ruft den Namen der primären Ansicht ab. In der geteilten Ansicht oder der Codeteilung ist die primäre Ansicht je nach Ausrichtung das obere bzw. das linke Fenster.

Argumente
Keine.

Rückgabewerte
Ein String mit dem Namen der primären Ansicht, der einen der folgenden Werte haben kann:

Wert	Beschreibung
`code`	Die primäre Ansicht ist das Codefenster.
`design`	Die primäre Ansicht ist das Entwurfsfenster.
`related file`	Die primäre Ansicht ist das Fenster für zugehörige Dateien. Dieser Wert wird zurückgegeben, wenn die Dokumentansicht geteilt und eine zugehörige Datei geöffnet ist.

dreamweaver.setPrimaryView()

Verfügbarkeit
Dreamweaver CS4.

Beschreibung
Diese Funktion ändert die primäre Ansicht. In der geteilten Ansicht oder der Codeteilung ist die primäre Ansicht je nach Ausrichtung das obere bzw. das linke Fenster.

Argumente
Ein String mit dem Namen der primären Ansicht, der einen der folgenden Werte haben kann:

Wert	Beschreibung
code	Die primäre Ansicht ist das Codefenster.
design	Die primäre Ansicht ist das Entwurfsfenster.
related file	Die primäre Ansicht ist das Fenster für zugehörige Dateien. Dieser Wert wird verwendet, wenn die Dokumentansicht geteilt und eine zugehörige Datei geöffnet ist.

Rückgabewerte
Ein boolescher Wert: `true`, wenn erfolgreich, andernfalls `false`.

dom.isRelatedFileViewOpen()

Verfügbarkeit
Dreamweaver CS4.

Beschreibung
Diese Funktion ermittelt, ob die Ansicht eine Ansicht für zugehörige Dateien enthält.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn es sich beim aktuellen Dokument um eine Vorlage handelt, andernfalls `false`.

Funktionen für das Ausblenden von Code

Funktionen für das Ausblenden von Code erlauben Ihnen, den Code visuell aus- und einzublenden. Sie können eine beliebige Codeauswahl oder Fragmente zwischen öffnenden und schließenden Tags aus- oder einblenden. Zwar sind diese Funktionen sowohl im DOM- als auch im HTML-Inspektor verfügbar, die ausgeblendeten Bereiche in der Codeansicht und im Codeinspektor sind jedoch dieselben.

dom.collapseFullTag()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion bestimmt, ob die Auswahl in der Codeansicht sich gänzlich innerhalb eines einzigen Start-End-Tag-Paars befindet oder ein einziges Paar Start- und End-Tags enthält. Ist das der Fall, wird das Codefragment ausgeblendet, das direkt vor dem Start-Tag beginnt und hinter dem End-Tag endet. Andernfalls hat die Funktion keine Auswirkung.

Argumente
allowCodeFragmentAdjustment

- Das Argument *allowCodeFragmentAdjustment* ist ein obligatorischer boolescher Wert. Wenn `true`, hat dieses Argument momentan keine Auswirkung bzw. dieselbe Auswirkung wie der Wert `false`. Wenn `false`, blendet Dreamweaver den Code aus, der unmittelbar vor dem öffnenden Tag beginnt und unmittelbar hinter dem schließenden Tag endet, und zwar ohne Veränderungen.

Rückgabewerte
Keine.

Beispiel
Im folgenden Beispiel wird das Codefragment in der aktuellen Auswahl der Codeansicht ausgeblendet, das direkt vor dem Start-Tag beginnt und nach dem End-Tag endet:

```
var currentDOM = dw.getDocumentDOM();
currentDOM.collapseFullTag(false);
```

dom.collapseFullTagInverse()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion bestimmt, ob die Auswahl in der Codeansicht sich gänzlich innerhalb eines einzigen Start-End-Tag-Paars befindet oder ein einziges Paar Start- und End-Tags enthält. Ist das der Fall, wird der Code vor dem Start-Tag und hinter dem End-Tag ausgeblendet. Andernfalls hat die Funktion keine Auswirkung.

Argumente
allowAdjustmentOfCodeFragments

- Das Argument *allowAdjustmentOfCodeFragments* ist ein obligatorischer boolescher Wert. Wenn `true`, passt Dreamweaver die Begrenzungen des Codes *vor* dem Start-Tag und *hinter* dem End-Tag an, um eine *intelligente Ausblendung* durchzuführen, d. h., aktuelle Einzüge und Leerräume bleiben erhalten. Wenn `false`, blendet Dreamweaver die Codefragmente *vor* dem öffnenden Tag und *hinter* dem schließenden Tag aus, genau wie von der Auswahl angezeigt.

Rückgabewerte
Keine.

Beispiel
Im folgenden Beispiel werden die Begrenzungen des Codes vor dem Start-Tag und hinter dem End-Tag angepasst, um eine *intelligente Ausblendung* durchzuführen, bei der die Einzüge und Leerräume erhalten bleiben:

```
var currentDOM = dw.getDocumentDOM();
currentDOM.collapseFullTagInverse(true);
```

dom.collapseSelectedCodeFragment()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion blendet den in der Codeansicht ausgewählten Code aus. Ist die Auswahl bereits ausgeblendet, hat diese Funktion keine Auswirkung.

Argumente
allowCodeFragmentAdjustment

- Das Argument *allowCodeFragmentAdjustment* ist ein obligatorischer boolescher Wert. Wenn `true`, ändert Dreamweaver die Begrenzungen der aktuellen Auswahl, um eine *intelligente Ausblendung* durchzuführen, d. h., aktuelle Einzüge und Leerräume bleiben erhalten. Wenn `false`, blendet Dreamweaver das aktuell ausgewählte Codefragment wie in der Auswahl angezeigt aus.

Rückgabewerte
Keine.

Beispiel
Im folgenden Beispiel wird das ausgewählte Codefragment ohne Änderung in der Codeansicht ausgeblendet.

```
var currentDOM = dw.getDocumentDOM();
currentDOM.collapseSelectedCodeFragment(false);
```

dom.collapseSelectedCodeFragmentInverse()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion blendet den gesamten Code *vor* und *hinter* dem in der Codeansicht ausgewählten Code aus.

Argumente
allowAdjustmentOfCodeFragments

- Das Argument *allowAdjustmentOfCodeFragments* ist ein obligatorischer boolescher Wert. Wenn `true`, passt Dreamweaver die Begrenzungen des Codes *vor* und *hinter* der aktuellen Auswahl an, um eine *intelligente Ausblendung* durchzuführen, d. h., aktuelle Einzüge und Leerräume bleiben erhalten. Wenn `false`, blendet Dreamweaver die Codefragmente wie in der Auswahl angezeigt aus.

Rückgabewerte
Keine.

Beispiel
Im folgenden Beispiel wird der gesamte Code vor und hinter dem in der Codeansicht ausgewählten Code angepasst und dann ausgeblendet.

```
var currentDOM = dw.getDocumentDOM();
currentDOM.collapseSelectedCodeFragmentInverse(true);
```

dom.expandAllCodeFragments()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion blendet alle in der Codeansicht ausgeblendeten Codefragmente wieder ein, einschließlich verschachtelter ausgeblendeter Codefragmente.

Argumente
Keine.

Rückgabewerte
Keine.

Beispiel
Im folgenden Beispiel wird der gesamte in der Codeansicht ausgeblendete Code wieder eingeblendet:

```
var currentDOM = dw.getDocumentDOM();
currentDOM.expandAllCodeFragments();
```

dom.expandSelectedCodeFragments()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion blendet alle in der Codeansicht ausgeblendeten Codefragmente wieder ein, die sich innerhalb der aktuellen Auswahl befinden. Umfasst die Auswahl keine Ausblendungen, hat diese Funktion keine Auswirkung.

Argumente
Keine.

Rückgabewerte
Keine.

Beispiel
Im folgenden Beispiel wird der gesamte ausgeblendete Code in der aktuellen Auswahl in der Codeansicht wieder eingeblendet:

```
var currentDOM = dw.getDocumentDOM();
currentDOM.expandSelectedCodeFragments();
```

dreamweaver.htmlInspector.collapseFullTag()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion bestimmt, ob die Auswahl im Codeinspektor sich gänzlich innerhalb eines einzigen Start-End-Tag-Paars befindet oder ein einziges Paar Start- und End-Tags enthält. Ist das der Fall, wird das Codefragment ausgeblendet, das direkt vor dem Start-Tag beginnt und hinter dem End-Tag endet. Andernfalls hat die Funktion keine Auswirkung.

Argumente
allowACodeFragmentAdjustment

- Das Argument *allowCodeFragmentAdjustment* ist ein obligatorischer boolescher Wert. Wenn `true`, hat dieses Argument momentan keine Auswirkung bzw. dieselbe Auswirkung wie der Wert `false`. Wenn `false`, blendet Dreamweaver den Code aus, der unmittelbar vor dem öffnenden Tag beginnt und unmittelbar hinter dem schließenden Tag endet, und zwar ohne Veränderungen.

Rückgabewerte
Keine.

Beispiel
Im folgenden Beispiel wird das Codefragment in der aktuellen Auswahl des Codeinspektors ausgeblendet, das direkt vor dem Start-Tag beginnt und nach dem End-Tag endet:

```
dreamweaver.htmlInspector.collapseFullTag(false);
```

dreamweaver.htmlInspector.collapseFullTagInverse()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion bestimmt, ob die Auswahl im Codeinspektor sich gänzlich innerhalb eines einzigen Start-End-Tag-Paars befindet oder ein einziges Paar Start- und End-Tags enthält. Ist das der Fall, wird der Code *vor* dem Start-Tag und *hinter* dem End-Tag ausgeblendet. Andernfalls hat die Funktion keine Auswirkung.

Argumente
allowAdjustmentOfCodeFragments

- Das Argument *allowAdjustmentOfCodeFragments* ist ein obligatorischer boolescher Wert. Wenn `true`, passt Dreamweaver die Begrenzungen des Codes vor dem Start-Tag und hinter dem End-Tag an, um eine *intelligente Ausblendung* durchzuführen, d. h., vorhandene Einzüge und Leerräume bleiben erhalten. Wenn `false`, blendet Dreamweaver den Code *vor* dem öffnenden Tag und *hinter* dem schließenden Tag aus, und zwar ohne Veränderungen.

Rückgabewerte
Keine.

Beispiel
Im folgenden Beispiel wird auf die Codeabschnitte vor dem Start- und hinter dem End-Tag der aktuellen Auswahl eine *intelligente Ausblendung* angewendet:

```
dreamweaver.htmlInspector.collapseFullTagInverse(true);
```

dreamweaver.htmlInspector.collapseSelectedCodeFragment()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion blendet den im Codeinspektor ausgewählten Code aus. Ist die Auswahl bereits ausgeblendet, hat diese Funktion keine Auswirkung.

Argumente
allowCodeFragmentAdjustment

- Das Argument *allowCodeFragmentAdjustment* ist ein obligatorischer boolescher Wert. Wenn `true`, ändert Dreamweaver die aktuelle Auswahl, um eine *intelligente Ausblendung* durchzuführen, d. h., vorhandene Einzüge und Leerräume bleiben erhalten. Wenn `false`, blendet Dreamweaver das aktuell ausgewählte Codefragment wie in der Auswahl angezeigt aus.

Rückgabewerte
Keine.

Beispiel
Im folgenden Beispiel wird der im Codeinspektor ausgewählte Code angepasst und ausgeblendet:

```
dreamweaver.htmlInspector.collapseSelectedCodeFragment(true);
```

dreamweaver.htmlInspector.collapseSelectedCodeFragmentInverse()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion blendet den gesamten Code *vor* und *hinter* dem im Codeinspektor ausgewählten Code aus. Ist die Auswahl bereits ausgeblendet, hat diese Funktion keine Auswirkung.

Argumente
allowAdjustmentOfCodeFragments

- Das Argument *allowAdjustmentOfCodeFragments* ist ein obligatorischer boolescher Wert. Wenn `true`, passt Dreamweaver die Begrenzungen der Codeabschnitte vor und hinter der aktuellen Auswahl an, um eine *intelligente Ausblendung* durchzuführen, d. h., aktuelle Einzüge und Leerräume bleiben erhalten. Wenn `false`, blendet Dreamweaver die Codeabschnitte wie in der Auswahl angezeigt aus.

Rückgabewerte
Keine.

Beispiel
Im folgenden Beispiel wird der gesamte Code vor und hinter dem im Codeinspektor ausgewählten Code ausgeblendet, genau wie in der Auswahl angezeigt:

```
dreamweaver.htmlInspector.collapseSelectedCodeFragmentInverse(false);
```

dreamweaver.htmlInspector.expandAllCodeFragments()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion blendet alle im Codeinspektor ausgeblendeten Codefragmente wieder ein, einschließlich verschachtelter ausgeblendeter Codefragmente.

Argumente
Keine.

Rückgabewerte
Keine.

Beispiel
Im folgenden Beispiel wird der gesamte im Codeinspektor ausgeblendete Code wieder eingeblendet:

```
dreamweaver.htmlInspector.expandAllCodeFragments();
```

dreamweaver.htmlInspector.expandSelectedCodeFragments()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion blendet alle innerhalb der aktuellen Auswahl im Codeinspektor ausgeblendeten Codefragmente wieder ein. Umfasst die Auswahl keine Ausblendungen, hat diese Funktion keine Auswirkung.

Argumente
Keine.

Rückgabewerte
Keine.

Beispiel
Im folgenden Beispiel wird der gesamte ausgeblendete Code in der aktuellen Auswahl im Codeinspektor wieder eingeblendet:

```
dreamweaver.htmlInspector.expandSelectedCodeFragments();
```

Symbolleistenfunktionen der Codeansicht

Mithilfe der Symbolleistenfunktionen der Codeansicht können Sie Text einfügen, Kommentare entfernen, Sonderzeichen für Leerräume anzeigen oder verbergen und den Pfad des aktuellen Dokuments abrufen.

Hinweis: *Es gibt zwei unterschiedliche Code-Symbolleisten: eine für die Codeansicht und eine für den Codeinspektor. Beide werden in der Datei „Configuration/Toolbars/toolbars.xml“ angepasst.*

dom.getOpenPathName()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion ruft den absoluten Dateipfad des geöffneten Dokuments ab.

Argumente
Keine.

Rückgabewerte
Ein String, der den absoluten Dateipfad des geöffneten Dokuments beschreibt.

Beispiel
Im folgenden Beispiel wird der String mit dem Pfad des gerade geöffneten Dokuments der Variablen `fileName` zugewiesen:

```
var fileName = dom.getOpenPathName();
```

dom.getShowHiddenCharacters()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion bestimmt, ob in der Codeansicht des Dokumentfensters die Sonderzeichen für Leerräume angezeigt werden.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn die versteckten Zeichen angezeigt werden, andernfalls `false`.

Beispiel
Im folgenden Beispiel wird die Anzeige der Leerraum-Sonderzeichen deaktiviert, wenn die Anzeige ursprünglich aktiviert war:

```
var currentDOM = dw.getDocumentDOM();
if (currentDOM.getShowHiddenCharacters()){
    currentDOM.setShowHiddenCharacters(false);
    }
```

dom.setShowHiddenCharacters()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion zeigt im Codeinspektor die Sonderzeichen für Leerräume an bzw. verbirgt sie.

Ein Beispiel finden Sie unter „dom.getShowHiddenCharacters()“ auf Seite 219.

Argumente
show

- Das obligatorische Argument *show* ist ein boolescher Wert, der angibt, ob die versteckten Zeichen angezeigt werden sollen.

Rückgabewerte
Keine.

dom.source.applyComment()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion fügt den im Argument *beforeText* angegebenen Text vor der aktuellen Auswahl ein und den im Argument *afterText* angegebenen Text hinter der aktuellen Auswahl. Die Funktion erweitert dann die aktuelle Auswahl um den hinzugefügten Text. Wurde kein Text ausgewählt, wählt die Funktion nichts aus. Ist der im Argument *afterText* angegebene Text null, fügt die Funktion den im Argument *beforeText* angegebenen Text zu Beginn jeder Zeile in der aktuellen Auswahl ein.

Argumente
beforeText, afterText

- Das Argument *beforeText* ist obligatorisch. Dieses Argument beschreibt den Text, der am Anfang der Auswahl einzufügen ist, bzw. den Text, der am Anfang jeder Zeile in der Auswahl einzufügen ist, wenn der Wert des Arguments *afterText* null ist.
- Das (optionale) Argument *afterText* beschreibt den Text, der am Ende der Auswahl eingefügt werden soll.

Rückgabewerte
Keine.

Beispiel
Im folgenden Beispiel wird aus der aktuellen Auswahl ein HTML-Kommentar gemacht:

```
dw.getDocumentDOM().source.applyComment('<!--', '-->')
```

dom.source.refreshVariableCodeHints()

Verfügbarkeit
Dreamweaver CS3.

Beschreibung
Durchsucht die Seite erneut nach Variablen und entsprechenden Klassenzuweisungen. Mit dieser Funktion werden die Farbzustands-Engine und die Variablenliste neu erstellt.

Argumente
bSyncDoc

- Dies ist ein boolescher Wert. Die Standardeinstellung ist `false`. Bei dem Wert `true` wird die Entwurfsansicht mit der Codeansicht synchronisiert.

Rückgabewerte
Keine.

Beispiel

```
dom.source.refreshVariableCodeHints();
```

dom.source.removeComment()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion entfernt Kommentare. Wenn Sie keine Argumente angeben, werden aus der aktuellen Auswahl alle Arten von Kommentaren entfernt außer serverseitigen Includes und Dreamweaver-spezifischen Kommentaren. Bei verschachtelten Kommentaren wird nur der äußerste Kommentar entfernt. Wurde kein Text ausgewählt, wird nur der erste Zeilenkommentar der Zeile entfernt, in der sich der Cursor befindet. Wenn Sie Argumente angeben, entfernt die Funktion nur die Kommentare, die den Werten in den Argumenten *beforeText* und *afterText* entsprechen, diese allerdings auch dann, wenn sie in anderen Arten von Kommentaren verschachtelt sind.

Argumente
beforeText, afterText

- Das Argument *beforeText* ist optional. Es beschreibt den Text, der den Beginn des Kommentars kennzeichnet, der aus der Auswahl entfernt werden soll, bzw., wenn das Argument *afterText* den Wert null hat, die Art des Zeilenkommentars, der aus der aktuellen Auswahl entfernt werden soll.
- Das (optionale) Argument *afterText* beschreibt den Text, der das Ende des Kommentars kennzeichnet, der aus der Auswahl entfernt werden soll.

Rückgabewerte
Keine.

Beispiel
Im folgenden Beispiel wird ein HTML-Kommentar entfernt:

```
dw.getDocumentDOM().source.removeComment('<!--', '-->')
```

dreamweaver.htmlInspector.getShowHiddenCharacters()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion bestimmt, ob in der Codeansicht des Codeinspektors die Sonderzeichen für Leerräume angezeigt werden.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn die versteckten Zeichen angezeigt werden, andernfalls `false`.

Beispiel
Im folgenden Beispiel wird die Anzeige der Leerraum-Sonderzeichen im Codeinspektor deaktiviert, wenn die Anzeige ursprünglich aktiviert war:

```
if (dreamweaver.htmlinspector.getShowHiddenCharacters()){
    dreamweaver.htmlinspector.setShowHiddenCharacters(false);
    }
```

dreamweaver.htmlInspector.setShowHiddenCharacters()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion zeigt im Codeinspektor die Sonderzeichen für Leerräume an bzw. verbirgt sie.

Argumente
show

- Das obligatorische Argument *show* ist ein boolescher Wert, der angibt, ob die verborgenen Leerraumzeichen angezeigt werden sollen.

Rückgabewerte
Ein boolescher Wert: `true`, wenn die versteckten Zeichen angezeigt werden, andernfalls `false`.

Beispiel
Siehe „dreamweaver.htmlInspector.getShowHiddenCharacters()“ auf Seite 221.

Farbfunktionen

Mit den folgenden Farbfunktionen können Sie sicherstellen, dass Erweiterungen dieselbe Skin wie die Benutzeroberfläche der Anwendung verwenden.

dreamweaver.getPanelColor()

Verfügbarkeit
Dreamweaver CS4.

Beschreibung
Mit dieser Funktion werden die Bedienfeldfarben der Benutzeroberfläche der Anwendung abgerufen. Sie können diese Farben als Bedienfeldfarben der Erweiterungen verwenden. Mit dieser Funktion können Sie besser sicherstellen, dass die Bedienfeldfarben von Erweiterungen an die Bedienfeldfarben der Benutzeroberfläche der Anwendung angepasst sind.

Argumente
Keine.

Rückgabewerte
Ein Array von vier Strings mit den folgenden Werten:

- Rot
- Grün
- Blau
- Alphatransparenz

Beispiel
```
var panelColorArray = dw.getPanelColor();
```

Dies sind die Rückgabewerte für dieses Beispiel:

- `panelColorArray[0]`: Rot
- `panelColorArray[1]`: Grün
- `panelColorArray[2]`: Blau
- `panelColorArray[3]`: Alphatransparenz

dreamweaver.getAppBarColor()

Verfügbarkeit
Dreamweaver CS4.

Beschreibung
Mit dieser Funktion werden die Leistenfarben der Benutzeroberfläche der Anwendung abgerufen. Sie können diese Farben als Leistenfarben der Erweiterungen verwenden. Mit dieser Funktion können Sie besser sicherstellen, dass die Leistenfarben von Erweiterungen an die Leistenfarben der Benutzeroberfläche der Anwendung angepasst sind.

Argumente
Keine.

Rückgabewerte
Ein Array von vier Strings mit den folgenden Werten:

- Rot
- Grün
- Blau
- Alphatransparenz

Beispiel
```
var appBarColorArray = dw.getAppBarColor();
```

Dies sind die Rückgabewerte für dieses Beispiel:

- `appBarColorArray[0]`: Rot
- `appBarColorArray[1]`: Grün
- `appBarColorArray[2]`: Blau
- `appBarColorArray[3]`: Alphatransparenz

Kapitel 13: Site

Mit den Site-Funktionen von Adobe® Dreamweaver® CS5 werden Vorgänge im Zusammenhang mit der Verwaltung von Websites durchgeführt. Zu diesen Operationen gehören u. a. die Anpassung von Berichten, das Definieren einer neuen Site, das Ein- und Auschecken von Dateien sowie die Durchführung einer Site-Prüfung.

Berichtfunktionen

Die Berichtfunktionen ermöglichen den Zugriff auf die Berichtfunktionalität, sodass Sie die Berichtvorgänge einleiten, überwachen und anpassen können. Weitere Informationen finden Sie in *Erweitern von Dreamweaver* unter „Berichte“.

dreamweaver.isReporting()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Überprüft, ob derzeit ein Berichtvorgang ausgeführt wird.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn ein Vorgang ausgeführt wird, andernfalls `false`.

dreamweaver.showReportsDialog()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Öffnet das Dialogfeld „Berichte“.

Argumente
Keine.

Rückgabewerte
Keine.

Site-Funktionen

Site-Funktionen beziehen sich auf Dateien in der Ansicht „Site-Dateien“. Mit diesen Funktionen können Sie folgende Aufgaben ausführen:

- Erstellen von Hyperlinks zwischen Dateien
- Abrufen, Ablegen, Einchecken und Auschecken von Dateien
- Auswählen von Dateien und Aufheben der Auswahl
- Erstellen und Entfernen von Dateien
- Abrufen von Informationen über die vom Benutzer definierten Sites
- Importieren und Exportieren von Site-Informationen

dom.getSiteURLPrefixFromDoc()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion ruft das Site-URL-Präfix ab, das aus der HTTP-Adresse extrahiert worden ist. Die HTTP-Adresse wird im Dialogfeld „Site-Definition“ auf der Registerkarte „Erweitert“ in der Kategorie „Lokale Infos“ definiert.

Argumente
Keine.

Rückgabewerte
Ein String, der das Site-URL-Präfix angibt.

Beispiel
Im folgenden Beispiel wird das Site-URL-Präfix des aktuellen Dokuments abgerufen:

```
var currentDOM = dw.getDocumentDOM();
var sitePrefix = currentDOM.getSiteURLPrefixFromDoc();
```

dom.localPathToSiteRelative()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion konvertiert einen lokalen Dateipfad in eine site-relative URI-Referenz.

Argumente
localFilePath

- Das obligatorische Attribut *localFilePath* ist ein String, der den Pfad zu einer lokalen Datei auf Ihrem lokalen Computer beschreibt.

Rückgabewerte
Ein String, der den site-relativen URI angibt.

Beispiel
Im folgenden Beispiel wird `"/myWebApp/myFile.cfm"` zurückgegeben. Der Pfad basiert auf Ihren Site-Zuordnungen und der HTTP-Adresse, die im Dialogfeld „Site-Definition“ auf dem Register „Erweitert“ in der Kategorie „Lokale Infos“ angegeben wurde.

```
var dom = dw.getDocumentDOM();
    var siteRelativeURI =
dom.localPathToSiteRelative("C:\Inetpub\wwwroot\siteA\myWebApp\myFile.cfm")
```

dom.siteRelativeToLocalPath()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion konvertiert eine site-relative URI-Referenz in einen lokalen Dateipfad.

Argumente
siteRelativeURI

- Das obligatorische Attribut *siteRelativeURI* ist ein String, der den site-relativen URI enthält.

Rückgabewerte
Ein String, der den Pfad zu einer lokalen Datei auf Ihrem lokalen Computer angibt.

Beispiel
Im folgenden Beispiel

```
var filePath = siteRelativeToLocalPath("/myWebApp/myFile.xml");
```

wird `"C:\Inetpub\wwwroot\siteA\myFile.xml"` zurückgegeben. Der Pfad basiert auf Ihren Site-Zuordnungen und der HTTP-Adresse, die im Dialogfeld „Site-Definition“ auf der Registerkarte „Erweitert“ in der Kategorie „Lokale Infos“ angegeben wurde.

dreamweaver.compareFiles()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion startet die Anwendung für Dateivergleiche, die der Benutzer im Dialogfeld „Voreinstellungen“ in der Kategorie „Dateien vergleichen“ angegeben hat.

Argumente
file1, file2

- Das obligatorische Attribut *file1* ist ein String, der den vollständigen Pfad zur ersten zu vergleichenden Datei beschreibt.
- Das obligatorische Attribut `file2` ist ein String, der den vollständigen Pfad zur zweiten zu vergleichenden Datei beschreibt.

Rückgabewerte
Keine.

Beispiel
Im folgenden Beispiel werden die beiden Dateien „red.htm“ und „blue.htm“ verglichen:

```
dw.compareFiles(hc:\data\red.htm", "e:\data\blue.htm");
```

dreamweaver.loadSitesFromPrefs()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Lädt die Site-Informationen für alle Sites aus der Systemregistrierung (Windows) bzw. aus der Dreamweaver-Voreinstellungsdatei (Macintosh) in Dreamweaver. Wenn diese Funktion aufgerufen wird, während die Site mit einem Remote-Server verbunden ist, wird diese Verbindung automatisch unterbrochen.

Argumente
Keine.

Rückgabewerte
Keine.

dreamweaver.saveSitesToPrefs()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Speichert alle Informationen über die vom Benutzer definierten Sites in der Systemregistrierung (Windows) bzw. in der Dreamweaver-Voreinstellungsdatei (Macintosh).

Argumente
Keine.

Rückgabewerte
Keine.

dreamweaver.siteSyncDialog.compare()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion führt die Anwendung für Dateivergleiche aus, die im Dialogfeld „Voreinstellungen“ in der Kategorie „Dateien vergleichen“ angegeben ist, um die auf den lokalen und Remote-Sites ausgewählten Dateien zu vergleichen.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dreamweaver.siteSyncDialog.canCompare()“ auf Seite 532.

dreamweaver.siteSyncDialog.markDelete()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion ändert die Aktion für die ausgewählten Elemente im Dialogfeld „Synchronisieren“ in „Löschen“.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dreamweaver.siteSyncDialog.canMarkDelete()“ auf Seite 532.

dreamweaver.siteSyncDialog.markGet()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion ändert die Aktion für die ausgewählten Elemente im Dialogfeld „Synchronisieren“ in „Abrufen“.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dreamweaver.siteSyncDialog.canMarkGet()“ auf Seite 532.

dreamweaver.siteSyncDialog.markIgnore()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion ändert die Aktion für die ausgewählten Elemente im Dialogfeld „Synchronisieren“ in „Ignorieren“.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dreamweaver.siteSyncDialog.canMarkIgnore()“ auf Seite 533.

dreamweaver.siteSyncDialog.markPut()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion ändert die Aktion für die ausgewählten Elemente im Dialogfeld „Synchronisieren“ in „Bereitstellen“.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dreamweaver.siteSyncDialog.canMarkPut()“ auf Seite 533.

dreamweaver.siteSyncDialog.markSynced()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion ändert die Aktion für die ausgewählten Elemente im Dialogfeld „Synchronisieren“ in „Synchronisiert“.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dreamweaver.siteSyncDialog.canMarkSynced()“ auf Seite 533.

dreamweaver.siteSyncDialog.toggleShowAllFiles()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Mithilfe dieser Funktion können Sie sich im Vorschaufenster für die Site-Synchronisation ansehen, welche Dateien auf den Remote- und lokalen Sites von Dreamweaver als gleich ermittelt wurden. Wenn die Funktion aufgerufen wird, während das Kontrollkästchen „Alle Dateien zeigen“ aktiviert ist, wird es deaktiviert (und umgekehrt).

Argumente
Keine.

Rückgabewerte
Keine.

site.addLinkToExistingFile()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Öffnet das Dialogfeld „HTML-Datei auswählen“, in dem der Benutzer eine Datei auswählen kann. Danach wird ein Hyperlink vom ausgewählten Dokument zu dieser Datei erstellt.

Argumente
Keine.

Rückgabewerte
Keine.

site.changeLinkSitewide()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Öffnet das Dialogfeld „Hyperlink für ganze Site ändern“.

Argumente
Keine.

Rückgabewerte
Keine.

site.changeLink()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Öffnet das Dialogfeld „HTML-Datei auswählen“, in dem der Benutzer eine neue Datei für den Hyperlink auswählen kann.

Argumente
Keine.

Rückgabewerte
Keine.

site.checkIn()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Checkt die ausgewählten Dateien ein und verarbeitet abhängige Dateien auf eine der folgenden Weisen:

- Wenn der Benutzer im Dialogfeld „Voreinstellungen“ in der Kategorie „Site“ die Option „Eingabeaufforderung beim Bereitstellen/Einchecken“ aktiviert, wird das Dialogfeld „Abhängige Dateien“ angezeigt.
- Wenn der Benutzer zuvor im Dialogfeld „Abhängige Dateien“ die Option „Diese Meldung nicht mehr anzeigen“ aktiviert und dann auf „Ja“ geklickt hat, werden die abhängigen Dateien hochgeladen, ohne dass zuvor ein Dialogfeld angezeigt wird.
- Wenn der Benutzer zuvor im Dialogfeld „Abhängige Dateien“ die Option „Diese Meldung nicht mehr anzeigen“ aktiviert und dann auf „Nein“ geklickt hat, werden die abhängigen Dateien nicht hochgeladen und es wird auch kein Dialogfeld angezeigt.

Argumente
siteOrURL

- Das Argument *siteOrURL* muss das Schlüsselwort `"site"` sein, damit sich die Funktion auf die Auswahl im Bedienfeld „Dateien“ auswirkt, oder eine URL, wenn es sich um eine einzelne Datei handelt.

Rückgabewerte
Keine.

Enabler
Siehe „site.canCheckIn()“ auf Seite 535.

site.checkLinks()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Öffnet das Dialogfeld „Hyperlink-Prüfer“ und überprüft die Hyperlinks in den angegebenen Dateien.

Argumente
scopeOfCheck

- Das Argument *scopeOfCheck* gibt den Umfang der Hyperlink-Überprüfung an. Der Wert muss entweder `"document"`, `"selection"` oder `"site"` lauten.

Rückgabewerte
Keine.

site.checkOut()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Checkt die ausgewählten Dateien aus und verarbeitet abhängige Dateien auf eine der folgenden Weisen:

- Wenn der Benutzer im Dialogfeld „Voreinstellungen“ in der Kategorie „Site“ die Option „Eingabeaufforderung beim Abrufen/Auschecken“ aktiviert, wird das Dialogfeld „Abhängige Dateien“ angezeigt.
- Wenn der Benutzer zuvor im Dialogfeld „Abhängige Dateien“ die Option „Diese Meldung nicht mehr anzeigen“ aktiviert und dann auf „Ja“ geklickt hat, werden die abhängigen Dateien heruntergeladen, ohne dass zuvor ein Dialogfeld angezeigt wird.
- Wenn der Benutzer zuvor im Dialogfeld „Abhängige Dateien“ die Option „Diese Meldung nicht mehr anzeigen“ aktiviert und dann auf „Nein“ geklickt hat, werden die abhängigen Dateien nicht heruntergeladen und es wird auch kein Dialogfeld angezeigt.

Argumente
siteOrURL

- Das Argument *siteOrURL* muss das Schlüsselwort `"site"` sein, damit sich die Funktion auf die Auswahl im Bedienfeld „Dateien“ auswirkt, oder eine URL, wenn es sich um eine einzelne Datei handelt.

Rückgabewerte
Keine.

Enabler
Siehe „site.canCheckOut()“ auf Seite 535.

site.checkTargetBrowsers()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Führt in den ausgewählten Dateien eine Zielbrowser-Überprüfung durch.

Argumente
Keine.

Rückgabewerte
Keine.

site.cloak()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Schließt die aktuelle Auswahl im Bedienfeld „Dateien“ oder den angegebenen Ordner mit dem Cloaking aus.

Argumente
siteOrURL

Das Argument *siteOrURL* muss einen der beiden folgenden Werte enthalten:

- Das Schlüsselwort `"site"`, das angibt, dass `cloak()` sich auf die Auswahl im Bedienfeld „Dateien“ auswirken soll.
- Die URL eines bestimmten Ordners, die angibt, dass `cloak()` sich auf den angegebenen Ordner und dessen Inhalt auswirken soll.

Rückgabewerte
Keine.

Enabler
Siehe „site.canCloak()“ auf Seite 536.

site.compareFiles()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion startet die Anwendung für den Dateivergleich, die zwei Dateien miteinander vergleicht.

Argumente
url

Das obligatorische Argument *url* muss einen der beiden folgenden Werte enthalten:

- Das Schlüsselwort `"site"`, das angibt, dass `compare()` sich auf die Auswahl im Bedienfeld „Dateien“ auswirken soll.
- Die URL einer lokalen Datei, die mit ihrer Remote-Version verglichen werden soll.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der Vergleich erfolgreich durchgeführt wird, andernfalls `false`.

Enabler
Siehe „site.canCompareFiles()“ auf Seite 536.

Beispiel
Im folgenden Beispiel werden die im Bedienfeld „Dateien“ ausgewählten Dateien mit ihren Remote-Versionen verglichen:

```
site.compareFiles("site");
```

site.defineSites()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion öffnet das Dialogfeld „Site-Definition“.

Argumente
Keine.

Rückgabewerte
Keine.

site.deleteSelection()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Löscht die ausgewählten Dateien.

Argumente
Keine.

Rückgabewerte
Keine.

site.deployFilesToTestingServerBin()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Legt eine bestimmte Datei (oder Dateien) im Ordner „bin“ des Testservers ab. Wenn für die aktuelle Site keine Einstellungen für die Bereitstellung unterstützender Dateien definiert sind, ruft diese Funktion das Dialogfeld „Unterstützende Dateien auf dem Testserver bereitstellen“ auf.

Argumente
filesToDeploy

- Das Argument *filesToDeploy* ist ein Array von Dateinamen, die Dreamweaver bereitstellt.

Rückgabewerte
Ein boolescher Wert: `true`, wenn die Dateien erfolgreich bereitgestellt wurden, andernfalls `false`.

Beispiel
In diesem Beispiel werden die Dateien „image1.jpg“ und „script1.js“ im Ordner „bin“ des Testservers bereitgestellt:

```
site.deployFilesToTestingServerBin("image1.jpg", "script1.js");
```

site.displaySyncInfoForFile()

Verfügbarkeit
Dreamweaver CS3.

Beschreibung
Zeigt ein Dialogfeld an, das die lokale Zeit, die Remote-Zeit und die Testzeit der Datei entsprechend dem übergebenen Parameter enthält. Diese Informationen werden in der Synchronisierungsdatei dwsync.xml gespeichert.

Das Dialogfeld zeigt vier Zeiten an:

- Die lokale Remote-Zeit, wobei es sich um den Zeitstempel der lokalen Datei für den letzten Abruf- oder Bereitstellungsvorgang auf dem Remote-Server handelt.
- Die Remote-Zeit, wobei es sich um den Zeitstempel der Datei auf dem Remote-Server für den letzten Abruf- oder Bereitstellungsvorgang auf dem Remote-Server handelt.
- Die lokale Testzeit, wobei es sich um den Zeitstempel der lokalen Datei für den letzten Abruf- oder Bereitstellungsvorgang auf dem Testserver handelt.
- Die Testzeit, wobei es sich um den Zeitstempel der Datei auf dem Testserver für den letzten Abruf- oder Bereitstellungsvorgang auf dem Testserver handelt.

Falls die Datei dwsync.xml keine Informationen zu einer Datei enthält, wird eine entsprechende Meldung angezeigt. Wenn die Zeit in der XML-Datei festgelegt wurde, wird sie im Datum/Uhrzeit-Format des jeweiligen Gebietsschemas angezeigt (z. B. 24.6.05 14:43 Uhr). Wenn die Zeit im Eintrag für die Datei nicht festgelegt wurde, wird ein Bindestrich (-) angezeigt.

Diese Funktion kann für die in der Ansicht „Lokale Dateien“ ausgewählte Datei verwendet werden, falls '`site`' übergeben wird, oder für die Datei, die der lokalen URL entspricht, falls eine URL übergeben wird.

Argumente
path, '`site`'

- *path* ist die URL zu einer lokalen Datei.
- '`site`' gibt an, dass die Funktion die im Bedienfeld „Dateien“ ausgewählte Datei verwendet.

Rückgabewerte
Keine.

Enabler
Siehe „site.canDisplaySyncInfoForFile()“ auf Seite 537.

site.editColumns()

Beschreibung
Diese Funktion zeigt das Dialogfeld „Site-Definition“ mit dem Bereich „Dateiansichtsspalten“ an.

Argumente
Keine.

Rückgabewerte
Keine.

site.exportSite()

Verfügbarkeit
Dreamweaver MX; aktualisiert in Dreamweaver CS4.

Beschreibung

Exportiert eine Dreamweaver-Site in eine XML-Datei, die dann in eine andere Dreamweaver-Instanz importiert werden kann, um die Site zu duplizieren.

Alle Angaben im Dialogfeld „Site-Definition“ werden in einer XML-Datei gespeichert. Dazu gehören die Liste der Ordner, für die Cloaking aktiviert ist, sowie die Details zum Standarddokumenttyp. Wenn der FTP-Zugriff festgelegt wurde, können der Anmeldename und das Kennwort des Benutzers jedoch weggelassen werden.

Argumente

siteName, {*askAboutLoginInfo*}, {*warnAboutSCS*}, {*savePath*}

- Das Argument *siteName* identifiziert die zu exportierende Site. Wenn *siteName* ein leerer String ist, exportiert Dreamweaver die aktuelle Site.
- Das Argument *askAboutLoginInfo* gibt an, ob ein Dialogfeld mit der Frage, ob der Benutzer seine Anmeldeinformationen speichern möchte, eingeblendet wird. Dieses Argument ist optional.
- Mit dem Argument *warnAboutSCS* können Sie angeben, ob beim Zugriff auf eine Site über die Quellcodeverwaltung der Warnhinweis eingeblendet wird, dass die Anmeldeinformationen nicht gespeichert werden. Dieses Argument ist optional.
- Das Argument *savePath* ist der lokale Pfad zu einem Ordner (z. B. `C:\sites\mySites\`). Wenn Sie *savePath* angeben, wird die Datei.ste immer unter dem Namen der Site gespeichert. Dieses Argument ist optional.

Rückgabewerte

Ein boolescher Wert: `true`, wenn die angegebene Site vorhanden ist und die XML-Datei erfolgreich exportiert wurde, andernfalls `false`.

Beispiel

Im folgenden Beispiel wird eine XML-Musterdatei dargestellt, die Dreamweaver erstellt, wenn Sie eine Site exportieren:

```
<?xml version="1.0" ?>
<site>
    <localinfo
        sitename="DW00"
        localroot="C:\Documents and Settings\jlondon\Desktop\DWServer\"
        imagefolder="C:\Documents and Settings\jlondon\Desktop\DWServer\Images\"
        spacerfilepath=""
        refreshlocal="TRUE"
        cache="FALSE"
        httpaddress="http://" curserver="webserver" />
    <remoteinfo
        accesstype="ftp"
        host="dreamweaver"
        remoteroot="kojak/"
        user="dream"
        checkoutname="Jay"
        emailaddress="jay@Adobe.com"
        usefirewall="FALSE"
        usepasv="TRUE"
        enablecheckin="TRUE"
        checkoutwhenopen="TRUE" />
    <designnotes
        usedesignnotes="TRUE"
```

```
        sharedesignnotes="TRUE" />
    <sitemap
        homepage="C:\Documents and Settings\jlondon\Desktop\DWServer\Untitled-2.htm"
        pagesperrow="200" columnwidth="125" showdependentfiles="TRUE"
        showpagetitles="FALSE" showhiddenfiles="TRUE" />
    <fileviewcolumns sharecolumns="TRUE">
        <column name="Local Folder"
            align="left" show="TRUE" share="FALSE" builtin="TRUE"
            localwidth="180" remotewidth="180" />
        <column name="Notes"
            align="center" show="TRUE" share="FALSE" builtin="TRUE"
            localwidth="36" remotewidth="36" />
        <column name="Size"
            align="right" show="TRUE" share="FALSE" builtin="TRUE"
            localwidth="-2" remotewidth="-2" />
        <column name="Type"
            align="left" show="TRUE" share="FALSE" builtin="TRUE"
            localwidth="60" remotewidth="60" />
        <column name="Modified"
            align="left" show="TRUE" share="FALSE" builtin="TRUE"
            localwidth="102" remotewidth="102" />
        <column name="Checked Out By"
            align="left" show="TRUE" share="FALSE" builtin="TRUE"
            localwidth="50" remotewidth="50" />
        <column name="Status" note="status"
            align="left" show="TRUE" share="FALSE" builtin="FALSE"
            localwidth="50" remotewidth="50" />
    </fileviewcolumns>
    <appserverinfo
        servermodel="ColdFusion"
        urlprefix="http://dreamweaver/kojak/"
        serverscripting="CFML"
        serverpageext=""
        connectionsmigrated="TRUE"
        useUD4andUD5pages="TRUE"
        defaultdoctype=""
        accesstype="ftp"
        host="dreamweaver"
        remoteroot="kojak/"
        user="dream"
        usefirewall="FALSE"
        usepasv="TRUE" />
    <cloaking enabled="TRUE" patterns="TRUE">
        <cloakedfolder folder="databases/" />
        <cloakedpattern pattern=".png" />
        <cloakedpattern pattern=".jpg" />
        <cloakedpattern pattern=".jpeg" />
    </cloaking>
</site>
```

site.get()

Verfügbarkeit

Dreamweaver 3.

Beschreibung
Ruft die angegebenen Dateien ab und verarbeitet abhängige Dateien auf eine der folgenden Weisen:

- Wenn der Benutzer im Dialogfeld „Voreinstellungen“ in der Kategorie „Site“ die Option „Eingabeaufforderung beim Abrufen/Auschecken“ aktiviert, wird das Dialogfeld „Abhängige Dateien“ angezeigt.
- Wenn der Benutzer zuvor im Dialogfeld „Abhängige Dateien“ die Option „Diese Meldung nicht mehr anzeigen“ aktiviert und dann auf „Ja“ geklickt hat, werden die abhängigen Dateien heruntergeladen, ohne dass zuvor ein Dialogfeld angezeigt wird.
- Wenn der Benutzer zuvor im Dialogfeld „Abhängige Dateien“ die Option „Diese Meldung nicht mehr anzeigen“ aktiviert und dann auf „Nein“ geklickt hat, werden die abhängigen Dateien nicht heruntergeladen und es wird auch kein Dialogfeld angezeigt.

Argumente
siteOrURL

- Das Argument *siteOrURL* muss das Schlüsselwort `"site"` sein, damit sich die Funktion auf die Auswahl im Bedienfeld „Dateien“ auswirkt, oder eine URL, wenn es sich um eine einzelne Datei handelt.

Rückgabewerte
Keine.

Enabler
Siehe „site.canGet()“ auf Seite 537.

site.getAppServerAccessType()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Gibt die Zugriffsmethode zurück, die für alle Dateien auf dem Anwendungsserver der aktuellen Site verwendet wird. Die aktuelle Site ist die Site, zu der das derzeit aktive Dokument gehört. Wenn kein Dokument aktiv ist, wird die Site verwendet, die Sie in Dreamweaver geöffnet haben.

Hinweis: *ColdFusion Component Explorer verwendet diese Funktion. Siehe „site.getAppServerPathToFiles()“ auf Seite 241 und „site.getLocalPathToFiles()“ auf Seite 244.*

Argumente
Keine.

Rückgabewerte
Einer der folgenden Strings:

- `"none"`
- `"local/network"`
- `"ftp"`
- `"source_control"`

site.getAppServerPathToFiles()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Bestimmt den Pfad der Remote-Dateien auf dem Anwendungsserver, der für die aktuelle Site definiert ist. Die aktuelle Site ist die Site, zu der das derzeit aktive Dokument gehört. Wenn kein Dokument aktiv ist, wird die Site verwendet, die Sie in Dreamweaver geöffnet haben.

Hinweis: *ColdFusion Component Explorer verwendet diese Funktion. Siehe „site.getAppServerAccessType()“ auf Seite 240 und „site.getLocalPathToFiles()“ auf Seite 244.*

Argumente
Keine.

Rückgabewerte
Wenn für die Dateien auf dem Anwendungsserver der Zugriffstyp `local/network` gilt, gibt diese Funktion einen Pfad zurück. Andernfalls gibt sie einen leeren String zurück.

site.getAppURLPrefixForSite()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Diese Funktion ruft den Wert des URL-Präfixes ab, das aus der HTTP-Adresse extrahiert worden ist. Die HTTP-Adresse wird im Dialogfeld „Site-Definition“ auf der Registerkarte „Erweitert“ in der Kategorie „Lokale Infos“ definiert. Dieser Wert entspricht dem Pfad hinter http://*hostname*:*portnumber*/.

Argumente
{*`siteName`*}

Das optionale Argument *siteName* ist der Name der Site, deren URL-Präfix Sie abrufen möchten. Wenn Sie keine Site angeben, ruft die Funktion das URL-Präfix der aktuellen Site ab.

Rückgabewerte
Ein String, der das URL-Präfix der derzeit ausgewählten Site enthält.

Beispiel
```
var sitePrefix = site.getAppURLPrefixForSite();
```

site.getCheckOutUser()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Ruft den Anmelde- und Auschecknamen ab, der zur aktuellen Site gehört.

Argumente
Keine.

Rückgabewerte
Ein String mit einem Anmelde- und Auschecknamen, falls definiert, oder ein leerer String, wenn das Ein-/Auschecken deaktiviert ist.

Beispiel
Beim Aufruf von `site.getCheckOutUser()` kann beispielsweise `"denise (deniseNotebook)"` zurückgegeben werden. Wenn kein Auscheckname festgelegt ist, wird nur der Anmeldename zurückgegeben (beispielsweise `"denise"`).

site.getCheckOutUserForFile()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Ruft den Anmelde- und Auschecknamen des Benutzers ab, der die angegebene Datei ausgecheckt hat.

Argumente
fileName

- Das Argument *fileName* ist der Pfad der abgefragten Datei im URL-Format „file://“.

Rückgabewerte
Ein String mit dem Anmelde- und Auschecknamen des Benutzers, der die Datei ausgecheckt hat, oder ein leerer String, wenn die Datei nicht ausgecheckt ist.

Beispiel
Beim Aufruf von `site.getCheckOutUserForFile("file://C:/sites/avocado8/index.html")` könnte beispielsweise `"denise (deniseLaptop)"` zurückgegeben werden. Wenn kein Auscheckname festgelegt ist, wird nur der Anmeldename zurückgegeben (beispielsweise `"denise"`).

site.getCloakingEnabled()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Bestimmt, ob das Cloaking für die aktuelle Site aktiviert ist.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert:`true`, wenn Cloaking für die aktuelle Site aktiviert ist, andernfalls `false`.

site.getConnectionState()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Ruft den aktuellen Verbindungsstatus ab.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert, der angibt, ob eine Verbindung zur Remote-Site besteht.

Enabler
Siehe „site.canConnect()“ auf Seite 537.

site.getCurrentSite()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Ruft die aktuelle Site ab.

Argumente
Keine.

Rückgabewerte
Ein String, der den Namen der aktuellen Site enthält.

Beispiel
Wenn Sie mehrere Sites definiert haben, wird beim Aufruf von `site.getCurrentSite()` die Site zurückgegeben, die im Bedienfeld „Dateien“ in der Liste der aktuellen Sites angezeigt wird.

site.getFocus()

Verfügbarkeit
Dreamweaver 3, aktualisiert in CS4.

Beschreibung
Ermittelt, welcher Bereich des Bedienfelds „Dateien“ sich gerade im Fokus befindet.

Argumente
Keine.

Rückgabewerte
Einer der folgenden Strings: „local“ oder „remote“.

site.getLocalPathToFiles()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Bestimmt den Pfad der lokalen Dateien, die für die aktuelle Site definiert sind. Die aktuelle Site ist die Site, zu der das derzeit aktive Dokument gehört. Wenn kein Dokument aktiv ist, wird die Site verwendet, die Sie in Dreamweaver geöffnet haben.

Hinweis: *ColdFusion Component Explorer verwendet diese Funktion. Siehe „site.getAppServerAccessType()“ auf Seite 240 und „site.getAppServerPathToFiles()“ auf Seite 241.*

Argumente
Keine.

Rückgabewerte
Der Pfad der Dateien, die sich auf dem lokalen Computer für die aktuelle Site befinden.

site.getLocalRootURL()

Verfügbarkeit
Dreamweaver CS4.

Beschreibung
Ruft den lokalen Stammordner der Site ab.

Argumente
`siteName`

- Das Argument `siteName` ist ein String, der den Namen der Site angibt.

Rückgabewerte
Ein String, der den lokalen Stammordner der angegebenen Site im URL-Format file:// enthält. Der String ist leer, wenn die angegebene Site nicht existiert.

site.getSelection()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Ermittelt, welche Dateien derzeit im Bedienfeld „Dateien“ ausgewählt sind.

Argumente
Keine.

Rückgabewerte
Ein Array von Strings mit den Pfaden der ausgewählten Dateien und Ordner (im URL-Format „file://“) bzw. ein leeres Array, wenn keine Dateien oder Ordner ausgewählt sind.

site.getSiteForURL()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Ruft den Namen der Site ab (falls vorhanden), zu der eine bestimmte Datei gehört.

Argumente
fileURL

- Das Argument *fileURL* ist die vollständige URL einer benannten Datei (einschließlich des Strings "*file:///*").

Rückgabewerte
Ein String mit dem Namen der Site (falls vorhanden), in der die angegebene Datei sich befindet. Wenn die angegebene Datei sich nicht in einer definierten Site befindet, ist der String leer.

site.getSites()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Ruft eine Liste der definierten Sites ab.

Argumente
Keine.

Rückgabewerte
Ein Array von Strings mit den Namen der definierten Sites bzw. ein leeres Array, wenn keine Site definiert ist.

site.getSiteRootForURL()

Verfügbarkeit
Dreamweaver CS4.

Beschreibung
Ruft den lokalen Stammordner der Site ab, zu der eine angegebene Datei gehört.

Argumente
`fileURL`

- Das Argument *`fileURL`* ist ein String, der die vollständige URL einer benannten Datei enthält (einschließlich des Strings `"file://"`).

Rückgabewerte
Ein String, der den lokalen Stammordner der Site enthält (im URL-Format `file://`), in der sich die angegebene Datei befindet. Wenn die angegebene Datei sich nicht in einer definierten Site befindet, ist der String leer.

Beispiel
```
var dom = dw.getDocumentDOM();
var siteRoot = site.getSiteRootForURL(dom.URL);
```

site.getSiteURLPrefix()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Ruft das Site-URL-Präfix ab, das aus der HTTP-Adresse extrahiert worden ist. Die HTTP-Adresse wird im Dialogfeld „Site-Definition“ auf der Registerkarte „Erweitert“ in der Kategorie „Lokale Infos“ definiert.

Argumente
Keine.

Rückgabewerte
Ein String, der das Site-URL-Präfix enthält.

Beispiel
```
sitePrefix = getSiteURLPrefix();
```

site.importSite()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Erstellt eine Dreamweaver-Site auf Grundlage einer XML-Datei. Dreamweaver verwendet das Attribut`localroot` des Elements `<localinfo>`, um den lokalen Stammordner der Site zu identifizieren. Falls dieser Ordner nicht auf dem lokalen Computer vorhanden ist, wird der Benutzer während des Importvorgangs zur Angabe eines anderen lokalen Stammordners aufgefordert. Das Gleiche geschieht, wenn Dreamweaver versucht, den Standard-Bilderordner zu finden, den das Attribut `imagefolder` des Elements `<localinfo>` vorgibt.

Argumente
`pathToSteFile`

- Das Argument *`pathToSteFile`* ist ein String, der die URL für die STE-Datei enthält. Dreamweaver verwendet diese Datei, um eine Site zu erstellen. Wenn *`pathToSteFile`* ein leerer String ist, fordert Dreamweaver den Benutzer auf, eine STE-Datei für den Import auszuwählen.

Rückgabewerte
Ein boolescher Wert: `true`, wenn die benannte STE-Datei vorhanden ist und die Site erfolgreich erstellt wird, andernfalls `false`.

site.isCloaked()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Ermittelt, ob das Cloaking für die aktuelle Auswahl im Bedienfeld „Dateien“ oder für den angegebenen Ordner gilt.

Argumente
siteOrURL

- Das Argument *siteOrURL* muss einen der beiden folgenden Werte enthalten:
  - Das Schlüsselwort `"site"`, das angibt, dass `isCloaked()` die Auswahl im Bedienfeld „Dateien“ testen soll.
  - Die URL eines bestimmten Ordners, die angibt, dass `isCloaked()` den angegebenen Ordner testen soll.

Rückgabewerte
Ein boolescher Wert: `true`, wenn das Cloaking für das angegebene Objekt aktiviert ist, andernfalls `false`.

site.locateInSite()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Sucht die angegebenen Dateien im festgelegten Bereich des Bedienfelds „Dateien“ und wählt die Dateien aus.

Argumente
localOrRemote, siteOrURL

- Das Argument *localOrRemote* muss entweder `"local"` oder `"remote"` sein.
- Das Argument *siteOrURL* muss das Schlüsselwort `"site"` sein, damit sich die Funktion auf die Auswahl im Bedienfeld „Dateien“ auswirkt, oder eine URL, wenn es sich um eine einzelne Datei handelt.

Rückgabewerte
Keine.

Enabler
Siehe „site.canLocateInSite()“ auf Seite 538.

site.makeEditable()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Hebt den Schreibschutz bei den ausgewählten Dateien auf.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „site.canMakeEditable()“ auf Seite 538.

site.makeNewDreamweaverFile()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Erstellt im Bedienfeld „Dateien“ eine neue Dreamweaver-Datei in dem Ordner, in dem sich auch die erste ausgewählte Datei bzw. der erste ausgewählte Ordner befindet.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „site.canMakeNewFileOrFolder()“ auf Seite 539.

site.makeNewFolder()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Erstellt im Bedienfeld „Dateien“ einen neuen Ordner in dem Ordner, in dem sich auch die erste ausgewählte Datei bzw. der erste ausgewählte Ordner befindet.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „site.canMakeNewFileOrFolder()“ auf Seite 539.

site.newSite()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Öffnet das Dialogfeld „Site-Definition“ für eine neue, unbenannte Site.

Argumente
Keine.

Rückgabewerte
Keine.

site.open()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Öffnet die Dateien, die derzeit im Bedienfeld „Dateien“ ausgewählt sind. Falls Ordner ausgewählt sind, werden sie in der Ansicht „Site-Dateien“ erweitert dargestellt.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „site.canOpen()“ auf Seite 539.

site.put()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Stellt die ausgewählten Dateien bereit und verarbeitet abhängige Dateien auf eine der folgenden Weisen:

- Wenn der Benutzer im Dialogfeld „Voreinstellungen“ in der Kategorie „Site“ die Option „Eingabeaufforderung beim Bereitstellen/Einchecken“ aktiviert, wird das Dialogfeld „Abhängige Dateien“ angezeigt.
- Wenn der Benutzer zuvor im Dialogfeld „Abhängige Dateien“ die Option „Diese Meldung nicht mehr anzeigen“ aktiviert und dann auf „Ja“ geklickt hat, werden die abhängigen Dateien hochgeladen, ohne dass zuvor ein Dialogfeld angezeigt wird.
- Wenn der Benutzer zuvor im Dialogfeld „Abhängige Dateien“ die Option „Diese Meldung nicht mehr anzeigen“ aktiviert und dann auf „Nein“ geklickt hat, werden die abhängigen Dateien nicht hochgeladen und es wird auch kein Dialogfeld angezeigt.

Argumente
siteOrURL

- Das Argument *siteOrURL* muss das Schlüsselwort `"site"` sein, damit sich die Funktion auf die Auswahl im Bedienfeld „Dateien“ auswirkt, oder eine URL, wenn es sich um eine einzelne Datei handelt.

Rückgabewerte
Keine.

Enabler
Siehe „site.canPut()“ auf Seite 539.

site.recreateCache()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Erstellt den Cache für die aktuelle Site neu.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „site.canRecreateCache()“ auf Seite 540.

site.refresh()

Verfügbarkeit
Dreamweaver 3, aktualisiert in CS4.

Beschreibung
Aktualisiert die Dateiliste im angegebenen Bereich des Bedienfelds „Dateien“.

Argumente
`whichSide`

- Das Argument *`whichSide`* muss `local` oder `remote` sein.

Rückgabewerte
Keine.

Enabler
Siehe „site.canRefresh()“ auf Seite 540.

site.remoteIsValid()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Bestimmt, ob die Remote-Site gültig ist.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert, der angibt, ob eine Remote-Site definiert ist. Falls es sich um den Servertyp „Lokal/Netzwerk“ handelt, gibt dieser Wert auch an, ob das Laufwerk bereitgestellt ist.

site.renameSelection()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Stellt den Namen der ausgewählten Datei in einem Textfeld dar, damit die Datei umbenannt werden kann. Wenn mehr als eine Datei ausgewählt ist, wirkt sich die Funktion auf die zuletzt ausgewählte Datei aus.

Argumente
Keine.

Rückgabewerte
Keine.

site.selectAll()

Verfügbarkeit
Dreamweaver 3, aktualisiert in CS4.

Beschreibung
Wählt alle Dateien in der aktiven Ansicht aus.

Argumente
Keine.

Rückgabewerte
Keine.

site.selectNewer()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Wählt alle Dateien aus, die im angegebenen Bereich des Bedienfelds „Dateien“ neuer sind.

Argumente
whichSide

- Das Argument *whichSide* muss entweder `"local"` oder `"remote"` sein.

Rückgabewerte
Keine.

Enabler
Siehe „site.canSelectNewer()“ auf Seite 541.

site.serverActivity()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion bestimmt, ob Dreamweaver momentan mit einem Server interagiert. Da Dreamweaver jeweils nur eine Serveraktivität ausführen kann, können Sie mithilfe dieser Funktion bestimmen, ob Funktionalitäten deaktiviert werden, die eine Server-Interaktion erfordern.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert, der angibt, ob Dreamweaver gerade mit einem Server interagiert.

Beispiel
Im folgenden Beispiel aus der Datei „menus.xml“ wird ein Menüelement angezeigt, wenn keine Serveraktivität vorliegt (und in Dreamweaver eine aktuelle Site angegeben ist):

```
<menuitem name="Remove Connection Scripts" enabled="!site.serverActivity() &&
site.getCurrentSite() != ''" command="alert(MMDB.removeConnectionScripts())"
id="SiteOptionsSiteMenu_RemoveConnectionScripts" />
```

site.setCloakingEnabled()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Bestimmt, ob das Cloaking für die aktuelle Site aktiviert sein soll.

Argumente
enable

- Das Argument *enable* ist ein boolescher Wert, der angibt, ob das Cloaking aktiviert sein soll. Der Wert `true` aktiviert das Cloaking für die aktuelle Site, der Wert `false` deaktiviert das Cloaking für die aktuelle Site.

Rückgabewerte
Keine.

site.setConnectionState()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Legt den Verbindungsstatus der aktuellen Site fest.

Argumente
bConnected

- Das Argument *bConnected* ist ein boolescher Wert, der angibt, ob eine Verbindung zur aktuellen Site besteht (`true`) oder nicht (`false`).

Rückgabewerte
Keine.

site.setCurrentSite()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Öffnet die angegebene Site im lokalen Bereich des Bedienfelds „Dateien“.

Argumente
whichSite

- Das Argument *whichSite* ist der Name einer definierten Site (wie in der Liste der aktuellen Sites im Bedienfeld „Dateien“ bzw. im Dialogfeld „Site-Definition“ angezeigt).

Rückgabewerte
Keine.

Beispiel
Wenn drei Sites definiert sind (z. B. „avocado8“, „dreamcentral“ und „testsite“), wird beim Aufruf von `site.setCurrentSite("dreamcentral");` „dreamcentral“ zur aktuellen Site.

site.setFocus()

Verfügbarkeit
Dreamweaver 3, aktualisiert in CS4.

Beschreibung
Übergibt den Fokus an einen angegebenen Bereich im Bedienfeld „Dateien“. Wenn der betreffende Bereich nicht sichtbar war, wird er durch diese Funktion angezeigt und erhält den Fokus.

