

GUIDA DI RIFERIMENTO DI ACTIONSCRIPT™ PER FLASH® LITE™ 1.x

© 2007 Adobe Systems Incorporated. Tutti i diritti riservati.

Guida di riferimento di ActionScript™ per Adobe® Flash® Lite™ 1.x

Se la presente guida viene distribuita con software che include un accordo di licenza per l'utente finale, la guida e il software in essa descritto sono concessi in licenza e possono essere usati e copiati solo in conformità con i termini di tale licenza. Ad eccezione di quanto eventualmente concesso da tale licenza, nessuna parte di questa guida può essere riprodotta, memorizzata in un sistema per il recupero dati o trasmessa in qualsiasi forma o con qualsiasi mezzo, elettronico, meccanico, di registrazione o altro, senza il previo consenso scritto da parte di Adobe Systems Incorporated. Il contenuto di questa guida è protetto dalle leggi sui diritti d'autore, anche se non distribuito con software corredato di accordo di licenza per l'utente finale.

Il contenuto di questa guida viene fornito unicamente a scopo informativo, è soggetto a modifiche senza preavviso e non comporta alcun impegno per Adobe Systems Incorporated. Adobe Systems Incorporated declina ogni responsabilità per eventuali errori o imprecisioni presenti in questa guida.

Se si inseriscono in un progetto grafica e immagini esistenti, si tenga presente che tali materiali potrebbero essere protetti dalla legge sul copyright. L'inserimento non autorizzato di tali materiali nel proprio lavoro potrebbe rappresentare una violazione dei diritti del titolare del copyright. Assicurarsi sempre di ottenere le eventuali autorizzazioni necessarie dal titolare dei diritti d'autore.

Tutti i riferimenti a nomi di società negli esempi forniti hanno scopo puramente dimostrativo e non intendono fare riferimento ad alcuna organizzazione realmente esistente.

Adobe, il logo Adobe, Flash Lite e Flash sono marchi o marchi registrati di Adobe Systems Incorporated negli Stati Uniti e/o in altri Paesi.

Informazioni su terze parti

Questo manuale contiene collegamenti a siti Web di terze parti che non sono sotto il controllo di Adobe Systems Incorporated. Adobe Systems Incorporated non potrà quindi essere ritenuta responsabile per il contenuto di qualsiasi sito collegato. Qualora si decida di accedere a un sito Web di terze parti menzionato nel presente documento, lo si farà sotto la propria completa responsabilità e a proprio rischio. Adobe Systems Incorporated fornisce questi collegamenti solo per comodità dell'utente e l'inclusione del collegamento non implica che Adobe Systems Incorporated sottoscriva o accetti qualsiasi responsabilità per il contenuto di tali siti di terze parti.



Tecnologia per la compressione e la decompressione video Sorenson™ Spark™, concessa in licenza da Sorenson Media, Inc.

Fraunhofer-IIS/Thomson Multimedia: tecnologia di compressione audio MPEG Layer-3 concessa in licenza da Fraunhofer IIS e Thomson Multimedia (<http://www.iis.fhg.de/amm/>).

Independent JPEG Group: questo software è parzialmente basato sul lavoro di Independent JPEG Group.

Nellymoser, Inc.: tecnologia per la compressione e la decompressione vocale concessa in licenza da Nellymoser, Inc. (<http://www.nelly-moser.com>).

Browser Opera ® Copyright © 1995-2002 di Opera Software ASA e dei suoi fornitori. Tutti i diritti riservati.

Macromedia Flash 8 video è alimentato mediante tecnologia video On2 TrueMotion. © 1992-2005 On2 Technologies, Inc. Tutti i diritti riservati. <http://www.on2.com>.

Visual SourceSafe è un marchio registrato o un marchio commerciale di Microsoft Corporation negli Stati Uniti e/o in altri Paesi.

Informazioni aggiornate e informazioni aggiuntive sul codice di terze parti sono disponibili all'indirizzo http://www.adobe.com/go/thirdparty_it/.

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

Avviso agli utenti finali di enti governativi degli Stati Uniti d'America: il Software e la Documentazione sono "Commercial Items" (Prodotti commerciali) secondo la definizione contenuta nell'articolo 48 C.F.R. §2.101, costituiti da "Commercial Computer Software" (Software commerciale per Computer) e "Commercial Computer Software Documentation" (Documentazione relativa a software commerciale per Computer) secondo la definizione contenuta nell'articolo 48 C.F.R. §12.212 o 48 C.F.R. §227.7202, secondo i casi. In conformità con l'articolo 48 C.F.R. §12.212 o con gli articoli da 48 C.F.R. §§227.7202-1 a 227.7202-4 incluso, secondo i casi, i "Commercial Computer Software" e "Commercial Computer Software Documentation" vengono concessi in licenza agli utenti appartenenti al Governo degli Stati Uniti d'America (a) esclusivamente come "Commercial Items" e (b) con i soli diritti concessi a tutti gli altri utenti finali ai termini e alle condizioni qui contenuti. Tutti i diritti non pubblicati riservati, ai sensi della legge sul diritto d'autore vigente negli Stati Uniti d'America. Adobe Systems Incorporated, 345 Park Avenue, San Jose, CA 95110-2704, USA. Nei confronti degli utenti finali del Governo degli Stati Uniti, Adobe accetta di rispettare tutte le leggi applicabili sul diritto alle pari opportunità, comprese, ove applicabili, le direttive dell'Executive Order 11246, seconda revisione, la sezione 402 del "Vietnam Era Veterans Readjustment Assistance Act" del 1974 (38 USC 4212) e la sezione 503 del "Rehabilitation Act" del 1973, seconda revisione, oltre ai regolamenti esposti in 41 CFR da 60-1 a 60-60, 60-250 e 60-741. La clausola di azione affermativa e i regolamenti sopra elencati saranno incorporati tramite riferimento.

