

ADOBE® FLASH® LITE™ 1.x

Adobe® ActionScript® リファレンスガイド

© 2008 Adobe Systems Incorporated. All rights reserved.

Adobe® Flash® Lite™ 1.x ActionScript™ リファレンスガイド

マニュアルがエンドユーザー使用許諾契約を含むソフトウェアと共に提供される場合、マニュアルおよびその中に記載されているソフトウェアは、エンドユーザー使用許諾契約にもとづいて提供されるものであり、当該エンドユーザー使用許諾契約の契約条件に従ってのみ使用または複製することが可能となるものです。当該エンドユーザー使用許諾契約により許可されている場合を除き、マニュアルのいかなる部分といえども、Adobe Systems Incorporated (アドビ システムズ社) の書面による事前の許可なしに、電子的、機械的、録音、その他いかなる形式・手段であれ、複製、検索システムへの保存、または伝送を行うことはできません。マニュアルの内容は、エンドユーザー使用許諾契約を含むソフトウェアと共に提供されていない場合であっても、著作権法により保護されていることにご留意ください。

マニュアルに記載される内容は、あくまでも参照用としてのみ使用されること、また、なんら予告なしに変更されることを条件として、提供されるものであり、従って、当該情報が、アドビ システムズ社による確約として解釈されてはなりません。アドビ システムズ社は、マニュアルにおけるいかなる誤りまたは不正確な記述に対しても、いかなる義務や責任を負うものではありません。

マニュアルには、アドビ システムズ社が管理していない、サードパーティの Web サイトへのリンクが掲載されていますが、アドビ システムズ社はいかなるリンク先サイトの内容についても責任を持ちません。マニュアルに記載されているサードパーティの Web サイトには、自己責任においてアクセスしてください。アドビ システムズ社はこれらのリンクを便宜上の目的においてのみ掲載しています。リンクを掲載することにより、アドビ システムズ社がこれらのサードパーティのサイトの内容について何らかの責任を持つことを示すものではありません。

新しいアートワークを創作するためにテンプレートとして取り込もうとする既存のアートワークまたは画像は、著作権法により保護されている可能性のあるものであることにご留意ください。保護されているアートワークまたは画像を新しいアートワークに許可なく取り込んだ場合、著作権者の権利を侵害することがあります。従って、著作権者から必要なすべての許可を必ず取得してください。

例として使用されている会社名は、実在の会社・組織を示すものではありません。

Adobe, the Adobe logo, ColdFusion, and Flash are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Symbian and all Symbian based marks and logos are trademarks of Symbian Limited. All other trademarks are the property of their respective owners.

Sorenson Spark™ video compression and decompression technology licensed from Sorenson Media, Inc.

MPEG Layer-3 audio compression technology licensed by Fraunhofer IIS and Thomson Multimedia (<http://www.iis.fhg.de/amm/>).

Portions licensed from Nellymoser, Inc. (www.nellymoser.com).

Adobe Flash 9.2 video is powered by On2 TrueMotion video technology. © 1992-2005 On2 Technologies, Inc. All Rights Reserved. <http://www.on2.com>.

Updated Information/Additional Third Party Code Information available at <http://www.adobe.com/go/thirdparty/>.

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

Notice to U.S. Government End Users. The Software and Documentation are "Commercial Items," as that term is defined at 48 C.F.R. §2.101, consisting of "Commercial Computer Software" and "Commercial Computer Software Documentation," as such terms are used in 48 C.F.R. §12.212 or 48 C.F.R. §227.7202, as applicable. Consistent with 48 C.F.R. §12.212 or 48 C.F.R. §§227.7202-1 through 227.7202-4, as applicable, the Commercial Computer Software and Commercial Computer Software Documentation are being licensed to U.S. Government end users (a) only as Commercial Items and (b) with only those rights as are granted to all other end users pursuant to the terms and conditions herein. Unpublished-rights reserved under the copyright laws of the United States. Adobe Systems Incorporated, 345 Park Avenue, San Jose, CA 95110-2704, USA. For U.S. Government End Users, Adobe agrees to comply with all applicable equal opportunity laws including, if appropriate, the provisions of Executive Order 11246, as amended, Section 402 of the Vietnam Era Veterans Readjustment Assistance Act of 1974 (38 USC 4212), and Section 503 of the Rehabilitation Act of 1973, as amended, and the regulations at 41 CFR Parts 60-1 through 60-60, 60-250, and 60-741. The affirmative action clause and regulations contained in the preceding sentence shall be incorporated by reference.

目次

第1章：はじめに

サンプルフォルダ	1
表記規則	1

第2章：Flash Lite のグローバル関数

call()	3
chr()	4
duplicateMovieClip()	4
eval ()	5
getProperty()	6
getTimer()	7
getURL()	7
gotoAndPlay()	9
gotoAndStop()	10
ifFrameLoaded()	10
int()	11
length()	12
loadMovie()	12
loadMovieNum()	13
loadVariables()	14
loadVariablesNum()	15
mbchr()	16
mblength()	16
mbord()	17
mbsubstring()	17
nextFrame()	18
nextScene()	18
Number()	19
on()	19
ord()	20
play()	21
prevFrame()	21
prevScene()	22
random()	22
removeMovieClip()	23
set()	23
setProperty()	24
stop()	25

stopAllSounds()	25
String()	26
substring()	26
tellTarget()	27
toggleHighQuality()	27
trace()	28
unloadMovie()	28
unloadMovieNum()	29

第3章：Flash Lite のプロパティ

/ (スラッシュ)	31
_alpha	31
_currentframe	32
_focusrect	32
_framesloaded	33
_height	33
_highquality	34
_level	34
maxscroll	35
_name	35
_rotation	36
scroll	36
_target	37
_totalframes	37
_visible	37
_width	38
_x	38
_xscale	39
_y	39
_yscale	40

第4章：Flash Lite のステートメント

break	41
case	42
continue	43
do..while	44
else	45
else if	46
for	46
if	47
switch	48
while	49

第 5 章：Flash Lite の演算子

add (ストリング連結)	53
+=(加算後代入)	53
および	54
=(代入)	54
/* (コメントのブロック)	55
,(コンマ)	55
// (コメント)	56
?:(条件)	57
-- (デクリメント)	57
/ (除算)	58
/=(除算後代入)	59
.(ドット)	59
++ (インクリメント)	60
&& (論理積 (AND))	61
! (論理否定 (NOT))	61
(論理和 (OR))	62
% (剰余)	63
%= (剰余代入)	63
*= (乗算後代入)	64
* (乗算)	64
+ (数値式加算)	65
==(数値の等価)	65
> (数値が対象より大きい)	66
>= (数値が対象より大きいか等しい)	67
<> (数値の不等価)	67
< (数値が対象より小さい)	68
<= (数値が対象より小さいか等しい)	68
() (括弧)	69
" (ストリング区切り記号)	70
eq (ストリングの等価)	70
gt (ストリングが対象より大きい)	71
ge (ストリングが対象より大きいか等しい)	71
ne (ストリングの不等価)	72
lt (ストリングが対象より小さい)	73
le (ストリングが対象より小さいか等しい)	73
- (減算)	74
-= (減算後代入)	75

第 6 章：Flash Lite 固有の言語エレメント

機能	78
_capCompoundSound	78

_capEmail	78
_capLoadData	79
_capMFi	79
_capMIDI	80
_capMMS	80
_capMP3	81
_capSMAF	81
_capSMS	82
_capStreamSound	82
_cap4WayKeyAS	83
\$version	84
fscommand()	84
Launch	84
fscommand2()	85
Escape	86
FullScreen	86
GetBatteryLevel	87
GetDateDay	87
GetDateMonth	88
GetDateWeekday	88
GetDateYear	89
GetDevice	90
GetDeviceID	91
GetFreePlayerMemory	92
GetLanguage	92
GetLocaleLongDate	95
GetLocaleShortDate	96
GetLocaleTime	96
GetMaxBatteryLevel	97
GetMaxSignalLevel	97
GetMaxVolumeLevel	97
GetNetworkConnectStatus	98
GetNetworkName	99
GetNetworkRequestStatus	99
GetNetworkStatus	101
GetPlatform	102
GetPowerSource	103
GetSignalLevel	103
GetTimeHours	104
GetTimeMinutes	104
GetTimeSeconds	105
GetTimeZoneOffset	105

GetTotalPlayerMemory	106
GetVolumeLevel	106
Quit	107
ResetSoftKeys	107
SetInputTextType	108
SetQuality	108
SetSoftKeys	109
StartVibrate	109
StopVibrate	110
Unescape	110
索引	112

第 1 章：はじめに

本マニュアルでは、アドビ システムズ社の Macromedia® Flash® Lite™ 1.0 および Macromedia® Flash® Lite™ 1.1 ソフトウェア（まとめて Flash Lite 1.x と呼びます）向けのアプリケーションを開発する際に使用する ActionScript エLEMENTのシンタックスと使用方法について説明します。Flash Lite 1.x ActionScript は、アドビ システムズ社の Macromedia® Flash® 4 で使用されたバージョンの ActionScript に基づいています。スクリプト内の例を使用するには、本マニュアルからコード例をコピーし、スクリプトペインまたは外部のスクリプトファイルにペーストしてください。本マニュアルには、すべての ActionScript エLEMENT（演算子、キーワード、ステートメント、コマンド、プロパティ、関数、クラス、およびメソッド）が記載されています。

サンプルフォルダ

実用的な ActionScript コードを含む完全な Flash Lite プロジェクトのサンプルについては、Flash Lite のサンプルとチュートリアル ページ (www.adobe.com/go/learn_flt_samples_and_tutorials) を参照してください。ActionScript バージョンの .zip ファイルを探してダウンロードおよび解凍し、Samples フォルダの中のサンプルファイルにアクセスします。

表記規則

本書では、次の表記規則を使用しています。

- **Italic font** (イタリック体のフォント) は、置き換える必要がある値 (フォルダパス内の値など) を示します。
- **Code font** (コードフォント) は ActionScript コードを示します。
- **Code font italic** (イタリック体のコードフォント) は ActionScript パラメータを示します。
- **Bold font** (ボールドフォント) は逐語的な入力を示します。
- サンプルコードの二重引用符 (") は区切られた文字列を示します。ただし、プログラマは一重引用符も使用できます。

第 2 章：Flash Lite のグローバル関数

このセクションでは、アドビ システムズ社の Macromedia Flash Lite 1.1 ソフトウェアの ActionScript グローバル関数のシンタックスおよび使用方法について説明します。ここで説明する関数は以下のとおりです。

関数	説明
<code>call()</code>	呼び出されたフレームで、そのフレームに再生ヘッドを移動せずに、スクリプトを実行します。
<code>chr()</code>	ASCII コード番号を文字に変換します。
<code>duplicateMovieClip()</code>	SWF ファイルの再生中にムービークリップのインスタンスを作成します。
<code>eval()</code>	変数、プロパティ、オブジェクト、ムービークリップに名前でアクセスします。
<code>getProperty()</code>	指定されたムービークリップの指定プロパティの値を返します。
<code>getTimer()</code>	SWF ファイルを再生し始めてからの経過時間をミリ秒単位で返します。
<code>getUrl()</code>	特定の URL からウィンドウにドキュメントをロードしたり、定義済みの URL に存在する別のアプリケーションに変数を渡したりします。
<code>gotoAndPlay()</code>	シーン内の指定されたフレームに再生ヘッドを送り、そのフレームから再生を開始します。シーンを指定しないと、再生ヘッドは現在のシーン内の指定されたフレームに移動します。
<code>gotoAndStop()</code>	シーン内の指定されたフレームに再生ヘッドを送り、停止します。シーンを指定しないと、再生ヘッドは現在のシーン内のフレームに送られます。
<code>ifframeLoaded()</code>	特定のフレームの内容がローカルに使用できるかどうかを確認します。
<code>int()</code>	小数値の小数部分を切り捨てて整数値にします。
<code>length()</code>	指定された文字列または変数の名前の文字数を返します。
<code>loadMovie()</code>	元の SWF ファイルの再生中に SWF ファイルを Flash Lite 内にロードします。
<code>loadMovieNum()</code>	ロードした元の SWF ファイルの再生中に SWF ファイルを Flash Lite のレベル内にロードします。
<code>loadVariables()</code>	テキストファイルや、Adobe ColdFusion®、CGI、ASP、PHP、または Perl スクリプトで作成されたテキストなどの外部ファイルからデータを読み取り、Flash Lite のいずれかのレベルの変数に値を設定します。この関数により、現在の SWF ファイルの変数を新しい値で更新することもできます。
<code>loadVariablesNum()</code>	テキストファイルや、ColdFusion、CGI、ASP、PHP または Perl スクリプトで作成されたテキストなどの外部ファイルからデータを読み取り、Flash Lite の特定のレベルの変数に値を設定します。この関数により、現在の SWF ファイルの変数を新しい値で更新することもできます。
<code>mbchr()</code>	ASCII コード番号をマルチバイト文字に変換します。
<code>mblength()</code>	マルチバイト文字の文字列長を返します。
<code>mbord()</code>	指定された文字をマルチバイト文字コード番号に変換します。
<code>mbsubstring()</code>	マルチバイト文字列から、新たにマルチバイト文字列を抽出します。
<code>nextFrame()</code>	次のフレームに再生ヘッドを送り、停止します。
<code>nextScene()</code>	次のシーンのフレーム 1 に再生ヘッドを送り、停止します。
<code>Number()</code>	式を数値に変換して、値を返します。
<code>on()</code>	イベントをトリガするユーザーイベントまたはキー押下を指定します。

関数	説明
<code>ord()</code>	文字を ASCII コード番号に変換します。
<code>play()</code>	タイムライン内で再生ヘッドを前へ進めます。
<code>prevFrame()</code>	直前のフレームに再生ヘッドを送り、停止します。現在のフレームが 1 である場合、再生ヘッドは移動しません。
<code>prevScene()</code>	前のシーンのフレーム 1 に再生ヘッドを送り、停止します。
22 ページの「 <code>random()</code> 」	ランダムな整数を返します。
<code>removeMovieClip()</code>	<code>duplicateMovieClip()</code> を使用して作成した元の指定ムービークリップを削除します。
<code>set()</code>	変数に値を代入します。
<code>setProperty()</code>	ムービーの再生時にムービークリップのプロパティ値を変更します。
<code>stop()</code>	再生中の SWF ファイルを停止します。
<code>stopAllSounds()</code>	再生ヘッドを停止せずに、SWF ファイルで再生中のサウンドをすべて停止します。
<code>String()</code>	指定されたパラメータの文字列表現を返します。
<code>substring()</code>	文字列の一部を抽出します。
<code>tellTarget()</code>	<i>statement(s)</i> パラメータで指定された指示を、 <i>target</i> パラメータで指定されたタイムラインに適用します。
<code>toggleHighQuality()</code>	Flash Lite でアンチエイリアス処理のオンとオフを切り替えます。アンチエイリアス処理を行うとオブジェクトのエッジが滑らかになりますが、SWF ファイルの再生は遅くなります。
<code>trace()</code>	式を評価し、その結果をプレビューモードで出力パネルに表示します。
<code>unloadMovie()</code>	<code>loadMovie()</code> 、 <code>loadMovieNum()</code> または <code>duplicateMovieClip()</code> を使用してロードしたムービークリップを Flash Lite から削除します。
<code>unloadMovieNum()</code>	<code>loadMovie()</code> 、 <code>loadMovieNum()</code> または <code>duplicateMovieClip()</code> を使用してロードしたムービークリップを Flash Lite の特定のレベルから削除します。

call()

利用状況

Flash Lite 1.0

シンタックス

```
call (frame)
```

```
call (movieClipInstance: frame)
```

オペランド

frame タイムラインのフレームのラベルまたは番号。

movieClipInstance ムービークリップのインスタンス名。

説明

関数。呼び出されたフレームで、そのフレームに再生ヘッドを移動せずに、スクリプトを実行します。スクリプトの実行後、ローカル変数は存在しません。call() 関数は、次のいずれかの形式で指定できます。

- デフォルト形式は、call() 関数が実行されたのと同じタイムラインの指定フレームに再生ヘッドを移動せずに、そのフレームのスクリプトを実行します。
- 指定されたクリップインスタンス形式は、ムービークリップインスタンスの指定フレームに再生ヘッドを移動せずに、そのフレームのスクリプトを実行します。

注意：call() 関数は、同期的に実行されます。つまり、call() 関数の後の ActionScript は、指定されたフレームの ActionScript がすべて完了するまで実行されません。

例

次の例では、myScript フレームのスクリプトが実行されます。

```
// to execute functions in frame with label "myScript"
thisFrame = "myScript";
trace ("Calling the script in frame: " + thisFrame);

// to execute functions in any other frame on the same timeline
call("myScript");
```

chr()

利用状況

Flash Lite 1.0

シンタックス

```
chr(number)
```

オペランド

number ASCII コード番号。

説明

ストリング関数。ASCII コード番号を文字に変換します。

例

次の例では、数値 65 が文字 **A** に変換され、変換後の文字が変数 myVar に代入されます。

```
myVar = chr(65);
trace (myVar); // Output: A
```

duplicateMovieClip()

利用状況

Flash Lite 1.0

シンタックス

```
duplicateMovieClip(target, newname, depth)
```

オペランド

target 複製するムービークリップのターゲットパス。

newname 複製したムービークリップの一意の識別子。

depth 複製したムービークリップ固有の深度。深度は、複製したムービークリップの重ね順を示します。この重ね順は、タイムラインの重ね順とよく似ています。深度の低いムービークリップは、深度の高いクリップの下に隠されます。既に占有されている深度の既存のムービークリップが上書きされないように、複製したムービークリップには、それぞれ固有の深度を割り当てる必要があります。

説明

関数。SWF ファイルの再生中にムービークリップのインスタンスを作成します。何も返しません。複製ムービークリップの再生ヘッドは、元の（親）ムービークリップでの再生ヘッドの位置に関係なく、常にフレーム 1 から始まります。親のムービークリップ内の変数は、複製されたムービークリップにコピーされません。親のムービークリップが削除されると、複製されたムービークリップも削除されます。`duplicateMovieClip()` で作成されたムービークリップインスタンスを削除するには、`removeMovieClip()` 関数またはメソッドを使用します。**newname** オペランドとして渡されたストリングを使用して、新しいムービークリップを参照します。

例

次の例では、`originalClip` という名前のムービークリップが複製され、`newClip` という名前の新しいクリップが深度 10 に作成されます。新しいクリップの `x` 位置は、100 ピクセルに設定されます。

```
duplicateMovieClip("originalClip", "newClip", 10);  
setProperty("newClip", _x, 100);
```

次の例では、`for` ループで `duplicateMovieClip()` を使用して、複数の新しいムービークリップが一度に作成されます。インデックス変数は、既に占有されている深度のうち最上位の深度を追跡します。重複したムービークリップの名前には、それぞれの深度に対応する数値の接尾辞が含まれています (`clip1`、`clip2`、`clip3`)。

```
for (i = 1; i <= 3; i++) {  
    newName = "clip" + i;  
    duplicateMovieClip("originalClip", newName); }  
}
```

関連項目

[removeMovieClip\(\)](#)

eval ()

利用状況

Flash Lite 1.0

シンタックス

```
eval (expression)
```

オペランド

expression 取得する変数、プロパティ、オブジェクトまたはムービークリップの名前が含まれたストリング。

説明

関数。変数、プロパティ、オブジェクト、ムービークリップに名前でアクセスします。**expression** が変数またはプロパティである場合は、変数またはプロパティの値が返されます。**expression** がオブジェクトまたはムービークリップである場合は、オブジェクトまたはムービークリップへの参照が返されます。**expression** に指定したエレメントが見つからない場合は、`undefined` が返されます。

`eval()` を使用すると、配列をシミュレートしたり、変数の値を動的に設定および取得したりすることができます。

例

次の例では、`eval()` を使用して、式 `"piece" + x` の値が決定されます。この式の結果は変数名 `piece3` です。そのため、`eval()` は、この変数の値を返して、`y` に代入します。

```
piece3 = "dangerous";
x = 3;
y = eval("piece" + x);
trace(y); // Output: dangerous.
```

次の例では、配列をどのようにシミュレートできるかを示します。

```
name1 = "mike";
name2 = "debbie";
name3 = "logan";
for(i = 1; i <= 3; i++) {
    trace (eval("name" + i)); // Output: mike, debbie, logan
}
```

getProperty()

利用状況

Flash Lite 1.0

シンタックス

```
getProperty(my_mc, property)
```

オペランド

`my_mc` プロパティを取得するムービークリップのインスタンス名。

`property` ムービークリップのプロパティ。

説明

関数。ムービークリップ **my_mc** の指定プロパティの値を返します。

例

次の例では、ルートムービータイムラインにあるムービークリップ `my_mc` の水平軸座標 (`_x`) が取得されます。

```
xPos = getProperty("person_mc", _x);
trace (xPos); // output: -75
```

関連項目

[setProperty\(\)](#)

getTimer()

利用状況

Flash Lite 1.0

シンタックス

```
getTimer()
```

オペランド

なし

説明

関数。SWF ファイルを再生し始めてからの経過時間をミリ秒単位で返します。

例

次の例では、SWF ファイルを再生し始めてからの経過時間（ミリ秒単位）に `timeElapsed` 変数が設定されます。

```
timeElapsed = getTimer();  
trace (timeElapsed); // Output: milliseconds of time movie has been playing
```

getURL()

利用状況

Flash Lite 1.0

シンタックス

```
getURL(url [ , window [, "variables"]])
```

オペランド

`url` ドキュメントを取得するための URL。

`window` ドキュメントのロード先のウィンドウまたは HTML フレームを指定するオプションのパラメータ。

注意: 携帯電話のブラウザは複数のウィンドウをサポートしていないので、**window** パラメータは Flash Lite アプリケーションに対して指定されません。

空の文字列または特定のウィンドウの名前を入力するか、次の予約されたターゲット名から選択できます。

- `_self` は、現在のウィンドウ内の現在のフレームを指定します。
- `_blank` は、新規ウィンドウを指定します。
- `_parent` は、現在のフレームの親を指定します。
- `_top` は、現在のウィンドウ内の最上位のフレームを指定します。

`variables` 変数を送信するための GET メソッドまたは POST メソッド。変数がない場合は、このパラメータを省略します。GET メソッドは、変数を URL の最後に追加します。このメソッドは、変数のデータ量が少ないときに使用します。POST メソッドは、別の HTTP ヘッダで変数を送信します。このメソッドは、長い文字列の変数に使用します。

説明

関数。特定の URL からウィンドウにドキュメントをロードしたり、定義済みの URL に存在する別のアプリケーションに変数を渡したりします。この関数をテストするには、ロードするファイルが指定した場所にあることを確認します。絶対 URL (<http://www.myserver.com> など) を使用するには、ネットワーク接続が確立されている必要があります。

Flash Lite 1.0 は、HTTP、HTTPS、mailto および tel プロトコルしか認識しません。Flash Lite 1.1 は、これらのプロトコルだけでなく、ファイルのプロトコル、SMS (Short Message Service) プロトコルと MMS (Multimedia Message Service) プロトコルも認識します。

Flash Lite は、この関数の呼び出しをオペレーティングシステムに渡します。オペレーティングシステムは、指定プロトコル用に登録されたデフォルトアプリケーションを使用して、呼び出しを処理します。

フレームまたはイベントハンドラごとに処理される `getURL()` 関数は 1 つだけです。

一部のハンドセットでは `getURL()` 関数がキーイベントに限定されており、キーイベントハンドラ内でトリガされた `getURL()` 呼び出しだけが処理されます。そのような場合でも、イベントハンドラごとに処理される `getURL()` 関数は 1 つだけです。

例

次の ActionScript では、Flash Lite プレーヤーによって `mobile.example.com` がデフォルトブラウザで開かれます。

```
myURL = "http://mobile.example.com";
on(keyPress "1") {
    getURL(myURL);
}
```

GET や POST を使用して、現在のタイムラインから変数を送信することもできます。次の例では、GET メソッドを使用して、変数が URL に追加されます。

```
firstName = "Gus";
lastName = "Richardson";
age = 92;
getURL("http://www.example.com", "_blank", "GET");
```