Argumente
whichPane, *nextTextView*

- Das Argument *whichPane* muss einer der folgenden Strings sein: `local` oder `remote`.
- Das Argument *nextTextView* schaltet in einer geteilten Ansicht den Fokus zwischen den Ansichten hin und her.

Rückgabewerte
Keine.

site.setSelection()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Wählt Dateien oder Ordner im aktiven Bereich des Bedienfelds „Dateien“ aus.

Argumente
arrayOfURLs

- Das Argument *arrayOfURLs* ist ein Array mit Strings, die jeweils den Pfad einer Datei oder eines Ordners in der aktuellen Site im URL-Format „file://“ angeben.

  Hinweis: *Verwenden Sie bei der Angabe von Ordnerpfaden am Ende keinen Schrägstrich (/).*

Rückgabewerte
Keine.

site.siteRelativeToLocalPath()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion konvertiert eine site-relative URI-Referenz in einen lokalen Dateipfad.

Argumente
siteRelativeURI

- Das obligatorische Attribut `siteRelativeURI` ist ein String, der den site-relativen URI enthält.

Rückgabewerte
Ein String, der den Pfad zu einer lokalen Datei auf Ihrem lokalen Computer angibt.

Beispiel
Im folgenden Beispiel

```
var filePath = site.siteRelativeToLocalPath("/myWebApp/myFile.xlml");
```

wird "C:\Inetpub\wwwroot\siteA\myFile.xml" zurückgegeben. Der Pfad basiert auf Ihren Site-Zuordnungen und der HTTP-Adresse, die im Dialogfeld „Site-Definition“ auf der Registerkarte „Erweitert“ in der Kategorie „Lokale Infos“ angegeben wurde.

site.synchronize()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Öffnet das Dialogfeld „Dateien synchronisieren“.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „site.canSynchronize()“ auf Seite 541.

site.uncloak()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Beendet das Ausschließen der aktuellen Auswahl im Bedienfeld „Dateien“ oder im angegebenen Ordner mit dem Cloaking.

Argumente
siteOrURL

- Das Argument *siteOrURL* muss einen der folgenden Werte enthalten:
  - Das Schlüsselwort `"site"`, das angibt, dass `unCloak()` sich auf die Auswahl im Bedienfeld „Dateien“ auswirken soll.
  - Die URL eines bestimmten Ordners, die angibt, dass `unCloak()` sich auf den angegebenen Ordner und dessen Inhalt auswirken soll.

Rückgabewerte
Keine.

Enabler
Siehe „site.canUncloak()“ auf Seite 541.

site.uncloakAll()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Hebt das Cloaking sämtlicher Ordner in der aktuellen Site auf und deaktiviert im Dialogfeld „Site-Definition“ auf der Registerkarte „Erweitert“ in der Kategorie „Cloaking“ das Kontrollkästchen „Cloaking von Dateien mit Erweiterung“.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „site.canUncloak()“ auf Seite 541.

site.undoCheckOut()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Entfernt die zu den angegebenen Dateien gehörenden Sperrdateien aus den lokalen und Remote-Sites und ersetzt die lokalen Kopien der angegebenen Dateien durch die Remote-Kopien.

Argumente
siteOrURL

- Das Argument *siteOrURL* muss das Schlüsselwort `"site"` sein, damit sich die Funktion auf die Auswahl im Bedienfeld „Dateien“ auswirkt, oder eine URL, wenn es sich um eine einzelne Datei handelt.

Rückgabewerte
Keine.

Enabler
Siehe „site.canUndoCheckOut()“ auf Seite 542.

Kapitel 14: Dokument

Mit den Dokumentfunktionen in Adobe® Dreamweaver® werden Vorgänge durchgeführt, die sich auf das Dokument auswirken, das der Benutzer bearbeitet. Mit den Dokumentfunktionen können Sie die folgenden Aktionen ausführen:

- Konvertieren von Tabellen in Ebenen
- Ausführen eines Befehls im Ordner „Configuration/Commands“
- Suchen nach Datei-URLs
- Konvertieren relativer URLs in absolute URLs
- Abrufen des aktuellen ausgewählten Knotens
- Durchführen der URL-Kodierung für einen String
- Ausführen eines Übersetzers für ein Dokument

Konvertierungsfunktionen

Konvertierungsfunktionen dienen dazu, Tabellen in Ebenen, Ebenen in Tabellen und CSS (Cascading Stylesheets) in HTML-Markup umzuwandeln. Diese Funktionen haben jeweils den gleichen Effekt wie die entsprechenden Konvertierungsbefehle im Menü „Datei“ bzw. „Modifizieren“.

dom.convertLayersToTable()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Öffnet das Dialogfeld „Ebenen in Tabelle konvertieren“.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dom.canConvertLayersToTable()“ auf Seite 508.

dom.convertTablesToLayers()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Öffnet das Dialogfeld „Tabellen in Ebenen konvertieren“.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dom.canConvertTablesToLayers()“ auf Seite 508.

Befehlsfunktionen

Mithilfe der Befehlsfunktionen können Sie die Dateien im Ordner „Configuration/Commands“ optimal nutzen. Sie dienen zur Verwaltung des Menüs „Befehle“ und ermöglichen den Aufruf von Befehlen in anderen Erweiterungsdateien.

dreamweaver.editCommandList()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Öffnet das Dialogfeld „Befehlsliste bearbeiten“.

Argumente
Keine.

Rückgabewerte
Keine.

dreamweaver.runCommand()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Führt den angegebenen Befehl aus. Dies entspricht der Auswahl des Befehls in einem Menü. Ist ein Dialogfeld mit dem Befehl verknüpft, wird es angezeigt. Das Befehlsskript verhindert andere Bearbeitungen, bis der Benutzer das Dialogfeld schließt. Diese Funktion bietet die Möglichkeit, einen Befehl aus einer anderen Erweiterungsdatei aufzurufen.

Hinweis: *Diese Funktion kann in der* `objectTag()` *-Funktion, von einem Skript in einer Befehlsdatei oder von der Eigenschafteninspektor-Datei aufgerufen werden.*

Argumente
commandFile, {commandArg1}, {commandArg2},...{commandArgN}

- Das Argument *commandFile* ist ein Dateiname im Ordner „Configuration/Commands".
- Die anderen (optionalen) Argumente wie *commandArg1, commandArg2* usw. werden an die Funktion `receiveArguments()` im Argument *commandFile* übergeben.

Rückgabewerte
Keine.

Beispiel
Sie können einen benutzerdefinierten Eigenschafteninspektor für Tabellen schreiben, in dem der Benutzer über eine Schaltfläche den Befehl „Format table" wählen kann. Dazu wird von der Ereignisprozedur `onClick` der Schaltfläche die folgende Funktion aufgerufen:

```
function callFormatTable(){
    dreamweaver.runCommand('Format Table.htm');
}
```

Dateibearbeitungsfunktionen

Zu den Dateibearbeitungsfunktionen gehört das Erstellen, Öffnen und Speichern von Dokumenten (auch in den Formaten XML und XHTML), das Konvertieren bestehender HTML-Dokumente in XHTML-Dokumente sowie das Exportieren von CSS-Dokumenten in externe Dateien. Darüber hinaus können Sie nach Dateien und Ordnern suchen, Dateien aus Vorlagen erstellen, Dokumente schließen und Informationen über die zuletzt geöffneten Dateien abrufen.

dom.cleanupXHTML()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Diese Funktion ähnelt `convertToXHTML()`, optimiert aber vorhandene XHTML-Dokumente. Die Funktion kann auf eine Auswahl innerhalb des Dokuments angewendet werden. Mit `cleanupXHTML()` können Sie die Syntax eines gesamten XHTML-Dokuments oder eines ausgewählten Dokumentabschnitts optimieren.

Argumente
bWholeDoc

- Das Argument *bWholeDoc* enthält einen booleschen Wert. Ist der Wert auf `true` gesetzt, optimiert `cleanupXHTML()` das ganze Dokument, andernfalls wird nur der ausgewählte Abschnitt bearbeitet.

Rückgabewerte
Ein Array von sechs Ganzzahlen, die die Anzahl der folgenden Elemente bezeichnen:

- XHTML-Fehler, die von Dreamweaver repariert wurden
- `map`-Elemente ohne `id`-Attribut, die nicht repariert werden können

- `script`-Elemente ohne `type`-Attribut, die nicht repariert werden können
- `style`-Elemente ohne `type`-Attribut, die nicht repariert werden können
- `img`-Elemente ohne `alt`-Attribut, die nicht repariert werden können
- `area`-Elemente ohne `alt`-Attribute, die nicht repariert werden können

dom.convertToXHTML()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Analysiert aus dem HTML-Code eine DOM-Struktur, fügt fehlende, für XHTML erforderliche Elemente ein, optimiert die Struktur und erstellt dann einen ihr entsprechenden neuen XHTML-Code. Zu den fehlenden Direktiven, Deklarationen, Elementen und Attributen, die `convertToXHTML()` ggf. in die DOM-Struktur aufnimmt, gehören folgende Elemente:

- Eine XML-Direktive
- Eine Deklaration vom Typ `doctype`
- Das `xmlns`-Attribut im `html`-Element
- Ein `head`-Abschnitt
- Ein `title`-Element
- Ein `body`-Abschnitt

Während der Konvertierung konvertiert die Funktion `dom.convertToXHTML()` reine HTML-Tags und -Attribute in Kleinbuchstaben, schreibt HTML-Tags und -Attribute in korrekte XHTML-Syntax um und fügt fehlende HTML-Attribute dort ein, wo es möglich ist. Diese Funktion behandelt Tags und Attribute von Drittanbietern entsprechend den Einstellungen im Dialogfeld „Voreinstellungen“.

Ist das Dokument eine Vorlage, gibt `dom.convertToXHTML()` eine Meldung an den Benutzer weiter, führt aber keine Konvertierung durch.

Argumente
Keine.

Rückgabewerte
Ein Array von sechs Ganzzahlen, die die Anzahl der folgenden Elemente angeben:

- XHTML-Fehler, die von Dreamweaver repariert wurden
- `map`-Elemente ohne `id`-Attribut, die nicht repariert werden können
- `script`-Elemente ohne `type`-Attribut, die nicht repariert werden können
- `style`-Elemente ohne `type`-Attribut, die nicht repariert werden können
- `img`-Elemente ohne `alt`-Attribut, die nicht repariert werden können
- `area`-Elemente ohne `alt`-Attribute, die nicht repariert werden können

Beispiel
In der Regel ruft eine Erweiterung zunächst `dreamweaver.openDocument()` oder `dreamweaver.getDocumentDOM()` auf, um eine Referenz zu dem Dokument zu erhalten. Die Erweiterung ruft dann `dom.getIsXHTMLDocument()` auf, um zu ermitteln, ob das Dokument bereits das XHTML-Format aufweist. Ist dies nicht der Fall, ruft die Erweiterung die Funktion `dom.convertToXHTML()` auf, um das Dokument in XHTML zu konvertieren. Abschließend ruft die Erweiterung `dreamweaver.saveDocument()` auf, um das konvertierte Dokument unter einem neuen Dateinamen zu speichern.

dom.getIsXHTMLDocument()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Überprüft ein Dokument (insbesondere die Deklaration `<!DOCTYPE>`) darauf, ob das XHTML-Format vorliegt.

Argumente
Keine.

Rückgabewerte
`true` für XHTML-Dokumente, andernfalls `false`.

dreamweaver.browseForFileURL()

Verfügbarkeit
Dreamweaver 1, erweitert in 2, 3 und 4.

Beschreibung
Öffnet den angegebenen Dialogfeldtyp mit der angegebenen Bezeichnung in der Titelleiste.

Argumente
openSelectOrSave, {*titleBarLabel*}, {*bShowPreviewPane*}, {*bSupressSiteRootWarnings*}, {*arrayOfExtensions*}, {*startFolder*}, {*allowDynamic*}, {*fileToLocate*}

- Das Argument *openSelectOrSave* ist ein String, der den Dialogfeldtyp als `open`, `select` oder `save` bezeichnet.
- *titleBarLabel* wurde in Dreamweaver 2 eingeführt und enthält die Bezeichnung, die in der Titelleiste des Dialogfelds angezeigt werden soll. Bei Auslassung dieses Arguments wird die Standardbezeichnung des Betriebssystems verwendet.
- *bShowPreviewPane* wurde in Dreamweaver 2 eingeführt. Dies ist ein boolescher Wert, der angibt, ob im Dialogfeld der Bildvorschaubereich angezeigt werden soll. Lautet der Wert dieses Arguments `true`, wird die Anzeige im Dialogfeld nach Bildern gefiltert. Bei fehlendem Argument gilt der Standardwert`false`.
- *bSupressSiteRootWarnings* wurde in Dreamweaver 3 eingeführt. Wenn sich die ausgewählte Datei außerhalb des Site-Stamms befindet, gibt dieser boolesche Wert an, ob entsprechende Warnungen unterdrückt werden sollen. Bei fehlendem Argument gilt der Standardwert `false`.

- Das Argument *arrayOfExtensions* wurde in Dreamweaver 4 eingeführt. Es handelt sich hierbei um einen Array von Strings. Es gibt den Standardinhalt des Listenmenüs „Dateityp“ an, das sich im unteren Bereich des Dialogfelds befindet. Die Syntax für dieses Argument lautet *menuEntryText*|*.xxx[;.yyy;.zzz]*|*CCCC*|, wobei Folgendes gilt:
  - *menuEntryText* ist der Name des Dateityps.
  - Sie können die Erweiterungen wie folgt angeben: *.xxx[;.yyy;.zzz]* oder *CCCC*:
    - *.xxx* bezeichnet die Dateinamenerweiterung des Dateityps. Verwenden Sie *.yyy* und *.zzz* für die Angabe weiterer Dateinamenerweiterungen.
    - *CCCC* ist die aus vier Zeichen bestehende Dateitypkonstante für Macintosh.

  Im folgenden Beispiel werden in einem Auswahldialogfeld zwei Filter bereitgestellt: einer für MP3-Dateien und ein zweiter für alle Dateien.

```
dw.browseForFileURL("select", "Please select an mp3",false, true, new Array("mp3 Files
(*.MP3)|*.mp3||","All Files (*.*)|*.*||"));
```

- Das Argument *startFolder* ist ein Stringwert, mit dem Sie die Datei-URL des Ordners angeben können, in dem die Suche beginnt. Falls dieses Argument ausgelassen wird, beginnt die Suche beim zuletzt verwendeten Verzeichnis. Dieses Argument ist optional.
- Das Argument *allowDynamic* ist ein boolescher Wert, der angibt, ob dynamische URLs oder Parameter zulässig sind. Lautet der Wert dieses Arguments `true`, sind dynamische URLs oder Parameter zulässig. Dieses Argument ist optional.
- Das Argument *fileToLocate* ist ein Stringwert, mit dem die Datei-URL der zu suchenden Datei angegeben wird. Dieses Argument ist optional.

Rückgabewerte
Ein String mit dem Namen der Datei im URL-Format file://.

dreamweaver.browseForFolderURL()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Öffnet das Dialogfeld „Ordner wählen“ mit der entsprechenden Bezeichnung in der Titelleiste.

Argumente
{titleBarLabel}, {directoryToStartIn}

- Das Argument *titleBarLabel* ist die Bezeichnung, die in der Titelleiste des Dialogfelds angezeigt werden soll. Bei Auslassung wird für *titleBarLabel* der Text „Ordner wählen“ angezeigt.
- Das Argument *directoryToStartIn* ist der Pfad im URL-Format „file://“, in dem der Ordner geöffnet werden soll.

Rückgabewerte
Ein String mit dem Ordnernamen, der im URL-Format „file://“ angegeben wird.

Beispiel
Mit dem folgenden Code wird die URL eines Ordners zurückgegeben:

```
return dreamweaver.browseForFolderURL('Select a Folder', ¬
dreamweaver.getSiteRoot());
```

dreamweaver.closeDocument()

Verfügbarkeit
Dreamweaver 2.

Beschreibung
Schließt das angegebene Dokument.

Argumente
documentObject

- Das Argument *documentObject* ist das Objekt am Stamm der DOM-Struktur eines Dokuments (der von `dreamweaver.getDocumentDOM()` zurückgegebene Wert). Wenn sich das Argument *documentObject* auf das aktive Dokument bezieht, wird das Dokumentfenster erst nach Beendigung des aufrufenden Skripts geschlossen.

Rückgabewerte
Keine.

dreamweaver.createDocument()

Verfügbarkeit
Dreamweaver 2, erweitert in Dreamweaver 4.

Beschreibung
Abhängig vom Argument, das Sie an diese Funktion übergeben, öffnet sie ein neues Dokument entweder im gleichen oder in einem neuen Fenster. Das neue Dokument wird zum aktiven Dokument.

Hinweis: *Diese Funktion kann nur über die Datei „menus.xml", eine Befehlsdatei oder eine Eigenschafteninspektor-Datei aufgerufen werden. Wenn eine Verhaltensaktion oder ein Objekt diese Funktion aufzurufen versucht, wird eine Fehlermeldung ausgegeben.*

Argumente
{bOpenInSameWindow}, {type}

- Das Argument *bOpenInSameWindow* ist ein boolescher Wert, der angibt, ob das neue Dokument im aktuellen Fenster geöffnet werden soll. Wenn *bOpenInSameWindow* den Wert `false` hat oder ausgelassen wird oder wenn die Funktion auf einem Macintosh aufgerufen wird, wird das neue Dokument in einem eigenen Fenster geöffnet.
- Das Argument *type* gibt den zu erstellenden Dokumenttyp an, entsprechend dem in der Dreamweaver-Datei „Configuration/DocumentTypes/MMDocumentTypes.xml" festgelegten `id`-Attribut des Tags `documenttype`. Das Argument *type* kann beispielsweise `"HTML", "ASP-JS", "ASP-VB", "ColdFusion", "CFC", "JSP", "ASP.NET_VB"` usw. lauten. Eine vollständige Liste möglicher Dokumenttypen finden Sie in der Datei „MMDocumentTypes.xml". Wenn Sie *type* nicht angeben, wird der Wert standardmäßig auf `"HTML"` gesetzt.

Hinweis: *Sie können die Datei „MMDocumentTypes" erweitern, indem Sie eigene Dokumenttypen hinzufügen. Weitere Informationen über das Erweitern von Dokumenttypen finden Sie in „Dreamweaver erweitern".*

Rückgabewerte
Das Dokumentobjekt für das neu erstellte Dokument. Dabei handelt es sich um den gleichen Wert, der von `dreamweaver.getDocumentDOM()` zurückgegeben wird.

dreamweaver.createXHTMLDocument()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Abhängig vom Argument, das Sie an diese Funktion weitergeben, öffnet sie das neue XHTML-Dokument entweder im gleichen oder in einem neuen Fenster. Das neue Dokument wird zum aktiven Dokument. Diese Funktion ähnelt der Funktion `dreamweaver.createDocument()`.

Wenn Sie in Dreamweaver ein neues XHTML-Dokument erstellen, wird die Datei „default.xhtml“ im Ordner „Configuration/Templates“ gelesen. Anhand dieser Datei wird eine Ausgabedatei mit folgenden Grunddeklarationen erstellt:

```
<?xml version="1.0">
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Untitled Document</title>
<meta http-equiv="Content-Type" content="text/html; charset=" />
</head>

<body bgcolor="#FFFFFF" text="#000000">

</body>
</html>
```

Die Standarddeklaration der Dokumententypdefinition (DTD) ist `XHTML 1.0 Transitional` und nicht `Strict`. Wenn ein Benutzer dem Dokument ein Frameset hinzufügt, wird die DTD in `XHTML 1.0 Frameset` geändert. `Content-Type` ist `text/html` und `charset` wird zunächst bewusst nicht in die Datei „default.xhtml“ aufgenommen, dann jedoch eingefügt, bevor der Benutzer das neue Dokument anzeigt. Die Direktive `?xml` ist nicht erforderlich, wenn das Dokument mit der UTF-8- oder UTF-16-Zeichenverschlüsselung arbeitet. Wenn sie vorhanden ist, kann sie möglicherweise von älteren Browsern wiedergegeben werden. Da diese Direktive aber in einem XHTML-Dokument enthalten sein sollte, wird sie von Dreamweaver sowohl für neue als auch für konvertierte Dokumente verwendet. Benutzer können die Direktive manuell löschen. Die Direktive `?xml` enthält das Verschlüsselungsattribut, das dem Argument `charset` im Attribut `Content-Type` entspricht.

Argumente
{bOpenInSameWindow}

- Das Argument *bOpenInSameWindow* ist ein boolescher Wert, der angibt, ob das neue Dokument im aktuellen Fenster geöffnet werden soll. Wenn dieses Argument den Wert `false` hat oder ausgelassen wird oder wenn die Funktion auf einem Macintosh aufgerufen wird, wird das neue Dokument in einem eigenen Fenster geöffnet.

Rückgabewerte
Das Dokumentobjekt für das neu erstellte Dokument, d. h. der gleiche Wert, der von `dreamweaver.getDocumentDOM()` zurückgegeben wird.

dreamweaver.createXMLDocument()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Erstellt und öffnet eine neue XML-Datei, die nur die XML-Direktive enthält.

Argumente
Keine.

Rückgabewerte
Das DOM der neuen XML-Datei.

Beispiel
Im folgenden Beispiel wird ein neues Dokument erstellt, das lediglich die XML-Direktive enthält:

```
var theDOM = dreamweaver.createXMLDocument("document");
```

dreamweaver.exportTemplateDataAsXML()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Exportiert das aktuelle Dokument als XML in die angegebene Datei. Diese Funktion wirkt sich auf das aktive Dokument aus, bei dem es sich um eine Vorlage handeln muss Wenn Sie kein Dateinamenargument angeben, öffnet Dreamweaver MX ein Dialogfeld, in dem der Exportdatei-String angefordert wird.

Argumente
{filePath}

- Das optionale Argument *filePath* ist ein String, der den Namen der Datei angibt, in die Dreamweaver die Vorlage exportiert. Das Argument *filepath* muss als URL-Dateistring angegeben werden, z. B. `"file:///c|/temp/mydata.txt"`.

Rückgabewerte
Keine.

Enabler
Siehe „dreamweaver.canExportTemplateDataAsXML()“ auf Seite 518.

Beispiel

```
if(dreamweaver.canExportTemplateDataAsXML())
{
    dreamweaver.exportTemplateDataAsXML("file:///c|/dw_temps/mytemplate.txt")
}
```

dreamweaver.getDocumentDOM()

Verfügbarkeit

Dreamweaver 2.

Beschreibung

Ermöglicht den Zugriff auf die Objektstruktur des angegebenen Dokuments. Nachdem die Objektstruktur an die aufrufende Funktion zurückgegeben wurde, kann diese die Struktur und damit den Dokumentinhalt ändern.

Argumente

{sourceDoc}

- Das Argument *sourceDoc* muss `"document"`, `"parent"`, `"parent.frames[number]"`, `"parent.frames['frameName']"` lauten oder eine URL sein. Der Wert `sourceDoc` wird standardmäßig auf `"document"` gesetzt, wenn Sie keinen Wert angeben. Diese Argumentwerte haben folgende Bedeutung:
  - Der Wert `document` gibt das Dokument an, das aktiv ist und die aktuelle Auswahl enthält.
  - Der Wert `parent` gibt das übergeordnete Frameset an (sofern sich das derzeit ausgewählte Dokument in einem Frame befindet).
  - Die Werte `parent.frames[number]` und `parent.frames['frameName']` geben ein Dokument in einem bestimmten Frame eines Framesets an, der das aktuelle Dokument enthält.
- Wenn das Argument eine relative URL ist, bezieht sie sich auf die Erweiterungsdatei.

Hinweis: *Wenn das Argument `"document"` lautet, muss die aufrufende Funktion entweder `applyBehavior()`, `deleteBehavior()`, `objectTag()` oder eine beliebige Funktion in einer Befehls- oder Eigenschafteninspektor-Datei sein, mit der das Dokument bearbeitet werden kann.*

Rückgabewerte

Das JavaScript-Dokumentobjekt am Stamm der Struktur.

Beispiele

Im folgenden Beispiel wird das aktuelle Dokument mit `dreamweaver.getDocumentDOM()` aufgerufen:

```
var theDOM = dreamweaver.getDocumentDOM("document");
```

Im folgenden Beispiel identifiziert das DOM eine Auswahl und fügt diese am Ende eines anderen Dokuments ein:

```
var currentDOM = dreamweaver.getDocumentDOM('document');
currentDOM.setSelection(100,200);
currentDOM.clipCopy();
var otherDOM = dreamweaver.openDocument(dreamweaver.¬
getSiteRoot() + "html/foo.htm");
otherDOM.endOfDocument();
otherDOM.clipPaste();
```

***Hinweis:** Das Argument `openDocument()` wird verwendet, da `DOM`-Methoden in der Regel nur auf geöffnete Dokumente angewendet werden können. Das Ausführen einer Funktion für ein geschlossenes Dokument ruft eine Dreamweaver-Fehlermeldung hervor. Bei `DOM`-Methoden, die nur auf das aktive Dokument oder auf geschlossene Dokumente angewendet werden können, wird in der jeweiligen Beschreibung auf diesen Umstand hingewiesen.*

dreamweaver.getNewDocumentDOM()

Verfügbarkeit
Dreamweaver MX. Argument `documentType` wurde in Dreamweaver 8 hinzugefügt.

Beschreibung
Ermöglicht den Zugriff auf die bearbeitbare Struktur für ein neues, leeres Dokument. Diese Funktion ähnelt der Funktion `getDocumentDOM()` mit dem Unterschied, dass sie auf ein neues Dokument statt auf ein vorhandenes verweist und das Dokument nicht öffnet.

Argumente
{documentType}

- Das Argument *documentType* ist ein String. Sein Wert muss einer der Dokumenttypen sein, die in der Datei „DocumentTypes.xml“ festgelegt sind.

Rückgabewerte
Ein Verweis auf ein neues, leeres Dokument.

Beispiel
Der folgende Code gibt das DOM für ein neues, leeres Dokument zurück:

```
var theDOM = dreamweaver.getNewDocumentDOM();
```

dreamweaver.getRecentFileList()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Ruft eine Liste der zuletzt geöffneten Dateien ab, die im unteren Bereich des Menüs „Datei“ angezeigt werden.

Argumente
Keine.

Rückgabewerte
Ein Array von Strings, die die Pfade der zuletzt aufgerufenen Dateien darstellen. Jeder Pfad wird im URL-Format „file://“ angegeben. Wenn keine zuletzt geöffneten Dateien vorhanden sind, wird kein Wert zurückgegeben.

dreamweaver.importXMLIntoTemplate()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Importiert eine XML-Textdatei in das aktuelle Vorlagendokument. Diese Funktion wirkt sich auf das aktive Dokument aus, bei dem es sich um eine Vorlage handeln muss Wenn Sie kein Dateinamenargument angeben, öffnet Dreamweaver ein Dialogfeld, in dem der Importdatei-String angefordert wird.

Argumente
{filePath}

- Das optionale Argument *filePath* ist ein String, der den Namen der Datei angibt, in die Dreamweaver die Vorlage importiert. Geben Sie das Argument *filepath* als URL-Dateistring an, z. B. als `"file:///c|/temp/mydata.txt"`.

Rückgabewerte
Keine.

dreamweaver.newDocument()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Öffnet in der aktuellen Site ein neues Dokument und ruft das Dialogfeld „Neues Dokument“ auf.

Argumente
{bopenWithCurSiteAndShowDialog}

- Das optionale Argument *bopenWithCurSiteAndShowDialog* hat den Wert `true` oder `false`. Geben Sie `true` an, um in der aktuellen Site ein neues Dokument zu öffnen und das Dialogfeld „Neues Dokument“ aufzurufen. Verwenden Sie andernfalls `false`.

Rückgabewerte
Keine.

dreamweaver.newFromTemplate()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Erstellt ein neues Dokument aus der angegebenen Vorlage. Wenn kein Argument übergeben wird, wird das Dialogfeld „Vorlage auswählen“ eingeblendet.

Argumente
{templateURL}, bMaintain

- Das Argument *templateURL* ist der Pfad zu einer Vorlage in der aktuellen Site im URL-Format „file://“.
- Das Argument *bmaintain* ist ein boolescher Wert (`true` oder `false`), der angibt, ob der Link zur ursprünglichen Vorlage beibehalten werden soll.

Rückgabewerte
Keine.

dreamweaver.openDocument()

Verfügbarkeit
Dreamweaver 2.

Beschreibung
Öffnet ein Dokument zur Bearbeitung in einem neuen Dreamweaver-Fenster und verlagert den Fokus auf dieses Fenster. Den gleichen Effekt erzielen Sie, wenn Sie den Befehl „Datei“ > „Öffnen“ wählen und dann eine Datei auswählen. Wenn die betreffende Datei bereits geöffnet ist, wird das zugehörige Dokumentfenster in den Vordergrund gestellt. Dieses Fenster wird zum aktuell ausgewählten Dokument. In Dreamweaver 2 wird die Datei bei aktiviertem Einchecken/Auschecken vor dem Öffnen ausgecheckt. In Dreamweaver 3 oder höher müssen Sie hierzu „dreamweaver.openDocumentFromSite()“ auf Seite 270 aufrufen.

Hinweis: *Wenn diese Funktion aus einer Verhaltensaktion oder einer Objektdatei aufgerufen wird, verursacht sie einen Fehler.*

Argumente
fileName

- Das Argument *fileName* ist der Name der zu öffnenden Datei im URL-Format. Falls es sich um eine relative URL handelt, bezieht sie sich auf die Skriptdatei, von der aus diese Funktion aufgerufen wurde.

Rückgabewerte
Das Dokumentobjekt für die angegebene Datei, d. h. derselbe Wert, der von `dreamweaver.getDocumentDOM()` zurückgegeben wird.

dreamweaver.openDocumentFromSite()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Öffnet ein Dokument zur Bearbeitung in einem neuen Dreamweaver-Fenster und verlagert den Fokus auf dieses Fenster. Den gleichen Effekt erzielen Sie, wenn Sie im Bedienfeld „Dateien“ auf eine Datei doppelklicken. Wenn die betreffende Datei bereits geöffnet ist, wird das zugehörige Dokumentfenster in den Vordergrund gestellt. Dieses Fenster wird zum aktuell ausgewählten Dokument.

Hinweis: *Diese Funktion kann nicht aus einer Verhaltensaktion oder einer Objektdatei aufgerufen werden, da dies eine Fehlermeldung verursacht.*

Argumente
fileName

- Das Argument *fileName* gibt die zu öffnende Datei im URL-Format an. Falls es sich um eine relative URL handelt, bezieht sie sich auf die Skriptdatei, von der aus diese Funktion aufgerufen wurde.

Rückgabewerte
Das Dokumentobjekt für die angegebene Datei, d. h. derselbe Wert, der von `dreamweaver.getDocumentDOM()` zurückgegeben wird.

dreamweaver.openInFrame()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Öffnet das Dialogfeld „Öffnen in Frame“. Wenn der Benutzer ein Dokument auswählt, wird es im aktiven Frame geöffnet.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dreamweaver.canOpenInFrame()“ auf Seite 520.

dreamweaver.releaseDocument()

Verfügbarkeit
Dreamweaver 2.

Beschreibung
Gibt ein zuvor referenziertes Dokument explizit aus dem Speicher frei.

Dokumente, die von `dreamweaver.getObjectTags()`, `dreamweaver.getObjectRefs()`, `dreamweaver.getDocumentPath()` oder `dreamweaver.getDocumentDOM()` referenziert werden, werden nach Beendigung des aufrufenden Skripts automatisch freigegeben. Wenn das Skript viele Dokumente öffnet, müssen Sie sie mit der Funktion explizit freigeben, bevor Sie das Skript beenden. Dadurch werden Speicherprobleme verhindert.

Hinweis: *Diese Funktion ist nur für Dokumente relevant, die von einer URL referenziert wurden, momentan nicht in einem Frame oder Dokumentfenster geöffnet sind und bei denen es sich nicht um Erweiterungsdateien handelt. Erweiterungsdateien werden beim Start in den Speicher geladen und erst beim Beenden von Dreamweaver freigegeben.*

Argumente
documentObject

- Das Argument *documentObject* ist das Objekt am Stamm der DOM-Struktur eines Dokuments, welches der von `dreamweaver.getDocumentDOM()` zurückgegebene Wert ist.

Rückgabewerte
Keine.

dreamweaver.revertDocument()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Stellt das angegebene Dokument wieder in der zuvor gespeicherten Version her.

Argumente
`documentObject`, *`warn`*

- Das Argument *`documentObject`* ist das Objekt am Stamm der DOM-Struktur des Dokuments, welches der von`dreamweaver.getDocumentDOM()` zurückgegebene Wert ist.
- Das Argument *`warn`* ist ein boolescher Wert, der angibt, ob eine Warnung mit dem Hinweis eingeblendet wird, dass nicht gespeicherte Änderungen verloren gehen. Bei Auslassung dieses Arguments wird der Standardwert `true` verwendet.

Rückgabewerte
Ein boolescher Wert: `true`, wenn in Dreamweaver eine Warnmeldung angezeigt werden soll, andernfalls `false`.

Enabler
Siehe „dreamweaver.canRevertDocument()“ auf Seite 521.

dreamweaver.saveAll()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Speichert alle geöffneten Dokumente. Für Dokumente, die noch nicht gespeichert wurden, wird das Dialogfeld „Speichern unter“ angezeigt.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dreamweaver.canSaveAll()“ auf Seite 522.

dreamweaver.saveDocument()

Verfügbarkeit
Dreamweaver 2.

Beschreibung
Speichert die angegebene Datei auf einem lokalen Computer.

Hinweis: *In Dreamweaver 2 wird versucht, die Datei auszuchecken, falls diese schreibgeschützt ist. Wenn das Dokument auch nach diesem Versuch noch schreibgeschützt ist bzw. nicht erstellt werden kann, wird eine Fehlermeldung angezeigt.*

Argumente
documentObject, {fileURL}

- Das Argument *documentObject* ist das Objekt am Stamm der DOM-Struktur eines Dokuments, welches der von `dreamweaver.getDocumentDOM()` zurückgegebene Wert ist.
- Das optionale Argument *fileURL* ist eine URL, die einen Speicherort auf dem lokalen Laufwerk angibt. Wenn es sich um eine relative URL handelt, bezieht sie sich auf die Erweiterungsdatei. In Dreamweaver 2 muss dieses Argument angegeben werden. Wenn das Argument *fileURL* in Dreamweaver 4 ausgelassen wird und die Datei zuvor bereits gespeichert wurde, wird sie am aktuellen Speicherort gespeichert. Andernfalls wird das Dialogfeld „Speichern“ angezeigt.

Rückgabewerte
Ein boolescher Wert, der angibt, ob der Vorgang erfolgreich war (`true`) oder fehlgeschlagen ist (`false`).

Enabler
Siehe „dreamweaver.canSaveDocument()“ auf Seite 522.

dreamweaver.saveDocumentAs()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Öffnet das Dialogfeld „Speichern unter“.

Argumente
documentObject

- Das Argument *documentObject* ist das Objekt am Stamm der DOM-Struktur eines Dokuments, welches der von `dreamweaver.getDocumentDOM()` zurückgegebene Wert ist.

Rückgabewerte
Keine.

dreamweaver.saveDocumentAsTemplate()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Öffnet das Dialogfeld „Als Vorlage speichern“.

Argumente
documentObject, {fileName}

- Das Argument *documentObject* ist das Objekt am Stamm der DOM-Struktur eines Dokuments, welches der von `dreamweaver.getDocumentDOM()` zurückgegebene Wert ist.
- Das optionale Argument *fileName* ist der Name der zu öffnenden Datei, der als absolute URL angegeben wird.

Rückgabewerte
Keine.

Enabler
Siehe „dreamweaver.canSaveDocumentAsTemplate()“ auf Seite 522.

dreamweaver.saveFrameset()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Speichert das angegebene Frameset bzw. öffnet das Dialogfeld „Speichern unter“, falls das Frameset zuvor noch nicht gespeichert wurde.

Argumente
documentObject

- Das Argument *documentObject* ist das Objekt am Stamm der DOM-Struktur eines Dokuments, welches der von `dreamweaver.getDocumentDOM()` zurückgegebene Wert ist.

Rückgabewerte
Keine.

Enabler
Siehe „dreamweaver.canSaveFrameset()“ auf Seite 523.

dreamweaver.saveFramesetAs()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Öffnet das Dialogfeld „Speichern unter“ für die Frameset-Datei, die das angegebene DOM enthält.

Argumente
documentObject

- Das Argument *documentObject* ist das Objekt am Stamm der DOM-Struktur eines Dokuments, welches der von `dreamweaver.getDocumentDOM()` zurückgegebene Wert ist.

Rückgabewerte
Keine.

Enabler
Siehe „dreamweaver.canSaveFramesetAs()“ auf Seite 523.

Globale Dokumentfunktionen

Globale Dokumentfunktionen wirken sich auf ein gesamtes Dokument aus. Mit ihrer Hilfe können Sie die Rechtschreibprüfung durchführen, Zielbrowser überprüfen, Seiteneigenschaften festlegen und für Elemente im Dokument die korrekten Objektreferenzen bestimmen.

dom.checkSpelling()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Führt im Dokument die Rechtschreibprüfung durch und öffnet dazu bei Bedarf das entsprechende Dialogfeld. Am Ende der Überprüfung wird eine Meldung angezeigt.

Argumente
Keine.

Rückgabewerte
Keine.

dom.checkTargetBrowsers()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Führt für ein Dokument eine Zielbrowser-Prüfung durch. Wie Sie für einen Ordner oder mehrere Dateien eine Zielbrowser-Prüfung durchführen, ist unter „site.checkTargetBrowsers()“ auf Seite 234 beschrieben.

Argumente
Keine.

Rückgabewerte
Keine.

dom.getParseMode()

Verfügbarkeit
Dreamweaver MX 2004.

Beschreibung
Ruft den aktuellen Analysemodus des Dokuments ab, der bestimmt, wie das Dokument ausgewertet wird und ob es im Hauptdokumentfenster als HTML angezeigt wird.

Argumente
Keine.

Rückgabewerte
Ein String, der den aktuellen Analysemodus angibt: `"html"`, `"xml"`, `"css"` oder `"text"`.

dom.hideInfoMessagePopup()

Verfügbarkeit
Dreamweaver MX 2004.

Beschreibung
Blendet die QuickInfo-ähnliche Meldung für das Dokumentfenster aus, falls sie angezeigt wird.

Argumente
Keine.

Rückgabewerte
Keine.

Siehe auch
„dom.showInfoMessagePopup()“ auf Seite 278.

dom.runValidation()

Verfügbarkeit
Dreamweaver MX, optionale Argumente in Dreamweaver MX 2004 hinzugefügt.

Beschreibung

Untersucht ein bestimmtes, einzelnes Dokument mithilfe des Validators. Der Validator überprüft, ob das Dokument mit der im „doctype“ angegebenen Sprache (z. B. HTML 4.0 oder HTML 3.2) und der durch das Servermodell bestimmten Sprache (z. B. ColdFusion oder ASP) übereinstimmt. Wenn das Dokument keinen „doctype“ hat, verwendet der Validator die im Dialogfeld „Voreinstellungen“ unter „Validator“ festgelegte Sprache.

Argumente

{controlString}, *{bOpenResultsWindow}*, *{bShowInfoMessage}*

- Das Argument *controlString* ist ein optionaler String, der einen der folgenden vier Werte aufweisen kann: leerer String, `"xml"`, `"auto-explicit"` oder `"auto-implicit"`.
  - Wenn das Argument ein leerer String ist, führt der Validator eine Standardüberprüfung durch. Wenn das Argument den Wert `"xml"` hat, wertet der Validator das Dokument als XML aus.
  - Lautet das Argument `"auto-explicit"` oder `"auto-implicit"`, führt Dreamweaver eine automatische Überprüfung durch (auch als *Inline*-Überprüfung bezeichnet). Das Fenster mit den Auswertungsergebnissen wird dabei nicht geöffnet; Fehler werden vielmehr in der Codeansicht unterstrichen (siehe „dom.source.getValidationErrorsForOffset()“ auf Seite 491 und„dom.getAutoValidationCount()“ auf Seite 484.
  - Wenn das Argument *controlString* den Wert `"auto-explicit"` hat, fordert Dreamweaver den Benutzer auf, ungespeicherte Dokumente zu speichern, bevor die Auswertung durchgeführt wird.
  - Wenn *controlString* den Wert `"auto-implicit"` hat, schlägt die Auswertung fehl, ohne dass der Benutzer darauf hingewiesen wird, dass das aktuelle Dokument noch nicht gespeichert wurde.

Hinweis: *Eine automatische Überprüfung (definiert durch den `controlString`-Wert `"auto-explicit"` oder `"auto-implicit"`) ist derzeit nur für die Browserkompatibilitätsprüfung verfügbar.*

- Das Argument *bOpenResultsWindow* ist ein optionaler boolescher Wert: Bei `true` wird das Auswertungsergebnisfenster geöffnet, bei `false` bleibt es geschlossen. Der Standardwert ist `false`.
- Das Argument *bShowInfoMessage* wird nur verwendet, wenn das Argument *controlString* als `"auto-explicit"` oder `"auto-implicit"` definiert ist. Das Argument *bShowInfoMessage* ist ein boolescher Wert. Bei `true` wird unterhalb des Symbolleistenelements `DW_ValidatorErrors` eine Meldung mit der Anzahl der gefundenen Fehler angezeigt, bei `false` wird nichts angezeigt. Der Standardwert ist `false`.

Rückgabewerte

Das Auswertungsergebnisfenster-Objekt.

Beispiel

Im folgenden Beispiel wird eine reguläre Überprüfung durchgeführt, wenn der Benutzer die Menüoption „Datei“ > „Überprüfen“ > „Markup“ (bzw. „Aktuelles Dokument überprüfen“ im Bedienfeld „Überprüfung“) auswählt:

```
dw.getDocumentDOM().runValidation('');
```

Im folgenden Beispiel wird der Benutzer zunächst zum Speichern eines ungespeicherten Dokuments aufgefordert. Dann wird eine automatische Überprüfung ausgeführt. Das Auswertungsergebnisfenster wird nicht geöffnet; die Gesamtanzahl von Fehlern wird vielmehr über der Symbolleistenschaltfläche `DW_ValidatorErrors` angezeigt:

```
dw.getDocumentDOM().runValidation('auto-explicit', false, true);
```

Im folgenden Beispiel wird der Benutzer nicht zum Speichern eines ungespeicherten Dokuments aufgefordert. Falls das Dokument nicht gespeichert wurde, wird die Überprüfung nicht gestartet. Wenn das Dokument gespeichert wurde, führt Dreamweaver eine automatische Überprüfung aus. Dabei wird das Auswertungsergebnisfenster nicht geöffnet und die Gesamtanzahl von Fehlern wird nicht in der Dokument-Symbolleiste angezeigt:

```
dw.getDocumentDOM().runValidation('auto-implicit', false);
```

dom.showInfoMessagePopup()

Verfügbarkeit
Dreamweaver MX 2004.

Beschreibung
Zeigt eine QuickInfo-ähnliche Meldung im Dokumentfenster oder unter einem Symbolleistenelement an.

Argumente
location, message, timeout

- Das Argument *location* ist entweder ein String, der ein Symbolleistenelement angibt, oder ein leerer String oder eines der folgenden Schlüsselworte: `"top"`, `"topright"`, `"right"`, `"bottomright"`, `"bottom"`, `"bottomleft"`, `"left"` oder `"topleft"`. Die QuickInfo wird auf der angegebenen Seite bzw. in der genannten Ecke angezeigt und zentriert. Bei einem leeren String wird sie im Dokument zentriert. Um ein Symbolleistenelement festzulegen, verwenden Sie `"toolbar:toolbarID:itemID"`, wobei „toolbarID“ und „itemID“ den IDs in der Datei „toolbars.xml“ entsprechen.
- Das Argument *message* ist ein String, der die Meldung enthält.
- Das Argument *timeout* ist eine Zahl, die die Anzeigedauer der Meldung in Millisekunden angibt. Der Standardwert ist 0. Wenn der Wert 0 ist, wird die Meldung dauerhaft angezeigt. Wenn der Benutzer darauf klickt oder zu einem anderen Dokument wechselt oder sobald das Zeitlimit überschritten wird, schließt Dreamweaver die Meldung automatisch.

Rückgabewerte
Keine.

Beispiel
Im folgenden Beispiel werden zwei QuickInfo-Meldungen angezeigt. Die erste Codezeile zeigt die Meldung `"This message is in the center"` in der Mitte des Dokuments an. Der zweite Aufruf an `showInfoMessagePopup()` blendet die QuickInfo-Meldung `"Don't forget the title for the Window"` für das Textfeld des Titels (ID `DW_SetTitle`) in der Symbolleiste mit der ID `DW_Toolbar_Main` ein.

```
dw.getDocumentDOM.showInfoMessagePopup('', 'This message is in the center', 5000);
dw.getDocumentDOM.showInfoMessagePopup('toolbar:DW_Toolbar_Main:DW_SetTitle', 'Don't
    forget the title for the window', 5000);
```

Siehe auch
„dom.hideInfoMessagePopup()“ auf Seite 276.

dom.showPagePropertiesDialog()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Öffnet das Dialogfeld „Seiteneigenschaften“.

Argumente
Keine.

Rückgabewerte
Keine.

dreamweaver.doURLDecoding()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Verwendet den internen URL-Dekodierungsmechanismus von Dreamweaver, um Sonderzeichen und Symbole in URL-Strings zu dekodieren. Diese Funktion dekodiert beispielsweise `%20` als Leerzeichen und den String `&quot` als gerades Anführungszeichen (").

Argumente
`inStr`

- Das Argument `inStr` ist der zu dekodierende String.

Rückgabewerte
Ein String, der die dekodierte URL enthält.

Beispiel
Im folgenden Beispiel wird `dw.doURLDecoding()` aufgerufen, um die Sonderzeichen im Argument zu dekodieren und das Ergebnis in `outstr` zu speichern:

```
outStr = dreamweaver.doURLDecoding("http://maps.yahoo.com/py/ddResults.py?Pyt= ¬
Tmap&tarname=&tardesc=&newname=&newdesc=&newHash=&newTHash=&newSts=&newTSts=&tlt=&tln= ¬
&slt=&sln=&newFL=Use+Address+Below&newaddr=2000+Shamrock+Rd&newcsz=Metroo+Park%2C+CA& ¬
newcountry=us&newTFL=Use+Address+Below&newtaddr=500+El+Camino&newtcsz=Santa+Clara%2C+CA& ¬
newtcountry=us&Submit=Get+Directions")
```

dreamweaver.getElementRef()

Verfügbarkeit
Dreamweaver 2.

Beschreibung
Ruft für ein bestimmtes Tag-Objekt in der DOM-Struktur die Objektreferenz für Netscape Navigator oder Microsoft Internet Explorer ab.

Argumente
NSorIE, tagObject

- Das Argument *NSorIE* muss entweder `"NS 4.0"` oder `"IE 4.0"` lauten. Das DOM und die Richtlinien für verschachtelte Referenzen sind in Netscape Navigator 4.0 und Internet Explorer 4.0 nicht identisch. Dieses Argument gibt an, für welchen Browser eine gültige Referenz zurückgegeben werden soll.
- Das Argument *tagObject* ist ein Tag-Objekt in der DOM-Struktur.

Rückgabewerte
Ein String mit einer gültigen JavaScript-Referenz zu dem Objekt, beispielsweise `document.layers['myLayer']`. Der String unterliegt folgenden Bedingungen:

- Bei Microsoft Internet Explorer werden für die Tags `A`, `AREA`, `APPLET`, `EMBED`, `DIV`, `SPAN`, `INPUT`, `SELECT`, `OPTION`, `TEXTAREA`, `OBJECT` und `IMG` korrekte Referenzen zurückgegeben.
- Bei Netscape Navigator werden für die Tags `A`, `AREA`, `APPLET`, `EMBED`, `LAYER`, `ILAYER`, `SELECT`, `OPTION`, `TEXTAREA`, `OBJECT` und `IMG` sowie für absolut positionierte Tags der Typen `DIV` und `SPAN` korrekte Referenzen zurückgegeben. Für nicht absolut positionierte Tags der Typen `DIV` und `SPAN` wird `"cannot reference <tag>"` zurückgegeben.
- Für unbenannte Objekte werden keine Referenzen zurückgegeben. Wenn ein Objekt weder das Attribut `NAME` noch das Attribut `ID` enthält, wird `"unnamed <tag>"` zurückgegeben. Wenn der Browser keine Referenz nach Name unterstützt, wird das Objekt nach Index referenziert (z. B. `document.myform.applets[3]`).
- Für benannte Objekte in unbenannten Formularen und Ebenen (z. B. `document.forms[2].myCheckbox`) werden Referenzen zurückgegeben.

dreamweaver.getPreferenceInt()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Ermöglicht den Abruf eines ganzzahligen Voreinstellungswerts für eine Erweiterung.

Argumente
section, key, default_value

- Das Argument *section* ist ein String mit dem Voreinstellungsabschnitt, der den Eintrag enthält.
- Das Argument *key* ist ein String, der den Eintrag des abzurufenden Werts angibt.
- Das Argument *default_value* ist der Standardwert, den Dreamweaver zurückgibt, wenn der Eintrag nicht gefunden werden konnte. Hierbei muss es sich um eine Ganzzahl ohne Vorzeichen im Bereich 0 bis 65.535 oder um eine Ganzzahl mit Vorzeichen im Bereich -32.768 bis 32.767 handeln.

Rückgabewerte
Ganzzahlwert des angegebenen Eintrags im angegebenen Abschnitt oder der Standardwert, wenn die Funktion den Eintrag nicht findet. Gibt 0 zurück, wenn der Wert des angegebenen Eintrags keine Ganzzahl ist.

Beispiel
Im folgenden Beispiel wird der Ganzzahlwert für den Ausrichtungsabstand (Snap Distance) im Abschnitt „My Extension“ der Voreinstellungen zurückgegeben. Wenn der Abschnitt „My Extension“ oder der Eintrag „Snap Distance“ nicht vorhanden ist, gibt die Funktion den Standardwert 0 zurück.

```
var snapDist; //default value if entry not found
snapDist = dreamweaver.getPreferenceInt("My Extension", "Snap Distance", 0);
```

dreamweaver.getPreferenceString()

Verfügbarkeit
Dreamweaver MX.

Hinweis: *Die Voreinstellungen für Sites sind erst ab Version 7.0.1 zugänglich. Überprüfen Sie* `dw.appVersion` *auf die korrekte Version, bevor Sie die Site-Informationen abrufen.*

Beschreibung
Ermöglicht den Abruf eines String-Voreinstellungswerts, den Sie für eine Erweiterung gespeichert haben.

Argumente
section, key, default_value

- Das Argument *section* ist ein String mit dem Voreinstellungsabschnitt, der den Eintrag enthält.
- *key* ist ein String, der den abzurufenden Wert angibt.
- Das Argument *default_value* ist der Standardstringwert, den Dreamweaver zurückgibt, wenn der Eintrag nicht gefunden werden konnte.

Rückgabewerte
Der angeforderte Voreinstellungsstring, oder, wenn dieser nicht gefunden werden konnte, der Standardwert.

Beispiel
Im folgenden Beispiel wird der Stringwert für den Texteditor im Abschnitt „My Extension“ der Voreinstellungen zurückgegeben. Wenn der Abschnitt „My Extension“ oder der Eintrag „Text Editor“ nicht vorhanden ist, gibt die Funktion den durch die Variable `txtEditor` festgelegten Standardwert zurück.

```
var txtEditor = getExternalTextEditor(); //set default text Editor value
txtEditor = dreamweaver.getPreferenceString("My Extension", "Text Editor", txtEditor);
```

dreamweaver.setPreferenceInt()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Ermöglicht die Festlegung eines ganzzahligen Voreinstellungswerts für eine Erweiterung. Diese Einstellung wird in den Dreamweaver-Voreinstellungen gespeichert, wenn Dreamweaver nicht ausgeführt wird.

Argumente
section, key, new_value

- Das Argument *section* ist ein String mit der Voreinstellungskategorie, in der die Option festgelegt wird. Wenn die Kategorie nicht vorhanden ist, wird sie von Dreamweaver erstellt.
- Das Argument *key* ist ein String, der die durch die Funktion festgelegte Kategorieoption angibt. Wenn die Option nicht vorhanden ist, wird sie von Dreamweaver erstellt.
- Das Argument *new_value* ist eine Ganzzahl, die den Wert der Kategorieoption angibt.

Rückgabewerte
`true`, wenn erfolgreich, andernfalls `false`.

Beispiel
Im folgenden Beispiel wird der Eintrag „Snap Distance“ in der Kategorie „My Extension“ der Voreinstellungen auf den Wert der Variablen `snapDist` gesetzt:

```
var snapDist = getSnapDistance();
if(snapDist > 0)
{
    dreamweaver.setPreferenceInt("My Extension", "Snap Distance", snapDist);
}
```

dreamweaver.setPreferenceString()

Verfügbarkeit
Dreamweaver MX.

Hinweis: *Die Voreinstellungen für Sites sind erst ab Version 7.0.1 zugänglich. Überprüfen Sie* `dw.appVersion` *auf die korrekte Version, bevor Sie die Site-Informationen abrufen.*

Beschreibung
Ermöglicht die Festlegung eines String-Voreinstellungswerts für eine Erweiterung. Diese Einstellung wird in den Dreamweaver-Voreinstellungen gespeichert, wenn Dreamweaver nicht ausgeführt wird.

Argumente
section, key, new_value

- Das Argument *section* ist ein String mit der Voreinstellungskategorie, in der die Option festgelegt wird. Wenn die Kategorie nicht vorhanden ist, wird sie von Dreamweaver erstellt.
- Das Argument *key* ist ein String, der die durch die Funktion festgelegte Kategorieoption angibt. Wenn die Kategorieoption nicht vorhanden ist, wird sie von Dreamweaver erstellt.
- Das Argument *new_value* ist ein String, der den Wert der Kategorieoption angibt.

Rückgabewerte
`true`, wenn erfolgreich, andernfalls `false`.

Beispiel

```
var txtEditor = getExternalTextEditor();
dreamweaver.setPreferenceString("My Extension", "Text Editor", txtEditor);
```

dreamweaver.showTargetBrowsersDialog()

Verfügbarkeit
Dreamweaver MX 2004.

Beschreibung
Öffnet das Dialogfeld „Zielbrowser“. Im Dialogfeld „Zielbrowser“ können Benutzer angeben, anhand welcher Browserversionen die Zielbrowser-Prüfungsfunktion die Browserkompatibilität der aktuellen Seite überprüfen soll.

Argumente
Keine.

Rückgabewerte
Keine.

Pfadfunktionen

Mit Pfadfunktionen lassen sich die Pfade zu verschiedenen Dateien und Ordnern auf der Festplatte eines Benutzers abrufen und bearbeiten. Unter anderem kann der Pfad zum Stammverzeichnis der Site des aktuellen Dokuments bestimmt werden und relative Pfade lassen sich in absolute URLs konvertieren.

dreamweaver.absoluteURLToDocRelative()

Verfügbarkeit
Dreamweaver 2.

Beschreibung
Wenn eine absolute URL und der Pfad zu einem Dokument vorliegen, konvertiert diese Funktion die absolute URL in einen Pfad relativ zum Dokument.

Argumente
docPathURL, *siteRootURL*, *absoluteURL*

- Das Argument *docPathURL* ist der Pfad zu einem Dokument auf dem Computer des Benutzers (z. B. das aktuelle Dokument) im URL-Format file://URL.
- Das Argument *siteRootURL* ist der Pfad zum Stamm der Site im URL-Format file://URL.
- Das Argument *absoluteURL* ist die im Format file://URL angegebene URL, die in einen Pfad relativ zum Dokument konvertiert werden soll.

Rückgabewerte
Ein String, der den Pfad zum Dokument unter *absoluteURL* darstellt, angegeben relativ zum Dokument unter *docPathURL*.

Beispiel

Im folgenden Beispiel ist, sofern die Werte von *docPathURL* bzw. *siteRootURL*`file://C:/sites/cherrystreet/archives/october.shtml` bzw. `file://C:/sites/cherrystreet/` lauten, der Rückgabewert `"../includes/header.html"`. Verwenden Sie diesen Wert, um `/includes/header.html` von `/archives/october.shtml` zu referenzieren.

```
var docPathURL = dw.getDocumentDOM().URL;
var siteRootURL = dw.getSiteRoot();
var absoluteURL= dw.relativeToAbsoluteURL(docPathURL, siteRootURL, "/includes/header.html");
var docRelPath = dw.absoluteURLToDocRelative(docPathURL, siteRootURL, absoluteURL);
```

dreamweaver.getConfigurationPath()

Verfügbarkeit

Dreamweaver 2.

Beschreibung

Ruft den Pfad zum Dreamweaver-Ordner „Configuration“ im URL-Format „file://“ ab.

Informationen dazu, wie Dreamweaver in einer Umgebung mit mehreren Benutzern auf den jeweiligen Configuration-Ordner zugreift, finden Sie im Hilfemodul „Dreamweaver erweitern“ unter „C-Level-Erweiterbarkeit“.

Argumente

Keine.

Rückgabewerte

Gibt den Pfad zu den Anwendungskonfigurationen zurück.

Beispiel

Diese Funktion ist dann nützlich, wenn andere Erweiterungsdateien referenziert werden, die alle im Ordner „Configuration“ des Dreamweaver-Anwendungsordners gespeichert sind:

```
var sortCmd = dreamweaver.getConfigurationPath() + ¬
"/Commands/Sort Table.htm"
var sortDOM = dreamweaver.getDocumentDOM(sortCmd);
```

dreamweaver.getDocumentPath()

Verfügbarkeit

Dreamweaver 1.2.

Beschreibung

Ruft den Pfad des angegebenen Dokuments im URL-Format „file://“ ab. Das gleiche Ergebnis erzielen Sie, wenn Sie `dreamweaver.getDocumentDOM()` aufrufen und die Eigenschaft `URL` des Rückgabewerts lesen.

Argumente
sourceDoc

- Der Wert des Arguments*sourceDoc* muss`"document"`, `"parent"`, `"parent.frames[`*`number`*`]"` oder `"parent.frames['`*`frameName`*`']"` sein. `"document"` gibt das Dokument an, das sich im Fokus befindet und die aktuelle Auswahl enthält. `"parent"` gibt das übergeordnete Frameset an (wenn sich das momentan ausgewählte Dokument in einem Frame befindet) und mit `"parent.frames[`*`number`*`]"` und `"parent.frames['`*`frameName`*`']"` wird ein Dokument festgelegt, das sich in einem bestimmten Frame innerhalb des Framesets mit dem aktuellen Dokument befindet.

Rückgabewerte
Wenn die Datei gespeichert wurde, ein String, der die URL des angegebenen Dokuments enthält; wenn die Datei nicht gespeichert wurde, ein leerer String.

dreamweaver.getSiteRoot()

Verfügbarkeit
Dreamweaver 1.2.

Beschreibung
Ruft den lokalen Stammordner (wie im Dialogfeld „Site-Definition“ festgelegt) für die Site des derzeit ausgewählten Dokuments ab, das im URL-Format „file://“ angegeben ist.

Argumente
Keine.

Rückgabewerte
Entweder ein String, der die URL des lokalen Stammordners der Site enthält, in dem die Datei gespeichert wurde, oder ein leerer String, wenn die Datei nicht mit einer Site verknüpft ist.

dreamweaver.getTempFolderPath()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Ruft den vollständigen Pfad eines temporären Ordners ab, in dem Sie temporäre Dateien speichern können. Diese Funktion sucht im Ordner „Configuration“ von Dreamweaver nach einem Ordner „Temp“. Wenn das System mehrere Benutzer unterstützt, sucht es im Ordner „Configuration“ des Benutzers. Ist kein Ordner „Temp“ vorhanden, wird er von der Funktion erstellt. Nicht temporäre gemeinsam genutzte Dateien sollten im Ordner „Configuration/Shared“ gespeichert werden.

Argumente
Keine.

Rückgabewerte
Der vollständige Ordnerpfad im URL-Format „file://“.

Beispiel
Die folgende Codezeile gibt den vollständigen Pfad für die angegebene Datei zurück. Die Funktion `dw.getTempFolderPath()` gibt im Gegensatz zu anderen Dreamweaver-Funktionen (wie z. B. `dreamweaver.getSiteRoot()`) keinen Schrägstrich (/) am Ende des Pfads zurück:

```
var myTempfile = dw.getTempFolderPath() + "/myTempFile.txt";
```

dreamweaver.relativeToAbsoluteURL()

Verfügbarkeit
Dreamweaver 2.

Beschreibung
Wenn eine relative URL und ein Bezugspunkt (Pfad zum aktuellen Dokument oder Sitestamm) vorliegen, konvertiert diese Funktion die relative URL in eine absolute Pfadangabe im URL-Format „file://“.

Argumente
docPath, siteRoot, relURL

- Das Argument *docPath* ist der Pfad zu einem Dokument auf dem Computer des Benutzers (z. B. das aktuelle Dokument) im URL-Format „file://“ oder ein leerer String, wenn *relURL* eine URL relativ zum Stammordner ist.
- Das Argument *siteRoot* ist der Pfad zum Site-Stammordner im URL-Format „file://“ oder ein leerer String, wenn *relURL* eine URL relativ zum Dokument ist.
- Das Argument *relURL* ist die zu konvertierende URL.

Rückgabewerte
Ein absoluter URL-String. Der Rückgabewert wird wie in der folgenden Liste beschrieben generiert:

- Wenn *relURL* eine absolute URL ist, findet keine Konvertierung statt und der Rückgabewert ist identisch mit dem Wert von *relURL*.
- Wenn *relURL* eine URL relativ zu einem Dokument ist, wird der Rückgabewert aus *docPath* und *relURL* zusammengesetzt.
- Wenn *relURL* eine URL relativ zum Stammordner ist, wird der Rückgabewert aus *siteRoot* und *relURL* zusammengesetzt.