Indice

Introduzione	11
Voci di esempio per la maggior parte degli elementi di ActionScript	11
Cartella Samples	12
Convenzioni tipografiche	12
Capitolo 1: Funzioni globali di Flash Lite	13
call()	15
chr()	16
duplicateMovieClip()	17
eval()	18
getProperty()	19
getTimer()	20
getURL()	20
gotoAndPlay()	23
gotoAndStop()	24
ifframeLoaded()	25
int()	26
length()	26
loadMovie()	27
loadMovieNum()	28
loadVariables()	30
loadVariablesNum()	31
mbchr()	32
mblength()	33
mbord()	34
mbsubstring()	35
nextFrame()	35
nextScene()	36
Number()	37
on()	38
ord()	39
play()	39
prevFrame()	40
prevScene()	41
random()	41

removeMovieClip()	42
set()	43
setProperty()	44
stop()	45
stopAllSounds()	45
String()	46
substring()	47
tellTarget()	47
toggleHighQuality()	48
trace()	49
unloadMovie()	50
unloadMovieNum()	51
Capitolo 2: Proprietà di Flash Lite	53
/ (Barra)	54
_alpha	55
_currentframe	56
_focusrect	56
_framesloaded	57
_height	58
_highquality	58
_level	59
maxscroll	60
_name	61
_rotation	61
scroll	62
_target	62
_totalframes	63
_visible	63
_width	64
_x	64
_xscale	65
_y	66
_yscale	67
Capitolo 3: Istruzioni di Flash Lite	69
break	70
case	71
continue	72
do..while	74
else	75
else if	76
for	77
if	78

switch	78
while	80
Capitolo 4: Operatori di Flash Lite	83
add (concatenazione stringhe)	86
+= (assegnazione addizione)	87
and	88
= (assegnazione)	89
/* (commento a blocchi)	89
, (virgola)	90
// (commento)	91
?: (condizionale)	92
-- (decremento)	93
/ (divisione)	94
/= (assegnazione divisione)	94
. (punto)	95
++ (incremento)	96
&& (AND logico)	97
! (NOT logico)	98
(OR logico)	99
% (modulo)	100
%= (assegnazione modulo)	100
*= (assegnazione moltiplicazione)	101
* (moltiplicazione)	102
+ (addizione numerica)	103
== (uguaglianza numerica)	104
> (maggiore di numerico)	105
>= (maggiore di o uguale a numerico)	105
<> (disuguaglianza numerica)	106
< (minore di numerico)	107
<= (minore di o uguale a numerico)	107
() (parentesi)	108
" " (delimitatore di stringa)	109
eq (uguaglianza stringhe)	110
gt (stringa maggiore di)	110
ge (stringa maggiore di o uguale a)	111
ne (disuguaglianza di stringa)	112
lt (stringa minore di)	113
le (stringa minore di o uguale a)	114
- (sottrazione)	115
-= (assegnazione sottrazione)	116

Capitolo 5: Elementi di linguaggio specifici di Flash Lite117

Funzionalità	121
_capCompoundSound	121
_capEmail	122
_capLoadData	122
_capMFi	123
_capMIDI	124
_capMMS	124
_capMP3	125
_capSMAF	126
_capSMS	126
_capStreamSound	127
_cap4WayKeyAS	128
\$version	129
fscommand()	129
Launch	130
fscommand2()	131
Escape	132
FullScreen	133
GetBatteryLevel	134
GetDateDay	134
GetDateMonth	135
GetDateWeekday	136
GetDateYear	136
GetDevice	137
GetDeviceID	139
GetFreePlayerMemory	139
GetLanguage	140
GetLocaleLongDate	143
GetLocaleShortDate	144
GetLocaleTime	145
GetMaxBatteryLevel	145
GetMaxSignalLevel	146
GetMaxVolumeLevel	146
GetNetworkConnectStatus	147
GetNetworkName	149
GetNetworkRequestStatus	150
GetNetworkStatus	152
GetPlatform	153
GetPowerSource	154
GetSignalLevel	154
GetTimeHours	155
GetTimeMinutes	155
GetTimeSeconds	156
GetTimeZoneOffset	157

GetTotalPlayerMemory	158
GetVolumeLevel	158
Quit	159
ResetSoftKeys.....	159
SetInputTextType	160
SetQuality	161
SetSoftKeys.....	162
StartVibrate	163
StopVibrate	163
Unescape	164
Indice analitico	165

Introduzione

Questo manuale descrive la sintassi e l'uso degli elementi di ActionScript per lo sviluppo di applicazioni per il software Macromedia® Flash® Lite™ 1.0 e Macromedia® Flash® Lite™ 1.1 di Adobe, definito globalmente come Flash Lite 1.x. ActionScript per Flash Lite 1.x si basa sulla versione di ActionScript utilizzata in Macromedia® Flash® 4 di Adobe. Per utilizzare gli esempi in uno script, copiare il codice dell'esempio da questo manuale e incollarlo nel riquadro Script o in un file di script esterno. Nel manuale sono elencati tutti gli elementi di ActionScript: operatori, parole chiave, istruzioni, comandi, proprietà, funzioni, classi e metodi.

Voci di esempio per la maggior parte degli elementi di ActionScript

Le seguenti voci di esempio illustrano le convenzioni utilizzate per tutti gli elementi di ActionScript.

Titolo della voce

Le voci sono elencate in ordine alfabetico all'interno di un capitolo, senza tenere conto dell'uso delle maiuscole, dei caratteri di sottolineatura iniziali e così via.

Disponibilità

Se non diversamente specificato, in questa sezione viene indicata la versione di Flash Lite che supporta l'elemento preso in esame.

Uso

In questa sezione viene fornita la sintassi corretta per l'uso degli elementi di ActionScript nel codice. La porzione di sintassi richiesta è indicata in *carattere codice*. Sia il codice inserito dall'utente che le informazioni sui tipi di dati sono indicati in *carattere codice corsivo*. I tipi di dati sono distinguibili dal codice inserito dall'utente dai due punti (:) che precedono sempre le informazioni sui tipi di dati. I parametri opzionali sono racchiusi tra parentesi quadre ([]).

Operandi

In questa sezione vengono descritti i parametri elencati nella sintassi.

Descrizione

Questa sezione descrive il tipo di elemento (ad esempio, l'operatore, la funzione, e così via), gli eventuali valori restituiti e il modo in cui utilizzarlo.

Esempio

Questa sezione fornisce un esempio di codice illustrativo sull'uso dell'elemento.

Vedere anche

In questa sezione sono elencate le voci correlate del Dizionario di ActionScript.

Cartella Samples

Alcuni esempi di progetti Flash Lite completi con codice ActionScript funzionante sono disponibili nella pagina degli esempi e delle esercitazioni Flash Lite all'indirizzo www.adobe.com/go/learn_ft_samples_and_tutorials_it. Individuare il file .zip corrispondente alla propria versione di ActionScript, scaricarlo e decomprimerlo, quindi accedere alla cartella Samples, che contiene i file degli esempi.

Convenzioni tipografiche

In questa pubblicazione vengono usate le seguenti convenzioni tipografiche:

- Il *carattere corsivo* indica un valore che deve essere sostituito dall'utente (ad esempio, il percorso di una cartella).
- Il *carattere codice* indica una stringa di codice ActionScript.
- Il *carattere codice corsivo* indica un parametro di ActionScript.
- Il **carattere grassetto** indica un testo da immettere così come scritto.
- Le virgolette doppie (" ") negli esempi di codice indicano le stringhe delimitate. Tuttavia, i programmatori possono utilizzare anche le virgolette singole.