次の ActionScript では、POST を使用して、HTTP ヘッダ内で変数が送信されます。

```
firstName = "Gus";
lastName = "Richardson";
age = 92;
getURL("http://www.example.com", "POST");
```

`address`、`subject` および `body` テキストフィールドの値が既に入力された電子メール作成ウィンドウが開くように、ボタン関数を割り当てることができます。ボタン関数を割り当てるには、次のいずれかのメソッドを使用します。メソッド 1 は Shift-JIS または英語文字エンコード用、メソッド 2 は英語文字エンコード用です。

メソッド 1：目的のパラメータのそれぞれの変数を設定します。以下はその例です。

```
on (release, keyPress "#"){
    subject = "email subject";
    body = "email body";
    getURL("mailto:somebody@anywhere.com", "", "GET");
}
```

メソッド 2：`getURL()` 関数内の各パラメータを定義します。以下はその例です。

```
on (release, keyPress "#"){
    getURL("mailto:somebody@anywhere.com?cc=cc@anywhere.com&bcc=bcc@anywhere.com&subject=I am the email subject&body=I am the email body");
}
```

メソッド 1 を使用すると、URL が自動的にエンコードされます。一方、メソッド 2 では、ストリング内のスペースが保持されます。例えば、メソッド 1 を使用した結果、次のストリングが生成されます。

```
email+subject  
email+body
```

それに対し、メソッド 2 を使用すると、次のストリングが生成されます。

```
email subject  
email body
```

次の例では、tel プロトコルが使用されます。

```
on (release, keyPress "#"){  
    getURL("tel:117");  
}
```

次の例では、getURL() を使用して、SMS メッセージが送信されます。

```
mySMS = "sms:4156095555?body=sample sms message";  
on(keyPress "5") {  
    getURL(mySMS);  
}
```

次の例では、getURL() を使用して、MMS メッセージが送信されます。

```
// mms example  
myMMS = "mms:4156095555?body=sample mms message";  
on(keyPress "6") {  
    getURL(myMMS);  
}
```

次の例では、getURL() を使用して、ローカルファイルシステム上に格納されているテキストファイルが開かれます。

```
// file protocol example  
filePath = "file:///c:/documents/flash/myApp/myvariables.txt";  
on(keyPress "7") {  
    getURL(filePath);  
}
```

gotoAndPlay()

利用状況

Flash Lite 1.0

シンタックス

```
gotoAndPlay([scene,] frame)
```

オペランド

scene 再生ヘッドの送り先となるシーンの名前を示すオプションのストリング。

frame 再生ヘッドの送り先となるフレーム番号を表す数値、または再生ヘッドの送り先となるフレームのラベルを表すストリング。

説明

関数。シーン内の指定されたフレームに再生ヘッドを送り、そのフレームから再生を開始します。シーンを指定しないと、再生ヘッドは現在のシーン内の指定されたフレームに移動します。

scene パラメータはルートタイムラインでのみ使用でき、ムービークリップやドキュメント内のその他のオブジェクトのタイムラインでは使用できません。

例

次の例では、gotoAndPlay() が割り当てられているボタンをユーザーがクリックすると、再生ヘッドが現在のシーン内のフレーム 16 に移動し、SWF ファイルの再生を開始します。

```
on(keyPress "7") {  
    gotoAndPlay(16);  
}
```

gotoAndStop()

利用状況

Flash 1.0

シンタックス

```
gotoAndStop([scene,] frame)
```

オペランド

scene 再生ヘッドの送り先となるシーンの名前を示すオプションのストリング。

frame 再生ヘッドの送り先となるフレーム番号を表す数値、または再生ヘッドの送り先となるフレームのラベルを表すストリング。

説明

関数。シーン内の指定されたフレームに再生ヘッドを送り、停止します。シーンを指定しないと、再生ヘッドは現在のシーン内のフレームに送られます。

scene パラメータはルートタイムラインでのみ使用でき、ムービークリップやドキュメント内のその他のオブジェクトのタイムラインでは使用できません。

例

次の例では、gotoAndStop() が割り当てられているボタンをユーザーがクリックすると、再生ヘッドが現在のシーン内のフレーム 5 に送られ、SWF ファイルの再生が停止します。

```
on(keyPress "8") {  
    gotoAndStop(5);  
}
```

ifFrameLoaded()

利用状況

Flash Lite 1.0

シンタックス

```
ifFrameLoaded([scene,] frame) {  
    statement(s);  
}
```

オペランド

scene ロードするシーンの名前を示すオプションのストリング。

frame 次のステートメントを実行する前にロードしておく必要のあるフレーム番号またはフレームラベル。

statement(s) 指定されたフレーム、またはシーンおよびフレームのロードに応じて実行される指示。

説明

関数。特定のフレームの内容がローカルに使用できるかどうかを確認します。SWF ファイル全体がローカルコンピュータにダウンロードされるのを待つ間に単純なアニメーションの再生を開始する場合は、`ifFrameLoaded` 関数を使用します。`_framesloaded` プロパティを使用して、外部 SWF ファイルのダウンロードの進捗状況をチェックすることもできます。`_framesloaded` が `ifFrameLoaded` と異なる点は、`_framesloaded` では、独自の `if` ステートメントや `else` ステートメントを追加できることです。

例

次の例では、`ifFrameLoaded` 関数を使用して、SWF ファイルのフレーム 10 がロードされているかどうかをチェックされます。フレームがロードされている場合、`trace()` コマンドは、「フレーム番号 10 はロードされています」という内容のメッセージが出力パネルに出力されます。出力変数も `frame loaded: 10` の変数と共に定義されます。

```
ifFrameLoaded(10) {  
    trace ("frame number 10 is loaded");  
    output = "frame loaded: 10";  
}
```

関連項目

[_framesloaded](#)

int()

利用状況

Flash Lite 1.0

シンタックス

```
int (value)
```

オペランド

value 切り捨てによって整数に変換された数値またはストリング。

説明

関数。小数値の小数部分を切り捨てて整数値にします。

例

次の例では、`distance` 変数と `myDistance` 変数の数値が切り捨てられます。

```
distance = 6.04 - 3.96;  
//trace ("distance = " add distance add " and rounded to:" add int(distance));  
// Output: distance = 2.08 and rounded to: 2  
myDistance = "3.8";  
//trace ("myDistance = " add int(myDistance));  
// Output: 3
```

length()

利用状況

Flash Lite 1.0

シンタックス

```
length(expression)
```

```
length(variable)
```

オペランド

expression ストリング。

variable 変数の名前。

説明

ストリング関数。指定されたストリングまたは変数の名前の文字数を返します。

例

次の例では、ストリング「Hello」の長さが返されます。

```
length("Hello");
```

結果は 5 です。

次の例では、電子メールアドレスの長さが 6 文字以上であることを確認することによって、アドレスの検証が行われます。

```
email = "someone@example.com";  
if (length(email) > 6) {  
    //trace ("email appears to have enough characters to be valid");  
}
```

loadMovie()

利用状況

Flash Lite 1.1

シンタックス

```
loadMovie(url, target [, method])
```

オペランド

url ロードする SWF ファイルの絶対 URL または相対 URL を指定するストリング。相対パスは、レベル 0 の SWF ファイルに相対的である必要があります。絶対 URL の場合は `http://` や `file:///` などのプロトコル参照を含める必要があります。

target ムービークリップへの参照またはターゲットムービークリップへのパスを表すストリング。ターゲットムービークリップは、ロードした SWF ファイルに置き換えられます。

method 変数を送信するための HTTP メソッドを指定するオプションのストリングパラメータ。パラメータはストリング GET または POST である必要があります。送る変数がない場合は、このパラメータを省略します。GET メソッドは、変数を URL の最後に追加します。このメソッドは、変数のデータ量が少ないときに使用します。POST メソッドは、別の HTTP ヘッダで変数を送信します。このメソッドは、長いストリングの変数に使用します。

説明

関数。元の SWF ファイルの再生中に SWF ファイルを Flash Lite 内にロードします。

SWF ファイルを特定のレベルにロードするには、loadMovie() 関数ではなく、loadMovieNum() 関数を使用します。

SWF ファイルをターゲットムービークリップにロードすると、そのムービークリップのターゲットパスを使用して、ロードした SWF ファイルをターゲットとすることができます。ターゲットにロードした SWF ファイルは、ターゲットムービークリップの位置、回転および拡大/縮小プロパティを継承します。ロードしたイメージまたは SWF ファイルの左上隅は、ターゲットムービークリップの基準点に位置が合わされます。一方、ターゲットがルートタイムラインである場合、イメージまたは SWF ファイルの左上隅はステージの左上隅に位置が合わされます。

loadMovie() を使用してロードした SWF ファイルを削除するには、unloadMovie() 関数を使用します。

例

次の例では、ステージに存在する mySquare というムービークリップが、同じディレクトリからロードした SWF ファイル circle.swf で置き換えられます。

```
loadMovie("circle.swf", "mySquare");  
// Equivalent statement: loadMovie("circle.swf", _level0.mySquare);
```

関連項目

[_level](#)、[loadMovieNum\(\)](#)、[unloadMovie\(\)](#)、[unloadMovieNum\(\)](#)

loadMovieNum()

利用状況

Flash Lite 1.1

シンタックス

```
loadMovieNum(url, level [, method])
```

オペランド

url ロードする SWF ファイルの絶対 URL または相対 URL を指定する文字列。相対パスは、レベル 0 の SWF ファイルに相対的である必要があります。スタンドアローンの Flash Lite プレーヤー内で使用する際、または Flash オーサリングアプリケーション内のプレビューモードでテストする際には、すべての SWF ファイルを同じフォルダに置く必要があり、ファイル名にフォルダやディスクドライブの指定を含めることはできません。

level SWF ファイルをロードする先の Flash Lite のレベルを指定する整数。

method 変数を送信するための HTTP メソッドを指定するオプションの文字列パラメータ。値は GET または POST である必要があります。送る変数がない場合は、このパラメータを省略します。GET メソッドは、変数を URL の最後に追加します。このメソッドは、変数のデータ量が少ないときに使用します。POST メソッドは、別の HTTP ヘッダで変数を送信します。このメソッドは、長い文字列の変数に使用します。

説明

関数。ロードした元の SWF ファイルの再生中に SWF ファイルを Flash Lite のレベル内にロードします。

通常、Flash Lite は 1 つの SWF ファイルを表示した後に閉じます。loadMovieNum() 関数を使用すると、複数の SWF ファイルを同時に表示し、別の HTML ドキュメントをロードせずに SWF ファイルを切り替えることができます。

レベルではなくターゲットを指定するには、loadMovieNum() 関数の代わりに loadMovie() 関数を使用します。

Flash Lite では、レベル 0 からレベルが積み重ねられます。これらのレベルは各レベルのオブジェクト以外は透明であり、フィルムのレイヤーに似ています。loadMovieNum() を使用する場合は、SWF ファイルをロードする先の Flash Lite のレベルを指定する必要があります。SWF ファイルをレベル内にロードすると、シンタックス `_levelN` を使用できます。N は SWF ファイルのターゲットとなるレベル番号です。

SWF ファイルをロードするときは、任意のレベル番号を指定できます。SWF ファイルが既にロードされているレベルに SWF ファイルをロードすることもできます。その場合は、新しい SWF ファイルによって既存の SWF ファイルが置き換えられます。SWF ファイルをレベル 0 内にロードすると、Flash Lite の各レベルはアンロードされ、レベル 0 が新しいファイルに置き換えられます。レベル 0 の SWF ファイルによって、ロードした他のすべての SWF ファイルのフレームレート、背景色およびフレームサイズが設定されます。

loadMovieNum() を使用してロードした SWF ファイルまたはイメージを削除するには、[unloadMovieNum\(\)](#) を使用します。

例

次の例では、SWF ファイルがレベル 2 にロードされます。

```
loadMovieNum("http://www.someserver.com/flash/circle.swf", 2);
```

関連項目

[_level](#)、[loadMovie\(\)](#)、[unloadMovieNum\(\)](#)

loadVariables()

利用状況

Flash Lite 1.1

シンタックス

```
loadVariables(url, target [, variables])
```

オペランド

url 変数が存在する絶対 URL または相対 URL を表す文字列。呼び出し元の SWF ファイルが Web ブラウザで実行されている場合、**url** は SWF ファイルと同じドメインに属している必要があります。

target ロードした変数を受け取るムービークリップへのターゲットパス。

variables 変数を送信するための HTTP メソッドを指定するオプションの文字列パラメータ。パラメータは文字列 GET または POST である必要があります。送る変数がない場合は、このパラメータを省略します。GET メソッドは、変数を URL の最後に追加します。このメソッドは、変数のデータ量が少ないときに使用します。POST メソッドは、別の HTTP ヘッダで変数を送信します。このメソッドは、長い文字列の変数に使用します。

説明

関数。テキストファイルや、ColdFusion、CGI、ASP (Active Server Pages)、パーソナルホームページ (PHP) または Perl スクリプトで作成されたテキストなどの外部ファイルからデータを読み取り、ターゲットムービークリップの変数に値を設定します。この関数により、現在の SWF ファイルの変数を新しい値で更新することもできます。

指定された URL にあるテキストは、標準の MIME 形式 **application/x-www-form-urlencoded** (CGI スクリプトで使用される標準形式) である必要があります。変数はいくつでも指定できます。たとえば、次の例では複数の変数が定義されています。

```
company=Adobe&address=600+Townsend&city=San+Francisco&zip=94103
```

変数を特定のレベルにロードするには、loadVariables() 関数の代わりに [loadVariablesNum\(\)](#) 関数を使用します。

例

次の例では、テキストファイルとサーバから変数がロードされます。

```
// load variables from text file on local file system (Symbian Series 60)
on(release, keyPress "1") {
    filePath = "file:///c:/documents/flash/myApp/myvariables.txt";
    loadVariables(filePath, _root);
}

// load variables (from server) into a movieclip
urlPath = "http://www.someserver.com/myvariables.txt";
loadVariables(urlPath, "myClip_mc");
```

関連項目

[loadMovieNum\(\)](#)、[loadVariablesNum\(\)](#)、[unloadMovie\(\)](#)

loadVariablesNum()

利用状況

Flash Lite 1.1

シンタックス

```
loadVariablesNum(url, level [, variables])
```

オペランド

url ロードする変数が存在する絶対 URL または相対 URL を表す文字列。呼び出し元の SWF ファイルが Web ブラウザで実行されている場合、**url** は SWF ファイルと同じドメインに属している必要があります。詳細については、次の「説明」を参照してください。

level 変数を受け取る Flash Lite のレベルを指定する整数。

variables 変数を送信するための HTTP メソッドを指定するオプションの文字列パラメータ。パラメータは文字列 GET または POST である必要があります。送る変数がない場合は、このパラメータを省略します。GET メソッドは、変数を URL の最後に追加します。このメソッドは、変数のデータ量が少ないときに使用します。POST メソッドは、別の HTTP ヘッダで変数を送信します。このメソッドは、長い文字列の変数に使用します。

説明

関数。テキストファイルや、ColdFusion、CGI、ASP、PHP または Perl スクリプトで作成されたテキストなどの外部ファイルからデータを読み取り、Flash Lite の特定のレベルの変数に値を設定します。この関数により、現在の SWF ファイルの変数を新しい値で更新することもできます。

指定された URL にあるテキストは、標準の MIME 形式 **application/x-www-form-urlencoded** (CGI スクリプトで使用される標準形式) である必要があります。変数はいくつでも指定できます。次の例では、複数の変数が定義されます。

```
company=Adobe&address=600+Townsend&city=San+Francisco&zip=94103
```

通常、Flash Lite は 1 つの SWF ファイルを表示した後に閉じます。loadVariablesNum() 関数を使用すると、複数の SWF ファイルを同時に表示し、別の HTML ドキュメントをロードせずに SWF ファイルを切り替えることができます。

変数をターゲットムービークリップにロードするには、loadVariablesNum() 関数の代わりに [loadVariables\(\)](#) 関数を使用します。

関連項目

[getUrl\(\)](#)、[loadMovie\(\)](#)、[loadMovieNum\(\)](#)、[loadVariables\(\)](#)

mbchr()

利用状況

Flash Lite 1.0

シンタックス

```
mbchr(number)
```

オペランド

number マルチバイト文字に変換する数値。

説明

ストリング関数。ASCII コード番号をマルチバイト文字に変換します。

例

次の例では、ASCII コード番号がそれに相当するマルチバイト文字に変換されます。

```
trace (mbchr(65));           // Output: A
trace (mbchr(97));          // Output: a
trace (mbchr(36));          // Output: $

myString = mbchr(51) + mbchr(49);
trace ("result = " + myString); // Output: result = 2
```

関連項目

[mblength\(\)](#)、[mbsubstring\(\)](#)

mblength()

利用状況

Flash Lite 1.0

シンタックス

```
mblength(string)
```

オペランド

string ストリング。

説明

ストリング関数。マルチバイト文字のストリング長を返します。

例

次の例では、`myString` 変数のストリングの長さが表示されます。

```
myString = mbchr(36) + mbchr(50);
trace ("string length = " + mblength(myString));
// Output: string length = 2
```

関連項目[mbchr\(\)](#)、[mbsubstring\(\)](#)

mbord()

利用状況

Flash Lite 1.0

シンタックス`mbord(character)`**オペランド**`character` マルチバイト文字コード番号に変換する文字。**説明**

ストリング関数。指定された文字をマルチバイト文字コード番号に変換します。

例次の例では、`myString` 変数の文字がマルチバイト文字コード番号に変換されます。

```
myString = "A";
trace ("ord = " add mbord(myString)); // Output: 65

myString = "$120";
for (i=1; i<=length(myString); i++) {
    char = substring(myString, i, 1);
    trace ("char ord = " add mbord(char)); // Output: 36, 49, 50, 48
}
```

関連項目[mbchr\(\)](#)、[mbsubstring\(\)](#)

mbsubstring()

利用状況

Flash Lite 1.0

シンタックス`mbsubstring(value, index, count)`**オペランド**`value` 新しいマルチバイト文字ストリングを抽出する元のマルチバイト文字ストリング。`index` 抽出する最初の文字の位置を表す数値。`count` 抽出するストリングの文字数。インデックス文字は除きます。**説明**

ストリング関数。マルチバイト文字ストリングから、新しいマルチバイト文字ストリングを抽出します。

例

次の例では、myString 変数に含まれているストリングからマルチバイト文字ストリングが新たに抽出されます。

```
myString = mbchr(36) add mbchr(49) add mbchr(50) add mbchr(48);  
trace (mbsubstring(myString, 0, 2));// Output: $1
```

関連項目

[mbchr\(\)](#)

nextFrame()

利用状況

Flash Lite 1.0

シンタックス

```
nextFrame()
```

オペランド

なし

説明

関数。次のフレームに再生ヘッドを送り、停止します。

例

次の例では、ユーザーがボタンをクリックすると、再生ヘッドが次のフレームに移動して停止します。

```
on (release) {  
    nextFrame();  
}
```

関連項目

[prevFrame\(\)](#)

nextScene()

利用状況

Flash Lite 1.0

シンタックス

```
nextScene()
```

オペランド

なし

説明

関数。次のシーンのフレーム 1 に再生ヘッドを送り、停止します。

例

次の例では、ユーザーがボタンを離すと、再生ヘッドが次のシーンのフレーム 1 に移動します。

```
on(release) {  
    nextScene();  
}
```

関連項目

[prevScene\(\)](#)

Number()

利用状況

Flash Lite 1.0

シンタックス

```
Number(expression)
```

オペランド

`expression` 数値に変換される式。

説明

関数。次に示すように、パラメータ **expression** を数値に変換し、値を返します。

- **expression** が数値の場合、戻り値は **expression** です。
- **expression** がブール値の場合は、**expression** が true であれば戻り値は 1 となり、**expression** が false であれば戻り値は 0 となります。
- **expression** が文字列である場合、関数は、**expression** を 1.57505e-3 のような指数表現を伴う 10 進数として解析しようとする場合があります。
- **expression** が undefined の場合は、戻り値は -1 です。

例

次の例では、`myString` 変数の文字列が数値に変換され、その数値が `myNumber` 変数に格納された後、数値に 5 を加算した結果が変数 `myResult` に格納されます。`Number()` をブール値に対して呼び出すと、最終行に結果が示されます。

```
myString = "55";  
myNumber = Number(myString);  
myResult = myNumber + 5;  
  
trace (myResult);           // Output: 60  
  
trace (Number(true));      // Output: 1
```

on()

利用状況

Flash Lite 1.0

シンタックス

```
on(event) {  
    // statement(s)  
}
```

オペランド

statement(s) イベントが発生したときに実行される命令。

イベントこのトリガのことをイベントと呼びます。ユーザーイベントが発生すると、中括弧 ({}) 内でそれに続くステートメントが実行されます。次の値のすべてを、**event** パラメータに指定できます。

- **press** - ポインタがボタン上にあるときにボタンを押した場合。
- **release** - ポインタがボタン上にあるときにボタンを離した場合。
- **rollOut** - ポインタがボタン領域の外側に移動した場合。
- **rollOver** - ポインタがボタン上に移動した場合。
- **keyPress "key"** - 指定されたキーを押した場合。パラメータの **key** の部分には、キーコードまたはキー定数を指定します。

説明

イベントハンドラ。関数をトリガするユーザーイベントまたはキー押下を指定します。サポートされていないイベントもあります。

例

次のコードを実行すると、ユーザーが 8 キーを押したときに、**maxscroll** に対するテストが行われた後で、**myText** フィールドが 1 行下にスクロールされ、スクロールする前に **maxscroll** に対してテストを実行します。

```
on (keyPress "8") {  
    if (myText.scroll < myText.maxscroll) {  
        myText.scroll++;  
    }  
}
```

ord()

利用状況

Flash Lite 1.0

シンタックス

```
ord(character)
```

オペランド

character ASCII コード番号に変換する文字。

説明

ストリング関数。文字を ASCII コード番号に変換します。

例

次の例では、**ord()** 関数を使用して、文字 **A** の ASCII コードが表示されます。

```
trace ("multibyte number = " + add ord("A")); // Output: multibyte number = 65
```

play()

利用状況

Flash Lite 1.0

シンタックス

```
play()
```

オペランド

なし

説明

関数。タイムライン内で再生ヘッドを前へ進めます。

例

次の例では、if ステートメントを使用して、ユーザーによって入力された名前の値がチェックされます。ユーザーが「Steve」と入力した場合は、play() 関数が呼び出され、再生ヘッドがタイムライン内で前に移動します。ユーザーが Steve 以外の名前を入力した場合、SWF ファイルは再生されず、変数名が alert のテキストフィールドが表示されます。

```
stop();  
if (name == "Steve") {  
    play();  
} else {  
    alert="You are not Steve!";  
}
```

prevFrame()

利用状況

Flash Lite 1.0

シンタックス

```
prevFrame()
```

オペランド

なし

説明

関数。直前のフレームに再生ヘッドを送り、停止します。現在のフレームが 1 である場合、再生ヘッドは移動しません。

例

次のハンドラが割り当てられているボタンをユーザーがクリックすると、再生ヘッドは前のフレームに送られます。

```
on(release) {  
    prevFrame();  
}
```

関連項目

[nextFrame\(\)](#)

prevScene()

利用状況

Flash Lite 1.0

シンタックス

```
prevScene()
```

オペランド

なし

説明

関数。前のシーンのフレーム 1 に再生ヘッドを送り、停止します。

例

この例では、次のハンドラが割り当てられているボタンをユーザーがクリックすると、再生ヘッドは前のシーンに送られます。

```
on(release) {  
    prevScene();  
}
```

関連項目

[nextScene\(\)](#)

random()

利用状況

Flash Lite 1.0

シンタックス

```
random(value)
```

オペランド

value 整数。

説明

関数。0 から **value** パラメータで指定された整数まで（この整数は含まない）の間のランダムな整数を返します。

例

次の例では、範囲を指定する整数に基づいて数値が生成されます。

```
//pick random number between 0 and 5  
myNumber = random(5);  
trace (myNumber);           // Output: could be 0,1,2,3,4  
  
//pick random number between 5 and 10  
myNumber = random(5) + 5;  
trace (myNumber);           // Output: could be 5,6,7,8,9
```