DWUri.isValidURI()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Bestimmt, ob das URI-Objekt gültig ist. Das URI-Objekt ist erst gültig, wenn es mit dem gültigen URI erstellt oder initialisiert wurde.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der URI gültig ist.

DWUri.isAbsolute()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Bestimmt, ob das URI-Objekt ein vollständig qualifizierter URI ist. Das URI-Objekt ist erst gültig, wenn es mit dem gültigen URI erstellt oder initialisiert wurde.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der URI vollständig qualifiziert ist.

DWUri.isRelative()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Bestimmt, ob das URI-Objekt ein relativer URI ist. Das URI-Objekt ist erst gültig, wenn es mit dem gültigen URI erstellt oder initialisiert wurde.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der URI relativ ist.

DWUri.isDirectory()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Bestimmt, ob das URI-Objekt ein Verzeichnis ist. Das URI-Objekt ist erst gültig, wenn es mit dem gültigen URI erstellt oder initialisiert wurde.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der URI ein Verzeichnis ist.

DWUri.isHierarchical()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Bestimmt, ob das URI-Objekt hierarchisch ist. Das URI-Objekt ist erst gültig, wenn es mit dem gültigen URI erstellt oder initialisiert wurde.

Hierarchische URI-Objekte verweisen auf eine Ressource mit einer hierarchischen Struktur, deren Hierarchie durchlaufen werden kann, z. B. „`http://somedomain/parts/orders/index.html`“. Nicht hierarchische URI_Objekte können nicht durchlaufen werden, z. B. „`mailto:joe@yahoo.com`„ oder “`about:blank`“.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der URI hierarchisch ist.

DWUri.isOfType()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Bestimmt, ob das URI-Objekt über das angegebene Diensttypschema verfügt. Das URI-Objekt ist erst gültig, wenn es mit dem gültigen URI erstellt oder initialisiert wurde.

Beispiele für Diensttypschemas sind „`http`“, „`file`“ und „`ftp`“. Bei dem URI „`http://www.adobe.com`“ ist „`http`“ das Diensttypschema.

Argumente
type

Das Argument *type* gibt das zu testende Diensttypschema an.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der URI dem angegebenen Diensttypschema entspricht.

DWUri.isOfFileType()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Bestimmt, ob das URI-Objekt auf eine Ressource mit dem angegebenen Dateityp verweist. Das URI-Objekt ist erst gültig, wenn es mit dem gültigen URI erstellt oder initialisiert wurde.

Bei dem URI „`http://www.adobe.com/index.html`“ ist „`html`“ der Dateityp.

Argumente
type

Das Argument *type* gibt das zu testende Diensttypschema an.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der URI auf eine Ressource mit dem angegebenen Dateityp verweist.

DWUri.getScheme()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Ruft das Diensttypschema ab. Das URI-Objekt ist erst gültig, wenn es mit dem gültigen URI erstellt oder initialisiert wurde.

Bei dem URI „`http://www.adobe.com/index.html`“ ist „`http`“ das Diensttypschema.

Argumente
Keine.

Rückgabewerte
String mit dem Diensttypschema des URI-Objekts.

DWUri.getAuthority()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Ruft die Domänenautorität ab. Das URI-Objekt ist erst gültig, wenn es mit dem gültigen URI erstellt oder initialisiert wurde.

Bei dem URI „`http://www.adobe.com/index.html`“ ist „`www.adobe.com`“ die Domänenautorität.

Argumente
Keine.

Rückgabewerte
Ein String mit der Domänenautorität des URI-Objekts.

DWUri.getUsername()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Ruft den Benutzernamen ab. Das URI-Objekt ist erst gültig, wenn es mit dem gültigen URI erstellt oder initialisiert wurde.

Bei dem URI „`ftp://jon@adobe.com`“ ist „`jon`“ der Benutzername.

Argumente
Keine.

Rückgabewerte
Ein String mit dem Benutzernamen des URI-Objekts.

DWUri.getPassword()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Ruft das Kennwort ab. Das URI-Objekt ist erst gültig, wenn es mit dem gültigen URI erstellt oder initialisiert wurde.

Bei dem URI „`ftp://jon:xxx@adobe.com`“ ist „`xxx`“ das Kennwort.

Argumente
Keine.

Rückgabewerte
Ein String mit dem Kennwort des URI-Objekts.

DWUri.getServerPort()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Ruft den Serverport ab. Das URI-Objekt ist erst gültig, wenn es mit dem gültigen URI erstellt oder initialisiert wurde.

Bei dem URI „`http://www.adobe.com:8080`“ ist „`8080`“ der Port.

Argumente
Keine.

Rückgabewerte
Ein String mit dem Serverport des URI-Objekts.

DWUri.getPath()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Ruft den Pfadbereich des Dateinamens ab. Das URI-Objekt ist erst gültig, wenn es mit dem gültigen URI erstellt oder initialisiert wurde.

Bei dem URI „`http://www.adobe.com/Dreamweaver/CS5/index.htm`“ ist „`/Dreamweaver/CS5/`“ der Pfadbereich.

Argumente
Keine.

Rückgabewerte
Ein String mit dem Pfadbereich des URI-Objekts.

DWUri.getQuery()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Ruft den Abfragestring ab. Das URI-Objekt ist erst gültig, wenn es mit dem gültigen URI erstellt oder initialisiert wurde.

Bei dem URI „`http://www.adobe.com/Dreamweaver/CS5/index.htm?q=1502`“ ist „`q=1502`“ der Abfragestring.

Argumente
Keine.

Rückgabewerte
Ein String mit dem Abfragestring des URI-Objekts.

DWUri.getFragment

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Ruft das URI-Ankerfragment ab. Das URI-Objekt ist erst gültig, wenn es mit dem gültigen URI erstellt oder initialisiert wurde.

Bei dem URI „`http://www.adobe.com/Dreamweaver/CS5/index.htm#toc`“ ist „`toc`“ das Ankerfragment.

Argumente
Keine.

Rückgabewerte
Ein String mit dem Ankerfragmentstring des URI-Objekts.

DWUri.getNonHierarchical()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Ruft den nicht hierarchischen URI-String ab. Das URI-Objekt ist erst gültig, wenn es mit dem gültigen URI erstellt oder initialisiert wurde.

Bei dem URI „`mailto:jon@adobe.com`“ ist „`jon@adobe.com`“ der nicht hierarchische Stringteil.

Argumente
Keine.

Rückgabewerte
Ein String mit dem nicht hierarchischen String des URI-Objekts.

DWUri.setScheme()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Legt das Diensttypschema für den URI fest.

Als Diensttypschema kann jeder beliebige Stringwert festgelegt werden. Dieser darf jedoch keine Sonderzeichen wie „:“, „/“ oder „\“ enthalten.

Argumente
scheme

Das Argument *scheme* gibt das Diensttypschema an.

Rückgabewerte
Keine.

DWUri.setAuthority()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Legt die Domänenautorität für den URI fest.

Als Domänenautorität kann jeder beliebige Stringwert festgelegt werden. Dieser darf jedoch keine Sonderzeichen wie „:“, „/“ oder „\“ enthalten.

Argumente
authority

Das Argument *authority* gibt den Namen der Domänenautorität oder die IP-Adresse an.

Rückgabewerte
Keine.

DWUri.setUsername()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Legt den Benutzernamen des URI fest.

In der Regel werden Benutzernamen nur für die FTP-Diensttypschemas verwendet.

Argumente
username

Das Argument gibt den Benutzernamen an.

Rückgabewerte
Keine.

DWUri.setPassword()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Legt das Kennwort des URI fest.

In der Regel werden Kennwörter nur für die FTP-Diensttypschemas verwendet.

Argumente
password

Das Argument gibt das Kennwort an.

Rückgabewerte
Keine.

DWUri.setPath()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Legt den Pfadbereich des URI-Dateinamens fest.

Sie können als Pfad einen leeren String oder „/“ festlegen, um als Pfad den Verzeichnisstamm anzugeben.

Argumente
path

Das Argument gibt den Pfadbereich des Dateinamens an.

Rückgabewerte
Keine.

DWUri.setServerPort()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Legt den Port des URI-Objekts fest.

Jedes Diensttypschema verfügt über einen durch die IETF-Normen vorgegebenen Standardport. Verwenden Sie diese Funktion, um dem URI einen nicht standardmäßigen Port hinzuzufügen. Für den Port muss ein numerischer Wert zwischen 1 und 65535 angegeben werden.

Argumente
port

Das Argument gibt den Serverport an.

Rückgabewerte
Keine.

DWUri.setQuery()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Legt den Abfragestring des URI-Objekts fest.

Mithilfe der Funktion „DWUri.setQueryValue()“ auf Seite 296 können Sie den Wert eines einzelnen Parameters ändern.

Argumente
query

Das Argument gibt den Abfragestring an.

Rückgabewerte
Keine.

DWUri.setFragment()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Legt den Ankerfragmentstring des URI-Objekts fest.

Argumente
anchor

Das Argument gibt den Ankernamen an.

Rückgabewerte
Keine.

DWUri.setNonHierarchical()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Legt den nicht hierarchischen String des URI-Objekts fest. Durch Aufrufen dieser Funktion werden die hierarchischen Attribute des URI gelöscht.

Argumente
nonHierarchical

Das Argument gibt den nicht hierarchischen String an.

Rückgabewerte
Keine.

DWUri.getQueryValue()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Ruft den Wert ab, der dem als Argument übergebenen URI-Namen zugeordnet ist.

Für den URI „`http://www.adobe.com/Dreamweaver/CS5/index.htm?q=1502`“ gibt `getQuery("q")` den Wert „`1502`“ zurück. Wenn ein Wert für einen nicht vorhandenen Namen angefordert wird, wird ein leerer String zurückgegeben.

Argumente
name

Das Argument gibt den Namen in der Abfrage an.

Rückgabewerte
Ein Stringwert, der den Wert des Namens in der Abfrage darstellt.

DWUri.setQueryValue()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Legt den Namen und den Wert fest, die dem Abfragestring des URI zugeordnet sind. Durch Festlegen eines leeren Strings als Wert wird der Name aus dem Abfragestring entfernt.

Argumente
name

Das Argument gibt den Namen in der Abfrage an.

value

Das Argument gibt den Wert des Namens in der Abfrage an.

Rückgabewerte
Keine.

DWUri.getQueryByObject()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Ruft das Abfrageeigenschaftenobjekt des URI-Objekts ab.

Änderungen an der Eigenschaftenzuordnung werden im URI-Objekt erst berücksichtigt, wenn Sie die Funktion „DWUri.setQueryByObject()“ auf Seite 297 aufrufen.

Argumente
Keine.

Rückgabewerte
Objekt

Die Eigenschaftenzuordnung des URI.

DWUri.setQueryByObject()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Ersetzt die gesamte Eigenschaftenzuordnung des URI-Objekts durch die angegebene Zuordnung.

Beispiel:

```
function main{
    var referrer = new Object;
    referrer.page = "index.html";
    referrer.user = "jon";
    DWUri uri = new DWUri;
    uri.setQueryByObject ( referrer );
}
```

Argumente
objectMap

Die Eigenschaftenzuordnung des URI.

Rückgabewerte
Keine.

DWUri.getRelation()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Bestimmt die Beziehung zwischen 2 URIs.

Argumente
other

Der URI, mit dem der Vergleich durchgeführt werden soll. Geben Sie einen gültigen String oder ein `DWUri`-Objekt an.

Rückgabewerte
Ganzzahl, die die Wertkonstante angibt. Folgende Konstanten sind möglich:

- `DWUri.NOT_REALTED`
- `DWUri.CHILD`
- `DWUri.EQUAL`

- `DWUri.PARENT`

DWUri.getCommonParent()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Bestimmt das gemeinsame übergeordnete Element der 2 URIs.

Wenn kein gemeinsames übergeordnetes Element vorhanden ist, gibt diese Funktion ein leeres DWUri-Objekt zurück, das durch Aufrufen der Funktion „DWUri.isValidURI()“ auf Seite 286 validiert werden sollte.

Argumente
other

Der URI, mit dem der Vergleich durchgeführt werden soll. Geben Sie einen gültigen String oder ein `DWUri`-Objekt an.

Rückgabewerte
Objekt

Ein `DWUri`-Objekt, das das gemeinsame übergeordnete Element darstellt.

DWUri.makeAbsolute()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Erstellt einen vollständig qualifizierten URI aus dem als Argument übergebenen URI.

Argumente
other

Geben Sie einen gültigen String oder ein `DWUri`-Objekt an.

Rückgabewerte
Ein boolescher Wert: `true` bei einem erfolgreichen Vorgang.

DWUri.makeRelative()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Erstellt durch Suchen des gemeinsamen übergeordneten Elements einen relativen URI aus dem als Argument übergebenen URI.

Argumente
other

Geben Sie einen gültigen String oder ein `DWUri`-Objekt an.

Rückgabewerte
Ein boolescher Wert: `true` bei einem erfolgreichen Vorgang.

DWUri.chDir()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Wechselt in das angegebene Verzeichnis. Der String „. .“ wird für das Wechseln in das übergeordnete Verzeichnis verwendet.

Argumente
dir

Geben Sie einen gültigen String als Verzeichnis an.

Rückgabewerte
Ein boolescher Wert: `true` bei einem erfolgreichen Vorgang.

DWUri.getFileName()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Ruft den Dateinamen aus dem URI-Objekt ab.

Argumente
stripExtension

Ein boolescher Wert: `true` zum Entfernen der Erweiterung aus dem Ergebnis. Geben Sie `false` an, damit der Dateinamen mit Erweiterung zurückgegeben wird. Das Standardverhalten ist, den Dateinamen mit der Erweiterung zurückzugeben. Dieses Argument ist optional.

Rückgabewerte
Ein String, der den Dateinamen angibt.

DWUri.getExtension()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Ruft die Erweiterung aus dem URI-Objekt ab.

Argumente
stripDot

Ein boolescher Wert: `true` zum Entfernen des vorangestellten Punkts aus dem Ergebnis. Geben Sie „false“ an, damit der vorangestellte Punkt zurückgegeben wird. Das Standardverhalten ist, die Erweiterung mit dem vorangestellten Punkt zurückzugeben. Dieses Argument ist optional.

Rückgabewerte
Ein String, der die Erweiterung angibt.

DWUri.getLastPathComponent()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Ruft den letzten Teil der Pfadkomponente im Dateinamen ab.

Argumente
Keine.

Rückgabewerte
Ein String, der den letzten Pfadbereich angibt.

Für den URI „`http://www.adobe.com/Dreamweaver/CS5/index.htm`“ gibt `getLastPathComponent()` den String „`CS5`“ zurück.

DWUri.removeLastPathComponent()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Entfernt den letzten Teil der Pfadkomponente im Dateinamen.

Für den URI „`http://www.adobe.com/Dreamweaver/CS5/index.htm`“ gibt `removeLastPathComponent()` den String „`CS5`“ zurück und entfernt diesen String aus dem URI.

Argumente
Keine.

Rückgabewerte
Ein String, der den letzten Pfadbereich angibt.

DWUri.isUnderDirectory()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Bestimmt, ob der URI einem angegebenen übergeordneten URI untergeordnet ist.

Beachten Sie, dass beim Vergleich von Verzeichnisnamen immer die Groß- und Kleinschreibung berücksichtigt wird.

Argumente
Ein String, der den übergeordneten URI darstellt.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der URI dem angegebenen übergeordneten URI untergeordnet ist.

DWUri.toLocalPath()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Wandelt einen URI mit dem Diensttypschema „file“ in einen mit dem Dateisystem kompatiblen String um.

Der Rückgabewert ist ein plattformspezifischer String. Dateinamen werden auf jeder Plattform unterschiedlich angegeben.

Argumente
Keine.

Rückgabewerte
Ein String, der einen Dateisystempfad darstellt, über den mithilfe der APIs auf Systemebene Dateien geöffnet werden können. Dieser String kann als Argument an andere `DWFile`-Funktionen übergeben werden.

DWUri.localPathToURI()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Wandelt einen lokalen Dateipfad in ein URI-Objekt um.

Argumente
Ein String, der den lokalen Dateinamen angibt, der als URI kodiert werden muss.

Rückgabewerte
Keine.

Auswahlfunktionen

Mit Auswahlfunktionen werden in geöffneten Dokumenten ausgewählte Bereiche abgerufen und festgelegt. Informationen über das Abrufen und Einstellen der Auswahl im Bedienfeld „Dateien“ finden Sie unter „Site-Funktionen“ auf Seite 226.

dom.getSelectedNode()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Ruft den ausgewählten Knoten ab. Das gleiche Ergebnis erzielen Sie, wenn Sie die Funktion `dom.getSelection()` aufrufen und den Rückgabewert an die Funktion `dom.offsetsToNode()` übergeben.

Argumente
Keine.

Rückgabewerte
Das Objekt vom Typ Tag, Text oder Kommentar, das den angegebenen Zeichenbereich vollständig enthält.

dom.getSelection()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Ruft die Auswahl ab, die in Form von Zeichen-Offsets im Quellcode des Dokuments angegeben wird.

Argumente
{bAllowMultiple}

- Das optionale Argument *bAllowMultiple* ist ein boolescher Wert, der angibt, ob die Funktion mehrere Offsets zurückgeben soll, wenn mehrere Tabellenzellen, Imagemap-Hotspots oder Ebenen ausgewählt sind.

  Bei fehlendem Argument gilt der Standardwert `false`.

Rückgabewerte
Bei einer einfachen Auswahl ein Array mit zwei Ganzzahlen. Die erste Ganzzahl ist der Zeichen-Offset zum Anfang der Auswahl. Die zweite Ganzzahl ist der Zeichen-Offset zum Ende der Auswahl. Wenn beide Zahlen übereinstimmen, handelt es sich bei der aktuellen Auswahl um eine Einfügemarke.

Bei einer komplexen Auswahl (mehrere Tabellenzellen, Ebenen oder Imagemap-Hotspots) ein Array mit 2*n* Ganzzahlen, wobei *n* die Anzahl der ausgewählten Elemente bezeichnet. Die erste Ganzzahl in jedem Wertepaar ist der Zeichen-Offset zum Anfang der Auswahl (einschließlich des öffnenden Tags vom Typ `TD`, `DIV`, `SPAN`, `LAYER`, `ILAYER` oder `MAP`). Die zweite Ganzzahl ist der Zeichen-Offset zum Ende der Auswahl (einschließlich der schließenden Tags vom Typ `TD`, `DIV`, `SPAN`, `LAYER`, `ILAYER` oder `MAP`). Wenn mehrere Tabellenzeilen ausgewählt sind, werden die Offsets aller Zellen in den einzelnen Zeilen zurückgegeben. Die Auswahl schließt niemals `TR`-Tags ein.

dom.getSelectorsDefinedInStylesheet()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Ruft ein Array von Selektoren ab, die mit dem als Attribut übergebenen Typ übereinstimmen.

Argumente
`selector`

- Das Argument *`selector`* ist ein String mit dem Wert`class` oder `ID`. Es gibt an, ob die Funktion Selektoren des Typs `class` oder `ID` zurückgibt.

Rückgabewerte
Ein Array von Selektoren des Typs `class` oder `ID`.

Beispiel
Mit dem folgenden Code wird ein Array von Selektoren des Typs `class` abgerufen:

```
var dom=dw.getDocumentDOM();
var classSelectors = dom.getSelectorsDefinedInStylesheet('class');
```

Mit dem folgenden Code wird ein Array von Selektoren des Typs `ID` abgerufen:

```
var dom=dw.getDocumentDOM();
var classSelectors = dom.getSelectorsDefinedInStylesheet('ID');
```

dom.nodeToOffsets()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Ruft die Position eines bestimmten Knotens in der DOM-Struktur ab, angegeben in Form von Zeichen-Offsets im Quellcode des Dokuments. Gültig für beliebige Dokumente auf einem lokalen Laufwerk.

Argumente
node

- Das Argument *node* muss ein Objekt des Typs Tag, Kommentar oder Textbereich sein, das einem Knoten in der von `dreamweaver.getDocumentDOM()` zurückgegebenen Struktur entspricht.

Rückgabewerte
Ein Array mit zwei Ganzzahlen. Die erste Ganzzahl ist der Zeichen-Offset zum Anfang des Tags, Texts oder Kommentars. Die zweite Ganzzahl ist der Zeichen-Offset zum Ende des Knotens, vom Anfang des HTML-Dokuments.

Beispiel
Mit dem folgenden Code wird das erste Bildobjekt im aktuellen Dokument ausgewählt:

```
var theDOM = dw.getDocumentDOM();
var theImg = theDOM.images[0];
var offsets = theDom.nodeToOffsets(theImg);
theDom.setSelection(offsets[0], offsets[1]);
```

dom.offsetsToNode()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Ruft das Objekt in der DOM-Struktur ab, das den vollständigen Zeichenbereich zwischen den angegebenen Anfangs- und Endpunkten enthält. Gültig für beliebige Dokumente auf einem lokalen Laufwerk.

Argumente
offsetBegin, offsetEnd

- Das Argument *offsetBegin* gibt den Offset vom Anfang des Dokuments bis zum Anfang des Zeichenbereichs an, der einem Objekt in der DOM-Struktur entspricht.
- Das Argument *offsetEnd* gibt den Offset vom Anfang des Dokuments bis zum Ende des Zeichenbereichs an, der einem Objekt in der DOM-Struktur entspricht.

Rückgabewerte
Das Objekt vom Typ Tag, Text oder Kommentar, das den angegebenen Zeichenbereich vollständig enthält.

Beispiel
Beim folgenden Code wird eine Warnmeldung angezeigt, wenn es sich bei der Auswahl um ein Bild handelt:

```
var offsets = dom.getSelection();
var theSelection = dreamweaver.offsetsToNode(offsets[0], ¬
offsets[1]);
if (theSelection.nodeType == Node.ELEMENT_NODE && ¬
theSelection.tagName == 'IMG'){
    alert('The current selection is an image.');
}
```

dom.selectAll()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Führt einen Vorgang des Typs „Alles auswählen“ aus.

Hinweis: *Im Regelfall wird mit dieser Funktion der gesamte Inhalt des aktiven Dokuments ausgewählt. In manchen Fällen (z. B. wenn sich die Einfügemarke in einer Tabelle befindet) wird nur ein Teil des aktiven Dokuments ausgewählt. Um bei der Auswahl das gesamte Dokument zu erfassen, verwenden Sie die Funktion* `dom.setSelection()`.

Argumente
Keine.

Rückgabewerte
Keine.

dom.setSelectedNode()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Setzt den ausgewählten Knoten. Das gleiche Ergebnis erzielen Sie, wenn Sie die Funktion `dom.nodeToOffsets()` aufrufen und den Rückgabewert an die Funktion `dom.setSelection()` übergeben.

Argumente
node, {bSelectInside}, {bJumpToNode}

- Das Argument *node* ist ein Text-, Kommentar- oder Elementknoten im Dokument.
- Das optionale Argument *bSelectInside* ist ein boolescher Wert, der angibt, ob sich die Auswahl auch auf `innerHTML` des Knotens erstrecken soll. Dieses Argument ist nur dann relevant, wenn *node* ein Elementknoten ist. Bei fehlendem Argument gilt der Standardwert `false`.
- Das optionale Argument *bJumpToNode* ist ein boolescher Wert, der angibt, ob ein Bildlauf im Dokumentfenster durchgeführt werden soll, damit die Auswahl sichtbar wird. Bei fehlendem Argument gilt der Standardwert `false`.

Rückgabewerte
Keine.

dom.setSelection()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Legt die Auswahl im Dokument fest.

Argumente
offsetBegin, offsetEnd

- Als Argumente werden der Anfangs- und der Endpunkt für die neue Auswahl übergeben, angegeben in Form von Zeichen-Offsets im Quellcode des Dokuments. Wenn beide Zahlen übereinstimmen, handelt es sich bei der neuen Auswahl um eine Einfügemarke. Wenn die neue Auswahl keine gültige HTML-Auswahl darstellt, wird sie durch Einbeziehung weiterer Zeichen zur ersten gültigen HTML-Auswahl erweitert. Wenn beispielsweise mit *offsetBegin* und *offsetEnd* der Bereich `SRC="myImage.gif"` innerhalb von `<IMG SRC="myImage.gif">` definiert wird, wird die Auswahl so erweitert, dass das vollständige `IMG`-Tag enthalten ist.

Rückgabewerte
Keine.

dreamweaver.nodeExists()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Bestimmt, ob der Verweis auf den angegebenen Knoten noch gültig ist. Beim Programmieren von Erweiterungen kommt es häufig vor, dass Sie auf einen Knoten verweisen und dann einen Vorgang ausführen, der diesen Knoten löscht (z. B. durch Festlegen der Eigenschaft `innerHTML` oder `outerHTML` des übergeordneten Knotens). Mit dieser Funktion können Sie sich davon überzeugen, dass der Knoten nicht gelöscht wurde, bevor Sie versuchen, auf eine seiner Eigenschaften oder Methoden zu verweisen. Der referenzierte Knoten muss sich nicht im aktuellen Dokument befinden.

Argumente
node

- Das Argument *node* ist der Name des zu überprüfenden Knotens.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der Knoten vorhanden ist, andernfalls `false`.

Beispiel
Im folgenden Beispiel wird der aktuelle Knoten abgerufen, die darin enthaltene Tabelle gesucht und später `dw.nodeExists()` aufgerufen, um zu überprüfen, ob der ursprüngliche Knoten noch vorhanden ist:

```
function applyFormatToSelectedTable(){

    // get current selection
    var selObj = dw.getDocumentDOM().getSelectedNode();
    alternateRows(dwscripts.findDOMObject("presetNames").selectedIndex,
        findTable());

// restore original selection, if it still exists; if not, just select the
// table.
    var selArr;
    if (dw.nodeExists(selObj))
        selArr = dom.nodeToOffsets(selObj);
    else
        selArr = dom.nodeToOffsets(findTable());

    dom.setSelection(selArr[0],selArr[1]);
}
```

dreamweaver.selectAll()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Führt im aktiven Dokumentfenster, im Bedienfeld „Dateien“ bzw. auf dem Macintosh für das aktive Textfeld in einem Dialogfeld oder schwebenden Bedienfeld einen Vorgang des Typs „Alles auswählen“ aus.

Hinweis: *Wenn der Vorgang im aktiven Dokument durchgeführt wird, wird im Regelfall der gesamte Inhalt des aktiven Dokuments ausgewählt. In manchen Fällen (z. B. wenn sich die Einfügemarke in einer Tabelle befindet) wird jedoch nur ein Teil des aktiven Dokuments ausgewählt. Um bei der Auswahl das gesamte Dokument zu erfassen, verwenden Sie die Funktion* `dom.setSelection()`.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dreamweaver.canSelectAll()“ auf Seite 524.

Stringbearbeitungsfunktionen

Mit den Stringbearbeitungsfunktionen können Sie Informationen über Strings abrufen sowie Strings von der Latin 1-Kodierung in plattformspezifische Kodierungen konvertieren und umgekehrt.

dreamweaver.doURLEncoding()

Verfügbarkeit
Dreamweaver 1.

Beschreibung
Diese Funktion konvertiert einen String und gibt einen als URL kodierten String zurück. Dazu werden alle Leerzeichen und Sonderzeichen durch angegebene Elemente ersetzt.

Argumente
stringToConvert

- Das Argument *stringToConvert* ist ein String mit der unkodierten URL, die von der Funktion kodiert wird.

Rückgabewerte
Ein als URL kodierter String.

Beispiel
Das folgende Beispiel zeigt den Wert `URL.value` für `"My URL-encoded string"`:

```
var URL = dw.doURLEncoding(theURL.value);
returns "My%20URL-encoded%20string"
```

dreamweaver.getTokens()

Verfügbarkeit
Dreamweaver 1.

Beschreibung
Teilt einen String in Token auf.

Argumente
searchString, separatorCharacters

- Das Argument *searchString* ist der String, der in Token aufgeteilt werden soll.
- Das Argument *separatorCharacters* ist das Zeichen (bzw. mehrere Zeichen), das das Ende eines Tokens darstellt. Trennzeichen in Strings, die zwischen Anführungszeichen stehen, werden ignoriert. Leerraumzeichen in `separatorCharacters` (z. B. Tabstopp-Zeichen) werden als Trennzeichen behandelt, als ob sie explizit definiert worden wären. Zwei oder mehr aufeinander folgende Leerraumzeichen werden als einfaches Trennzeichen behandelt.

Rückgabewerte
Ein Array von Token-Strings.

Beispiel
Der folgende Aufruf der Funktion `dw.getTokens()` gibt die Token zurück, die darauf folgen:

```
dreamweaver.getTokens('foo("my arg1", 34)', '(),')
```

- `foo`
- `"my arg 1"`
- `34`

dreamweaver.latin1ToNative()

Verfügbarkeit
Dreamweaver 2.

Beschreibung
Konvertiert einen in Latin 1-Kodierung vorliegenden String in die plattformspezifische Kodierung des jeweiligen Computers. Diese Funktion dient dazu, die Benutzerschnittstelle einer Erweiterungsdatei in einer anderen Sprache anzuzeigen.

Hinweis: *Unter Windows hat dies keine Auswirkung, da Kodierungen in Windows ohnehin bereits auf Latin 1 basieren.*

Argumente
stringToConvert

- Das Argument *stringToConvert* ist der String, der von der Latin 1-Kodierung in die plattformspezifische Kodierung konvertiert werden soll.

Rückgabewerte
Der konvertierte String.

dreamweaver.nativeToLatin1()

Verfügbarkeit
Dreamweaver 2.

Beschreibung
Konvertiert einen in der plattformspezifischen Kodierung vorliegenden String in die Latin 1-Kodierung.

Hinweis: *Unter Windows hat dies keine Auswirkung, da Kodierungen in Windows ohnehin bereits auf Latin 1 basieren.*

Argumente
stringToConvert

- Das Argument *stringToConvert* ist der String, der von der plattformspezifischen Kodierung in die Latin 1-Kodierung konvertiert werden soll.

Rückgabewerte
Der konvertierte String.

dreamweaver.scanSourceString()

Verfügbarkeit
Dreamweaver UltraDev 1.

Beschreibung
Analysiert einen HTML-String und ermittelt die Tags, Attribute, Direktiven und Text. Die Funktion `scanSourceString()` startet für alle gefundenen Tags, Attribute, Direktiven und Textbereiche eine Callback-Funktion, die Sie bereitstellen müssen. Dreamweaver unterstützt folgende Callback-Funktionen:

- `openTagBegin()`
- `openTagEnd()`
- `closeTagBegin()`
- `closeTagEnd()`
- `directive()`
- `attribute()`
- `text()`

Dreamweaver ruft die sieben Callback-Funktionen in folgenden Fällen auf:

- Dreamweaver ruft `openTagBegin()` für jedes öffnende Tag (z. B. `<font>`, nicht jedoch `</font>`) und jedes leere Tag (z. B. `<img>` oder `<hr>`) auf. Die Funktion `openTagBegin()` akzeptiert zwei Argumente: den Namen des Tags (z. B.`"font"` oder `"img"`) und den Dokument-Offset. Beim Dokument-Offset handelt es sich um die Anzahl der Byte, die im Dokument vor dem Anfang des Tags stehen. Die Funktion gibt `true` zurück, wenn die Analyse fortgesetzt werden soll, und `false`, wenn die Analyse abgebrochen werden soll.

- Nach der Ausführung von `openTagBegin()` ruft Dreamweaver für jedes HTML-Attribut die Funktion `attribute()` auf. Die Funktion `attribute()` akzeptiert zwei Argumente: einen String mit dem Namen des Attributs (z. B. `"color"` oder `"src"`) und einen String mit dem Wert des Attributs (z. B.`"#000000"` oder `"foo.gif"`). Die Funktion `attribute()` gibt einen booleschen Wert zurück, der angibt, ob die Analyse fortgesetzt werden soll.
- Nachdem alle im Tag enthaltenen Attribute analysiert wurden, ruft Dreamweaver die Funktion `openTagEnd()` auf. Die Funktion `openTagEnd()` akzeptiert ein Argument: den Dokument-Offset. Beim Dokument-Offset handelt es sich um die Anzahl der Byte, die im Dokument vor dem Ende des öffnenden Tags stehen. Sie gibt einen booleschen Wert zurück, der angibt, ob die Analyse fortgesetzt werden soll.
- Dreamweaver ruft `closeTagBegin()` für jedes schließende Tag auf (z. B.`</font>`). Diese Funktion akzeptiert zwei Argumente: den Namen des zu schließenden Tags (z. B. `"font"`) und den Dokument-Offset. Beim Dokument-Offset handelt es sich um die Anzahl der Byte, die im Dokument vor dem Anfang des schließenden Tags stehen. Die Funktion gibt einen booleschen Wert zurück, der angibt, ob die Analyse fortgesetzt werden soll.
- Nach der Rückgabe der Funktion `closeTagBegin()` ruft Dreamweaver `closeTagEnd()` auf. Die Funktion `closeTagEnd()` akzeptiert ein Argument: den Dokument-Offset. Beim Dokument-Offset handelt es sich um die Anzahl der Byte, die im Dokument vor dem Ende des schließenden Tags stehen. Sie gibt einen booleschen Wert zurück, der angibt, ob die Analyse fortgesetzt werden soll.
- Dreamweaver ruft die Funktion `directive()` für alle HTML-Kommentare sowie ASP-, JSP- und PHP-Skripts auf. Die Funktion `directive()` akzeptiert zwei Argumente: einen String mit der Direktive und den Dokument-Offset. Beim Dokument-Offset handelt es sich um die Anzahl der Byte, die im Dokument vor dem Ende des End-Tags stehen. Die Funktion gibt einen booleschen Wert zurück, der angibt, ob die Analyse fortgesetzt werden soll.
- Dreamweaver ruft die Funktion `text()` für alle Textbereiche im Dokument auf, das heißt für alle Bereiche, bei denen es sich weder um Tags noch um Direktiven handelt. Zu Textbereichen gehört Text, der für den Benutzer nicht sichtbar ist, z. B. Text innerhalb eines `<title>`- oder `<option>`-Tags. Die Funktion `text()` akzeptiert zwei Argumente: einen String mit dem Text und den Dokument-Offset. Beim Dokument-Offset handelt es sich um die Anzahl der Byte, die im Dokument vor dem Ende des End-Tags stehen. Die Funktion `text()` gibt einen booleschen Wert zurück, der angibt, ob die Analyse fortgesetzt werden soll.

Argumente
HTMLstr, parserCallbackObj

- Das Argument *HTMLstr* ist ein String mit Code.
- Das Argument *parserCallbackObj* ist ein JavaScript-Objekt mit einer oder mehreren der folgenden Methoden: `openTagBegin()`, `openTagEnd()`, `closeTagBegin()`, `closeTagEnd()`, `directive()`, `attribute()` und `text()`. Damit eine optimale Leistung erzielt werden kann, sollte es sich bei *parserCallbackObj* um eine freigegebene Bibliothek handeln, die mit der Schnittstelle für die C-Level-Erweiterbarkeit definiert wurde. Die Leistung wird auch verbessert, wenn *parserCallbackObj* nur die tatsächlich benötigten Callback-Funktionen definiert.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der Vorgang erfolgreich war, andernfalls `false`.

Beispiel
Im folgenden Beispiel wird die Verwendung der Funktion `dreamweaver.scanSourceString()` Schritt für Schritt erläutert:

1 Erstellen Sie eine Implementierung für eine oder mehrere der sieben Callback-Funktionen.

2 Verfassen Sie ein Skript, das die Funktion `dreamweaver.scanSourceString()` aufruft.

3 Die Funktion `dreamweaver.scanSourceString()` übergibt einen String mit HTML-Code und Zeigern zu den Callback-Funktionen, die Sie geschrieben haben. Der HTML-String lautet beispielsweise`"<font size=2>hello</font>"`.

4 Dreamweaver analysiert diesen String und stellt fest, dass er ein font-Tag enthält. Daraufhin ruft Dreamweaver die Callback-Funktionen in folgender Reihenfolge auf:

- Funktion `openTagBegin()`
- Funktion `attribute()` (für das size-Attribut)
- Funktion `openTagEnd()`
- Funktion `text()` (für den String `"hello"`)
- Funktion `closeTagBegin()` und `closeTagEnd()`

Übersetzungsfunktionen

Übersetzungsfunktionen werden entweder direkt auf Übersetzer oder auf die Ergebnisse von Übersetzungen angewendet. Es lassen sich Informationen über Übersetzer abrufen, Übersetzer ausführen und Inhalte von gesperrten Bereichen bearbeiten. Darüber hinaus kann festgelegt werden, dass beim Abrufen und Einstellen von Auswahl-Offsets die übersetzte Quelle verwendet werden soll.

dom.runTranslator()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion führt den angegebenen Übersetzer für das Dokument aus. Diese Funktion ist nur für das aktive Dokument gültig.

Argumente
translatorName

- Das Argument *translatorName* ist der Name des Übersetzers, wie er in den Voreinstellungen für die Übersetzung angegeben ist.

Rückgabewerte
Keine.

dreamweaver.editLockedRegions()

Verfügbarkeit
Dreamweaver 2.

Beschreibung
Macht gesperrte Bereiche bearbeitbar bzw. nicht bearbeitbar, je nach dem Wert des Arguments. Standardmäßig sind gesperrte Bereiche nicht bearbeitbar. Wenn Sie versuchen, einen gesperrten Bereich zu bearbeiten, ohne ihn zuvor ausdrücklich mit dieser Funktion als bearbeitbar gekennzeichnet zu haben, wird ein Warnton ausgegeben und die Änderung wird nicht akzeptiert.

Hinweis: *Das Bearbeiten gesperrter Bereiche kann ungewollte Folgen für Bibliothekselemente und Vorlagen haben. Sie sollten diese Funktion daher nur im Zusammenhang mit Datenübersetzern verwenden.*

Argumente
bAllowEdits

- Das Argument *bAllowEdits* ist ein boolescher Wert: `true` wenn die Bearbeitung zulässig ist, andernfalls `false`. Für gesperrte Bereiche wird nach Beendigung des aufrufenden Skripts automatisch wieder der Standardstatus (nicht bearbeitbar) hergestellt.

Rückgabewerte
Keine.

dreamweaver.getTranslatorList()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion ruft eine Liste der installierten Übersetzer ab.

Argumente
Keine.

Rückgabewerte
Ein Array von Strings, die die Namen der Übersetzer enthalten, wie sie in den Übersetzungsvoreinstellungen angezeigt werden.

dreamweaver.useTranslatedSource()

Verfügbarkeit
Dreamweaver 2.

Beschreibung
Legt die Werte fest, die von `dom.nodeToOffsets()` und `dom.getSelection()` zurückgegeben werden. Diese Werte werden von `dom.offsetsToNode()` und `dom.setSelection()` verwendet und stellen Offsets in den übersetzten Quellcode dar (der im DOM nach der Ausführung eines Übersetzers enthaltene HTML-Code), nicht für den nicht übersetzten Quellcode.

Hinweis: *Diese Funktion ist nur bei Eigenschafteninspektor-Dateien relevant.*

Argumente
bUseTranslatedSource

- Das Argument *bUseTranslatedSource* ist ein boolescher Wert. Bei `true` verwendet die Funktion Offsets in den übersetzten Quellcode und bei `false` verwendet sie den nicht übersetzten Quellcode.

  Der Standardwert für das Argument lautet `false`. Für nachfolgende Aufrufe von `dw.getSelection()`, `dw.setSelection()`, `dw.nodeToOffsets()` und `dw.offsetsToNode()` wird nach Beendigung des Skripts, das `dw.useTranslatedSource()` aufgerufen hat, automatisch der unübersetzte Quellcode verwendet, falls `dw.useTranslatedSource()` nicht bereits zuvor explizit mit dem Argument `false` aufgerufen wurde.

Rückgabewerte
Keine.

XSLT-Funktionen

XSLT-Funktionen gelten für XML-Dateien. Diese Funktionen rufen Informationen über XML-Dokumente ab, einschließlich der Schemastruktur oder des Verweises auf ein XML-Dokument, und fordern den Benutzer auf, das mit dem aktuellen XSLT-Dokument verbundene XML-Dokument anzugeben.

MMXSLT.getXML()

Verfügbarkeit
Dreamweaver CS3.

Beschreibung
Ruft einen XML-Quellstring für eine XML-Datei ab.

Argumente
xmlSourceURI

- Ein String, der einen URI zu einer XML-Datei darstellt. Dieser kann absolut (http oder https), relativ zu einer Site oder relativ zu einem Dokument sein.

Rückgabewerte
Ein String, der die Inhalte der XML-Datei enthält.

Beispiel
```
var xmlSource = MMXSLT.getXML(this.fileDataSetURL);
```

MMXSLT.getXMLSchema()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion gibt die Schemastruktur der angegebenen XML-Datei zurück.

Argumente
schemaURI, {bRefresh}

- Das erforderliche Argument *schemaURI* ist ein String, der eine Referenz auf eine lokale oder Remote-XML-Datei darstellt.
- Das optionale Argument *bRefresh* ist ein boolescher Wert. Bei `true` wird eine Aktualisierung des Schemas erzwungen. Bei `false` wird die Kopie des Schemas aus dem XML-Schema-Cache zurückgegeben. Der Standardwert ist `false`.

Rückgabewerte
Ein String, der die XML-Schemastruktur enthält.

Beispiel
Im folgenden Beispiel wird die Schemastruktur aus dem XML-Schema-Cache für „menus.xml“ abgerufen:

```
var theSchema = MMXSLT.getXMLSchema("file:///c:/Program Files/Adobe/¬
    Adobe Dreamweaver CS5/Configuration/Menus/menus.xml"");
```

MMXSLT.getXMLSourceURI()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion ruft eine Referenz auf das mit dem aktuellen XSLT-Dokument verbundene XML-Quelldokument ab.

Argumente
xsltfileURI, {bUseTempForRemote}

- Das Argument *xsltfileURI* ist ein String mit dem lokalen Datei-URI, der auf den Speicherort der XSL-Datei zeigt.
- Das optionale Argument *bUseTempForRemote* ist ein boolescher Wert: `true` gibt eine Referenz auf die temporäre XML-Datei zurück (z. B. `file:///C:/Dokumente und Einstellungen/Benutzername/Lokale Einstellungen/Temporary Internet Files/Content.IE5/GTSLQ9KZ/rss[1].xml`), die heruntergeladen wird, wenn die ursprüngliche XML-Datei remote ist (z. B. `http://myHost/rssfeed.xml`). Beim Wert `false` wird eine absolute Referenz zurückgegeben.

Rückgabewerte
Ein String, der eine Referenz auf das mit dem aktuellen XSLT-Dokument verbundene XML-Quelldokument enthält. Handelt es sich bei der XML-Quellreferenz um eine Remote-Referenz, gibt die Funktion den heruntergeladenen Dateipfad zum temporären Speicherort zurück.

Beispiel
Im folgenden Beispiel wird die Referenz auf das mit „c:\myxslt\myxsltdocument.xsl“ verbundene XML-Quelldokument abgerufen:

```
var theXMLSource = MMXSLT.getXMLSourceURI("file:///c:/myxslt/myxsltdocument.xsl");
```

MMXSLT.launchXMLSourceDialog()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion fordert den Benutzer auf, das mit dem aktuellen XSLT-Dokument verbundene XML-Quelldokument anzugeben. Der Benutzer kann entweder eine lokale oder eine Remote-Referenz auf ein XML-Dokument angeben.

Argumente
{xsltfileURI}, {bUseTempForRemote}, {bAddSchemaReference}

- Das Argument *xsltfileURI* ist optional. Es ist ein String, der den lokalen Datei-URI bezeichnet, der auf den Speicherort der XSL-Datei zeigt. Bei fehlendem Argument gilt als Standardwert das aktuell geöffnete Dokument.
- Das optionale Argument *bUseTempForRemote* ist ein boolescher Wert: `true` gibt eine Referenz auf die temporäre XML-Datei zurück (z. B. `file:///C:/Dokumente und Einstellungen/Benutzername/Lokale Einstellungen/Temporary Internet Files/Content.IE5/GTSLQ9KZ/rss[1].xml`), die heruntergeladen wird, wenn die ursprüngliche XML-Datei remote ist (z. B. `http://myHost/rssfeed.xml`). Beim Wert `false` wird eine absolute Referenz zurückgegeben.
- Das Argument *bAddSchemaReference* ist optional. Es fügt eine Referenz in das aktuelle Dokument ein, die auf den XML-Quell-URI verweist, der im Dialogfeld „XML-Quelle“ angegeben wurde. Bei fehlendem Argument gilt als Standardwert das aktuell geöffnete Dokument.

Rückgabewerte
Ein String, der eine Referenz auf das mit dem aktuellen XSLT-Dokument verbundene XML-Quelldokument enthält. Handelt es sich bei der XML-Quellreferenz um eine Remote-Referenz, gibt die Funktion den heruntergeladenen Dateipfad zum temporären Speicherort zurück.

Beispiel
Im folgenden Beispiel wird das Dialogfeld „XML-Quelle“ ohne Angabe von Werten gestartet:

```
MMXSLT.launchXMLSourceDialog()
```

Kapitel 15: Seiteninhalt

Mit den Seiteninhaltsfunktionen von Adobe® Dreamweaver® werden Vorgänge durchgeführt, die sich auf den Inhalt von Webseiten auswirken. Zu diesen Vorgängen gehören die folgenden:

- Bearbeiten der Elemente im Bedienfeld „Elemente“
- Hinzufügen von Verhalten
- Ausschneiden von Elementen und Einfügen aus der Zwischenablage
- Anwenden von Vorlagen
- Einfügen von Codefragmenten
- Erstellen von Spry-XML-Datensätzen
- Erweitertes Bearbeiten von Spry- und anderen Widgets
- Einfügen von Widgets
- Erstellen von Seitenlayouts, die für verschiedene Browser geeignet sind, mithilfe der Funktionen für die Browserkompatibilitätsprüfung

Funktionen für das Bedienfeld „Elemente“

Mit den Funktionen für das Bedienfeld „Elemente“ (als „asset.Palette“ in der API programmiert) können Sie die Elemente im Bedienfeld verwalten und einsetzen (bei diesen Elementen kann es sich um Vorlagen, Bibliotheken, Bilder, Adobe Shockwave- und Adobe Flash-Inhalt, URLs, Farben und Skripts handeln).

dreamweaver.assetPalette.addToFavoritesFrom Document()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Fügt der Favoritenliste das Element hinzu, das im Dokumentfenster ausgewählt ist. Diese Funktion kann nur für Bilder, Shockwave- und Flash-Dateien, Schriftfarben und URLs eingesetzt werden.

Argumente
Keine.

Rückgabewerte
Keine.

dreamweaver.assetPalette.addToFavoritesFromSiteAssets()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Fügt der Favoritenliste die Elemente hinzu, die in der Siteliste ausgewählt sind, und weist jedem Element in der Favoritenliste einen Kurznamen zu. Dadurch werden die Elemente nicht aus der Siteliste entfernt.

Argumente
Keine.

Rückgabewerte
Keine.

dreamweaver.assetPalette.addToFavoritesFromSiteWindow()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Fügt der Favoritenliste die Elemente hinzu, die im Bedienfeld „Dateien“ ausgewählt sind. Diese Funktion kann nur für Bilder, Filme, Skripts sowie Shockwave- und FLA-Dateien eingesetzt werden. Wenn andere Dateien oder Ordner ausgewählt sind, werden sie ignoriert.

Argumente
Keine.

Rückgabewerte
Keine.

dreamweaver.assetPalette.copyToSite()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Kopiert ausgewählte Elemente in eine andere Site und fügt sie der Favoritenliste dieser Site hinzu. Wenn es sich bei den Elementen nicht um Farben oder URLs, sondern um Dateien handelt, wird die jeweilige Datei in die andere Site kopiert.

Argumente
targetSite

- Das Argument *targetSite* ist der Name der Ziel-Site, der vom Aufruf `site.getSites()` zurückgegeben wird.

Rückgabewerte
Keine.

dreamweaver.assetPalette.edit()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Bearbeitet ausgewählte Elemente mit dem primären externen Editor oder mit dem Steuerelement zur benutzerdefinierten Bearbeitung. Bei Farben wird die Farbauswahl eingeblendet. Bei URLs wird ein Dialogfeld eingeblendet, und der Benutzer wird aufgefordert, eine URL und einen Kurznamen einzugeben. Diese Funktion steht für die Siteliste der Farben und URLs nicht zur Verfügung.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dreamweaver.assetPalette.canEdit()“ auf Seite 516.

dreamweaver.assetPalette.getSelectedCategory()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Gibt die aktuell ausgewählte Kategorie zurück.

Argumente
Keine.

Rückgabewerte
Die derzeit ausgewählte Kategorie. Dabei kann es sich um Folgendes handeln: `"templates"`, `"library"`, `"images"`, `"movies"`, `"shockwave"`, `"flash"`, `"scripts"`, `"colors"` oder `"urls"`.

dreamweaver.assetPalette.getSelectedItems()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Gibt ein Array der ausgewählten Elemente im Bedienfeld „Elemente“ (Siteliste oder Favoritenliste) zurück.

Argumente
Keine.

Rückgabewerte
Ein Array aus drei Strings für jedes ausgewählte Element:

- Der *name*-String, bei dem es sich um den im Bedienfeld „Elemente“ angezeigten Namen bzw. Dateinamen oder Kurznamen handelt.
- Der `value`-String, bei dem es sich um den vollständigen Verzeichnispfad, die URL oder den Farbwert handelt, abhängig vom jeweils ausgewählten Element.
- Der *type*-String, der entweder `"folder"` lautet oder eine der folgenden Kategorien ist: `"templates"`, `"library"`, `"images"`, `"movies"`, `"shockwave"`, `"flash"`, `"scripts"`, `"colors"` oder `"urls"`.

Hinweis: *Wenn im Bedienfeld „Elemente“ nichts ausgewählt ist, gibt diese Funktion ein Array mit einem leeren String zurück.*

Beispiel
Wenn die Kategorie „urls“ lautet und in der Favoritenliste der Ordner „MyFolderName“ und die URL „MyFavoriteURL“ ausgewählt sind, gibt die Funktion Folgendes zurück:

```
items[0] = "MyFolderName"
items[1] = "//path/FolderName"
items[2] = "folder"
items[3] = "MyFavoriteURL"
items[4] = "http://www.MyFavoriteURL.com"
items[5] = "urls"
```

dreamweaver.assetPalette.getSelectedView()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Zeigt an, welche Liste derzeit im Bedienfeld „Elemente“ angezeigt wird.

Argumente
Keine.

Rückgabewerte
Gibt einen String mit dem Wert `"site"` oder `"favorites"` zurück.

dreamweaver.assetPalette.insertOrApply()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Fügt die ausgewählten Elemente ein oder wendet das Element auf die aktuelle Auswahl an. Wendet Vorlagen, Farben und URLs auf die Auswahl an und fügt URLs und andere Elemente an der Einfügemarke ein. Diese Funktion steht nur zur Verfügung, wenn ein Dokument geöffnet ist.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dreamweaver.assetPalette.canInsertOrApply()“ auf Seite 517.

dreamweaver.assetPalette.locateInSite()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Wählt Dateien aus, die mit den im lokalen Bereich des Bedienfelds „Dateien“ ausgewählten Elementen verknüpft sind. Diese Funktion kann für Farben und URLs nicht eingesetzt werden. Sie steht sowohl in der Siteliste als auch in der Favoritenliste zur Verfügung. Wenn ein Ordner in der Favoritenliste ausgewählt ist, wird er ignoriert.

Argumente
Keine.

Rückgabewerte
Keine.

dreamweaver.assetPalette.newAsset()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Erstellt ein neues Element für die aktuelle Kategorie in der Favoritenliste. Bei Bibliotheken und Vorlagen wird eine neue, leere Bibliotheks- oder Vorlagendatei erstellt, die sofort benannt werden kann. Bei Farben wird die Farbauswahl eingeblendet. Bei URLs wird ein Dialogfeld eingeblendet, und der Benutzer wird aufgefordert, eine URL und einen Kurznamen einzugeben. Diese Funktion steht für Bilder, Shockwave- oder Flash-Dateien und Skripts nicht zur Verfügung.

Argumente
Keine.

Rückgabewerte
Keine.

dreamweaver.assetPalette.newFolder()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Erstellt in der aktuellen Kategorie einen neuen Ordner und weist diesem Ordner den Standardnamen („unbenannt“) zu. Der Standardname wird in einem Textfeld angezeigt. Diese Funktion steht nur in der Favoritenliste zur Verfügung.

Argumente
Keine.

Rückgabewerte
Keine.

dreamweaver.assetPalette.recreateLibraryFrom Document()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Ersetzt die nicht mehr empfohlene Funktion `libraryPalette`, `recreateLibraryFromDocument()`. Diese Funktion erstellt für die ausgewählte Instanz eines Bibliothekselements im aktuellen Dokument eine LBI-Datei (LBI = Library Item). Die gleiche Wirkung erzielen Sie, wenn Sie im Eigenschafteninspektor auf „Neu erstellen“ klicken.

Argumente
Keine.

Rückgabewerte
Keine.

dreamweaver.assetPalette.refreshSiteAssets()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Analysiert die Site, wechselt zur Siteliste und füllt die Liste aus.

Argumente
Keine.

Rückgabewerte
Keine.

dreamweaver.assetPalette.removeFromFavorites()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Entfernt die ausgewählten Elemente aus der Favoritenliste. Die zugehörigen Dateien auf der Festplatte werden dabei jedoch nicht gelöscht, es sei denn, es handelt sich um Bibliotheken oder Vorlagen. Bei diesen beiden Kategorien wird der Benutzer zur Bestätigung aufgefordert, bevor die Datei gelöscht wird. Diese Funktion steht nur in der Favoritenliste bzw. für die Kategorien „Bibliothek“ und „Vorlagen“ zur Verfügung.

Argumente
Keine.

Rückgabewerte
Keine.

dreamweaver.assetPalette.renameNickname()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Ermöglicht die Bearbeitung eines Ordnernamens oder des Kurznamens einer Datei, indem der vorhandene Name in einem Textfeld angezeigt wird. Diese Funktion steht nur in der Favoritenliste bzw. für die Kategorien „Bibliothek“ und „Vorlagen“ zur Verfügung.

Argumente
Keine.

Rückgabewerte
Keine.

dreamweaver.assetPalette.setSelectedCategory()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Zeigt eine andere Kategorie an.

Argumente
categoryType

- Das Argument *categoryType* kann eine der folgenden Kategorien sein: `"templates"`, `"library"`, `"images"`, `"movies"`, `"shockwave"`, `"flash"`, `"scripts"`, `"colors"` oder `"urls"`.

Rückgabewerte
Keine.

dreamweaver.assetPalette.setSelectedView()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Wechselt zwischen der Anzeige der Siteliste und der Favoritenliste.

Argumente
viewType

- Beim Argument *viewType* handelt es sich um einen String, der `"site"` oder `"favorites"` lauten kann.

Rückgabewerte
Keine.

dreamweaver.referencePalette.getFontSize()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Gibt die aktuelle Schriftgröße zurück, die im Anzeigebereich des Bedienfelds „Referenz“ verwendet wird.

Argumente
Keine.

Rückgabewerte
Die relative Schriftgröße: `small`, `medium` oder `large`.

dreamweaver.referencePalette.setFontSize()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Ändert die im Bedienfeld „Referenz“ verwendete Schriftgröße.

Argumente
fontSize

- Das Argument *fontSize*ist eine der folgenden relativen Größen: `small`, `medium` oder `large`.

Rückgabewerte
Keine.

Verhaltensfunktionen

Mit Verhaltensfunktionen können Sie einem Objekt Verhalten hinzufügen sowie Verhalten aus einem Objekt entfernen. Zudem können Sie feststellen, welche Verhalten mit einem Objekt verknüpft sind, und Informationen zu den mit einem Verhalten verknüpften Objekten abrufen und vieles mehr. Die Methoden des Objekts `dreamweaver.behaviorInspector` gelten nicht für die Auswahl im aktuellen Dokument, sondern steuern bzw. ändern ausschließlich die Auswahl im Bedienfeld „Verhalten“.

dom.addBehavior()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Fügt dem ausgewählten Element ein neues Ereignis-Aktion-Paar hinzu. Diese Funktion ist nur für das aktive Dokument gültig.

Argumente
event, action, {eventBasedIndex}

- Das Argument *event* bezeichnet die JavaScript-Ereignisprozedur, die zum Verknüpfen des Verhaltens mit dem Element verwendet werden soll (beispielsweise `onClick`, `onMouseOver` oder `onLoad`).
- Beim Argument *action* handelt es sich um den Funktionsaufruf, der von `applyBehavior()` zurückgegeben wird, wenn die Aktion mit dem Bedienfeld „Verhalten“ hinzugefügt wird (beispielsweise `"MM_popupMsg('Hello World')"`).
- Das Argument *eventBasedIndex* (optional) bezeichnet die Position, an der diese Aktion hinzugefügt werden soll. Das Argument *eventBasedIndex* ist ein nullbasierter Index. Wenn mit dem angegebenen Ereignis bereits zwei Aktionen verknüpft sind und für *eventBasedIndex* der Wert 1 festgelegt wird, wird die Aktion zwischen den anderen beiden ausgeführt. Bei Auslassung dieses Arguments wird die Aktion nach allen anderen Aktionen ausgeführt, die für das Ereignis definiert sind.

Rückgabewerte
Keine.

dom.getBehavior()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Ruft die Aktion an der angegebenen Position innerhalb des jeweiligen Ereignisses ab. Diese Funktion wirkt sich auf die aktuelle Auswahl aus und ist nur für das aktive Dokument gültig.

Argumente
event, {eventBasedIndex}

- Das Argument *event* bezeichnet die JavaScript-Ereignisprozedur, über die die Aktion mit dem Element verknüpft wird (beispielsweise `onClick`, `onMouseOver` oder `onLoad`).
- Das Argument *eventBasedIndex* (optional) bezeichnet die Position der abzurufenden Aktion. Wenn beispielsweise zwei Aktionen mit dem Ereignis verknüpft sind, bezeichnet 0 die erste und 1 die zweite Aktion. Bei Auslassung dieses Arguments gibt die Funktion alle Aktionen für das betreffende Ereignis zurück.

Rückgabewerte
Ein String mit dem Funktionsaufruf (Beispiel: `"MM_swapImage('document.Image1','document.Image1','foo.gif','#933292969950')"`) oder ein Array von Strings, wenn *eventBasedIndex* weggelassen wird.

dom.reapplyBehaviors()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob die Funktionen, die an dem betreffenden Knoten mit Verhaltensaufrufen verknüpft sind, sich im Bereich `HEAD` des Dokuments befinden. Wenn nicht, werden sie eingefügt.

Argumente
elementNode

- Das Argument *elementNode* ist ein Elementknoten im aktuellen Dokument. Bei Auslassung des Arguments werden alle Elementknoten im Dokument auf verwaiste Verhaltenaufrufe überprüft.

Rückgabewerte
Keine.

dom.removeBehavior()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Entfernt die Aktion an der angegebenen Position im jeweiligen Ereignis. Diese Funktion wirkt sich auf die aktuelle Auswahl aus und ist nur für das aktive Dokument gültig.

Argumente
event, {eventBasedIndex}

- Das Argument *event* bezeichnet die Ereignisprozedur, über die die Aktion mit dem Element verknüpft wird (beispielsweise `onClick`, `onMouseOver` oder `onLoad`). Bei Auslassung des Arguments werden alle Aktionen vom Element entfernt.

- Das Argument *eventBasedIndex* (optional) bezeichnet die Position der zu entfernenden Aktion. Wenn beispielsweise zwei Aktionen mit dem Ereignis verknüpft sind, bezeichnet 0 die erste und 1 die zweite Aktion. Bei Auslassung dieses Arguments werden alle Aktionen für das betreffende Ereignis entfernt.

Rückgabewerte
Keine.

dreamweaver.getBehaviorElement()

Verfügbarkeit
Dreamweaver 2, aktualisiert in CS4.

Beschreibung
Ruft das DOM-Objekt ab, das dem Tag entspricht, auf das das Verhalten angewendet wird. Diese Funktion ist nur in Verhaltensaktionsdateien gültig.

Argumente
Keine.

Rückgabewerte
Ein DOM-Objekt oder `null`-Wert. Diese Funktion gibt unter folgenden Umständen einen `null`-Wert zurück:

- Wenn das aktuelle Skript nicht im Zusammenhang mit dem Bedienfeld „Verhalten“ ausgeführt wird
- Wenn `dreamweaver.popupAction()` das derzeit ausgeführte Skript startet
- Wenn das Bedienfeld „Verhalten“ ein Ereignis mit einem Hyperlink-Wrapper verknüpft, der noch nicht existiert
- Wenn diese Funktion außerhalb einer Aktionsdatei auftritt

Beispiel
Die Funktion `dreamweaver.getBehaviorElement()` kann genau wie „dreamweaver.getBehaviorTag()“ auf Seite 327 verwendet werden, um feststellen, ob die ausgewählte Aktion für das ausgewählte HTML-Tag geeignet ist. Der Unterschied besteht darin, dass Sie bei dieser Funktion auf zusätzliche Informationen über das Tag und dessen Attribute zugreifen können. Wenn Sie eine Aktion schreiben, die nur auf einen Hyperlink (`A HREF`) angewendet werden kann, der keinen anderen Frame bzw. kein anderes Fenster zum Ziel hat, können Sie die Funktion `getBehaviorElement()` verwenden. Sie können die Funktion „getBehaviorElement()“ als Teil der Funktion einsetzen, die die Benutzeroberfläche des Dialogfelds „Parameter“ initialisiert. Dies wird im folgenden Beispiel demonstriert:

```
function initializeUI(){
    var theTag = dreamweaver.getBehaviorElement();
    var CANBEAPPLIED = (theTag.tagName == "A" && ¬
    theTag.getAttribute("HREF") != null && ¬
    theTag.getAttribute("TARGET") == null);
    if (CANBEAPPLIED) {
        // display the action user interface
    } else{
        // display a helpful message that tells the user
        // that this action can only be applied to a
        // link without an explicit target]
    }
}
```

dreamweaver.getBehaviorTag()

Verfügbarkeit
Dreamweaver 1.2.