Questa sezione descrive la sintassi e l'uso delle funzioni globali di ActionScript per Macromedia Flash Lite 1.1 di Adobe. Comprende le funzioni seguenti:

Funzione	Descrizione
<code>call()</code>	Esegue lo script nel fotogramma chiamato senza spostare l'indicatore di riproduzione sul fotogramma.
<code>chr()</code>	Converte i numeri di codice ASCII in caratteri.
<code>duplicateMovieClip()</code>	Crea un'istanza di un clip filmato durante la riproduzione del file SWF.
<code>eval()</code>	Accede a variabili, proprietà, oggetti e clip filmato in base al nome.
<code>getProperty()</code>	Restituisce il valore della proprietà specificata per il clip filmato specificato.
<code>getTimer()</code>	Restituisce il numero di millisecondi trascorsi dall'inizio della riproduzione del file SWF.
<code>getURL()</code>	Carica un documento da un URL specifico in una finestra, oppure trasmette le variabili a un'altra applicazione in un URL definito.
<code>gotoAndPlay()</code>	Invia l'indicatore di riproduzione al fotogramma specificato in una scena e avvia la riproduzione da quel fotogramma. Se non viene specificata alcuna scena, l'indicatore di riproduzione passa al fotogramma specificato nella scena corrente.
<code>gotoAndStop()</code>	Invia l'indicatore di riproduzione al fotogramma specificato in una scena e lo arresta. Se non viene specificata alcuna scena, l'indicatore di riproduzione viene inviato al fotogramma nella scena corrente.
<code>ifFrameLoaded()</code>	Verifica se il contenuto di un fotogramma specifico è disponibile localmente.
<code>int()</code>	Converte un numero decimale in valore intero troncando il valore decimale.
<code>length()</code>	Restituisce il numero di caratteri del nome della variabile o della stringa specificata.

Funzione	Descrizione
<code>loadMovie()</code>	Carica un file SWF in Flash Lite durante la riproduzione del file SWF originale.
<code>loadMovieNum()</code>	Carica un file SWF in un livello di Flash Lite durante la riproduzione del file SWF originale.
<code>loadVariables()</code>	Legge i dati da un file esterno (ad esempio, un file di testo o un testo generato da ColdFusion, uno script CGI, ASP, PHP o Perl) e imposta i valori delle variabili in un livello di Flash Lite. Questa funzione può anche aggiornare le variabili del file SWF attivo con nuovi valori.
<code>loadVariablesNum()</code>	Legge i dati da un file esterno (ad esempio, un file di testo o un testo generato da ColdFusion, uno script CGI, ASP, PHP o Perl) e imposta i valori delle variabili in un livello di Flash Lite. Questa funzione può anche aggiornare le variabili del file SWF attivo con nuovi valori.
<code>mbchr()</code>	Converte un numero di codice ASCII in un carattere multibyte.
<code>mblength()</code>	Restituisce la lunghezza della stringa a caratteri multibyte.
<code>mbord()</code>	Converte il carattere specificato in un numero multibyte.
<code>mbsubstring()</code>	Estrae una nuova stringa a caratteri multibyte da una stringa a caratteri multibyte.
<code>nextFrame()</code>	Invia l'indicatore di riproduzione al fotogramma successivo e lo ferma.
<code>nextScene()</code>	Invia l'indicatore di riproduzione al fotogramma 1 della scena successiva e lo ferma.
<code>Number()</code>	Converte un'espressione in numero e restituisce un valore.
<code>on()</code>	Specifica l'evento associato all'utente o la pressione di un tasto che attiva un evento.
<code>ord()</code>	Converte i caratteri in numeri di codice ASCII.
<code>play()</code>	Sposta l'indicatore di riproduzione in avanti nella linea temporale.
<code>prevFrame()</code>	Invia l'indicatore di riproduzione al fotogramma precedente e lo ferma. Se il fotogramma corrente è il fotogramma 1, l'indicatore di riproduzione non si sposta.
<code>prevScene()</code>	Invia l'indicatore di riproduzione al fotogramma 1 della scena precedente e lo ferma.
<code>removeMovieClip()</code>	Elimina il clip filmato specificato creato originariamente mediante <code>duplicateMovieClip()</code> .
<code>set()</code>	Assegna un valore a una variabile.
<code>setProperty()</code>	Modifica un valore di proprietà di un clip filmato durante la riproduzione del filmato.

Funzione	Descrizione
<code>stop()</code>	Arresta il file SWF in esecuzione.
<code>stopAllSounds()</code>	Interrompe tutti i suoni attualmente riprodotti in un file SWF senza arrestare l'indicatore di riproduzione.
<code>String()</code>	Restituisce una rappresentazione sotto forma di stringa del parametro specificato.
<code>substring()</code>	Estrae parte di una stringa.
<code>tellTarget()</code>	Applica le istruzioni specificate nel parametro <i>statement(s)</i> alla linea temporale specificata nel parametro <i>target</i> .
<code>toggleHighQuality()</code>	Attiva e disattiva l'antialiasing in Flash Lite. L'antialiasing smussa i bordi degli oggetti ma rallenta la riproduzione dei file SWF.
<code>trace()</code>	Valuta l'espressione e visualizza il risultato nel pannello Output nella modalità di prova.
<code>unloadMovie()</code>	Rimuove da Flash Lite un clip filmato caricato mediante <code>loadMovie()</code> , <code>loadMovieNum()</code> o <code>duplicateMovieClip()</code> .
<code>unloadMovieNum()</code>	Rimuove da un livello di Flash Lite un clip filmato caricato mediante <code>loadMovie()</code> , <code>loadMovieNum()</code> o <code>duplicateMovieClip()</code> .

call()

Disponibilità

Flash Lite 1.0.

Uso

`call(frame)`

`call(movieClipInstance: frame)`

Operandi

frame L'etichetta o il numero di un fotogramma nella linea temporale.

movieClipInstance Il nome dell'istanza di un clip filmato.

Descrizione

Funzione; esegue lo script nel fotogramma chiamato senza spostare l'indicatore di riproduzione sul fotogramma. Le variabili locali non sono presenti dopo l'esecuzione dello script. La funzione `call()` può avere due forme possibili:

- Quella predefinita esegue lo script nel fotogramma specificato della stessa linea temporale in cui è stata eseguita la funzione `call()`, senza portare l'indicatore di riproduzione su tale fotogramma.
- L'istanza di clip specificata esegue lo script nel fotogramma specificato dell'istanza del clip filmato, senza portare l'indicatore di riproduzione su tale fotogramma.

NOTA

La funzione `call()` opera in modo sincrono; qualunque codice `ActionScript` che segue la funzione `call()` non viene eseguito finché non viene completato tutto il codice `ActionScript` nel fotogramma specificato.