次の例では、数値が生成された後、変数名として評価されているストリングの末尾にこの数値が連結されます。これは、Flash Lite 1.1 のシンタックスを使用して配列をシミュレートする方法を示した例です。

```
// select random name from list
myNames1 = "Mike";
myNames2 = "Debbie";
myNames3 = "Logan";

ran = random(3) + 1;
ranName = "myNames" + ran;
trace (eval(ranName)); // Output: will be mike, debbie, or logan
```

removeMovieClip()

利用状況

Flash Lite 1.0

シンタックス

```
removeMovieClip(target)
```

オペランド

target `duplicateMovieClip()` を使用して作成されたムービークリップのターゲットパス。

説明

関数。 `duplicateMovieClip()` を使用して作成した元の指定ムービークリップを削除します。

例

次の例では、 `second_mc` という名前の重複したムービークリップが削除されます。

```
duplicateMovieClip("person_mc", "second_mc", 1);
second_mc._x = 55;
second_mc._y = 85;
removeMovieClip("second_mc");
```

関連項目

[duplicateMovieClip\(\)](#)

set()

利用状況

Flash Lite 1.0

シンタックス

```
set(variable, expression)
```

オペランド

variable **expression** パラメータの値を保持する識別子。

expression 変数に代入される値。

説明

ステートメント。変数に値を代入します。変数とは、データを格納する容器のようなものです。容器そのものは変化しませんが、その内容は変化します。SWF ファイルの再生時に変数の値を変更すると、ユーザーが実行した処理に関する情報を記録および保存できます。また、SWF ファイルの再生時に変化した値を記録し、条件が true (真) なのか false (偽) なのかを評価することもできます。

変数には、文字列、数値、ブール値、ムービークリップなどのすべてのデータ型を保持できます。各 SWF ファイルおよびムービークリップのタイムラインには、それぞれ独自の変数のセットがあり、各変数の値は他のタイムラインの変数には影響されません。

例

次の例では、`orig_x_pos` という変数が設定されます。この変数は、`ship` ムービークリップを後で SWF ファイル内の開始位置にリセットできるように、ムービークリップの元の **x** 軸位置を格納します。

```
on(release) {  
    set("orig_x_pos", getProperty("ship", _x));  
}
```

次のコードを実行すると、上記のコードと同じ結果がもたらされます。

```
on(release) {  
    orig_x_pos = ship._x;  
}
```

setProperty()

利用状況

Flash Lite 1.0

シンタックス

```
setProperty(target, property, value/expression)
```

オペランド

target 設定対象のプロパティがあるムービークリップのインスタンス名へのパス。

property 設定するプロパティ。

value プロパティの新しいリテラル値。

expression 評価結果がプロパティの新しい値になる等式。

説明

関数。ムービーの再生時にムービークリップのプロパティ値を変更します。

例

次のステートメントを実行すると、このイベントハンドラに関連するボタンをユーザーがクリックしたときに、`star` ムービークリップの `_alpha` プロパティが 30% に設定されます。

```
on(release) {  
    setProperty("star", _alpha, "30");  
}
```

関連項目

[getProperty\(\)](#)

stop()

利用状況

Flash Lite 1.0

シンタックス

```
stop()
```

オペランド

なし

説明

関数。再生中の SWF ファイルを停止します。この関数は、ボタンでムービークリップを制御する場合に最もよく使用します。

例

次のステートメントを実行すると、このイベントハンドラに関連するボタンをユーザーがクリックしたときに、stop() 関数が呼び出されます。

```
on(release) {  
    stop();  
}
```

stopAllSounds()

利用状況

Flash Lite 1.0

シンタックス

```
stopAllSounds()
```

オペランド

なし

説明

関数。再生ヘッドを停止せずに、SWF ファイルで再生中のサウンドをすべて停止します。ストリーミングするために設定されたサウンドは、そのサウンドが置かれているフレームに再生ヘッドが移動すると再生を再開します。

例

次のコードは、SWF ファイル内のすべてのサウンドをクリックで停止するボタンに適用できます。

```
on(release) {  
    stopAllSounds();  
}
```

String()

利用状況

Flash Lite 1.0

シンタックス

```
String(expression)
```

オペランド

expression スtringに変換される式。

説明

関数。次のリストに記述するように、指定されたパラメータのString表現を返します。

- **expression** が数値である場合、返されるStringは数値のテキスト表現です。
- **expression** がStringである場合、返されるStringは **expression** です。
- **expression** がブール値である場合、返されるStringは true または false です。
- **expression** がムービークリップである場合、戻り値はスラッシュ (/) 表記のムービークリップのターゲットパスです。

例

次の例では、`birthYearNum` が 1976 に設定され、`String()` 関数を使用してStringに変換された後、`eq` 演算子を使用してString「1976」と比較されます。

```
birthYearNum = 1976;  
birthYearStr = String(birthYearNum);  
if (birthYearStr eq "1976") {  
    trace ("birthYears match");  
}
```

substring()

利用状況

Flash Lite 1.0

シンタックス

```
substring(string, index, count)
```

オペランド

string 新しいStringを抽出する元のString。

index 抽出する最初の文字の位置を表す数値。

count 抽出するStringの文字数。インデックス文字は除きます。

説明

関数。Stringの一部を抽出します。この関数は 1 から始まります。一方、`String` クラスのメソッドは 0 から始まります。

例

次の例では、ストリング「Hello World」から最初の 5 文字が抽出されます。

```
origString = "Hello World!";  
newString = substring(origString, 0, 5);  
trace (newString); // Output: Hello
```

tellTarget()

利用状況

Flash Lite 1.0

シンタックス

```
tellTarget(target) {  
    statement(s);  
}
```

オペランド

target 制御するタイムラインのターゲットパスを指定するストリング。

statement(s) 条件の評価結果が true である場合に実行される命令。

説明

関数。 **statement(s)** パラメータで指定された命令を、 **target** パラメータで指定されたタイムラインに適用します。 **tellTarget()** 関数は、ナビゲーションコントロールに使用すると便利です。ステージ上のムービークリップを停止または開始するボタンに、 **tellTarget()** を割り当てます。ムービークリップを、そのクリップ内の特定のフレームにジャンプさせることもできます。たとえば、ボタンに **tellTarget()** を指定して、ステージ上のムービークリップを停止または開始したり、特定のフレームに移動したりするように指示することができます。

例

次の例では、メインタイムライン上のムービークリップインスタンス **ball** が **tellTarget()** によって制御されます。 **ball** インスタンスのフレーム 1 は空白であり、 **stop()** 関数があるので、ステージ上では見えません。ユーザーが 5 キーを押すと、 **tellTarget()** は **ball** 内の再生ヘッドに対して、アニメーションが始まるフレーム 2 に進むように指示します。

```
on(keyPress "5") {  
    tellTarget("ball") {  
        gotoAndPlay(2);  
    }  
}
```

toggleHighQuality()

利用状況

Flash Lite 1.0

シンタックス

```
toggleHighQuality()
```

オペランド

なし

説明

関数。Flash Lite でアンチエイリアス処理のオンとオフを切り替えます。アンチエイリアス処理を行うとオブジェクトのエッジが滑らかになりますが、SWF ファイルの再生は遅くなります。この関数は、Flash Lite のすべての SWF ファイルに影響します。

例

次のコードは、アンチエイリアス処理のオンとオフをクリックで切り替えるボタンに適用できます。

```
on(release) {  
    toggleHighQuality();  
}
```

trace()

利用状況

Flash Lite 1.0

シンタックス

```
trace(expression)
```

オペランド

expression 評価する式。「ムービープレビュー」コマンドを使用して SWF ファイルを Flash オーサリングツールで開くと、**expression** パラメータの値が出力パネルに表示されます。

説明

関数。式を評価し、その結果をプレビューモードで出力パネルに表示します。

この関数は、SWF ファイルのテスト中にプログラミングのメモを記録したり、出力パネルにメッセージを表示したりする場合に使用します。**expression** パラメータを使用して、条件の有無を確認したり、値を出力パネルに表示したりします。trace() 関数は、JavaScript の alert 関数に似ています。

パブリッシュ設定で「Trace アクションを省略」コマンドを使用すると、書き出す SWF ファイルから trace() 関数を削除することができます。

例

次の例では、trace() 関数を使用して、while ループの動作が追跡されます。

```
i = 0;  
while (i++ < 5){  
    trace("this is execution " + i);  
}
```

unloadMovie()

利用状況

Flash Lite 1.0

シンタックス

```
unloadMovie(target)
```

オペランド

target ムービークリップのターゲットパス。

説明

関数。loadMovie()、loadMovieNum() または duplicateMovieClip() を使用してロードしたムービークリップを Flash Lite から削除します。

例

次のコードを実行すると、ユーザーが 3 キーを押したときに、メインタイムライン上の draggable_mc ムービークリップがアンロードされ、movie.swf がドキュメントスタックのレベル 4 にロードされます。

```
on (keypress "3") {  
    unloadMovie ("/draggable_mc");  
    loadMovieNum("movie.swf", 4);  
}
```

次の例では、ユーザーが 3 キーを押すと、レベル 4 にロードされたムービーがアンロードされます。

```
on (keypress "3") {  
    unloadMovieNum(4);  
}
```

関連項目

[loadMovie\(\)](#)

unloadMovieNum()

利用状況

Flash Lite 1.0

シンタックス

```
unloadMovieNum(level)
```

オペランド

level ロードされたムービーのレベル (_levelN)。

説明

関数。loadMovie()、loadMovieNum() または duplicateMovieClip() を使用してロードしたムービークリップを Flash Lite から削除します。

通常、Flash Lite は 1 つの SWF ファイルを表示した後に閉じます。unloadMovieNum() 関数を使用すると、複数の SWF ファイルに同時に影響を与え、別の HTML ドキュメントをロードせずに SWF ファイルを切り替えることができます。

関連項目

[loadMovieNum\(\)](#)

第3章：Flash Liteのプロパティ

このセクションでは、アドビシステムズ社の Macromedia Flash Lite 1.x ソフトウェアが認識するプロパティについて説明します。すべての項目はアルファベット順に並んでおり、先行アンダースコアが無視されます。次の表にプロパティを概説しています。

プロパティ	説明
/ (スラッシュ)	メインムービータイムラインへの参照を指定するか返します。
<code>_alpha</code>	ムービークリップのアルファ透明度の値を返します。
<code>_currentframe</code>	タイムライン内で再生ヘッドがあるフレーム番号を返します。
<code>_focusrect</code>	現在のフォーカスがあるボタンまたはテキストフィールドの周囲に黄色の矩形を表示するかどうかを指定します。
<code>_framesloaded</code>	動的にロードされた SWF ファイルからロードされたフレーム数を返します。
<code>_height</code>	ピクセル単位で示したムービークリップの高さを指定します。
<code>_highquality</code>	現在の SWF ファイルに適用されるアンチエイリアスのレベルを指定します。
<code>_level</code>	<code>_levelN</code> のルートタイムラインへの参照を返します。 <code>_level</code> プロパティを使用してターゲットとする SWF ファイルは、 <code>loadMovieNum()</code> 関数を使用して Flash Lite プレーヤー内に事前にロードしておく必要があります。 <code>_levelN</code> を使用すると、ロードされている SWF ファイルを N によって割り当てられたレベルのターゲットにすることもできます。
<code>maxscroll</code>	スクロール可能なテキストフィールド内の最終行も表示されている場合に、そのフィールドの先頭に表示されるテキストの行番号を示します。
<code>_name</code>	ムービークリップのインスタンス名を返します。これはムービークリップに適用するのみで、メインタイムラインには適用しません。
<code>_rotation</code>	ムービークリップの元の位置からの回転角度を度数で返します。
<code>scroll</code>	変数に関連するテキストフィールドの情報表示を制御します。 <code>scroll</code> プロパティは、テキストフィールドにおける内容の表示開始位置を定義します。設定後、Flash Lite はユーザがテキストフィールドをスクロールするたびに、内容を更新します。
<code>_target</code>	ムービークリップインスタンスのターゲットパスを返します。
<code>_totalframes</code>	ムービークリップのフレームの総数を返します。
<code>_visible</code>	ムービークリップが表示されるかどうかを示します。
<code>_width</code>	ピクセル単位で示したムービークリップの幅を返します。
<code>_x</code>	ムービークリップの x 座標の整数を含みます。
<code>_xscale</code>	ムービークリップの基準点を基点とする、ムービークリップの水平方向の拡大 / 縮小率 (パーセント値) を設定します。
<code>_y</code>	親ムービークリップのローカル座標に相対的なムービークリップの x 座標を設定する整数を含みます。
<code>_yscale</code>	ムービークリップの基準点を基点とする、ムービークリップの垂直方向の拡大 / 縮小率 (パーセント値) を設定します。

/ (スラッシュ)

利用状況

Flash Lite 1.0

シンタックス

/

/targetPath

/:varName

説明

識別子。メインムービータイムラインへの参照を指定するか返します。このプロパティにより提供される機能は、Macromedia Flash 5 の `_root` プロパティにより提供されるものと同様です。

例

タイムラインの変数を指定するには、コロン (:) と組み合わせたスラッシュシンタックス (/) を使用します。

例 1: メインタイムラインの `car` 変数:

```
/:car
```

例 2: メインタイムラインにあるムービークリップインスタンス `mc1` の `car` 変数:

```
/mc1/:car
```

例 3: メインタイムラインにあるムービークリップインスタンス `mc1` にネストされているムービークリップインスタンス `mc2` の `car` 変数:

```
/mc1/mc2/:car
```

例 4: 現在のタイムラインにあるムービークリップインスタンス `mc2` の `car` 変数:

```
mc2/:car
```

_alpha

利用状況

Flash Lite 1.0

シンタックス

```
my_mc:_alpha
```

説明

プロパティ。**my_mc** 変数で指定されたムービークリップのアルファ透明度の値。有効な値は、0 (完全な透明) からデフォルト値の 100 (完全な不透明) までです。ムービークリップ内で `_alpha` が 0 に設定されているオブジェクトは、非表示の場合でもアクティブです。たとえば、`_alpha` プロパティが 0 に設定されたムービークリップのボタンをクリックできます。

例

ボタンイベントハンドラの次のコードにより、ユーザがボタンをクリックしたときの `my_mc` ムービークリップの `_alpha` プロパティが 30% に設定されます。

```
on(release) {  
    tellTarget("my_mc") {  
        _alpha = 30;  
    }  
}
```

`_currentframe`

利用状況

Flash Lite 1.0

シンタックス

```
my_mc:_currentframe
```

説明

プロパティ（読み取り専用）。**my_mc** 変数により指定されるタイムライン内で再生ヘッドがあるフレーム番号を返します。

例

次の例では、`_currentframe` プロパティと `gotoAndStop()` 関数を使用して、ムービークリップ `my_mc` の再生ヘッドを現在の位置から 5 つ先のフレームに進めます。

```
tellTarget("my_mc") {  
    gotoAndStop(_currentframe + 5);  
}
```

関連項目

[gotoAndStop\(\)](#)

`_focusrect`

利用状況

Flash Lite 1.0

シンタックス

```
_focusrect = Boolean;
```

説明

プロパティ（グローバル）。現在のフォーカスがあるボタンまたはテキストフィールドの周囲に黄色の矩形を表示するかどうかを指定します。デフォルト値の `true` では、ユーザーが電話またはモバイルデバイスで上向き矢印キーか下向き矢印キーを押して SWF ファイル内のオブジェクト間を移動するときに、現在フォーカスのあるボタンまたはテキストフィールドの周囲に黄色の矩形が表示されます。黄色の矩形を表示しない場合は、`false` を指定します。

例

次の例では、フォーカスの黄色の矩形がアプリケーションから表示されなくなります。

```
_focusrect = false;
```

`_framesloaded`

利用状況

Flash Lite 1.0

シンタックス

```
my_mc:_framesloaded
```

説明

プロパティ（読み取り専用）。動的にロードされた SWF ファイルからロードされたフレーム数。このプロパティは、特定のフレームとその前のすべてのフレームの内容がロードされており、ユーザのブラウザでローカルに使用できるかどうかを判別する場合に使用できます。これは、大きな SWF ファイルのダウンロード中のモニタとしても役立ちます。たとえば、SWF ファイルの指定されたフレームがロードを完了するまで、その SWF ファイルがロード中であることを示すメッセージをユーザーに表示する場合に使用できます。

例

次の例では、`_framesloaded` プロパティを使用して、すべてのフレームがロードされた時点で SWF ファイルを開始します。すべてのフレームのロードが完了するまでは、ムービークリップインスタンス `loader` の `_xscale` プロパティは比例的に増加し、ロードの進行状況を示すプログレスバーを作成します。

```
if (_framesloaded >= _totalframes) {
    gotoAndPlay ("Scene 1", "start");
} else {
    tellTarget("loader") {
        _xscale = (_framesloaded/_totalframes)*100;
    }
}
```

`_height`

利用状況

Flash Lite 1.0

シンタックス

```
my_mc:_height
```

説明

プロパティ（読み取り専用）。ピクセル単位で示したムービークリップの高さ。

例

イベントハンドラコードの次の例では、ユーザがマウスボタンをクリックしたときのムービークリップの高さが設定されません。

```
on(release) {
    tellTarget("my_mc") {
        _height = 200;
    }
}
```

_highquality

利用状況

Flash Lite 1.0

シンタックス

```
_highquality
```

説明

プロパティ (グローバル)。現在の SWF ファイルに適用されるアンチエイリアスのレベルを指定します。最高品質アンチエイリアス処理には、2 を指定します。高品質アンチエイリアス処理には、1 を指定します。アンチエイリアス処理を避けるには、0 を指定します。

例

次のステートメントでは、現在の SWF ファイルに高品質アンチエイリアス処理が適用されます。

```
_highquality = 1;
```

関連項目

[toggleHighQuality\(\)](#)

_level

利用状況

Flash Lite 1.0

シンタックス

```
_levelN
```

説明

識別子。_levelN のルートタイムラインへの参照を返します。_level プロパティを使用してターゲットとする SWF ファイルは、loadMovieNum() 関数を使用して Flash Lite プレーヤー内に事前にロードしておく必要があります。_levelN を使用すると、ロードされている SWF ファイルを N によって割り当てられたレベルのターゲットにすることもできます。

Flash Lite プレーヤーのインスタンス内にロードした最初の SWF ファイルは、自動的に _level0 内にロードされます。_level0 の SWF ファイルによって、その後にロードするすべての SWF ファイルのフレームレート、背景色、およびフレームサイズが設定されます。次に、SWF ファイルは _level0 の SWF ファイルより高い番号のレベルに積み重ねられます。

Flash Lite プレーヤー内にロードする SWF ファイルごとに、loadMovieNum() 関数を使用してレベルを割り当てる必要があります。レベルは任意の順番で割り当てることができます。SWF ファイルが既に含まれているレベル (_level0 を含む) を割り当てると、そのレベルの SWF ファイルはアンロードされ、新しい SWF ファイルに置き換えられます。

例

次の例では、SWF ファイルをレベル 1 にロードし、その後、フレーム 6 でロードされた SWF ファイルの再生ヘッドを停止します。

```
loadMovieNum("mySWF.swf", 1);

// at least 1 frame later
tellTarget(_level1) {
    gotoAndStop(6);
}
```

関連項目

[loadMovie\(\)](#)

maxscroll

利用状況

Flash Lite 1.1

シンタックス

```
variable_name:maxscroll
```

説明

プロパティ（読み取り専用）。スクロール可能なテキストフィールド内の最終行も表示されている場合に、そのフィールドの先頭に表示されるテキストの行番号を示します。maxscroll プロパティは、scroll プロパティと連携してテキストフィールド内の情報表示を制御します。このプロパティは読み取り専用であり、変更できません。

例

次のコードは、ユーザが 8 キーを押したときに、myText テキストフィールドを 1 行下にスクロールするもので、スクロール前に maxscroll をテストします。

```
on(keyPress "8") {
    if (myText:scroll < myText:maxscroll) {
        myText:scroll++;
    }
}
```

関連項目

[scroll](#)

_name

利用状況

Flash Lite 1.0

シンタックス

```
my_mc:_name
```

説明

プロパティ。my_mc により指定されるムービークリップのインスタンス名。これはムービークリップに適用するのみで、メインタイムラインには適用しません。

例

次の例では、出力パネルにストリングとして `bigRose` ムービークリップの名前を表示します。

```
trace(bigRose: _name);
```

_rotation

利用状況

Flash Lite 1.0

シンタックス

```
my_mc: _rotation
```

説明

プロパティ。ムービークリップの元の位置からの回転角度を度数で示したものの。時計回りに回転させる場合は `0 ~ 180` の値を指定します。反時計回りに回転させる場合は `0 ~ -180` の値を指定します。この範囲を超える値は、`360` の倍数を加算または減算され、範囲内に収まる値になるように調整されます。たとえば、`my_mc: _rotation = 450` というステートメントは `my_mc: _rotation = 90` と同義です。

例

次の例では、ユーザが `2` キーを押したときに `my_mc` ムービークリップを時計回りに `15` 度回転します。

```
on (keyPress "2") {  
    my_mc: _rotation += 15;  
}
```

scroll

利用状況

Flash Lite 1.1.