Beschreibung
Ruft den Quellcode des Tags ab, auf das das Verhalten angewendet wird. Diese Funktion ist nur in Aktionsdateien gültig.

Argumente
Keine.

Rückgabewerte
Ein String mit dem Quellcode des Tags. Dies ist der gleiche String, der als Argument (*HTMLelement*) an die Funktion `canAcceptBehavior()` übergeben wird. Wenn diese Funktion außerhalb einer Aktionsdatei auftritt, wird ein leerer String zurückgegeben.

Beispiel
Wenn Sie eine Aktion schreiben, die nur auf einen Hyperlink (`A HREF`) angewendet werden kann, können Sie die Funktion `getBehaviorTag()` in der Funktion verwenden, die die Benutzerschnittstelle des Dialogfelds „Parameter“ initialisiert. Dies wird im folgenden Beispiel veranschaulicht:

```
function initializeUI(){
    var theTag = dreamweaver.getBehaviorTag().toUpperCase();
    var CANBEAPPLIED = (theTag.indexOf('HREF') != -1));
    if (CANBEAPPLIED) {
        // display the action UI
    } else{
        // display a helpful message that tells the user
        // that this action can only be applied to a
        // hyperlink
    }
}
```

dreamweaver.popupAction()

Verfügbarkeit
Dreamweaver 2, aktualisiert in CS4.

Beschreibung
Ruft das Dialogfeld „Parameter“ für die betreffende Verhaltensaktion auf. Für den Benutzer ist die Wirkung dieselbe, als würde er die Aktion im Bedienfeld „Verhalten“ im Popupmenü der Aktionen auswählen. Mit dieser Funktion können Erweiterungsdateien, bei denen es sich nicht um Aktionen handelt, Verhalten mit Objekten im Benutzerdokument verknüpfen. Andere Bearbeitungsvorgänge sind so lange gesperrt, bis der Benutzer das Dialogfeld schließt.

Hinweis: *Diese Funktion kann innerhalb der Funktion* `objectTag()` *oder in einem Skript in einer Befehlsdatei oder Eigenschafteninspektor-Datei aufgerufen werden.*

Argumente
`actionName, {funcCall}`

- Das Argument *`actionName`* ist ein String, der den Namen einer Datei im Ordner „Configuration/Behaviors/Actions“ enthält. Die Datei enthält eine JavaScript-Verhaltensaktion (z. B. "*`Swap Image.htm`*").
- Das optionale Argument *`funcCall`* ist ein String mit einem Funktionsaufruf für die in *`actionName`* angegebene Aktion, z. B. "*`MM_SwapImage(...)`*". Die Funktion *`applyBehavior()`* in der Aktionsdatei stellt dieses Argument bereit, sofern angegeben.

Rückgabewerte
Der Funktionsaufruf für die Verhaltensaktion. Wenn der Benutzer im Dialogfeld „Parameter“ auf „OK“ klickt, wird das Verhalten dem aktuellen Dokument hinzugefügt. Die entsprechenden Funktionen werden dem Abschnitt `HEAD` des Dokuments hinzugefügt. HTML-Code wird oben im Abschnitt `BODY` eingefügt und das Dokument kann weiter bearbeitet werden. Der Funktionsaufruf (z. B. "*`MM_SwapImage(...)`*") wird nicht dem Dokument hinzugefügt, sondern wird zum Rückgabewert dieser Funktion.

dreamweaver.behaviorInspector.getBehaviorAt()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Ruft das Ereignis-Aktion-Paar an der angegebenen Position im Bedienfeld „Verhalten“ ab.

Argumente
positionIndex

- Das Argument *positionIndex* ist die Position der Aktion im Bedienfeld „Verhalten“. Die erste Aktion in der Liste befindet sich an Position 0.

Rückgabewerte
Ein Array mit zwei Elementen:

- Eine Ereignisprozedur
- Ein Funktionsaufruf bzw. eine JavaScript-Anweisung

Beispiel
`positionIndex` ist ein nullbasierter Index. Wenn die Liste im Bedienfeld „Verhalten“ angezeigt wird, gibt ein Aufruf an die Funktion `dreamweaver.behaviorInspector.getBehaviorAt(2)` ein Array mit zwei Strings zurück: `"onMouseOver"` und `"MM_changeProp('document.moon','document.moon','src','sun.gif','MG')"`.

dreamweaver.behaviorInspector.getBehaviorCount()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Zählt die mit dem aktuell ausgewählten Element über Ereignisprozeduren verknüpften Aktionen.

Argumente
Keine.

Rückgabewerte
Eine Ganzzahl für die Anzahl der Aktionen, die mit dem Element verknüpft sind. Dieser Wert entspricht der im Bedienfeld „Verhalten“ angezeigten Anzahl von Aktionen und beinhaltet sowohl Dreamweaver-Verhaltensaktionen als auch benutzerdefinierten JavaScript-Code.

Beispiel
Beim Aufruf von `dreamweaver.behaviorInspector.getBehaviorCount()` für den ausgewählten Hyperlink `<A HREF="javascript:setCookie()" onClick="MM_popupMsg('A cookie has been set.');parent.rightframe.location.href='aftercookie.html'">` wird der Wert 2 zurückgegeben.

dreamweaver.behaviorInspector.getSelectedBehavior()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Ruft die Position der im Bedienfeld „Verhalten“ ausgewählten Aktion ab.

Argumente
Keine.

Rückgabewerte
Eine Ganzzahl, die die Position der im Bedienfeld „Verhalten“ ausgewählten Aktion darstellt, oder –1, wenn keine Aktion ausgewählt ist.

Beispiel
Wenn im Bedienfeld „Verhalten“ die erste Aktion ausgewählt ist, wird beim Aufruf von `dreamweaver.behaviorInspector.getSelectedBehavior()` der Wert 0 zurückgegeben.

dreamweaver.behaviorInspector.moveBehaviorDown()

Verfügbarkeit
Dreamweaver 3.

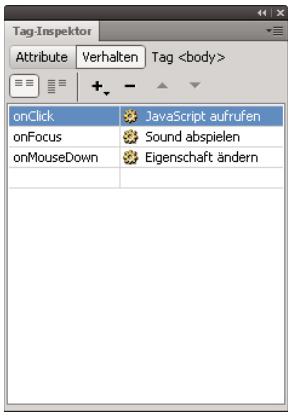
Beschreibung
Verschiebt eine Verhaltensaktion für ein Ereignis in der Ausführungsreihenfolge nach unten.

Argumente
positionIndex

- Das Argument *positionIndex* ist die Position der Aktion im Bedienfeld „Verhalten“. Die erste Aktion in der Liste befindet sich an Position 0.

Rückgabewerte
Keine.

Beispiel
Beim Aufruf der Funktion `dreamweaver.behaviorInspector.moveBehaviorDown(2)` werden die Positionen der Aktionen „Bilder vorausladen“ und „Eigenschaft ändern“ des Ereignisses `onMouseDown` vertauscht. Der Aufruf von `dreamweaver.behaviorInspector.moveBehaviorDown()` für eine andere Position hat keine Wirkung, weil die Ereignisse `onClick` und `onFocus` jeweils mit nur einem Verhalten verknüpft sind und das Verhalten an Position 3 sich bereits an der untersten Stelle der Gruppe `onMouseDown` befindet.



Verwandte Themen
„dreamweaver.behaviorInspector.getSelectedBehavior()“ auf Seite 329

dreamweaver.behaviorInspector.moveBehaviorUp()

Verfügbarkeit
Dreamweaver 3.

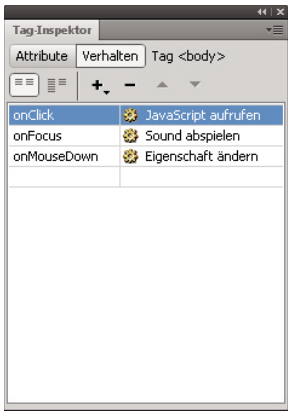
Beschreibung
Verschiebt eine Verhaltensaktion für ein Ereignis in der Ausführungsreihenfolge nach oben.

Argumente
positionIndex

- Das Argument *positionIndex* ist die Position der Aktion im Bedienfeld „Verhalten“. Die erste Aktion in der Liste befindet sich an Position 0.

Rückgabewerte
Keine.

Beispiel
Beim Aufruf der Funktion `dreamweaver.behaviorInspector.moveBehaviorUp(3)` werden die Positionen der Aktionen „Bilder vorausladen“ und „Eigenschaft ändern“ des Ereignisses `onMouseOver` vertauscht. Der Aufruf von `dreamweaver.behaviorInspector.moveBehaviorUp()` für eine andere Position hat keine Wirkung, weil die Ereignisse `onClick` und `onFocus` jeweils mit nur einem Verhalten verknüpft sind und das Verhalten an Position 2 sich bereits an oberster Stelle der Gruppe `onMouseDown` befindet.



Verwandte Themen
„dreamweaver.behaviorInspector.getSelectedBehavior()“ auf Seite 329

dreamweaver.behaviorInspector.setSelectedBehavior()

Verfügbarkeit
Dreamweaver 3.

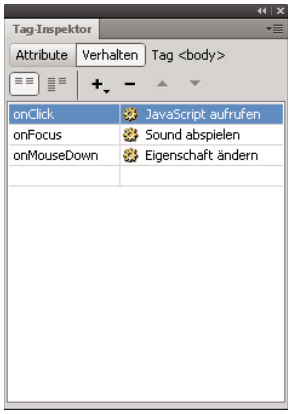
Beschreibung
Wählt die Aktion an der angegebenen Position im Bedienfeld „Verhalten“ aus.

Argumente
positionIndex

- Das Argument *positionIndex* ist die Position der Aktion im Bedienfeld „Verhalten“. Die erste Aktion in der Liste befindet sich an Position 0. Um die Auswahl aller Aktionen aufzuheben, geben Sie für *positionIndex* den Wert –1 an. Die Angabe einer Position, für die keine Aktion definiert ist, ist gleichbedeutend mit dem Wert –1.

Rückgabewerte
Keine.

Beispiel
Beim Aufruf der Funktion `dreamweaver.behaviorInspector.setSelection(2)` wird die mit dem Ereignis `onMouseDown` verknüpfte Aktion „Eigenschaft ändern“ ausgewählt:



Verwandte Themen
„dreamweaver.behaviorInspector.getSelectedBehavior()“ auf Seite 329

Zwischenablagefunktionen

Die Zwischenablagefunktionen dienen zum Ausschneiden, Kopieren und Einfügen von Elementen. Beim Macintosh können sich einige der Zwischenablagefunktionen auch auf Textfelder in Dialogfeldern und schwebenden Bedienfeldern beziehen. Funktionen, die auf Textfelder angewendet werden können, sind als Methoden des `dreamweaver`-Objekts und des DOM-Objekts implementiert. Die `dreamweaver`-Version der Funktion wirkt sich auf die Auswahl im aktiven Fenster aus, d. h. auf das aktuelle Dokumentfenster, den Codeinspektor oder das Bedienfeld „Dateien“. Auf dem Macintosh kann die Funktion auch auf die Auswahl in einem Textfeld angewendet werden, wenn dies das aktuelle Feld ist. Die DOM-Version der Funktion wirkt sich dagegen stets auf die Auswahl im angegebenen Dokument aus.

dom.clipCopy()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Kopiert die Auswahl in die Zwischenablage, einschließlich des HTML-Codes, der die Auswahl definiert.

Argumente
Keine.

Rückgabewerte
Keine.

dom.clipCopyText()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Kopiert den ausgewählten Text in die Zwischenablage. Eventuell vorhandener HTML-Code wird dabei ignoriert.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dom.canClipCopyText()“ auf Seite 507.

dom.clipCut()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Schneidet die Auswahl aus und kopiert sie in die Zwischenablage, einschließlich des HTML-Codes, der die Auswahl definiert.

Argumente
Keine.

Rückgabewerte
Keine.

dom.clipPaste()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Fügt den Inhalt der Zwischenablage im aktuellen Dokument an der Einfügemarke bzw. anstelle der momentanen Auswahl ein. Wenn die Zwischenablage HTML-Code enthält, wird dieser entsprechend interpretiert.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dom.canClipPaste()“ auf Seite 507.

Beispiel
Wenn die Zwischenablage `ABC Widgets` enthält, führt ein Aufruf von `dw.getDocumentDOM().clipPaste()` zu folgendem Ergebnis:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Untitled Document</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>

<body>
ABC Widgets</body>
</html>
```

ABC Widgets

dreamweaver.clipCopy()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Kopiert die aktuelle Auswahl aus dem aktiven Dokumentfenster, Dialogfeld, schwebenden Bedienfeld oder Bedienfeld „Dateien“ in die Zwischenablage.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dreamweaver.canClipCopy()“ auf Seite 517.

dreamweaver.clipCut()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Verschiebt die aktuelle Auswahl aus dem aktiven Dokumentfenster, Dialogfeld, schwebenden Bedienfeld oder Bedienfeld „Dateien“ in die Zwischenablage.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dreamweaver.canClipCut()“ auf Seite 517.

dreamweaver.clipPaste()

Verfügbarkeit
Dreamweaver 3. In Dreamweaver 8 wurde das Argument *strPasteOption* hinzugefügt.

Beschreibung
Fügt den Inhalt der Zwischenablage im aktuellen Dokumentfenster, Dialogfeld, schwebenden Bedienfeld oder Bedienfeld „Dateien“ ein.

Argumente
{strPasteOption}

- Das optionale Argument *strPasteOption* gibt an, welche Art von Einfügeoperation durchzuführen ist. Zu den Werten gehören `"text"`, `"structured"`, `"basicFormat"` und `"fullFormat"`.

Rückgabewerte
Keine.

Enabler
Siehe „dreamweaver.canClipPaste()“ auf Seite 518.

Beispiel
Im folgenden Beispiel wird der Inhalt der Zwischenablage als Text eingefügt:

```
dw.clipPaste("text");
```

dreamweaver.getClipboardText()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Ruft den gesamten in der Zwischenablage gespeicherten Text ab.

Argumente
{bAsText}

- Der optionale boolesche Wert *bAsText* gibt an, ob der Inhalt der Zwischenablage als Text abgerufen wird. Wenn *bAsText* auf `true` gesetzt ist, wird der Inhalt der Zwischenablage als Text abgerufen. Wenn *bAsText* auf `false` gesetzt ist, wird die Formatierung des Inhalts beibehalten. Der Standardwert dieses Arguments lautet `false`.

Rückgabewerte
Ein String mit dem Inhalt der Zwischenablage, falls diese Text enthält (auch HTML), andernfalls kein Rückgabewert.

Beispiel
Wenn `dreamweaver.getClipboardText()` den String `"text <b>bold</b> text"` zurückgibt, gibt `dreamweaver.getClipboardText(true)` den String `"text bold text"` zurück.

Bibliotheks- und Vorlagenfunktionen

Mit Bibliotheks- und Vorlagenfunktionen lassen sich Verknüpfungen zwischen Dokumenten und Vorlagen bzw. Bibliothekselementen erstellen, aktualisieren und entfernen. Die Methoden des Objekts `dreamweaver.libraryPalette` gelten nicht für die Auswahl im aktuellen Dokument, sondern steuern bzw. ändern die ausgewählten Bibliothekselemente im Bedienfeld „Elemente“. Entsprechend wirken sich die Methoden des Objekts `dreamweaver.templatePalette` auf die ausgewählten Vorlagenelemente im Bedienfeld „Elemente“ aus.

dom.applyTemplate()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Wendet eine Vorlage auf das aktive Dokument an. Wenn kein Argument übergeben wird, wird das Dialogfeld „Vorlage auswählen“ eingeblendet. Diese Funktion ist nur für das aktive Dokument gültig.

Argumente
{templateURL}, bMaintainLink

- Das Argument *templateURL* ist der Pfad zu einer Vorlage in der aktuellen Site im URL-Format „file://“.
- Das Argument *bMaintainLink* ist ein boolescher Wert, der angibt, ob die Verknüpfung zur Originalvorlage beibehalten werden soll (`true`) oder nicht (`false`).

Rückgabewerte
Keine.

Enabler
Siehe „dom.canApplyTemplate()“ auf Seite 506.

dom.detachFromLibrary()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Löst die ausgewählte Instanz eines Bibliothekselements von der zugehörigen LBI-Datei, indem die Sperr-Tags um die Auswahl entfernt werden. Diese Funktion hat die gleiche Wirkung wie die Schaltfläche „Von Original trennen“ im Eigenschafteninspektor.

Argumente
Keine.

Rückgabewerte
Keine.

dom.detachFromTemplate()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Löst das aktuelle Dokument von der zugehörigen Vorlage.

Argumente
Keine.

Rückgabewerte
Keine.

dom.getAttachedTemplate()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Ruft den Pfad der Vorlage ab, die zum Dokument gehört.

Argumente
Keine.

Rückgabewerte
Ein String mit dem Pfad der Vorlage im URL-Format „file://“.

dom.getEditableRegionList()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Ruft eine Liste aller bearbeitbaren Bereiche im Body des Dokuments ab.

Argumente
Keine.

Rückgabewerte
Ein Array von Elementknoten.

Beispiel
„dom.getSelectedEditableRegion()“ auf Seite 339.

dom.getIsLibraryDocument()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Bestimmt, ob es sich bei dem Dokument um ein Bibliothekselement handelt.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert, der angibt, ob das Dokument eine LBI-Datei ist.

dom.getIsTemplateDocument()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Bestimmt, ob es sich bei dem Dokument um eine Vorlage handelt.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert, der angibt, ob das Dokument eine DWT-Datei ist.

dom.getSelectedEditableRegion()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Wenn sich die Auswahl bzw. die Einfügemarke innerhalb eines bearbeitbaren Bereichs befindet, ruft diese Funktion die Position des bearbeitbaren Bereichs im Body des Dokuments ab.

Argumente
Keine.

Rückgabewerte
Ein Index in das Array, das von der Funktion `dom.getEditableRegionList()` zurückgegeben wird. Weitere Informationen siehe „dom.getEditableRegionList()“ auf Seite 338.

Beispiel
Mit dem folgenden Code wird ein Dialogfeld mit dem Inhalt des ausgewählten bearbeitbaren Bereichs angezeigt:

```
var theDOM = dw.getDocumentDOM();
var edRegs = theDOM.getEditableRegionList();
var selReg = theDOM.getSelectedEditableRegion();
alert(edRegs[selReg].innerHTML);
```

dom.insertLibraryItem()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Fügt eine Instanz eines Bibliothekselements in das Dokument ein.

Argumente
libraryItemURL

- Das Argument *libraryItemURL* ist der Pfad zu einer LBI-Datei im URL-Format „file://“.

Rückgabewerte
Keine.

dom.markSelectionAsEditable()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Zeigt das Dialogfeld „Neuer bearbeitbarer Bereich“ an. Wenn der Benutzer auf „Neuer Bereich“ klickt, wird die Auswahl als bearbeitbar markiert. Der vorhandene Text bleibt unverändert.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dom.canMarkSelectionAsEditable()“ auf Seite 512.

dom.newEditableRegion()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Zeigt das Dialogfeld „Neuer bearbeitbarer Bereich“ an. Wenn der Benutzer auf „Neuer Bereich“ klickt, wird der Name des Bereichs in geschweiften Klammern ({ }) an der Einfügemarke im Dokument eingefügt.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dom.canMakeNewEditableRegion()“ auf Seite 512.

dom.removeEditableRegion()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Entfernt einen bearbeitbaren Bereich aus dem Dokument. Eventuell vorhandene Inhalte bleiben erhalten; nur die Markierungen für den bearbeitbaren Bereich werden entfernt.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dom.canRemoveEditableRegion()“ auf Seite 513.

dom.updateCurrentPage()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Aktualisiert die Bibliothekselemente und/oder Vorlagen des Dokuments. Diese Funktion ist nur für das aktive Dokument gültig.

Argumente
{typeOfUpdate}

- Das optionale Argument *typeOfUpdate* muss `"library"`, `"template"` oder `"both"` lauten. Wenn das Argument weggelassen wird, lautet der Standardwert `"both"`.

Rückgabewerte
Keine.

dreamweaver.updatePages()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Öffnet das Dialogfeld „Seiten aktualisieren“ und wählt die angegebenen Optionen aus.

Argumente
{typeOfUpdate}

- Das optionale Argument *typeOfUpdate* muss `"library"`, `"template"` oder `"both"` lauten, sofern Sie es angeben. Bei fehlendem Argument gilt der Standardwert `"both"`.

Rückgabewerte
Keine.

Funktionen für das Bedienfeld „Codefragmente“

Mit Dreamweaver können Webentwickler wiederverwendbare Codeblöcke im Bedienfeld „Codefragmente“ bearbeiten, speichern und bei Bedarf abrufen.

Im Bedienfeld „Codefragmente“ werden die einzelnen Codefragmente in einer CSN-Datei im Ordner „Configuration/Snippets“ gespeichert. Die im Lieferumfang von Dreamweaver enthaltenen Codefragmente sind in den folgenden Ordnern gespeichert:

- Accessible
- Comments
- Content_tables
- Filelist.txt
- Footers
- Form_elements
- Headers
- Javascript
- Meta
- Navigation
- Text

Codefragment-Dateien sind XML-Dokumente. Sie können daher die Kodierung in der XML-Direktive angeben, wie im folgenden Beispiel dargestellt:

```
<?XML version="1.0" encoding="utf-8">
```

Das folgende Beispiel zeigt eine Codefragment-Datei:

```
<snippet name="Detect Flash" description="VBscript to check for Flash ActiveX control"
preview="code" factory="true" type="wrap" >
    <insertText location="beforeSelection">
        <![CDATA[ ------- code --------- ]]>
    </insertText>
    <insertText location="afterSelection">
        <![CDATA[ ------- code --------- ]]>
    </insertText>
</snippet>
```

Codefragment-Tags in CSN-Dateien haben folgende Attribute:

Attribut	**Beschreibung**
`name`	Name des Codefragments
`description`	Beschreibung des Codefragments
`preview`	Art der Vorschau: `code"`, wenn das Codefragment im Vorschaubereich angezeigt werden soll, oder `"design"`, wenn das Codefragment als HTML im Vorschaubereich wiedergegeben werden soll.
`type`	`"wrap"`, wenn mit dem Codefragment eine Benutzerauswahl umbrochen wird; `"block"`, wenn das Codefragment vor der Auswahl eingefügt werden soll.

Mit den im Folgenden erläuterten Methoden können Sie Ihren Erweiterungen Funktionen des Bedienfelds „Codefragmente“ hinzuzufügen.

dreamweaver.snippetPalette.getCurrentSnippetPath()

Verfügbarkeit
Dreamweaver MX 2004.

Beschreibung
Gibt den Pfad des Codefragments zurück, das derzeit im Bedienfeld „Codefragmente“ ausgewählt ist.

Argumente
Keine.

Rückgabewerte
Der Pfad des im Bedienfeld „Codefragmente“ ausgewählten Codefragments, relativ zum Ordner „Snippets“. Gibt einen leeren String zurück, wenn kein Codefragment ausgewählt ist.

dreamweaver.snippetPalette.newFolder()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Erstellt einen neuen Ordner mit dem Standardnamen *untitled* und zeigt diesen Namen in einem Textfeld an.

Argumente
Keine.

Rückgabewerte
Keine.

dreamweaver.snippetPalette.newSnippet()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Öffnet das Dialogfeld zum Hinzufügen eines Codefragments und übergibt ihm den Fokus.

Argumente
Keine.

Rückgabewerte
Keine.

dreamweaver.snippetPalette.editSnippet()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Öffnet das Dialogfeld zum Bearbeiten eines Codefragments, übergibt ihm den Fokus und aktiviert die Bearbeitungsfunktionen für das ausgewählte Element.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dreamweaver.snippetpalette.canEditSnippet()“ auf Seite 534.

dreamweaver.snippetPalette.insert()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Wendet das im Bedienfeld „Codefragmente“ ausgewählte Codefragment auf die aktuelle Auswahl an.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dreamweaver.snippetpalette.canInsert()“ auf Seite 534.

dreamweaver.snippetPalette.insertSnippet()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Fügt das angegebene Codefragment in die aktuelle Auswahl ein.

Argumente
path

- Ein String, der den Pfad zum Codefragment relativ zum Ordner „Snippets“ angibt.

Rückgabewerte
Ein boolescher Wert.

Enabler
Siehe „dreamweaver.snippetpalette.canInsert()“ auf Seite 534.

Beispiel
Der folgende Aufruf von `dw.snippetPalette.insertSnippet()` fügt das Codefragment an der durch das Argument angegebenen Position in das aktuelle Dokument an der Einfügemarke ein:

```
dw.snippetPalette.insertSnippet('Text\\Different_Link_Color.csn');
```

dreamweaver.snippetPalette.rename()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Aktiviert ein Textfeld um den ausgewählten Ordnernamen oder Datei-Kurznamen und ermöglicht die Bearbeitung des ausgewählten Elements.

Argumente
Keine.

Rückgabewerte
Keine.

dreamweaver.snippetPalette.remove()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Löscht das ausgewählte Element oder den ausgewählten Ordner aus dem Bedienfeld „Codefragmente“ und löscht die Datei von der Festplatte.

Rückgabewerte
Keine.

Bearbeitungsfunktionen für Spry-Widgets

Dreamweaver CS5 bietet erweiterte Bearbeitungsfunktionen für Spry-Widgets und andere dynamische Widgets.

element.getTranslatedAttribute()

Verfügbarkeit
Dreamweaver CS3.

Beschreibung
Diese Funktion ist identisch mit der W3C-Funktion `getAttribute()`, gilt jedoch für übersetzte Attribute. Die Funktion `element.getTranslatedAttribute()` ruft einen Attributwert nach Namen ab.

Argumente
name

- Das Argument *name* ist ein DOM-String, der den Namen des abzurufenden Attributs enthält.

Rückgabewerte
Gibt den Namen des Attributs als DOM-String zurück. Wenn das Attribut keinen festgelegten Wert und auch keinen Standardwert hat, gibt diese Funktion einen leeren String zurück.

element.removeTranslatedAttribute()

Verfügbarkeit
Dreamweaver CS3.

Beschreibung
Diese Funktion ist identisch mit der W3C-Funktion `removeAttribute()`, gilt jedoch für übersetzte Attribute. Die Funktion `element.removeTranslatedAttribute()` entfernt ein Attribut nach Namen. Wenn das Attribut einen Standardwert hat, wird ein Attribut mit dem Standardwert und dem entsprechenden Namespace-URI, dem lokalen Namen und Präfix, sofern vorhanden, angezeigt.

Argumente
name

- Das Argument *name* ist ein DOM-String, der den Namen des zu entfernenden Attributs enthält.

Rückgabewerte
Keine.

element.setTranslatedAttribute()

Verfügbarkeit
Dreamweaver CS3.

Beschreibung
Diese Funktion ist identisch mit der W3C-Funktion `setAttribute()`, gilt jedoch für übersetzte Attribute. Die Funktion `element.setTranslatedAttribute()` fügt ein neues Attribut mit dem angegebenen Wert hinzu. Wenn ein Attribut mit dem angegebenen Namen bereits im Element vorhanden ist, wird der Wert zu dem im Argument *value* angegebenen Wert geändert.

Der Wert *value* ist ein einfacher String; er wird nicht analysiert, da er festgelegt wird. Deshalb wird jede im String enthaltene Syntax als einfacher Text behandelt und muss von der Implementierung entsprechend ausgeschaltet werden, wenn sie ausgeschrieben wird.

Um einen Attributwert zuzuweisen, der Syntax enthält, die als Entity-Referenz erkannt werden soll, müssen Sie einen `Attr`-Knoten sowie ggf. `Text-` und `EntityReference`-Knoten erstellen, die entsprechende Teilstruktur erstellen und `setAttributeNode` verwenden, um dieses als Wert des Attributs zuzuweisen.

Argumente
name, value

- Das Argument *name* ist ein DOM-String, der den Namen des zu erstellenden oder zu ändernden Attributs enthält.
- Das Argument *value* ist ein DOM-String, der den für das Attribut festzulegenden Wert enthält.

Rückgabewerte
Keine.

element.translatedClassName

Verfügbarkeit
Dreamweaver CS3.

Beschreibung
Diese Funktion ist identisch mit der Funktion `element.ClassName`, gilt jedoch für das übersetzte `className`-Attribut.

element.translatedStyle

Verfügbarkeit
Dreamweaver CS3.

Beschreibung
Diese Funktion ist identisch mit der Funktion `element.style()`, gilt jedoch für das übersetzte style-Attribut.

Beispiel

```
var div1 = dom.getElementById("div1");
div1.translatedStyle.display = "none";
```

Einfügen von Spry-Widget-Funktionen

Dreamweaver bietet die im Folgenden beschriebenen Funktionen, um das Einfügen von Spry-Widgets zu vereinfachen.

dom.addJavaScript()

Verfügbarkeit
Dreamweaver CS3.

Beschreibung
Diese Funktion weist Dreamweaver an, einen JavaScript-Block entweder in den head-Bereich oder in den body-Bereich einzufügen. Beim Einfügen in den body-Bereich wird der JavaScript-Block direkt vor dem Tag `</body>` eingefügt. Wenn das Dokument dort bereits einen JavaScript-Block aufweist, fügt Dreamweaver kein neues `<script>`-Tag ein, sondern hängt `"code"` an den Inhalt des `<script>`-Tags an.

Argumente
code, insideHead

- Das Argument *`code`* ist ein String, der den JavaScript-Code enthält, der in die Seite eingefügt werden soll.
- Das Argument *`insideHead`* ist ein boolescher Wert, der angibt, ob der JavaScript-Block in den head-Bereich oder in den body-Bereich eingefügt werden soll. Der Standardwert ist `true`, womit der Code in den head-Bereich eingefügt wird. Beim Wert `false` wird der Code im body-Bereich direkt vor dem `</body>`-Tag eingefügt. Dieses Argument ist optional.

Rückgabewerte
Keine.

Beispiel

```
function objectTag()
{
    .
    .
    .
    var dom = dw.getDocumentDOM();
    var id = dwscripts.getUniqueId("accordion");
    var code = "new Accordion('" + id + "',250,{duration:200,step:20})";
    dom.addJavaScript(code, false);

    return retVal;
}
```

dom.copyAssets()

Verfügbarkeit
Dreamweaver CS3, aktualisiert in CS4.

Beschreibung

Mit dieser API kann ein Erweiterungsautor externe unabhängige Dateien in die Site des Benutzers kopieren. Der Autor kann außerdem die notwendigen Dateireferenzen in den head-Bereich der Seite einfügen.

Argumente

assetArray

Ein Array von JavaScript-Objekten. Jedes JavaScript-Objekt verfügt über die Felder *srcURL*, *destURL*, *referenceType*, *useDefaultFolder* und *documentRelative*.

- Das Argument *srcURL* ist ein Pfad zum Element im Format `file://URL`.
- Das Argument *destURL* ist ein relativer Pfad, der den Speicherort angibt, in den das Element kopiert werden soll. Wozu *destURL* relativ ist, hängt vom Wert von *useDefaultFolder* ab. Wenn *useDefaultFolder* den Wert `true` hat, ist der Pfad relativ zum standardmäßigen Elementordner. Wenn *useDefaultFolder* den Wert `false` hat, ist der Pfad relativ zum Site-Stamm. Ist die Site nicht definiert, ist der Pfad relativ zum Dokument. Siehe Beschreibung zu *useDefaultFolder*.
- Das Argument *referenceType* ist erforderlich, wenn der Autor der Erweiterung einen Dateiverweis in den head-Bereich einfügen möchte. Die gültigen Werte für *referenceType* lauten wie folgt:
  - `link`, um ein `LINK`-Tag für eine externe CSS-Datei einzufügen
  - `import`, um ein `STYLE`-Tag mit `@import` einzufügen
  - `javascript`, um ein `SCRIPT`-Tag mit `type=text/javascript` einzufügen
  - `vbscript`, um ein `SCRIPT`-Tag mit `type=text/vbscript` einzufügen
  - `""`, um keinen Verweis in den head-Bereich einzufügen
- Das Argument *useDefaultFolder* ist ein boolescher Wert, der angibt, ob der in *destURL* angegebene Pfad relativ zum Standardordner „Elemente“ ist. Wenn der Wert `false` lautet, ist diese Eigenschaft nicht festgelegt. In diesem Fall wird davon ausgegangen, dass *destURL* relativ zum Site-Stamm ist. Ist die Site nicht definiert, wird davon ausgegangen, dass *destURL* relativ zum Dokument ist. Der Standardwert für dieses Argument lautet `false`.
- Das Argument *documentRelative* ist ein boolescher Wert. Der Standardwert ist `false`. Wenn dieser Parameter den Wert `false` hat, werden die Elemente in den in *destURL* angegebenen Ordner relativ zum Site-Stamm kopiert, wenn die Datei in einer Site gespeichert wird. Ist der Wert `true`, werden die Elemente in den in *destURL* angegebenen Pfad relativ zum Dokument kopiert.

Rückgabewerte

Ein Array von Strings im URL-Format „file://“. Jeder String stellt eine Datei dar, die über ein Skript oder ein link-Tag im head-Bereich des Dokuments eingefügt wurde.

Beispiel

```
function objectTag()
{
    .
    .
    .
    var dom = dw.getDocumentDOM();
    var assetList = new Array();
    var assetInfo = new AssetInfo("Objects/Ajax/Accordion.css",
                                  "Objects/Ajax/Accordion.css",
                                  "Accordion.css", "link");
    assetList.push(assetInfo);
    assetInfo = new AssetInfo("Objects/Ajax/Accordion.js", "Accordion.js",
                              "javascript");
    assetList.push(assetInfo);
    assetInfo = new AssetInfo("Objects/Ajax/Images", "Images", "");
    assetList.push(assetInfo);
    dom.copyAssets(assetList);
    return retVal;
}
```

dom.getDefaultAssetFolder()

Verfügbarkeit

Dreamweaver CS3.

Beschreibung

Ruft den standardmäßigen Elementordner des Dokuments ab.

Argumente

Keine.

Rückgabewerte

Ein String, der den Namen des standardmäßigen Elementordners enthält.

Beispiel

```
function objectTag()
{
.
.
.
var defaultAssetFolder = dom.getDefaultAssetFolder();
.
.
.
return retVal;
}
```

Funktionen für die Browserkompatibilitätsprüfung

Die folgenden Funktionen erleichtern das Suchen von Kombinationen von HTML und CSS, die Browserfehler auslösen können (weitere Informationen finden Sie in *Dreamweaver erweitern* im Kapitel „API für die Browserkompatibilitätsprüfung"), sie können jedoch auch in anderen Erweiterungstypen (z. B. in Befehlen) verwendet werden.

Hinweis: *Die Werte, die diese Funktionen zurückgeben, stellen die Stile dar, die derzeit in der Entwurfsansicht gelten. Wenn die Funktionen in Problemdateien als Teil der Browserkompatibilitätsprüfung verwendet werden, filtert Dreamweaver die Stile automatisch entsprechend den Zielbrowsern (Stile, die mit Star HTML definiert wurden, werden z. B. berücksichtigt, wenn als Zielbrowser Internet Explorer bis Version 6 verwendet wird). Diese Filterung wird jedoch nicht durchgeführt, wenn die Funktionen nicht im Rahmen einer Browserkompatibilitätsprüfung verwendet werden.*

elem.getComputedStyleProp()

Verfügbarkeit
Dreamweaver CS3.

Beschreibung
Ruft den Wert der angegebenen CSS-Eigenschaft ab, die zur Darstellung des angegebenen Elements verwendet wird, unabhängig davon, wo die Eigenschaft im CSS definiert ist. Die Länge wird in Pixel angegeben (obwohl im Gegensatz zu den Browsern „px" nicht mit dem Wert angegeben wird).

Argumente
propName, pseudoElt

- Das Argument *propName* ist der Name einer CSS-Eigenschaft. (Verwenden Sie keine Bindestriche, sondern gemischte Groß- und Kleinschreibung. `"font-size"` wird z. B. `"fontSize"`.)
- Das Argument *pseudoElt* ist das CSS-Pseudoelement oder `null`, wenn dieses nicht vorhanden ist.

Rückgabewerte
Ein String, der den berechneten Wert dieser Eigenschaft enthält.

Hinweis: *Numerische Werte werden ebenfalls als Strings zurückgegeben; um diese Werte in Berechnungen zu verwenden, konvertieren Sie sie mit* `parseInt()` *oder* `parseFloat()` *in Zahlen.*

Beispiel

```
var dom = dw.getDocumentDOM();
var myDiv = dom.getElementsByTagName('myDiv');
var float = myDiv.getComputedStyleProp("float");
if (float == "left")
    alert("This div is floated left.");
```

window.getDeclaredStyle()

Verfügbarkeit
Dreamweaver CS3.

Beschreibung
Ruft die CSS-Stile ab, die für das angegebene Element deklariert wurden. Unterscheidet sich insofern von der Funktion `getComputedStyle()`, dass nicht ausdrücklich deklarierte Stile undefiniert sind. Außerdem werden tatsächliche Längenwerte gemäß Deklaration im Stylesheet (z. B. 20%, 0,8em) angegeben, und nicht die berechneten Pixelwerte. Wenn `bGetInherited` den Wert „false“ hat (dies ist die Standardvorgabe), ruft `getDeclaredStyle()` nur die Stile ab, die direkt auf das Element angewendet werden; vom übergeordneten Element übernommene Stile sind nicht eingeschlossen.

Argumente
elt, pseudoElt, psuedoClassList, bGetInherited

- *elt* – ein Knoten in dem Dokument, dessen Stilinformationen gewünscht werden.
- *pseudoElt* – das CSS-Pseudoelement oder `null`, falls dieses nicht vorhanden ist.
- *pseudoClassList* – ein optionaler String, der aus einer Liste von Pseudoklassen besteht, die durch Leerzeichen voneinander getrennt sind.
- *bGetInherited* – ein optionaler boolescher Wert, der angibt, ob von übergeordneten Elementen übernommene Stile einzuschließen sind (standardmäßig `false`).

Rückgabewerte
Ein schreibgeschütztes Objekt mit Stileigenschaften, auf die anhand der Namen zugegriffen werden kann.

Beispiel
```
var dom = dw.getDocumentDOM();
var myDiv = dom.getElementById('myDiv');
var props = window.getDeclaredStyle(myDiv);
var marleft = "";
var units = "";
if (typeof(props.marginLeft) != "undefined"){
    marleft = props.marginLeft;
    units = marleft.replace(/\d+/,""); // remove digits, leaving units
    alert(units); // should show %, px, pt, em, etc.
}
else
alert("no margin-left property has been set for myDiv.");
```

dom.getMinDisplayWidth()

Verfügbarkeit
Dreamweaver CS3.

Beschreibung
Ruft die Mindestbreite ab, die für einen Container auf Blockebene erforderlich ist, um den gesamten Inhalt anzuzeigen.

Hinweis: *Die tatsächliche Breite des Containers kann kleiner sein, falls mithilfe von CSS ein Wert festgelegt ist, der kleiner ist als der von der Funktion* `dom.minDisplayWidth()` *zurückgegebene Wert.*

Argumente
container

- *container* – das Container-Element, für das eine Mindestbreite erforderlich ist.

Rückgabewerte
Eine Ganzzahl, die die kleinste Anzeigebreite des angegebenen Containers in Pixel darstellt, oder -1, falls das Element kein Container ist oder seine Mindestbreite nicht bestimmt werden kann.

Beispiel

```
var dom = dw.getDocumentDOM();
var myDiv = dom.getElementById('myDiv');
var props = window.getComputedStyle(myDiv);
var minW = dom.getMinDisplayWidth(myDiv);
var setW = props.width;
if (minW > setW)
alert("Depending on the browser, your content will either be \n" +
        "clipped, or the container will expand beyond its set width.");
```

dom.getBlockElements() elem.getBlockElements()

Verfügbarkeit
Dreamweaver CS3.

Beschreibung
Durchsucht das Dokument (oder das Element) nach untergeordneten Elementen mit einem inhärenten oder angegebenen Anzeigewert `'block'`.

Argumente
Keine.

Rückgabewerte
Ein Array von Elementknoten.

Beispiel

```
[...]
var blocks = DOM.getBlockElements();
var dProps = null, children = null;
for (var i=0; i < blocks.length; i++){
    // get the declared styles so we can see whether width
    // or height have been specified explicitly
    dProps = window.getDeclaredStyle(blocks[i]);
    // if the block has children, border-left, and padding-bottom
    // but no width or height
    if (blocks[i].hasChildNodes() && |
        issueUtils.hasBorder(blocks[i],null,"left") &&
        (parseFloat(blocks[i].getComputedStyleProp("padding-bottom")) > 0) &&
        typeof(dProps.width) == "undefined" && typeof(dProps.height) == "undefined"){
        children = blocks[i].getBlockElements();
        var hasLayout = false;
        // loop through the block-level children to see if
        // any have width or height defined. width or height on any
        // of the children of the outer block will prevent the bug.
        for (var j=0; j < children.length; j++){
            dProps = window.getDeclaredStyle(children[j]);
            if (typeof(dProps.width) != "undefined" || typeof(dProps.height) !=
            "undefined"){
            hasLayout = true;
                break;
            }
        }
    [...]
    }
}
[...]
```

dom.getInlineElements() elem.getInlineElements()

Verfügbarkeit

Dreamweaver CS3.

Beschreibung

Durchsucht das Dokument (oder das Element) nach untergeordneten Elementen mit einem inhärenten oder angegebenen Anzeigewert `'inline'`.

Argumente

Keine.

Rückgabewerte

Ein Array von Elementknoten.

Beispiel

```
[...]
var DOM = dw.getDocumentDOM();
var inEls = DOM.body.getInlineElements();
var next = null, prev = null, parent = null;
var props = null;

// look through all inline elements for replaced elements.
// if no replaced elements are found, don't bother going forward.
for (var i=0; i < inEls.length; i++){
    if (inEls[i].tagName == 'IMG' ||
        inEls[i].tagName == 'INPUT' ||
        inEls[i].tagName == 'TEXTAREA' ||
        inEls[i].tagName == 'SELECT' ||
        inEls[i].tagName == 'OBJECT'){
        // do something
    }
}
[...]
```

dom.getHeaderElements() elem.getHeaderElements()

Verfügbarkeit

Dreamweaver CS3.

Beschreibung

Durchsucht das Dokument (oder das Element) nach Header-Tags (H1 bis H6).

Argumente

Keine.

Rückgabewerte

Ein Array von Elementknoten.

Beispiel

```
var DOM = dw.getDocumentDOM();
var headers = DOM.getHeaderElements();

for (var i=0; i < headers.length; i++){
    alert(headers[i].tagName);
}
```

dom.getListElements() elem.getListElements()

Verfügbarkeit

Dreamweaver CS3.

Beschreibung

Durchsucht das Dokument (oder das Element) nach geordneten Listen, ungeordneten Listen und Definitionslisten.

Argumente
Keine.

Rückgabewerte
Ein Array von Elementknoten.

Beispiel
```
[...]
var DOM = dw.getDocumentDOM();
// get all the UL, OL, and DL elements in the document.
var lists = DOM.getListElements();
var props = null;
for (var i=0; i < lists.length; i++){
    props = window.getDeclaredStyle(lists[i]);
    if ((props.cssFloat == "left" || props.cssFloat == "right") && props.overflow == "auto"){
        // do something
    }
}
[...]
```

elem.isBlockElement()

Verfügbarkeit
Dreamweaver CS3.

Beschreibung
Überprüft, ob das Element einen inhärenten oder angegebenen Anzeigewert `'block'` hat.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert, der angibt, ob es sich bei dem Objekt um ein Element auf Blockebene handelt.

Beispiel
```
[...]
    var DOM = dw.getDocumentDOM();
    var blocks = DOM.body.getBlockElements();
    var next = null;
    for (var i=0; i < blocks.length; i++){
        // next is the node right after blocks[i]
        next = blocks[i].nextSibling;
        // if next isn't null AND next is an element node AND next is a block element,
        // we've met the "second of two consecutive block elements" test.
        if (next && (next.nodeType == 1) && next.isBlockElement()){
        // do something
    }
}
[...]
```

elem.isInlineElement()

Verfügbarkeit
Dreamweaver CS3.

Beschreibung
Überprüft, ob das Element einen inhärenten oder angegebenen Anzeigewert `'inline'` hat.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert, der angibt, ob es sich bei dem Objekt um ein Inline-Element handelt.

Beispiel

```
[...]
var DOM = dw.getDocumentDOM();
var floats = issueUtils.getFloats(DOM.body);
var next = null;
for (var i=0; i < floats.length; i++){
    next = floats[i].nextSibling;
    // if nextSibling of float is a text node or an inline element
    if (next && (next.nodeType == Node.TEXT_NODE ||
        (next.nodeType == Node.ELEMENT_NODE && next.isInlineElement()))){
    // do something
    }
}
[...]
```

elem.isHeaderElement()

Verfügbarkeit
Dreamweaver CS3.

Beschreibung
Überprüft, ob es sich bei dem Element um eines der folgenden Tags handelt: h1, h2, h3, h4, h5, h6.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert, der angibt, ob es sich bei dem Objekt um ein Header-Element handelt.

Beispiel

```
[...]
var DOM = dw.getDocumentDOM();
var floats = issueUtils.getFloats(DOM.body);
var prev = null;
// first float in the document isn't affected, so start
// at 1.
for (var i=1; i < floats.length; i++){
    prev = floats[i].previousSibling;
    // if the element before the float is a header
    if (prev && prev.isHeaderElement()){
    // do something
    }
}
[...]
```

elem.isListElement()

Verfügbarkeit

Dreamweaver CS3.

Beschreibung

Überprüft, ob es sich bei dem Element um eines der folgenden Tags handelt: ul, ol, dl.

Argumente

Keine.

Rückgabewerte

Ein boolescher Wert, der angibt, ob es sich bei dem Objekt um ein Listenelement handelt.

Beispiel

```
[...]
var DOM = dw.getDocumentDOM();
var floats = issueUtils.getFloats(DOM.body);
var prev = null, children = null;
for (var i=0; i < floats.length; i++){
    children = floats[i].childNodes;
    for (var k=0; k < children.length; k++){
        if (children[k].isListElement()){
        // do something
        }
    }
}
[...]
```

Kapitel 16: Dynamische Dokumente

Mit den Funktionen für dynamische Dokumente in Adobe® Dreamweaver® werden Vorgänge durchgeführt, die sich auf Webserver-Seiten beziehen. Zu diesen Vorgängen gehören die folgenden:

- Zurückgeben einer Eigenschaft für den im Bedienfeld „Komponenten“ ausgewählten Knoten
- Abrufen einer Liste aller Datenquellen im Benutzerdokument
- Anzeigen dynamischer Inhalte in der Entwurfsansicht
- Anwenden eines Serververhaltens auf ein Dokument
- Abrufen der Namen aller derzeit definierten Servermodelle

Funktionen für Serverkomponenten

Die Funktionen für Serverkomponenten ermöglichen den Zugriff auf den Knoten, der derzeit im Bedienfeld „Komponenten“ in der Strukturansicht „Serverkomponenten“ ausgewählt ist. Mit diesen Funktionen können Sie auch die Ansicht der Komponentenstruktur aktualisieren.

dreamweaver.serverComponents.getSelectedNode()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Gibt die in der Strukturansicht „Serverkomponenten“ derzeit ausgewählte Eigenschaft `ComponentRec` zurück.

Argumente
Keine.

Rückgabewerte
Die Eigenschaft `ComponentRec`.

dreamweaver.serverComponents.refresh()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Aktualisiert die Ansicht der Komponentenstruktur.

Argumente
Keine.

Rückgabewerte
Keine.

Datenquellenfunktionen

Datenquellendateien werden im Ordner „Configuration/DataSources“ gespeichert. Für jedes Servermodell gibt es einen eigenen Unterordner: ASP.Net/C#, ASP.Net/VisualBasic, ASP/JavaScript, ASP/VBScript, ColdFusion, JSP und PHP/MySQL. In jedem Unterordner wiederum befinden sich HTML- und EDML-Dateien, die mit den Datenquellen des jeweiligen Servermodells in Zusammenhang stehen.

Weitere Informationen über die Verwendung von Datenquellen in Dreamweaver finden Sie im Handbuch *Dreamweaver erweitern* unter „Datenquellen“.

dreamweaver.dbi.getDataSources

Verfügbarkeit
Dreamweaver UltraDev 4.

Beschreibung
Ruft die Funktion `findDynamicSources()` für alle Dateien im Ordner „Configuration/DataSources“ auf. Mit dieser Funktion können Sie eine Liste aller Datenquellen im Dokument des Benutzers erstellen. Diese Funktion durchläuft alle Dateien im Ordner „Configuration/DataSources“, ruft in jeder Datei die Funktion `findDynamicSources()` auf, verkettet alle zurückgegebenen Arrays und gibt das verkettete Array der Datenquellen zurück.

Argumente
Keine.

Rückgabewerte
Ein Array, das eine Gesamtliste aller Datenquellen enthält, die im Benutzerdokument vorhanden sind. Bei jedem Element im Array handelt es sich um ein Objekt. Alle Objekte verfügen über die folgenden Eigenschaften:

- Die Eigenschaft `title` ist der Beschriftungsstring, der rechts neben dem Symbol der übergeordneten Knoten angezeigt wird. Die Eigenschaft `title` wird immer definiert.
- Die Eigenschaft `imageFile` ist der Pfad der GIF-Datei für das Symbol, das den übergeordneten Knoten im Dialogfeld „Dynamische Daten“, „Dynamischer Text“ oder im Bedienfeld „Bindungen“ darstellt. Die Eigenschaft `imageFile` wird immer definiert.
- Die Eigenschaft `allowDelete` ist optional. Wenn diese Eigenschaft auf `false` gesetzt wird und der Benutzer im Bedienfeld „Bindungen“ auf diesen Knoten klickt, wird die Schaltfläche mit dem Minuszeichen (–) deaktiviert. Wird die Eigenschaft auf `true` gesetzt, wird die Schaltfläche mit dem Minuszeichen (–) aktiviert. Wenn die Eigenschaft nicht definiert wurde und der Benutzer auf das Element klickt, wird die Schaltfläche mit dem Minuszeichen (–) aktiviert (als wäre die Eigenschaft auf `true` gesetzt).
- Die Eigenschaft `dataSource` ist der Name der Datei, in der die Funktion `findDynamicSources()` definiert ist. Beispielsweise setzt die Funktion `findDynamicSources()` in der Datei „Session.htm“, die sich im Ordner „Configuration/DataSources/ASP_Js“ befindet, die Eigenschaft `dataSource` auf `session.htm`. Diese Eigenschaft wird immer definiert.

- Die Eigenschaft `name` bezeichnet den Namen des Serververhaltens, das zur Datenquelle `dataSource` gehört, sofern diese Datenquelle vorhanden ist. Die Eigenschaft `name` wird immer definiert, kann aber ein leerer String (`""`) sein, wenn kein Serververhalten mit der Datenquelle (z. B. Sitzungsvariable) verknüpft ist.

dw.dbi.setExpanded()

Verfügbarkeit
Dreamweaver CS3.

Beschreibung
Legt den ein ein- oder auszublenden Knoten fest.

Argumente
data-source-node-name, expanded

- *data-source-node-name* ist ein String, der den Namen der Datenquelle, die ein- oder ausgeblendet werden soll, angibt.
- *expanded* ist ein boolescher Wert, der angibt, ob der Datensatzknoten ein- oder ausgeblendet werden soll.

Rückgabewerte
Keine.

Beispiel
```
dw.dbi.setExpanded(dsName, true);        //expand the data source node
```

Extension Data Manager-Funktionen

Die in diesem Abschnitt beschriebenen APIs bilden den Extension Data Manager (EDM). Sie können auf die in den Gruppen- und Mitgliederdateien enthaltenen Daten programmtechnisch zugreifen und sie verändern, indem Sie diese Funktionen aufrufen. Der EDM arbeitet auf folgende Weise:

- Der EDM übernimmt die gesamte EDML-Datei-E/A für Gruppen- und Mitgliederdateien.
- Der EDM arbeitet als Servermodell-Filter, indem er für das aktuelle Servermodell alle Datenanforderungen ausführt.

dreamweaver.getExtDataValue()

Verfügbarkeit
Dreamweaver UltraDev 4.

Beschreibung
Diese Funktion ruft die Feldwerte für die angegebenen Knoten aus einer EDML-Datei ab.

Argumente
qualifier(s)

- Das Argument *qualifier(s)* ist eine Liste variabler Länge, die sich aus den durch Kommas getrennten Knotenbezeichnern einschließlich Gruppen- oder Mitgliedsname, Teilblock (soweit vorhanden) und Feldname zusammensetzt. Die Länge der Liste hängt vom benötigten Informationsumfang ab.

Rückgabewerte
Dreamweaver erwartet einen Feldwert. Ist kein Wert angegeben, gibt Dreamweaver den Standardwert zurück.

Beispiel
Im folgenden Beispiel wird der Attributwert „location“ für das Tag „insertText“ des Mitglieds „recordset_main“ abgerufen:

```
dw.getExtDataValue("recordset_main", "insertText", "location");
```

dreamweaver.getExtDataArray()

Verfügbarkeit
Dreamweaver UltraDev 4.

Beschreibung
Diese Funktion ruft ein Array von Werten für die angegebenen Knoten aus einer EDML-Datei ab.

Argumente
qualifier(s)

- Das Argument *qualifier(s)* ist eine Liste variabler Länge, die sich aus den durch Kommas getrennten Knotenbezeichnern einschließlich Gruppen- oder Mitgliedsname, Teilblock (soweit vorhanden) und Feldname zusammensetzt.

Rückgabewerte
Dreamweaver erwartet ein Array von Namen untergeordneter Knoten.

dreamweaver.getExtParticipants()

Verfügbarkeit
Dreamweaver UltraDev 4.

Beschreibung
Diese Funktion ruft die Liste der Mitglieder aus einer EDML-Gruppendatei oder aus EDML-Mitgliederdateien ab.

Argumente
value, qualifier(s)

- Das Argument *value* ist ein Eigenschaftenwert oder ist leer und wird ignoriert. Zum Beispiel `dreamweaver.getExtParticipants("", "participant");`

- Das Argument *qualifier(s)* ist eine Liste mit variabler Länge, die sich aus den durch Kommas getrennten Knotenbezeichnern der benötigten Eigenschaft zusammensetzt.

Rückgabewerte
Dreamweaver erwartet ein Array mit Mitgliedernamen, die die angegebene Eigenschaft haben (sofern festgelegt), und die Eigenschaft wiederum muss mit dem angegebenen Wert übereinstimmen (sofern festgelegt).

dreamweaver.getExtGroups()

Verfügbarkeit
Dreamweaver UltraDev 4.

Beschreibung
Ruft den Namen der Gruppe, die dem Namen des Serververhaltens entspricht, aus einer EDML-Gruppendatei ab.

Argumente
value, qualifier(s)

- Das Argument *value* ist ein Eigenschaftenwert oder ist leer und wird ignoriert.
- Das Argument *qualifier(s)* ist eine Liste mit variabler Länge, die sich aus den durch Kommas getrennten Knotenbezeichnern der benötigten Eigenschaft zusammensetzt.

Rückgabewerte
Dreamweaver erwartet ein Array mit Gruppennamen, die die angegebene Eigenschaft haben (sofern festgelegt), und die Eigenschaft wiederum muss mit dem angegebenen Wert übereinstimmen (sofern festgelegt).

dreamweaver.refreshExtData()

Verfügbarkeit
Dreamweaver UltraDev 4.

Beschreibung
Lädt alle Datendateien der Erweiterung erneut.

Sie können mithilfe dieser Funktion einen nützlichen Befehl erstellen, indem Sie angeben, dass Änderungen an Serververhalten-EDML-Dateien neu geladen werden, ohne dass Dreamweaver neu gestartet werden muss.

Argumente
Keine.

Rückgabewerte
Dreamweaver erwartet neu geladene Daten.

Live Data-Funktionen

Mit den folgenden Live Data-Funktionen können Sie die Menüfunktionalität simulieren:

- Die Funktion `showLiveDataDialog()` wird für das Menüelement „Ansicht“ > „Live Data-Einstellungen“ verwendet.
- Die Funktion `setLiveDataMode()` wird für die Menüelemente „Ansicht“ > „Live Data“ und „Ansicht“ > „Live Data aktualisieren“ verwendet.
- Die Funktion `getLiveDataMode()` stellt fest, ob der Live Data-Modus aktiv ist.

Die restlichen Live Data-Funktionen können Sie bei der Implementierung der API-Übersetzungsfunktion `liveDataTranslateMarkup()` verwenden.

dreamweaver.getLiveDataInitTags()

Verfügbarkeit
Dreamweaver UltraDev 1.

Beschreibung
Gibt die Initialisierungs-Tags für das derzeit aktive Dokument zurück. Bei den Initialisierungs-Tags handelt es sich um die HTML-Tags, die der Benutzer im Dialogfeld „Live Data-Einstellungen“ angibt. Diese Funktion wird normalerweise von der Funktion `liveDataTranslateMarkup()` eines Übersetzers aufgerufen, damit der Übersetzer die Tags an die Funktion `liveDataTranslate()` übergeben kann.

Argumente
Keine.

Rückgabewerte
Ein String mit den Initialisierungs-Tags.

dreamweaver.getLiveDataMode()

Verfügbarkeit
Dreamweaver UltraDev 1.

Beschreibung
Bestimmt, ob das Live Data-Fenster derzeit sichtbar ist.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn das Live Data-Fenster sichtbar ist, andernfalls `false`.

dreamweaver.getLiveDataParameters ()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Ruft die als Live Data-Einstellungen angegebenen URL-Parameter ab.

Im Live Data-Modus können Sie Webseiten in der Entwurfsphase anzeigen (als wären sie vom Anwendungsserver übersetzt und zurückgegeben worden). Da dynamische Inhalte für die Anzeige in der Entwurfsansicht generiert werden, können Sie Ihr Seitenlayout mit Live Data anzeigen und bei Bedarf anpassen.

Bevor Sie Live Data anzeigen, müssen Sie für alle URL-Parameter, auf die Sie in Ihrem Dokument verweisen, Live Data-Einstellungen eingeben. Hierdurch wird verhindert, dass der Webserver für Parameter, die sonst in der Simulation nicht definiert sind, Fehler zurückgibt.

Sie geben die URL-Parameter in Form von Name-Wert-Paaren ein. Wenn Sie in Serverskripts in Ihrem Dokument beispielsweise auf die URL-Variablen `ID` und `Name` verweisen, müssen Sie diese URL-Parameter definieren, bevor Sie Live Data anzeigen.

In Dreamweaver stehen zum Eingeben von Live Data-Einstellungen die folgenden Möglichkeiten zur Verfügung:

- Im Dialogfeld „Live Data-Einstellungen“, das Sie im Menü „Ansicht“ aufrufen können.
- Im Textfeld „URL“, das oberhalb des Dokuments angezeigt wird, wenn Sie in der Symbolleiste auf die Schaltfläche „Live Data-Ansicht“ klicken.

  Für die Parameter `ID` und `Name` können Sie die folgenden Paare eingeben:

  ```
  ID          22
  Name         Samuel
  ```

  Im Feld „URL“ werden diese Parameter im gegebenen Beispiel wie folgt angezeigt:

  ```
  http://someURL?ID=22&Name=Samuel
  ```

  Mit dieser Funktion können Sie diese Live Data-Einstellung über JavaScript abrufen.

Argumente
Keine.

Rückgabewerte
Ein Array, das die URL-Parameter für das aktuelle Dokument enthält. Das Array enthält eine gerade Anzahl von Parameterstrings. Jeweils zwei Elemente bilden ein Name-Wert-Paar für den URL-Parameter. Die Elemente an gerader Position sind die Parameternamen, die Elemente an ungerader Position sind die Werte. Beispielsweise gibt `getLiveDataParameters()` für die Parameter `ID` und `Name` des vorhergehenden Beispiels das folgende Array zurück: `['ID,'22','Name','Samuel']`.

Beispiel
Das folgende Beispiel gibt die Parameter zurück, die als Live Data-Einstellungen angegeben sind, und speichert sie in `paramsArray`:

```
var paramsArray = dreamweaver.getLiveDataParameters();
```

dreamweaver.liveDataTranslate()

Verfügbarkeit
Dreamweaver UltraDev 1.

Beschreibung
Sendet ein ganzes HTML-Dokument an einen Anwendungsserver, fordert den Server zur Ausführung der Skripts im Dokument auf und gibt dann das resultierende HTML-Dokument zurück. Diese Funktion kann ausschließlich von der Funktion `liveDataTranslateMarkup()` eines Übersetzers aufgerufen werden. Andernfalls tritt ein Fehler auf. Die Funktion `dreamweaver.liveDataTranslate()` führt die folgenden Operationen aus:

- Das animierte Bild (in der Nähe der rechten Kante des Live Data-Fensters) wird wiedergegeben.
- Es wird auf eine Benutzereingabe gewartet. Wenn der Benutzer auf das Stopp-Symbol klickt, erfolgt die Rückgabe der Funktion sofort.
- Ein Argument, das aus einem einzelnen String besteht, wird von der aufrufenden Funktion entgegengenommen. (Bei diesem String handelt es sich normalerweise um den gesamten Quellcode des Benutzerdokuments. Dieser String wird auch im nächsten Vorgang verwendet.)
- Der HTML-String aus dem Benutzerdokument wird als temporäre Datei auf dem Live Data-Server gespeichert.
- Eine HTTP-Anforderung wird an den Live Data-Server gesendet, und zwar mit den Parametern, die im Dialogfeld „Live Data-Einstellungen“ angegeben wurden.
- Die HTML-Antwort vom Live Data-Server wird entgegengenommen.
- Die temporäre Datei wird vom Live Data-Server entfernt.
- Die Wiedergabe des animierten Bildes wird gestoppt.
- Die HTML-Antwort wird an die aufrufende Funktion zurückgegeben.

Argumente
string

- Ein einziger String, der normalerweise aus dem gesamten Quellcode des aktuellen Benutzerdokuments besteht.

Rückgabewerte
Ein `httpReply`-Objekt. Dieses Objekt ist identisch mit dem Wert, der von der Funktion `MMHttp.getText()` zurückgegeben wird. Wenn der Benutzer auf das Stopp-Symbol klickt, hat der Rückgabewert `httpReply.statusCode` den Wert „200“ (Status OK). Der Rückgabewert `httpReply.data` enthält einen leeren String. Weitere Informationen zum `httpReply`-Objekt finden Sie unter „Funktionen der HTTP-API“ auf Seite 14.

dreamweaver.setLiveDataError()

Verfügbarkeit
Dreamweaver UltraDev 1.

Beschreibung
Definiert die Fehlermeldung, die angezeigt werden soll, wenn bei der Ausführung der Funktion `liveDataTranslateMarkup()` eines Übersetzers ein Fehler auftritt. Wenn das von Dreamweaver an `liveDataTranslate()` übergebene Dokument Fehler enthält, gibt der Server eine in HTML formatierte Fehlermeldung zurück. Sobald der Übersetzer (der Code, durch den `liveDataTranslate()` aufgerufen wurde) feststellt, dass der Server eine Fehlermeldung zurückgegeben hat, ruft er die Funktion `setLiveDataError()` auf, mit der die Fehlermeldung in Dreamweaver angezeigt wird. Diese Meldung wird erst angezeigt, wenn die Funktion `liveDataTranslateMarkup()` vollständig ausgeführt wurde. Dreamweaver zeigt die Beschreibung in einem Fehlerdialogfeld an. Die Funktion `setLiveDataError()` sollte ausschließlich von der Funktion `liveDataTranslateMarkup()` aufgerufen werden.

Argumente
source

- Das Argument *source* ist ein String mit Quellcode. Dieser Code wird analysiert und im Fehlerdialogfeld angezeigt.

Rückgabewerte
Keine.

dreamweaver.setLiveDataMode()

Verfügbarkeit
Dreamweaver UltraDev 1.

Beschreibung
Blendet das Live Data-Fenster ein oder aus.

Argumente
bIsVisible

- Das Argument *bIsVisible* ist ein boolescher Wert, der angibt, ob das Live Data-Fenster sichtbar sein soll. Wenn das Live Data-Fenster derzeit in Dreamweaver angezeigt wird und `true` an diese Funktion übergeben wird, hat dies die gleiche Wirkung wie das Klicken auf die Schaltfläche „Aktualisieren“.

Rückgabewerte
Keine.

dreamweaver.setLiveDataParameters()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Legt die URL-Parameter, auf die Sie in Ihrem Dokument verweisen, zur Verwendung im Live Data-Modus fest.

Im Live Data-Modus können Sie Webseiten in der Entwurfsphase anzeigen (als wären sie vom Anwendungsserver übersetzt und zurückgegeben worden). Da dynamische Inhalte für die Anzeige in der Entwurfsansicht generiert werden, können Sie Ihr Seitenlayout mit Live Data anzeigen und bei Bedarf anpassen.

Bevor Sie Live Data anzeigen, müssen Sie für alle URL-Parameter, auf die Sie in Ihrem Dokument verweisen, Live Data-Einstellungen eingeben. Hierdurch wird verhindert, dass der Webserver für Parameter, die sonst in der Simulation nicht definiert sind, Fehler zurückgibt.

Sie geben die URL-Parameter in Form von Name-Wert-Paaren ein. Wenn Sie in Serverskripts in Ihrem Dokument beispielsweise auf die URL-Variablen `ID` und `Name` verweisen, müssen Sie diese URL-Parameter definieren, bevor Sie Live Data anzeigen.

Mit dieser Funktion können Sie Live Data-Werte über JavaScript festlegen.

Argumente
liveDataString

- Das Argument *liveDataString* ist ein String, der die festzulegenden URL-Parameter in Name-Wert-Paaren enthält.

Rückgabewerte
Keine.

Beispiel
```
dreamweaver.setLiveDataParameters("ID=22&Name=Samuel")
```

dreamweaver.showLiveDataDialog()

Verfügbarkeit
Dreamweaver UltraDev 1.

Beschreibung
Zeigt das Dialogfeld „Live Data-Einstellungen“ an.

Argumente
Keine.

Rückgabewerte
Keine.

Live-Ansichtsfunktionen

Live-Ansichtsfunktionen werden für folgende Zwecke eingesetzt:

- Abrufen und Festlegen der Entwurfsansicht
- Abrufen und Festlegen der Live-Ansicht unter Verwendung des Servers
- Abrufen von Standardwerten für die Live-Ansicht
- Abrufen und Festlegen abhängiger Dateien für die Live-Ansicht
- Anzeigen der Parameter für die Live-Ansicht

dom.getDesignViewMode()

Verfügbarkeit
Dreamweaver CS4.

Beschreibung
Mit dieser Funktion wird der Ansichtsstatus oder Modus der Entwurfsansicht ermittelt. Die Entwurfsansicht kann als herkömmliche bearbeitbare Entwurfsansicht oder als Live-Ansicht angezeigt werden.

Argumente
Keine.

Rückgabewerte
Ein Stringwert. Gibt `live` zurück, wenn die Entwurfsansicht sich im Modus „Live-Ansicht“ befindet. Gibt `editable` zurück, wenn die Entwurfsansicht die herkömmliche bearbeitbare Entwurfsansicht ist.

dom.setDesignViewMode()

Verfügbarkeit
Dreamweaver CS4.

Beschreibung
Mit dieser Funktion wird der Modus der Entwurfsansicht festgelegt. Beispielsweise kann mit der Funktion die Live-Ansicht aktiviert werden.

Argumente
`mode`

- Das Argument *`mode`* ist ein String, der die Werte `live` oder `editable` enthalten kann.

Rückgabewerte
Keine.

dom.getLiveViewUsingServer()

Verfügbarkeit
Dreamweaver CS4.

Beschreibung
Mit dieser Funktion können Sie feststellen, ob die Vorschau der aktuellen Seite mithilfe eines Servers erfolgt.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn die Vorschau der aktuellen Seite mithilfe eines Servers erfolgen muss, andernfalls `false`.

dom.setLiveViewUsingServer()

Verfügbarkeit
Dreamweaver CS4.

Beschreibung
Mit dieser Funktion können Sie festlegen, ob die Vorschau einer Seite mithilfe eines Servers erfolgen kann.

Argumente
`bool`

- Das Argument *`bool`* ist ein boolescher Wert, der angibt, ob die Vorschau einer Seite mithilfe eines Servers erfolgen kann. Wenn dieser Funktion der Parameter `true` übergeben wird, kann die Vorschau der Seite mithilfe eines Servers erfolgen.

Rückgabewerte
Keine.

dom.getLiveViewDefaultsToUsingServer()

Verfügbarkeit
Dreamweaver CS4.

Beschreibung
Mit dieser Funktion wird ermittelt, ob die Vorschau mithilfe eines Servers die Standardaktion ist.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert. Wenn als Standardaktion die Vorschau von Seiten mithilfe eines Servers festgelegt wurde, ist dies der Wert `true`, andernfalls `false`.

dom.getLiveViewDependentsUsingServer()

Verfügbarkeit
Dreamweaver CS4.

Beschreibung
Mit dieser Funktion wird ermittelt, ob abhängige CSS- und JavaScript-Dateien vom Server angefordert werden.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert. Wenn abhängige CSS- und JavaScript-Dateien vom Server angefordert werden, ist dieser Wert `true`, andernfalls `false`.

dom.setLiveViewDependentsUsingServer()

Verfügbarkeit
Dreamweaver CS4.

Beschreibung
Mit dieser Funktion wird festgelegt, ob abhängige CSS- und JavaScript-Dateien vom Server angefordert werden müssen.

Argumente
`bool`

- Ein boolescher Wert, der angibt, ob abhängige CSS- und JavaScript-Dateien vom Server angefordert werden. Wenn dieser Funktion der Parameter `true` übergeben wird, werden die Dateien vom Server angefordert.

Rückgabewerte
Keine.

dom.showLiveViewParametersDialog()

Verfügbarkeit
Dreamweaver CS4.

Beschreibung
Diese Funktion wird verwendet, um das Dialogfeld mit den Parametern für die Live-Vorschau anzuzeigen.

Argumente
Keine.

Rückgabewerte
Keine.

dom.setLiveViewFollowsLinks()

Beschreibung
Mit dieser Funktion wird die Funktion zum Aufrufen von Hyperlinks für das aktuelle Dokument aktiviert. Mithilfe der Funktion zum Aufrufen von Hyperlinks kann der Benutzer im Modus „Live-Ansicht“ das Dokumentverhalten beim Klicken auf einen Hyperlink ermitteln.

Verfügbarkeit
Dreamweaver CS5.

Argumente
bool

Ein boolescher Wert, der angibt, ob die Funktion zum Aufrufen von Hyperlinks aktiviert werden soll.

Rückgabewerte
Keine.

dom.getLiveViewFollowsLinks()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Mit dieser Funktion wird ermittelt, ob die Funktion zum Aufrufen von Hyperlinks für das Dokument aktiviert ist.

Argumente
Keine.

Rückgabewerte
bool

Ein boolescher Wert, der angibt, ob die Funktion zum Aufrufen von Hyperlinks aktiviert ist.

dom.isLiveViewBrowsingHomeURI()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Mit dieser Funktion wird ermittelt, ob die im Browser in der Live-Ansicht angezeigte URL der URL entspricht, die auf der Registerkarte des Hauptdokuments angezeigt wird.

Argumente
Keine.

Rückgabewerte
bool

Ein boolescher Wert, der angibt, ob der Modus „Live-Ansicht“ aktiviert ist. Der Wert `false` wird zurückgegeben, wenn der Benutzer nach dem Aktivieren der Live-Ansicht einen Hyperlink für ein anderes Dokument aufgerufen hat.

dreamweaver.findSiteForURI()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Mit dieser Funktion wird die Site ermittelt, auf die mit dem URI verwiesen wird.

Argumente
`DWUri`

Ein URI, der auf eine lokale Datei oder eine Remote-Site verweist (angegeben als Pfad im Format `file://` oder als Hyperlink im Format `http://`).

Rückgabewerte
Objekt

Ein Objekt mit dem Kontext der Site. Das zurückgegebene Objekt verfügt über die folgenden Eigenschaften:

Eigenschaft	Beschreibung
siteName	Der Name der Site, auf die mit dem URI verwiesen wird.
localURI	Das DWUri-Objekt, das den URI der lokalen Site darstellt. Wenn der URI auf eine Remote-Site verweist, hat die Eigenschaft `localURI` den Wert „null“.

dom.browser.isCmdEnabled()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Mit dieser Funktion wird ermittelt, ob ein bestimmter Browserbefehl zur Verwendung verfügbar ist.

Argumente
String

Ein String, der den Browserbefehl darstellt. In der folgenden Tabelle sind die gültigen Strings aufgeführt:

Eigenschaft	Beschreibung
cut	Gibt an, ob die Aktion zum Ausschneiden in der Browseransicht zulässig ist.
copy	Gibt an, ob die Aktion zum Kopieren in der Browseransicht zulässig ist.
paste	Gibt an, ob die Aktion zum Einfügen in der Browseransicht zulässig ist.
clear	Gibt an, ob die Aktion zum Löschen in der Browseransicht zulässig ist.
selectAll	Gibt an, ob in der Browseransicht der gesamte Inhalt ausgewählt werden kann.
find	Gibt an, ob die Aktion zum Suchen in der Browseransicht zulässig ist.
undo	Gibt an, ob die Aktion zum Rückgängigmachen in der Browseransicht zulässig ist.

Eigenschaft	Beschreibung
redo	Gibt an, ob die Aktion zum Wiederherstellen in der Browseransicht zulässig ist.
print	Gibt an, ob die Aktion zum Drucken in der Browseransicht zulässig ist.
back	Gibt an, ob über die Navigationsleiste des Browsers die Aktion zum Zurücknavigieren verfügbar ist.
forward	Gibt an, ob über die Navigationsleiste des Browsers die Aktion zum Vorwärtsnavigieren verfügbar ist.
stop	Gibt an, ob über die Navigationsleiste des Browsers die Aktion zum Anhalten verfügbar ist.
refresh	Gibt an, ob über die Navigationsleiste des Browsers die Aktion zum Aktualisieren verfügbar ist.
setURL	Gibt an, ob in der Browseransicht eine URL festgelegt werden kann.
pageNavigationHistory	Gibt an, ob die Funktionen zum Verlauf der Seitennavigation zur Verwendung verfügbar sind. Dabei handelt es sich um folgende Funktionen: `dom.browser.isPageNavigationHistoryEnabled()`, `dom.browser.enablePageNavigationHistory()`, `dom.browser.getPageNavigationHistoryLength()`, `dom.browser.getPageNavigationHistoryPosition()`, `dom.browser.goToPageNavigationHistoryPosition()`, `dom.browser.getNavigationHistoryItem()`,`dom.browser.setHomePage()` und `dom.browser.getHomePage()`
home	Gibt an, ob über die Navigationsleiste des Browsers die Aktion zum Aufrufen der Homepage verfügbar ist.
followLinkContextMenuItem	Gibt an, ob ein Hyperlink ausgewählt und aufgerufen werden kann.

Rückgabewerte

bool

Ein boolescher Wert, der angibt, ob der entsprechende Browserbefehl zur Verwendung verfügbar ist.

dom.browser.isPageNavigationHistoryEnabled()

Verfügbarkeit

Dreamweaver CS5.

Beschreibung

Mithilfe dieser Funktion wird ermittelt, ob der Browser die Seiten verfolgt, die der Benutzer angezeigt hat.

Argumente

Keine.

Rückgabewerte

bool

Ein boolescher Wert, der angibt, ob der Browser die angezeigten Seiten verfolgt.

dom.browser.enablePageNavigationHistory()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Mit dieser Funktion wird der Browserverlauf aktiviert oder deaktiviert.

Argumente
bool

Ein boolescher Wert, der angibt, ob der Browserverlauf aktiviert oder deaktiviert werden soll. Geben Sie `true` an, um den Browserverlauf zu aktivieren.

Rückgabewerte
bool

Ein boolescher Wert: `true` bei einem erfolgreichen Vorgang.

dom.browser.getPageNavigationHistoryLength()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Mit dieser Funktion wird die Anzahl der Elemente in der Browserverlaufsliste ermittelt.

Argumente
Keine.

Rückgabewerte
int

Eine Ganzzahl, die die Anzahl der Elemente in der Verlaufsliste angibt.

dom.browser.getPageNavigationHistoryPosition()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Mit dieser Funktion wird die aktuelle Position des Benutzers in der Browserverlaufsliste ermittelt. Die aktuelle Position wird in der Regel mit dem Wert `getPageNavigationHistoryLength()-1` für die aktuellste Seite angegeben. Wenn der Benutzer jedoch in der Verlaufsliste zurück navigiert ist, ändert sich der Wert der aktuellen Position.

Argumente
Keine.

Rückgabewerte
int

Eine Ganzzahl, die die aktuelle Position des Benutzers in der Browserverlaufsliste angibt.

dom.browser.goToPageNavigationHistoryPosition()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Mit dieser Funktion wird der Browser angewiesen, zur angegebenen Position in der Verlaufsliste zu navigieren.

Argumente
int

Eine Ganzzahl, die die Position in der Browserverlaufsliste angibt.

Rückgabewerte
Keine.

dom.browser.getPageNavigationHistoryItem()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Mit dieser Funktion werden Informationen zu einem bestimmten Element der Browserverlaufsliste abgerufen.

Argumente
int

Eine Ganzzahl, die die Position in der Browserverlaufsliste angibt.

Rückgabewerte
Objekt

Ein Objekt, das die Informationen zu dem Element in der Browserverlaufsliste enthält. Das zurückgegebene Objekt verfügt über die folgenden Eigenschaften:

Eigenschaft	Beschreibung
uri	Der vom Browser für das Verlaufselement verwendete URI. Der Typ der Eigenschaft ist ein `DWUri`-Objekt.
originalUri	Der Original-URI für das Verlaufselement. Der Original-URI entspricht in der Regel dem URI-Wert. Der Typ der Eigenschaft ist ein `DWUri`-Objekt.
title	Der Titel der Seite, die angezeigt wurde. Der Typ der Eigenschaft ist ein `String`-Wert.
isPost	Gibt an, ob das Element die Formulardaten beim Laden neu sendet. Der Typ der Eigenschaft ist ein `bool`-Wert.

dom.browser.setHomePage()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Mit dieser Funktion wird der Browser angewiesen, die angegebene URL als Homepage festzulegen.

Argumente
`DWUri`

Ein `DWUri`-Objekt, das die URI-Informationen enthält.

Rückgabewerte
`bool`

Ein boolescher Wert: `true` bei einem erfolgreichen Vorgang.

dom.browser.getHomePage()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Mit dieser Funktion wird die aktuelle Homepage des Browsers abgerufen.

Argumente
Keine.

Rückgabewerte
DWUri-Objekt

Ein `DWUri`-Objekt, das die URI-Informationen enthält.

dom.browser.getSelection()

Verfügbarkeit
Dreamweaver CS4.

Beschreibung
Diese Funktion wird verwendet, um in der Live-Ansicht die aktuelle Auswahl im Browser abzurufen.

Argumente
Keine.

Rückgabewerte
Gibt ein Array mit zwei Offsets (für die Start- und die Endeposition der Auswahl im Quellcode) zurück.

dom.browser.getStatusText()

Verfügbarkeit
Dreamweaver CS4.

Beschreibung
Diese Funktion wird verwendet, um den aktuellen Statustext des Browsers abzurufen. Dies ist in der Regel ein leerer String oder z. B. der String „`Datei wird geladen …`“.

Argumente
Keine.

Rückgabewerte
Gibt den Text zurück, der im Statusbereich eines Browserfensters angezeigt wird.

dom.browser.getWindow()

Verfügbarkeit
Dreamweaver CS4.

Beschreibung
Diese Funktion wird verwendet, um das Fensterobjekt des Browser-Steuerelements abzurufen. Über das Fensterobjekt können Sie auf das Browser-DOM zugreifen.

Argumente
Keine.

Rückgabewerte
Gibt ein Objekt zurück.

dom.browserEle.loadHTML()

Verfügbarkeit
Dreamweaver CS4.

Beschreibung
Diese Funktion lädt einen HTML-String direkt in das Browser-Steuerelement. Diese Funktion ist nützlich, wenn Sie sofort einsetzbare HTML-Strings verwenden. Die HTML-Strings dürfen nicht mit einem Dokument verknüpft sein.

Argumente
Keine.

Rückgabewerte
Keine.

dom.browser.interactivityPaused

Verfügbarkeit
Dreamweaver CS4.

Beschreibung
Mit dieser Eigenschaft können Sie ermitteln, ob die Interaktivität aktiviert ist.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn die Interaktivität aktiviert ist, andernfalls `false`.

dom.browser.javaScriptEnabled

Verfügbarkeit
Dreamweaver CS4.

Beschreibung
Mit dieser Eigenschaft können Sie ermitteln, ob JavaScript aktiviert ist. Diese Eigenschaft funktioniert wie `dom.interactivityPaused()`, in diesem Fall jedoch für JavaScript.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn JavaScript aktiviert ist, andernfalls `false`.

<mm:browsercontrol>

Sie können dieses Tag in erweiterbaren Dialogfeldern verwenden, um innerhalb des Dialogfelds einen Browser anzuzeigen. Für dieses Tag gibt es keine speziellen Attribute. Sie können die Größe des Browserfensters mit CSS-Stilelementen steuern. Das vom Tag zurückgegebene DOM-Objekt hat denselben Objekttyp wie `dom.browser`. Es handelt sich jedoch nicht um dieselbe Instanz. Die Funktion `dom.browser` kann nicht in Befehlen verwendet werden. Sie müssen dazu das Browser-Objekt im DOM abrufen. Beispiel für dieses Tag:

```
<mm:browsercontol id="myBrowser" style="width: 500px; height:300px;" />
```

Im head-Bereich des Dokuments muss sich Skriptcode wie der folgende befinden:

```
var browserEle = document.getElementById("myBrowser");
alert(browserEle.getWindow().document.documentElement.outerHTML);
```

Das Browser-Steuerelement sendet außerdem zwei spezielle Ereignisse:

BrowserControlLoad Dieses Ereignis wird sofort nach dem Ladeereignis des Browsers aufgerufen, sodass Sie dem geladenen Browser-DOM eigene Elemente anfügen können.

BrowserControlBeforeNavigation Dieses Ereignis wird aufgerufen, kurz bevor der Browser eine neue Seite aufruft. Wenn das Ereignis abgebrochen wird, gilt dies auch für die Navigationsanforderung, und im Browser-Steuerelement wird weiterhin die aktuelle Seite angezeigt. Der Ereigniskontext enthält auch die angeforderte URL.

Im folgenden Beispiel wird der Ablauf dieser Ereignisse veranschaulicht:

```
var browserEle = document.getElementById("myBrowser");
browserEle.addEventListener("BrowserControlBeforeNavigation",
    function(e){ if (e.requestedLocation = "foo.com")
                 e.preventDefault(); //don't allow navigation to this site!}, true);
```

Serververhalten-Funktionen

Die Serververhalten-Funktionen wirken sich auf das Bedienfeld „Serververhalten“ aus. Dieses Bedienfeld wird mit dem Befehl „Fenster“ > „Serververhalten“ angezeigt. Mit diesen Funktionen können Sie alle Serververhalten auf einer Seite ermitteln, programmatisch ein neues Verhalten auf ein Dokument anwenden oder ein vorhandenes Verhalten modifizieren.

Hinweis: *`dw.serverBehaviorInspector` kann mit dw.sbi abgekürzt werden.*

dreamweaver.getParticipants()

Verfügbarkeit
Dreamweaver UltraDev 4.

Beschreibung
Die JavaScript-Funktion `dreamweaver.getParticipants()` ruft eine Liste der Mitglieder aus dem Dokument des Benutzers ab. Dreamweaver sucht alle Mitglieder des Serververhaltens und speichert diese Listen. Diese Funktion wird meist mit der Funktion `findServerBehaviors()` kombiniert (siehe „Serververhalten“ in *Dreamweaver erweitern*), um Instanzen eines Verhaltens im Dokument des Benutzers zu finden.

Argumente
edmlFilename

- Das Argument *edmlFilename* ist der Name der Gruppen- oder Mitgliederdatei mit den Namen der Mitglieder, die im Dokument des Benutzers gefunden werden sollen. Der String ist der Dateiname ohne die Erweiterung .edml.

Rückgabewerte
Die Funktion gibt ein Array zurück, das alle Instanzen des angegebenen Mitglieds (bzw. bei Gruppendateien alle Instanzen eines Mitglieds der Gruppe) enthält, die im Dokument des Benutzers vorkommen. Das Array enthält JavaScript-Objekte, wobei jedes dieser Elemente für je eine Instanz eines Mitglieds steht, das im Dokument des Benutzers gefunden wird. Das Array wird in der Reihenfolge sortiert, in der die Mitglieder im Dokument vorkommen. Jedes JavaScript-Objekt hat folgende Eigenschaften:

- *participantNode* ist ein Zeiger auf den Knoten des Mitglieds im Dokument des Benutzers.
- *participantName* ist der Name der EDML-Datei des Mitglieds (ohne die .edml-Erweiterung).
- *parameters* ist ein JavaScript-Objekt, das alle Parameter-Wert-Paare speichert.
- *matchRangeMin* definiert das Zeichen-Offset vom Knoten des Mitglieds im Dokument bis zum Anfang des Mitgliedsinhalts.
- *matchRangeMax* ist eine Ganzzahl des Mitglieds, die das Offset vom Anfang des Knotens des Mitglieds bis zum letzten Zeichen des Mitgliedsinhalts definiert.

dreamweaver.serverBehaviorInspector.getServerBehaviors()

Verfügbarkeit
Dreamweaver UltraDev 1.

Beschreibung
Ruft eine Liste aller Verhalten auf der Seite ab. Wenn Dreamweaver feststellt, dass die interne Liste der Serververhalten eventuell nicht mehr auf dem neuesten Stand ist, wird die Funktion `findServerBehaviors()` für alle derzeit installierten Verhalten aufgerufen. Jede Funktion gibt ein Array zurück. Dreamweaver verbindet alle Arrays zu einem einzigen Array und sortiert es in der Reihenfolge, in der die `selectedNode`-Objekte der einzelnen Verhalten im Dokument angeordnet sind. Das verbundene Array wird intern in Dreamweaver gespeichert. Die Funktion `getServerBehaviors()` gibt einen Zeiger auf das verbundene Array zurück.

Argumente
Keine.

Rückgabewerte
Ein Array von JavaScript-Objekten. Beim Aufruf von `findServerBehaviors()` werden die Objekte im Array zurückgegeben. Sie werden in der Reihenfolge sortiert, in der sie im Bedienfeld „Serververhalten“ angeordnet sind.

dreamweaver.popupServerBehavior()

Verfügbarkeit
Dreamweaver UltraDev 1.

Beschreibung
Wendet ein neues Serververhalten auf das Dokument an oder modifiziert ein vorhandenes Verhalten. Wenn der Benutzer Parameter für das Verhalten angeben muss, wird ein Dialogfeld eingeblendet.

Argumente
{behaviorName} oder {behaviorObject}

- Das optionale Argument *behaviorName* ist ein String mit dem Namen des Verhaltens, dem title-Tag einer Datei oder einem Dateinamen.
- Das optionale Argument *behaviorObject* ist ein Verhaltensobjekt.

  Wenn Sie das Argument nicht angeben, führt Dreamweaver das derzeit ausgewählte Serververhalten aus. Handelt es sich beim Argument um den Namen eines Serververhaltens, fügt Dreamweaver dieses Verhalten in die Seite ein. Wenn das Argument eines der Objekte in dem Array ist, das von der Funktion `getServerBehaviors()` zurückgegeben wird, wird ein Dialogfeld angezeigt, damit der Benutzer die Parameter für das Verhalten modifizieren kann.

Rückgabewerte
Keine.

Servermodell-Funktionen

In Dreamweaver hat jedes Dokument einen zugehörigen Dokumenttyp. Bei dynamischen Dokumenttypen weist Dreamweaver zudem ein Servermodell zu (wie ASP-JS, ColdFusion oder PHP-MySQL).

Servermodelle dienen zur Gruppierung der Funktionalität, die für eine bestimmte Servertechnologie kennzeichnend ist. Je nach dem Servermodell, das zum Dokument gehört, werden unterschiedliche Serververhalten, Datenquellen usw. angezeigt.

Anhand der Servermodell-Funktionen können Sie Folgendes ermitteln: die derzeit definierten Servermodelle, Name, Sprache und Version des aktuellen Servermodells und ob das aktuelle Servermodell einen benannten Zeichensatz (wie UTF-8) unterstützt.

Hinweis: *Dreamweaver liest alle Daten in der HTML-Datei des Servermodells und speichert diese Informationen, wenn das Servermodell zum ersten Mal geladen wird. Wenn eine Erweiterung dann Funktionen wie* `dom.serverModel.getServerName()`, `dom.serverModel.getServerLanguage()`, *und* `dom.serverModel.getServerVersion()` *aufruft, geben diese Funktionen die gespeicherten Werte zurück.*

dom.serverModel.getAppURLPrefix()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Gibt die URL des Site-Stammordners auf dem Testserver zurück. Diese URL stimmt mit der URL überein, die im Dialogfeld „Site-Definition“ auf der Registerkarte „Erweitert“ in der Kategorie „Testserver“ angegeben wurde.

Bei der Kommunikation mit dem Testserver verwendet Dreamweaver HTTP (wie ein Browser). Dabei wird diese URL für den Zugriff auf den Site-Stammordner verwendet.

Argumente
Keine.

Rückgabewerte
Ein String mit der URL des Anwendungsservers, der für Live Data- und Debug-Zwecke verwendet wird.

Beispiel
Angenommen, der Benutzer erstellt eine Site und gibt an, dass der Testserver sich auf dem lokalen Computer befindet und der Stammordner den Namen `"employeeapp"` hat. In diesem Fall wird bei einem Aufruf der Funktion `dom.serverModel.getAppURLPrefix()` der folgende String zurückgegeben: `http://localhost/employeeapp/`

dom.serverModel.getDelimiters()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Hiermit kann der JavaScript-Code das Skript-Trennzeichen für die einzelnen Servermodelle abrufen. Auf diese Weise kann der Servermodell-Code getrennt von dem vom Benutzer erstellten Code verwaltet werden.

Argumente
Keine.

Rückgabewerte
Ein Array von Objekten, wobei jedes Objekt die drei folgenden Eigenschaften enthält:

- *startPattern* ist ein regulärer Ausdruck, der dem Anfangstrennzeichen des Skripts entspricht.
- *endPattern* ist ein regulärer Ausdruck, der dem Schlusstrennzeichen des Skripts entspricht.
- Die Eigenschaft *participateInMerge* ist ein boolescher Wert, der festlegt, ob der zwischen Trennzeichen stehende Inhalt für das Zusammenführen von Codeblöcken geeignet ist (`true`) oder nicht (`false`).

dom.serverModel.getDisplayName()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Ruft den Servermodellnamen ab, der in der Benutzeroberfläche angezeigt wird.

Argumente
Keine.

Rückgabewerte
Ein String, dessen Wert der Name des Servermodells ist.

dom.serverModel.getFolderName()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Ruft den Namen des Ordners ab, der innerhalb des Ordners „Configuration“ für dieses Servermodell verwendet wird (z. B. im Unterordner „ServerModels“).

Argumente
Keine.

Rückgabewerte
Ein String, dessen Wert der Name des Ordners ist.

dom.serverModel.getServerIncludeUrlPatterns()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Gibt eine Liste mit Eigenschaften zurück, mit denen Sie auf Folgendes zugreifen können:

- Übersetzer-URL-Muster
- Dateiverweise
- Typ

Argumente
Keine.

Rückgabewerte
Eine Objektliste mit je einem Objekt pro `searchPattern`. Jedes Objekt hat die drei folgenden Eigenschaften:

Eigenschaft	Beschreibung
`pattern`	Ein regulärer JavaScript-Ausdruck, der im Feld `searchPattern` einer EDML-Datei angegeben wird. (Ein regulärer Ausdruck wird von zwei Schrägstrichen (`//`) begrenzt.)
`fileRef`	Der 1-basierte Index der untergeordneten Übereinstimmung eines regulären Ausdrucks, die dem eingeschlossenen Dateiverweis entspricht.
`type`	Der Teil des Werts `paramName`, der übrig bleibt, nachdem das Suffix „_includeUrl“ entfernt wurde. Dieser Typ wird dem type-Attribut des Tags `<MM:BeginLock>` zugewiesen. Ein Beispiel finden Sie in der Datei „Server Model SSI.htm“ im Ordner „Configuration/Translators“.

Beispiel
Das folgende Codefragment einer Mitgliederdatei veranschaulicht ein Übersetzer-Tag `searchPatterns`:

```
<searchPatterns whereToSearch="comment">
    <searchPattern paramNames=",ssi_comment_includeUrl">
        <![CDATA[/<!--\s*#include\s+(file|virtual)\s*=\s*"([^"]*)"\s*-->/i]]>
    </searchPattern>
</searchPatterns>
```

Das Suchmuster enthält einen regulären JavaScript-Ausdruck, der zwei untergeordnete Übereinstimmungen angibt (beide sind in Klammern eingeschlossen). Die erste untergeordnete Übereinstimmung bezieht sich auf den Textstring `file` oder `virtual`. Die zweite untergeordnete Übereinstimmung ist ein Dateiverweis.

Für den Zugriff auf das Übersetzer-URL-Muster muss Ihr Code wie im folgenden Beispiel aussehen:

```
var serverModel = dw.getDocumentDOM().serverModel;
var includeArray = new Array();
includeArray = serverModel.getServerIncludeUrlPatterns();
```

Der Aufruf von `serverModel.getServerIncludeUrlPatterns()` gibt die drei folgenden Eigenschaften zurück:

Eigenschaft	Rückgabewert
`pattern`	`/<!--\s*#include\s+(file|virtual)\s*=\s*"([^"]*)"\s*-->/i`
`fileRef`	`2`
`type`	`ssi_comment`

dom.serverModel.getServerInfo()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Gibt Informationen zurück, die sich konkret auf das aktuelle Servermodell beziehen. Diese Informationen sind in der HTML-Definitionsdatei des Servermodells definiert, die sich im Ordner „Configuration/ServerModels“ befindet.

Sie können die Informationen in der HTML-Definitionsdatei bearbeiten bzw. zusätzliche Variablenwerte oder Funktionen in die Datei einfügen. Beispielsweise können Sie die Eigenschaften `serverName`, `serverLanguage` und `serverVersion` ändern. Die Funktion `dom.serverModel.getServerInfo()` gibt die Informationen zurück, die der Autor des Servermodells in die Definitionsdatei einfügt.

Hinweis: *Die anderen Werte, die in den Servermodell-Standarddateien definiert sind, sind nur für die interne Verwendung vorgesehen.*

`serverName`, `serverLanguage` und `serverVersion` sind besondere Eigenschaften, da Entwickler direkt auf diese Eigenschaften zugreifen können, und zwar mithilfe der folgenden zugehörigen Funktionen:

- `dom.serverModel.getServerName()`
- `dom.serverModel.getServerLanguage()`
- `dom.serverModel.getServerVersion()`

Argumente
Keine.

Rückgabewerte
Ein JavaScript-Objekt mit verschiedenen Informationen, die sich konkret auf das aktuelle Servermodell beziehen.

dom.serverModel.getServerName()

Verfügbarkeit
Dreamweaver 1, verbessert in Dreamweaver MX.

Beschreibung
Ruft das Servermodell ab, das zum Dokument gehört, und gibt diesen Wert zurück. Der Servername unterscheidet zwischen verschiedenen Servertechnologien (wie z. B. ASP.NET und JSP), nicht aber zwischen den Sprachen einer Servertechnologie (wie z. B. ASP.NET VB und ASP.NET C#). Zu den möglichen Werten gehören `ASP`, `ASP.NET`, `Cold Fusion`, `JSP` und `PHP`.

Informationen zum Abrufen des mit dem Dokument verknüpften Servermodellnamens finden Sie unter „dom.serverModel.getDisplayName()“ auf Seite 383 und „dom.serverModel.getFolderName()“ auf Seite 384.

Hinweis: *In Dreamweaver MX oder höher liest* `dom.serverModel.getServerName()` *die Eigenschaft* `serverName` *des Objekts, das bei einem Aufruf der Funktion* `getServerInfo()` *in der Servermodell-API zurückgegeben wird.*

Argumente
Keine.

Rückgabewerte
Ein String, der den Servernamen enthält.

dom.serverModel.getServerSupportsCharset()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Bestimmt, ob das zum Dokument gehörige Servermodell den benannten Zeichensatz unterstützt.

Hinweis: *Sie können diese Funktion über die JavaScript-Ebene aufrufen. Zusätzlich ruft Dreamweaver diese Funktion auf, wenn der Benutzer die Kodierung im Dialogfeld „Seiteneigenschaften“ ändert. Wenn das Servermodell die neue Zeichenkodierung nicht unterstützt, gibt diese Funktion* `false` *zurück, und in Dreamweaver wird ein Warndialogfeld mit der Frage eingeblendet, ob die Konvertierung wirklich durchgeführt werden soll. Diese Situation tritt beispielsweise ein, wenn ein Benutzer versucht, ein ColdFusion 4.5-Dokument in UTF-8 zu konvertieren, da ColdFusion die UTF-8-Kodierung nicht unterstützt.*

Argumente
metaCharSetString

- Das Argument *metaCharSetString* ist ein String, der einen bestimmten Zeichensatz angibt. Dieser Wert entspricht dem Wert des Attributs `"charset="` des Tags `meta`, das zum Dokument gehört. Die Werte, die für ein bestimmtes Servermodell unterstützt werden, sind in der HTML-Definitionsdatei des Servermodells definiert. Diese Datei befindet sich im Ordner „Configuration/ServerModels“.

Rückgabewerte
Ein boolescher Wert: `true`, wenn das Servermodell den benannten Zeichensatz unterstützt, andernfalls `false`.

dom.serverModel.getServerVersion()

Verfügbarkeit
Dreamweaver 1, verbessert in Dreamweaver MX.

Beschreibung
Ermittelt das Servermodell, das zum Dokument gehört, und gibt diesen Wert zurück. Jedes Servermodell verfügt über die Funktion `getVersionArray()`, wie in der Servermodell-API definiert, die eine Tabelle mit Name-Version-Paaren zurückgibt.

Hinweis: *In Dreamweaver liest* `dom.serverModel.getServerVersion()` *zunächst die Eigenschaft* `serverVersion` *des Objekts, das bei einem Aufruf der Funktion* `getServerInfo()` *in der Servermodell-API zurückgegeben wird. Wenn diese Eigenschaft nicht vorhanden ist, liest* `dom.serverModel.getServerVersion()` *die Eigenschaft aus der Funktion* `getVersionArray()`.

Argumente
name

- Das Argument *name* ist ein String, der den Namen des Servermodells angibt.

Rückgabewerte
Ein String mit der Version des genannten Servermodells.

dom.serverModel.testAppServer()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Prüft, ob eine Verbindung zum Anwendungsserver hergestellt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert, der angibt, ob die Verbindung zum Anwendungsserver erfolgreich hergestellt werden konnte.

dreamweaver.getServerModels()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Ruft die Namen aller derzeit definierten Servermodelle ab. Dies sind die gleichen Namen, die in der Benutzeroberfläche im Feld „Servermodell“ des Dialogfelds „Site-Definition“ angezeigt werden.

Argumente
Keine.

Rückgabewerte
Ein String-Array. Jedes String-Element enthält den Namen eines derzeit definierten Servermodells.

Kapitel 17: Entwurf

Mit den Entwurfsfunktionen in Adobe® Dreamweaver® können Vorgänge durchgeführt werden, die die Darstellung von Dokumenten betreffen. Dazu gehören Funktionen, mit denen Sie die folgenden Aktionen ausführen können:

- Anwenden eines bestimmten CSS-Stils (Cascading Stylesheet)
- Vertikales oder horizontales Teilen eines ausgewählten Frames
- Ausrichten ausgewählter Ebenen oder Hotspots
- Wiedergeben eines ausgewählten Plug-In-Elements
- Erstellen einer Layoutzelle
- Bearbeiten von Zeilen oder Spalten einer Tabelle

CSS-Layoutfunktionen

Mit CSS-Funktionen werden CSS-Stile angewendet, entfernt, erstellt und gelöscht. Die Methoden des Objekts `dreamweaver.cssRuleTracker` gelten für die Auswahl im Bedienfeld „CSS-Regelverfolgung“ des Auswahlinspektors. Die Methoden des Objekts `dreamweaver.cssStylePalette` gelten für die Auswahl im Bedienfeld „CSS-Stile“ und nicht für die Auswahl im aktuellen Dokument.

dom.applyLayout()

Verfügbarkeit
Dreamweaver CS3.

Beschreibung
Wendet ein CSS-basiertes Layout auf das Dokument an. Der Hauptteil des Dokuments muss leer sein und bei dem Dokument muss es sich um eine Seite handeln, auf die ein Layout angewendet werden kann. Das trifft zu auf:

- Eine HTML-basierte Seite, z. B. HTML, XHTML, ColdFusion, PHP usw. (jedoch *nicht* CSS, XML, JavaScript usw.)
- Eine Seite, bei der es sich *nicht* um ein Frameset oder eine Vorlageninstanz handelt (eine Vorlage kann jedoch verwendet werden)

Argumente
layout-index, *CSS, cssFileName, preventOverwrite*

- *layout-index* ist ein nullbasierter Ganzzahlindex, der das zu verwendende Layout angibt. Dies ist ein Index für die Liste der Layouts, der verwendet wird, um die Namen (`layoutNames`) und Beschreibungen (`layoutDescriptions`) in den entsprechenden Funktionen zurückzugeben.
- *CSS* gibt an, an welcher Position das CSS-Layout eingefügt werden soll. Folgende Werte sind möglich:
  - `"embed"` – Einbetten des CSS-Codes in den Head-Bereich des Dokuments
  - `"link"` – Verknüpfung mit *cssFileName*
  - `"create_and_link"` – Speichern des CSS-Codes in *cssFileName* und Erstellen einer Verknüpfung

- "`import`" – Importieren von *`cssFileName`*
- "`create_and_import`" – Speichern des CSS-Codes in *cssFileName* und Importieren des Arguments

- *cssFileName* ist der Name der CSS-Datei, die verknüpft oder importiert und gegebenenfalls erstellt werden soll.
- Das Argument *preventOverwrite* ist ein boolescher Wert für das Erstellen einer neuen CSS-Datei: wenn der Wert `true` ist, schlägt die Funktion bei bereits vorhandener Datei fehl, wenn der Wert `false` ist, wird eine ggf. vorhandene Datei überschrieben.

Rückgabewerte
Ein boolescher Wert: `true`, wenn das Layout erfolgreich angewendet wurde, andernfalls `false`.

Beispiel
```
dw.getLayoutNames();
var theDOM = dw.getDocumentDOM();
alert (theDOM.canApplyLayout());
if (theDOM.canApplyLayout())
    theDOM.applyLayout(1, "embed");
else
    alert("can't apply layout to this doc");
```

dom.canApplyLayout()

Verfügbarkeit
Dreamweaver CS3.

Beschreibung
Überprüft, ob ein CSS-basiertes Layout auf das Dokument angewendet werden kann. Es wird überprüft, ob der Hauptteil des Dokuments leer ist und ob es sich um eine Seite handelt, auf die ein Layout angewendet werden kann. Das trifft zu auf:

- Eine hauptsächlich HTML-basierte Seite, z. B. HTML, XHTML, ColdFusion, PHP usw. (jedoch *nicht* CSS, XML, JavaScript usw.)
- Eine Seite, bei der es sich *nicht* um ein Frameset oder eine Vorlageninstanz handelt (eine Vorlage kann jedoch verwendet werden) evelyn

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn das Layout angewendet werden kann, `false`, wenn das Layout nicht angewendet werden kann.

dw.getFilesForLayout()

Verfügbarkeit
Dreamweaver CS3.

Beschreibung
Ruft die Pfade der Konfigurationsdateien für das angegebene Layout ab.

Argumente
layoutIndex

- *layoutIndex* ist ein nullbasierter Ganzzahlindex, der das Layout angibt. Dies ist ein Index für die Liste der Layouts, der verwendet wird, um die Namen (`layoutNames`) und Beschreibungen (`layoutDescriptions`) in den entsprechenden Funktionen zurückzugeben.

Rückgabewerte
Ein String-Array, das die vollständigen Pfade der HTML- und Vorschaubilddateien enthält (kann `null` sein).

dw.getLayoutNames()

Verfügbarkeit
Dreamweaver CS3.

Beschreibung
Ruft die Namen der verfügbaren CSS-basierten Layouts ab.

Argumente
Keine.

Rückgabewerte
Ein String-Array mit Layoutnamen.

dw.getLayoutDescriptions()

Verfügbarkeit
Dreamweaver CS3.

Beschreibung
Ruft die Beschreibungen der verfügbaren CSS-basierten Layouts ab.

Argumente
Keine.

Rückgabewerte
Ein String-Array mit Layoutbeschreibungen.

dom.applyCSSStyle()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Wendet den angegebenen Stil auf das betreffende Element an. Diese Funktion ist nur für das aktive Dokument gültig.

Argumente
elementNode, styleName, {classOrID}, {bForceNesting}

- Das Argument *elementNode* ist ein Elementknoten im DOM. Wenn für *elementNode* der Wert `null` oder ein leerer String (`''`) angegeben wird, wirkt sich die Funktion auf die aktuelle Auswahl aus.
- Das Argument *styleName* ist der Name eines CSS-Stils.
- Das optionale Argument *classOrID* ist das Attribut, mit dem der Stil angewendet werden soll (entweder `"class"` oder `"id"`). Wenn für das Argument *elementNode* der Wert `null` oder ein leerer String angegeben wird und die Auswahl nicht exakt von einem Tag umschlossen ist, wird der Stil mit `SPAN`-Tags angewendet. Wenn es sich bei der Auswahl um eine Einfügemarke handelt, wird der Stil heuristisch bestimmt.
- Das optionale Argument *bForceNesting* ist ein boolescher Wert, der angibt, ob ein Verschachteln zulässig ist. Wenn *bForceNesting* angegeben ist, wird in Dreamweaver ein neues `SPAN`-Tag eingefügt, anstatt Änderungen an den vorhandenen Tags im Dokument vorzunehmen. Wenn dieses Argument nicht angegeben wird, gilt der Standardwert `false`.

Rückgabewerte
Keine.

Beispiel
Mit dem folgenden Code wird der Stil `red` auf die Auswahl angewendet. Dazu werden entweder `SPAN`-Tags um die Auswahl gelegt oder ein `CLASS`-Attribut wird auf das die Auswahl umschließende Tag angewendet.

```
var theDOM = dreamweaver.getDocumentDOM('document');
theDOM.applyCSSStyle('','red');
```

dom.getElementView()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion ruft für das im Dokument ausgewählte Element die Elementansicht auf. Wenn es sich um ein normales Element handelt, sucht die Funktion `getElementView()` nach dem ersten Vorfahren des ausgewählten Elements, der entweder vollständig angezeigt wird oder ausgeblendet ist.

Argumente
Keine.

Rückgabewerte
Ein String, der den Status des ausgewählten Elements anzeigt. Folgende Werte sind zulässig:

- `"hidden"` gibt an, dass für das Element CSS-Eigenschaften festgelegt sind, die bewirken, dass in der Entwurfsansicht Inhalte teilweise oder vollständig ausgeblendet werden. Folgende CSS-Eigenschaften werden unterstützt:
  - overflow: hidden, scroll oder auto

- display: none

- `"full"` gibt an, dass das Element standardmäßig zwar `"hidden"` ist, derzeit entsprechend der Einstellung durch die Funktion `setElementView("full")` jedoch vollständig angezeigt wird.
- `"normal"` gibt an, dass das Element weder ausgeblendet ist (`"hidden"`) noch vollständig angezeigt wird (`"full"`).

Beispiel
Im folgenden Beispiel wird der Status des ausgewählten Elements zu `"full"` geändert, wenn es ausgeblendet (`"hidden"`) ist.

```
var currentDOM = dw.getDocumentDOM();
if (currentDOM && getElementView() == "hidden"){
    currentDOM.setElementView("full");
    }
```

dom.getShowDivBackgrounds()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion ruft den Status des visuellen Hilfsmittels „Layoutblock-Hintergründe“ ab.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn Layoutblock-Hintergründe aktiviert sind, andernfalls `false`.

Beispiel
Im folgenden Beispiel wird überprüft, ob das visuelle Hilfsmittel „Layoutblock-Hintergründe“ aktiviert ist. Ist das nicht der Fall, wird es aktiviert.

```
var currentDOM = dw.getDocumentDOM();
if (currentDOM.getShowDivBackgrounds() == false){
    currentDOM.setShowDivBackgrounds(true);
    }
```

dom.getShowDivBoxModel()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion ruft den Status des visuellen Hilfsmittels „Layoutblock-Box-Modell“ ab.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn das Layoutblock-Box-Modell aktiviert ist, andernfalls `false`.

Beispiel
Im folgenden Beispiel wird überprüft, ob das visuelle Hilfsmittel „Layoutblock-Box-Modell“ aktiviert ist. Ist das nicht der Fall, wird es aktiviert.

```
var currentDOM = dw.getDocumentDOM();
if (currentDOM.getShowDivBoxModel() == false){
    currentDOM.setShowDivBoxModel(true);
    }
```

dom.getShowDivOutlines()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion ruft den Status des visuellen Hilfsmittels „Layoutblock-Konturen“ ab.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn Layoutblock-Konturen aktiviert sind, andernfalls `false`.

Beispiel
Im folgenden Beispiel wird überprüft, ob das visuelle Hilfsmittel „Layoutblock-Konturen“ aktiviert ist. Ist das nicht der Fall, wird es aktiviert.

```
var currentDOM = dw.getDocumentDOM();
if (currentDOM.getShowDivOutlines() == false){
    currentDOM.setShowDivOutlines(true);
    }
```

dom.removeCSSStyle()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Entfernt das Attribut `CLASS` oder `ID` vom angegebenen Element bzw. entfernt das `SPAN`-Tag, das das angegebene Element vollständig umschließt. Diese Funktion ist nur für das aktive Dokument gültig.

Argumente
elementNode, {classOrID}

- Das Argument *elementNode* ist ein Elementknoten im DOM. Wenn für *elementNode* ein leerer String ("") angegeben wird, wirkt sich die Funktion auf die aktuelle Auswahl aus.

- Das optionale Argument *classOrID* ist das zu entfernende Attribut (entweder `"class"` oder `"id"`). Wenn das Argument *classOrID* nicht angegeben wird, gilt der Standardwert `"class"`. Wenn kein `CLASS`-Attribut für das Argument *elementNode* definiert ist, wird das `SPAN`-Tag entfernt, das das Argument *elementNode* umschließt.

Rückgabewerte
Keine.

dom.resetAllElementViews()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion setzt die Elementansicht aller Elemente im Dokument auf die ursprüngliche Ansicht zurück, indem der gesamte intern generierte CSS-Code entfernt wird.

Argumente
{forceRefresh}

- Das optionale Argument *forceRefresh* ist ein boolescher Wert, der angibt, ob die Darstellung des gesamten Dokuments aktualisiert werden soll, wenn kein interner CSS-Code vorliegt. Der Wert `true` bewirkt eine Aktualisierung. Der Standardwert ist `false`.

Rückgabewerte
Keine.

Beispiel
Im folgenden Beispiel wird die Elementansicht aller Elemente im Dokument zurückgesetzt, ohne dass die Darstellung aktualisiert wird.

```
var currentDOM = dw.getDocumentDOM();
currentDOM.resetAllElementViews(false);
```

dom.setElementView()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion legt die Elementansicht für das im Dokument ausgewählte Element fest. Wenn das Element `"normal"` ausgewählt ist, sucht die Funktion `setElementView()` nach dem ersten Vorfahren des ausgewählten Elements, der vollständig angezeigt wird (`"full"`) oder ausgeblendet ist (`"hidden"`).

Argumente
view

- Das obligatorische Argument *view* ist ein String, der das derzeit ausgewählte Element auf `"full"` oder `"hidden"` setzt. Ist das Element `"normal"` ausgewählt ist, sucht die Funktion `setElementView()` nach dem ersten Vorfahren des ausgewählten Elements, der vollständig angezeigt wird (`"full"`) oder ausgeblendet ist (`"hidden"`). Weitere Informationen finden Sie unter „dom.getElementView()“ auf Seite 392. Folgende Werte sind möglich:
  - `"full"` – Entfernt den internen CSS-Code, durch den das Element vollständig angezeigt wurde, sodass das Element wieder in den ursprünglichen Status zurückgesetzt wird.
  - `"hidden"` – Wenn das ausgewählte Element ausgeblendet ist, erzeugt Dreamweaver den erforderlichen CSS-Code, um den gesamten Inhalt anzuzeigen, und wendet dann den CSS-Code als internes Entwurfszeit-Stylesheet an.

Rückgabewerte
Keine.

Beispiel
Siehe „dom.getElementView()“ auf Seite 392.

dom.setShowDivBackgrounds()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion aktiviert oder deaktiviert das visuelle Hilfsmittel „Layoutblock-Hintergründe“.

Argumente
show

- Das obligatorische Argument *show* ist ein boolescher Wert, der angibt, ob die Layoutblock-Hintergründe aktiviert werden sollen. Wenn *show* auf `true` gesetzt ist, werden die Layoutblock-Hintergründe aktiviert.

Rückgabewerte
Keine.

Beispiel
Siehe „dom.getShowDivBackgrounds()“ auf Seite 393.

dom.setShowDivBoxModel()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion aktiviert oder deaktiviert das visuelle Hilfsmittel „Layoutblock-Box-Modell“.

Argumente
show

- Das obligatorische Argument *show* ist ein boolescher Wert, der angibt, ob das Layoutblock-Box-Modell aktiviert werden soll. Wenn *show* auf `true` gesetzt ist, wird das visuelle Hilfsmittel „Layoutblock-Box-Modell“ aktiviert.

Rückgabewerte
Keine.

Beispiel
Siehe „dom.getShowDivBoxModel()“ auf Seite 393.

dom.setShowDivOutlines()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion aktiviert oder deaktiviert das visuelle Hilfsmittel „Layoutblock-Konturen“.

Argumente
show

- Das obligatorische Argument *show* ist ein boolescher Wert, der angibt, ob die Layoutblock-Konturen aktiviert werden sollen. Wenn *show* auf `true` gesetzt ist, werden die Layoutblock-Konturen aktiviert.

Rückgabewerte
Keine.

Beispiel
Siehe „dom.getShowDivOutlines()“ auf Seite 394.

dom.getLiveViewInspectMode()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Mit dieser Funktion wird bestimmt, ob der Prüfmodus für die Live-Ansicht für das aktuelle Dokument aktiviert ist. Weitere Informationen finden Sie unter CSS in der Live-Ansicht prüfen.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der Prüfmodus für das aktuelle Dokument aktiviert ist.

dom.setLiveViewInspectMode()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Mit dieser Funktion wird der Prüfmodus für die Live-Ansicht für das aktuelle Dokument aktiviert oder deaktiviert. Weitere Informationen finden Sie unter CSS in der Live-Ansicht prüfen.

Argumente
Ein boolescher Wert. Geben Sie `true` an, um den Prüfmodus für die Live-Ansicht zu aktivieren.

Rückgabewerte
Keine.

dreamweaver.cssRuleTracker.editSelectedRule()

Verfügbarkeit
Dreamweaver MX 2004.

Beschreibung
Ermöglicht dem Benutzer, die derzeit ausgewählte Regel in der Regelverfolgung zu bearbeiten. Diese Funktion zeigt die ausgewählte Regel im CSS-Eigenschaftenraster und gegebenenfalls das Eigenschaftenraster sowie die entsprechenden schwebenden Fenster an.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dreamweaver.cssRuleTracker.canEditSelectedRule()“ auf Seite 525.

dreamweaver.cssRuleTracker.newRule()

Verfügbarkeit
Dreamweaver MX 2004.

Beschreibung
Öffnet das Dialogfeld „Neuer CSS-Stil“, in dem der Benutzer eine neue Regel erstellen kann.

Argumente
Keine.

Rückgabewerte
Keine.

dreamweaver.cssStylePalette.applySelectedStyle()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Wendet den ausgewählten Stil auf das derzeit aktive Dokument oder das damit verknüpfte Stylesheet an, abhängig von der Auswahl im Bedienfeld „CSS-Stile“.

Argumente
{pane}

- Das optionale Argument *pane* ist ein String, der den Bereich des Bedienfelds „CSS-Stile“ angibt, auf den diese Funktion angewendet wird. Mögliche Werte sind: `"stylelist"`, die Liste der Stile im Modus „Alle“, `"cascade"`, die Liste anwendbarer, relevanter Regeln im Modus „Aktuell“, `"summary"`, die Liste der Eigenschaften für die aktuelle Auswahl im Modus „Aktuell“ und `"ruleInspector"`, die bearbeitbare Liste bzw. das Raster mit den Eigenschaften im Modus „Aktuell“. Der Standardwert ist `"stylelist"`.

Rückgabewerte
Keine.

Enabler
Siehe „dreamweaver.cssStylePalette.canApplySelectedStyle()“ auf Seite 525.

dreamweaver.cssStylePalette.attachStyleSheet()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Zeigt ein Dialogfeld an, in dem der Benutzer ein Stylesheet mit dem derzeit aktiven Dokument oder einem der angefügten Stylesheets verknüpfen kann, abhängig von der Auswahl im Bedienfeld „CSS-Stile“.

Argumente
Keine.

Rückgabewerte
Keine.

dreamweaver.cssStylePalette.deleteSelectedStyle()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Löscht den derzeit im Bedienfeld „CSS-Stile“ ausgewählten Stil aus dem Dokument.

Argumente
{pane}

- Das optionale Argument *pane* ist ein String, der den Bereich des Bedienfelds „CSS-Stile“ angibt, auf den diese Funktion angewendet wird. Mögliche Werte sind: `"stylelist"`, die Liste der Stile im Modus „Alle“, `"cascade"`, die Liste anwendbarer, relevanter Regeln im Modus „Aktuell“, `"summary"`, die Liste der Eigenschaften für die aktuelle Auswahl im Modus „Aktuell“ und `"ruleInspector"`, die bearbeitbare Liste bzw. das Raster mit den Eigenschaften im Modus „Aktuell“. Der Standardwert ist `"stylelist"`.

Rückgabewerte
Keine.

Enabler
Siehe „dreamweaver.cssStylePalette.canDeleteSelectedStyle()“ auf Seite 526.

dreamweaver.cssStylePalette.duplicateSelectedStyle()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Dupliziert den Stil, der derzeit im Bedienfeld „CSS-Stile ausgewählt“ ist, und zeigt das Dialogfeld „Duplizierter Stil“ an, in dem der Benutzer dem neuen Stil einen Namen oder einen Selektor zuweisen kann.

Argumente
{pane}

- Das optionale Argument *pane* ist ein String, der den Bereich des Bedienfelds „CSS-Stile“ angibt, auf den diese Funktion angewendet wird. Mögliche Werte sind: `"stylelist"`, die Liste der Stile im Modus „Alle“, `"cascade"`, die Liste anwendbarer, relevanter Regeln im Modus „Aktuell“, `"summary"`, die Liste der Eigenschaften für die aktuelle Auswahl im Modus „Aktuell“ und `"ruleInspector"`, die bearbeitbare Liste bzw. das Raster mit den Eigenschaften im Modus „Aktuell“. Der Standardwert ist `"stylelist"`.

Rückgabewerte
Keine.

Enabler
Siehe „dreamweaver.cssStylePalette.canDuplicateSelectedStyle()“ auf Seite 526.

dreamweaver.cssStylePalette.editSelectedStyle()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Öffnet das Dialogfeld „Stildefinition“ für den Stil, der im Bedienfeld „CSS-Stile“ derzeit ausgewählt ist.

Argumente
{pane}

- Das optionale Argument *pane* ist ein String, der den Bereich des Bedienfelds „CSS-Stile“ angibt, auf den diese Funktion angewendet wird. Mögliche Werte sind: `"stylelist"`, die Liste der Stile im Modus „Alle“, `"cascade"`, die Liste anwendbarer, relevanter Regeln im Modus „Aktuell“, `"summary"`, die Liste der Eigenschaften für die aktuelle Auswahl im Modus „Aktuell“ und `"ruleInspector"`, die bearbeitbare Liste bzw. das Raster mit den Eigenschaften im Modus „Aktuell“. Der Standardwert ist `"stylelist"`.

Rückgabewerte
Keine.

Enabler
Siehe „dreamweaver.cssStylePalette.canEditSelectedStyle()“ auf Seite 527.

dreamweaver.cssStylePalette.editSelectedStyleInCodeview()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion wechselt in die Codeansicht und setzt den Mauszeiger auf den Code für den Stil, der gerade im Bedienfeld „CSS-Stile“ ausgewählt ist.

Argumente
{pane}

- Das optionale Argument *pane* ist ein String, der den Bereich des Bedienfelds „CSS-Stile“ angibt, auf den diese Funktion angewendet wird. Mögliche Werte sind: `"stylelist"`, die Liste der Stile im Modus „Alle“, `"cascade"`, die Liste anwendbarer, relevanter Regeln im Modus „Aktuell“, `"summary"`, die Liste der Eigenschaften für die aktuelle Auswahl im Modus „Aktuell“ und `"ruleInspector"`, die bearbeitbare Liste bzw. das Raster mit den Eigenschaften im Modus „Aktuell“. Der Standardwert ist `"stylelist"`.

Rückgabewerte
Keine.

Enabler
Siehe „dreamweaver.cssStylePalette.canEditSelectedStyleInCodeview()“ auf Seite 527.

dreamweaver.cssStylePalette.editStyleSheet()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Öffnet das Dialogfeld „Stylesheet bearbeiten“.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dreamweaver.cssStylePalette.canEditStyleSheet()“ auf Seite 527.

dreamweaver.cssStylePalette.getDisplayStyles()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion legt fest, ob CSS-Stile dargestellt werden. Der Standardwert ist `true`.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn CSS-Stile dargestellt werden, andernfalls `false`.

Beispiel
```
var areStylesRendered = dw.cssStylePalette.getDisplayStyles();
```

dreamweaver.cssStylePalette.getMediaType()

Verfügbarkeit
Dreamweaver MX 2004.

Beschreibung
Ruft Zielmedientypen für die Darstellung ab. Der Standardmedientyp lautet `"screen"`.

Argumente
Keine.

Rückgabewerte
Ein String, der den Zielmedientyp angibt.

Beispiel
```
var mediaType = dw.cssStylePalette.getMediaType();
```

dreamweaver.cssStylePalette.getSelectedStyle()

Verfügbarkeit
Dreamweaver 3, `fullSelector` verfügbar in Dreamweaver MX.

Beschreibung
Ruft den Namen des Stils ab, der im Bedienfeld „CSS-Stile“ ausgewählt ist.

Argumente
fullSelector

- Das Argument *fullSelector* ist ein boolescher Wert, der anzeigt, ob der vollständige Selektor oder nur die Klasse zurückgegeben werden soll. Wenn kein Argument angegeben ist, wird nur der Klassenname zurückgegeben. Der Selektor `p.class1` legt z. B. fest, dass der Stil auf alle `p`-Tags von `class1`, jedoch nicht auf das Tag `div` von `class1` angewendet wird. Ohne das Argument *fullSelector* gibt die Funktion `dreamweaver.cssStylePalette.getSelectedStyle()` nur den Klassennamen `class1` für den Selektor an. Das Argument *fullSelector* weist die Funktion an, `p.class1` anstelle von `class1` zurückzugeben.

Rückgabewerte
Wenn das Argument *fullSelector* auf `true` gesetzt ist, gibt die Funktion den vollständigen Selektor zurück. Wenn der Stylesheetknoten ausgewählt ist, wird ein leerer String zurückgegeben.

Wenn das Argument *fullSelector* auf `false` gesetzt ist oder nicht angegeben wurde, wird ein String für den Klassennamen des ausgewählten Stils zurückgegeben. Wenn der ausgewählte Stil keine Klasse hat oder ein Stylesheetknoten ausgewählt wurde, wird ein leerer String zurückgegeben.

Beispiel
Wenn der Stil `red` ausgewählt ist, wird beim Aufruf der Funktion `dw.cssStylePalette.getSelectedStyle()` der String `"red"` zurückgegeben.

dreamweaver.cssStylePalette.getStyles()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Ruft eine Liste aller class-Stile im aktiven Dokument ab. Wenn keine Argumente übergeben wurden, werden die Klassenselektornamen zurückgegeben. Wenn das Argument *bGetIDs* den Wert `true` hat, werden ID-Selektornamen zurückgegeben. In jedem Fall gilt, dass der vollständige Selektorname zurückgegeben wird, wenn das Argument *bGetFullSelector* den Wert `true` hat.

Eine HTML-Datei kann z. B. den folgenden Code enthalten:

```
<style>
.test{ background:none };
p.foo{ background:none };
#bar {background:none };
div#hello p.world {background:none};
```

Die Aufrufe in der folgenden Tabelle geben die Werte in der Ergebnisspalte zurück.

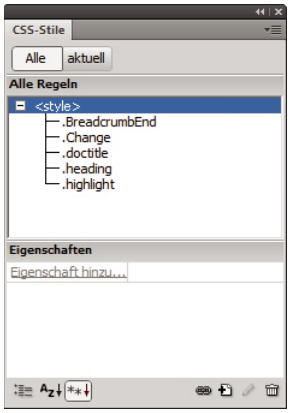
Funktionsaufruf	Ergebnis
dw.cssStylePalette.getStyles()	foo,test,world
dw.cssStylePalette.getStyles(true)	bar,hello
dw.cssStylePalette.getStyles(false, true)	p.foo,.test,div#hello p.world
dw.cssStylePalette.getStyles(true, true)	#bar,div#hello p.world

Argumente
{bGetIDs}, {bGetFullSelector}

- Das Argument *bGetIDs* ist optional. Wenn der boolesche Wert des Arguments `true` ist, gibt die Funktion ID-Selektornamen (jeweils der Teil nach dem „#“) zurück. Der Standardwert ist `false`.
- Das Argument *bGetFullSelector* ist optional. Wenn der boolesche Wert des Arguments `true` ist, wird der vollständige Selektorstring und nicht nur der Name zurückgegeben. Der Standardwert ist `false`.

Rückgabewerte
Ein Array von Strings mit den Namen aller class-Stile im Dokument.

Beispiel
Wenn das Bedienfeld „CSS-Stile“ wie in der folgenden Abbildung eingerichtet ist, wird durch Aufruf der Funktion `dreamweaver.cssStylePalette.getStyles()` ein Array mit den folgenden Strings zurückgegeben: `"BreadcrumbEnd"`, `"change"`, `"doctitle"`, `"heading"` und `"highlight"`.



dreamweaver.cssStylePalette.newStyle()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Öffnet das Dialogfeld „Neuer CSS-Stil“.

Argumente
Keine.

Rückgabewerte
Keine.

dreamweaver.cssStylePalette.renameSelectedStyle()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Benennt den Klassennamen in der ausgewählten Regel im Bedienfeld „CSS-Stile“ sowie alle Instanzen des Klassennamens in der ausgewählten Regel um.

Argumente
{pane}

- Das optionale Argument *pane* ist ein String, der den Bereich des Bedienfelds „CSS-Stile“ angibt, auf den diese Funktion angewendet wird. Mögliche Werte sind: `"stylelist"`, die Liste der Stile im Modus „Alle“, `"cascade"`, die Liste anwendbarer, relevanter Regeln im Modus „Aktuell“, `"summary"`, die Liste der Eigenschaften für die aktuelle Auswahl im Modus „Aktuell“ und `"ruleInspector"`, die bearbeitbare Liste bzw. das Raster mit den Eigenschaften im Modus „Aktuell“. Der Standardwert ist `"stylelist"`.

Rückgabewerte
Keine.

Enabler
Siehe „dreamweaver.cssStylePalette.canRenameSelectedStyle()“ auf Seite 528.

dreamweaver.cssStylePalette.setDisplayStyles()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion legt fest, ob CSS-Stile dargestellt werden sollen, und aktualisiert die Darstellung aller geöffneten Dokumente.

Argumente
display

- Das Argument *display* ist ein boolescher Wert: `true`, wenn CSS-Stile dargestellt werden, andernfalls „false“.

Rückgabewerte
Keine.

Beispiel
Im folgenden Beispiel wird die Darstellung von CSS-Stilen deaktiviert:

```
dw.cssStylePalette.setDisplayStyles(false);
```

dreamweaver.cssStylePalette.setMediaType()

Verfügbarkeit
Dreamweaver MX 2004.

Beschreibung
Legt den Zielmedientyp für die Darstellung fest. Aktualisiert die Darstellung aller geöffneten Dokumente.

Argumente
mediaType

- Das Argument *mediaType* gibt den neuen Zielmedientyp an.

Rückgabewerte
Keine.

Beispiel

```
dw.cssStylePalette.setMediaType("print");
```

dreamweaver.getBlockVisBoxModelColors()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion ruft die Farben für die Darstellung des Box-Modells für einen ausgewählten Block ab, wenn das visuelle Hilfsmittel „Layoutblock-Box-Modell“ aktiviert ist.

Argumente
Keine.

Rückgabewerte
Ein String-Array mit zwei Strings:

- `marginColor`, der hexadezimale Wert der RGB-Farbe im Format #RRGGBB.
- `paddingColor`, der hexadezimale Wert der RGB-Farbe im Format #RRGGBB.

Beispiel
Im folgenden Beispiel werden die Werte der Rand- und der Auffüllungsfarbe überprüft. Ist keiner der Werte Weiß, werden beide auf Weiß gesetzt.