Esempio

L'esempio seguente esegue lo script nel fotogramma `myScript`:

```
// per eseguire funzioni nel fotogramma con l'etichetta "myScript"
thisFrame = "myScript";
trace ("Calling the script in frame: " + thisFrame);

// per eseguire funzioni in qualunque altro fotogramma della stessa
// linea temporale
call("myScript");
```

chr()

Disponibilità

Flash Lite 1.0.

Uso

`chr(number)`

Operandi

number Un numero di codice ASCII.

Descrizione

Funzione di stringa; converte i numeri di codice ASCII in caratteri.

Esempio

L'esempio seguente converte il numero 65 nella lettera *A* e la assegna alla variabile `myVar`:

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

duplicateMovieClip()

Disponibilità

Flash Lite 1.0.

Uso

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

Operandi

target Il percorso di destinazione del clip filmato da duplicare.

newname Un identificatore univoco del clip filmato duplicato.

depth Un livello di profondità univoco per il clip filmato duplicato. Il livello di profondità rappresenta l'ordine di impilamento dei clip filmato duplicati, che è simile a quello dei livelli nella linea temporale: i clip filmato con un livello di profondità inferiore sono nascosti sotto i clip con un ordine di impilamento superiore. Assegnare a ogni clip filmato duplicato un livello di profondità univoco in modo che non vengano sovrascritti i clip filmato esistenti nei livelli di profondità occupati.

Descrizione

Funzione; crea un'istanza di un clip filmato durante la riproduzione del file SWF. Non restituisce alcun valore. L'indicatore di riproduzione in un clip filmato duplicato parte sempre dal fotogramma 1, indipendentemente dal punto in cui si trova l'indicatore nel clip filmato originale (principale). Le variabili del clip filmato principale non vengono copiate nel clip filmato duplicato. Se si elimina il clip filmato principale, vengono eliminati anche i clip filmato duplicati. Utilizzare la funzione o il metodo `removeMovieClip()` per eliminare un'istanza di clip filmato creata con `duplicateMovieClip()`. Per creare il riferimento al nuovo clip filmato, utilizzare la stringa passata come operando *newname*.

Esempio

L'esempio seguente duplica un clip filmato di nome `originalClip` per creare un nuovo clip di nome `newClip`, con livello di profondità 10. La posizione *x* del nuovo clip viene impostata su 100 pixel.

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

L'esempio seguente utilizza `duplicateMovieClip()` in un ciclo `for` per creare contemporaneamente più clip filmato nuovi. Una variabile di indice tiene traccia della profondità di impilamento massima occupata. Il nome di ogni clip filmato duplicato contiene un suffisso numerico che corrisponde alla rispettiva profondità di impilamento (`clip1`, `clip2`, `clip3`).

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

Vedere anche

[removeMovieClip\(\)](#)

eval()

Disponibilità

Flash Lite 1.0.

Uso

`eval(expression)`

Operandi

expression Il nome della variabile, della proprietà, dell'oggetto o del clip filmato da recuperare.

Descrizione

Funzione; accede a variabili, proprietà, oggetti e clip filmato in base al nome. Se *expression* è una variabile o una proprietà, viene restituito il valore della variabile o della proprietà. Se *expression* è un oggetto o un clip filmato, viene restituito un riferimento all'oggetto o al clip filmato. Se l'elemento nominato in *expression* non viene trovato, viene restituito il valore `undefined`.

È possibile utilizzare `eval()` per simulare gli array o per impostare e recuperare dinamicamente il valore di una variabile.

Esempio

L'esempio seguente utilizza `eval()` per determinare il valore dell'espressione "piece" + x. Dal momento che il risultato è un nome di variabile, `piece3`, `eval()` restituisce il valore della variabile e lo assegna a `y`:

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

L'esempio seguente mostra come simulare un array:

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

getProperty()

Disponibilità

Flash Lite 1.0.

Uso

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

Operandi

my_mc Il nome di istanza di un clip filmato per il quale viene recuperata la proprietà.

property Una proprietà di un clip filmato.

Descrizione

Funzione; restituisce il valore della proprietà specificata per il clip filmato *my_mc*.

Esempio

L'esempio seguente recupera la coordinata dell'asse orizzontale (`_x`) per il clip filmato `my_mc` nella linea temporale principale del filmato:

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

Vedere anche

[setProperty\(\)](#)

getTimer()

Disponibilità

Flash Lite 1.0.

Uso

```
getTimer()
```

Operandi

Nessuno.

Descrizione

Funzione; restituisce il numero di millisecondi trascorsi dall'inizio della riproduzione del file SWF.

Esempio

L'esempio seguente imposta la variabile `timeElapsed` sul numero di millisecondi che sono trascorsi dall'inizio della riproduzione del file SWF:

```
timeElapsed = getTimer();  
trace (timeElapsed); // Output: il numero di millisecondi trascorsi  
                        // dall'inizio della riproduzione del filmato
```

getURL()

Disponibilità

Flash Lite 1.0.

Uso

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

Operandi

url L'URL da cui ottenere il documento.

window Parametro opzionale che specifica la finestra o il frame HTML nel quale il documento deve essere caricato.

NOTA

Il parametro *window* non è specificato per le applicazioni Flash Lite, poiché i browser dei telefoni cellulari non supportano le finestre multiple.

È possibile immettere una stringa vuota o il nome di una finestra specifica oppure selezionare tra i seguenti nomi riservati di destinazione:

- `_self` indica il frame corrente nella finestra corrente.
- `_blank` indica una nuova finestra.
- `_parent` indica l'elemento principale del frame corrente.
- `_top` specifica il frame di primo livello nella finestra corrente.

variables Un metodo GET o POST per l'invio di variabili. Se non vi sono variabili, omettere questo parametro. Il metodo GET aggiunge le variabili alla fine dell'URL e viene utilizzato quando il numero di variabili è ridotto; il metodo POST invia le variabili in un'intestazione HTTP separata e viene utilizzato per l'invio di stringhe di variabili lunghe.

Descrizione

Funzione; carica un documento da un URL specifico in una finestra, oppure trasmette le variabili a un'altra applicazione in un URL definito. Per eseguire la prova di questa funzione, accertarsi che il file da caricare si trovi nella posizione specificata. Per utilizzare un URL assoluto (ad esempio, `http://www.myserver.com`), è necessario disporre di una connessione di rete.

Flash Lite 1.0 riconosce solo i protocolli HTTP, HTTPS, mailto e tel. Lo stesso vale per Flash Lite 1.1, che tuttavia riconosce anche i protocolli file, SMS (short message service) e MMS (multimedia message service).

Flash Lite passa la chiamata al sistema operativo, il quale la gestisce con l'applicazione predefinita registrata per il protocollo specificato.

Per ogni fotogramma o gestore di eventi viene elaborata una sola funzione `getUrl()`.