シンタックス

```
textFieldVariableName: scroll
```

説明

プロパティ。変数に関連するテキストフィールドの情報表示を制御します。`scroll` プロパティは、テキストフィールドにおける内容の表示開始位置を定義します。設定後、Flash Lite はユーザがテキストフィールドをスクロールするたびに、内容を更新します。`scroll` プロパティを使用して、スクロールテキストフィールドを作成したり、長い文節内の特定の段落にユーザを誘導することができます。

例

次のコードでは、ユーザが `2` キーをクリックするたびに、`myText` テキストフィールドが `1` 行上にスクロールされます。

```
on(keyPress "2") {  
    if (myText:scroll > 1) {  
        myText:scroll--;  
    }  
}
```

関連項目

[maxscroll](#)

_target

利用状況

Flash Lite 1.0

シンタックス

```
my_mc:_target
```

説明

プロパティ（読み取り専用）。**my_mc** により指定されるムービークリップインスタンスのターゲットパスを返します。

_totalframes

利用状況

Flash Lite 1.0

シンタックス

```
my_mc:_totalframes
```

説明

プロパティ（読み取り専用）。**my_mc** ムービークリップのフレームの総数を返します。

例

次のコードでは、**mySWF.swf** をレベル 1 にロードし、25 フレーム後にロードされたかどうかを確認します。

```
loadMovieNum("mySWF.swf", 1);

// 25 frames later in the main timeline
if (_level1._framesloaded >= _level1._totalframes) {
    tellTarget("_level1/") {
        gotoAndStop("myLabel");
    }
} else {
    // loop...
}
```

_visible

利用状況

Flash Lite 1.0

シンタックス

```
my_mc:_visible
```

説明

プロパティ。my_mc により指定されるムービークリップを表示するかどうかを示すブール値。_visible プロパティが false に設定されている非表示のムービークリップは、使用できません。たとえば、_visible プロパティが false に設定されたムービークリップ内のボタンは、クリックできません。この方法により明示的に非表示にしない限り、ムービークリップは表示されます。

例

次のコードでは、ユーザが 3 キーを押すと my_mc ムービークリップが無効になり、4 キーを押すと有効になります。

```
on(keyPress "3") {
    my_mc._visible = 0;
}

on(keyPress "4") {
    my_mc._visible = 1;
}
```

_width

利用状況

Flash Lite 1.0

シンタックス

my_mc:_width

説明

プロパティ。ピクセル単位で示したムービークリップの幅。

例

次の例では、ユーザが 5 キーを押したときにムービークリップの幅プロパティが設定されます。

```
on(keyPress "5") {
    my_mc._width = 10;
}
```

_x

利用状況

Flash Lite 1.0

シンタックス

my_mc:_x

説明

プロパティ。親ムービークリップのローカル座標に相対的なムービークリップ（ここでは my_mc により表示）の x 座標を設定する整数。ムービークリップがメインタイムラインにある場合、その座標系はステージの左上隅を (0, 0) として参照します。

変形を含んでいる別のムービークリップの内部にムービークリップがある場合、そのムービークリップの座標系は、それを囲むムービークリップのローカル座標系になります。たとえば、ムービークリップを反時計回りに 90 度回転すると、子ムービークリップは反時計回りに 90 度回転した座標系を継承します。ムービークリップの座標は、基準点の位置を参照します。

例

次の例では、ユーザが 6 キーを押したときに my_mc ムービークリップの水平座標が変更されます。

```
on(keyPress "6") {  
    my_mc._x = 10;  
}
```

関連項目

[_xscale](#)、[_y](#)、[_yscale](#)

_xscale

利用状況

Flash Lite 1.0

シンタックス

```
my_mc._xscale
```

説明

プロパティ。ムービークリップの基準点を基点とする、ムービークリップの水平方向の拡大 / 縮小率（パーセント値）を設定します。デフォルトの基準点は (0, 0) です。

ローカル座標系を拡大または縮小すると、[_x](#) プロパティと [_y](#) プロパティの設定に影響します。この設定はピクセル単位で表されます。たとえば、親ムービークリップを 50% に縮小して、[_x](#) プロパティを設定すると、ムービークリップ内のオブジェクトの移動距離は、ムービーの拡大 / 縮小率が 100% だったときの半分のピクセル数になります。

例

次の例では、ユーザが 7 キーを押したときに my_mc ムービークリップの水平方向の拡大 / 縮小率が変更されます。

```
on(keyPress "7") {  
    my_mc._xscale = 10;  
}
```

関連項目

[_x](#)、[_y](#)、[_yscale](#)

_y

利用状況

Flash Lite 1.0

シンタックス

```
my_mc._y
```

説明

プロパティ。親ムービークリップのローカル座標に相対的なムービークリップ（ここでは `my_mc` により表示）の `y` 座標を設定する整数。ムービークリップがメインタイムラインにある場合、その座標系はステージの左上隅を (0,0) として参照します。

ムービークリップが、変形されている別のムービークリップの内部にある場合、そのムービークリップの座標系は、それを囲むムービークリップのローカル座標系になります。たとえば、ムービークリップを反時計回りに 90 度回転すると、子ムービークリップは反時計回りに 90 度回転した座標系を継承します。ムービークリップの座標は、基準点の位置を参照します。

例

次のコードは、ユーザが 1 キーを押したときに親クリップの (0,0) 座標の 10 ピクセル下に `my_mc` ムービークリップの `y` 座標を設定します。

```
on(keyPress "1") {  
    my_mc:_y = 10;  
}
```

関連項目

[_x](#)、[_xscale](#)、[_yscale](#)

_yscale

利用状況

Flash Lite 1.0

シンタックス

```
my_mc:_yscale
```

説明

プロパティ。ムービークリップの基準点を基点とする、ムービークリップの垂直方向の拡大 / 縮小率（パーセント値）を設定します。デフォルトの基準点は (0,0) です。

ローカル座標系を拡大または縮小すると、`_x` プロパティと `_y` プロパティの設定に影響します。この設定はピクセル単位で表されます。たとえば、親ムービークリップを 50% に縮小して、`_y` プロパティを設定すると、ムービークリップ内のオブジェクトの移動距離は、ムービーの拡大 / 縮小率が 100% だったときの半分のピクセル数になります。

例

次の例では、ユーザが 1 キーを押したときに `my_mc` ムービークリップの垂直方向の拡大 / 縮小率が 10% に変更されます。

```
on(keyPress "1") {  
    my_mc:_yscale = 10;  
}
```

関連項目

[_x](#)、[_xscale](#)、[_y](#)

第4章：Flash Lite のステートメント

このセクションでは、アドビ システムズ社の Macromedia Flash Lite 1.x の ActionScript ステートメントのシンタックスおよび使用法について説明します。ステートメントとは、特定のアクションを実行または指定する言語エレメントです。次の表にステートメントの要約を示します。

ステートメント	説明
<code>break</code>	ループ本体の残りの部分をスキップし、繰り返し処理を停止して、ループステートメントの次のステートメントを実行するように、Flash Lite に指示します。
<code>case</code>	<code>switch</code> ステートメントの条件を定義します。 <code>statements</code> パラメータのステートメントが実行されるのは、 <code>case</code> キーワードに続く <code>expression</code> パラメータが <code>switch</code> ステートメントの <code>expression</code> パラメータと等しい場合です。
<code>continue</code>	ループの終わりまで制御が正常に通過したかのように、最も内側のループ内の残りのステートメントをすべてスキップして、ループの次の反復を開始します。
<code>do..while</code>	ステートメントを実行した後で、ループ内の条件が <code>true</code> である場合は、その条件を評価します。
<code>else</code>	if ステートメントの条件が <code>false</code> として評価されたときに実行されるステートメントを指定します。
<code>else if</code>	条件を評価し、最初の if ステートメントの条件から <code>false</code> 値が返された場合に実行するステートメントを指定します。
<code>for</code>	<code>init</code> (初期化) 式を 1 度評価してから、ループシーケンスを開始します。これにより、条件が <code>true</code> として評価された場合に、 <code>statement</code> が実行され、次の式が評価されます。
<code>if</code>	条件を評価して、SWF ファイル内の次のアクションを決定します。条件が <code>true</code> の場合は、条件に続く中括弧 (<code>{</code>) 内のステートメントが実行されます。条件が <code>false</code> の場合、中括弧内のステートメントはスキップされ、中括弧の後のステートメントが実行されます。
<code>switch</code>	if ステートメントと同様に、 <code>switch</code> ステートメントは条件を評価し、評価結果が <code>true</code> の場合にステートメントを実行します。
<code>while</code>	式を評価して、評価結果が <code>true</code> の場合は、ループ内の単一のステートメントまたは一連のステートメントを繰り返し実行します。

break

利用状況

Flash Lite 1.0

シンタックス

`break`

パラメータ

なし

説明

ステートメント。ループ (`for`、`do..while` または `while`) 内で使用します。または、`switch` ステートメント内の特定のケースと関連するステートメントのブロック内でも使用します。`break` ステートメントは、ループ本体の残りの部分をスキップし、繰り返し処理を停止して、ループステートメントの次のステートメントを実行するように、Flash Lite に指示します。`break`

ステートメントを使用すると、ActionScript インタプリタがその case ブロック内の残りのステートメントをスキップして、周りを囲んでいる switch ステートメントの後の最初のステートメントにジャンプします。ネストされている一連のループを終了するには、このステートメントを使用します。

例

次の例では、break ステートメントを使用して無限ループを終了します。

```
i = 0;
while (true) {
    if (i >= 100) {
        break;
    }
    i++;
}
```

関連項目

[case](#)、[do..while](#)、[for](#)、[switch](#)、[while](#)

case

利用状況

Flash Lite 1.0

シンタックス

```
case expression: statements
```

パラメータ

expression 任意の式。

statements 任意のステートメント。

説明

ステートメント。switch ステートメントの条件を定義します。**statements** パラメータのステートメントが実行されるのは、case キーワードに続く **expression** パラメータが switch ステートメントの **expression** パラメータと等しい場合です。

case ステートメントを switch ステートメントの外側で使用すると、エラーが発生し、コードはコンパイルされません。

例

次の例では、myNum パラメータが 1 として評価された場合は case 1 に続く [trace\(\)](#) ステートメント、myNum パラメータが 2 として評価された場合は case 2 に続く [trace\(\)](#) ステートメントの順で実行されます。いずれの case 式も number パラメータと一致しない場合は、default キーワードの後にある [trace\(\)](#) ステートメントが実行されます。

```
switch (myNum) {  
  case 1:  
    trace ("case 1 tested true");  
    break;  
  case 2:  
    trace ("case 2 tested true");  
    break;  
  case 3:  
    trace ("case 3 tested true");  
    break;  
  default:  
    trace ("no case tested true")  
}
```

次の例では、最初のケースグループで分割が発生しないので、数値が 1 の場合は、A と B の両方が出力パネルに表示されません。

```
switch (myNum) {  
  case 1:  
    trace ("A");  
  case 2:  
    trace ("B");  
    break;  
  default:  
    trace ("D")  
}
```

関連項目

[switch](#)

continue

利用状況

Flash Lite 1.0

シンタックス

continue

パラメータ

なし

説明

ステートメント。ループの終わりまで制御が正常に通過したかのように、最も内側のループ内の残りのステートメントをすべてスキップして、ループの次の反復を開始します。ループの外部では作用しません。

- while ループで continue を使用すると、Flash インタプリタはループ本体の残りの部分をスキップし、ループの上端にジャンプします。そこで、条件が再度評価されます。
- do..while ループで continue を使用すると、Flash インタプリタはループ本体の残りの部分をスキップし、ループの下端にジャンプします。そこで、条件が再度評価されます。
- for ループで continue を使用すると、Flash インタプリタはループ本体の残りの部分をスキップし、for ループの式後の評価にジャンプします。

例

次の while ループで continue を使用すると、Flash Lite はループ本体の残りの部分をスキップし、ループの上端にジャンプします。そこで、条件が再度評価されます。

```
i = 0;
while (i < 10) {
    if (i % 3 == 0) {
        i++;
        continue;
    }
    trace(i);
    i++;
}
```

次の do..while ループで continue を使用すると、Flash Lite はループ本体の残りの部分をスキップし、ループの下端にジャンプします。そこで、条件が再度評価されます。

```
i = 0;
do {
    if (i % 3 == 0) {
        i++;
        continue;
    }
    trace(i);
    i++;
} while (i < 10);
```

for ループで continue を使用すると、Flash Lite はループ本体の残りの部分をスキップします。次の例では、3 を法とする i が 0 に等しい場合、trace(i) ステートメントはスキップされます。

```
for (i = 0; i < 10; i++) {
    if (i % 3 == 0) {
        continue;
    }
    trace(i);
}
```

関連項目

[do..while](#)、[for](#)、[while](#)

do..while

利用状況

Flash Lite 1.0

シンタックス

```
do {
    statement(s)
} while (condition)
```

パラメータ

statement(s) **condition** パラメータが true として評価された場合に実行するステートメント。

condition 評価する条件。

説明

ステートメント。ステートメントを実行した後で、ループ内の条件が `true` である場合は、その条件を評価します。

例

次の例では、インデックス変数の値が 10 未満の場合に、この変数がインクリメントされます。

```
i = 0;
do {
    //trace (i);           // output: 0,1,2,3,4,5,6,7,8,9
    i++;
} while (i<10);
```

関連項目

[break](#)、[continue](#)、[for](#)、[while](#)

else

利用状況

Flash Lite 1.0

シンタックス

```
if (condition){
    t-statement(s);
} else {
    f-statement(s);
}
```

パラメータ

condition 評価結果が `true` または `false` になる式。

t-statement(s) 条件の評価結果が `true` である場合に実行される命令。

f-statement(s) 条件の評価結果が `false` である場合に実行される一連の代替命令。

説明

ステートメント。if ステートメントの条件が `false` として評価されたときに実行されるステートメントを指定します。

例

次の例では、条件付きの `else` ステートメントの使用法を示します。実際の例には、条件に基づいて特定のアクションを実行するためのコードが含まれます。

```
currentHighestDepth = 1;
if (currentHighestDepth == 2) {
    //trace ("currentHighestDepth is 2");
} else {
    //trace ("currentHightestDepth is not 2");
}
```

関連項目

[if](#)

else if

利用状況

Flash Lite 1.0

シンタックス

```
if (condition){
    statement(s);
} else if (condition){
    statement(s);
}
```

パラメータ

condition 評価結果が true または false になる式。

statement(s) if ステートメントで指定された条件が false のときに実行される一連のステートメント。

説明

ステートメント。条件を評価し、最初の if ステートメントの条件から false 値が返された場合に実行するステートメントを指定します。else if 条件から true 値が返されると、Flash インタプリタは中括弧 ({}) 内の else if 条件に続くステートメントを実行します。else if 条件が false の場合、中括弧内のステートメントはスキップされ、中括弧の後のステートメントが実行されます。スクリプト内に分岐処理を作成するには、elseif ステートメントを使用します。

例

次の例では、else if ステートメントを使用して、オブジェクトの両側が特定の境界内にあるかどうかをチェックされます。

```
person_mc.xPos = 100;
leftBound = 0;
rightBound = 100;
if (person_mc.xPos <= leftBound) {
    //trace ("Clip is to the far left");
} else if (person_mc.xPos >= rightBound) {
    //trace ("Clip is to the far right");
} else {
    //trace ("Your clip is somewhere in between");
}
```

関連項目

[if](#)

for

利用状況

Flash Lite 1.0

シンタックス

```
for (init; condition; next) {
    statement(s);
}
```

パラメータ

init ループの開始前に評価される式。通常は代入式です。

condition 評価結果が **true** または **false** になる式。条件の評価は、ループの各反復の前に行われます。条件の評価結果が **false** の場合、ループは終了します。

next ループの各反復の後で評価される式。通常は、インクリメント演算子 (++) またはデクリメント演算子 (--) を使用した代入式です。

statement(s) ループ内で実行する 1 つ以上の命令。

説明

ステートメント。**init** (初期化) 式を 1 度評価した後で、ループシーケンスを開始するループコンストラクト。これにより、**condition** が **true** として評価された場合に、**statement** が実行され、次の式が評価されます。

プロパティの中には、**for** **or** **for..in** ステートメントで列挙できないものがあります。たとえば、**_x** や **_y** のようなムービークリッププロパティは列挙されません。

例

次の例では、**for** ループを使用して、1 から 100 の数値が加算されます。

```
sum = 0;
for (i = 1; i <= 100; i++) {
    sum = sum + i;
}
```

関連項目

[++ \(インクリメント\)](#)、[-- \(デクリメント\)](#)、[do..while](#)、[while](#)

if

利用状況

Flash Lite 1.0

シンタックス

```
if (condition) {
    statement(s);
}
```

パラメータ

condition 評価結果が **true** または **false** になる式。

statement(s) 条件の評価結果が **true** である場合に実行される命令。

説明

ステートメント。条件を評価して、SWF ファイル内の次のアクションを決定します。条件が **true** の場合は、条件に続く中括弧 ({}) 内のステートメントが実行されます。条件が **false** の場合、中括弧内のステートメントはスキップされ、中括弧の後のステートメントが実行されます。スクリプト内に分岐処理を作成するには、**if** ステートメントを使用します。

例

次の例では、括弧内の条件で変数 **name** を評価し、リテラル値 "Erica" が含まれているかどうかを確認します。「Erica」が含まれている場合は、**play()** 関数が実行されます。

```
if(name eq "Erica"){  
    play();  
}
```

switch

利用状況

Flash Lite 1.0

シンタックス

```
switch (expression){  
    caseClause:  
    [defaultClause:]  
}
```

パラメータ

expression 任意の数値式。

caseClause 式、コロンおよび一連のステートメントの前に置かれる **case** キーワード。ステートメントが実行されるのは、式が **switch** の **expression** パラメータと一致する場合です。

defaultClause ステートメントの前に置かれるオプションの **default** キーワード。ステートメントが実行されるのは、いずれの **case** 式も **switch** の **expression** パラメータと一致しない場合です。

説明

ステートメント。ActionScript ステートメントの分岐構造を作成します。if ステートメントと同様に、switch ステートメントは条件を評価し、評価結果が true の場合にステートメントを実行します。

Switch ステートメントには、**default** と呼ばれるフォールバックオプションが含まれています。評価結果が **true** のステートメントがほかにはない場合は、**default** ステートメントが実行されます。

例

次の例では、myNum パラメータが 1 として評価された場合は case 1 に続く **trace()** ステートメント、myNum パラメータが 2 として評価された場合は case 2 に続く **trace()** ステートメントの順で実行されます。いずれの case 式も number パラメータと一致しない場合は、**default** キーワードの後にある **trace()** ステートメントが実行されます。

```
switch (myNum) {  
    case 1:  
        trace ("case 1 tested true");  
        break;  
    case 2:  
        trace ("case 2 tested true");  
        break;  
    case 3:  
        trace ("case 3 tested true");  
        break;  
    default:  
        trace ("no case tested true")  
}
```

次の例では、最初のケースグループに分割が含まれていないので、数値が 1 の場合は、A と B の両方が出力パネルに表示されます。

```
switch (myNum) {  
    case 1:  
        trace ("A");  
    case 2:  
        trace ("B");  
        break;  
    default:  
        trace ("D")  
}
```

関連項目

[case](#)

while

利用状況

Flash Lite 1.0

シンタックス

```
while(condition) {  
    statement(s);  
}
```

パラメータ

condition while ステートメントが実行されるたびに評価される式。

statement(s) 条件の評価結果が true である場合に実行される命令。

説明

ステートメント。式を評価して、評価結果が true の場合は、ループ内の単一のステートメントまたは一連のステートメントを繰り返し実行します。

条件の評価は、ステートメントブロックが実行される前に行われます。テストから true が返された場合は、ステートメントブロックが実行されます。条件が false の場合、ステートメントブロックはスキップされ、while ステートメントのステートメントブロックの後にある最初のステートメントが実行されます。

一般にループ処理は、カウンタ変数が指定値より小さい間はアクションを実行するという場合に使用します。各ループの最後で、指定された値に達するまでカウンタがインクリメントされます。指定された値に達すると、条件は true でなくなり、ループは終了します。

while ステートメントは、次の手順を実行します。手順 1 ~ 4 の各繰り返しはループの反復と呼ばれます。条件の評価は、各反復の最初に行われます。

- 1 式 **condition** が評価されます。
- 2 **condition** の評価結果が true であるか、ブール値 true に変換される値（ゼロ以外の数値など）である場合は、手順 3 に進みます。
それ以外の場合は、while ステートメントが完了し、while ループの直後のステートメントから実行が再開されます。
- 3 ステートメントブロック **statement(s)** を実行します。
- 4 手順 1 に進みます。

例

次の例では、インデックス変数 `i` の値が 10 未満の場合に、ループが実行されます。

```
i = 0;
while(i < 10) {
    trace ("i = " + i); // Output: 1,2,3,4,5,6,7,8,9
}
```

関連項目

[continue](#)、[do..while](#)、[for](#)

第 5 章：Flash Lite の演算子

このセクションでは、アドビ システムズ社の Macromedia Flash Lite 1.x の ActionScript 演算子のシンタックスおよび使用法について説明します。すべての項目はアルファベット順に並んでいます。しかし、一部の演算子はシンボルで、テキスト記述によってアルファベット順に並べられます。

次の表に、この節の演算子を概説します。

演算子	説明
add (ストリング連結)	2 つ以上のストリングを連結 (結合) します。
+=(加算後代入)	<i>expression1</i> に <i>expression1</i> + <i>expression2</i> の値を代入します。
および	論理積 (AND) 演算を行います。
=(代入)	<i>expression2</i> の値 (右側のオペランド) を <i>expression1</i> の変数またはプロパティに代入します。
/* (コメントのブロック)	スクリプトコメントのブロックを示します。開始コメントタグ (/*) と終了コメントタグ (*/) の間の文字はすべてコメントと解釈され、ActionScript インタプリタでは無視されます。
,(コンマ)	最初に <i>expression1</i> 、次に <i>expression2</i> を評価し、 <i>expression2</i> の値を返します。
// (コメント)	スクリプトコメントの先頭を示します。コメント行区切り記号 (//) と行末の間に表示される文字はすべてコメントと解釈され、ActionScript インタプリタで無視されます。
?(条件)	<i>expression1</i> を評価し、 <i>expression1</i> の値が true である場合は <i>expression2</i> の値を返します。それ以外の場合は <i>expression3</i> の値を返します。
-- (デクリメント)--	<i>expression</i> から 1 を減算します。プリデクリメント形式の演算子 (-- <i>expression</i>) は、 <i>expression</i> から 1 を減算し、結果を数値として返します。 ポストデクリメント形式の演算子 (<i>expression</i> --) は、 <i>expression</i> から 1 を減算し、 <i>expression</i> の初期値 (減算前の値) を返します。
/(除算)	<i>expression1</i> を <i>expression2</i> で除算します。
/(= (除算後代入)	<i>expression1</i> に <i>expression1</i> / <i>expression2</i> の値を代入します。
.(ドット)	ネストされた子のムービークリップ、変数、またはプロパティにアクセスするためにムービークリップの階層をナビゲートする場合に使用します。
++ (インクリメント)	<i>expression</i> に 1 を加算します。式は、変数、配列の要素、またはオブジェクトのプロパティです。プリインクリメント形式の演算子 (++ <i>expression</i>) は、 <i>expression</i> に 1 を加算し、結果を数値として返します。ポストインクリメント形式の演算子 (<i>expression</i> ++) は、 <i>expression</i> に 1 を加算し、 <i>expression</i> の初期値 (加算前の値) を返します。
&& (論理積 (AND))	<i>expression1</i> (演算子の左側の式) を評価し、その式の評価が false の場合は false を返します。 <i>expression1</i> の評価が true であれば、 <i>expression2</i> (演算子の右側の式) が評価されます。 <i>expression2</i> の評価が true であれば、最終結果は true です。そうでなければ、false です。
!(論理否定 (NOT))	変数や式のブール値を反転します。 <i>expression</i> が変数で、その絶対値または変換された値が true である場合、! <i>expression</i> の値は false です。式 <i>x</i> && <i>y</i> の評価結果が false の場合、式 !(<i>x</i> && <i>y</i>) の評価結果は true です。
(論理和 (OR))	<i>expression1</i> と <i>expression2</i> を評価します。一方の式または両式の評価が true である場合、結果は true になります。両式の評価が false である場合のみ、結果は false になります。論理和 (OR) 演算子は任意の数のオペランドに適用できます。いずれかのオペランドの評価が true であれば、結果は true です。

演算子	説明
% (剰余)	<i>expression1</i> を <i>expression2</i> で割ったときの剰余を計算します。 <i>expression</i> のオペランドが非数値である場合、剰余演算子はこれを数値に変換しようとします。
%= (剰余代入)	<i>expression1</i> に <i>expression1</i> % <i>expression2</i> の値を代入します。
*= (乗算後代入)	<i>expression1</i> に <i>expression1</i> * <i>expression2</i> の値を代入します。
* (乗算)	2つの数値や式を乗算します。
+ (数値式加算)	数値式を加算します。
== (数値の等価)	等価性をテストします。 <i>expression1</i> が <i>expression2</i> と等しい場合、結果は true です。
> (数値が対象より大きい)	2つの式を比較し、 <i>expression1</i> が <i>expression2</i> より大きいかどうかを判定します。大きい場合は true を返します。 <i>expression1</i> が <i>expression2</i> より小さいか等しい場合は、false を返します。
>= (数値が対象より大きいか等しい)	2つの式を比較し、 <i>expression1</i> が <i>expression2</i> より大きいか等しいか (true)、または <i>expression1</i> が <i>expression2</i> より小さいか (false) を判定します。
<> (数値の不等価)	不等価性をテストします。 <i>expression1</i> が <i>expression2</i> と等しい場合、結果は false です。
< (数値が対象より小さい)	2つの式を比較し、 <i>expression1</i> が <i>expression2</i> より小さいかどうかを判定します。小さい場合は true を返します。 <i>expression1</i> が <i>expression2</i> より大きいか等しい場合は、false を返します。
<= (数値が対象より小さいか等しい)	2つの式を比較し、 <i>expression1</i> が <i>expression2</i> より小さいか等しいかどうかを判定します。小さいか等しい場合は true を返します。それ以外の場合は、false を返します。
() (括弧)	パラメータをグループ化するか、複数の式を順番に評価します。または、パラメータを囲み、結果をパラメータとして括弧の外側にある関数に渡します。
" (ストリング区切り記号)	0個以上の文字のシーケンスが引用符で囲まれている場合、リテラル値の文字を表し、変数や数値、その他の ActionScript エレメントではなく、ストリングと見なされます。
eq (ストリングの等価)	2つの式が等しいかどうか比較し、 <i>expression1</i> のストリング表現が <i>expression2</i> のストリング表現と等しい場合は true を返し、それ以外の場合は false を返します。
gt (ストリングが対象より大きい)	<i>expression1</i> のストリング表現を <i>expression2</i> のストリング表現と比較し、 <i>expression1</i> が <i>expression2</i> より大きい場合は true を返します。それ以外の場合は false を返します。
ge (ストリングが対象より大きいか等しい)	<i>expression1</i> のストリング表現を <i>expression2</i> のストリング表現と比較し、 <i>expression1</i> が <i>expression2</i> より大きいか等しい場合は true を返します。それ以外の場合は false を返します。
ne (ストリングの不等価)	<i>expression1</i> のストリング表現を <i>expression2</i> のストリング表現と比較し、 <i>expression1</i> が <i>expression2</i> と等しくない場合は true を返します。それ以外の場合は false を返します。
lt (ストリングが対象より小さい)	<i>expression1</i> のストリング表現を <i>expression2</i> のストリング表現と比較し、 <i>expression1</i> が <i>expression2</i> より小さい場合は true を返します。それ以外の場合は false を返します。
le (ストリングが対象より小さいか等しい)	<i>expression1</i> のストリング表現を <i>expression2</i> のストリング表現と比較し、 <i>expression1</i> が <i>expression2</i> より小さいか等しい場合は true を返します。それ以外の場合は false を返します。
- (減算)	符号反転や減算に使用します。
-= (減算後代入)	<i>expression1</i> に <i>expression1</i> - <i>expression2</i> の値を代入します。

add (ストリング連結)