```
var boxColors = dreamweaver.getBlockVisBoxModelColors();
if ((boxColors[0] != "#FFFFFF") || (boxColors[1] != "#FFFFFF")){
        currentDOM.setBlockVisBoxModelColors("#FFFFFF", "#FFFFFF");
        }
```

dreamweaver.getBlockVisOutlineProperties()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion ruft die Kontureneigenschaften für die visuellen Hilfsmittel zur Blockdarstellung ab.

Argumente
forWhat

- Das obligatorische Argument *forWhat* ist ein String. Mögliche Werte sind `"divs"`, `"selectedDiv"` oder `"layers"`. Wenn das Argument *forWhat* den Wert `"divs"` hat, gibt die Funktion die verwendeten Eigenschaften für das visuelle Hilfsmittel zurück, mit dem die Konturen aller Layoutblöcke angezeigt werden. Wenn das Argument *forWhat* den Wert `"selectedDiv"` hat, gibt die Funktion die verwendete Eigenschaft für das visuelle Hilfsmittel zurück, mit dem die Konturen der ausgewählten Layoutblöcke angezeigt werden. Der Wert `"layers"` gibt Ebenen an.

Rückgabewerte
Ein String-Array mit drei Strings:

- `color`, der hexadezimale Wert der RGB-Farbe im Format #RRGGBB
- `width`, die Breite in Pixel
- `style` mit dem Wert `"SOLID"`, `"DOTTED"`, `"DASHED"` oder `"OUTSET"`

Beispiel
Im folgenden Beispiel werden die Kontureneigenschaften für `"divs"` abgerufen und der Konturenstil `"SOLID"` zugewiesen:

```
var outlineStyle = dw.getBlockVisOutlineProperties("divs");
if (outlineStyle[2] != "SOLID"){
    dw.setBlockVisOutlineProperties("divs", outlineStyle[0], outlineStyle[1], "SOLID");
    }
```

dreamweaver.getDivBackgroundColors()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion ruft die vom visuellen Hilfsmittel „Layoutblock-Hintergründe“ verwendeten Farben ab.

Argumente
Keine.

Rückgabewerte
Ein String-Array mit 16 Farben. Jede Farbe wird als Hexadezimalwert der RGB-Farbe im Format #RRGGBB angegeben.

Beispiel
Im folgenden Beispiel werden die vom visuellen Hilfsmittel „Layoutblock-Hintergründe“ verwendeten Farben abgerufen:

```
var backgroundColors = dreamweaver.getDivBackgroundColors();
```

dreamweaver.setBlockVisOutlineProperties()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion legt die Kontureneigenschaften für die visuellen Hilfsmittel zur Blockdarstellung fest.

Argumente
forWhat, color, width, {style}

- Das obligatorische Argument *forWhat* ist ein String, der angibt, für welche Elemente die angegebene Farbe und Breite verwendet werden. Mögliche Werte sind `"divs"`, `"selectedDiv"` oder `"layers"`. Wenn der Wert `"layers"` ausgewählt ist, werden die angegebene Farbe und Breite für die Konturen aller Ebenen verwendet, falls das visuelle Hilfsmittel „Layoutblock-Konturen“ aktiviert ist. Wenn der Wert `"divs"` ausgewählt ist, werden mit den Argumenten *color* und *width* die Konturen aller div-Blöcke und anderen Layoutblöcke angezeigt. Wenn der Wert `"selectedDiv"` ausgewählt ist, werden mit den Argumenten *color* und *width* die Konturen aller ausgewählten div- oder Layoutblöcke angezeigt.
- Das obligatorische Argument *color* ist ein String mit einem Hexadezimalwert, der die RGB-Farbe im Format #RRGGBB angibt.
- Das obligatorische Argument *width* ist eine Ganzzahl, die die Stärke der Kontur in Pixel angibt.
- Das optionale Argument *style* ist ein String, der den Stil der Kontur angibt. Mögliche Werte sind `"SOLID"`, `"DOTTED"`, `"DASHED"` und `"OUTSET"`. Der Wert `"OUTSET"` ist nur auf Ebenen anwendbar. Dieses Argument wird ignoriert, wenn das Argument *forWhat* den Wert `"selectedDiv"` hat.

Rückgabewerte
Keine.

Beispiel
Siehe „dreamweaver.getBlockVisOutlineProperties()“ auf Seite 407.

dreamweaver.setDivBackgroundColors()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion legt die vom visuellen Hilfsmittel „Layoutblock-Hintergründe“ verwendeten Farben fest.

Argumente
colors

- Das obligatorische Argument *colors* ist ein String-Array mit allen Farben als Hexadezimalwert im Format #RRGGBB. Das Array muss 16 Farben enthalten.

Rückgabewerte
Keine.

Beispiel
Mit dem folgenden Codebeispiel wird sichergestellt, dass nicht mehr als 16 Farben als div-Hintergrundfarben angegeben wurden. Wenn mehr als 16 Farben angegeben sind, werden die als Hintergrundfarben verwendeten Farben auf Graustufen gesetzt.

```
var currentDOM = dw.getDocumentDOM();
var divColors = currentDOM.getDivBackgroundColors("divs");
var shadesOfGray = new Array["#000000", "#111111", "#222222", "#333333",¬
    "#444444", "#555555", "#666666", "#777777", "#888888", "#999999", ¬
    "#AAAAAA", "#BBBBBB", "#CCCCCC", "#DDDDDD", "#EEEEEE", "#FFFFFF"]¬
var howManyColors = divColors.length;
if howManyColors <= 16{
    for (var i = 0; i < howManyColors; i++)
        {
            currentDOM.setDivBackgroundColors("divs", shadeOfGray[i]);
        }
```

dreamweaver.getSelectedStyleIsDisabled()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Diese Funktion ruft den Status des ausgewählten Stils ab, unabhängig davon, ob die ausgewählte CSS-Deklaration deaktiviert ist.

Mit der Funktion „CSS-Eigenschaft deaktivieren/aktivieren“ können Sie CSS-Codeabschnitte aus dem Bedienfeld „CSS-Stile“ auskommentieren, ohne direkt im Code Änderungen vornehmen zu müssen. Wenn Sie CSS-Codeabschnitte auskommentieren, können Sie sehen, welche Auswirkungen bestimmte Eigenschaften und Werte auf Ihre Seite haben. Wenn Sie eine CSS-Eigenschaft deaktivieren, werden dieser Eigenschaft CSS-Kommentar-Tags und die Beschriftung `[disabled]` (deaktiviert) hinzugefügt.

Weitere Informationen finden Sie unter CSS-Eigenschaft deaktivieren/aktivieren.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der ausgewählte Stil deaktiviert ist.

dreamweaver.setSelectedStyleIsDisabled()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Diese Funktion legt den Status des ausgewählten Stils fest.

Argumente
Ein boolescher Wert zum Aktivieren oder Deaktivieren des ausgewählten Stils. Geben Sie `true` an, um den ausgewählten Stil zu deaktivieren.

Rückgabewerte
Keine.

dreamweaver.deleteAllDisabled()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Diese Funktion löscht alle deaktivierten Deklarationen in der ausgewählten CSS-Regel.

Argumente
Keine.

Rückgabewerte
Keine.

dreamweaver.enableAllDisabled()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Diese Funktion aktiviert alle deaktivierten Deklarationen in der ausgewählten CSS-Regel.

Argumente
Keine.

Rückgabewerte
Keine.

dreamweaver.canDisableSelectedStyle()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Diese Funktion überprüft die aktuelle Auswahl, um zu ermitteln, ob der ausgewählte Stil deaktiviert werden kann.

Argumente
pane. Ein String, der den Fensterbereich darstellt. Dieses Argument ist optional. Folgende Werte sind möglich:

- `styleList` – Bereich „Alle Regeln“ im Modus „Alle“. Dies ist der Standardwert.
- `summary` – Bereich „Zusammenfassung“ im Modus „Aktuell“.
- `cascade` – Bereich „Regeln“ im Modus „Aktuell“.
- `ruleInspector` – Bereich „Eigenschaften“ im Modus „Alle“ oder „Aktuell“.

Weitere Informationen zu den Modi des Bedienfelds „CSS-Stile“ finden Sie unter Bedienfeld „CSS-Stile“.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der ausgewählte Stil deaktiviert werden kann.

dreamweaver.canDeleteAllDisabled()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Diese Funktion überprüft die aktuelle Auswahl im Fensterbereich, um zu ermitteln, ob die Funktion „dreamweaver.deleteAllDisabled()“ auf Seite 410 ausgeführt werden kann.

Argumente
pane. Ein String, der den Fensterbereich darstellt. Dieses Argument ist optional. Die möglichen Werte entsprechen den für die Funktion „dreamweaver.canDisableSelectedStyle()“ auf Seite 411 angegebenen Werten.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der Befehl zur Verwendung verfügbar ist.

dreamweaver.canEnableAllDisabled()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Diese Funktion überprüft die aktuelle Auswahl im Fensterbereich, um zu ermitteln, ob die Funktion „dreamweaver.enableAllDisabled()“ auf Seite 410 ausgeführt werden kann.

Argumente
pane. Ein String, der den Fensterbereich darstellt. Dieses Argument ist optional. Die möglichen Werte entsprechen den für die Funktion „dreamweaver.canDisableSelectedStyle()“ auf Seite 411 angegebenen Werten.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der Befehl zur Verwendung verfügbar ist.

Funktionen für Frames und Framesets

Mit den Funktionen für Frames und Framesets werden zwei Aufgaben ausgeführt: Abrufen der Namen der Frames in einem Frameset und Teilen von Frames.

dom.getFrameNames()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Ruft eine Liste aller benannten Frames im Frameset ab.

Argumente
Keine.

Rückgabewerte
Ein String-Array, in dem jeder String der Name eines Frames im aktuellen Frameset ist. Unbenannte Frames werden übergangen. Wenn keiner der Frames im Frameset benannt ist, wird ein leeres Array zurückgegeben.

Beispiel
Für ein Dokument mit vier Frames (von denen zwei benannt sind) wird bei Aufruf der Funktion `dom.getFrameNames()` beispielsweise ein Array mit den folgenden Strings zurückgegeben:

- `"navframe"`
- `"main_content"`

dom.isDocumentInFrame()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Gibt an, ob das aktuelle Dokument innerhalb eines Framesets angezeigt wird.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn sich das Dokument in einem Frameset befindet, andernfalls `false`.

dom.saveAllFrames()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Wenn ein Dokument ein Frameset ist oder sich in einem Frameset befindet, werden mit dieser Funktion alle Frames und Framesets aus dem Dokumentfenster gespeichert. Wenn sich das angegebene Dokument nicht in einem Frameset befindet, wird mit dieser Funktion das Dokument gespeichert. Für Dokumente, die noch nicht gespeichert wurden, wird das Dialogfeld „Speichern unter“ geöffnet.

Argumente
Keine.

Rückgabewerte
Keine.

dom.splitFrame()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Teilt den ausgewählten Frame vertikal oder horizontal.

Argumente
splitDirection

- Das Argument *splitDirection* ist ein String, der einen der folgenden Richtungsparameter angeben muss: `"up"`, `"down"`, `"left"` oder `"right"`.

Rückgabewerte
Keine.

Enabler
Siehe „dom.canSplitFrame()“ auf Seite 515.

Funktionen für Ebenen und Imagemaps

Mit den Funktionen für Ebenen und Imagemaps können Ebenen und Imagemap-Hotspots ausgerichtet, in der Größe geändert und verschoben werden. In den Funktionsbeschreibungen ist angegeben, ob sich die jeweilige Funktion auf Ebenen oder auf Hotspots bezieht.

dom.align()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Richtet die ausgewählten Ebenen oder Hotspots links, rechts, oben oder unten aus.

Argumente
alignDirection

- Das Argument *alignDirection* ist ein String, der den Rand angibt, an dem die Ebenen oder Hotspots ausgerichtet werden sollen (`"left"`, `"right"`, `"top"` oder `"bottom"`).

Rückgabewerte
Keine.

Enabler
Siehe „dom.canAlign()“ auf Seite 506.

dom.arrange()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Verschiebt die ausgewählten Hotspots in die angegebene Richtung.

Argumente
toBackOrFront

- Mit dem Argument *toBackOrFront* wird die Richtung angegeben, in die die Hotspots verschoben werden, entweder nach vorn oder nach hinten.

Rückgabewerte
Keine.

Enabler
Siehe „dom.canArrange()“ auf Seite 507.

dom.makeSizesEqual()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Gleicht die ausgewählten Ebenen oder Hotspots in der Höhe und/oder in der Breite an. Für die Größe ist die zuletzt ausgewählte Ebene oder der zuletzt ausgewählte Hotspot maßgeblich.

Argumente
bHoriz, bVert

- Das Argument *bHoriz* ist ein boolescher Wert, der angibt, ob die Größe der Ebenen oder Hotspots in horizontaler Richtung geändert werden soll.
- Das Argument *bHoriz* ist ein boolescher Wert, der angibt, ob die Größe der Ebenen oder Hotspots in vertikaler Richtung geändert werden soll.

Rückgabewerte
Keine.

dom.moveSelectionBy()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Verschiebt die ausgewählten Ebenen oder Hotspots horizontal und vertikal um die angegebene Anzahl Pixel.

Argumente
x, y

- Das Argument *x* gibt an, um wie viele Pixel die Auswahl horizontal verschoben werden soll.
- Das Argument *y* gibt an, um wie viele Pixel die Auswahl vertikal verschoben werden soll.

Rückgabewerte
Keine.

dom.resizeSelectionBy()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Ändert die Größe der ausgewählten Ebene bzw. des ausgewählten Hotspots.

Argumente
left, top, bottom, right

- Das Argument *left* ist die neue Position der linken Ebenen- oder Hotspot-Begrenzung.
- Das Argument *top* ist die neue Position der oberen Ebenen- oder Hotspot-Begrenzung.
- Das Argument *bottom* ist die neue Position der unteren Ebenen- oder Hotspot-Begrenzung.

- Das Argument *right* ist die neue Position der rechten Ebenen- oder Hotspot-Begrenzung.

Rückgabewerte
Keine.

Beispiel
Wenn die ausgewählte Ebene die abgebildeten Abmessungen aufweist, werden beim Aufruf von `dw.getDocumentDOM().resizeSelectionBy(10,30,30,10)` die Positionswerte „Links“ zu 40, „Oben“ zu 20, „Breite“ zu 240 und „Höhe“ zu 240 geändert.

dom.setLayerTag()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Gibt das HTML-Tag an, mit dem die ausgewählten Ebenen definiert werden.

Argumente
tagName

- Das Argument *tagName* muss `"layer"`, `"ilayer"`, `"div"` oder `"span"` lauten.

Rückgabewerte
Keine.

Funktionen für die Layout-Umgebung

Diese Funktionen beziehen sich auf die Einstellungen zum Bearbeiten von Dokumenten. Mithilfe dieser Funktionen können Quelle, Position und Transparenz von Tracing-Bildern geändert, Linealursprung und Linealeinheiten abgerufen und festgelegt, Raster aktiviert und deaktiviert und Rastereinstellungen geändert sowie die Wiedergabe von Plug-Ins gestartet oder beendet werden.

dom.getRulerOrigin()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Ruft den Ursprung des Lineals ab.

Argumente
Keine.

Rückgabewerte
Ein Array mit zwei Ganzzahlen. Der erste Wert enthält die *x*-Koordinate des Ursprungs, der zweite die *y*-Koordinate. Beide Werte sind in Pixel angegeben.

dom.getRulerUnits()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Ruft die aktuellen Linealeinheiten ab.

Argumente
Keine.

Rückgabewerte
Ein String, der einen der folgenden Werte enthält:

- `"in"`
- `"cm"`
- `"px"`

dom.getTracingImageOpacity()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Ruft die Transparenzeinstellung für das Tracing-Bild des Dokuments ab.

Argumente
Keine.

Rückgabewerte
Ein Wert zwischen 0 und 100 bzw. kein Rückgabewert, wenn keine Transparenz festgelegt ist.

Enabler
Siehe „dom.hasTracingImage()“ auf Seite 516.

dom.loadTracingImage()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Öffnet das Dialogfeld „Bildquelle auswählen“. Wenn der Benutzer ein Bild auswählt und auf „OK“ klickt, wird das Dialogfeld „Seiteneigenschaften“ mit dem Tracing-Bild angezeigt.

Argumente
Keine.

Rückgabewerte
Keine.

dom.playAllPlugins()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Gibt den gesamten Plug-In-Inhalt des Dokuments wieder.

Argumente
Keine.

Rückgabewerte
Keine.

dom.playPlugin()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Gibt das ausgewählte Plug-In-Element wieder.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dom.canPlayPlugin()“ auf Seite 513.

dom.setRulerOrigin()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Legt den Ursprung des Lineals fest.

Argumente
xCoordinate, yCoordinate

- Das Argument *xCoordinate* ist ein Wert in Pixel auf der horizontalen Achse.
- Das Argument *yCoordinate* ist ein Wert in Pixel auf der vertikalen Achse.

Rückgabewerte
Keine.

dom.setRulerUnits()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Legt die aktuellen Linealeinheiten fest.

Argumente
units

- Das Argument *units* muss `"px"`, `"in"` oder `"cm"` lauten.

Rückgabewerte
Keine.

dom.setTracingImagePosition()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Verschiebt die obere linke Ecke des Tracing-Bilds an die angegebene Position. Wenn keine Argumente angegeben werden, wird das Dialogfeld „Position des Tracing-Bilds einstellen“ angezeigt.

Argumente
x, y

- Das Argument *x* gibt den Wert in Pixel für die horizontale Koordinate an.
- Das Argument *y* gibt den Wert in Pixel für die vertikale Koordinate an.

Rückgabewerte
Keine.

Enabler
Siehe „dom.hasTracingImage()“ auf Seite 516.

dom.setTracingImageOpacity()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Legt die Transparenz des Tracing-Bilds fest.

Argumente
opacityPercentage

- Das Argument *opacityPercentage* muss ein Wert zwischen 0 und 100 sein.

Rückgabewerte
Keine.

Enabler
Siehe „dom.hasTracingImage()“ auf Seite 516.

Beispiel
Mit dem folgenden Code wird die Transparenz des Tracing-Bilds auf 30 % gesetzt:

```
dw.getDocumentDOM().setTracingOpacity('30');
```

dom.snapTracingImageToSelection()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Richtet die obere linke Ecke des Tracing-Bilds an der oberen linken Ecke der aktuellen Auswahl aus.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dom.hasTracingImage()“ auf Seite 516.

dom.stopAllPlugins()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Stoppt die Wiedergabe des gesamten Plug-In-Inhalts des Dokuments.

Argumente
Keine.

Rückgabewerte
Keine.

dom.stopPlugin()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Stoppt die Wiedergabe des ausgewählten Plug-In-Elements.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert, der angibt, ob die Auswahl derzeit mit einem Plug-In wiedergegeben wird.

Enabler
Siehe „dom.canStopPlugin()“ auf Seite 515.

dreamweaver.arrangeFloatingPalettes()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Verschiebt die sichtbaren schwebenden Bedienfelder an ihre Standardpositionen.

Argumente
Keine.

Rückgabewerte
Keine.

dreamweaver.showGridSettingsDialog()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Öffnet das Dialogfeld „Rastereinstellungen“.

Argumente
Keine.

Rückgabewerte
Keine.

Funktionen für die Layoutansicht

Diese Funktionen beziehen sich auf Vorgänge, mit denen die Layoutelemente in einem Dokument geändert werden. Sie wirken sich auf die Einstellungen für Tabellen, Spalten und Zellen aus, u. a. auf Position, Eigenschaften und Darstellung.

dom.getClickedHeaderColumn()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Wenn der Benutzer in der Kopfzeile einer Tabelle in der Layoutansicht auf eine Menüschaltfläche klickt und dadurch das Kopfzeilenmenü der Tabelle aufgerufen wird, gibt diese Funktion den Index der Spalte zurück, auf die der Benutzer geklickt hat. Wenn das Kopfzeilenmenü der Tabelle nicht sichtbar ist, ist das Ergebnis undefiniert.

Argumente
Keine.

Rückgabewerte
Eine Ganzzahl, die den Index der Spalte angibt.

dom.getShowLayoutTableTabs()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Legt fest, ob das aktuelle Dokument in der Layoutansicht Registerkarten für Layouttabellen enthält.

Argumente
Keine.

Rückgabewerte
`true`, wenn das aktuelle Dokument in der Layoutansicht Registerkarten für Layouttabellen enthält, andernfalls `false`.

dom.getShowLayoutView()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Legt die Ansicht des aktuellen Dokuments fest, entweder die Layoutansicht oder die Standardansicht.

Argumente
Keine.

Rückgabewerte
`true`, wenn sich das aktuelle Dokument in der Layoutansicht befindet, `false`, wenn sich das Dokument in der Standardansicht befindet.

dom.getShowBlockBackgrounds()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion ruft den Status des visuellen Hilfsmittels ab, das Hintergrundfarben für alle Blöcke oder div-Tags festlegt.

Argumente
allblocks

- Das obligatorische Argument *allblocks* ist ein boolescher Wert. Setzen Sie den Wert auf `true`, wenn die Funktion nur auf div-Tags angewendet werden soll. Setzen Sie den Wert auf `false`, wenn die Funktion auf alle Blockelemente angewendet werden soll.

Rückgabewerte
Ein boolescher Wert. Wenn dieser `true` ist, werden Hintergründe erzwungen. Wenn der Wert `false` ist, werden keine Hintergründe erzwungen.

Beispiel
Im folgenden Beispiel wird zunächst geprüft, ob Hintergrundfarben für alle Blöcke erzwungen werden. Ist dies nicht der Fall, werden Hintergrundfarben für alle Blöcke erzwungen.:

```
var currentDOM = dw.getDocumentDOM();
if (currentDOM.getShowBlockBackgrounds(false) == false){
    currentDOM.setShowBlockBackgrounds(false);
    }
```

dom.getShowBlockBorders()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion ruft den Status des visuellen Hilfsmittels ab, das Rahmen für alle Blöcke oder div-Tags zeichnet.

Argumente
allblocks

- Das obligatorische Argument *allblocks* ist ein boolescher Wert. Setzen Sie den Wert auf `true`, wenn der Status nur für div-Tags abgerufen werden soll. Setzen Sie den Wert auf `false`, wenn der Status für alle Blockelemente abgerufen werden soll.

Rückgabewerte
Ein boolescher Wert. Wenn dieser `true` ist, werden Rahmen angezeigt. Wenn der Wert `false` ist, werden keine Rahmen angezeigt.

Beispiel
Im folgenden Beispiel wird überprüft, ob das visuelle Hilfsmittel für Blockrahmen aktiviert ist. Ist dies nicht der Fall, wird es aktiviert.

```
var currentDOM = dw.getDocumentDOM();
if (currentDOM.getShowBlockBorders(false) == false){
    currentDOM.setShowBlockBorders(true);
    }
```

dom.getShowBlockIDs()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion ruft den Status des visuellen Hilfsmittels ab, das ID- und Klasseninformationen für alle Blöcke oder div-Tags anzeigt.

Argumente
allblocks

- Das obligatorische Argument *allblocks* ist ein boolescher Wert. Setzen Sie den Wert auf `true`, wenn ID und Klasse nur für div-Tags angezeigt werden sollen. Setzen Sie den Wert auf `false`, wenn ID und Klasse für alle Blockelemente angezeigt werden sollen.

Rückgabewerte
Ein boolescher Wert: Wenn dieser `true` ist, werden IDs angezeigt. Wenn der Wert `false` ist, werden keine IDs angezeigt.

Beispiel
Im folgenden Beispiel wird geprüft, ob die Block-IDs angezeigt werden. Ist dies nicht der Fall, werden sie angezeigt.

```
var currentDOM = dw.getDocumentDOM();
if (currentDOM.getShowBlockIDs(false) == false){
    currentDOM.setShowBlockIDs(true);
    }
```

dom.getShowBoxModel()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion aktiviert oder deaktiviert das visuelle Hilfsmittel, mit dem das vollständige Box-Modell des ausgewählten Blocks farblich dargestellt wird.

Argumente
Keine.

Rückgabewerte
Keine.

Beispiel
Im folgenden Beispiel wird geprüft, ob das vollständige Box-Modell des ausgewählten Blocks in Farbe angezeigt wird. Ist dies nicht der Fall, wird es in Farbe angezeigt.

```
var currentDOM = dw.getDocumentDOM();
if (currentDOM.getShowBoxModel() == false){
    currentDOM.setShowBoxModel(true);
    }
```

dom.setShowBlockBackgrounds()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion aktiviert oder deaktiviert das visuelle Hilfsmittel, das Hintergrundfarben für alle Blöcke oder div-Tags festlegt.

Argumente
allblocks

- Das obligatorische Argument *allblocks* ist ein boolescher Wert. Setzen Sie den Wert auf `true`, wenn Hintergrundfarben nur auf div-Tags angewendet werden sollen. Setzen Sie den Wert auf `false`, wenn Hintergrundfarben auf alle Blockelemente angewendet werden sollen.

Rückgabewerte
Keine.

Beispiel
Siehe „dom.getShowBlockBackgrounds()“ auf Seite 423.

dom.setShowBlockBorders()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion ruft den Status des visuellen Hilfsmittels ab, das Rahmen für alle Blöcke oder div-Tags zeichnet.

Argumente
allblocks

- Das obligatorische Argument *allblocks* ist ein boolescher Wert. Setzen Sie den Wert auf `true`, wenn Rahmen nur auf div-Tags angewendet werden sollen. Setzen Sie den Wert auf `false`, wenn Rahmen auf alle Blockelemente angewendet werden sollen.

Rückgabewerte
Keine.

Beispiel
Siehe „dom.getShowBlockBorders()“ auf Seite 424.

dom.setShowBlockIDs()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion aktiviert oder deaktiviert das visuelle Hilfsmittel, das die ID und die Klasse für alle Blöcke oder div-Tags anzeigt.

Argumente
allblocks

- Das obligatorische Argument *allblocks* ist ein boolescher Wert. Setzen Sie den Wert auf `true`, wenn ID und Klasse nur für div-Tags angezeigt werden sollen. Setzen Sie den Wert auf `false`, wenn ID und Klasse für alle Blockelemente angezeigt werden sollen.

Rückgabewerte
Keine.

Beispiel
Siehe „dom.getShowBlockIDs()“ auf Seite 424.

dom.setShowBoxModel()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion legt den Status des visuellen Hilfsmittels fest, das das vollständige Box-Modell des ausgewählten Blocks farblich darstellt.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn das Box-Modell angezeigt wird, `false`, wenn es nicht angezeigt wird.

Beispiel
Siehe „dom.getShowBoxModel()“ auf Seite 425.

dom.setShowLayoutTableTabs()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Legt fest, dass im aktuellen Dokument Registerkarten für Layouttabellen angezeigt werden, wenn sich das Dokument in der Layoutansicht befindet. Durch diese Funktion wird das Dokument nicht in der Layoutansicht angezeigt.

Argumente
bShow

- Das Argument *bShow* gibt an, ob Registerkarten für Layouttabellen angezeigt werden sollen, wenn sich das aktuelle Dokument in der Layoutansicht befindet. Wenn *bShow* den Wert `true` hat, werden Registerkarten angezeigt. Wenn *bShow* den Wert `false` hat, werden keine Registerkarten angezeigt.

Rückgabewerte
Keine.

dom.setShowLayoutView()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Zeigt das aktuelle Dokument in der Layoutansicht an, wenn *bShow* den Wert `true` hat.

Argumente
bShow

- Das Argument *bShow* ist ein boolescher Wert, mit dem für das aktuelle Dokument zwischen der Layoutansicht und der Standardansicht gewechselt wird. Wenn für *bShow*`true` angegeben ist, wird das aktuelle Dokument in der Layoutansicht angezeigt. Wenn *bShow* den Wert `false` hat, wird das Dokument in der Standardansicht angezeigt.

Rückgabewerte
Keine.

Funktionen für die Auflösungsverwaltung

dreamweaver.canFitSize()

Verfügbarkeit
Dreamweaver CS5.5.

Beschreibung
Prüft, ob in einer aktiven Entwurfsansicht ein Objekt ausgewählt wurde. Ist dies der Fall, können die Funktionen `fitAll()` und `fitWidth()` aufgerufen werden.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn eine aktive Entwurfsansicht vorhanden ist, andernfalls `false`.

dom.getViewSizeMenuItems()

Verfügbarkeit
Dreamweaver CS5.5.

Beschreibung
Gibt einen String-Array für das Menü zur Größenauswahl in der Entwurfs- oder Live-Ansicht zurück. Dieser Array enthält vorgegebene, vom Benutzer definierte Größen und Medienabfragen mit Größeninformationen.

Argumente
Keine.

Rückgabewerte
Ein String-Array.

dom.isViewSizeMenuItemChecked()

Verfügbarkeit
Dreamweaver CS5.5.

Beschreibung
Dient zur Auswahl einer Option aus der Liste der Menüoptionen, die mit „dom.getViewSizeMenuItems()“ auf Seite 428 abgerufen wurde.

Argumente
Nullbasierter Index für die Menüoption.

Rückgabewerte
Ein boolescher Wert: `true`, wenn das Element aktiviert ist.

dom.isViewSizeMenuItemEnabled()

Verfügbarkeit
Dreamweaver CS5.5.

Beschreibung
Dient zur Aktivierung oder Deaktivierung von Optionen aus dem Menü zur Größenauswahl, die mit dom.getViewSizeMenuItems()„dom.getViewSizeMenuItems()“ auf Seite 428 abgerufen wurden.

Argumente
Nullbasierter Index für die Menüoption.

Rückgabewerte
Ein boolescher Wert: „true“ für aktiviert, „false“ für deaktiviert.

dom.isViewSizeMenuItemEnabled()

Verfügbarkeit
Dreamweaver CS5.5.

Beschreibung
Dient zur Aktivierung oder Deaktivierung von Optionen aus dem Menü zur Größenauswahl, die mit `dom.getViewSizeMenuItems()` abgerufen wurden.

Argumente
Nullbasierter Index für das Menüelement.

Rückgabewerte
Ein boolescher Wert: „true“ für aktiviert, „false“ für deaktiviert.

Medienabfrage

dw.mediaQueryListToJSON(strMediaQueryList)

Verfügbarkeit
Dreamweaver CS5.5.

Beschreibung
Analysiert einen String mit einer Medienabfrageliste, und gibt einen JSON-String an die aufrufende Funktion zurück.

Argumente
`strMediaQueryList` Ein String, der eine Medienabfrageliste darstellt, ähnlich dem Attribut „media“ eines link-Tags.

Rückgabewerte
Ein JSON-String, der die analysierte Medienabfrageliste repräsentiert. Die aufrufende Funktion kann diesen String untersuchen und/oder `eval` aufrufen, um ihn in ein JavaScript-Objekt zu konvertieren.

Trifft die Analyse auf einen Fehler, kann beim Aufruf `errorStr` verwendet werden, ein JSON-Objekt für das Testen. Ist diese Eigenschaft nicht vorhanden oder ist sie leer, wird kein Fehler zurückgegeben. Ist die Analyse erfolgreich, besitzt das JSON-Objekt die Eigenschaft `mediaQueryList` in Form eines Arrays.

Beispiel
```
var strJSON = dw.mediaQueryListToJSON('only screen and (min-width:769px)');
//strJSON is now:
{ mediaQueryList : [ { restrictor : 'only',
                       mediaType : 'screen',
                       mediaFeatures : [ { feature : 'width', comparisonType : 'min', value
: '769px' } ] }
],
errorStr : ''
}
```

site.getMediaQueryFile()

Verfügbarkeit
Dreamweaver CS5.5.

Beschreibung
Gibt die Position der Medienabfragedatei der aktuellen Site. Zum Beispiel „C:\Dokumente und Einstellungen\Benutzername\Eigene Dokumente\dw sites\slash site\css\devices.css.

Rückgabewerte
String mit dem vollständigen Pfad der SWMQF.

site.setMediaQueryFile()

Verfügbarkeit
Dreamweaver CS5.5.

Beschreibung
Legt die Medienabfragedatei fest, die für die gesamte aktuelle Site gültig ist.

Rückgabewerte
Keine.

dom.collectMediaQueries()

Verfügbarkeit
Dreamweaver CS5.5.

Beschreibung
Ruft die Medienabfrageinformationen zum Dokument ab (CSS, Medienabfrage, Beschreibung/Kommentar, Offsets).

Rückgabewerte
Ein Array mit Strings.

Beispiel
```
{ type: 'link', offsets: {start: 109, end: 213}, desc: 'for phone', descOffsets: {start: 89, end: 107}, mq: 'only screen and (max-width:320px)', css: 'phone.css' },{ type: 'link', offsets: {start: 236, end: 364}, desc: 'for tablet', descOffsets: {start: 215, end: 234}, mq: 'only screen and (min-width:3210px) and (max-width:700px)', css: 'tablet.css' }
```

Zoom-Funktionen

Mithilfe der Zoom-Funktionen können Sie die Entwurfsansicht vergrößern oder verkleinern.

dreamweaver.activeViewScale()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Die Eigenschaft *activeViewScale* kann einen änderbaren Gleitkommawert abrufen oder festlegen. Wenn Sie den Wert abrufen, wird der Skalierungsfaktor der aktiven Ansicht, wie er im Zoom-Kombinationsfeld angezeigt wird, geteilt durch 100 zurückgegeben. 100 % entspricht beispielsweise 1.0 und 50 % entspricht 0.5 usw. Wenn Sie den Wert setzen, wird er auch im Zoom-Kombinationsfeld festgelegt. Zulässig sind Werte zwischen 0.06 und 64.00 – das entspricht einem Skalierungsfaktor zwischen 6 % und 6400 %.

Beispiel
Im folgenden Beispiel wird der Skalierungsfaktor der aktuellen Ansicht abgerufen. Wenn der Skalierungsfaktor kleiner oder gleich 100 % ist, wird die Ansicht auch verkleinert.

```
if (canZoom() && dreamweaver.activeViewScale <= 1.0) {
    zoomIn();
    }
```

Im folgenden Beispiel wird der Skalierungsfaktor der aktuellen Ansicht auf 50 % gesetzt.

```
dreamweaver.activeViewScale = 0.50;
```

dreamweaver.fitAll()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion vergrößert oder verkleinert die Anzeige so, dass die Größe des gesamten Dokuments an den sichtbaren Teil des Entwurfsfensters angepasst wird.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dreamweaver.canZoom()“ auf Seite 525.

Beispiel

```
if (canZoom()){
    fitAll();
    }
```

dreamweaver.fitSelection()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion vergrößert oder verkleinert die Anzeige so, dass die Größe der aktuellen Auswahl an den sichtbaren Teil des Entwurfsfensters angepasst wird.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dreamweaver.canFitSelection()“ auf Seite 519.

Beispiel
```
if (canFitSeletion()){
    fitSelection();
    }
```

dreamweaver.fitWidth()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion vergrößert oder verkleinert die Anzeige so, dass das Dokument in seiner gesamten Breite an die Größe des Entwurfsfensters angepasst wird.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dreamweaver.canZoom()“ auf Seite 525.

Beispiel
```
if (canZoom()){
    fitWidth();
    }
```

dreamweaver.zoomIn()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion vergrößert die aktive Entwurfs- oder Live-Ansicht. Als Vergrößerungsstufe wird der nächste voreingestellte Wert im Menü „Vergrößerung“ verwendet. Wenn kein nächster voreingestellter Wert vorhanden ist, hat diese Funktion keine Auswirkung.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dreamweaver.canZoom()“ auf Seite 525.

Beispiel
```
if (canZoom()){
    zoomIn();
    }
```

dreamweaver.zoomOut()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion verkleinert die aktive Entwurfs- oder Live-Ansicht. Als Vergrößerungsstufe wird der nächste voreingestellte Wert im Menü „Vergrößerung“ verwendet. Wenn kein nächster voreingestellter Wert vorhanden ist, hat diese Funktion keine Auswirkung.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dreamweaver.canZoom()“ auf Seite 525.

Beispiel
```
if (canZoom()){
    zoomOut();
    }
```

Funktionen und Eigenschaften für Hilfslinien

Mit den Funktionen und Eigenschaften für Hilfslinien können Hilfslinien angezeigt, bearbeitet und gelöscht werden, mit denen Benutzer Elemente auf HTML-Seiten ausmessen und anordnen können.

dom.clearGuides()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion legt fest, ob alle Hilfslinien im Dokument gelöscht werden sollen.

Argumente
Keine.

Rückgabewerte
Keine.

Beispiel
Im folgenden Beispiel werden alle Hilfslinien im Dokument gelöscht, wenn im Dokument mindestens eine Hilfslinie vorhanden ist.

```
var currentDOM = dw.getDocumentDOM();
    if (currentDOM.hasGuides() == true) {
    currentDOM.clearGuides();
}
```

dom.createHorizontalGuide()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion erstellt an der aktuellen Position im Dokument eine horizontale Hilfslinie.

Argumente
location

- Das Argument *location* gibt die Position der Hilfslinie an, wobei sowohl der Wert als auch die Einheit in einem String angegeben werden, und zwar ohne Leerzeichen zwischen Wert und Einheit. Zulässige Einheiten sind `"px"` für Pixel und `"%"` für Prozent. Für 10 Pixel geben Sie beispielsweise `location = "10px"` an und für 50 Prozent `location = "50%"`.

Rückgabewerte
Keine.

Beispiel
Im folgenden Beispiel wird an der aktuellen Position im Dokument eine horizontale Hilfslinie erstellt.

```
var currentDOM = dw.getDocumentDOM();
currentDOM.createHorizontalGuide("10px");
```

dom.createVerticalGuide()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion erstellt an der aktuellen Position im Dokument eine vertikale Hilfslinie.

Argumente
location

- Das Argument *location* gibt die Position der Hilfslinie an, wobei sowohl der Wert als auch die Einheit in einem String angegeben werden, und zwar ohne Leerzeichen zwischen Wert und Einheit. Zulässige Einheiten sind `"px"` für Pixel und `"%"` für Prozent. Für 10 Pixel geben Sie beispielsweise `location = "10px"` an und für 50 Prozent `location = "50%"`.

Rückgabewerte
Keine.

Beispiel
Im folgenden Beispiel wird an der aktuellen Position im Dokument eine vertikale Hilfslinie erstellt.

```
var currentDOM = dw.getDocumentDOM();
currentDOM.createVerticalGuide("10px");
```

dom.deleteHorizontalGuide()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion löscht die horizontale Hilfslinie an der angegebenen Position.

Argumente
location

- Das Argument *location* ist ein String, der die Position im zu prüfenden Dokument angibt, wobei sowohl der Wert als auch die Einheit in einem String angegeben werden, und zwar ohne Leerzeichen zwischen Wert und Einheit. Zulässige Einheiten sind `"px"` für Pixel und `"%"` für Prozent. Für 10 Pixel geben Sie beispielsweise `location = "10px"` an und für 50 Prozent `location = "50%"`.

Rückgabewerte
Keine.

Beispiel
Im folgenden Beispiel wird die horizontale Hilfslinie an der angegebenen Position im Dokument gelöscht.

```
var currentDOM = dw.getDocumentDOM();
if (currentDOM.hasHorizontalGuide("10px") == true) {
    currentDOM.deleteHorizonalGuide("10px");
}
```

dom.deleteVerticalGuide()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion löscht die vertikale Hilfslinie an der angegebenen Position.

Argumente
location

- Das Argument *location* ist ein String, der die Position im zu prüfenden Dokument angibt, wobei sowohl der Wert als auch die Einheit in einem String angegeben werden, und zwar ohne Leerzeichen zwischen Wert und Einheit. Zulässige Einheiten sind `"px"` für Pixel und `"%"` für Prozent. Für 10 Pixel geben Sie beispielsweise `location = "10px"` an und für 50 Prozent `location = "50%"`.

Rückgabewerte
Keine.

Beispiel
Im folgenden Beispiel wird die vertikale Hilfslinie an der angegebenen Position im Dokument gelöscht.

```
var currentDOM = dw.getDocumentDOM();
if (currentDOM.hasVerticalGuide("10px") == true) {
    currentDOM.deleteVerticalGuide("10px");
}
```

dom.guidesColor

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese änderbare Farbeigenschaft legt die Farbe der Hilfslinien im Dokument fest. Sie können diese Eigenschaft setzen und abrufen.

Argumente
Keine.

Rückgabewerte
Keine.

Beispiel
Im folgenden Beispiel wird Grau als Farbe der Hilfslinien festgelegt.

```
var currentDOM = dw.getDocumentDOM();
if (currentDOM.guidesColor != "#444444"){
    currentDOM.guidesColor = "#444444";
    }
```

dom.guidesDistanceColor

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese änderbare Farbeigenschaft legt die Farbe der Hilfslinien für die Abstandsanzeige fest. Sie können diese Eigenschaft setzen und abrufen.

Argumente
Keine.

Rückgabewerte
Keine.

Beispiel
Im folgenden Beispiel erhalten die Hilfslinien für die Abstandsanzeigen die Farbe Grau.

```
var currentDOM = dw.getDocumentDOM();
if (currentDOM.guidesDistanceColor != "#CCCCCC"){
    currentDOM.guidesDistanceColor = "#CCCCCC";
    }
```

dom.guidesLocked

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese änderbare boolesche Eigenschaft legt fest, ob die Hilfslinien im Dokument gesperrt werden. Sie können diese Eigenschaft setzen und abrufen.

Argumente
Keine.

Rückgabewerte
Keine.

Beispiel
Im folgenden Beispiel werden Hilfslinien gesperrt, wenn sie noch nicht gesperrt sind.

```
var currentDOM = dw.getDocumentDOM();
if (currentDOM.guidesLocked == false) {
    currentDOM.guidesLocked = true;
}
```

dom.guidesSnapToElements

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese änderbare boolesche Eigenschaft legt fest, ob die Hilfslinien im Dokument an Elementen ausgerichtet werden. Sie können diese Eigenschaft setzen und abrufen.

Argumente
Keine.

Rückgabewerte
Keine.

Beispiel
Im folgenden Beispiel werden die Hilfslinien im Dokument an Elementen ausgerichtet.

```
var currentDOM = dw.getDocumentDOM();
if (currentDOM.guidesSnapToElements == false) {
    currentDOM.guidesSnapToElements = true;
    }
```

dom.guidesVisible

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese änderbare boolesche Eigenschaft legt fest, ob die Hilfslinien im Dokument sichtbar sind. Sie können diese Eigenschaft setzen und abrufen.

Argumente
Keine.

Rückgabewerte
Keine.

Beispiel
Im folgenden Beispiel wird die Anzeige von Hilfslinien aktiviert, wenn sie nicht sichtbar sind.

```
var currentDOM = dw.getDocumentDOM();
if (currentDOM.guidesVisible == false) {
    currentDOM.guidesVisible = true;
}
```

dom.hasGuides()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion prüft, ob im Dokument mindestens eine Hilfslinie vorhanden ist. Sie können diese Eigenschaft setzen und abrufen.

Argumente
Keine.

Rückgabewerte
Keine.

Beispiel
Im folgenden Beispiel werden alle Hilfslinien im Dokument gelöscht, wenn im Dokument mindestens eine Hilfslinie vorhanden ist.

```
var currentDOM = dw.getDocumentDOM();
    if (currentDOM.hasGuides() == true) {
    currentDOM.clearGuides();
}
```

dom.hasHorizontalGuide()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion prüft, ob im Dokument an der angegebenen Position eine horizontale Hilfslinie vorhanden ist.

Argumente
location

- Das Argument *location* ist ein String, der die Position im zu prüfenden Dokument angibt, wobei sowohl der Wert als auch die Einheit in einem String angegeben werden, und zwar ohne Leerzeichen zwischen Wert und Einheit. Zulässige Einheiten sind `"px"` für Pixel und `"%"` für Prozent. Für 10 Pixel geben Sie beispielsweise `location = "10px"` an und für 50 Prozent `location = "50%"`.

Rückgabewerte
Ein boolescher Wert: `true`, wenn sich an der Position eine horizontale Hilfslinie befindet, andernfalls `false`.

Beispiel
Im folgenden Beispiel werden alle Hilfslinien im Dokument gelöscht, wenn sich im Dokument an der angegebenen Position eine horizontale Hilfslinie befindet.

```
var currentDOM = dw.getDocumentDOM();
if (currentDOM.hasHorizontalGuide("10px") == true) {
    currentDOM.clearGuides();
}
```

dom.hasVerticalGuide()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion prüft, ob im Dokument an der aktuellen Position eine vertikale Hilfslinie vorhanden ist.

Argumente
location

- Das Argument *location* ist ein String, der die Position im zu prüfenden Dokument angibt, wobei sowohl der Wert als auch die Einheit in einem String angegeben werden, und zwar ohne Leerzeichen zwischen Wert und Einheit. Zulässige Einheiten sind `"px"` für Pixel und `"%"` für Prozent. Für 10 Pixel geben Sie beispielsweise `location = "10px"` an und für 50 Prozent `location = "50%"`.

Rückgabewerte
Ein boolescher Wert: `true`, wenn sich an der Position eine vertikale Hilfslinie befindet, andernfalls `false`.

Beispiel
Im folgenden Beispiel werden alle Hilfslinien im Dokument gelöscht, wenn sich im Dokument an der angegebenen Position eine vertikale Hilfslinie befindet.

```
var currentDOM = dw.getDocumentDOM();
if (currentDOM.hasVerticalGuide("10px") == true) {
    currentDOM.clearGuides();
}
```

dom.snapToGuides

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese änderbare boolesche Eigenschaft legt fest, ob Elemente an den Hilfslinien im Dokument ausgerichtet werden. Sie können diese Eigenschaft setzen und abrufen.

Argumente
Keine.

Rückgabewerte
Keine.

Beispiel
Im folgenden Beispiel werden die Elemente im Dokument an den Hilfslinien ausgerichtet.

```
var currentDOM = dw.getDocumentDOM();
if (currentDOM.snapToGuides == false) {
    currentDOM.snapToGuides = true;
}
```

Funktionen zum Bearbeiten von Tabellen

Mit Funktionen zum Bearbeiten von Tabellen können Tabellenzeilen und -spalten hinzugefügt und entfernt, Spaltenbreiten und Zeilenhöhen geändert, die Maßangaben von Pixel in Prozent und von Prozent in Pixel geändert sowie andere Standardvorgänge der Tabellenbearbeitung durchgeführt werden.

dom.convertWidthsToPercent()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion konvertiert alle `WIDTH`-Attribute in der aktuellen Tabelle von Pixel in Prozent.

Argumente
Keine.

Rückgabewerte
Keine.

dom.convertWidthsToPixels()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Diese Funktion konvertiert alle `WIDTH`-Attribute in der aktuellen Tabelle von Prozent in Pixel.

Argumente
Keine.

Rückgabewerte
Keine.

dom.decreaseColspan()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion verkleinert die Spaltenbreite um 1 Einheit.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dom.canDecreaseColspan()“ auf Seite 509.

dom.decreaseRowspan()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion verkleinert die Zeilenhöhe um 1 Einheit.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dom.canDecreaseRowspan()“ auf Seite 509.

dom.deleteTableColumn()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion entfernt die ausgewählten Tabellenspalten.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dom.canDeleteTableColumn()“ auf Seite 509.

dom.deleteTableRow()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion entfernt die ausgewählten Tabellenzeilen.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dom.canDeleteTableRow()“ auf Seite 510 .

dom.doDeferredTableUpdate()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Wenn in den allgemeinen Voreinstellungen die Option „Schnellere Tabellenbearbeitung“ aktiviert ist, werden die zuletzt vorgenommenen Änderungen im Tabellenlayout angezeigt, ohne dass die Auswahl außerhalb der Tabelle verschoben wird. Diese Funktion hat keine Auswirkung, wenn die Option „Schnellere Tabellenbearbeitung“ nicht aktiviert ist.

Argumente
Keine.

Rückgabewerte
Keine.

dom.getShowTableWidths()

Verfügbarkeit
Dreamweaver MX 2004, aktualisiert in CS4.

Beschreibung
Gibt an, ob die Tabellenbreite im Standardmodus oder erweiterten Tabellenmodus angezeigt wird. Informationen zur Anzeige von Registerkarten für Tabellen im Layoutmodus finden Sie unter „dom.getShowLayoutTableTabs()“ auf Seite 422.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn die Tabellenbreite im Standardmodus oder erweiterten Tabellenmodus angezeigt wird, andernfalls `false`.

dom.getTableExtent()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion ruft die Anzahl der Spalten und Zeilen in der ausgewählten Tabelle ab.

Argumente
Keine.

Rückgabewerte
Ein Array mit zwei Ganzzahlen. Das erste Element gibt die Anzahl der Spalten an, das zweite Element die Anzahl der Zeilen. Wenn keine Tabelle ausgewählt ist, wird kein Wert zurückgegeben.

dom.increaseColspan()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion vergrößert die Spaltenbreite um 1 Einheit.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dom.canIncreaseColspan()“ auf Seite 510.

dom.increaseRowspan()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion vergrößert die Zeilenhöhe um 1 Einheit.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dom.canIncreaseRowspan()“ auf Seite 511.

dom.insertTableColumns()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion fügt die angegebene Anzahl Spalten in die aktuelle Tabelle ein.

Argumente
numberOfCols, bBeforeSelection

- Das Argument *numberOfCols* ist die Anzahl der einzufügenden Spalten.
- Das Argument *bBeforeSelection* ist ein boolescher Wert: `true`, wenn die Spalten vor der Spalte eingefügt werden sollen, die die Auswahl enthält, andernfalls `false`.

Rückgabewerte
Keine.

Enabler
Siehe „dom.canInsertTableColumns()“ auf Seite 511.

dom.insertTableRows()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion fügt die angegebene Anzahl Zeilen in die aktuelle Tabelle ein.

Argumente
numberOfRows, bBeforeSelection

- Das Argument *numberOfRows* ist die Anzahl der einzufügenden Zeilen.
- Das Argument *bBeforeSelection* ist ein boolescher Wert: `true`, wenn die Zeilen über der Zeile eingefügt werden sollen, die die Auswahl enthält, andernfalls `false`.

Rückgabewerte
Keine.

Enabler
Siehe „dom.canInsertTableRows()“ auf Seite 511.

dom.mergeTableCells()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion verbindet die ausgewählten Tabellenzellen.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dom.canMergeTableCells()“ auf Seite 512.

dom.removeAllTableHeights()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion entfernt alle `HEIGHT`-Attribute aus der ausgewählten Tabelle.

Argumente
Keine.

Rückgabewerte
Keine.

dom.removeAllTableWidths()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion entfernt alle `WIDTH`-Attribute aus der ausgewählten Tabelle.

Argumente
Keine.

Rückgabewerte
Keine.

dom.removeColumnWidth()

Verfügbarkeit
Dreamweaver MX 2004.

Beschreibung
Diese Funktion entfernt alle `WIDTH`-Attribute aus einer einzelnen ausgewählten Spalte.

Argumente
Keine.

Rückgabewerte
Keine.

dom.selectTable()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Wählt eine ganze Tabelle aus.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dom.canSelectTable()“ auf Seite 514.

dom.setShowTableWidths()

Verfügbarkeit
Dreamweaver MX 2004, aktualisiert in CS4.

Beschreibung
Aktiviert oder deaktiviert die Anzeige der Tabellenbreiten im Standardmodus oder erweiterten Tabellenmodus. Diese Funktion legt den Wert für das aktuelle Dokument und alle weiteren Dokumente fest, sofern nicht anders angegeben. Informationen zum Festlegen der Anzeige von Registerkarten für Tabellen im Layoutmodus finden Sie unter „dom.setShowLayoutTableTabs()“ auf Seite 427.

Argumente
`bShow`

- Das Argument *`bShow`* ist ein boolescher Wert, der angibt, ob Tabellenbreiten für Tabellen angezeigt werden sollen, wenn sich das aktuelle Dokument im Standardmodus oder erweiterten Tabellenmodus befindet. Wenn für *`bShow`* der Wert `true` festgelegt ist, werden die Tabellenbreiten angezeigt. Wenn für *`bShow`* der Wert `false` festgelegt ist, werden die Tabellenbreiten nicht angezeigt.

Rückgabewerte
Keine.

dom.setTableCellTag()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion legt das Tag für die ausgewählte Zelle fest.

Argumente
tdOrTh

- Das Argument *tdOrTh* muss entweder `"td"` oder `"th"` sein.

Rückgabewerte
Keine.

dom.setTableColumns()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion legt die Anzahl der Spalten in der ausgewählten Tabelle fest.

Argumente
numberOfCols

- Das Argument *numberOfCols* ist die Anzahl der in der Tabelle festzulegenden Spalten.

Rückgabewerte
Keine.

dom.setTableRows()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion legt die Anzahl der Zeilen in der ausgewählten Tabelle fest.

Argumente
numberOfRows

- Das Argument *numberOfRows* ist die Anzahl der in der ausgewählten Tabelle festzulegenden Zeilen.

Rückgabewerte
Keine.

dom.showInsertTableRowsOrColumnsDialog()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Diese Funktion öffnet das Dialogfeld „Zeilen oder Spalten einfügen“.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dom.canInsertTableColumns()“ auf Seite 511 oder „dom.canInsertTableRows()“ auf Seite 511.

dom.splitTableCell()

Verfügbarkeit
Dreamweaver 3.

Beschreibung

Diese Funktion teilt die aktuelle Tabellenzelle in die angegebene Anzahl von Zeilen oder Spalten. Wenn ein oder beide Argumente nicht angegeben werden, wird das Dialogfeld „Zelle teilen“ angezeigt.

Argumente

{colsOrRows}, {numberToSplitInto}

- Das optionale Argument *colsOrRows* muss entweder `"columns"` oder `"rows"` lauten.
- Das optionale Argument *numberToSplitInto* ist die Anzahl der Zeilen oder Spalten, in die die Zelle aufgeteilt werden soll.

Rückgabewerte

Keine.

Enabler

Siehe „dom.canSplitTableCell()“ auf Seite 515.

Kapitel 18: Code

Mithilfe der Codefunktionen können Sie ein Dokument bearbeiten, das in der Codeansicht angezeigt wird. Sie haben die Möglichkeit, neue menu- oder function-Tags zu einem Menü für Codehinweise hinzuzufügen, Stringmuster zu suchen und zu ersetzen, die aktuelle Auswahl aus einem Dokument zu löschen, den gesamten oder ausgewählten Code zu löschen, Tags zu bearbeiten oder Syntaxformatierungen zum ausgewählten Code hinzuzufügen.

Codefunktionen

Codehinweise sind Menüs, die in Adobe® Dreamweaver® CS5 angezeigt werden, wenn Sie in der Codeansicht bestimmte Zeichenmuster eingeben. Codehinweise ermöglichen eine schnellere Eingabe, indem eine Liste der Strings angezeigt wird, mit denen der eingegebene String vervollständigt werden kann. Wenn der eingegebene String im Menü angezeigt wird, können Sie ihn in der Liste auswählen und die Eingabe durch Drücken der Eingabetaste bzw. des Zeilenschalters automatisch vervollständigen. Wenn Sie beispielsweise < eingeben, wird im Popupmenü eine Liste mit Tag-Namen angezeigt. Anstatt den Rest des Tag-Namens einzugeben, können Sie das Tag im Menü auswählen, um es in den Text einzufügen.

Sie können in Dreamweaver Menüs für Codehinweise hinzufügen, indem Sie sie in der Datei „CodeHints.xml“ definieren. Informationen zur Datei „CodeHints.xml“ finden Sie im Handbuch *Dreamweaver erweitern*.

Sie können neue Menüs für Codehinweise auch dynamisch über JavaScript hinzufügen, nachdem der Inhalt der Datei „CodeHints.xml“ in Dreamweaver geladen wurde. Die Liste der Sitzungsvariablen im Bedienfeld „Bindungen“ wird beispielsweise durch JavaScript-Code gefüllt. Mit demselben Code können Sie auch ein Menü für Codehinweise hinzufügen. Wenn ein Benutzer in der Codeansicht **Session.** eingibt, wird ein Menü mit Sitzungsvariablen angezeigt.

Die Datei „CodeHints.xml“ sowie die JavaScript-API stellen eine beträchtliche Teilmenge des Moduls für Codehinweise bereit. Einige Dreamweaver-Funktionen sind jedoch nicht verfügbar. Beispielsweise gibt es keine JavaScript-Funktion zum Einblenden der Farbauswahl. Somit kann das Menü „Attributwerte“ nicht über JavaScript dargestellt werden. Sie können lediglich ein Menü mit Textelementen öffnen, aus dem Sie Text einfügen können.

Mit der Funktion für Codefarben können Sie Codefarbstile festlegen sowie vorhandene Codefarbschemas bearbeiten oder neue Schemas erstellen. Sie können Codefarbstile und Codefarbschemas durch Bearbeiten der Datei „Colors.xml“ und der Dateien mit den Codefarbschemas festlegen. Weitere Informationen zu diesen Dateien finden Sie im Handbuch *Dreamweaver erweitern*.

Die JavaScript-API für Codehinweise und Codefarben besteht aus den folgenden Funktionen.

dreamweaver.codeHints.addMenu()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Definiert in der Datei „CodeHints.xml“ dynamisch ein neues `menu`-Tag. Wenn ein menu-Tag mit identischem Muster und Dokumenttyp vorhanden ist, fügt diese Funktion dem bestehenden Menü weitere Elemente hinzu.

Argumente

menuGroupId, *pattern*, *labelArray*, *{valueArray}*, *{iconArray}*, *{doctypes}*, *{casesensitive}*, *{object}*, *{descriptionArray}*, *{dismissChars}*, *{allowWhitespacePrefix}*, *{restriction}*, *{type}*, *{bForcedOnly}*, *{allowMultipleTimes}*, *{docURI}*, *{alias}*

- Das Argument *menuGroupId* ist das ID-Attribut für eines der `menugroup`-Tags.
- Das Argument *pattern* ist das Musterattribut für das neue `menu`-Tag.
- Das Argument *labelArray ist ein String-Array.* Jeder String enthält den Text für ein einzelnes Menüelement des Popupmenüs.
- Das optionale Argument *valueArray* ist ein String-Array, das die gleiche Länge wie das Argument *labelArray* aufweisen muss. Wenn ein Benutzer im Popupmenü ein Element auswählt, wird der String aus diesem Array in das Benutzerdokument eingefügt. Wenn der einzufügende String und die Menübeschriftung immer identisch sind, kann dieses Argument den Wert `null` aufweisen.
- Das optionale Argument *iconArray* ist entweder ein String oder ein String-Array. Wenn es sich um einen String handelt, gibt dieser die URL für eine einzelne Bilddatei an, die Dreamweaver für alle Elemente im Menü verwendet. Wenn es sich um ein String-Array handelt, muss dieses dieselbe Länge wie das Argument *labelArray* aufweisen. Jeder String enthält die URL für eine Bilddatei, relativ zum Dreamweaver-Ordner „Configuration“, die in Dreamweaver als Symbol für das entsprechende Menüelement verwendet wird. Wenn dieses Argument den Wert `null` hat, wird das Menü ohne Symbole angezeigt.
- Das optionale Argument *doctypes* gibt an, dass das Menü nur für bestimmte Dokumenttypen aktiv ist. Sie können das Argument `doctypes` als eine durch Kommas getrennte Liste von Dokumenttyp-IDs angeben. Eine Liste der Dokumenttypen von Dreamweaver finden Sie in der Dreamweaver-Datei „Configuration/Documenttypes/MMDocumentTypes.xml“.
- Das optionale Argument *casesensitive* gibt an, ob bei dem Muster zwischen Groß- und Kleinschreibung unterschieden werden soll. Mögliche Werte für das Argument *casesensitive* sind die booleschen Werte `true` und `false`. Wenn das Argument nicht angegeben ist, wird automatisch als Standardwert `false` verwendet. Wenn das Argument *casesensitive* den Wert `true` hat, wird das Menü „Codehinweise“ angezeigt. Dieses Menü wird nur angezeigt, wenn der vom Benutzer eingegebene Text dem mit dem pattern-Attribut angegebenen Muster entspricht. Wenn *casesensitive* den Wert `false` aufweist, wird das Menü auch angezeigt, wenn sich die Groß- und Kleinschreibung von Muster und Text unterscheiden.
- Das Argument *object* gibt den Namen des Strings an. Dieses Argument ist optional. Es wird verwendet, wenn das Objekt den Typ „static“ aufweist.
- Das Argument *descriptionArray* beschreibt detailliert die Elemente, die in den Codehinweisen angezeigt werden. Dieses Argument ist optional.
- Das Argument *dismissChars* gibt die Sonderzeichen an, die der Benutzer eingibt, um das Menü „Codehinweise“ zu schließen. Dieses Argument ist optional.
- Das Argument *allowWhitespacePrefix* ist ein boolescher Wert, der Leerräume vor dem Codehinweis zulässt. Dieses Argument ist optional und der Standardwert ist `false`.
- Das Argument *restriction* ist ein String. Dieses Argument ist optional. Wenn es nicht angegeben ist, wird keine Beschränkung festgelegt. Auf einer Webseite mit clientseitigen und serverseitigen Sprachen können Sie mit diesem Argument die Verwendung von Codehinweisen auf eines der folgenden Elemente beschränken:
  - Auf den Bereich einer bestimmten Sprache
  - Auf einen Codeblock
- Mithilfe des Arguments *type* werden die Menütypen der Benutzeroberfläche definiert. Dieses Argument ist optional und der Standardwert ist „Enumerated drop down UI“. Weitere mögliche Werte sind `color`, `font` und `url`.

- Das Argument *bForcedOnly* ist ein boolescher Wert. Wenn das Argument den Wert `true` hat, wird das Menü „Codehinweise“ nur bei Verwendung des Tastaturbefehls Strg+Leertaste angezeigt. Dieses Argument ist optional und der Standardwert ist `false`.
- Das Argument *allowMultipleTimes* ist ein boolescher Wert. Wenn das Argument den Wert `true` hat, kann dasselbe Menü mehrmals angezeigt werden. Dieses Argument ist optional und der Standardwert ist `false`.
- Mithilfe des Arguments *docURI* kann der Benutzer die Codehinweise durch Eingeben des Dokument-URI (Betriebssystem-spezifischer Dateipfad) auf ein bestimmtes Dokument beschränken. Dieses Argument ist optional. Wenn der Dokument-URI nicht angegeben ist, wird keine Beschränkung festgelegt.
- Das Argument *alias* erlaubt es dem Benutzer, die Codehinweise mit einem anderen Muster aufzurufen als dem in dem Argument „pattern“ bzw. „classpattern“ aufgeführten. Dieses Argument ist optional.

Rückgabewerte
Keine.

Beispiel
Wenn der Benutzer eine Datensatzgruppe mit dem Namen „myRs“ erstellt, wird mit dem folgenden Code ein Menü für myRS generiert:

```
dw.codeHints.addMenu(
    "CodeHints_object_methods",          // menu is enabled if object methods are enabled
    "myRS.",                             // pop up menu if user types "myRS."
    new Array("firstName", "lastName"),  // items in pop-up menu for myRS
    new Array("firstName", "lastName"),  // text to actually insert in document
    null,                                // no icons for this menu
    "ASP_VB, ASP_JS");                     // specific to the ASP doc types
```

dreamweaver.codeHints.addFunction()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Definiert dynamisch ein neues *function*-Tag. Wenn ein *function*-Tag mit identischem Muster und Dokumenttyp vorhanden ist, ersetzt diese Funktion das vorhandene function-*Tag*.

Argumente
menuGroupId, *pattern*, *{doctypes}*, *{casesensitive}*, *{object}*, *{description}*, *{icon}*, *source*, *{docURI}*, *{bClassPattern}*, *{bAddToObjectMethodList}*, *{restriction}*

- Das Argument *menuGroupId* ist das ID-Stringattribut eines `menugroup`-Tags.
- Das Argument *pattern* ist ein String, der das Musterattribut für das neue *function*-Tag angibt.
- Das optionale Argument *doctypes* gibt an, dass diese Funktion nur für bestimmte Dokumenttypen aktiv ist. Sie können das Argument `doctypes` als eine durch Kommas getrennte Liste von Dokumenttyp-IDs angeben. Eine Liste der Dokumenttypen von Dreamweaver finden Sie in der Dreamweaver-Datei „Configuration/Documenttypes/MMDocumentTypes.xml“.

- Das optionale Argument *`casesensitive` gibt an, ob bei dem Muster zwischen Groß- und Kleinschreibung unterschieden werden soll.* Mögliche Werte für das Argument *`casesensitive`* sind die booleschen Werte `true` und `false`. Wenn das Argument nicht angegeben ist, wird automatisch als Standardwert `false` verwendet. Wenn das Argument *`casesensitive`* den Wert `true` hat, wird das Menü „Codehinweise“ angezeigt. Dieses Menü wird nur angezeigt, wenn der vom Benutzer eingegebene Text dem mit dem pattern-Attribut angegebenen Muster entspricht. Wenn *`casesensitive`* auf `false` gesetzt ist, wird das Menü auch angezeigt, wenn sich die Groß- und Kleinschreibung von Muster und Text unterscheiden.
- Das Argument *`object`* gibt den Namen des Strings an. Dieses Argument ist optional. Es wird verwendet, wenn das Objekt den Typ „static“ aufweist.
- Das Argument `description` enthält eine detaillierte Beschreibung der Funktion. Dieses Argument ist optional.
- Das Argument `icon` gibt den Pfad des benutzerdefinierten Symbols für das Funktions-Dropdownmenü an. Dieses Argument ist optional.
- Das Argument `source` enthält einen Wert, der in der zweiten Spalte des Codehinweises angezeigt wird. Der Standardwert dieses Arguments ist `empty`.
- Mithilfe des Arguments `docURI` kann der Benutzer die Codehinweise durch Eingeben des Dokument-URI (Betriebssystem-spezifischer Dateipfad) auf ein bestimmtes Dokument beschränken. Dieses Argument ist optional. Wenn der Dokument-URI nicht angegeben ist, wird keine Beschränkung festgelegt.
- Das Argument `bClassPattern` ist ein boolescher Wert. Der Wert `true` gibt an, dass die Funktion zu einer class-Instanz gehört und nicht statisch ist. Der Standardwert ist `false`. Dieses Argument ist optional.
- Das Argument `bAddToObjectMethodList` ist ein boolescher Wert. Wenn der Wert „true“ festgelegt ist, kann der Benutzer eine Liste statischer Funktionen hinzufügen. Der Standardwert ist `false`. Dieses Argument ist optional.
- Das Argument *`restriction`* ist ein String. Dieses Argument ist optional. Wenn es nicht angegeben ist, wird keine Beschränkung festgelegt. Auf einer Webseite mit clientseitigen und serverseitigen Sprachen können Sie mit diesem Argument die Verwendung von Codehinweisen auf eines der folgenden Elemente beschränken:
  - Auf den Bereich einer bestimmten Sprache
  - Auf einen Codeblock

Rückgabewerte
Keine.

Beispiel
Im folgenden Beispiel für die Funktion `dw.codeHints.addFunction()` wird das Funktionsnamenmuster `out.newLine()` zur Menügruppe für Codehinweise `CodeHints_Object_Methods` hinzugefügt und nur für `JSP`-Dokumenttypen aktiviert.

```
dw.codeHints.addFunction(
    "CodeHints_Object_Methods",
    "out.newLine()",
    "JSP")
```

dreamweaver.codeHints.resetMenu()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Setzt das angegebene menu- oder function-Tag auf den Status unmittelbar nach dem Einlesen der Datei „CodeHints.xml“ zurück. Ein Aufruf dieser Funktion hebt daher die Wirkung vorheriger Aufrufe der Funktionen `addMenu()` und `addFunction()` auf.

Argumente
menuGroupId, pattern, {doctypes}

- Das Argument *menuGroupId* ist das ID-Stringattribut eines `menugroup`-Tags.
- Das Argument *pattern* ist ein String, der das Musterattribut für das neue zurückzusetzende `menu`- oder `function`-Tag angibt.
- Das optionale Argument *doctypes* gibt an, dass das Menü nur für bestimmte Dokumenttypen aktiv ist. Sie können das Argument `doctypes` als eine durch Kommas getrennte Liste von Dokumenttyp-IDs angeben. Eine Liste der Dokumenttypen von Dreamweaver finden Sie in der Dreamweaver-Datei „Configuration/Documenttypes/MMDocumentTypes.xml“.

Rückgabewerte
Keine.

Beispiel
Ihr JavaScript-Code soll ein Menü für Codehinweise generieren, das benutzerdefinierte Sitzungsvariablen enthält. Bei jeder Änderung der Liste der Sitzungsvariablen muss dieser Code das Menü aktualisieren. Bevor die neue Liste der Sitzungsvariablen in das Menü geladen werden kann, muss die alte Liste entfernt werden. Ein Aufruf dieser Funktion entfernt die alten Sitzungsvariablen.

dreamweaver.codeHints.showCodeHints()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Dreamweaver ruft diese Funktion auf, wenn der Benutzer das Menüelement „Bearbeiten“ > „Codehinweise anzeigen“ öffnet. Die Funktion öffnet das Menü „Codehinweise“ in der Codeansicht an der aktuellen Auswahl.

Argumente
Keine.

Rückgabewerte
Keine.

Beispiel
Im folgenden Beispiel wird das Menü „Codehinweise“ an der aktuellen Einfügemarke im Dokument geöffnet, wenn es sich in der Codeansicht befindet.

```
dw.codeHints.showCodeHints()
```

dreamweaver.reloadCodeColoring()

Beschreibung
Lädt Dateien für Codefarben aus dem Dreamweaver-Ordner „Configuration/Code Coloring“ neu.

Argumente
Keine.

Rückgabewerte
Keine.

Beispiel
```
dreamweaver.reloadCodeColoring()
```

Funktionen zum Suchen und Ersetzen

Mit diesen Funktionen können Sie Vorgänge zum Suchen und Ersetzen durchführen. Es stehen sowohl grundlegende Funktionen (z. B. Suchen der nächsten Instanz eines Suchbegriffs) als auch komplexe Ersetzungsfunktionen zur Verfügung, die keinen Benutzereingriff erfordern.

dreamweaver.findNext()

Verfügbarkeit
Dreamweaver 3, in Dreamweaver MX 2004 geändert.

Beschreibung
Sucht die nächste Instanz des Suchbegriffs, der zuvor mit den Funktionen „dreamweaver.setUpFind()“ auf Seite 460 oder „dreamweaver.setUpComplexFind()“ auf Seite 459 oder vom Benutzer im Dialogfeld „Suchen“ definiert wurde, und wählt die Instanz im Dokument aus.

Argumente
{bUseLastSetupSearch}, {document}, {clearSearchResults}

- Das optionale Argument *bUseLastSetupSearch* ist ein boolescher Wert. Wenn *bUseLastSetupSearch* den Wert `true` aufweist (Standardeinstellung, wenn kein Argument angegeben ist), führt die Funktion eine Operation zum Weitersuchen anhand der Parameter durch, die durch einen vorherigen Aufruf von `dreamweaver.setupComplexFind()` oder `dreamweaver.setupComplexFindReplace()` angegeben wurden. Wenn Sie *bUseLastSetupSearch* auf `false` setzen, ignoriert die Funktion die zuvor festgelegte Suche und führt eine Suche nach der nächsten Instanz des Texts durch, der aktuell im Dokument ausgewählt ist.
- Das optionale Argument *document* stellt das zu durchsuchende Dokument dar.
- Das optionale Argument *clearSearchResults* gibt an, ob der Bereich „Ergebnisse“ im Bedienfeld „Suchen“ vor dem Durchführen eines Suchvorgangs gelöscht werden soll.

Rückgabewerte
Keine.

Enabler
Siehe „dreamweaver.canFindNext()“ auf Seite 519.

dreamweaver.findAll()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Sucht alle Instanzen des Suchbegriffs, der zuvor mit `dreamweaver.setUpFind()` oder `dreamweaver.setUpComplexFind()` angegeben oder vom Benutzer im Dialogfeld „Suchen“ festgelegt wurde, und wählt alle Instanzen im Dokument aus.

Argumente
Die Argumente entsprechen den für die Funktion `dreamweaver.findNext()` angegebenen Argumenten. Siehe „dreamweaver.findNext()“ auf Seite 457.

Rückgabewerte
Keine.

dreamweaver.replace()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob die aktuelle Auswahl mit den Suchkriterien übereinstimmt, die von „dreamweaver.setUpFindReplace()“ auf Seite 461, von „dreamweaver.setUpComplexFindReplace()“ auf Seite 460 oder vom Benutzer im Dialogfeld „Ersetzen“ angegeben wurden. Die Funktion ersetzt dann die Auswahl durch den mit der Suchanforderung angegebenen Ersetzungstext.

Argumente
Die Argumente entsprechen den für die Funktion `dreamweaver.findNext()` angegebenen Argumenten. Siehe „dreamweaver.findNext()“ auf Seite 457.

Rückgabewerte
Keine.

dreamweaver.replaceAll()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Ersetzt alle Bereiche im aktuellen Dokument, die den Suchkriterien entsprechen, die zuvor mit der Funktion „dreamweaver.setUpFindReplace()“ auf Seite 461 bzw. „dreamweaver.setUpComplexFindReplace()“ auf Seite 460 oder vom Benutzer im Dialogfeld „Ersetzen“ festgelegt wurden, durch den angegebenen Inhalt.

Argumente
Die Argumente entsprechen den für die Funktion `dreamweaver.findNext()` angegebenen Argumenten. Siehe „dreamweaver.findNext()“ auf Seite 457.

Rückgabewerte
Keine.

dreamweaver.setUpComplexFind()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Bereitet eine erweiterte Text- oder Tag-Suche durch Laden der angegebenen XML-Abfrage vor.

Argumente
xmlQueryString

- Das Argument *xmlQueryString* ist ein String aus XML-Code, der mit `dwquery` beginnt und mit `/dwquery` endet. (Um einen String im richtigen Format abzurufen, erstellen Sie eine Abfrage im Dialogfeld „Suchen“, klicken Sie auf die Schaltfläche „Abfrage speichern“, öffnen Sie die Abfragedatei in einem Texteditor und kopieren Sie den gesamten Bereich vom öffnenden `dwquery`-Tag bis zum schließenden `/dwquery`-Tag.)

  Hinweis: *In einer Abfrage muss die spezielle Bedeutung bestimmter Sonderzeichen, z. B. der umgekehrte Schrägstrich (\), aufgehoben werden. Wenn Sie einen umgekehrten Schrägstrich in einer Abfrage verwenden möchten, müssen Sie deshalb „\\“ eingeben.*

Rückgabewerte
Keine.

Beispiel
In der ersten Zeile des folgenden Beispiels wird eine Tag-Suche mit dem aktuellen Dokument als Suchbereich angegeben. In der zweiten Zeile wird der Suchvorgang durchgeführt.

```
dreamweaver.setUpComplexFind('<dwquery><queryparams matchcase="false" ¬
ignorewhitespace="true" useregexp="false"/><find>¬
<qtag qname="a"><qattribute qname="href" qcompare="=" qvalue="#">¬
    </qattribute><qattribute qname="onMouseOut" qcompare="=" qvalue="" qnegate="true">¬
    </qattribute></qtag></find></dwquery>');
dw.findNext();
```

dreamweaver.setUpComplexFindReplace()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Bereitet eine erweiterte Text- oder Tag-Suche durch Laden der angegebenen XML-Abfrage vor.

Argumente
xmlQueryString

- Das Argument *xmlQueryString* ist ein String aus XML-Code, der mit dem Tag `dwquery` beginnt mit dem Tag `/dwquery` endet. (Um einen String im richtigen Format abzurufen, erstellen Sie eine Abfrage im Dialogfeld „Suchen“, klicken Sie auf die Schaltfläche „Abfrage speichern“, öffnen Sie die Abfragedatei in einem Texteditor und kopieren Sie den gesamten Bereich vom öffnenden `dwquery`-Tag bis zum schließenden `/dwquery`-Tag.)

  Hinweis: *In einer Abfrage muss die spezielle Bedeutung bestimmter Sonderzeichen, z. B. der umgekehrte Schrägstrich (\), aufgehoben werden. Wenn Sie einen umgekehrten Schrägstrich in einer Abfrage verwenden möchten, müssen Sie deshalb „\\“ eingeben.*

Rückgabewerte
Keine.

Beispiel
In der ersten Anweisung des folgenden Beispiels wird eine Tag-Suche mit vier Dateien als Suchbereich angegeben. Mit der zweiten Anweisung wird der Vorgang zum Suchen und Ersetzen durchgeführt.

```
dreamweaver.setUpComplexFindReplace('<dwquery><queryparams ¬
matchcase="false" ignorewhitespace="true" useregexp="false"/>¬
<find><qtag qname="a"><qattribute qname="href" qcompare="=" qvalue="#">¬
</qattribute><qattribute qname="onMouseOut" ¬qcompare="=" qvalue="" qnegate="true">¬
</qattribute></qtag></find><replace action="setAttribute" param1="onMouseOut" ¬
param2="this.style.color='#000000';this.style.¬
fontWeight='normal'"/></dwquery>');
dw.replaceAll();
```

dreamweaver.setUpFind()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Bereitet eine Text- oder HTML-Quellcode-Suche vor, indem die Suchparameter für einen anschließenden Vorgang vom Typ `dreamweaver.findNext()` definiert werden.

Argumente
searchObject

Das Argument *searchObject* ist ein Objekt, für das die folgenden Eigenschaften definiert werden können:

- *searchString* ist der zu suchende Text.

- Mit dem Argument *searchWhat* wird festgelegt, wo der Suchvorgang durchgeführt wird. Folgende Werte sind möglich:
  - `document` – Das aktuelle aktive Dokument wird durchsucht.
  - `allOpenDocuments` – Alle geöffneten Dokumente werden durchsucht.
  - `site` – Die aktuelle Site wird durchsucht.
  - `selectedFiles` – Die ausgewählten Dateien werden durchsucht.
  - `selectedText` - Der ausgewählte Text oder Ordnerpfad wird durchsucht.
- Die Eigenschaft *searchSource* ist ein boolescher Wert, der angibt, ob der HTML-Quellcode durchsucht werden soll.
- Die optionale Eigenschaft *{matchCase}* ist ein boolescher Wert, der angibt, ob bei der Suche zwischen Groß- und Kleinschreibung unterschieden werden soll. Wenn diese Eigenschaft nicht explizit angegeben wird, gilt der Standardwert `false`.
- Die optionale Eigenschaft *{matchWholeWord}* ist ein boolescher Wert, der angibt, ob die Übereinstimmungen ganze Wörter sein sollen.
- Die optionale Eigenschaft *{ignoreWhitespace}* ist ein boolescher Wert, der angibt, ob Unterschiede bei Leerräumen ignoriert werden sollen. Der Standardwert für *ignoreWhitespace* ist `false`, wenn die Eigenschaft *useRegularExpressions* den Wert `true` hat. Er lautet `true`, wenn die Eigenschaft *useRegularExpressions* den Wert `false` aufweist.
- Die Eigenschaft *{useRegularExpressions}* ist ein boolescher Wert, der angibt, ob die Eigenschaft `searchString` reguläre Ausdrücke enthält. Wenn diese Eigenschaft nicht explizit angegeben wird, gilt der Standardwert `false`.

Rückgabewerte
Keine.

dreamweaver.setUpFindReplace()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Bereitet eine Text- oder HTML-Quellcode-Suche vor, indem die Suchparameter und der Suchbereich für anschließende Vorgänge vom Typ `dreamweaver.replace()` oder `dreamweaver.replaceAll()` definiert werden.

Argumente
searchObject

Das Argument *searchObject* ist ein Objekt, für das die folgenden Eigenschaften definiert werden können:

- Die Eigenschaft *searchString* ist der zu suchende Text.
- Mit dem Argument *searchWhat* wird festgelegt, wo der Suchvorgang durchgeführt wird. Folgende Werte sind möglich:
  - `document` – Das aktuelle aktive Dokument wird durchsucht.
  - `allOpenDocuments` – Alle geöffneten Dokumente werden durchsucht.
  - `site` – Die aktuelle Site wird durchsucht.
  - `selectedFiles` – Die ausgewählten Dateien werden durchsucht.

  - `selectedText` – Der ausgewählte Text wird durchsucht.
- Die Eigenschaft *replaceString* ist der Text, durch den die Auswahl ersetzt werden soll.
- Die Eigenschaft *searchSource* ist ein boolescher Wert, der angibt, ob der HTML-Quellcode durchsucht werden soll.
- Die optionale Eigenschaft *{matchCase}* ist ein boolescher Wert, der angibt, ob bei der Suche zwischen Groß- und Kleinschreibung unterschieden werden soll. Wenn diese Eigenschaft nicht explizit angegeben wird, gilt der Standardwert `false`.
- Die optionale Eigenschaft *{matchWholeWord}* ist ein boolescher Wert, der angibt, ob die Übereinstimmungen als ganze Wörter betrachtet werden sollen.
- Die optionale Eigenschaft *{ignoreWhitespace}* ist ein boolescher Wert, der angibt, ob Unterschiede bei Leerräumen ignoriert werden sollen. Der Standardwert für die Eigenschaft *ignoreWhitespace* ist `false`, wenn die Eigenschaft *useRegularExpressions* den Wert `true` hat. Der Standardwert lautet `true`, wenn die Eigenschaft *useRegularExpressions* den Wert `false` hat.
- Die Eigenschaft *{useRegularExpressions}* ist ein boolescher Wert, der angibt, ob die Eigenschaft `searchString` reguläre Ausdrücke enthält. Wenn diese Eigenschaft nicht explizit angegeben wird, gilt der Standardwert `false`.

Rückgabewerte
Keine.

dreamweaver.showFindDialog()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Öffnet das Dialogfeld „Suchen“.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dreamweaver.canShowFindDialog()“ auf Seite 524.

dreamweaver.showFindReplaceDialog()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Öffnet das Dialogfeld „Ersetzen“.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dreamweaver.canShowFindDialog()“ auf Seite 524.

Allgemeine Bearbeitungsfunktionen

Mit diesen Funktionen können Sie im Dokumentfenster allgemeine Bearbeitungsvorgänge durchführen. Unter anderem können Text, HTML-Code und Objekte eingefügt, Schrift- und Zeichen-Markups angewendet, geändert und entfernt sowie Tags und Attribute bearbeitet werden.

dom.applyCharacterMarkup()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Wendet das angegebene Zeichen-Markup auf die Auswahl an. Wenn die Auswahl eine Einfügemarke ist, wird das Zeichen-Markup auf den anschließend eingegebenen Text angewendet.

Argumente
tagName

- Das Argument *tagName* ist der Tag-Name, der dem Zeichen-Markup zugewiesen ist. Es muss sich um einen der folgenden Strings handeln: `"b"`, `"cite"`, `"code"`, `"dfn"`, `"em"`, `"i"`, `"kbd"`, `"samp"`, `"s"`, `"strong"`, `"tt"`, `"u"` oder `"var"`.

Rückgabewerte
Keine.

dom.applyFontMarkup()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Wendet das Tag `FONT` mit dem angegebenen Attribut und Wert auf die aktuelle Auswahl an.

Argumente
attribute, value

- Das Argument *attribute* muss `"face"`, `"size"` oder `"color"` lauten.

- Das Argument *value* ist der Wert, der dem Attribut zugewiesen werden soll, beispielsweise `"Arial, Helvetica, sans-serif"`, `"5"` oder `"#FF0000"`.

Rückgabewerte
Keine.

dom.deleteSelection()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Löscht die Auswahl im Dokument.

Argumente
Keine.

Rückgabewerte
Keine.

dom.editAttribute()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Zeigt die für die Bearbeitung des angegebenen Attributs geeignete Benutzeroberfläche an. In der Regel handelt es sich dabei um ein Dialogfeld. Diese Funktion ist nur für das aktive Dokument gültig.

Argumente
attribute

- Bei *attribute* handelt es sich um einen String, der das zu bearbeitende Tag-Attribut angibt.

Rückgabewerte
Keine.

dom.exitBlock()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Beendet den aktuellen Absatz- oder Überschriftenblock und positioniert die Einfügemarke außerhalb aller Blockelemente.

Argumente
Keine.

Rückgabewerte
Keine.

dom.getCharSet()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Gibt das Attribut `charset` im Meta-Tag des Dokuments zurück.

Argumente
Keine.

Rückgabewerte
Die Kodierungs-ID des Dokuments. In einem Dokument mit der Kodierung „Latin1“ wird beispielsweise `iso-8859-1` zurückgegeben.

dom.getFontMarkup()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Ruft für die aktuelle Auswahl im Tag `FONT` den Wert des angegebenen Attributs ab.

Argumente
attribute

- Das Argument *attribute* muss `"face"`, `"size"` oder `"color"` lauten.

Rückgabewerte
Ein String mit dem Wert des angegebenen Attributs oder ein leerer String, wenn das Attribut nicht gesetzt ist.

dom.getLineFromOffset()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Sucht die Zeilennummer eines bestimmten Zeichen-Offsets im Text (HTML- oder JavaScript-Code) der Datei.

Argumente
offset

- Das Argument *offset* ist eine Ganzzahl, die die Zeichenposition relativ zum Anfang der Datei angibt.

Rückgabewerte
Eine Ganzzahl, die die Zeilennummer im Dokument angibt.

dom.getLinkHref()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Ruft den Hyperlink ab, der die aktuelle Auswahl umgibt. Diese Funktion entspricht dem Durchlaufen der übergeordneten Elemente des aktuellen Knotens sowie der diesen Elementen übergeordneten Elemente, bis ein Hyperlink gefunden und dann für diesen die Funktion `getAttribute('HREF')` aufgerufen wird.

Argumente
Keine.

Rückgabewerte
Ein String mit dem Namen der verknüpften Datei im URL-Format „file://“.

dom.getLinkTarget()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Ruft das Ziel des Hyperlinks ab, der die aktuelle Auswahl umgibt. Diese Funktion entspricht dem Durchlaufen der übergeordneten Elemente des aktuellen Knotens sowie der diesen Elementen übergeordneten Elemente, bis ein Hyperlink gefunden und dann für diesen die Funktion `getAttribute('TARGET')` aufgerufen wird.

Argumente
Keine.

Rückgabewerte
Ein String mit dem Wert des Attributs `TARGET` für den Hyperlink bzw. ein leerer String, wenn kein Ziel angegeben ist.

dom.getListTag()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Ruft den Stil der ausgewählten Liste ab.

Argumente
Keine.

Rückgabewerte
Ein String mit dem Tag, das mit der Liste (`"ul"`, `"ol"` oder `"dl"`) verknüpft ist, oder ein leerer String, wenn kein Tag mit der Liste verknüpft ist. Dieser Wert wird immer in Kleinbuchstaben zurückgegeben.

dom.getTextAlignment()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Ruft die Ausrichtung des Blocks ab, der die Auswahl enthält.

Argumente
Keine.

Rückgabewerte
Ein String mit dem Wert des mit dem Block verknüpften Attributs `ALIGN` oder ein leerer String, wenn das Attribut `ALIGN` für das Tag nicht gesetzt ist. Dieser Wert wird immer in Kleinbuchstaben zurückgegeben.

dom.getTextFormat()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Ruft das Blockformat des ausgewählten Texts ab.

Argumente
Keine.

Rückgabewerte
Ein String mit dem zugewiesenen Block-Tag (z. B. `"p"`, `"h1"`, `"pre"` usw.) oder ein leerer String, wenn mit der Auswahl kein Block-Tag verknüpft ist. Dieser Wert wird immer in Kleinbuchstaben zurückgegeben.

dom.hasCharacterMarkup()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob die Auswahl bereits das angegebene Zeichen-Markup aufweist.

Argumente
markupTagName

- Das Argument *markupTagName* ist der Name des Tags, das überprüft wird. Es muss sich um einen der folgenden Strings handeln: `"b"`, `"cite"`, `"code"`, `"dfn"`, `"em"`, `"i"`, `"kbd"`, `"samp"`, `"s"`, `"strong"`, `"tt"`, `"u"` oder `"var"`.

Rückgabewerte
Ein boolescher Wert, der angibt, ob die gesamte Auswahl das angegebene Zeichen-Markup aufweist. Wenn nur ein Teil der Auswahl über das angegebene Markup verfügt, wird der Wert `false` zurückgegeben.

dom.indent()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Weist der Auswahl mit `BLOCKQUOTE`-Tags einen Einzug zu. Wenn die Auswahl ein Listenelement ist, wird für sie ein Einzug erzeugt, indem das ausgewählte Element in eine verschachtelte Liste konvertiert wird. Diese verschachtelte Liste hat den gleichen Typ wie die äußere Liste und enthält mit der ursprünglichen Auswahl nur ein Element.

Argumente
Keine.

Rückgabewerte
Keine.

dom.insertHTML()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Fügt im Dokument an der Einfügemarke HTML-Inhalt ein.

Argumente
contentToInsert, {bReplaceCurrentSelection}

- Das Argument *contentToInsert* ist der einzufügende Inhalt.
- Das optionale Argument *bReplaceCurrentSelection* ist ein boolescher Wert, der angibt, ob der Inhalt die aktuelle Auswahl ersetzen soll. Wenn das Argument *bReplaceCurrentSelection* den Wert `true` aufweist, ersetzt der Inhalt die aktuelle Auswahl. Wenn der Wert `false` lautet, wird der Inhalt nach der aktuellen Auswahl eingefügt.

Rückgabewerte
Keine.

Beispiel
Mit dem folgenden Code wird der HTML-String `<b>130</b>` in das aktuelle Dokument eingefügt:

```
var theDOM = dw.getDocumentDOM();
theDOM.insertHTML('<b>130</b>');
```

Das Ergebnis wird im Dokumentfenster angezeigt.

dom.insertObject()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Fügt das angegebene Objekt ein und fordert den Benutzer gegebenenfalls zur Eingabe von Parametern auf.

Argumente
objectName

- Das Argument *objectName* ist der Name eines Objekts im Ordner „Configuration/Objects“.

Rückgabewerte
Keine.

Beispiel
Ein Aufruf der Funktion `dom.insertObject('Button')` fügt nach der aktuellen Auswahl eine Formularschaltfläche in das aktive Dokument ein. Wenn kein Objekt ausgewählt ist, fügt diese Funktion die Schaltfläche an der aktuellen Einfügemarke ein.

Hinweis: *Auch wenn Objektdateien in eigenen Ordnern gespeichert werden können, dürfen die Dateinamen jeweils nur einmal vorkommen. Wenn sich sowohl im Ordner „Forms“ als auch im Ordner „MyObjects“ eine Datei mit dem Namen „Button.htm“ befindet, können diese beiden Dateien nicht unterschieden werden.*

dom.insertText()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Fügt im Dokument an der Einfügemarke Textinhalt ein.

Argumente
contentToInsert, {bReplaceCurrentSelection}

- Das Argument *contentToInsert* ist der einzufügende Inhalt.
- Das optionale Argument *bReplaceCurrentSelection* ist ein boolescher Wert, der angibt, ob der Inhalt die aktuelle Auswahl ersetzen soll. Wenn das Argument *bReplaceCurrentSelection* den Wert `true` aufweist, ersetzt der Inhalt die aktuelle Auswahl. Wenn der Wert `false` lautet, wird der Inhalt nach der aktuellen Auswahl eingefügt.

Rückgabewerte
Keine.

Beispiel
Mit dem folgenden Code wird der Text `<b>130</b>` in das aktuelle Dokument eingefügt:

```
var theDOM = dreamweaver.getDocumentDOM();
theDOM.insertText('<b>130</b>');
```

Die Ergebnisse werden im Dokumentfenster angezeigt.

dom.newBlock()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Erstellt einen neuen Block mit dem gleichen Tag und den gleichen Attributen wie der Block, in dem sich die aktuelle Auswahl befindet, oder erstellt einen neuen Absatz, wenn sich der Zeiger außerhalb aller Blöcke befindet.

Argumente
Keine.

Rückgabewerte
Keine.

Beispiel
Wenn sich die aktuelle Auswahl in einem zentrierten Absatz befindet, wird durch Aufruf der Funktion `dom.newBlock()` der Code `p align="center"` nach dem aktuellen Absatz eingefügt.

dom.notifyFlashObjectChanged()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Teilt Dreamweaver mit, dass sich die aktuelle Flash-Objektdatei geändert hat. Dreamweaver aktualisiert die Vorschau und nimmt die erforderliche Größenänderung vor, wobei das ursprüngliche Verhältnis zwischen Breite und Höhe beibehalten wird.

Argumente
Keine.

Rückgabewerte
Keine.

dom.outdent()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Weist der Auswahl einen hängenden Einzug zu.

Argumente
Keine.

Rückgabewerte
Keine.

dom.removeCharacterMarkup()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Entfernt das angegebene Zeichen-Markup aus der Auswahl.

Argumente
tagName

- Das Argument *tagName* ist der Tag-Name, der dem Zeichen-Markup zugewiesen ist. Es muss sich um einen der folgenden Strings handeln: `"b"`, `"cite"`, `"code"`, `"dfn"`, `"em"`, `"i"`, `"kbd"`, `"samp"`, `"s"`, `"strong"`, `"tt"`, `"u"` oder `"var"`.

Rückgabewerte
Keine.

dom.removeFontMarkup()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Entfernt das angegebene Attribut und den entsprechenden Wert aus einem `FONT`-Tag. Wenn nach dem Entfernen des Attributs nur das Tag `FONT` verbleibt, wird auch das Tag `FONT` entfernt.

Argumente
attribute

- Das Argument *attribute* muss `"face"`, `"size"` oder `"color"` lauten.

Rückgabewerte
Keine.

dom.resizeSelection()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Ändert die Größe des ausgewählten Objekts entsprechend den angegebenen Abmessungen.

Argumente
newWidth, newHeight

- Das Argument *newWidth* gibt die neue Breite an, die die Funktion auf das ausgewählte Objekt anwendet.
- Das Argument *newHeight* gibt die neue Höhe an, die die Funktion auf das ausgewählte Objekt anwendet.

Rückgabewerte
Keine.

dom.setAttributeWithErrorChecking()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Setzt das festgelegte Attribut für die aktuelle Auswahl auf den angegebenen Wert. Wenn der Typ des Werts nicht korrekt ist oder sich der Wert außerhalb des zulässigen Bereichs befindet, wird eine Meldung angezeigt. Diese Funktion ist nur für das aktive Dokument gültig.

Argumente
attribute, value

- Das Argument *attribute* gibt das für die aktuelle Auswahl festzulegende Attribut an.
- Das Argument *value* gibt den Wert für das Attribut an.

Rückgabewerte
Keine.

dom.setLinkHref()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Macht aus der Auswahl einen Hyperlink oder ändert den URL-Wert des HREF-Tags, das die aktuelle Auswahl einschließt.

Argumente
linkHREF

- Das Argument *linkHREF* ist die den Hyperlink enthaltende URL (Pfad relativ zum Dokument oder zum Stammordner bzw. absolute URL). Wenn dieses Argument nicht angegeben ist, wird das Dialogfeld „HTML-Datei auswählen“ angezeigt.

Rückgabewerte
Keine.

Enabler
Siehe „dom.canSetLinkHref()“ auf Seite 514.

dom.setLinkTarget()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Legt das Ziel des Hyperlinks fest, der die aktuelle Auswahl umgibt. Diese Funktion entspricht dem Durchlaufen der übergeordneten Elemente des aktuellen Knotens sowie der diesen Elementen übergeordneten Elemente, bis ein Hyperlink gefunden und dann für diesen die Funktion `setAttribute('TARGET')` aufgerufen wird.

Argumente
{linkTarget}

- Das optionale Argument *linkTarget* ist ein String, der einen Frame, ein Fenster oder eines der reservierten Ziele (`"_self"`, `"_parent"`, `"_top"` oder `"_blank"`) angibt. Wenn das Argument nicht angegeben ist, wird das Dialogfeld „Ziel einstellen“ angezeigt.

Rückgabewerte
Keine.

dom.setListBoxKind()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Ändert die Art des ausgewählten `SELECT`-Menüs.

Argumente
kind

- Das Argument *kind* muss entweder `"menu"` oder `"list box"` lauten.

Rückgabewerte
Keine.

dom.showListPropertiesDialog()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Öffnet das Dialogfeld „Listeneigenschaften“.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dom.canShowListPropertiesDialog()“ auf Seite 514.

dom.setListTag()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Legt den Stil der ausgewählten Liste fest.

Argumente
listTag

- Das Argument *listTag* ist das Tag, das der Liste zugeordnet ist. Es muss sich um `"ol"`, `"ul"`, `"dl"` oder einen leeren String handeln.

Rückgabewerte
Keine.

dom.setTextAlignment()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Setzt das Attribut `ALIGN` des Blocks, der die Auswahl enthält, auf den angegebenen Wert.

Argumente
alignValue

- Das Argument *alignValue* muss `"left"`, `"center"` oder `"right"` sein.

Rückgabewerte
Keine.

dom.setTextFieldKind()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Legt das Format des ausgewählten Textfelds fest.

Argumente
fieldType

- Das Argument *fieldType* muss `"input"`, `"textarea"` oder `"password"` sein.

Rückgabewerte
Keine.

dom.setTextFormat()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Legt das Blockformat des ausgewählten Texts fest.

Argumente
blockFormat

- Das Argument *blockFormat* ist ein String, der eines der folgenden Formate angibt: `""` (kein Format), `"p"`, `"h1"`, `"h2"`, `"h3"`, `"h4"`, `"h5"`, `"h6"` oder `"pre"`.

Rückgabewerte
Keine.

dom.showFontColorDialog()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Öffnet das Dialogfeld „Farbauswahl“.

Argumente
Keine.

Rückgabewerte
Keine.

dreamweaver.deleteSelection()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Löscht die Auswahl im aktiven Dokument oder im Bedienfeld „Dateien“. Beim Macintosh wird das in einem Dialogfeld oder einem schwebenden Bedienfeld aktive Textfeld gelöscht.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dreamweaver.canDeleteSelection()“ auf Seite 518.

dreamweaver.editFontList()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Öffnet das Dialogfeld „Schriftliste bearbeiten“.

Argumente
Keine.

Rückgabewerte
Keine.

dreamweaver.getFontList()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Ruft eine Liste aller Schriftartgruppen ab, die im Eigenschafteninspektor für Text und im Dialogfeld „Stildefinition“ angezeigt werden.

Argumente
Keine.

Rückgabewerte
Ein String-Array mit den einzelnen Elementen der Schriftliste.

Beispiel
In der Standardinstallation von Dreamweaver wird beim Aufruf der Funktion `dreamweaver.getFontList()` ein Array mit den folgenden Elementen zurückgegeben:

- `"Arial, Helvetica, sans-serif"`
- `"Times New Roman, Times, serif"`
- `"Courier New, Courier, mono"`
- `"Georgia, Times New Roman, Times, serif"`
- `"Verdana, Arial, Helvetica, sans-serif"`

dreamweaver.getFontStyles()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Gibt die Stile zurück, die von einer angegebenen TrueType-Schriftart unterstützt werden.

Argumente
fontName

- Das Argument *fontName* ist ein String, der den Namen der Schriftart enthält.

Rückgabewerte
Ein Array mit drei booleschen Werten, die angeben, welche Stile die Schriftart unterstützt. Der erste Wert gibt an, ob die Schriftart *bold* unterstützt, der zweite Wert definiert die Unterstützung für *italic* und der dritte Wert die Unterstützung für *bold* und *italic*.

dreamweaver.getKeyState()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Stellt fest, ob die angegebene Taste gedrückt ist.

Argumente
key

- Das Argument *key* muss einen der folgenden Werte haben: `"Cmd"`, `"Ctrl"`, `"Alt"` oder `"Shift"`. Unter Windows beziehen sich `"Cmd"` und `"Ctrl"` auf die Strg-Taste. Beim Macintosh bezieht sich `"Alt"` auf die Wahltaste.

Rückgabewerte
Ein boolescher Wert, der angibt, ob die entsprechende Taste gedrückt ist.

Beispiel
Mit dem folgenden Code wird vor dem Durchführen eines Vorgangs überprüft, ob die Umschalttaste und die Strg-Taste (Windows) bzw. die Umschalt- und Befehlstaste (Macintosh) gedrückt sind.

```
if (dw.getKeyState("Shift") && dw.getKeyState("Cmd")){
    // execute code
}
```

dreamweaver.getNaturalSize()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Gibt die Breite und die Höhe eines grafischen Objekts zurück.

Argumente
url

- Das Argument *url* verweist auf ein grafisches Objekt, dessen Abmessungen gesucht werden. Dieses Objekt muss in Dreamweaver unterstützt werden (GIF, JPEG, PNG, Flash und Shockwave). Als Argument für die Funktion `getNaturalSize()` muss eine absolute, auf eine lokale Datei verweisende URL angegeben werden. Relative URLs sind nicht zulässig.

Rückgabewerte
Ein Array mit zwei Ganzzahlen, wobei die erste Ganzzahl die Breite und die zweite die Höhe des Objekts definiert.

dreamweaver.getSystemFontList()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Gibt eine Schriftliste für das System zurück. Mit dieser Funktion können entweder alle Schriftarten oder nur TrueType-Schriften abgerufen werden.

Argumente
`fontTypes`

- Das Argument *`fontTypes`* ist ein String, der entweder den Wert `all` oder `TrueType` enthält.

Rückgabewerte
Ein Array von Strings, die alle Schriftartnamen enthalten. Wenn keine Schriftarten gefunden werden, wird der Wert `null` zurückgegeben.

dreamweaver.getSystemFontName()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Gibt den Namen der Systemschriftart zurück.

Argumente
Keine.

Rückgabewerte
Ein String mit dem Namen der Systemschriftart.

Druckfunktion

Mit der Druckfunktion kann der Benutzer Code aus der Codeansicht drucken.

dreamweaver.printCode()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Unter Windows druckt diese Funktion den gesamten Code oder ausgewählte Codeabschnitte aus der Codeansicht. Unter Mac OS druckt diese Funktion den gesamten Code oder einen Seitenbereich des Codes.

Argumente

showPrintDialog, document

- Das Argument *showPrintDialog* ist entweder `true` oder `false`. Wenn dieses Argument den Wert `true` hat, zeigt die Funktion `dreamweaver.PrintCode()` unter Windows das Dialogfeld „Drucken“ an, in dem der Benutzer angeben kann, ob der gesamte Text oder eine Textauswahl gedruckt werden soll. Auf dem Macintosh zeigt `dreamweaver.PrintCode()` das Dialogfeld „Drucken“ an, in dem der Benutzer angeben kann, ob der gesamte Text oder ein Seitenbereich gedruckt werden soll.

  Wenn das Argument den Wert `false` hat, verwendet `dreamweaver.PrintCode()` die vorherige Auswahl des Benutzers. Der Standardwert ist `false`.

- Das Argument *document* ist das DOM (Dokumentobjektmodell) des zu druckenden Dokuments. Informationen zum Abrufen des DOM für ein Dokument finden Sie unter „dreamweaver.getDocumentDOM()“ auf Seite 267.

Rückgabewerte

Der boolesche Wert `true`, wenn der Code gedruckt werden kann, andernfalls `false`.

Beispiel

Im folgenden Beispiel wird `dw.PrintCode()` aufgerufen, um das Dialogfeld „Drucken“ für das Dokument des Benutzers zu öffnen. Wenn die Funktion den Wert `false` zurückgibt, zeigt der Code eine Warnung an, um den Benutzer zu informieren, dass der Druckvorgang nicht ausgeführt werden kann.

```
var theDOM = dreamweaver.getDocumentDOM("document");
if(!dreamweaver.PrintCode(true, theDOM))
{
    alert("Unable to execute your print request!");
}
```

Funktionen für den Quick Tag Editor

Die Funktionen für den Quick Tag Editor werden auf Tags angewendet, die sich entweder innerhalb der aktuellen Auswahl befinden oder diese umgeben. Tags können aus der Hierarchie entfernt und ausgewählte Bereiche in neue Tags eingeschlossen werden. Zudem kann der Quick Tag Editor aufgerufen werden, mit dem spezifische Attribute des Tags bearbeitet werden können.

dom.selectChild()

Verfügbarkeit

Dreamweaver 3.

Beschreibung

Wählt ein der aktuellen Auswahl untergeordnetes Objekt aus. Das Aufrufen dieser Funktion entspricht dem Auswählen des nächsten Tags rechts im Tag-Selektor im unteren Bereich des Dokumentfensters.

Argumente

Keine.

Rückgabewerte
Keine.

dom.selectParent()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Wählt das der aktuellen Auswahl übergeordnete Objekt aus. Das Aufrufen dieser Funktion entspricht dem Auswählen des nächsten Tags links im Tag-Selektor im unteren Bereich des Dokumentfensters.

Argumente
Keine.

Rückgabewerte
Keine.

dom.stripTag()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Entfernt das die aktuelle Auswahl umgebende Tag, jedoch nicht den Inhalt. Wenn die Auswahl mehrere Tags oder kein Tag enthält, wird eine Fehlermeldung ausgegeben.

Argumente
Keine.

Rückgabewerte
Keine.

dom.wrapTag()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Fügt das angegebene Tag um die aktuelle Auswahl ein. Wenn die Auswahl nicht ausgeglichen ist, wird eine Fehlermeldung ausgegeben.

Argumente
`startTag`, *`{bAlwaysBalance}`*, *`{bMakeLegal}`*

- Das Argument *`startTag`* ist der Quellcode des öffnenden Tags.

- Das Argument *bAlwaysBalance* ist ein boolescher Wert, der angibt, ob die Auswahl zuvor ausgeglichen werden soll. Dieses Argument ist optional.
- Das Argument *bMakeLegal* ist ein boolescher Wert, der angibt, ob sichergestellt werden soll, dass gültiger HTML-Code generiert wird. Dieses Argument ist optional.

Rückgabewerte
Keine.

Beispiel
Mit dem folgenden Code wird ein Hyperlink um die aktuelle Auswahl gelegt:

```
var theDOM = dw.getDocumentDOM();
var theSel = theDOM.getSelectedNode();
if (theSel.nodeType == Node.TEXT_NODE){
    theDOM.wrapTag('<a href="foo.html">');
}
```

dreamweaver.showQuickTagEditor()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Zeigt für die aktuelle Auswahl den Quick Tag Editor an.

Argumente
{nearWhat}, {mode}

- Für das optionale *nearWhat*-Argument muss entweder `"selection"` oder `"tag selector"` angegeben werden. Wenn das Argument nicht angegeben ist, gilt der Standardwert `"selection"`.
- Für das optionale *mode*-Argument muss entweder `"default"`, `"wrap"`, `"insert"` oder `"edit"` angegeben werden. Wenn für *mode* der Wert `"default"` ausgewählt oder das Argument nicht angegeben wurde, wird der Modus für die aktuelle Auswahl heuristisch bestimmt. Das Argument *mode* wird ignoriert, wenn für *nearWhat* der Wert `"tag selector"` angegeben wurde.

Rückgabewerte
Keine.

Funktionen für die Codeansicht

Zu den Funktionen für die Codeansicht gehören Bearbeitungsvorgänge am Quellcode eines Dokuments (mit Auswirkungen auf die Entwurfsansicht). Mit den Funktionen in diesem Abschnitt können Sie der Codeansicht in einer geteilten Dokumentansicht oder im Fenster des Codeinspektors Navigationssteuerelemente hinzufügen.

dom.formatRange()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Wendet auf einen angegebenen Zeichenbereich in der Codeansicht entsprechend den Einstellungen im Dialogfeld „Voreinstellungen“ > „Codeformat“ die automatische Syntaxformatierung von Dreamweaver an.

Argumente
startOffset, endOffset

- Das Argument *startOffset* ist eine Ganzzahl, die den Beginn des angegebenen Bereichs als Offset ab dem Anfang des Dokuments angibt.
- Das Argument *endOffset* ist eine Ganzzahl, die das Ende des angegebenen Bereichs als Offset ab dem Anfang des Dokuments angibt.

Rückgabewerte
Keine.

dom.formatSelection()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Wendet auf den ausgewählten Inhalt entsprechend den Einstellungen im Dialogfeld „Voreinstellungen“ > „Codeformat“ die automatische Syntaxformatierung von Dreamweaver an (entspricht dem Auswählen von „Befehle“ > „Quellenformatierung auf Auswahl anwenden“).

Argumente
Keine.

Rückgabewerte
Keine.

dom.getShowNoscript()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Ruft den aktuellen Status der Inhaltsoption `noscript` ab (über die Menüoption „Ansicht“ > „Noscript-Inhalt“). Das `noscript`-Tag (standardmäßig aktiviert) kennzeichnet Skriptinhalt, der (optional) im Browser wiedergegeben werden kann oder nicht.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der Inhalt des Tags `noscript` derzeit wiedergegeben wird, andernfalls `false`.

dom.getAutoValidationCount()

Verfügbarkeit
Dreamweaver MX 2004

Beschreibung
Ruft die Anzahl der Fehler, Warnungen und Informationsmeldungen für die letzte automatische Prüfung (auch als Inline-Prüfung bezeichnet) ab. Derzeit wird bei der automatischen Prüfung nur eine Zielbrowser-Prüfung durchgeführt (siehe „dom.runValidation()“ auf Seite 276).

Hinweis: *Diese Funktion gibt nur die Ergebnisse zurück, die derzeit im Ergebnisfenster des Dokuments angezeigt werden. Um sicherzustellen, dass die Werte der Zähler aktuell sind, können Sie vor dem Aufrufen dieser Funktion* `dom.runValidation()` *aufrufen.*

Argumente
Keine.

Rückgabewerte
Ein Objekt mit den folgenden Eigenschaften:

- Die Eigenschaft `numError` enthält die Anzahl der Fehler.
- Die Eigenschaft `numWarning` enthält die Anzahl der Warnungen.
- Die Eigenschaft `numInfo` enthält die Anzahl der Informationsmeldungen.

Beispiel

```
theDom = dw.getDocumentDOM();
theDom.runValidation();
theDom.getAutoValidationCount();
```

dom.isDesignViewUpdated()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Legt fest, ob der Inhalt der Entwurfsansicht und der Textansicht bei den Dreamweaver-Vorgängen, die einen gültigen Dokumentstatus erfordern, synchronisiert wird.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn die Entwurfsansicht (WYSIWYG) mit dem Text in der Textansicht synchronisiert wird, andernfalls `false`.

dom.isSelectionValid()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Legt fest, ob eine Auswahl gültig ist, d. h., ob sie derzeit mit der Entwurfsansicht synchronisiert ist oder vor einer Operation verschoben werden muss.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn es sich bei der aktuellen Auswahl um gültigen Code handelt, `false`, wenn das Dokument nicht synchronisiert wurde, da die Auswahl nicht aktualisiert ist.

dom.setShowNoscript()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Aktiviert oder deaktiviert die Inhaltsoption `noscript` (entspricht der Verwendung der Menüoption „Ansicht“ > „Noscript-Inhalt“). Das `noscript`-Tag (standardmäßig aktiviert) kennzeichnet Skriptinhalt, der (optional) im Browser wiedergegeben werden kann oder nicht.

Argumente
{bShowNoscript}

- Das optionale Argument *bShowNoscript* ist ein boolescher Wert, der angibt, ob der Inhalt des Tags `noscript` dargestellt werden soll: `true`, wenn der Inhalt des Tags `noscript` wiedergegeben werden soll, andernfalls `false`.

Rückgabewerte
Keine.

dom.source.arrowDown()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Verschiebt die Einfügemarke in der Codeansicht des Dokuments zeilenweise nach unten. Wenn bereits ein Bereich ausgewählt wurde, wird die Auswahl durch diese Funktion zeilenweise erweitert.

Argumente
{nTimes}, {bShiftIsDown}

- Das optionale Argument *nTimes* gibt die Anzahl der Zeilen an, um die die Einfügemarke verschoben werden soll. Wenn *nTimes* nicht angegeben wird, gilt der Standardwert 1.
- Das optionale Argument *bShiftIsDown* ist ein boolescher Wert, der angibt, ob der Inhalt ausgewählt wurde. Wenn *bShiftIsDown* den Wert `true` hat, wird Inhalt ausgewählt.

Rückgabewerte
Keine.

dom.source.arrowLeft()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Verschiebt die Einfügemarke in der aktuellen Zeile der Codeansicht nach links. Wenn bereits ein Bereich ausgewählt wurde, wird die Auswahl durch diese Funktion nach links erweitert.

Argumente
{nTimes}, {bShiftIsDown}

- Das optionale Argument *nTimes* gibt die Anzahl der Zeichen an, um die die Einfügemarke verschoben werden soll. Wenn *nTimes* nicht angegeben wird, gilt der Standardwert 1.
- Das optionale Argument *bShiftIsDown* ist ein boolescher Wert, der angibt, ob der Inhalt ausgewählt wurde. Wenn *bShiftIsDown* den Wert `true` hat, wird Inhalt ausgewählt.

Rückgabewerte
Keine.

dom.source.arrowRight()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Verschiebt die Einfügemarke in der aktuellen Zeile der Codeansicht nach rechts. Wenn bereits ein Bereich ausgewählt wurde, wird die Auswahl durch diese Funktion nach rechts erweitert.

Argumente
{nTimes}, {bShiftIsDown}

- Das optionale Argument *nTimes* gibt die Anzahl der Zeichen an, um die die Einfügemarke verschoben werden soll. Wenn *nTimes* nicht angegeben wird, gilt der Standardwert 1.
- Das optionale Argument *bShiftIsDown* ist ein boolescher Wert, der angibt, ob Inhalt ausgewählt wird. Wenn *bShiftIsDown* den Wert `true` hat, wird der Inhalt ausgewählt.

Rückgabewerte
Keine.

dom.source.arrowUp()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Verschiebt die Einfügemarke in der Codeansicht des Dokuments zeilenweise nach oben. Wenn bereits ein Bereich ausgewählt wurde, wird die Auswahl durch diese Funktion zeilenweise erweitert.

Argumente
{nTimes}, {bShiftIsDown}

- Das Argument *nTimes* gibt die Anzahl der Zeilen an, um die die Einfügemarke verschoben werden soll. Wenn *nTimes* nicht angegeben wird, gilt der Standardwert 1.
- Das Argument *bShiftIsDown* ist ein boolescher Wert, der angibt, ob Inhalt ausgewählt wird. Wenn *bShiftIsDown* den Wert `true` hat, wird der Inhalt ausgewählt.

Rückgabewerte
Keine.

dom.source.balanceBracesTextView()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Bei dieser Funktion handelt es sich um eine Erweiterung der Codeansicht, die ausgeglichene Klammern ermöglicht. Sie können `dom.source.balanceBracesTextView()` aufrufen, um die aktuelle Auswahl oder Einfügemarke zu erweitern. Die Erweiterung erstreckt sich vom Anfang bis zum Ende der Anweisung in Klammern. So werden folgende Zeichen ausgeglichen: `[]`, `{}` und `()`. Bei anschließenden Aufrufen der Funktion wird die Auswahl durch weitere verschachtelte Anweisungen in Klammern erweitert.

Argumente
Keine.

Rückgabewerte
Keine.

dom.source.doCodeNavItem()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Diese Funktion lädt den Code-Navigator und füllt ihn mit Zielen für die aktuelle Auswahl. Sie führt jedoch keine Navigationsvorgänge aus und öffnet keine zugehörigen Dateien.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der Code-Navigator geöffnet ist, `false`, wenn der Code-Navigator nicht geöffnet werden kann, da die aktuelle Auswahl keine Navigationsziele enthält.

dom.source.endOfDocument()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Platziert die Einfügemarke in der Codeansicht am Ende des aktuellen Dokuments. Wenn bereits ein Bereich ausgewählt wurde, wird die Auswahl durch diese Funktion bis zum Ende des Dokuments erweitert.

Argumente
bShiftIsDown

- Das Argument *bShiftIsDown* ist ein boolescher Wert, der angibt, ob der Inhalt ausgewählt wurde. Wenn *bShiftIsDown* den Wert `true` hat, wird Inhalt ausgewählt.

Rückgabewerte
Keine.

dom.source.endOfLine()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Platziert die Einfügemarke am Ende der aktuellen Zeile. Wenn bereits ein Bereich ausgewählt wurde, wird die Auswahl durch diese Funktion bis zum Ende der aktuellen Zeile erweitert.

Argumente
bShiftIsDown

- Das Argument *bShiftIsDown* ist ein boolescher Wert, der angibt, ob der Inhalt ausgewählt wurde. Wenn *bShiftIsDown* den Wert `true` hat, wird Inhalt ausgewählt.

Rückgabewerte
Keine.

dom.source.endPage()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Verschiebt die Einfügemarke an das Ende der aktuellen Seite oder an das Ende der nächsten Seite (wenn die Einfügemarke sich bereits am Seitenende befindet). Wenn bereits ein Bereich ausgewählt wurde, wird die Auswahl durch diese Funktion seitenweise erweitert.

Argumente
{nTimes}, {bShiftIsDown}

- Das optionale Argument *nTimes* gibt die Anzahl der Seiten an, um die die Einfügemarke verschoben werden soll. Wenn *nTimes* nicht angegeben wird, gilt der Standardwert 1.
- Das optionale Argument *bShiftIsDown* ist ein boolescher Wert, der angibt, ob der Inhalt ausgewählt wurde. Wenn `bShiftIsDown` den Wert `true` hat, wird Inhalt ausgewählt.

Rückgabewerte
Keine.

dom.source.getCurrentLines()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Gibt die Zeilennummern für die angegebenen Offset-Positionen ab dem Anfang des Dokuments zurück.

Argumente
Keine.

Rückgabewerte
Die Zeilennummern für die aktuelle Auswahl.

dom.source.getSelection()

Beschreibung
Ruft die Auswahl im aktuellen Dokument ab, die als Zeichen-Offset in der Codeansicht des Dokuments angegeben ist.

Argumente
Keine.

Rückgabewerte
Zwei Ganzzahlen, die Offsets ab dem Anfang des Quelldokuments angeben. Die erste Ganzzahl gibt den Anfang der Auswahl an, die zweite das Ende der Auswahl. Wenn die beiden Zahlen identisch sind, handelt es sich bei der Auswahl um eine Einfügemarke. Wenn das Quelldokument keine Auswahl enthält, gilt für beide Zahlen der Wert -1.

dom.source.getLineFromOffset()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Setzt einen Offset in das Quelldokument.

Argumente
Keine.

Rückgabewerte
Die zugehörige Zeilennummer oder -1, wenn der Offset negativ ist oder über das Dateiende hinauszeigt.

dom.source.getText()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Gibt den String zwischen den angegebenen Offsets im Quelldokument zurück.

Argumente
startOffset, endOffset

- Das Argument *startOffset* ist eine Ganzzahl, die den Offset ab dem Anfang des Dokuments angibt.
- Das Argument *endOffset* ist eine Ganzzahl, die das Ende des Dokuments angibt.

Rückgabewerte
Ein String mit dem Text, der im Quellcode zwischen den Offsets *start* und *end* steht.

dom.source.getValidationErrorsForOffset()

Verfügbarkeit
Dreamweaver MX 2004

Beschreibung
Gibt eine Liste der Prüfungsfehler am angegebenen Offset zurück oder sucht ab dem Offset nach dem nächsten Fehler. Wenn keine Fehler gefunden werden, wird `null` zurückgegeben.

Argumente
offset, {searchDirection}

- Das Argument *offset* ist eine Zahl, die den Offset im Code angibt, für den die Funktion Fehler zurückgibt.
- Das optionale Argument *searchDirection* ist ein String, der einen der folgenden Werte enthält: `"empty"`, `"forward"` oder `"back"`. Wenn das Argument angegeben wird, sucht die Funktion vom entsprechenden Offset aus vorwärts oder rückwärts bis zu den nächsten Zeichen, die Fehler aufweisen, und gibt diese zurück. Wenn das Argument nicht angegeben ist, sucht die Funktion nur nach Fehlern am entsprechenden Offset.

Rückgabewerte
Ein Objekt-Array oder der Wert `null`. Jedes Objekt im Array hat folgende Eigenschaften:

- Das Objekt `message` ist ein String, der die Fehlermeldung enthält.
- Das Objekt `floaterName` ist ein String, der den Namen des Ergebnisfensters enthält. Sie können diesen Wert an die Funktionen `showResults()` oder `setFloaterVisibility()` übergeben.
- Das Objekt `floaterIndex` ist ein Index der Elemente in der Ergebnisliste für schwebende Fenster.
- Das Objekt `start` ist der Startindex des unterstrichenen Codes.
- Das Objekt `end` ist der schließende Index des unterstrichenen Codes.

Hinweis: *Die zurückgegebenen Indizes für schwebende Fenster sollten nicht gespeichert werden, da sie sich regelmäßig ändern, beispielsweise wenn Dokumente geöffnet und geschlossen werden.*

Beispiel
Im folgenden Beispiel wird `getValidationErrorsForOffset()` aufgerufen, um nach Fehlern am Offset der aktuellen Auswahl zu suchen. Wenn die Funktion einen Fehler zurückgibt, wird die Funktion `alert()` aufgerufen, um eine Fehlermeldung anzuzeigen.