Alcuni microtelefoni limitano la funzione `getUrl()` ai soli eventi di tastiera, nel qual caso la chiamata a `getUrl()` viene elaborata solo se è attivata in un gestore di eventi di tastiera. Anche in tali circostanze, per ogni gestore di eventi viene elaborata una sola funzione `getUrl()`.

Esempio

Nel codice ActionScript seguente, Flash Lite Player apre `mobile.example.com` nel browser predefinito:

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

È possibile utilizzare GET o POST per inviare le variabili dalla linea temporale corrente.

L'esempio seguente utilizza il metodo GET per aggiungere delle variabili a un URL:

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

Il codice ActionScript seguente utilizza POST per inviare le variabili in un'intestazione HTTP:

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

È possibile assegnare una funzione pulsante per aprire una finestra di composizione e-mail con i campi di testo `address` (indirizzo), `subject` (oggetto) e `body` (corpo del testo) già compilati. Utilizzare uno dei metodi seguenti per assegnare una funzione pulsante: il metodo 1 per la codifica caratteri Maiusc-JIS o inglese; il metodo 2 solo per la codifica caratteri inglese.

Metodo 1: impostare le variabili per ognuno dei parametri desiderati, come nell'esempio seguente:

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

Metodo 2: definire ogni parametro all'interno della funzione `getURL()`, come nell'esempio seguente:

```
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");
}
```

Il metodo 1 restituisce un risultato con codifica URL automatica, mentre il metodo 2 conserva gli spazi nelle stringhe. Ad esempio, le stringhe risultanti dall'uso del metodo 1 sono simili alle seguenti:

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

Al contrario, il metodo 2 produce stringhe simili alle seguenti:

```
email subject
email body
```

L'esempio seguente utilizza il protocollo `tel`:

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

Nell'esempio seguente, `getURL()` viene utilizzato per inviare un messaggio SMS:

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

Nell'esempio seguente, `getUrl()` viene utilizzato per inviare un messaggio MMS:

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

Nell'esempio seguente, `getUrl()` viene utilizzato per aprire un file di testo memorizzato nel file system locale:

```
// esempio di protocollo file
filePath = "file:///c:/documents/flash/myApp/myvariables.txt";
on(keyPress "7") {
    getUrl(filePath);
}
```

gotoAndPlay()

Disponibilità

Flash Lite 1.0.

Uso

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

Operandi

scene Una stringa opzionale che specifica il nome della scena a cui viene inviato l'indicatore di riproduzione.

frame Un numero che rappresenta il numero del fotogramma, o una stringa che rappresenta l'etichetta del fotogramma a cui viene inviato l'indicatore di riproduzione.

Descrizione

Funzione; invia l'indicatore di riproduzione al fotogramma specificato in una scena e avvia la riproduzione da quel fotogramma. Se non viene specificata alcuna scena, l'indicatore di riproduzione passa al fotogramma specificato nella scena corrente.

È possibile utilizzare il parametro *scene* solo nella linea temporale principale, non all'interno delle linee temporali per clip filmato o altri oggetti del documento.

Esempio

Nell'esempio seguente, quando l'utente fa clic su un pulsante a cui è assegnato `gotoAndPlay()`, l'indicatore di riproduzione passa al fotogramma 16 della scena corrente e avvia la riproduzione del file SWF:

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

gotoAndStop()

Disponibilità

Flash 1.0.

Uso

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

Operandi

scene Una stringa opzionale che specifica il nome della scena a cui viene inviato l'indicatore di riproduzione.

frame Un numero che rappresenta il numero del fotogramma, o una stringa che rappresenta l'etichetta del fotogramma a cui viene inviato l'indicatore di riproduzione.

Descrizione

Funzione; invia l'indicatore di riproduzione al fotogramma specificato in una scena e lo arresta. Se non viene specificata alcuna scena, l'indicatore di riproduzione viene inviato al fotogramma nella scena corrente.

È possibile utilizzare il parametro *scene* solo nella linea temporale principale, non all'interno delle linee temporali per clip filmato o altri oggetti del documento.

Esempio

Nell'esempio seguente, quando l'utente fa clic su un pulsante a cui è assegnato `gotoAndStop()`, l'indicatore di riproduzione passa al fotogramma 5 della scena corrente e la riproduzione del file SWF viene interrotta:

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

ifFrameLoaded()

Disponibilità

Flash Lite 1.0.

Uso

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

Operandi

scene Una stringa opzionale che specifica il nome della scena da caricare.

frame Il numero di fotogramma o l'etichetta di fotogramma da caricare prima dell'esecuzione dell'istruzione successiva.

statement(s) Le istruzioni da eseguire se viene caricata la scena specificata o la scena con il fotogramma.

Descrizione

Funzione; verifica se il contenuto di un fotogramma specifico è disponibile localmente.

Utilizzare `ifFrameLoaded` per avviare la riproduzione di un'animazione semplice mentre il resto del file SWF viene scaricato nel computer locale. È anche possibile utilizzare la proprietà `_framesloaded` per verificare l'avanzamento dello scaricamento di un file SWF esterno. La differenza tra l'utilizzo di `_framesloaded` e `ifFrameLoaded` consiste nel fatto che `_framesloaded` consente di aggiungere istruzioni `if` o `else` personalizzate.

Esempio

L'esempio seguente utilizza la funzione `ifFrameLoaded` per verificare se viene caricato il fotogramma 10 del file SWF. Se è così, il comando `trace()` visualizza la frase "frame number 10 is loaded" nel pannello Output. Viene anche definita la variabile di output per mezzo della variabile `frame loaded: 10`.

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

Vedere anche

[_framesloaded](#)

int()

Disponibilità

Flash Lite 1.0.

Uso

`int(value)`

Operandi

value Un numero o una stringa da troncare per formare un numero intero.

Descrizione

Funzione; converte un numero decimale in valore intero troncando il valore decimale.