利用状況

Flash Lite 1.0

シンタックス

```
string1 add string2
```

オペランド

string1, string2 ストリング。

説明

演算子。2つ以上のストリングを連結 (結合) します。

例

次の例では、2つのストリング値を連結して **catalog** というストリングを作成しています。

```
conStr = "cat" add "alog";  
trace (conStr); // output: catalog
```

関連項目

[+ \(数値式加算\)](#)

+= (加算後代入)

利用状況

Flash Lite 1.0

シンタックス

```
expression1 += expression2
```

オペランド

expression1, expression2 数値またはストリング。

説明

演算子 (算術複合値の代入)。**expression1** に **expression1 + expression2** の値を代入します。たとえば、次の2つのステートメントは同じ結果になります。

```
x += y;  
x = x + y;
```

加算 (+) 演算子のすべての規則が、加算して代入 (+=) 演算子に適用されます。

例

次の例では、加算して代入 (+=) 演算子を使用して x の値を y 増分しています。

```
x = 5;  
y = 10;  
x += y;  
trace(x); // output: 15
```

関連項目

[+ \(数値式加算\)](#)

および

利用状況

Flash Lite 1.0

シンタックス

```
condition1 and condition2
```

オペランド

condition1, condition2 true または false に評価される条件または式。

説明

演算子。論理積 (AND) 演算を行います。

例

次の例では、and 演算子を使用してゲームの勝敗判定をテストしています。turns 変数と score 変数は、ゲームの回数または得点に応じて更新されます。次のスクリプトでは、3 回以内に 75 点以上を得点すると、出力パネルに "You Win the Game!" と表示されます。

```
turns = 2;  
score = 77;  
winner = (turns <= 3) and (score >= 75);  
if (winner) {  
    trace("You Win the Game!");  
} else {  
    trace("Try Again!");  
}  
// output: You Win the Game!
```

関連項目

[&& \(論理積 \(AND\)\)](#)

= (代入)

利用状況

Flash Lite 1.0

シンタックス

```
expression1 = expression2
```

オペランド

expression1 変数またはプロパティ。

expression2 値。

説明

演算子。 **expression2** の値 (右側のオペランド) を **expression1** の変数またはプロパティに代入します。

例

次の例では、代入 (=) 演算子を使用して数値を変数 **weight** に代入しています。

```
weight = 5;
```

次の例では、代入 (=) 演算子を使用してストリング値を変数 **greeting** に代入しています。

```
greeting = "Hello, " and personName;
```

/* (コメントのブロック)

利用状況

Flash Lite 1.0

シンタックス

```
/* comment */  
/* comment  
comment */
```

オペランド

comment 任意の文字。

説明

コメント区切り記号。スクリプトコメントのブロックを示します。開始コメントタグ (**/***) と終了コメントタグ (***/**) の間の文字はすべてコメントと解釈され、ActionScript インタプリタでは無視されます。

単一行のコメントを識別するには、**//** (コメント行区切り記号) を使用します。連続した複数行のコメントブロックを識別するには、**/*** (コメントブロック区切り記号) を使用します。この形式のコメントブロック区切り記号を使用する際に閉じるタグ (***/**) を省略すると、エラーメッセージが返されます。コメントをネストしようとすると、エラーメッセージが返されます。

コメントの終わりは、開始コメントタグ (**/***) の後、最初に出現する終了コメントタグ (***/**) によって示されます。開始コメントタグ (**/***) がその間にいくつ指定されていても結果は同じです。

関連項目

[// \(コメント\)](#)

, (コンマ)

利用状況

Flash Lite 1.0

シンタックス

```
expression1, expression2
```

オペランド

expression1, expression2 数値、または評価結果が数値になる式。

説明

演算子。最初に **expression1**、次に **expression2** を評価し、**expression2** の値を返します。

例

次の例では、括弧 () 演算子なしでコンマ (,) 演算子を使用しています。括弧 () 演算子がない場合、コンマ演算子は最初の式の値だけを返していることがわかります。

```
v = 0;
v = 4, 5, 6;
trace(v); // output: 4
```

次の例では、括弧 () 演算子と組み合わせてコンマ (,) 演算子を使用しています。括弧 () 演算子と組み合わせた場合、コンマ演算子は最後の式の値を返していることがわかります。

```
v = 0;
v = (4, 5, 6);
trace(v); // output: 6
```

次の例では、括弧 () 演算子なしでコンマ (,) 演算子を使用しています。コンマ演算子はすべての式を順番に評価した上で最初の式の値を返していることがわかります。2 番目の式 z++ を評価して、z に 1 を加えています。

```
v = 0;
z = 0;
v = v + 4, z++, v + 6;
trace(v); // output: 4
trace(z); // output: 1
```

次の例は、括弧 () 演算子が使われている以外は、前の例と同じです。括弧 () 演算子と組み合わせた場合、コンマ (,) 演算子は一連の式の最後の式の値を返していることがわかります。

```
v = 0;
z = 0;
v = (v + 4, z++, v + 6);
trace(v); // output: 6
trace(z); // output: 1
```

関連項目

[for、\(\) \(括弧\)](#)

// (コメント)

利用状況

Flash Lite 1.0

シンタックス

```
// comment
```

オペランド

comment 任意の文字。

説明

コメント区切り記号。スクリプトコメントの先頭を示します。コメント行区切り記号 (`//`) と行末の間に表示される文字はすべてコメントと解釈され、ActionScript インタプリタで無視されます。

例

次の例は、コメント行区切り記号を使用して 1 行目、3 行目、5 行目、7 行目をコメントとして識別しています。

```
// Record the X position of the ball movie clip.  
ballX = ball._x;  
// Record the Y position of the ball movie clip.  
ballY = ball._y;  
// Record the X position of the bat movie clip.  
batX = bat._x;  
// Record the Y position of the bat movie clip.  
batY = bat._y;
```

関連項目

[/* \(コメントのブロック\)](#)

?: (条件)

利用状況

Flash Lite 1.0

シンタックス

```
expression1 ? expression2 : expression3
```

オペランド

expression1 評価結果がブール値になる式。通常は $x < 5$ などの比較式。

expression2, expression3 任意のタイプの値。

説明

演算子。 **expression1** を評価し、 **expression1** の値が `true` である場合は **expression2** の値を返します。それ以外の場合は **expression3** の値を返します。

例

次の例では、 **expression1** の評価結果が `true` なので、変数 `x` の値が `z` に代入されます。

```
x = 5;  
y = 10;  
z = (x < 6) ? x : y;  
trace (z); // output: 5
```

-- (デクリメント)

利用状況

Flash Lite 1.0

シンタックス

```
--expression
```

```
expression--
```

オペランド

なし

説明

演算子 (算術)。**expression** から 1 を引くプリデクリメント単項演算子またはポストデクリメント単項演算子です。プリデクリメント形式の演算子 (**--expression**) は、**expression** から 1 を減算し、結果を数値として返します。ポストデクリメント形式の演算子 (**expression--**) は、**expression** から 1 を減算し、**expression** の初期値 (減算前の値) を返します。

例

次の例は、aWidth を 2 にデクリメント (aWidth - 1 = 2) して、結果を bWidth として返すプリデクリメント形式の演算子を示します。

```
aWidth = 3;  
bWidth = --aWidth;  
// The bWidth value is equal to 2.
```

次の例は、aWidth を 2 にデクリメント (aWidth - 1 = 2) して、aWidth の元の値を結果 bWidth として返すポストデクリメント形式の演算子を示します。

```
aWidth = 3;  
bWidth = aWidth--;  
// The bWidth value is equal to 3.
```

/ (除算)

利用状況

Flash Lite 1.0

シンタックス

```
expression1 / expression2
```

オペランド

expression1, expression2 数値、または評価結果が数値になる式。

説明

演算子 (算術)。**expression1** を **expression2** で除算します。除算演算の結果は倍精度の浮動小数点数です。

例

次のステートメントでは、浮動小数点数 22.0 を 7.0 で除算し、結果を出力パネルに表示します。

```
trace(22.0 / 7.0);
```

結果は、浮動小数点数の 3.1429 です。

/= (除算後代入)

利用状況

Flash Lite 1.0

シンタックス

```
expression1 /= expression2
```

オペランド

expression1, expression2 数値、または評価結果が数値になる式。

説明

演算子 (算術複合値の代入)。**expression1** に **expression1/expression2** の値を代入します。たとえば、次の 2 つのステートメントは同じです。

```
x /= y  
x = x / y
```

例

次の例では、/= 演算子に変数と数値を使用しています。

```
x = 10;  
y = 2;  
x /= y;  
// The expression x now contains the value 5.
```

.(ドット)

利用状況

Flash Lite 1.0

シンタックス

```
instancename.variable
```

```
instancename.childinstance.variable
```

オペランド

instancename ムービークリップのインスタンス名。

childinstance 別のムービークリップの子になっている (ネストされている) ムービークリップインスタンス。

variable 指定されたムービークリップインスタンス名のタイムラインの変数。

説明

演算子。ネストされた子のムービークリップ、変数、またはプロパティにアクセスするためにムービークリップの階層をナビゲートする場合に使用します。

例

次の例では、ムービークリップ person_mc 内の変数 hairColor の現在の値を識別します。

```
person_mc.hairColor
```

これは、次のスラッシュ表記のシンタックスと同じです。

```
/person_mc:hairColor
```

関連項目

[/ \(スラッシュ\)](#)

++ (インクリメント)

利用状況

Flash Lite 1.0

シンタックス

```
++expression
```

```
expression++
```

オペランド

なし

説明

演算子 (算術)。**expression** に 1 を加えるプリインクリメント単項演算子またはポストインクリメント単項演算子です。式は、変数、配列のエレメント、またはオブジェクトのプロパティです。プリインクリメント形式の演算子 (**++expression**) は、**expression** に 1 を加算し、結果を数値として返します。ポストインクリメント形式の演算子 (**expression++**) は、**expression** に 1 を加算し、**expression** の初期値 (加算前の値) を返します。

例

次の例では、++ をポストインクリメント演算子として使用し、while ループを 5 回実行します。

```
i = 0;
while (i++ < 5){
    trace("this is execution " + i);
}
```

次の例では、++ をプリインクリメント演算子として使用します。

```
a = "";
i = 0;
while (i < 10) {
    a = a add (++i) add ",";
}
trace(a); // output: 1,2,3,4,5,6,7,8,9,10,
```

このスクリプトを実行すると、次の結果が出力パネルに表示されます。

```
1,2,3,4,5,6,7,8,9,10,
```

次の例では、++ をポストインクリメント演算子として使用します。

```
a = "";
i = 0;
while (i < 10) {
    a = a add (i++) add ",";
}
trace(a); // output: 0,1,2,3,4,5,6,7,8,9,
```

このスクリプトを実行すると、次の結果が出力パネルに表示されます。

```
0,1,2,3,4,5,6,7,8,9,
```

&& (論理積 (AND))

利用状況

Flash Lite 1.0

シンタックス

```
expression1 && expression2
```

オペランド

expression1, expression2 ブール値、またはブール値に変換される式。

説明

演算子 (論理)。ブール値演算を式のいずれか 1 つまたは両方の値で行います。**expression1** (演算子の左側の式) を評価し、その式の評価が **false** の場合は **false** を返します。**expression1** の評価が **true** であれば、**expression2** (演算子の右側の式) が評価されます。**expression2** の評価が **true** であれば、最終結果は **true** です。そうでなければ、**false** です。

例

次の例では、&& 演算子を使用してゲームの勝敗判定をテストしています。turns 変数と score 変数は、ゲームの回数または得点に応じて更新されます。次のスクリプトでは、3 回以内に 75 点以上を得点すると、出力パネルに "You Win the Game!" と表示されます。

```
turns = 2;  
score = 77;  
winner = (turns <= 3) && (score >= 75);  
if (winner) {  
    trace("You Win the Game!");  
} else {  
    trace("Try Again!");  
}
```

次の例では、虚の x 位置が次の範囲内にあるかどうかをテストしています。

```
xPos = 50;  
if (xPos >= 20 && xPos <= 80) {  
    trace ("the xPos is in between 20 and 80");  
}
```

!(論理否定 (NOT))

利用状況

Flash Lite 1.0

シンタックス

```
!expression
```

オペランド

なし

説明

演算子 (論理)。変数や式のブール値を反転します。**expression** が変数で、その絶対値または変換された値が **true** である場合、**!expression** の値は **false** です。式 **x && y** の評価結果が **false** の場合、式 **!(x && y)** の評価結果は **true** です。

次の式は、**!** 演算子を使用した結果を示します。

!true の場合は **false** が返されます。

!false の場合は **true** が返されます。

例

次の例では、変数 **happy** が **false** に設定されています。if ステートメントの条件部 **!happy** が評価され、その結果が **true** である場合、**trace()** ステートメントによりストリングが出力パネルに送られます。

```
happy = false;
if (!happy) {
    trace("don't worry, be happy");
}
```

|| (論理和 (OR))

利用状況

Flash Lite 1.0

シンタックス

```
expression1 || expression2
```

オペランド

expression1, expression2 ブール値、またはブール値に変換される式。

説明

演算子 (論理)。**expression1** と **expression2** を評価します。一方の式または両式の評価が **true** である場合、結果は **true** になります。両式の評価が **false** である場合のみ、結果は **false** になります。論理和 (OR) 演算子は任意の数のオペランドに適用できます。いずれかのオペランドの評価が **true** であれば、結果は **true** です。

非ブール値の式の場合、論理和 (OR) 演算子により左側の式が評価されます。これを **true** に変換できる場合は、結果が **true** になります。それ以外の場合は、右側の式が評価され、結果がその式の値になります。

例

シンタックス 1: 次の例では、if ステートメントで **||** 演算子を使用しています。2 番目の式の評価結果が **true** なので、最終結果は **true** になります。

```
theMinimum = 10;
theMaximum = 250;
start = false;
if (theMinimum > 25 || theMaximum > 200 || start){
    trace("the logical OR test passed");
}
```

% (剰余)

利用状況

Flash Lite 1.0

シンタックス

```
expression1 % expression2
```

オペランド

expression1, expression2 数値、または評価結果が数値になる式。

説明

演算子 (算術)。**expression1** を **expression2** で割ったときの剰余を計算します。**expression** のオペランドが非数値である場合、剰余演算子はこれを数値に変換しようとします。式は数値、または数値に変換される文字列です。

Flash Lite 1.0 か 1.1 をターゲットにする場合、Flash コンパイラは、次の式を使用してパブリッシュする SWF ファイルの % 演算子を拡張します。

```
expression1 - int(expression1/expression2) * expression2
```

この概算の実行は、剰余演算子をネイティブにサポートする Flash Player のバージョンほど高速または正確でない場合があります。

例

次のコードは、剰余 (%) 演算子を使用する数値の例を示します。

```
trace (12 % 5); // output: 2  
trace (4.3 % 2.1); // output: 0.0999...
```

%= (剰余代入)

利用状況

Flash Lite 1.0

シンタックス

```
expression1 %= expression2
```

オペランド

expression1, expression2 数値、または評価結果が数値になる式。

説明

演算子 (算術複合値の代入)。**expression1** に **expression1 % expression2** の値を代入します。たとえば、次の 2 つの式は同じです。

```
x %= y  
x = x % y
```

例

次の例では、値 4 を変数 x に代入します。

```
x = 14;  
y = 5;  
trace(x %= y); // output: 4
```

関連項目

[% \(剰余\)](#)

*= (乗算後代入)

利用状況

Flash Lite 1.0

シンタックス

```
expression1 *= expression2
```

オペランド

expression1, expression2 数値、または評価結果が数値になる式。

説明

演算子 (算術複合値の代入)。**expression1** に **expression1 * expression2** の値を代入します。

たとえば、次の 2 つの式は同じです。

```
x *= y  
x = x * y
```

例

シンタックス 1: 次の例では、値 50 を変数 x に代入します。

```
x = 5;  
y = 10;  
trace (x *= y); // output: 50
```

シンタックス 2: 次の例の 2 行目と 3 行目は、等号 (=) の右側の式を計算し、その結果を x と y にそれぞれ代入します。

```
i = 5;  
x = 4 - 6;  
y = i + 2;  
trace(x *= y); // output: -14
```

* (乗算)

利用状況

Flash Lite 1.0

シンタックス

```
expression1 * expression2
```

オペランド

expression1, expression2 数式。

説明

演算子 (算術)。2つの数式を乗算します。両方の式が整数であれば、その積は整数です。いずれかの式または両方の式が浮動小数点数であれば、その積は浮動小数点数になります。

例

シンタックス 1: 次のステートメントは、整数 2 と 3 を乗算します。

```
2 * 3
```

結果は、整数の 6 です。

シンタックス 2: 次のステートメントは、浮動小数点数 2.0 と 3.1416 を乗算します。

```
2.0 * 3.1416
```

結果は、浮動小数点数の 6.2832 です。

+ (数値式加算)

利用状況

Flash Lite 1.0

シンタックス

```
expression1 + expression2
```

オペランド

expression1, expression2 数値。

説明

演算子。数値式を加算します。+ は単なる数値演算子で、ストリング連結には使用できません。

両方の式が整数の場合、合計は整数になります。いずれかの式または両方の式が浮動小数点数の場合、合計は浮動小数点数になります。

例

次の例では、整数 2 と 3 を加算し、結果の整数 5 を出力パネルに表示します。

```
trace (2 + 3);
```

この例では、浮動小数点数 2.5 と 3.25 を加算し、結果の浮動小数点数 5.75 を出力パネルに表示します。

```
trace (2.5 + 3.25);
```

関連項目

[add \(ストリング連結\)](#)

== (数値の等価)

利用状況

Flash Lite 1.0

シンタックス

```
expression1 == expression2
```

オペランド

expression1, expression2 数値、ブール値、または変数。

説明

演算子 (比較)。等価性をテストします。<> 演算子の正反対です。**expression1** が **expression2** と等しい場合、結果は true です。<> 演算子の場合と同様に、等価の定義は比較するデータ型により異なります。

- 数値およびブール値は、値により比較されます。
- 変数は参照により比較されます。

例

次の例は、true と false の戻り値を示します。

```
trees = 7;
bushes = "7";
shrubs = "seven";

trace (trees == "7");// output: 1(true)
trace (trees == bushes);// output: 1(true)
trace (trees == shrubs);// output: 0(false)
```

関連項目

[eq \(ストリングの等価\)](#)

> (数値が対象より大きい)

利用状況

Flash Lite 1.0

シンタックス

```
expression1 > expression2
```

オペランド

expression1, expression2 数値、または評価結果が数値になる式。

説明

演算子 (比較)。2つの式を比較し、**expression1** が **expression2** より大きいかどうかを判定します。大きい場合は true を返します。**expression1** が **expression2** より小さいか等しい場合は、false を返します。

例

次の例では、数値の比較について true と false が返される場合を示します。

```
trace(3.14 > 2);// output: 1(true)
trace(1 > 2);// output: 0(false)
```

関連項目

[gt \(ストリングが対象より大きい\)](#)

>= (数値が対象より大きいか等しい)

利用状況

Flash Lite 1.0

シンタックス

```
expression1 >= expression2
```

オペランド

expression1, expression2 整数または浮動小数点数。

説明

演算子 (比較)。2つの式を比較し、**expression1** が **expression2** より大きいか等しいか (true)、または **expression1** が **expression2** より小さいか (false) を判定します。

例

次の例は、true と false の結果を示します。

```
trace(3.14 >= 2); // output: 1 (true)
trace(3.14 >= 4); // output: 0 (false)
```

関連項目

[ge \(スtringが対象より大きいか等しい \)](#)

<> (数値の不等価)

利用状況

Flash Lite 1.0

シンタックス

```
expression1 <> expression2
```

オペランド

expression1, expression2 数値、ブール値、または変数。

説明

演算子 (比較)。不等価性をテストします。等価 (==) 演算子の正反対です。**expression1** が **expression2** と等しい場合、結果は false です。等価 (==) 演算子の場合と同様に、等価の定義は比較するデータ型により異なります。

- 数値およびブール値は、値により比較されます。
- 変数は参照により比較されます。

例

次の例は、true と false が返される場合を示します。

```
trees = 7;  
B = "7";  
  
trace(trees <> 3); // output: 1(true)  
trace(trees <> B); // output: 0(false)
```

関連項目

[ne \(ストリングの不等価\)](#)

< (数値が対象より小さい)

利用状況

Flash Lite 1.0

シンタックス

```
expression1 < expression2
```

オペランド

expression1, expression2 数値。

説明

演算子 (比較)。2つの式を比較し、**expression1** が **expression2** より小さいかどうかを判定します。小さい場合は true を返します。**expression1** が **expression2** より大きいか等しい場合は、false を返します。< (より小さい) 演算子は数値演算子です。

例

次の例では、数値の比較とストリングの比較の両方について true と false が返される場合を示します。

```
trace (3 < 10); // output: 1(true)  
  
trace (10 < 3); // output: 0(false)
```

関連項目

[lt \(ストリングが対象より小さい\)](#)

<= (数値が対象より小さいか等しい)

利用状況

Flash Lite 1.0

シンタックス

```
expression1 <= expression2
```

オペランド

expression1, expression2 数値。

説明

演算子 (比較)。2つの式を比較し、**expression1** が **expression2** より小さいか等しいかどうかを判定します。小さいか等しい場合は **true** を返します。それ以外の場合は、**false** を返します。この演算子は、数値の比較専用です。

例

次の例では、数値の比較について **true** と **false** が返される場合を示します。

```
trace(5 <= 10); // output: 1(true)
trace(2 <= 2); // output: 1(true)
trace(10 <= 3); // output: 0 (false)
```

関連項目

[le \(ストリングが対象より小さいか等しい\)](#)

() (括弧)

利用状況

Flash Lite 1.0

シンタックス

```
(expression1 [, expression2])
(expression1, expression2)
```

expression1, **expression2** 数値、ストリング、変数、またはテキスト。

オペランド

parameter1、...、**parameterN**。これらのパラメータの実行結果がパラメータとして括弧の外側の関数に渡されます。

説明

演算子。パラメータをグループ化するか、複数の式を順番に評価します。または、パラメータを囲み、結果をパラメータとして括弧の外側にある関数に渡します。

シンタックス 1: 式内での演算子の実行順序を制御します。括弧は通常の優先順位を無効にし、括弧内の式を最初に評価します。括弧がネストされている場合は、最も内側の括弧から順に外側の括弧へと内容が評価されます。