```
var offset = dw.getDocumentDOM().source.getSelection()[0];
var errors = dw.getDocumentDOM().source.getValidationErrorsForOffset(offset);
if ( errors && errors.length > 0 )
    alert( errors[0].message );
```

dom.source.indentTextview()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Verschiebt den in der Codeansicht ausgewählten Text um einen Tabulatorstopp nach rechts.

Argumente
Keine.

Rückgabewerte
Keine.

dom.source.insert()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Fügt den angegebenen String am definierten Offset ab dem Anfang der Quelldatei in den Quellcode ein. Wenn der Offset nicht größer oder gleich Null ist, schlägt der Einfügevorgang fehl und die Funktion gibt `false` zurück.

Argumente
offset, string

- Das Argument *offset* ist der Offset ab dem Anfang der Datei, ab dem der String eingefügt werden soll.
- Das Argument *string* ist der einzufügende String.

Rückgabewerte
Ein boolescher Wert: `true` bei einem erfolgreichen Vorgang, andernfalls `false`.

dom.source.nextWord()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Verschiebt die Einfügemarke in der Codeansicht an den Anfang des nächsten Worts (oder der nächsten Wörter, sofern angegeben). Wenn bereits ein Bereich ausgewählt wurde, wird die Auswahl durch diese Funktion nach rechts erweitert.

Argumente
{nTimes}, {bShiftIsDown}

- Das optionale Argument *nTimes* gibt die Anzahl der Wörter an, um die die Einfügemarke verschoben werden soll. Wenn *nTimes* nicht angegeben wird, gilt der Standardwert 1.
- Das optionale Argument *bShiftIsDown* ist ein boolescher Wert, der angibt, ob der Inhalt ausgewählt wurde. Wenn *bShiftIsDown* den Wert `true` hat, wird Inhalt ausgewählt.

Rückgabewerte
Keine.

dom.source.outdentTextview()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Verschiebt den in der Codeansicht ausgewählten Text um einen Tabulatorstopp nach links.

Argumente
Keine.

Rückgabewerte
Keine.

dom.source.pageDown()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Verschiebt die Einfügemarke in der Codeansicht des Dokuments seitenweise nach unten. Wenn bereits ein Bereich ausgewählt wurde, wird die Auswahl durch diese Funktion seitenweise erweitert.

Argumente
{nTimes}, {bShiftIsDown}

- Das optionale Argument *nTimes* gibt die Anzahl der Seiten an, um die die Einfügemarke verschoben werden soll. Wenn *nTimes* nicht angegeben wird, gilt der Standardwert 1.
- Das optionale Argument *bShiftIsDown* ist ein boolescher Wert, der angibt, ob der Inhalt ausgewählt wurde. Wenn *bShiftIsDown* den Wert `true` hat, wird Inhalt ausgewählt.

Rückgabewerte
Keine.

dom.source.pageUp()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Verschiebt die Einfügemarke in der Codeansicht des Dokuments seitenweise nach oben. Wenn bereits ein Bereich ausgewählt wurde, wird die Auswahl durch diese Funktion seitenweise erweitert.

Argumente
{nTimes}, {bShiftIsDown}

- Das optionale Argument *nTimes* gibt die Anzahl der Seiten an, um die die Einfügemarke verschoben werden soll. Wenn *nTimes* nicht angegeben wird, gilt der Standardwert 1.
- Das optionale Argument *bShiftIsDown* ist ein boolescher Wert, der angibt, ob der Inhalt ausgewählt wurde. Wenn *bShiftIsDown* den Wert `true` hat, wird Inhalt ausgewählt.

Rückgabewerte
Keine.

dom.source.previousWord()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Verschiebt die Einfügemarke in der Codeansicht an den Anfang des vorherigen Worts (oder der vorherigen Wörter, sofern angegeben). Wenn bereits ein Bereich ausgewählt wurde, wird die Auswahl durch diese Funktion nach links erweitert.

Argumente
{nTimes}, {bShiftIsDown}

- Das optionale Argument *nTimes* gibt die Anzahl der Wörter an, um die die Einfügemarke verschoben werden soll. Wenn *nTimes* nicht angegeben wird, gilt der Standardwert 1.
- Das optionale Argument *bShiftIsDown* ist ein boolescher Wert, der angibt, ob der Inhalt ausgewählt wurde. Wenn *bShiftIsDown* den Wert `true` hat, wird Inhalt ausgewählt.

Rückgabewerte
Keine.

dom.source.replaceRange()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Ersetzt den Quelltext zwischen *startOffset* und *endOffset* durch *string*. Wenn *startOffset* größer als *endOffset* ist oder wenn es sich bei einem der Offsets nicht um eine positive Ganzzahl handelt, hat diese Funktion keine Wirkung. In diesem Fall wird `false` zurückgegeben. Wenn der Wert für *endOffset* die Anzahl der Zeichen in der Datei übersteigt, wird der Bereich zwischen *startOffset* und dem Ende der Datei ersetzt. Wenn die Werte für *startOffset* und *endOffset* beide die Anzahl der Zeichen in der Datei übersteigen, wird der Text am Ende der Datei eingefügt.

Argumente
startOffset, endOffset, string

- Das Argument *startOffset* ist der Offset, der den Anfang des zu ersetzenden Bereichs angibt.
- Das Argument *endOffset* ist der Offset, der das Ende des zu ersetzenden Bereichs angibt.
- Das Argument *string* ist der einzufügende String.

Rückgabewerte
Ein boolescher Wert: `true` bei einem erfolgreichen Vorgang, andernfalls `false`.

dom.source.scrollEndFile()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Führt in der Codeansicht einen Bildlauf zum Ende des Dokuments durch, ohne dabei die Einfügemarke zu verschieben.

Argumente
Keine.

Rückgabewerte
Keine.

dom.source.scrollLineDown()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Führt in der Codeansicht zeilenweise einen Bildlauf nach unten durch, ohne dabei die Einfügemarke zu verschieben.

Argumente
nTimes

- Das Argument *nTimes* gibt die Anzahl der Zeilen für den Bildlauf an. Wenn *nTimes* nicht angegeben wird, gilt der Standardwert 1.

Rückgabewerte
Keine.

dom.source.scrollLineUp()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Führt in der Codeansicht zeilenweise einen Bildlauf nach oben durch, ohne dabei die Einfügemarke zu verschieben.

Argumente
nTimes

- Das Argument *nTimes* gibt die Anzahl der Zeilen für den Bildlauf an. Wenn *nTimes* nicht angegeben wird, gilt der Standardwert 1.

Rückgabewerte
Keine.

dom.source.scrollPageDown()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Führt in der Codeansicht seitenweise einen Bildlauf nach unten durch, ohne dabei die Einfügemarke zu verschieben.

Argumente
nTimes

- Das Argument *nTimes* gibt die Anzahl der Seiten für den Bildlauf an. Wenn *nTimes* nicht angegeben wird, gilt der Standardwert 1.

Rückgabewerte
Keine.

dom.source.scrollPageUp()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Führt in der Codeansicht seitenweise einen Bildlauf nach oben durch, ohne dabei die Einfügemarke zu verschieben.

Argumente
nTimes

- Das Argument *nTimes* gibt die Anzahl der Seiten für den Bildlauf an. Wenn *nTimes* nicht angegeben wird, gilt der Standardwert 1.

Rückgabewerte
Keine.

dom.source.scrollTopFile()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Führt in der Codeansicht einen Bildlauf zum Anfang des Dokuments durch, ohne dabei die Einfügemarke zu verschieben.

Argumente
Keine.

Rückgabewerte
Keine.

dom.source.selectParentTag()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Bei dieser Funktion handelt es sich um eine Erweiterung der Codeansicht, die ausgeglichene Tags ermöglicht. Sie können die Funktion `dom.source.selectParentTag()` aufrufen, um die aktuelle Auswahl oder die aktuelle Einfügemarke ab dem umgebenden öffnenden Tag bis zum schließenden Tag zu erweitern. Bei anschließenden Aufrufen der Funktion wird die Auswahl bis zu weiteren umgebenden Tags erweitert, bis keine umschließenden Tags mehr vorhanden sind.

Argumente
Keine.

Rückgabewerte
Keine.

dom.source.setCurrentLine()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Platziert die Einfügemarke am Anfang der angegebenen Zeile. Wenn das Argument *lineNumber* keine positive Ganzzahl ist, gibt die Funktion den Wert `false` zurück. Die Funktion ist in diesem Fall wirkungslos. Wenn der Wert für *lineNumber* die Anzahl der Zeilen im Quelldokument übersteigt, wird die Einfügemarke am Anfang der letzten Zeile platziert.

Argumente
lineNumber

- Das Argument *lineNumber* ist die Zeile, an deren Anfang die Einfügemarke platziert wird.

Rückgabewerte
Ein boolescher Wert: `true` bei einem erfolgreichen Vorgang, andernfalls `false`.

dom.source.startOfDocument()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Platziert die Einfügemarke in der Codeansicht am Anfang des Dokuments. Wenn bereits ein Bereich ausgewählt wurde, wird die Auswahl durch diese Funktion bis zum Anfang des Dokuments erweitert.

Argumente
bShiftIsDown

- Das Argument *bShiftIsDown* ist ein boolescher Wert, der angibt, ob der Inhalt ausgewählt wurde. Wenn *bShiftIsDown* den Wert `true` hat, wird Inhalt ausgewählt.

Rückgabewerte
Keine.

dom.source.startOfLine()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Platziert die Einfügemarke am Anfang der aktuellen Zeile. Wenn bereits ein Bereich ausgewählt wurde, wird die Auswahl durch diese Funktion bis zum Anfang der aktuellen Zeile erweitert.

Argumente
bShiftIsDown

- Das Argument *bShiftIsDown* ist ein boolescher Wert, der angibt, ob der Inhalt ausgewählt wurde. Wenn *bShiftIsDown* den Wert `true` hat, wird Inhalt ausgewählt.

Rückgabewerte
Keine.

dom.source.topPage()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Verschiebt die Einfügemarke an den Anfang der aktuellen Seite oder an den Anfang der vorherigen Seite, wenn die Einfügemarke sich bereits am Anfang einer Seite befindet. Wenn bereits ein Bereich ausgewählt wurde, wird die Auswahl durch diese Funktion seitenweise erweitert.

Argumente
{nTimes}, {bShiftIsDown}

- Das optionale Argument *nTimes* gibt die Anzahl der Seiten an, um die die Einfügemarke verschoben werden soll. Wenn *nTimes* nicht angegeben wird, gilt der Standardwert 1.
- Das optionale Argument *bShiftIsDown* ist ein boolescher Wert, der angibt, ob der Inhalt ausgewählt wurde. Wenn *bShiftIsDown* den Wert `true` hat, wird Inhalt ausgewählt.

Rückgabewerte
Keine.

dom.source.wrapSelection()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Fügt den Text für *startTag* vor der aktuellen Auswahl und den Text für *endTag* nach der aktuellen Auswahl ein. Die Funktion wählt dann den gesamten Bereich zwischen den eingefügten Tags einschließlich der Tags aus. Wenn es sich bei der aktuellen Auswahl um eine Einfügemarke handelt, platziert die Funktion die Einfügemarke zwischen *startTag* und *endTag*. (*startTag* und *endTag* müssen nicht unbedingt Tags sein, es kann sich auch um Text handeln.)

Argumente
startTag, endTag

- Das Argument *startTag* ist der Text, der am Anfang der Auswahl eingefügt werden soll.
- Das Argument *endTag* ist der Text, der am Ende der Auswahl eingefügt werden soll.

Rückgabewerte
Keine.

dom.synchronizeDocument()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Synchronisiert die Entwurfs- und die Codeansicht.

Argumente
Keine.

Rückgabewerte
Keine.

Funktionen für die Live-Codeansicht

Der in der Live-Codeansicht angezeigte Code entspricht der Darstellung des Seitenquelltexts in einem Browser. Während Browser-Seitenquelltext statisch ist und nur den zum jeweiligen Zeitpunkt vom Browser zur Darstellung verwendeten Code enthält, ist die Live-Codeansicht dynamisch und wird bei Interaktionen mit der Seite in der Live-Ansicht kontinuierlich aktualisiert.

Wenn der Benutzer interaktive Elemente einer Seite aktiviert, wird die Quelle in der Live-Codeansicht mit dem neuen Zustand angezeigt. Zudem wird der Code hervorgehoben, der sich zwischen den verschiedenen Zuständen geändert hat.

dom.getLiveCodeHighlightsChanges()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Mit dieser Funktion wird bestimmt, ob die Funktion zum Hervorheben von Code für das aktuelle Dokument aktiviert ist.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert, der angibt, ob die Funktion zum Hervorheben von Code aktiviert ist.

dom.setLiveCodeHighlightsChanges()

Verfügbarkeit
Dreamweaver CS5.

Beschreibung
Mithilfe dieser Funktion wird die Hervorhebung von Code für das aktuelle Dokument aktiviert oder deaktiviert.

Argumente
Ein boolescher Wert, der angibt, ob die Funktion zum Hervorheben von Code aktiviert ist.

Rückgabewerte
Keine.

Funktionen für Tag-Editor und Tag-Bibliothek

Mit Tag-Editoren können Sie neue Tags einfügen, vorhandene Tags bearbeiten und auf Referenzinformationen zu Tags zugreifen. Mit der Tag-Auswahl können Benutzer ihre Tags anordnen, um häufig verwendete Tags schneller auswählen zu können. Die im Lieferumfang von Dreamweaver enthaltenen Tag-Bibliotheken speichern Informationen über Tags, die in standardisierten Markup-Sprachen und den gängigsten Tag-basierten Skriptsprachen verwendet werden. Mithilfe der Funktionen für JavaScript-Tag-Editor, Tag-Auswahl und Tag-Bibliothek können Sie auf Tag-Editoren und Tag-Bibliotheken in Erweiterungen zugreifen und diese verwenden.

dom.getTagSelectorTag()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Diese Funktion ruft den DOM-Knoten für das Tag ab, das derzeit im Tag-Selektor im unteren Bereich des Dokumentfensters ausgewählt ist.

Argumente
Keine.

Rückgabewerte
Der DOM-Knoten für das derzeit ausgewählte Tag oder `null`, wenn kein Tag ausgewählt ist.

dreamweaver.popupInsertTagDialog()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Diese Funktion überprüft die VTM-Dateien, um festzustellen, ob ein Tag-Editor für das Tag definiert wurde. Ist dies der Fall, wird der Editor für dieses Tag angezeigt und das öffnende Tag übernommen. Andernfalls wird das öffnende Tag unverändert in das Benutzerdokument eingefügt.

Argumente
start_tag_string

Ein String für das öffnende Tag, der einen der folgenden Anfangswerte enthält:

- Ein Tag, z. B. `<input>`
- Ein Tag mit Attributen, z. B. `<input type='text'>`
- Eine Direktive, z. B. `<%= %>`

Rückgabewerte
Ein boolescher Wert: `true`, wenn ein Element in das Dokument eingefügt wird, andernfalls `false`.

dreamweaver.popupEditTagDialog()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Wenn ein Tag ausgewählt ist, öffnet diese Funktion den entsprechenden Tag-Editor, sodass Sie das Tag bearbeiten können.

Argumente
Keine.

Rückgabewerte
Keine.

Enabler
Siehe „dreamweaver.canPopupEditTagDialog()“ auf Seite 521.

dreamweaver.showTagChooser()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Diese Funktion zeigt das Dialogfeld „Tag-Auswahl“ im Vordergrund an und aktiviert es.

Argumente
Keine.

Rückgabewerte
Keine.

dreamweaver.showTagLibraryEditor()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Diese Funktion öffnet den Tag-Bibliothek-Editor.

Argumente
Keine.

Rückgabewerte
Keine.

dreamweaver.tagLibrary.getTagLibraryDOM()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Mit der URL einer Datei des Typs *Dateiname.vtm* als Argument gibt diese Funktion das DOM für diese Datei zurück, sodass der Inhalt bearbeitet werden kann.Diese Funktion sollte nur aufgerufen werden, wenn der Tag-Bibliothek-Editor aktiv ist.

Argumente
fileURL

- Das Argument *fileURL* ist die URL einer Datei vom Typ *Dateiname*`.vtm` relativ zum Ordner „Configuration/TagLibraries“, z. B. "HTML/img.vtm".

Rückgabewerte
Ein DOM-Zeiger auf eine neue oder bereits vorhandene Datei im Ordner „TagLibraries“.

dreamweaver.tagLibrary.getSelectedLibrary()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Wenn im Tag-Bibliothek-Editor ein Bibliotheksknoten ausgewählt ist, gibt diese Funktion den Bibliotheksnamen zurück.

Argumente
Keine.

Rückgabewerte
Ein String mit dem Namen der Bibliothek, die im Tag-Bibliothek-Editor derzeit ausgewählt ist. Wenn keine Bibliothek ausgewählt ist, wird ein leerer String zurückgegeben.

dreamweaver.tagLibrary.getSelectedTag()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Wenn derzeit ein Attributknoten ausgewählt ist, gibt diese Funktion den Namen des Tags zurück, das das Attribut enthält.

Argumente
Keine.

Rückgabewerte
Ein String mit dem Namen des Tags, das im Tag-Bibliothek-Editor derzeit ausgewählt ist. Wenn kein Tag ausgewählt ist, wird ein leerer String zurückgegeben.

dreamweaver.tagLibrary.importDTDOrSchema()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Diese Funktion importiert eine DTD-Datei oder eine Schemadatei von einem Remote-Server in die Tag-Bibliothek.

Argumente
fileURL, Prefix

- Das Argument *fileURL* ist der Pfad zur DTD- oder Schemadatei im lokalen URL-Format.
- Das Argument *Prefix* ist der Präfix-String, der allen Tags in dieser Tag-Bibliothek hinzugefügt wird.

Rückgabewerte
Name der importierten Tag-Bibliothek.

dreamweaver.tagLibrary.getImportedTagList()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Diese Funktion generiert eine Liste mit `tagInfo`-Objekten aus einer importierten Tag-Bibliothek.

Argumente
libname

- Das Argument *libname* ist der Name der importierten Tag-Bibliothek.

Rückgabewerte
Array von `tagInfo`-Objekten.

Ein `tagInfo`-Objekt enthält Informationen über ein einzelnes Tag, das in der Tag-Bibliothek enthalten ist. In einem `tagInfo`-Objekt sind die folgenden Eigenschaften definiert:

- Bei der Eigenschaft `tagName` handelt es sich um einen String.
- Bei der Eigenschaft `attributes` handelt es sich um ein Array von Strings. Jeder String enthält den Namen eines Attributs, das für dieses Tag definiert ist.

Beispiel

Im folgenden Beispiel ist dargestellt, wie durch Verwendung der Funktion `dw.tagLibrary.getImportedTagList()` ein Tag-Array aus der Bibliothek `libName` abgerufen wird.

```
    // "fileURL" and "prefix" have been entered by the user.
    // tell the Tag Library to Import the DTD/Schema
var libName = dw.tagLibrary.importDTDOrSchema(fileURL, prefix);

    // get the array of tags for this library
    // this is the TagInfo object
var tagArray = dw.tagLibrary.getImportedTagList(libName);

    // now I have an array of tagInfo objects.
    // I can get info out of them. This gets info out of the first one.
    // note: this assumes there is at least one TagInfo in the array.
var firstTagName = tagArray[0].name;
var firstTagAttributes = tagArray[0].attributes;
    // note that firstTagAttributes is an array of attributes.
```

Kapitel 19: Enabler

Die Enabler-Funktionen von Adobe® Dreamweaver® CS5 legen fest, ob eine andere Funktion im aktuellen Kontext einen bestimmten Vorgang durchführen kann. In den Funktionsbeschreibungen werden die allgemeinen Bedingungen erläutert, unter denen die jeweilige Funktion den Wert `true` zurückgibt. Die Beschreibungen sind jedoch nicht als vollständig anzusehen und bestimmte Fälle, in denen die betreffende Funktion den Wert `false` zurückgibt, sind eventuell nicht erfasst.

Enabler-Funktionen

Die Enabler-Funktionen in der JavaScript-API umfassen die folgenden Funktionen.

dom.canAlign()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob die Vorgänge „Linksbündig“, „Rechtsbündig“, „Oben ausrichten“ bzw. „Unten ausrichten“ ausgeführt werden können.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert, der angibt, ob zwei oder mehr Ebenen oder Hotspots ausgewählt sind.

dom.canApplyTemplate()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob der Vorgang „Auf Seite anwenden“ ausgeführt werden kann. Diese Funktion ist nur für das aktive Dokument gültig.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert, der angibt, ob es sich beim Dokument nicht um ein Bibliothekselement oder eine Vorlage handelt und ob die Auswahl sich nicht innerhalb des `NOFRAMES`-Tags befindet.

dom.canArrange()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob der Vorgang „In Vordergrund stellen“ bzw. „In Hintergrund stellen“ ausgeführt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert, der angibt, ob ein Hotspot ausgewählt ist.

dom.canClipCopyText()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob der Vorgang „Als Text kopieren“ ausgeführt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn sich die öffnenden und schließenden Offsets der Auswahl unterscheiden, andernfalls `false`, um anzugeben, dass nichts ausgewählt wurde.

dom.canClipPaste()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob der Vorgang „Einfügen“ ausgeführt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn die Zwischenablage Inhalt enthält, der in Dreamweaver eingefügt werden kann, andernfalls `false`.

dom.canClipPasteText()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob der Vorgang „Als Text einfügen“ ausgeführt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn die Zwischenablage Inhalt enthält, der in Dreamweaver als Text eingefügt werden kann, andernfalls `false`.

dom.canConvertLayersToTable()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob der Vorgang „Ebenen in Tabelle konvertieren“ ausgeführt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn sich der gesamte Inhalt im Abschnitt `BODY` des Dokuments in Ebenen befindet, andernfalls `false`.

dom.canConvertTablesToLayers()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob der Vorgang „Tabellen in Ebenen konvertieren“ ausgeführt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn sich der gesamte Inhalt des Abschnitts `BODY` des Dokuments in Tabellen befindet und das Dokument nicht auf einer Vorlage basiert, andernfalls `false`.

dom.canDecreaseColspan()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob der Vorgang „Spaltenraum verkleinern“ ausgeführt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn die aktuelle Zelle über das Attribut `COLSPAN` verfügt und der Wert dieses Attributs größer oder gleich 2 ist, andernfalls `false`.

dom.canDecreaseRowspan()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob der Vorgang „Zeilenraum verkleinern“ ausgeführt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn die aktuelle Zelle über das Attribut `ROWSPAN` verfügt und der Wert dieses Attributs größer oder gleich 2 ist, andernfalls `false`.

dom.canDeleteTableColumn()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob der Vorgang „Spalte löschen“ ausgeführt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn sich die Einfügemarke in einer Zelle befindet oder wenn eine Zelle bzw. Spalte ausgewählt ist, andernfalls `false`.

dom.canDeleteTableRow()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob der Vorgang „Zeile löschen“ ausgeführt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn sich die Einfügemarke in einer Zelle befindet oder wenn eine Zelle bzw. Zeile ausgewählt ist, andernfalls `false`.

site.canEditColumns()

Beschreibung
Prüft, ob eine Site vorhanden ist.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn eine Site vorhanden ist, andernfalls `false`.

dom.canEditNoFramesContent()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob der Vorgang „NoFrames-Inhalt bearbeiten“ ausgeführt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn das aktuelle Dokument ein Frameset ist oder sich in einem Frameset befindet, andernfalls `false`.

dom.canIncreaseColspan()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob der Vorgang „Spaltenraum vergrößern“ ausgeführt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn sich Zellen rechts neben der aktuellen Zelle befinden, andernfalls `false`.

dom.canIncreaseRowspan()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob der Vorgang „Zeilenraum vergrößern“ ausgeführt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn sich Zellen unterhalb der aktuellen Zelle befinden, andernfalls `false`.

dom.canInsertTableColumns()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob der Vorgang „Spalte(n) einfügen“ ausgeführt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn sich die Auswahl innerhalb einer Tabelle befindet, `false`, wenn die Auswahl keine vollständige Tabelle ist oder sich nicht in einer Tabelle befindet.

dom.canInsertTableRows()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob der Vorgang „Zeile(n) einfügen“ ausgeführt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn sich die Auswahl innerhalb einer Tabelle befindet, `false`, wenn die Auswahl keine vollständige Tabelle ist oder sich nicht in einer Tabelle befindet.

dom.canMakeNewEditableRegion()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob der Vorgang „Neuer bearbeitbarer Bereich“ ausgeführt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn es sich beim aktuellen Dokument um eine Vorlage handelt (DWT-Datei).

dom.canMarkSelectionAsEditable()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob der Vorgang „Auswahl als bearbeitbar markieren“ ausgeführt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn eine Auswahl vorhanden und das aktuelle Dokument eine DWT-Datei ist, andernfalls `false`.

dom.canMergeTableCells()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob der Vorgang „Zellen verbinden“ ausgeführt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn es sich bei der Auswahl um eine Gruppe benachbarter Zellen handelt, andernfalls `false`.

dom.canPlayPlugin()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob der Vorgang „Wiedergeben“ ausgeführt werden kann. Diese Funktion ist nur für das aktive Dokument gültig.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn die Auswahl mit einem Plug-In wiedergegeben werden kann.

dom.canRedo()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob der Vorgang „Wiederherstellen“ ausgeführt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn es noch wiederherstellbare Schritte gibt, andernfalls `false`.

dom.canRemoveEditableRegion()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob der Vorgang „Markierung als bearbeitbaren Bereich aufheben“ ausgeführt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn es sich beim aktuellen Dokument um eine Vorlage handelt, andernfalls `false`.

dom.canSelectTable()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob der Vorgang „Tabelle auswählen“ ausgeführt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn sich die Einfügemarke oder Auswahl innerhalb einer Tabelle befindet, andernfalls `false`.

dom.canSetLinkHref()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob der Hyperlink um die aktuelle Auswahl geändert oder gegebenenfalls ein Hyperlink erstellt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true,` wenn es sich bei der Auswahl um ein Bild oder Text handelt oder wenn sich die Einfügemarke in einem Hyperlink befindet, andernfalls `false`. Eine Textauswahl ist eine Auswahl, bei der der Eigenschafteninspektor für Text angezeigt wird.

dom.canShowListPropertiesDialog()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Bestimmt, ob das Dialogfeld „Listeneigenschaften“ angezeigt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn sich die Auswahl innerhalb eines `LI`-Tags befindet, andernfalls `false`.

dom.canSplitFrame()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob der Vorgang „Frame [links | rechts | oben | unten] teilen“ ausgeführt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn sich die Auswahl in einem Frame befindet, andernfalls `false`.

dom.canSplitTableCell()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob der Vorgang „Zelle teilen“ ausgeführt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn sich die Einfügemarke in einer Tabellenzelle befindet oder wenn die Auswahl eine Tabellenzelle ist, andernfalls `false`.

dom.canStopPlugin()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob der Vorgang „Stoppen“ ausgeführt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn die Auswahl derzeit mit einem Plug-In wiedergegeben wird, andernfalls `false`.

dom.canUndo()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob der Vorgang „Rückgängig" ausgeführt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn es noch rückgängig zu machende Schritte gibt, andernfalls `false`.

dom.hasTracingImage()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob das Dokument ein Tracing-Bild enthält.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn das Dokument ein Tracing-Bild enthält, andernfalls `false`.

dreamweaver.assetPalette.canEdit()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Aktiviert Menüeinträge im Bedienfeld „Elemente" zur Bearbeitung.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn das Element bearbeitet werden kann, andernfalls `false`. Für Farben und URLs in der Siteliste wird `false` zurückgegeben. Für mehrere ausgewählte Farben und URLs in der Favoritenliste wird ebenfalls `false` zurückgegeben.

dreamweaver.assetPalette.canInsertOrApply()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Überprüft, ob die ausgewählten Elemente eingefügt oder angewendet werden können. Gibt `true` oder `false` zurück, sodass die Menüeinträge zum Einfügen oder Anwenden aktiviert oder deaktiviert werden können.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn die ausgewählten Elemente eingefügt oder angewendet werden können, `false`, wenn es sich bei der aktuellen Seite um eine Vorlage handelt und die aktuelle Kategorie „Vorlagen“ lautet. Die Funktion gibt auch dann den Wert `false` zurück, wenn kein Dokument geöffnet ist oder wenn ein Bibliothekselement im Dokument ausgewählt ist und die aktuelle Kategorie „Bibliothek“ lautet.

dreamweaver.canClipCopy()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob der Vorgang „Kopieren“ ausgeführt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn Inhalt ausgewählt ist, der in die Zwischenablage kopiert werden kann, andernfalls `false`.

dreamweaver.canClipCut()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob der Vorgang „Ausschneiden“ ausgeführt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn Inhalt ausgewählt ist, der ausgeschnitten und in die Zwischenablage kopiert werden kann, andernfalls `false`.

dreamweaver.canClipPaste()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob der Vorgang „Einfügen“ ausgeführt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn die Zwischenablage Inhalt enthält, der in das aktuelle Dokument oder in das aktive Fenster des Bedienfelds „Dateien“ (auf dem Macintosh ein Textfeld in einem schwebenden Bedienfeld oder Dialogfeld) eingefügt werden kann, andernfalls `false`.

dreamweaver.canDeleteSelection()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob die aktuelle Auswahl gelöscht werden kann. Je nachdem, auf welchem Fenster sich der Fokus befindet, wird der Löschvorgang im Dokumentfenster oder im Bedienfeld „Dateien“ bzw. auf dem Macintosh in einem Textfeld eines Dialogfelds oder eines schwebenden Bedienfelds durchgeführt.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn sich die öffnenden und schließenden Offsets für die Auswahl unterscheiden (d. h. eine Auswahl ist vorhanden), `false`, wenn die Offsets identisch sind, was darauf hinweist, dass nur eine Einfügemarke vorhanden ist.

dreamweaver.canExportTemplateDataAsXML()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Überprüft, ob das aktuelle Dokument als XML exportiert werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn für das aktuelle Dokument ein Exportvorgang ausgeführt werden kann, andernfalls `false`.

Beispiel
Im folgenden Beispiel wird `dw.canExportTemplateDataAsXML()` aufgerufen, um festzustellen, ob Dreamweaver das aktuelle Dokument im XML-Format exportieren kann; bei der Rückgabe von `true` wird `dw.ExportTemplateDataAsXML()` zum Exportieren aufgerufen:

```
if(dreamweaver.canExportTemplateDataAsXML())
{
    dreamweaver.exportTemplateDataAsXML("file:///c|/dw_temps/mytemplate.txt")
}
```

dreamweaver.canFindNext()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob der Vorgang „Weitersuchen“ ausgeführt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn bereits ein Suchmuster festgelegt wurde, andernfalls `false`.

dreamweaver.canFitSelection()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Überprüft, ob in einer aktiven Entwurfsansicht eine Auswahl vorhanden ist, was bedeutet, dass `fitSelection()` aufgerufen werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn in einer aktiven Entwurfsansicht eine Auswahl vorhanden ist, andernfalls `false`.

dreamweaver.canOpenInFrame()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob der Vorgang „Öffnen in Frame“ ausgeführt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn sich die Auswahl oder Einfügemarke in einem Frame befindet, andernfalls `false`.

dreamweaver.canPasteSpecial()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Überprüft, ob der Vorgang „Inhalte einfügen“ ausgeführt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn sich in der Zwischenablage Text, HTML-Code oder Dreamweaver-HTML-Code befindet und die Codeansicht, die Entwurfsansicht oder der Codeinspektor den Fokus hat, andernfalls `false`.

dreamweaver.canPlayRecordedCommand()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob der Vorgang „Aufgezeichneten Befehl abspielen“ ausgeführt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn ein aktives Dokument und ein zuvor aufgezeichneter Befehl, der wiedergegeben werden kann, vorhanden sind, andernfalls `false`.

dreamweaver.canPopupEditTagDialog()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Überprüft, ob es sich bei der aktuellen Auswahl um ein Tag handelt und ob das Menüelement „Tag bearbeiten“ aktiv ist.

Argumente
Keine.

Rückgabewerte
Der Name des gerade ausgewählten Tags oder `null`, wenn kein Tag ausgewählt ist.

dreamweaver.canRedo()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob der Vorgang „Wiederherstellen“ im aktuellen Kontext ausgeführt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert, der angibt, ob Vorgänge rückgängig gemacht werden können.

dreamweaver.canRevertDocument()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob der Vorgang „Wiederherstellen“ (der zuletzt gespeicherten Version) ausgeführt werden kann.

Argumente
documentObject

- Das Argument *documentObject* ist das Objekt am Stamm der DOM-Struktur eines Dokuments (der von `dreamweaver.getDocumentDOM()` zurückgegebene Wert).

Rückgabewerte
Ein boolescher Wert, der angibt, ob das Dokument ungespeichert ist und ob eine gespeicherte Version des Dokuments auf einem lokalen Laufwerk vorhanden ist.

dreamweaver.canSaveAll()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob der Vorgang „Alles speichern“ ausgeführt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert, der angibt, ob ungespeicherte Dokumente geöffnet sind.

dreamweaver.canSaveDocument()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob beim angegebenen Dokument der Vorgang „Speichern“ ausgeführt werden kann.

Argumente
documentObject

- Das Argument *documentObject* ist der Stamm der DOM-Struktur eines Dokuments (der von `dreamweaver.getDocumentDOM()` zurückgegebene Wert).

Rückgabewerte
Ein boolescher Wert, der angibt, ob das Dokument ungespeicherte Änderungen enthält.

dreamweaver.canSaveDocumentAsTemplate()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob beim angegebenen Dokument der Vorgang „Als Vorlage speichern“ ausgeführt werden kann.

Argumente
documentObject

- Das Argument *documentObject* ist der Stamm der DOM-Struktur eines Dokuments (der von `dreamweaver.getDocumentDOM()` zurückgegebene Wert).

Rückgabewerte
Ein boolescher Wert, der angibt, ob das Dokument als Vorlage gespeichert werden kann.

dreamweaver.canSaveFrameset()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob beim angegebenen Dokument der Vorgang „Frameset speichern“ ausgeführt werden kann.

Argumente
documentObject

- Das Argument *documentObject* ist der Stamm der DOM-Struktur eines Dokuments (der von `dreamweaver.getDocumentDOM()` zurückgegebene Wert).

Rückgabewerte
Ein boolescher Wert, der angibt, ob es sich bei dem Dokument um ein Frameset mit ungespeicherten Änderungen handelt.

dreamweaver.canSaveFramesetAs()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob beim angegebenen Dokument der Vorgang „Frameset speichern unter“ ausgeführt werden kann.

Argumente
documentObject

- Das Argument *documentObject* ist der Stamm der DOM-Struktur eines Dokuments (der von `dreamweaver.getDocumentDOM()` zurückgegebene Wert).

Rückgabewerte
Ein boolescher Wert, der angibt, ob es sich beim aktuellen Dokument um ein Frameset handelt.

dreamweaver.canSelectAll()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob der Vorgang „Alles auswählen“ ausgeführt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert, der angibt, ob der Vorgang „Alles auswählen“ ausgeführt werden kann.

dreamweaver.canShowFindDialog()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob der Vorgang „Suchen“ ausgeführt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn das Bedienfeld „Dateien“ oder ein Dokumentfenster geöffnet ist. Diese Funktion gibt den Wert `false` zurück, wenn sich die Auswahl im Bereich `HEAD` befindet.

dreamweaver.canUndo()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob der Vorgang „Rückgängig“ im aktuellen Kontext ausgeführt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert, der angibt, ob Vorgänge rückgängig gemacht werden können.

dreamweaver.canZoom()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Prüft, ob eine Entwurfs- oder Live-Ansicht aktiv ist, was bedeutet, dass grundlegende Vergrößerungsbefehle angewendet werden können.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn eine aktive Entwurfsansicht vorhanden ist, andernfalls `false`.

dreamweaver.cssRuleTracker.canEditSelectedRule()

Verfügbarkeit
Dreamweaver MX 2004.

Beschreibung
Überprüft, ob der Eigenschaftenraster-Editor auf die ausgewählte Regel angewendet werden kann. Da das Eigenschaftenraster Regeln in gesperrten Dateien anzeigen kann, gewährleistet der Rückgabewert `true` nicht, dass die Regel geändert werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der Eigenschaftenraster-Editor auf die ausgewählte Regel angewendet werden kann, andernfalls `false`.

Beispiel
Der folgende Code überprüft, ob die Enabler-Funktion auf den Wert `true` gesetzt wurde, bevor die Bearbeitung der ausgewählten Regel zugelassen wird:

```
if(dw.cssRuleTracker.canEditSelectedRule()){
    dw.cssRuleTracker.editSelectedRule();
}
```

dreamweaver.cssStylePalette.canApplySelectedStyle()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Überprüft das aktuelle Dokument, um zu ermitteln, ob der ausgewählte Stil angewendet werden kann.

Argumente
{pane}

- Das optionale Argument *pane* ist ein String, der den Bereich des Bedienfelds „CSS-Stile“ angibt, auf den diese Funktion angewendet wird. Mögliche Werte sind: `"stylelist"`, die Liste der Stile im Modus „Alle“, `"cascade"`, die Liste anwendbarer, relevanter Regeln im Modus „Aktuell“, `"summary"`, die Liste der Eigenschaften für die aktuelle Auswahl im Modus „Aktuell“ und `"ruleInspector"`, die bearbeitbare Liste bzw. das Raster mit den Eigenschaften im Modus „Aktuell“. Der Standardwert ist `"stylelist"`.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der ausgewählte Stil einen Class-Selektor hat, andernfalls `false`.

dreamweaver.cssStylePalette.canDeleteSelectedStyle()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Überprüft die aktuelle Auswahl, um zu ermitteln, ob der ausgewählte Stil gelöscht werden kann.

Argumente
{pane}

- Das optionale Argument *pane* ist ein String, der den Bereich des Bedienfelds „CSS-Stile“ angibt, auf den diese Funktion angewendet wird. Mögliche Werte sind: `"stylelist"`, die Liste der Stile im Modus „Alle“, `"cascade"`, die Liste anwendbarer, relevanter Regeln im Modus „Aktuell“, `"summary"`, die Liste der Eigenschaften für die aktuelle Auswahl im Modus „Aktuell“ und `"ruleInspector"`, die bearbeitbare Liste bzw. das Raster mit den Eigenschaften im Modus „Aktuell“. Der Standardwert ist `"stylelist"`.

Rückgabewerte
Ein boolescher Wert: `true`, wenn die Auswahl gelöscht werden kann, andernfalls `false`.

dreamweaver.cssStylePalette.canDuplicateSelectedStyle()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Überprüft das aktuelle aktive Dokument, um zu ermitteln, ob der ausgewählte Stil dupliziert werden kann.

Argumente
{pane}

- Das optionale Argument *pane* ist ein String, der den Bereich des Bedienfelds „CSS-Stile“ angibt, auf den diese Funktion angewendet wird. Mögliche Werte sind: `"stylelist"`, die Liste der Stile im Modus „Alle“, `"cascade"`, die Liste anwendbarer, relevanter Regeln im Modus „Aktuell“, `"summary"`, die Liste der Eigenschaften für die aktuelle Auswahl im Modus „Aktuell“ und `"ruleInspector"`, die bearbeitbare Liste bzw. das Raster mit den Eigenschaften im Modus „Aktuell“. Der Standardwert ist `"stylelist"`.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der ausgewählte Stil dupliziert werden kann, andernfalls `false`.

dreamweaver.cssStylePalette.canEditSelectedStyle()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Überprüft das aktuelle Dokument, um zu ermitteln, ob der ausgewählte Stil bearbeitet werden kann.

Argumente
{pane}

- Das optionale Argument *pane* ist ein String, der den Bereich des Bedienfelds „CSS-Stile“ angibt, auf den diese Funktion angewendet wird. Mögliche Werte sind: `"stylelist"`, die Liste der Stile im Modus „Alle“, `"cascade"`, die Liste anwendbarer, relevanter Regeln im Modus „Aktuell“, `"summary"`, die Liste der Eigenschaften für die aktuelle Auswahl im Modus „Aktuell“ und `"ruleInspector"`, die bearbeitbare Liste bzw. das Raster mit den Eigenschaften im Modus „Aktuell“. Der Standardwert ist `"stylelist"`.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der ausgewählte Stil bearbeitet werden kann, andernfalls `false`.

dreamweaver.cssStylePalette.canEditSelectedStyleInCodeview()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Überprüft das aktuelle Dokument, um zu ermitteln, ob der ausgewählte Stil in der Codeansicht bearbeitet werden kann.

Argumente
{pane}

- Das optionale Argument *pane* ist ein String, der den Bereich des Bedienfelds „CSS-Stile“ angibt, auf den diese Funktion angewendet wird. Mögliche Werte sind: `"stylelist"`, die Liste der Stile im Modus „Alle“, `"cascade"`, die Liste anwendbarer, relevanter Regeln im Modus „Aktuell“, `"summary"`, die Liste der Eigenschaften für die aktuelle Auswahl im Modus „Aktuell“ und `"ruleInspector"`, die bearbeitbare Liste bzw. das Raster mit den Eigenschaften im Modus „Aktuell“. Der Standardwert ist `"stylelist"`.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der ausgewählte Stil bearbeitet werden kann, andernfalls `false`.

dreamweaver.cssStylePalette.canEditStyleSheet()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Überprüft das aktuelle Dokument, um zu ermitteln, ob es Stylesheet-Elemente enthält, die bearbeitet werden können.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true` wenn es sich bei der Auswahl um einen Stylesheet-Knoten oder eine Stildefinition innerhalb eines Stylesheet-Knotens handelt und das Stylesheet weder versteckt noch dieses Dokument ist, `false`, wenn die Auswahl versteckt ist oder es sich um dieses Dokument handelt.

dreamweaver.cssStylePalette.canRenameSelectedStyle()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Überprüft das aktuelle Dokument, um zu ermitteln, ob der ausgewählte Stil umbenannt werden kann.

Argumente
{pane}

- Das optionale Argument *pane* ist ein String, der den Bereich des Bedienfelds „CSS-Stile“ angibt, auf den diese Funktion angewendet wird. Mögliche Werte sind: `"stylelist"`, die Liste der Stile im Modus „Alle“, `"cascade"`, die Liste anwendbarer, relevanter Regeln im Modus „Aktuell“, `"summary"`, die Liste der Eigenschaften für die aktuelle Auswahl im Modus „Aktuell“ und `"ruleInspector"`, die bearbeitbare Liste bzw. das Raster mit den Eigenschaften im Modus „Aktuell“. Der Standardwert ist `"stylelist"`.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der ausgewählte Stil dupliziert werden kann, andernfalls `false`.

dreamweaver.isRecording()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Bestimmt, ob gerade ein Befehl aufgezeichnet wird.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert, der angibt, ob ein Befehl aufgezeichnet wird.

dreamweaver.htmlStylePalette.canEditSelection()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob die Auswahl im Bedienfeld „HTML-Stile“ bearbeitet, gelöscht oder dupliziert werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn Dreamweaver die Auswahl im Bedienfeld „HTML-Stile“ bearbeiten, löschen oder duplizieren kann; `false`, wenn kein Stil oder einer der Löschen-Stile ausgewählt ist.

dreamweaver.resultsPalette.canClear()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Überprüft, ob der Inhalt, der sich im aktiven Fenster des Bedienfelds „Ergebnisse“ befindet, gelöscht werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der Inhalt gelöscht werden kann, andernfalls `false`.

dreamweaver.resultsPalette.canCopy()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Überprüft, ob das aktuelle Ergebnisfenster eine kopierte Meldung als Inhalt anzeigen kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der Inhalt angezeigt werden kann, andernfalls `false`.

dreamweaver.resultsPalette.canCut()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Überprüft, ob das aktuelle Ergebnisfenster eine Ausgeschnitten-Meldung als Inhalt anzeigen kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der Inhalt angezeigt werden kann, andernfalls `false`.

dreamweaver.resultsPalette.canPaste()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Überprüft, ob das aktuelle Ergebnisfenster eine Eingefügt-Meldung als Inhalt anzeigen kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der Inhalt angezeigt werden kann, andernfalls `false`.

dreamweaver.resultsPalette.canOpenInBrowser()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Überprüft, ob der aktuelle Bericht in einem Browser angezeigt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der Inhalt angezeigt werden kann, andernfalls `false`.

dreamweaver.resultsPalette.canOpenInEditor()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Überprüft, ob der aktuelle Bericht in einem Editor angezeigt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der Inhalt angezeigt werden kann, andernfalls `false`.

dreamweaver.resultsPalette.canSave()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Überprüft, ob für das aktuelle Bedienfeld das Dialogfeld „Speichern“ aufgerufen werden kann. Derzeit unterstützen die Bedienfelder „Site-Berichte“, „Browserkompatibilität“, „Überprüfung“ und „Hyperlink-Prüfer“ das Dialogfeld „Speichern“.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn das Dialogfeld „Speichern“ angezeigt werden kann, andernfalls `false`.

dreamweaver.resultsPalette.canSelectAll()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Überprüft, ob eine Nachricht „Alles auswählen“ an das derzeit aktive Fenster gesendet werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn die Nachricht „Alles auswählen“ gesendet werden kann, andernfalls `false`.

dreamweaver.siteSyncDialog.canCompare()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion prüft, ob im Dialogfeld „Synchronisieren“ der Kontextmenüeintrag „Lokale und Remote-Dateien vergleichen“ angezeigt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn im Dialogfeld „Synchronisieren“ der Kontextmenüeintrag „Lokale und Remote-Dateien vergleichen“ angezeigt werden kann, andernfalls `false`.

dreamweaver.siteSyncDialog.canMarkDelete()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion prüft, ob im Dialogfeld „Synchronisieren“ der Kontextmenüeintrag „Aktion in Löschen ändern“ angezeigt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der Kontextmenüeintrag „Aktion in Löschen ändern“ angezeigt werden kann, andernfalls `false`.

dreamweaver.siteSyncDialog.canMarkGet()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion prüft, ob im Dialogfeld „Synchronisieren“ der Kontextmenüeintrag „Aktion in Abrufen ändern“ angezeigt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der Kontextmenüeintrag „Aktion in Abrufen ändern“ angezeigt werden kann, andernfalls `false`.

dreamweaver.siteSyncDialog.canMarkIgnore()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion prüft, ob im Dialogfeld „Synchronisieren“ der Kontextmenüeintrag „Auswahl ignorieren“ angezeigt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der Kontextmenüeintrag „Auswahl ignorieren“ angezeigt werden kann, andernfalls `false`.

dreamweaver.siteSyncDialog.canMarkPut()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion prüft, ob im Dialogfeld „Synchronisieren“ der Kontextmenüeintrag „Aktion in Bereitstellen ändern“ angezeigt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der Kontextmenüeintrag „Aktion in Bereitstellen ändern“ angezeigt werden kann, andernfalls `false`.

dreamweaver.siteSyncDialog.canMarkSynced()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion prüft, ob im Dialogfeld „Synchronisieren“ der Kontextmenüeintrag „Als synchronisiert markieren“ angezeigt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn der Kontextmenüeintrag „Als synchronisiert markieren“ angezeigt werden kann, andernfalls `false`.

dreamweaver.snippetpalette.canEditSnippet()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Überprüft, ob Sie das gerade ausgewählte Element bearbeiten können, und gibt `true` bzw. `false` zurück, sodass Sie Menüelemente zum Bearbeiten aktivieren oder deaktivieren können.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn Sie das gerade ausgewählte Element bearbeiten können, andernfalls `false`.

dreamweaver.snippetpalette.canInsert()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Überprüft, ob Sie das ausgewählte Element einfügen oder anwenden können, und gibt `true` bzw. `false` zurück, sodass Sie Menüelemente zum Einfügen oder Anwenden aktivieren oder deaktivieren können.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert, `true`, wenn Sie das ausgewählte Element einfügen oder anwenden können, andernfalls `false`.

site.browseDocument()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Öffnet alle ausgewählten Dokumente in einem Browserfenster. Dies ist mit dem Befehl „Vorschau in Browser“ identisch.

Argumente
browserName

- Das Argument *browserName* ist der Name des Browsers, wie im Dialogfeld „Voreinstellungen“ in der Kategorie „Vorschau in Browser“ definiert. Wenn dieses Argument ausgelassen wird, wird standardmäßig der Primärbrowser des Benutzers verwendet.

Rückgabewerte
Keine.

site.canCheckIn()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob der Vorgang „Einchecken“ ausgeführt werden kann.

Argumente
siteOrURL

- Das Argument *siteOrURL* muss das Schlüsselwort `site` sein, damit die Funktion sich auf die Auswahl im Bedienfeld „Dateien“ auswirkt, oder eine URL, wenn es sich um eine einzelne Datei handelt.

Rückgabewerte
Ein boolescher Wert: `true`, wenn die folgenden Bedingungen erfüllt sind, andernfalls `false`:

- Eine Remote-Site ist definiert.
- Falls sich der Fokus auf einem Dokumentfenster befindet: Die Datei wurde in einer lokalen Site gespeichert. Falls sich der Fokus auf dem Bedienfeld „Dateien“ befindet: Mindestens eine Datei bzw. ein Ordner ist ausgewählt.
- Die Funktion „Einchecken/Auschecken“ ist für die Site aktiviert.

site.canCheckOut()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob für die angegebenen Dateien der Vorgang „Auschecken“ ausgeführt werden kann.

Argumente
siteOrURL

- Das Argument *siteOrURL* muss das Schlüsselwort `site` sein, damit die Funktion sich auf die Auswahl im Bedienfeld „Dateien“ auswirkt, oder eine URL, wenn es sich um eine einzelne Datei handelt.

Rückgabewerte
Ein boolescher Wert: `true`, wenn alle folgenden Bedingungen erfüllt sind, andernfalls `false`:

- Eine Remote-Site ist definiert.
- Falls sich der Fokus auf einem Dokumentfenster befindet: Die Datei gehört zu einer lokalen Site und ist noch nicht ausgecheckt. Falls sich der Fokus auf dem Bedienfeld „Dateien“ befindet: Eine oder mehrere Dateien bzw. Ordner sind ausgewählt und mindestens eine der ausgewählten Dateien ist noch nicht ausgecheckt.
- Die Funktion „Einchecken/Auschecken“ ist für die Site aktiviert.

site.canCloak()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Überprüft, ob Dreamweaver einen Cloaking-Vorgang ausführen kann.

Argumente
siteOrURL

- Das Argument *siteOrURL* muss das Schlüsselwort `site` sein, wenn sich `canCloak()` auf die Auswahl im Bedienfeld „Dateien“ auswirken soll, oder die URL eines bestimmten Ordners, wenn sich `canCloak()` auf den angegebenen Ordner und dessen Inhalt auswirken soll.

Rückgabewerte
Ein boolescher Wert: `true`, wenn Dreamweaver den Cloaking-Vorgang für die aktuelle Site oder den angegebenen Ordner ausführen kann, andernfalls `false`.

site.canCompareFiles()

Verfügbarkeit
Dreamweaver 8.

Beschreibung
Diese Funktion prüft, ob Dreamweaver für die ausgewählten Dateien die Vergleichsfunktion durchführen kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn zwei Dateien (eine lokale und eine Remote-Datei, zwei lokale Dateien oder zwei Remote-Dateien) ausgewählt sind, andernfalls `false`.

site.canConnect()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob eine Verbindung zur Remote-Site hergestellt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn die aktuelle Remote-Site eine FTP-Site ist, andernfalls `false`.

site.canDisplaySyncInfoForFile()

Verfügbarkeit
Dreamweaver CS3.

Beschreibung
Überprüft, ob der Vorgang „Synchronisierungsdaten anzeigen“ ausgeführt werden kann.

Argumente
path, '`site`'

- *path* ist die URL zu einer lokalen Datei.
- '`site`' gibt an, dass die Funktion die im Bedienfeld „Dateien“ ausgewählte Datei verwendet.

Rückgabewerte
Gibt `true` zurück, wenn in der lokalen Dateiansicht eine Datei ausgewählt ist (falls 'site' der Parameter ist), oder wenn der übergebene Pfad Teil einer Site ist. Andernfalls wird `false` zurückgegeben.

site.canGet()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob der Vorgang „Abrufen“ ausgeführt werden kann.

Argumente
siteOrURL

- Das Argument *siteOrURL* muss das Schlüsselwort `site` sein, damit die Funktion sich auf die Auswahl im Bedienfeld „Dateien“ auswirkt, oder eine URL, wenn es sich um eine einzelne Datei handelt.

Rückgabewerte
Wenn das Argument `site` lautet, wird ein boolescher Wert zurückgegeben, der angibt, ob eine oder mehrere Dateien bzw. Ordner im Bedienfeld „Dateien“ ausgewählt sind und ob eine Remote-Site definiert ist. Wenn das Argument eine URL ist, wird ein boolescher Wert zurückgegeben, der angibt, ob das Dokument zu einer Site gehört, für die eine Remote-Site definiert ist.

site.canLocateInSite()

Verfügbarkeit
Dreamweaver 3, aktualisiert in CS4.

Beschreibung
Bestimmt, ob der Vorgang „Auf lokaler Site lokalisieren“ bzw. „Auf entfernter Site lokalisieren“ (je nach Argument) ausgeführt werden kann.

Argumente
`localOrRemote`, *`siteOrURL`*

- Das Argument *`localOrRemote`* muss entweder `local` oder `remote` sein.
- Das Argument *`siteOrURL`* muss das Schlüsselwort `site` sein, damit die Funktion sich auf die Auswahl im Bedienfeld „Dateien“ auswirkt, oder eine URL, wenn es sich um eine einzelne Datei handelt.

Rückgabewerte
Einer der folgenden Werte:

- Ein boolescher Wert, der angibt, ob das Dokument zu einer Site gehört. Der boolesche Wert wird zurückgegeben, wenn das erste Argument das Schlüsselwort `local` und das zweite Argument eine URL ist.
- Ein boolescher Wert. Der boolesche Wert wird zurückgegeben, wenn das erste Argument das Schlüsselwort `remote` und das zweite Argument eine URL ist. Der boolesche Wert gibt an:
  - Ob das Dokument zu einer Site gehört, für die eine Remote-Site definiert ist.
  - Ob die Festplatte bereitgestellt ist, wenn der Servertyp „Lokal/Netzwerk“ lautet.
- Ein boolescher Wert, der angibt, ob beide Fenster Site-Dateien enthalten und ob sich die Auswahl in dem zum Argument entgegengesetzten Bedienfeld befindet. Der boolesche Wert wird zurückgegeben, wenn das zweite Argument das Schlüsselwort `site.` ist.

site.canMakeEditable()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob der Vorgang „Schreibschutz deaktivieren“ ausgeführt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn Dreamweaver die Operation „Schreibschutz deaktivieren“ durchführen kann; `false`, wenn eine oder mehrere der ausgewählten Dateien gesperrt sind.

site.canMakeNewFileOrFolder()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob im Bedienfeld „Dateien“ der Vorgang „Neue Datei“ bzw. „Neuer Ordner“ ausgeführt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn im ausgewählten Bereich des Bedienfelds „Dateien“ Dateien sichtbar sind, andernfalls `false`.

site.canOpen()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob die im Bedienfeld „Dateien“ gerade ausgewählten Dateien bzw. Ordner geöffnet werden können.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn im Bedienfeld „Dateien“ Dateien oder Ordner ausgewählt sind, andernfalls `false`.

site.canPut()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob der Vorgang „Bereitstellen“ ausgeführt werden kann.

Argumente
siteOrURL

- Das Argument *siteOrURL* muss das Schlüsselwort `site` sein, damit die Funktion sich auf die Auswahl im Bedienfeld „Dateien“ auswirkt, oder eine URL, wenn es sich um eine einzelne Datei handelt.

Rückgabewerte
Einer der folgenden Werte:

- Wenn das Argument das Schlüsselwort `site` ist, wird der Wert `true` zurückgegeben, sofern Dateien bzw. Ordner im Bedienfeld „Dateien“ ausgewählt sind und eine Remote-Site definiert wurde, andernfalls `false`.
- Wenn als Argument eine URL übergeben wird, wird der Wert `true` zurückgegeben, sofern das Dokument zu einer Site gehört, für die eine Remote-Site definiert ist, andernfalls wird `false` zurückgegeben.

site.canRecreateCache()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob der Vorgang „Sitecache neu erstellen“ ausgeführt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn die Option „Mit Cache Hyperlink-Updates beschleunigen“ für die aktuelle Site aktiviert ist.

site.canRefresh()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob der Vorgang „Aktualisieren [Lokal | Remote]“ ausgeführt werden kann.

Argumente
localOrRemote

- Das Argument *localOrRemote* muss entweder das Schlüsselwort `local` oder `remote` sein .

Rückgabewerte
`true`, wenn *localOrRemote* auf `local` gesetzt ist. Andernfalls ein boolescher Wert, der angibt, ob eine Remote-Site definiert ist.

site.canSelectAllCheckedOutFiles()

Verfügbarkeit
Dreamweaver 4.

Beschreibung
Bestimmt, ob die Option „Einchecken/Auschecken“ für die aktuelle Site aktiviert ist.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn das Ein- und Auschecken für die Site zulässig ist, andernfalls `false`.

site.canSelectNewer()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob der Vorgang „Auswählen: Neuere (lokal)“ bzw. „Auswählen: Neuere (Remote)“ ausgeführt werden kann.

Argumente
localOrRemote

- Das Argument *localOrRemote* muss entweder das Schlüsselwort `local` oder `remote` sein .

Rückgabewerte
Ein boolescher Wert, der angibt, ob das Dokument zu einer Site gehört, für die eine Remote-Site definiert ist.

site.canSynchronize()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob der Vorgang „Synchronisieren“ ausgeführt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert, der angibt, ob eine Remote-Site definiert ist.

site.canUncloak()

Verfügbarkeit
Dreamweaver MX.

Beschreibung
Überprüft, ob der Vorgang „Cloaking deaktivieren“ ausgeführt werden kann.

Argumente
siteOrURL

- Das Argument *siteOrURL* muss das Schlüsselwort `site` sein, wenn sich `canUncloak()` auf die Auswahl im Bedienfeld „Dateien“ auswirken soll, oder die URL eines bestimmten Ordners, wenn sich `canUncloak()` auf den angegebenen Ordner und dessen Inhalt auswirken soll.

Rückgabewerte
Ein boolescher Wert: `true`, wenn Dreamweaver den Vorgang „Cloaking deaktivieren“ für die aktuelle Site oder den angegebenen Ordner ausführen kann, andernfalls `false`.

site.canUndoCheckOut()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob der Vorgang „Auschecken rückgängig“ ausgeführt werden kann.

Argumente
siteOrURL

- Das Argument *siteOrURL* muss das Schlüsselwort `site` sein, damit die Funktion sich auf die Auswahl im Bedienfeld „Dateien“ auswirkt, oder eine URL, wenn es sich um eine einzelne Datei handelt.

Rückgabewerte
Ein boolescher Wert: `true`, wenn die angegebene Datei oder mindestens eine der ausgewählten Dateien ausgecheckt wurde.

site.canViewAsRoot()

Verfügbarkeit
Dreamweaver 3.

Beschreibung
Überprüft, ob der Vorgang „Als Stammordner anzeigen“ ausgeführt werden kann.

Argumente
Keine.

Rückgabewerte
Ein boolescher Wert: `true`, wenn es sich bei der angegebenen Datei um eine HTML- oder Flash-Datei handelt, andernfalls `false`.