Esempio

L'esempio seguente tronca i numeri nelle variabili `distance` e `myDistance`:

```
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()

Disponibilità

Flash Lite 1.0.

Uso

`length(expression)`

`length(variable)`

Operandi

expression Una stringa.

variable Il nome di una variabile.

Descrizione

Funzione di stringa; restituisce il numero di caratteri del nome della variabile o della stringa specificata.

Esempio

L'esempio seguente restituisce la lunghezza della stringa "Hello":

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

Il risultato è 5.

L'esempio seguente convalida un indirizzo e-mail verificando che contenga almeno sei caratteri:

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

loadMovie()

Disponibilità

Flash Lite 1.1.

Uso

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

Operandi

url Una stringa che specifica l'URL assoluto o relativo del file SWF da caricare.

Un percorso relativo deve essere tale rispetto al file SWF che si trova sul livello 0. Gli URL assoluti devono includere il riferimento al protocollo, ad esempio `http://` o `file:///`.

target Un riferimento a un clip filmato o una stringa che rappresenta il percorso di un clip filmato `target`. Il clip filmato `target` viene sostituito dal file SWF caricato.

method Un parametro di stringa opzionale che specifica un metodo HTTP per l'invio di variabili. Il parametro deve essere la stringa `GET` o `POST`. Se non vi sono variabili da inviare, omettere questo parametro. Il metodo `GET` aggiunge le variabili alla fine dell'URL e viene utilizzato quando il numero di variabili è ridotto; il metodo `POST` invia le variabili in un'intestazione HTTP separata e viene utilizzato per l'invio di stringhe di variabili lunghe.

Descrizione

Funzione; carica un file SWF in Flash Lite durante la riproduzione del file SWF originale.

Per caricare un file SWF in un livello specifico, utilizzare la funzione `loadMovieNum()` anziché `loadMovie()`.

Quando un file SWF viene caricato in un clip filmato target, è possibile utilizzare il percorso target del clip filmato per indicare come destinazione il file SWF. Un file SWF caricato in un clip filmato target eredita le proprietà delle modifiche in scala, la posizione e la rotazione del clip filmato target. L'angolo superiore sinistro dell'immagine o del file SWF caricato viene allineato con il punto di registrazione del clip filmato target. Tuttavia, se il target è la linea temporale principale, l'angolo superiore sinistro dell'immagine o del file SWF viene allineato all'angolo superiore sinistro dello stage.

Utilizzare la funzione `unloadMovie()` per rimuovere i file SWF caricati mediante `loadMovie()`.

Esempio

L'esempio seguente carica il file SWF di nome `circle.swf` dalla stessa directory e sostituisce il clip filmato `mySquare` che è già presente sullo stage:

```
loadMovie("circle.swf", "mySquare");  
// Istruzione equivalente: loadMovie("circle.swf", _level0.mySquare);
```

Vedere anche

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

loadMovieNum()

Disponibilità

Flash Lite 1.1.

Uso

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

Operandi

url Una stringa che specifica l'URL assoluto o relativo del file SWF da caricare. Un percorso relativo deve essere tale rispetto al file SWF nel livello 0. Per l'utilizzo con la versione autonoma di Flash Lite Player o in modalità di prova nell'applicazione di creazione Flash, tutti i file SWF devono essere archiviati nella stessa cartella e i nomi file non devono contenere specifiche relative alla cartella o al disco.

level Un numero intero che specifica il livello di Flash Lite in cui viene caricato il file SWF.

method Un parametro di stringa opzionale che specifica un metodo HTTP per l'invio di variabili. Deve avere il valore GET o POST. Se non vi sono variabili da inviare, omettere questo parametro. Il metodo GET aggiunge le variabili alla fine dell'URL e viene utilizzato quando il numero di variabili è ridotto; il metodo POST invia le variabili in un'intestazione HTTP separata e viene utilizzato per l'invio di stringhe di variabili lunghe.

Descrizione

Funzione; carica un file SWF in un livello di Flash Lite durante la riproduzione del file SWF caricato originariamente.

Normalmente, Flash Lite viene chiuso dopo aver visualizzato un file SWF. La funzione `loadMovieNum()` consente di visualizzare diversi file SWF contemporaneamente e di passare da uno all'altro senza caricare un altro documento HTML.

Per specificare un oggetto target anziché un livello, utilizzare la funzione `loadMovie()` anziché `loadMovieNum()`.

Flash Lite prevede un ordine di impilamento dei livelli a partire dal livello 0. Questi livelli sono paragonabili a fogli di acetato, in quanto sono completamente trasparenti a eccezione degli oggetti su ciascun livello. Quando si utilizza `loadMovieNum()`, è necessario specificare un livello di Flash Lite in cui caricare il file SWF. Quando un file SWF viene caricato in un livello, è possibile utilizzare la sintassi `_levelN`, dove *N* è il numero del livello, per indicare come destinazione il file SWF.

Quando si carica un file SWF, è possibile specificare qualsiasi numero di livello. È possibile caricare i file SWF in un livello in cui sia già caricato un file SWF: il nuovo file SWF sostituisce quello esistente. Se si carica un file SWF nel livello 0, vengono scaricati tutti i livelli presenti in Flash Lite e il livello 0 viene sostituito con il nuovo file. Il file SWF nel livello 0 imposta la frequenza dei fotogrammi, il colore di sfondo e le dimensioni dei fotogrammi per tutti i file SWF caricati.

Utilizzare `unloadMovieNum()` per rimuovere i file SWF o le immagini che erano state caricate mediante `loadMovieNum()`.

Esempio

L'esempio seguente carica il file SWF nel livello 2:

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

Vedere anche

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

loadVariables()

Disponibilità

Flash Lite 1.1.

Uso

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

Operandi

url Una stringa che rappresenta un URL assoluto o relativo in cui si trovano le variabili. Se il file SWF che emette la chiamata viene eseguito in un browser Web, *url* deve trovarsi nello stesso dominio del file SWF.

target Il percorso target di un clip filmato che riceve le variabili caricate.

variables Un parametro di stringa opzionale che specifica un metodo HTTP per l'invio di variabili. Il parametro deve essere la stringa GET o POST. Se non vi sono variabili da inviare, omettere questo parametro. Il metodo GET aggiunge le variabili alla fine dell'URL e viene utilizzato quando il numero di variabili è ridotto; il metodo POST invia le variabili in un'intestazione HTTP separata e viene utilizzato per l'invio di stringhe di variabili lunghe.

Descrizione

Funzione; legge i dati da un file esterno, quali un file di testo o testo generato da uno script ColdFusion, CGI, ASP, PHP o Perl, e imposta i valori delle variabili in un clip filmato target. Questa funzione può anche aggiornare le variabili del file SWF attivo con nuovi valori.

Il testo nell'URL specificato deve essere nel formato MIME standard *application/x-www-form-urlencoded* (formato standard utilizzato dagli script CGI). È possibile specificare un numero di variabili qualsiasi. La seguente espressione definisce diverse variabili:

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

Per caricare le variabili in un livello specifico, utilizzare la funzione `loadVariablesNum()` anziché la funzione `loadVariables()`.

Esempio

Gli esempi seguenti caricano le variabili da un file di testo e da un server:

```
// carica le variabili dal file di testo nel file system
// locale (Symbian Series 60)
on(release, keyPress "1") {
    filePath = "file://c:/documents/flash/myApp/myvariables.txt";
    loadVariables(filePath, _root);
}
```