シンタックス 2: コンマ区切りの一連の式を順番に評価し、最後に実行された式の結果を返します。

例

シンタックス 1: 次のステートメントは、括弧を使用して式の実行順序を制御しています (各式の値が出力パネルに表示されます)。

```
trace((2 + 3) * (4 + 5)); // displays 45
trace(2 + (3 * (4 + 5))); // // displays 29
trace(2 + (3 * 4) + 5); // displays 19
```

シンタックス 1: 次のステートメントは、括弧を使用して式の実行順序を制御しています (各式の値がログファイルに書き込まれます)。

```
trace((2 + 3) * (4 + 5)); // writes 45
trace(2 + (3 * (4 + 5))); // writes 29
trace(2 + (3 * 4) + 5); // writes 19
```

" " (ストリング区切り記号)

利用状況

Flash Lite 1.0

シンタックス

```
"text"
```

オペランド

text 0 個以上の文字。

説明

ストリング区切り記号。0 個以上の文字のシーケンスが引用符で囲まれている場合、リテラル値の文字を表し、変数や数値、その他の ActionScript エレメントではなく、ストリングと見なされます。

例

この例では、引用符を使用して、変数 `yourGuess` の値がリテラルストリング "Prince Edward Island" であり、変数名ではないことを示します。`province` の値は変数であり、リテラルではありません。`province` の値を決定するには、`yourGuess` の値を特定する必要があります。

```
yourGuess = "Prince Edward Island";

on(release){
    province = yourGuess;
    trace(province); // output: Prince Edward Island
}
```

eq (ストリングの等価)

利用状況

Flash Lite 1.0

シンタックス

```
expression1 eq expression2
```

オペランド

`expression1`, `expression2` 数値、ストリング、または変数。

説明

比較演算子。2 つの式が等しいかどうか比較し、`expression1` のストリング表現が `expression2` のストリング表現と等しい場合は **true** を返し、それ以外の場合は **false** を返します。

例

次の例は、**true** と **false** の結果を示します。

```
word = "persons";  
figure = "55";  
  
trace("persons" eq "people");// output: 0(false)  
trace("persons" eq word);// output: 1(true)  
trace(figure eq 50 + 5);// output: 1(true)  
trace(55.0 eq 55);// output: 1(true)
```

関連項目

[== \(数値の等価\)](#)

gt (ストリングが対象より大きい)

利用状況

Flash Lite 1.0

シンタックス

```
expression1 gt expression2
```

オペランド

expression1, expression2 数値、ストリング、または変数。

説明

演算子 (比較)。**expression1** のストリング表現を **expression2** のストリング表現と比較し、**expression1** が **expression2** より大きい場合は **true** を返します。それ以外の場合は **false** を返します。ストリングはアルファベット順に比較されます。数がすべての文字に先行し、すべての大文字は小文字に先行します。

例

次の例は、true と false の結果を示します。

```
animals = "cats";  
breeds = 7;  
  
trace ("persons" gt "people");// output: 1(true)  
trace ("cats" gt "cattle");// output: 0(false)  
trace (animals gt "cats");// output: 0(false)  
trace (animals gt "Cats");// output: 1(true)  
trace (breeds gt "5"); // output: 1(true)  
trace (breeds gt 7); // output: 0(false)
```

関連項目

[> \(数値が対象より大きい\)](#)

ge (ストリングが対象より大きいか等しい)

利用状況

Flash Lite 1.0

シンタックス

```
expression1 ge expression2
```

オペランド

expression1、**expression2** 数値、ストリング、または変数。

説明

演算子 (比較)。**expression1** のストリング表現を **expression2** のストリング表現と比較し、**expression1** が **expression2** より大きいか等しい場合は **true** を返します。それ以外の場合は **false** を返します。ストリングはアルファベット順に比較されます。数がすべての文字に先行し、すべての大文字は小文字に先行します。

例

次の例は、true と false の結果を示します。

```
animals = "cats";  
breeds = 7;  
  
trace ("cats" ge "cattle");// output: 0(false)  
trace (animals ge "cats");// output: 1(true)  
trace ("persons" ge "people");// output: 1(true)  
trace (animals ge "Cats");// output: 1(true)  
trace (breeds ge "5");// output: 1(true)  
trace (breeds ge 7); // output: 1(true)
```

関連項目

[>= \(数値が対象より大きいか等しい\)](#)

ne (ストリングの不等価)

利用状況

Flash Lite 1.0

シンタックス

```
expression1 ne expression2
```

オペランド

expression1、**expression2** 数値、ストリング、または変数。

説明

演算子 (比較)。**expression1** のストリング表現を **expression2** のストリング表現と比較し、**expression1** が **expression2** と等しくない場合は **true** を返します。それ以外の場合は **false** を返します。

例

次の例は、true と false の結果を示します。

```
word = "persons";  
figure = "55";  
  
trace ("persons" ne "people");           // output: 1(true)  
trace ("persons" ne word);               // output: 0(false)  
trace (figure ne 50 + 5);                 // output: 0(false)  
trace (55.0 ne 55);                       // output: 0(false)
```

関連項目

[<> \(数値の不等価\)](#)

lt (ストリングが対象より小さい)

利用状況

Flash Lite 1.0

シンタックス

```
expression1 lt expression2
```

オペランド

expression1、**expression2** 数値、ストリング、または変数。

説明

演算子 (比較)。**expression1** のストリング表現を **expression2** のストリング表現と比較し、**expression1** が **expression2** より小さい場合は **true** を返します。それ以外の場合は **false** を返します。ストリングはアルファベット順に比較されます。数がすべての文字に先行し、すべての大文字は小文字に先行します。

例

次の例は、さまざまなストリング比較の出力を示します。ストリングを整数と比較する場合、ActionScript 1.0 のシンタックスが整数データ型をストリングに変換しようとして、**false** が返されるので、最後の行に lt のエラーが返されていないことに注意してください。

```
animals = "cats";  
breeds = 7;  
  
trace ("persons" lt "people");// output: 0(false)  
trace ("cats" lt "cattle");// output: 1(true)  
trace (animals lt "cats");// output: 0(false)  
trace (animals lt "Cats");// output: 0(false)  
trace (breeds lt "5");       // output: 0(false)  
trace (breeds lt 7);         // output: 0(false)
```

関連項目

[< \(数値が対象より小さい\)](#)

le (ストリングが対象より小さいか等しい)

利用状況

Flash Lite 1.0

シンタックス

```
expression1 le expression2
```

オペランド

expression1, expression2 数値、ストリング、または変数。

説明

演算子 (比較)。**expression1** のストリング表現を **expression2** のストリング表現と比較し、**expression1** が **expression2** より小さいか等しい場合は **true** を返します。それ以外の場合は **false** を返します。ストリングはアルファベット順に比較されます。数がすべての文字に先行し、すべての大文字は小文字に先行します。

例

次の例は、さまざまなストリング比較の出力を示します。

```
animals = "cats";
breeds = 7;

trace ("persons" le "people");// output: 0(false)
trace ("cats" le "cattle");// output: 1(true)
trace (animals le "cats");// output: 1(true)
trace (animals le "Cats");// output: 0(false)
trace (breeds le "5");          // output: 0(false)
trace (breeds le 7);           // output: 1(true)
```

関連項目

[<= \(数値が対象より小さいか等しい\)](#)

－ (減算)

利用状況

Flash Lite 1.0

シンタックス

(Negation) *-expression*

(Subtraction) *expression1 - expression2*

オペランド

expression1, expression2 数値、または評価結果が数値になる式。

説明

演算子 (算術)。符号反転や減算に使用します。

シンタックス 1: 符号反転に使用する場合、数式の符号を逆にします。

シンタックス 2: 減算に使用する場合、2つの数式に対して算術的に減算し、**expression1** から **expression2** を減算します。両方の式が整数であれば、その差は整数です。いずれかの式または両方の式が浮動小数点数であれば、その差は浮動小数点数です。

例

シンタックス 1: 次のステートメントは、式 $2 + 3$ の符号を逆にします。

```
trace(-(2 + 3));  
// output: -5.
```

シンタックス 2: 次のステートメントは、整数 5 から整数 2 を減算します。

```
trace(5 - 2);  
// output: 3.
```

結果は、整数の 3 です。

シンタックス 3: 次のステートメントは、浮動小数点数 3.25 から浮動小数点数 1.5 を減算します。

```
trace(3.25 - 1.5);  
// output: 1.75.
```

結果は、浮動小数点数の 1.75 です。

-= (減算後代入)

利用状況

Flash Lite 1.0

シンタックス

```
expression1 -= expression2
```

オペランド

expression1, expression2 数値、または評価結果が数値になる式。

説明

演算子 (算術複合値の代入)。**expression1** に **expression1 - expression2** の値を代入します。返される値はありません。

たとえば、次の 2 つのステートメントは同じです。

```
x -= y;  
x = x - y;
```

ストリング式は、数値に変換する必要があります。そうでないと、-1 が返されます。

例

シンタックス 1: 次の例では、-= 演算子を使用して、5 から 10 を減算し、その結果を変数 x に代入します。

```
x = 2;  
y = 3;  
x -= y  
trace(x); // output: -1
```

シンタックス 2: 次の例では、ストリングがどのように数値に変換されるかを示します。

```
x = "2";  
y = "5";  
x -= y;  
trace(x); // output: -3
```

第6章：Flash Lite 固有の言語エレメント

この節では、アドビ システムズ社の Macromedia Flash Lite 1.1 ソフトウェアが認識するプラットフォーム機能および変数と、`fscommand()` および `fscommand2()` 関数を使用して実行可能な Flash Lite コマンドについて説明します。この節で説明される機能は、Flash Lite に固有のものであります。

次の表に、この節の内容を概説します。

言語エレメント	説明
<code>_capCompoundSound</code>	Flash Lite でコンパウンドサウンドを処理できるかどうかを示します。
<code>_capEmail</code>	Flash Lite クライアントが <code>GetURL()</code> ActionScript コマンドを使用して、電子メールメッセージを送信できるかどうかを示します。
<code>_capLoadData</code>	ホストアプリケーションが <code>loadMovie()</code> 、 <code>loadMovieNum()</code> 、 <code>loadVariables()</code> 、および <code>loadVariablesNum()</code> 関数の呼び出しによって、動的に追加のデータをロードできるかどうかを示します。
<code>_capMFi</code>	デバイスが MFi (Melody Format for i-mode) オーディオ形式でサウンドデータを再生できるかどうかを示します。
<code>_capMIDI</code>	デバイスが MIDI (Musical Instrument Digital Interface) オーディオ形式でサウンドデータを再生できるかどうかを示します。
<code>_capMMS</code>	Flash Lite が MMS (Multimedia Messaging Service) メッセージを <code>GetURL()</code> ActionScript コマンドを使用して送信できるかどうかを示します。
<code>_capMP3</code>	デバイスが MPEGAudio Layer 3 (MP3) オーディオ形式でサウンドデータを再生できるかどうかを示します。
<code>_capSMAF</code>	デバイスがマルチメディアファイルを SMAF (Synthetic music Mobile Application Format) で再生できるかどうかを示します。
<code>_capSMS</code>	Flash Lite が <code>GetURL()</code> ActionScript コマンドを使用して SMS (Short Message Service) メッセージを送信できるかどうかを示します。
<code>_capStreamSound</code>	デバイスがストリーミング (同期) サウンドを再生できるかどうかを示します。
<code>_cap4WayKeyAS</code>	右向き、左向き、上向き、下向き矢印キーに関連付けられたキーイベントハンドラに割り当てられた ActionScript の式を、Flash Lite が実行できるかどうかを示します。
<code>\$version</code>	Flash Lite のバージョン番号が含まれます。
<code>fscommand()</code>	Launch コマンドを実行するのに使用される関数 (次の項目を参照)。
<code>Launch</code>	<code>fscommand()</code> ではこのコマンドしかサポートされていません。Flash Lite、または携帯端末オペレーティングシステムやデバイスオペレーティングシステムなどのホスト環境と SWF ファイルを通信させることができます。
<code>fscommand2()</code>	この表のコマンド (<code>fscommand()</code> を除く) を実行するのに使用される関数。
<code>Escape</code>	任意の文字列をネットワーク転送に安全な形式にエンコードします。
<code>FullScreen</code>	レンダリングに使用する表示領域のサイズを設定します。
<code>GetBatteryLevel</code>	現在のバッテリーレベルを返します。
<code>GetDateDay</code>	現在の日付の日の部分を数値で返します。
<code>GetDateMonth</code>	現在の日付の月の部分を数値で返します。
<code>GetDateWeekday</code>	現在の日付の曜日を数値で返します。

言語エレメント	説明
GetDateYear	現在の日付の年を 4 桁の数値で返します。
GetDevice	Flash Lite が実行されているデバイスを識別するパラメータを設定します。
GetDeviceID	デバイスの一意的識別子 (シリアル番号など) を表すパラメータを設定します。
GetFreePlayerMemory	Flash Lite が現在使用できるヒープメモリの量をキロバイトで返します。
GetLanguage	デバイスによって現在使用されている言語を識別するパラメータを設定します。
GetLocaleLongDate	パラメータを、現在定義されているロケールに従ってフォーマットされた現在の日付を表すストリング (長い形式) に設定します。
GetLocaleShortDate	パラメータを、現在定義されているロケールに従ってフォーマットされた現在の日付を表すストリング (省略形) に設定します。
GetLocaleTime	パラメータを、現在定義されているロケールに従ってフォーマットされた現在の時刻を表すストリングに設定します。
GetMaxBatteryLevel	デバイスの最大バッテリーレベルを返します。
GetMaxSignalLevel	最大の信号の強さを返します。
GetMaxVolumeLevel	デバイスの最大音量レベルを数値で返します。
GetNetworkConnectStatus	現在のネットワーク接続ステータスを示す値を返します。
GetNetworkName	パラメータを現在のネットワーク名に設定します。
GetNetworkRequestStatus	最新の HTTP 要求のステータスを示す値を返します。
GetNetworkStatus	携帯端末のネットワークステータス (ネットワークが登録されているか、携帯端末が現在ローミング中かどうか) を示す値を返します。
GetPlatform	現在のプラットフォーム (広い意味ではデバイスのクラス) を識別するパラメータを設定します。オープンなオペレーティングシステムを使用しているデバイスの場合、通常はオペレーティングシステムの名前とバージョンが識別されます。
GetPowerSource	現在、バッテリーを電源としているか、外部電源を使用しているかを示す値を返します。
GetSignalLevel	現在の信号の強さを数値で返します。
GetTimeHours	24 時間形式に基づいて現在の時刻の時間を数値で返します。
GetTimeMinutes	現在の時刻の分を数値で返します。
GetTimeSeconds	現在の時刻の秒を数値で返します。
GetTimeZoneOffset	現地のタイムゾーンと世界時 (UTC) 間の分数にパラメータを設定します。
GetTotalPlayerMemory	Flash Lite に割り当てられたヒープメモリの合計量をキロバイトで返します。
GetVolumeLevel	デバイスの現在の音量レベルを数値で返します。
Quit	Flash Lite プレーヤーの再生を停止して終了します。
ResetSoftKeys	ソフトキーを元の設定にリセットします。
SetInputTextType	テキスト入力フィールドを開く際のモードを指定します。
SetQuality	アニメーションのレンダリング品質を設定します。
SetSoftKeys	デバイスにアクセスして再マッピングできる場合、デバイスの左および右のソフトキーを再マッピングします。

言語エレメント	説明
<code>StartVibrate</code>	携帯端末のバイブレータ機能を作動させます。
<code>StopVibrate</code>	バイブレータが動作している場合に停止します。
<code>Unescape</code>	ネットワーク転送に安全な形式でエンコードされた任意のストリングを標準のエンコードされていない形式にデコードします。

機能

この節では、Macromedia Flash Lite 1.1 が認識するプラットフォーム機能および変数について説明します。項目は、アルファベット順（先頭のアンダースコアを除く）に示されています。

`_capCompoundSound`

利用状況

Flash Lite 1.1

シンタックス

`_capCompoundSound`

説明

数値変数。Flash Lite がコンパウンドサウンドデータを処理できるかどうかを示します。できる場合は、この変数が定義され、値が 1 に設定されます。できない場合、この変数は定義されません。

たとえば、1 つの Flash ファイルに、同じサウンドを MIDI と MFi の両方の形式で入れることができます。Player は、デバイスがサポートする形式に基づいて適切な形式でデータを再生します。この変数は、Flash Lite プレーヤーが現在の携帯端末でこの機能サポートするかどうかを定義します。

次の例では、`useCompoundSound` が Flash Lite 1.1 では 1 に設定されていますが、Flash Lite 1.0 では未定義です。

```
useCompoundSound = _capCompoundSound;

if (useCompoundSound == 1) {
    gotoAndPlay("withSound");
} else {
    gotoAndPlay("withoutSound");
}
```

`_capEmail`

利用状況

Flash Lite 1.1

シンタックス

`_capEmail`

説明

数値変数。Flash Lite クライアントが GetURL() ActionScript コマンドを使用して電子メールメッセージを送信できるかどうかを示します。できる場合は、この変数が定義され、値が 1 に設定されます。できない場合、この変数は定義されません。

例

次の例では、ホストアプリケーションが GetURL() ActionScript コマンドを使用して電子メールメッセージを送信できる場合、canEmail を 1 に設定します。

```
canEmail = _capEmail;

if (canEmail == 1) {
    getURL("mailto:someone@somewhere.com?subject=foo&body=bar");
}
```

_capLoadData

利用状況

Flash Lite 1.1

シンタックス

_capLoadData

説明

数値変数。ホストアプリケーションが loadMovie()、loadMovieNum()、loadVariables()、および loadVariablesNum() 関数の呼び出しによって、動的に追加のデータをロードできるかどうかを示します。できる場合は、この変数が定義され、値が 1 に設定されます。できない場合、この変数は定義されません。

例

次の例では、ホストアプリケーションがムービーと変数を動的にロードできる場合、iCanLoad を 1 に設定します。

```
canLoad = _capLoadData;

if (canLoad == 1) {
    loadVariables("http://www.somewhere.com/myVars.php", GET);
} else {
    trace ("client does not support loading dynamic data");
}
```

_capMFi

利用状況

Flash Lite 1.1

シンタックス

_capMFi

説明

数値変数。デバイスが MFi (Melody Format for i-mode) オーディオ形式でサウンドデータを再生できるかどうかを示します。できる場合は、この変数が定義され、値が 1 に設定されます。できない場合、この変数は定義されません。

例

次の例では、デバイスが MFi サウンドデータを再生できる場合、canMfi を 1 に設定します。

```
canMfi = _capMfi;

if (canMfi == 1) {
    // send movieclip buttons to frame with buttons that trigger events sounds
    tellTarget("buttons") {
        gotoAndPlay(2);
    }
}
```

_capMIDI

利用状況

Flash Lite 1.1

シンタックス

_capMIDI

説明

数値変数。デバイスが MIDI (Musical Instrument Digital Interface) オーディオ形式でサウンドデータを再生できるかどうかを示します。できる場合は、この変数が定義され、値が 1 に設定されます。できない場合、この変数は定義されません。

例

次の例では、デバイスが MIDI サウンドデータを再生できる場合、canMidi を 1 に設定します。

```
canMIDI = _capMIDI;

if (canMIDI == 1) {
    // send movieclip buttons to frame with buttons that trigger events sounds
    tellTarget("buttons") {
        gotoAndPlay(2);
    }
}
```

_capMMS

利用状況

Flash Lite 1.1

シンタックス

_capMMS

説明

数値変数。Flash Lite が MMS (Multimedia Messaging Service) メッセージを GetURL() ActionScript コマンドを使用して送信できるかどうかを示します。できる場合は、この変数が定義され、値が 1 に設定されます。できない場合、この変数は定義されません。

例

次の例では、Flash Lite 1.1 の場合は canMMS は 1 に設定されますが、Flash Lite 1.0 の場合は未定義のままになります（ただし、すべての Flash Lite 1.1 の携帯端末が MMS メッセージを送信できるわけではないので、このコードは携帯端末に依存します）。

```
on(release) {
    canMMS = _capMMS;
    if (canMMS == 1) {
        // send an MMS
        myMessage = "mms:4156095555?body=sample mms message";
    } else {
        // send an SMS
        myMessage = "sms:4156095555?body=sample sms message";
    }
    getURL(myMessage);
}
```

__capMP3

利用状況

Flash Lite 1.1

シンタックス`__capMP3`**説明**

数値変数。デバイスが MPEGAudio Layer 3 (MP3) オーディオ形式でサウンドデータを再生できるかどうかを示します。できる場合は、この変数が定義され、値が 1 に設定されます。できない場合、この変数は定義されません。

例

次の例では、デバイスが MP3 サウンドデータを再生できる場合、canMP3 を 1 に設定します。

```
canMP3 = __capMP3;
if (canMP3 == 1) {
    tellTarget("soundClip") {
        gotoAndPlay(2);
    }
}
```

__capSMAF

利用状況

Flash Lite 1.1

シンタックス`__capSMAF`

説明

数値変数。デバイスが SMAF (Synthetic music Mobile Application Format) 形式のマルチメディアファイルを再生できるかどうかを示します。できる場合は、この変数が定義され、値が 1 に設定されます。できない場合、この変数は定義されません。

例

次の例では、Flash Lite 1.1 の場合は canSMAF は 1 に設定されますが、Flash Lite 1.0 の場合は未定義のままになります（ただし、すべての Flash Lite 1.1 の携帯端末が SMAF メッセージを送信できるわけではないので、このコードは携帯端末に依存します）。

```
canSMAF = _capSMAF;

if (canSMAF) {
    // send movieclip buttons to frame with buttons that trigger events sounds
    tellTarget("buttons") {
        gotoAndPlay(2);
    }
}
```

_capSMS

利用状況

Flash Lite 1.1

シンタックス

_capSMS

説明

数値変数。Flash Lite が GetURL() ActionScript コマンドを使用して SMS (**Short Message Service**) メッセージを送信できるかどうかを示します。できる場合は、この変数が定義され、値が 1 に設定されます。できない場合、この変数は定義されません。

例

次の例では、Flash Lite 1.1 の場合は canSMS は 1 に設定されますが、Flash Lite 1.0 の場合は未定義のままになります（ただし、すべての Flash Lite 1.1 の携帯端末が SMS メッセージを送信できるわけではないので、このコードは携帯端末に依存します）。

```
on(release) {
    canSMS = _capSMS;
    if (canSMS) {
        // send an SMS
        myMessage = "sms:4156095555?body=sample sms message";
        getURL(myMessage);
    }
}
```

_capStreamSound

利用状況

Flash Lite 1.1

シンタックス

`_capStreamSound`

説明

数値変数。デバイスがストリーミング (同期) サウンドを再生できるかどうかを示します。できる場合は、この変数が定義され、値が 1 に設定されます。できない場合、この変数は定義されません。

例

次の例では、`canStreamSound` が有効な場合にストリーミングサウンドを再生します。

```
on(press) {
    canStreamSound = _capStreamSound;
    if (canStreamSound) {
        // play a streaming sound in a movieclip with this button
        tellTarget("music") {
            gotoAndPlay(2);
        }
    }
}
```

`_cap4WayKeyAS`

利用状況

Flash Lite 1.1

シンタックス

`_cap4WayKeyAS`

説明

数値変数。右向き、左向き、上向き、下向き矢印キーに関連付けられたキーイベントハンドラに割り当てられた ActionScript の式を、Flash Lite が実行できるかどうかを示します。ホストアプリケーションが 4 方向のキーナビゲーションモードを使用して Flash コントロール (ボタンおよびテキスト入力フィールド) 間を移動する場合のみ、この変数が定義され、値が 1 に設定されます。それ以外の場合、この変数は定義されません。

この変数の値が 1 の場合、4 方向のキーのいずれかが押されると、Flash Lite は最初にそのキーのハンドラを探します。ハンドラが見つからない場合、Flash コントロールナビゲーションが発生します。しかし、イベントハンドラが見つかった場合は、そのキーのナビゲーションアクションは発生しません。たとえば、下向き矢印キーの `keypress` ハンドラが見つかった場合、ユーザーは移動できません。