```
// carica le variabili (dal server) in un clip filmato
urlPath = "http://www.someserver.com/myvariables.txt";
loadVariables(urlPath, "myClip_mc");
```

Vedere anche

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

loadVariablesNum()

Disponibilità

Flash Lite 1.1.

Uso

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

Operandi

url Una stringa che rappresenta un URL assoluto o relativo in cui si trovano le variabili da caricare. Se il file SWF che emette la chiamata viene eseguito in un browser Web, *url* deve trovarsi nello stesso dominio del file SWF; per maggiori informazioni, consultare la sezione Descrizione.

level Un numero intero che specifica il livello di Flash Lite in cui ricevere le variabili.

variables Un parametro di stringa opzionale che specifica un metodo HTTP per l'invio di variabili. Il parametro deve essere la stringa GET o POST. Se non vi sono variabili da inviare, omettere questo parametro. Il metodo GET aggiunge le variabili alla fine dell'URL e viene utilizzato quando il numero di variabili è ridotto; il metodo POST invia le variabili in un'intestazione HTTP separata e viene utilizzato per l'invio di stringhe di variabili lunghe.

Descrizione

Funzione; legge i dati da un file esterno, ad esempio un file di testo o un testo generato da ColdFusion, uno script CGI, ASP, PHP o Perl, e imposta i valori delle variabili in un livello di Flash Lite. Questa funzione può anche aggiornare le variabili del file SWF attivo con nuovi valori.

Il testo nell'URL specificato deve essere nel formato MIME standard *application/x-www-form-urlencoded* (formato standard utilizzato dagli script CGI). È possibile specificare un numero di variabili qualsiasi. La seguente espressione di esempio definisce diverse variabili:

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

Normalmente, Flash Lite viene chiuso dopo aver visualizzato un singolo file SWF. La funzione `loadVariablesNum()` consente di visualizzare diversi file SWF contemporaneamente e di passare da uno all'altro senza caricare un altro documento HTML.

Per caricare le variabili in un clip filmato target, utilizzare la funzione `loadVariables()` anziché la funzione `loadVariablesNum()`.

Vedere anche

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

mbchr()

Disponibilità

Flash Lite 1.0.

Uso

```
mbchr(number)
```

Operandi

number Il numero da convertire in carattere multibyte.

Descrizione

Funzione di stringa; converte un numero di codice ASCII in un carattere multibyte.

Esempio

L'esempio seguente converte i numeri di codice ASCII nei rispettivi equivalenti a carattere multibyte:

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

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

Vedere anche

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

mblength()

Disponibilità

Flash Lite 1.0.

Uso

```
mblength(string)
```

Operandi

string Una stringa.

Descrizione

Funzione di stringa; restituisce la lunghezza della stringa a caratteri multibyte.

Esempio

L'esempio seguente visualizza la lunghezza della stringa nella variabile `myString`:

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

Vedere anche

[mbchr\(\)](#), [mbsubstring\(\)](#)

mbord()

Disponibilità

Flash Lite 1.0.

Uso

`mbord(character)`

Operandi

character Il carattere da convertire in numero multibyte.

Descrizione

Funzione di stringa; converte il carattere specificato in un numero multibyte.

Esempio

Gli esempi seguenti convertono i caratteri della variabile `myString` in numeri multibyte:

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

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

Vedere anche

[mbchr\(\)](#), [mbsubstring\(\)](#)

mbsubstring()

Disponibilità

Flash Lite 1.0.

Uso

```
mbsubstring(value, index, count)
```

Operandi

value La stringa multibyte da cui deve essere estratta una nuova stringa multibyte.

index Il numero del primo carattere da estrarre.

count Il numero di caratteri da includere nella stringa estratta, escluso il carattere *index*.

Descrizione

Funzione di stringa; estrae una nuova stringa a caratteri multibyte da una stringa a caratteri multibyte.

Esempio

L'esempio seguente estrae una nuova stringa di caratteri multibyte dalla stringa contenuta nella variabile `myString`:

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

Vedere anche

[mbchr\(\)](#)

nextFrame()

Disponibilità

Flash Lite 1.0.

Uso

```
nextFrame()
```

Operandi

Nessuno.

Descrizione

Funzione; invia l'indicatore di riproduzione al fotogramma successivo e lo arresta.

Esempio

Nell'esempio seguente, quando l'utente fa clic sul pulsante, l'indicatore di riproduzione passa al fotogramma successivo e si arresta:

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

Vedere anche

[prevFrame\(\)](#)

nextScene()

Disponibilità

Flash Lite 1.0.

Uso

```
nextScene()
```

Operandi

Nessuno.

Descrizione

Funzione; invia l'indicatore di riproduzione al fotogramma 1 della scena successiva e lo arresta.

Esempio

Nell'esempio seguente, quando un utente rilascia il pulsante, l'indicatore di riproduzione passa al fotogramma 1 della scena successiva:

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

Vedere anche

[prevScene\(\)](#)

Number()

Disponibilità

Flash Lite 1.0.

Uso

`Number(expression)`

Operandi

expression Un'espressione che consente di convertire in numero.

Descrizione

Funzione; converte il parametro *expression* in numero e restituisce un valore come descritto nell'elenco seguente:

- Se *expression* è un numero, il valore restituito è *expression*.
- Se *expression* è un valore booleano, viene restituito il valore 1 se *expression* è true oppure 0 se *expression* è false.
- Se *expression* è una stringa, la funzione cerca di analizzare *expression* come numero decimale con un esponente finale opzionale, vale a dire 1.57505e-3.
- Se *expression* è undefined, il valore restituito è -1.

Esempio

L'esempio seguente converte la stringa della variabile `myString` in un numero, memorizza il numero nella variabile `myNumber`, aggiunge 5 al numero e memorizza il risultato nella variabile `myResult`. La riga finale mostra il risultato quando si chiama `Number()` per un valore booleano.

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

on()

Disponibilità

Flash Lite 1.0.

Uso