例

次の例では、`canUse4Way` が Flash Lite 1.1 では 1 に設定されていますが、Flash Lite 1.0 では未定義です (ただし、Flash Lite 1.1 を使用したすべての携帯端末が 4 方向キーをサポートするわけではないため、このコードは携帯端末に依存します)。

```
canUse4Way = _cap4WayKeyAS;
if (canUse4Way == 1) {
    msg = "Use your directional joystick to navigate this application";
} else {
    msg = "Please use the 2 key to scroll up, the 6 key to scroll right, the 8 key to scroll down, and the 4 key to scroll left.";
}
```

\$version

利用状況

Flash Lite 1.1

シンタックス

```
$version
```

説明

ストリング変数。Flash Lite のバージョン番号が含まれます。この変数には、メジャー番号、マイナー番号、ビルド番号、および内部ビルド番号が格納されます。リリースバージョンの場合、内部ビルド番号は通常は 0 です。

すべての Flash Lite 1.x 製品のメジャー番号は 5 です。Flash Lite 1.0 はマイナー番号 1 を持ち、Flash Lite 1.1 はマイナー番号 2 を持ちます。

例

Flash Lite 1.1 プレーヤーでは、次のコードによって myVersion の値が "5, 2, 12, 0" に設定されます。

```
myVersion = $version;
```

fscommand()

利用状況

Flash Lite 1.1

シンタックス

```
status = fscommand("Launch", "application-path, arg1, arg2,..., argn")
```

パラメータ

"Launch" コマンド指定子。fscommand() 関数の実行に使用できるコマンドは、Launch コマンドだけです。

"application-path, arg1, arg2,..., argn" 開始されるアプリケーションの名前とそのパラメータ。コンマで区切ります。

説明

関数。Flash Lite、または携帯端末オペレーティングシステムやデバイスオペレーティングシステムなどのホスト環境と SWF ファイルを通信できるようにします。

関連項目

[fscommand2\(\)](#)

Launch

利用状況

Flash Lite 1.1

シンタックス

```
status = fscommand("Launch", "application-path, arg1, arg2,..., argn")
```

パラメータ

"Launch" コマンド指定子。Flash Lite では、Launch コマンドを実行するためだけに fscommand() 関数を使用します。

"**application-path, arg1, arg2,..., argn**" 開始されるアプリケーションの名前とそのパラメータ。コンマで区切ります。

説明

fscommand() 関数により実行されるコマンド。デバイスで別のアプリケーションを起動します。起動されるアプリケーションの名前とそのパラメータが 1 つの引数として渡されます。

注意: この機能は、オペレーティングシステムに依存しています。

このコマンドは、Flash Lite プレーヤーがスタンドアロンモードで実行されている場合にのみサポートされています。Flash Lite プレーヤーが他のアプリケーションのコンテキストで実行されている場合 (ブラウザのプラグインとして実行されている場合など)、このコマンドは使用できません。

例

次の例では、Series 60 携帯端末のサービス /Web ブラウザで wap.yahoo.com を開きます。

```
on(keyPress "9") {  
    status = fscommand("launch", "z:\\system\\apps\\browser\\browser.app,http://wap.yahoo.com");  
}
```

関連項目

[fscommand2\(\)](#)

fscommand2()

利用状況

Flash Lite 1.1

シンタックス

```
returnValue = fscommand2(command [, expression1 ... expressionN])
```

パラメータ

command ホストアプリケーションに任意の用途で渡されるストリング、または Flash Lite プレーヤーに渡されるコマンド。

parameter1...parameterN **command** によって指定されたコマンドにパラメータとして渡されるストリングのコンマ区切りリスト。

説明

関数。Flash Lite、または携帯端末オペレーティングシステムやデバイスオペレーティングシステムなどのホスト環境と SWF ファイルを通信できるようにします。fscommand2() によって返される値は、特定のコマンドにより異なります。

fscommand2() 関数は fscommand() 関数と似ていますが、次のような違いがあります。

- fscommand2() 関数は、任意の数の引数を取得します。
- fscommand2() はすぐに実行されますが、fscommand() は処理中のフレームの最後に実行されます。

- `fscommand2()` 関数は、成功または失敗の報告に使用できる値、またはコマンドの結果を返します。
コマンドおよびパラメータとして関数に渡す文字列と式については、この節の表を参照してください。
表には、次の 3 つの列があります。
- コマンド列は、コマンドを識別する文字列リテラルパラメータを示します。
- パラメータ列では、追加のパラメータとして渡す値の種類を説明します。
- 戻り値列では、予測される戻り値について説明します。

例

この節の残りの部分では、`fscommand2()` 関数を使用して実行する特定のコマンドの例について説明します。

関連項目

[fscommand\(\)](#)

Escape

利用状況

Flash Lite 1.1

説明

任意の文字列をネットワーク転送に安全な形式にエンコードします。英数字以外の各文字を 16 進エスケープシーケンス (%**xx**、またはマルチバイト文字の場合は %**xx%xx**) で置き換えます。

コマンド	パラメータ	戻り値
"Escape"	<p><i>original URL</i> として安全な形式にエンコードされた文字列。</p> <p><i>encoded</i> 結果としてエンコードされた文字列。</p> <p>これらのパラメータは変数の名前または定数文字列値です (たとえば、"Encoded_String")。</p>	<p>0: 失敗。</p> <p>1: 成功。</p>

例

次の例では、エンコードされた形式へのサンプル文字列の変換を示します。

```
original_string = "Hello, how are you?";
status = fscommand2("escape", original_string, "encoded_string");
trace (encoded_string); // output: Hello%2C%20how%20are%20you%3F
```

関連項目

[Unescape](#)

FullScreen

利用状況

Flash Lite 1.1

説明

レンダリングに使用する表示領域のサイズを設定します。サイズは、全画面表示にすることも、全画面表示より小さくすることもできます。

このコマンドは、Flash Lite がスタンドアロンモードで実行されている場合のみサポートされています。Flash Lite プレーヤーが他のアプリケーションのコンテキストで実行されている場合（ブラウザのプラグインとして実行されている場合など）、このコマンドは使用できません。

コマンド	パラメータ	戻り値
"FullScreen"	<i>size</i> true（全画面表示）または false（全画面表示より小さい）のいずれかの値を持つ定義済みの変数が定数ストリング値です。これ以外の値は false 値と見なされます。	-1: サポートされない。 0: サポートされる。

例

次の例では、表示領域を全画面表示に設定します。戻り値が 0 以外の場合は、再生ヘッドを `smallScreenMode` というフレームに送信します。

```
status = fscommand2("FullScreen", true);  
if(status != 0) {  
    gotoAndPlay("smallScreenMode");  
}
```

GetBatteryLevel

利用状況

Flash Lite 1.1

説明

現在のバッテリーレベルを返します。返される値は、0 から `GetMaxBatteryLevel` が返す最大値の間の数値です。

コマンド	パラメータ	戻り値
"GetBatteryLevel"	なし	-1: サポートされない。 その他の数値: 現在のバッテリー容量。

例

次の例では、`battLevel` 変数を現在のバッテリーの容量に設定します。

```
battLevel = fscommand2("GetBatteryLevel");
```

関連項目

[GetMaxBatteryLevel](#)

GetDateDay

利用状況

Flash Lite 1.1

説明

現在の日付の日の部分を返します。返される値は数値です（先頭に 0 なし）。指定できる値は、1 ~ 31 です。

コマンド	パラメータ	戻り値
"getDateDay"	なし	-1: サポートされない。 1 ~ 31: その月の日付。

例

次の例では、日付情報を収集し、完全な日付ストリングを作成します。

```
today = fscommand2("getDateDay");  
weekday = fscommand2("getDateWeekday");  
thisMonth = fscommand2("getDateMonth");  
thisYear = fscommand2("getDateYear");  
when = weekday add ", " add ThisMonth add " " add today add ", " add thisYear;
```

関連項目

[getDateMonth](#)、[getDateWeekday](#)、[getDateYear](#)

getDateMonth

利用状況

Flash Lite 1.1

説明

現在の日付の月を数値で返します（先頭に 0 なし）。

コマンド	パラメータ	戻り値
"getDateMonth"	なし	-1: サポートされない。 1 ~ 12: 現在の月の数字。

例

次の例では、日付情報を収集し、完全な日付ストリングを作成します。

```
today = fscommand2("getDateDay");  
weekday = fscommand2("getDateWeekday");  
thisMonth = fscommand2("getDateMonth");  
thisYear = fscommand2("getDateYear");  
when = weekday add ", " add thisMonth add " " add today add ", " add thisYear;
```

関連項目

[getDateDay](#)、[getDateWeekday](#)、[getDateYear](#)

getDateWeekday

利用状況

Flash Lite 1.1

説明

現在の日付の曜日を数値で返します。

コマンド	パラメータ	戻り値
"getDateWeekday"	なし	-1: サポートされない。 0: 日曜日。 1: 月曜日。 2: 火曜日。 3: 水曜日。 4: 木曜日。 5: 金曜日。 6: 土曜日。

例

次の例では、日付情報を収集し、完全な日付ストリングを作成します。

```
today = fscommand2("getDateDay");  
weekday = fscommand2("getDateWeekday");  
thisMonth = fscommand2("getDateMonth");  
thisYear = fscommand2("getDateYear");  
when = weekday add ", " add thisMonth add " " add today add ", " add thisYear;
```

関連項目

[getDateDay](#)、[getDateMonth](#)、[getDateYear](#)

getDateYear

利用状況

Flash Lite 1.1

説明

現在の日付の年を 4 桁の数値で返します。

コマンド	パラメータ	戻り値
"getDateYear"	なし	-1: サポートされない。 0 ~ 9999: 現在の年

例

次の例では、日付情報を収集し、完全な日付ストリングを作成します。

```
today = fscommand2("getDateDay");  
weekday = fscommand2("getDateWeekday");  
thisMonth = fscommand2("getDateMonth");  
thisYear = fscommand2("getDateYear");  
when = weekday add ", " add thisMonth add " " add today add ", " add thisYear;
```

関連項目

[getDateDay](#)、[getDateMonth](#)、[getDateWeekday](#)

GetDevice

利用状況

Flash Lite 1.1

説明

Flash Lite が実行されているデバイスを識別するパラメータを設定します。この識別子は、通常、モデル名です。

コマンド	パラメータ	戻り値
"GetDevice"	<i>device</i> デバイスの識別子を受け取るストリング。これは変数の名前、または変数の名前が含まれるストリング値です。	-1: サポートされない。 0: サポートされる。

例

次のコード例では、デバイス識別子を `statusdevice` 変数に代入し、汎用デバイス名を使用してテキストフィールドを更新します。

次に結果のサンプルとそれが表すデバイスを示します。

D506i Mitsubishi 506i 携帯端末。

DFOMA1 Mitsubishi FOMA1 携帯端末。

F506i Fujitsu 506i 携帯端末。

FFOMA1 Fujitsu FOMA1 携帯端末。

N506i NEC 506i 携帯端末。

NFOMA1 NEC FOMA1 携帯端末。

Nokia3650 Nokia 3650 携帯端末。

p506i Panasonic 携帯端末。

PFOMA1 Panasonic FOMA1 携帯端末。

SH506i Sharp 506i 携帯端末。

SHFOMA1 Sharp FOMA1 携帯端末。

SO506i Sony 506i 携帯端末。

```
statusdevice = fscommand2("GetDevice", "devicename");
switch(devicename) {
    case "D506i":
        /:myText += "device: Mitsubishi 506i" add newline;
        break;
    case "DFOMA1":
        /:myText += "device: Mitsubishi FOMA1" add newline;
        break;
    case "F506i":
        /:myText += "device: Fujitsu 506i" add newline;
        break;
    case "FFOMA1":
        /:myText += "device: Fujitsu FOMA1" add newline;
        break;
    case "N506i":
        /:myText += "device: NEC 506i" add newline;
        break;
    case "NFOMA1":
        /:myText += "device: NEC FOMA1" add newline;
        break;
    case "Nokia 3650":
        /:myText += "device: Nokia 3650" add newline;
        break;
    case "P506i":
        /:myText += "device: Panasonic 506i" add newline;
        break;
    case "PFOMA1":
        /:myText += "device: Panasonic FOMA1" add newline;
        break;
    case "SH506i":
        /:myText += "device: Sharp 506i" add newline;
        break;
    case "SHFOMA1":
        /:myText += "device: Sharp FOMA1" add newline;
        break;
    case "SO506i":
        /:myText += "device: Sony 506i" add newline;
        break;
}
```

GetDeviceID

利用状況

Flash Lite 1.1

説明

デバイスの一意の識別子 (シリアル番号など) を表すパラメータを設定します。

コマンド	パラメータ	戻り値
"GetDeviceID"	<i>id</i> デバイスの一意の識別子を受け取る文字列。これは変数の名前、または変数の名前が含まれる文字列値です。	-1: サポートされない。 0: サポートされる。

例

次の例では、一意の識別子を変数 `deviceID` に代入します。

```
status = fscommand2("GetDeviceID", "deviceID");
```

GetFreePlayerMemory

利用状況

Flash Lite 1.1

説明

Flash Lite が現在使用できるヒープメモリの量をキロバイトで返します。

コマンド	パラメータ	戻り値
"GetFreePlayerMemory"	なし	-1: サポートされない。 0 または 正の値: 使用可能なヒープメモリのキロバイト数。

例

次の例では、ステータスを使用可能なメモリと等しい量に設定します。

```
status = fscommand2("GetFreePlayerMemory");
```

関連項目

[GetTotalPlayerMemory](#)

GetLanguage

利用状況

Flash Lite 1.1

説明

デバイスによって現在使用されている言語を識別するパラメータを設定します。言語は、名前で渡された変数で文字列として返されます。

コマンド	パラメータ	戻り値
"GetLanguage"	<p><i>language</i> 言語コードを受け取る文字列。これは変数の名前、または変数の名前が含まれる文字列値です。戻り値は、次のいずれかの値になります。</p> <p>cs : チェコ語。 da : デンマーク語。 de : ドイツ語。 en-UK : 英語 (英国) または国際英語。 en-US : 英語 (米国)。 es : スペイン語。 fi : フィンランド語。 fr : フランス語。 hu : ハンガリー語。 it : イタリア語。 ja : 日本語。 ko : 韓国語。 nl : オランダ語。 no : ノルウェー語。 pl : ポーランド語。 pt : ポルトガル語。 ru : ロシア語。 sv : スウェーデン語。 tr : トルコ語。 xu : 判別できない言語。 zh-CN : 簡体字中国語。 zh-TW : 繁体字中国語。</p>	<p>-1 : サポートされない。 0 : サポートされる。</p>

注意 : 日本語の携帯端末が英語を表示するように設定されている場合、*language* に対して *en_US* が返されます。

例

次のコード例では、言語コードを *language* 変数に代入し、Flash Lite プレーヤーによって認識される言語を使用してテキストフィールドを更新します。

```
statuslanguage = fscommand2("GetLanguage", "language");
switch(language) {
  case "cs":
    /:myText += "language is Czech" add newline;
    break;
  case "da":
    /:myText += "language is Danish" add newline;
    break;
  case "de":
    /:myText += "language is German" add newline;
    break;
  case "en-UK":
    /:myText += "language is UK" add newline;
    break;
  case "en-US":
    /:myText += "language is US" add newline;
    break;
  case "es":
    /:myText += "language is Spanish" add newline;
    break;
  case "fi":
    /:myText += "language is Finnish" add newline;
    break;
  case "fr":
    /:myText += "language is French" add newline;
    break;
  case "hu":
    /:myText += "language is Hungarian" add newline;
    break;
  case "it":
    /:myText += "language is Italian" add newline;
    break;
  case "jp":
    /:myText += "language is Japanese" add newline;
    break;
  case "ko":
    /:myText += "language is Korean" add newline;
    break;
  case "nl":
    /:myText += "language is Dutch" add newline;
    break;
  case "no":
    /:myText += "language is Norwegian" add newline;
    break;
  case "pl":
    /:myText += "language is Polish" add newline;
    break;
  case "pt":
    /:myText += "language is Portuguese" add newline;
```

```

        break;
    case "ru":
        /:myText += "language is Russian" add newline;
        break;
    case "sv":
        /:myText += "language is Swedish" add newline;
        break;
    case "tr":
        /:myText += "language is Turkish" add newline;
        break;
    case "xu":
        /:myText += "language is indeterminable" add newline;
        break;
    case "zh-CN":
        /:myText += "language is simplified Chinese" add newline;
        break;
    case "zh-TW":
        /:myText += "language is traditional Chinese" add newline;
        break;
}

```

GetLocaleLongDate

利用状況

Flash Lite 1.1

説明

パラメータを、現在定義されているロケールに従ってフォーマットされた現在の日付を表す文字列（長い形式）に設定します。

コマンド	パラメータ	戻り値
"GetLocaleLongDate"	<p><i>longdate</i> "October 16, 2004" や "16 October 2004" などの現在の日付の値（長い形式）を受け取る文字列変数。</p> <p>これは変数の名前、または変数の名前が含まれる文字列値です。</p> <p><i>longdate</i> で返される値は、複数文字から成る可変長の文字列です。実際のフォーマットはデバイスとロケールによって異なります。</p>	<p>-1: サポートされない。</p> <p>0: サポートされる。</p>

例

次の例では、*longDate* 変数で長い形式の現在の日付を返そうとします。また、*status* の値を設定し、実行できたかどうかを報告します。

```

status = fscommand2("GetLocaleLongDate", "longdate");
trace (longdate); // output: Tuesday, June 14, 2005

```

関連項目

[GetLocaleShortDate](#)、[GetLocaleTime](#)

GetLocaleShortDate

利用状況

Flash Lite 1.1

説明

パラメータを、現在定義されているロケールに従ってフォーマットされた現在の日付を表す文字列（省略形）に設定します。

コマンド	パラメータ	戻り値
"GetLocaleShortDate"	<p><i>shortdate</i> "10/16/2004" や "16-10-2004" のように短い形式の現在の日付の値を受け取る文字列変数。</p> <p>これは変数の名前、または変数の名前が含まれる文字列値です。</p> <p><i>shortdate</i> の戻り値は、複数文字から成る可変長の文字列です。実際のフォーマットはデバイスとロケールによって異なります。</p>	<p>-1: サポートされない。</p> <p>0: サポートされる。</p>

例

次の例では、*shortDate* 変数で短い形式の現在の日付を返そうとします。また、*status* の値を設定し、実行できたかどうかを報告します。

```
status = fscommand2("GetLocaleShortDate", "shortdate");  
trace (shortdate); // output: 06/14/05
```

関連項目

[GetLocaleLongDate](#)、[GetLocaleTime](#)

GetLocaleTime

利用状況

Flash Lite 1.1

説明

パラメータを、現在定義されているロケールに従ってフォーマットされた現在の時刻を表す文字列に設定します。

コマンド	パラメータ	戻り値
"GetLocaleTime"	<p><i>time</i> "6:10:44 PM" や "18:10:44" のような現在の時刻の値を受け取る文字列変数。</p> <p>これは変数の名前、または変数の名前が含まれる文字列値です。</p> <p><i>time</i> の戻り値は、複数文字から成る可変長の文字列です。実際のフォーマットはデバイスとロケールによって異なります。</p>	<p>-1: サポートされない。</p> <p>0: サポートされる。</p>

例

次の例では、現在のローカル時刻を *time* 変数に返そうとします。また、*status* の値を設定し、実行できたかどうかを報告します。

```
status = fscommand2("GetLocaleTime", "time");  
trace (time); // output: 14:30:21
```

関連項目

[getLocaleLongDate](#)、[getLocaleShortDate](#)

GetMaxBatteryLevel

利用状況

Flash Lite 1.1

説明

デバイスの最大バッテリーレベルを返します。返される値は、正の数値です。

コマンド	パラメータ	戻り値
"GetMaxBatteryLevel"	なし	-1: サポートされない。 その他の値: 最大バッテリー容量。

例

次の例では、変数 maxBatt を最大バッテリーレベルに設定します。

```
maxBatt = fscommand2("GetMaxBatteryLevel");
```

GetMaxSignalLevel

利用状況

Flash Lite 1.1

説明

最大の信号の強さを返します。返される値は、正の数値です。

コマンド	パラメータ	戻り値
"GetMaxSignalLevel"	なし	-1: サポートされない。 その他の数値: 最大信号レベル

例

次の例では、最大信号強さを変数 sigStrengthMax に代入します。

```
sigStrengthMax = fscommand2("GetMaxSignalLevel");
```

GetMaxVolumeLevel

利用状況

Flash Lite 1.1

説明

デバイスの最大音量レベルを数値で返します。

コマンド	パラメータ	戻り値
"GetMaxVolumeLevel"	なし	-1: サポートされない。 その他の値: 最大音量レベル

例

次の例では、変数 `maxvolume` をデバイスの最大音量レベルに設定します。

```
maxvolume = fscommand2("GetMaxVolumeLevel");  
trace (maxvolume); // output: 80
```

関連項目

[GetVolumeLevel](#)

GetNetworkConnectStatus

利用状況

Flash Lite 1.1

説明

現在のネットワーク接続ステータスを示す値を返します。

コマンド	パラメータ	戻り値
"GetNetworkConnectStatus"	なし	-1: サポートされない。 0: 現在アクティブなネットワーク接続がある。 1: デバイスがネットワークへの接続を試行中。 2: 現在アクティブなネットワーク接続がない。 3: ネットワーク接続が中断している。 4: ネットワーク接続を確認できない。

例

次の例では、ネットワーク接続ステータスを変数 `connectstatus` に代入し、`switch` ステートメントを使用して接続のステータスでテキストフィールドを更新します。

```

connectstatus = fscommand2("GetNetworkConnectStatus");
switch (connectstatus) {
    case -1 :
        /:myText += "connectstatus not supported" add newline;
        break;
    case 0 :
        /:myText += "connectstatus shows active connection" add newline;
        break;
    case 1 :
        /:myText += "connectstatus shows attempting connection" add newline;
        break;
    case 2 :
        /:myText += "connectstatus shows no connection" add newline;
        break;
    case 3 :
        /:myText += "connectstatus shows suspended connection" add newline;
        break;
    case 4 :
        /:myText += "connectstatus shows indeterminable state" add newline;
        break;
}

```

GetNetworkName

利用状況

Flash Lite 1.1

説明

パラメータを現在のネットワーク名に設定します。

コマンド	パラメータ	戻り値
"GetNetworkName"	<p><i>networkName</i> ネットワーク名を表すストリング。これは変数の名前、または変数の名前が含まれるストリング値です。</p> <p>ネットワークが登録され、ネットワーク名が確認できる場合、<i>networkname</i> はネットワーク名に設定されます。確認できない場合は空のストリングに設定されます。</p>	<p>-1: サポートされない。</p> <p>0: ネットワークが登録されていない。</p> <p>1: ネットワークが登録されているが、ネットワーク名がわからない。</p> <p>2: ネットワークが登録されていて、ネットワーク名がわかる。</p>

例

次の例では、現在のネットワークの名前を変数 `myNetName` に代入し、ステータス値を変数 `netNameStatus` に代入します。

```
netNameStatus = fscommand2("GetNetworkName", myNetName);
```

GetNetworkRequestStatus

利用状況

Flash Lite 1.1

説明

最新の HTTP 要求のステータスを示す値を返します。

コマンド	パラメータ	戻り値
"GetNetworkRequestStatus"	なし	-1: コマンドはサポートされない。 0: 保留中の要求があり、ネットワーク接続が確立され、サーバーのホスト名が解決されており、サーバーに接続されている。 1: 保留中の要求があり、ネットワーク接続を確立中。 2: 保留中の要求があるが、ネットワーク接続がまだ確立されていない。 3: 保留中の要求があり、ネットワーク接続が確立され、サーバーのホスト名を解決中。 4: ネットワークエラーにより要求が失敗した。 5: サーバーへの接続エラーにより要求が失敗した。 6: サーバーから HTTP エラー (404 など) が返された。 7: DNS サーバーにアクセスできないかサーバー名を解決できないために要求が失敗した。 8: 要求が正常に完了した。 9: タイムアウトにより要求が失敗した。 10: まだ要求が発行されていない。

例

次の例では、最新の HTTP 要求のステータスを変数 `requeststatus` に代入し、`switch` ステートメントを使用してステータスでテキストフィールドを更新します。

```
requeststatus = fscommand2("GetNetworkRequestStatus");
switch (requeststatus) {
    case -1:
        /:myText += "requeststatus not supported" add newline;
        break;
    case 0:
        /:myText += "connection to server has been made" add newline;
        break;
    case 1:
        /:myText += "connection is being established" add newline;
        break;
    case 2:
        /:myText += "pending request, contacting network" add newline;
        break;
    case 3:
        /:myText += "pending request, resolving domain" add newline;
        break;
    case 4:
        /:myText += "failed, network error" add newline;
        break;
    case 5:
        /:myText += "failed, couldn't reach server" add newline;
        break;
    case 6:
        /:myText += "HTTP error" add newline;
        break;
    case 7:
        /:myText += "DNS failure" add newline;
        break;
    case 8:
        /:myText += "request has been fulfilled" add newline;
        break;
    case 9:
        /:myText += "request timedout" add newline;
        break;
    case 10:
        /:myText += "no HTTP request has been made" add newline;
        break;
}
```

GetNetworkStatus

利用状況

Flash Lite 1.1

説明

携帯端末のネットワークステータス (ネットワークが登録されているか、携帯端末が現在ローミング中かどうか) を示す値を返します。

コマンド	パラメータ	戻り値
"GetNetworkStatus"	なし	-1: コマンドはサポートされない。 0: ネットワークが登録されていない。 1: ホームネットワークに接続されている。 2: 拡張ホームネットワークに接続されている。 3: ローミング中 (ホームネットワーク以外に接続)。

例

次の例では、ネットワーク接続のステータスを変数 `networkstatus` に代入し、`switch` ステートメントを使用してステータスでテキストフィールドを更新します。

```
networkstatus = fscommand2("GetNetworkStatus");
switch(networkstatus) {
    case -1:
        /:myText += "network status not supported" add newline;
        break;
    case 0:
        /:myText += "no network registered" add newline;
        break;
    case 1:
        /:myText += "on home network" add newline;
        break;
    case 2:
        /:myText += "on extended home network" add newline;
        break;
    case 3:
        /:myText += "roaming" add newline;
        break;
}
```

GetPlatform

利用状況

Flash Lite 1.1

説明

現在のプラットフォーム (広い意味ではデバイスのクラス) を識別するパラメータを設定します。オープンなオペレーティングシステムを使用しているデバイスの場合、通常はオペレーティングシステムの名前とバージョンが識別されます。

コマンド	パラメータ	戻り値
"GetPlatform"	<i>platform</i> プラットフォームの識別子を受け取る文字列。これは変数の名前、または変数の名前が含まれる文字列値です。	-1: サポートされない。 0: サポートされる。

例

次のコード例では、プラットフォーム識別子を `statusplatform` 変数に代入し、汎用プラットフォーム名を使用してテキストフィールドを更新します。

次に `myPlatform` の結果のサンプルとそれが表すデバイスのクラスを示します。

506i 506i 携帯端末。

FOMA1 FOMA1 携帯端末。

Symbian6.1_s60.1 Symbian 6.1、Series 60 バージョン 1 携帯端末。

Symbian7.0 Symbian 7.0 携帯端末。

```
statusplatform = fscommand2("GetPlatform", "platform");
switch(platform) {
    case "506i":
        /:myText += "platform: 506i" add newline;
        break;
    case "FOMA1":
        /:myText += "platform: FOMA1" add newline;
        break;
    case "Symbian6.1-Series60v1":
        /:myText += "platform: Symbian6.1, Series 60 version 1 phone" add newline;
        break;
    case "Symbian7.0":
        /:myText += "platform: Symbian 7.0" add newline;
        break;
}
```

GetPowerSource

利用状況

Flash Lite 1.1

説明

現在、バッテリーを電源としているか、外部電源を使用しているかを示す値を返します。

コマンド	パラメータ	戻り値
"GetPowerSource"	なし	-1: サポートされない。 0: デバイスはバッテリー電源で動作している。 1: デバイスは外部電源で動作している。

例

次の例では、電源を示すように変数 myPower を設定します。設定できない場合は、-1 に設定します。

```
myPower = fscommand2("GetPowerSource");
```

GetSignalLevel

利用状況

Flash Lite 1.1

説明

現在の信号の強さを数値で返します。

コマンド	パラメータ	戻り値
"GetSignalLevel"	なし	-1: サポートされない。 その他の数値: 現在の信号レベル (0 から GetMaxSignalLevel が返す最大値の範囲)。

例

次の例では、信号レベルの値を変数 sigLevel に代入します。

```
sigLevel = fscommand2("GetSignalLevel");
```

GetTimeHours

利用状況

Flash Lite 1.1

説明

24 時間形式に基づいて現在の時刻の時間を返します。返される値は数値です (先頭に 0 なし)。

コマンド	パラメータ	戻り値
"GetTimeHours"	なし	-1: サポートされない。 0 ~ 23: 現在の時間。

例

次の例では、hour 変数を現在の時刻の時間の部分、または -1 に設定します。

```
hour = fscommand2("GetTimeHours");  
trace(hour); // output: 14
```

関連項目

[GetTimeMinutes](#)、[GetTimeSeconds](#)、[GetTimeZoneOffset](#)

GetTimeMinutes

利用状況

Flash Lite 1.1

説明

現在の時刻の分部分を返します。返される値は数値です (先頭に 0 なし)。

コマンド	パラメータ	戻り値
"GetTimeMinutes"	なし	-1: サポートされない。 0 ~ 59: 現在の分。

例

次の例では、minutes 変数を現在の時刻の分の部分、または -1 に設定します。

```
minutes = fscommand2("GetTimeMinutes");  
trace (minutes); // output: 38
```

関連項目

[GetTimeHours](#)、[GetTimeSeconds](#)、[GetTimeZoneOffset](#)

GetTimeSeconds

利用状況

Flash Lite 1.1

説明

現在の時刻の秒部分を返します。返される値は数値です（先頭に 0 なし）。

コマンド	パラメータ	戻り値
"GetTimeSeconds"	なし	-1: サポートされない。 0 ~ 59: 現在の秒。

例

次の例では、seconds 変数を現在の時刻の秒の部分、または -1 に設定します。

```
seconds = fscommand2("GetTimeSeconds");  
trace (seconds); // output: 41
```

関連項目

[GetTimeHours](#)、[GetTimeMinutes](#)、[GetTimeZoneOffset](#)

GetTimeZoneOffset

利用状況

Flash Lite 1.1

説明

現地のタイムゾーンと世界時（UTC）間の分数にパラメータを設定します。

コマンド	パラメータ	戻り値
"GetTimeZoneOffset"	<i>timezoneOffset</i> 現地のタイムゾーンと UTC の間の分数。これは変数の名前、または変数の名前が含まれる文字列値です。 次のような正または負の数値が返されます。 540: 日本標準時 -420: 太平洋標準時の夏時間	-1: サポートされない。 0: サポートされる。

例

次の例では、UTC からのオフセットの分数を *timezoneoffset* 変数に代入して *status* を 0 に設定するか、*status* を -1 に設定します。

```
status = fscommand2("GetTimeZoneOffset", "timezoneoffset");  
trace (timezoneoffset);           // output: 300
```

関連項目

[GetTimeHours](#)、[GetTimeMinutes](#)、[GetTimeSeconds](#)

GetTotalPlayerMemory

利用状況

Flash Lite 1.1

説明

Flash Lite に割り当てられたヒープメモリの合計量をキロバイトで返します。

コマンド	パラメータ	戻り値
"GetTotalPlayerMemory"	なし	-1: サポートされない。 0 または正の値: ヒープメモリの総量 (キロバイト単位)。

例

次の例では、status 変数をヒープメモリの総量に設定します。

```
status = fscommand2("GetTotalPlayerMemory");
```

関連項目

[GetFreePlayerMemory](#)

GetVolumeLevel

利用状況

Flash Lite 1.1

説明

デバイスの現在の音量レベルを数値で返します。

コマンド	パラメータ	戻り値
"GetVolumeLevel"	なし	-1: サポートされない。 その他の数値: 現在の音量レベル (0 から fscommand2("GetMaxVolumeLevel") が返す値の範囲)。

例

次の例では、現在の音量レベルを変数 volume に代入します。

```
volume = fscommand2("GetVolumeLevel");  
trace (volume);           // output: 50
```

Quit

利用状況

Flash Lite 1.1

説明

Flash Lite プレーヤーの再生を停止して終了します。

このコマンドは、Flash Lite がスタンドアロンモードで実行されている場合のみサポートされています。Flash Lite プレーヤーが他のアプリケーションのコンテキストで実行されている場合（ブラウザのプラグインとして実行されている場合など）、このコマンドは使用できません。

コマンド	パラメータ	戻り値
"Quit"	なし	-1: サポートされない。

例

次の例では、スタンドアロンモードで実行中に Flash Lite の再生を停止して終了します。

```
status = fscommand2("Quit");
```

ResetSoftKeys

利用状況

Flash Lite 1.1

説明

ソフトキーを元の設定にリセットします。

このコマンドは、Flash Lite がスタンドアロンモードで実行されている場合のみサポートされています。Flash Lite プレーヤーが他のアプリケーションのコンテキストで実行されている場合（ブラウザのプラグインとして実行されている場合など）、このコマンドは使用できません。

コマンド	パラメータ	戻り値
"ResetSoftKeys"	なし	-1: サポートされない。 0: サポートされる。

例

次のステートメントは、ソフトキーを元の設定にリセットします。

```
status = fscommand2("ResetSoftKeys");
```

関連項目

[SetSoftKeys](#)

SetInputTextType

利用状況

Flash Lite 1.1

説明

テキスト入力フィールドを開く際のモードを指定します。

コマンド	パラメータ	戻り値
"SetInputTextType"	<i>variableName</i> テキスト入力フィールドの名前。これは変数の名前、または変数の名前が含まれる文字列値です。 <i>type</i> Numeric、Alpha、Alphanumeric、Latin、NonLatin、または NoRestriction のいずれかの値。	0 : 失敗。 1 : 成功。

Flash Lite では、ホストアプリケーションに汎用のデバイス依存テキスト入力インターフェイスを開始するように要求することで、入力テキスト機能をサポートします。これは通常 FEP (**Front-End Processor** : 前置プロセッサ) と呼ばれます。SetInputTextType コマンドを使用しない場合は、FEP がデフォルトモードで開きます。

下の表に、各モードによる影響と置換されるモードを示します。

指定されるモード	FEP をこれらの相互に排他的なモードのいずれかに設定する	現在のデバイスでサポートされていない場合は、このモードで FEP を開く
Numeric	数値のみ (0 ~ 9)	Alphanumeric
Alpha	アルファベット文字のみ (A ~ Z、a ~ z)	Alphanumeric
Alphanumeric	英数字のみ (0 ~ 9、A ~ Z、a ~ z)	Latin
Latin	ラテン文字のみ (英数字と句読記号)	NoRestriction
NonLatin	非ラテン文字のみ (漢字とかななど)	NoRestriction
NoRestriction	デフォルトのモード (FEP に制限を設定しない)	

注意 : すべての携帯端末でこれらのテキスト入力フィールドの種類がサポートされているとは限らないので、入力されるテキストデータは検証する必要があります。

例

次のコードは、数値データを受け取るために、変数 input1 に関連付けられたテキスト入力フィールドの種類を設定します。

```
status = fscommand2("SetInputTextType", "input1", "Numeric");
```

SetQuality

利用状況

Flash Lite 1.1

説明

アニメーションのレンダリング品質を設定します。

コマンド	パラメータ	戻り値
"SetQuality"	<i>quality</i> レンダリング品質。"high"、"medium"、または "low" に設定する必要があります。	-1: サポートされない。 0: サポートされる。

例

次の例では、レンダリング品質を low に設定します。

```
status = fscommand2("SetQuality", "low");
```

SetSoftKeys

利用状況

Flash Lite 1.1

説明

デバイスにアクセスして再マッピングできる場合、デバイスの左および右のソフトキーを再マッピングします。

コマンドの実行後、左のキーを押すと PageUp キー押下イベントが発生し、右のキーを押すと PageDown キー押下イベントが発生します。それぞれ対応するキーを押すと、PageUp および PageDown キー押下イベントに関連付けられている ActionScript が実行されます。

このコマンドは、Flash Lite がスタンドアロンモードで実行されている場合のみサポートされています。Flash Lite プレーヤーが他のアプリケーションのコンテキストで実行されている場合（ブラウザのプラグインとして実行されている場合など）、このコマンドは使用できません。

コマンド	パラメータ	戻り値
"SetSoftKeys"	<i>left</i> Left ソフトキーに表示するテキスト。 <i>right</i> Right ソフトキーに表示するテキスト。 これらのパラメータは変数の名前または定数ストリング値です（たとえば、"Previous"）。	-1: サポートされない。 0: サポートされる。

例

次の例では、Left ソフトキーに Previous、Right ソフトキーに Next と表示します。

```
status = fscommand2("SetSoftKeys", "Previous", "Next");
```

関連項目

[ResetSoftKeys](#)

StartVibrate

利用状況

Flash Lite 1.1

説明

携帯端末のバイブレータ機能を作動させます。バイブレータが既に動作している場合は、Flash Lite は現在の動作を停止してから、新しく指定されたバイブレータ動作を開始します。Flash アプリケーションの再生を停止または一時停止したとき、および Flash Lite プレーヤーを終了したときにもバイブレータは停止します。

コマンド	パラメータ	戻り値
"StartVibrate"	<p><i>time_on</i> バイブレータがオンになっているミリ秒単位の期間 (最大 5 秒)。</p> <p><i>time_off</i> バイブレータがオフになっているミリ秒単位の期間 (最大 5 秒)。</p> <p><i>repeat</i> この動作を繰り返す回数 (最大 3 回)。</p>	<p>-1 : サポートされない。</p> <p>0 : バイブレーションが開始した。</p> <p>1 : エラーが発生してバイブレーションを開始できなかった。</p>

例

次の例では、2.5 秒間オン、1 秒間オフのバイブレーションシーケンスを開始して 2 回繰り返します。そして、成否を示す変数 *status* に値を代入します。

```
status = fscommand2("StartVibrate", 2500, 1000, 2);
```

関連項目

[StopVibrate](#)

StopVibrate

利用状況

Flash Lite 1.1

説明

バイブレータが動作している場合に停止します。

コマンド	パラメータ	戻り値
"StopVibrate"	なし	<p>-1 : サポートされない。</p> <p>0 : バイブレーションが停止した。</p>

例

次の例では、StopVibrate を呼び出して、結果 (サポートされない、またはバイブレーションが停止) を *status* 変数に格納します。

```
status = fscommand2("StopVibrate");
```

関連項目

[StartVibrate](#)

Unescape

利用状況

Flash Lite 1.1

説明

ネットワーク転送に安全な形式でエンコードされた任意の文字列を標準のエンコードされていない形式にデコードします。16進形式のすべての文字、つまり2桁の16進数の後に続くパーセント記号(%)は、デコードされた形式に変換されます。

コマンド	パラメータ	戻り値
"Unescape"	<i>original</i> URLとして安全な形式から標準の形式にデコードされる文字列。 <i>decoded</i> 結果のデコードされた文字列 (このパラメータは変数の名前、または変数の名前が含まれる文字列値です)。	0: 失敗。 1: 成功。

例

次の例では、エンコードされた文字列のデコードを示します。

```
encoded_string = "Hello%2C%20how%20are%20you%3F";  
status = fscommand2("unescape", encoded_string, "normal_string");  
trace (normal_string); // output: Hello, how are you?
```

関連項目

[Escape](#)

索引

記号

_alpha 変数 31
 _cap4WayKeyAS 変数 83
 _capCompoundSound 変数 78
 _capEmail 変数 78
 _capLoadData 変数 79
 _capMFi 変数 79
 _capMIDI 変数 80
 _capMMS 変数 80
 _capSMAF 変数 81
 _capSMSx0d x0d 変数 82
 _capStreamSoundx0d x0d 変数 82
 _currentframe プロパティ 32
 _focusrect プロパティ 32
 _framesloaded プロパティ 33
 _height プロパティ 33
 _highquality プロパティ 34
 _level プロパティ 34
 _name プロパティ 35
 _rotation プロパティ 36
 _scroll プロパティ 36
 _target プロパティ 37
 _visible プロパティ 37
 _width プロパティ 38
 _x プロパティ 38
 _xscale プロパティ 39
 _y プロパティ 39
 _yscale プロパティ 40
 ,(コンマ) 演算子 55
 !(論理否定 (NOT)) 演算子 61
 ?(条件) 演算子 57
 .(ドット) 演算子 59
 ""(ストリング区切り記号) 演算子 70
 *(乗算) 演算子 64
 *(乗算後代入) 演算子 64
 /(除算) 演算子 58
 /(スラッシュ - ルートタイムライン) プロパティ 31
 /*(コメントのブロック) 演算子 55
 //(コメント) 演算子 56
 /(除算) 演算子 59
 \ 68, 68
 \ (数値の不等価) 演算子 67
 &&(論理積 (AND)) 演算子 61
 ||(論理和 (OR)) 演算子 62
 %(剰余) 演算子 63
 %=(剰余代入) 演算子 63

+(数値式加算) 演算子 65
 ++(インクリメント) 演算子 60
 +=(加算後代入) 演算子 53
 -=(減算後代入演算子) 演算子 75
 =(代入) 演算子 54
 ==(数値の等価) 演算子 65
 >(より大きい) 演算子 66
 >(より大きいか等しい) 演算子 67
 \$version 変数 84

A

add(ストリング連結) 演算子 53
 _alpha 変数 31
 AND 演算子 54, 61

B

break ステートメント 41

C

call 3
 case ステートメント 42
 chr() 関数 4
 comments
 one-line 56
 continue ステートメント 43
 _currentframe プロパティ 32

D

do..while ステートメント 44
 duplicateMovieClip() 関数 4

E

else if ステートメント 46
 else ステートメント 45
 eq(ストリングの等価) 演算子 70
 eval() 関数 5

F

_focusrect プロパティ 32
 for ステートメント 46
 for ループ 46

_framesloaded プロパティ 33
fscommand() コマンド 84

G

ge (ストリングが対象より大きいか等しい) 演算子 71
getProperty() 関数 6
getTimer() 関数 7
getURL() 関数 7
gotoAndPlay() 関数 9
gotoAndStop() 関数 10
gt (ストリングが対象より大きい) 演算子 71

H

_height プロパティ 33
_highquality プロパティ 34

I

if ステートメント 47
iffFrameLoaded() 関数 10
int() 関数 11

L

le (ストリングが対象より小さいか等しい) 演算子 73
length() 関数 12
_level プロパティ 34
loadMovie() 関数 12
loadMovieNum() 関数 13
loadVariables() 関数 14
loadVariablesNum() 関数 15
lt (ストリングが対象より小さい) 演算子 73

M

maxscroll プロパティ 35
mbchr() 関数 16
mbsubstring() 関数 17
MFI サウンド 79
MIDI サウンド 80
MMS メッセージング 80

N

_name プロパティ 35
ne (ストリングの不等価) 演算子 72
nextFrame() 関数 18
nextScene() 関数 18
NOT 演算子 61
Number() 関数 19

O

on() 関数 19
OR 演算子 62
ord() 関数 20

P

play() 関数 21
prevFrame() 関数 21
prevScene() 関数 22

R

random() 関数 22
removeMovieClip() 関数 23
_rotation プロパティ 36

S

scroll プロパティ 36
set() 関数 23
setProperty() 関数 24
statements
 logical NOT 61
stop() 関数 25
stopAllSounds() 関数 25
String() 関数 26
substring() 関数 26
switch ステートメント 48

T

_target プロパティ 37
tellTarget() 関数 27
toggleHighQuality() 関数 27
_totalframes プロパティ 37
trace() 関数 28

U

unloadMovie() 関数 28
unloadMovieNum() 関数 29

V

_visible プロパティ 37

W

while ステートメント 49
while ループ 44
_widthx11 プロパティ 38

X

_x プロパティ 38
xd0 (減算) 演算子 74
xd0 xd0 (デクリメント) 演算子 57
_xscale プロパティ 39

Y

_y プロパティ 39
_yscale プロパティ 40

Z

演算子

加算 54
加算代入 53
減算後代入 75
除算 58
除算後代入 59
乗算 64
剰余 63
剰余代入 63
条件 57
数値式加算 65
数値が対象より小さい 68
数値が対象より小さいか等しい 68
数値の等価 65
数値の不等価 67
代入 54
論理積 61
論理否定 61
論理和 62
インクリメント 60
コメント 56
コメントのブロック 55
コンマ 55
文字列区切り記号 70
文字列連結 53
文字列が対象より小さい 73
文字列が対象より小さいか等しい 73
文字列が対象より大きい 71
文字列が対象より大きいか等しい 71
文字列の等価 70
文字列の不等価 72
ドット 59
より大きい 66
より大きいか等しい 67
加算後代入演算子 53
減算後代入演算子 75
除算 58
除算後代入演算子 59

乗算 64
剰余演算子 63
剰余代入 63
条件 47
条件演算子 57
数値式加算 65
代入演算子 54
不等価演算子 67
連結 53
論理積 (AND) 演算子 61
論理否定 (NOT) 演算子 61
論理和 (OR) 演算子 62

い

インクリメント演算子 60

か

functions
loadMovie() 12
関数
chr() 4
duplicateMovieClip() 4
eval() 5
fscommand() 84
getProperty() 6
getTimer() 7
getURL() 7
gotoAndPlay() 9
gotoAndStop() 10
ifFrameLoaded() 10
int() 11
length() 12
loadMovieNum() 13
loadVariables() 14
loadVariablesNum() 15
mbchr() 16
mbsubstring() 17
nextFrame() 18
nextScene() 18
Number() 19
on() 19
ord() 20
play() 21
prevFrame() 21
prevScene() 22
random() 22
removeMovieClip() 23
set() 23
setProperty() 24

stop() 25
stopAllSounds() 25
String() 26
substring() 26
tellTarget() 27
toggleHighQuality() 27
trace() 28
unloadMovie() 28
unloadMovieNum() 29

こ

コメント
 ブロック 55
コメントのブロック演算子 55
コンマ演算子 55

さ

サウンド変数 78, 79, 80, 81, 82

す

ステートメント
 break 41
 case 42
 continue 43
 do..while 44
 else 45
 else if 46
 for 46
 if 47
 switch 48
 while 49
 文字列区切り記号演算子 70
 文字列が対象より小さいか等しい 73
 文字列が対象より大きい演算子 71
 文字列が対象より大きい等しい 71
 文字列の等価演算子 70

て

電子メール機能変数 78

と

ドット演算子 59

ふ

プロパティ
 _alpha 31
 _currentframe 32

 _focusrect 32
 _framesloaded 33
 _height 33
 _highquality 34
 _level 34
 _name 35
 _rotation 36
 _scroll 36
 _target 37
 _visible 37
 _width 38
 _x 38
 _xscale 39
 _y 39
 _yscale 40
 maxscroll 35
 scroll 36
 スラッシュ 31

へ

変数
 _alpha 31
 _cap4WayKeyAS 83
 _capCompoundSound 78
 _capEmail 78
 _capLoadData 79
 _capMFi 79
 _capMIDI 80
 _capMMS 80
 _capSMAF 81
 _capSMS 82
 _capStreamSound 82
 \$version 84
 Flash Lite のバージョン番号 84
 データをロードする機能 79
 電子メール機能 78
 矢印キーのナビゲーション 83
変数、サウンド
 _capCompoundSound 78
 _capMFi 79
 _capMIDI 80
 _capSMAF 81
 _capStreamSound 82
変数、メッセージング
 _capMMS 80
 _capSMS 82

め

メッセージング変数 80, 82

よ

より小さい演算子 68

より小さいか等しい演算子 68

より大きい演算子 66

より大きいか等しい演算子 67

る

ルートタイムラインの識別子 31