```
on(event) {  
    // istruzione/i  
}
```

Operandi

statement(s) Le istruzioni da eseguire quando si verifica *event*.

event Questo attivatore viene definito *evento*. Quando si verifica un evento utente, vengono eseguite le istruzioni che lo seguono, racchiuse tra parentesi graffe ({}). Per il parametro *event* possono essere specificati i valori seguenti:

- `press` Il pulsante del mouse viene premuto quando il puntatore si trova sopra il pulsante.
- `release` Il pulsante del mouse viene rilasciato quando il puntatore si trova sopra il pulsante.
- `rollOut` Il puntatore esce dall'area del pulsante.
- `rollOver` Il puntatore del mouse scorre sopra il pulsante.
- `keyPress "key"` Viene premuto il tasto specificato. Per la porzione `key` del parametro, specificare un codice tasto o una costante `key`.

Descrizione

Gestore di eventi; specifica l'evento associato all'utente o alla pressione di un tasto che attiva una funzione. Non tutti gli eventi sono supportati.

Esempio

Il codice seguente, che scorre il campo `myText` verso il basso di una riga quando l'utente preme il tasto 8, effettua una verifica rispetto a `maxscroll` prima di scorrere:

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

ord()

Disponibilità

Flash Lite 1.0.

Uso

`ord(character)`

Operandi

character Il carattere da convertire in numero di codice ASCII.

Descrizione

Funzione di stringa; converte i caratteri in numeri di codice ASCII.

Esempio

L'esempio seguente utilizza la funzione `ord()` per visualizzare il codice ASCII per il carattere *A*:

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

play()

Disponibilità

Flash Lite 1.0.

Uso

`play()`

Operandi

Nessuno.

Descrizione

Funzione; sposta l'indicatore di riproduzione più avanti nella linea temporale.

Esempio

L'esempio seguente utilizza un'istruzione `if` per verificare il valore di un nome immesso dall'utente. Se l'utente immette `Steve`, viene chiamata la funzione `play()` e l'indicatore di riproduzione avanza nella linea temporale. Se l'utente immette un nome qualsiasi diverso da `Steve`, il file SWF non viene riprodotto e viene visualizzato un campo di testo con il nome di variabile `alert`.

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

prevFrame()

Disponibilità

Flash Lite 1.0.

Uso

`prevFrame()`

Operandi

Nessuno.

Descrizione

Funzione; invia l'indicatore di riproduzione al fotogramma precedente e lo arresta. Se il fotogramma corrente è il fotogramma 1, l'indicatore di riproduzione non si sposta.

Esempio

Quando l'utente fa clic su un pulsante a cui è associato il gestore seguente, l'indicatore di riproduzione passa al fotogramma precedente:

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

Vedere anche

[nextFrame\(\)](#)

prevScene()

Disponibilità

Flash Lite 1.0.

Uso

```
prevScene()
```

Operandi

Nessuno.

Descrizione

Funzione; invia l'indicatore di riproduzione al fotogramma 1 della scena precedente e lo arresta.

Esempio

In questo esempio, quando l'utente fa clic su un pulsante a cui è associato il gestore seguente, l'indicatore di riproduzione passa alla scena precedente:

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

Vedere anche

[nextScene\(\)](#)

random()

Disponibilità

Flash Lite 1.0.

Uso

```
random(value)
```

Operandi

value Un numero intero.

Descrizione

Funzione; restituisce un numero intero casuale compreso tra 0 e un valore inferiore al numero intero specificato nel parametro *value*.

Esempio

Gli esempi seguenti generano un numero basandosi su un numero intero che specifica l'intervallo:

```
//scegli un numero casuale tra 0 e 5
myNumber = random(5);
trace (myNumber);// Output: può essere 0,1,2,3,4
```

```
//scegli un numero casuale tra 5 e 10
myNumber = random(5) + 5;
trace (myNumber);// Output: può essere 5,6,7,8,9
```

Gli esempi seguenti generano un numero, quindi lo concatenano alla fine di una stringa che si sta valutando come nome di variabile. Si tratta di un esempio dell'utilizzo della sintassi di Flash Lite 1.1 per simulare gli array.

```
// scegli un nome casuale dall'elenco
myNames1 = "Mike";
myNames2 = "Debbie";
myNames3 = "Logan";

ran = random(3) + 1;
ranName = "myNames" + ran;
trace (eval(ranName));// Output: può essere mike, debbie o logan
```

removeMovieClip()

Disponibilità

Flash Lite 1.0.

Uso

```
removeMovieClip(target)
```

Operandi

target Il percorso di destinazione di un'istanza di clip filmato creata con `duplicateMovieClip()`.

Descrizione

Funzione; elimina il clip filmato specificato creato originariamente mediante `duplicateMovieClip()`.

Esempio

L'esempio seguente elimina il clip filmato duplicato di nome `second_mc`:

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

set()

Disponibilità

Flash Lite 1.0.

Uso

set(variable, expression)

Operandi

variable Un identificatore che contiene il valore del parametro *expression*.

expression Un valore assegnato alla variabile.

Descrizione

Istruzione; assegna un valore a una variabile. Una *variabile* è un “contenitore” di dati; il contenitore resta sempre invariato, ma il contenuto può variare. Mediante la modifica del valore di una variabile durante la riproduzione del file SWF, è possibile registrare e salvare informazioni sulle azioni svolte dall’utente, registrare i valori che vengono modificati durante la riproduzione del file o verificare se una condizione è `true` o `false`.

Le variabili possono contenere qualsiasi tipo di dati: stringa, numero, booleano o clip filmato. La linea temporale di ciascun file SWF e clip filmato dispone del proprio set di variabili, e ogni variabile ha il proprio valore, che è indipendente dalle variabili delle altre linee temporali.

Esempio

L’esempio seguente imposta una variabile di nome `orig_x_pos`, che memorizza la posizione originale dell’asse `x` del clip filmato `ship` per riportare la barca alla posizione di partenza più avanti nel file SWF:

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

Il codice precedente produce lo stesso risultato del codice seguente:

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

setProperty()

Disponibilità

Flash Lite 1.0.

Uso

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

Operandi

target Il percorso del nome dell'istanza del clip filmato di cui si sta impostando la proprietà.

property La proprietà da impostare.

value Il nuovo valore letterale della proprietà.

expression Un'equazione che restituisce il nuovo valore della proprietà.

Descrizione

Funzione; modifica un valore di proprietà di un clip filmato durante la riproduzione del filmato.

Esempio

L'istruzione seguente imposta la proprietà `_alpha` del clip filmato `star` sul 30% quando l'utente fa clic sul pulsante associato a questo gestore di eventi:

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

Vedere anche

[getProperty\(\)](#)

stop()

Disponibilità

Flash Lite 1.0.

Uso

```
stop()
```

Operandi

Nessuno.

Descrizione

Funzione; arresta il file SWF in esecuzione. Questa funzione viene comunemente utilizzata per controllare i clip filmato mediante i pulsanti.

Esempio

L'istruzione seguente chiama la funzione `stop()` quando l'utente fa clic sul pulsante associato a questo gestore di eventi:

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

stopAllSounds()

Disponibilità

Flash Lite 1.0.

Uso

```
stopAllSounds()
```

Operandi

Nessuno.

Descrizione

Funzione; interrompe tutti i suoni attualmente riprodotti in un file SWF senza arrestare l'indicatore di riproduzione. La riproduzione dei suoni impostati per la ripetizione riprende quando l'indicatore di riproduzione raggiunge i fotogrammi in cui si trovano tali suoni.

Esempio

Il codice seguente può essere applicato a un pulsante che, se selezionato, interrompe la riproduzione di tutti i suoni nel file SWF:

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

String()

Disponibilità

Flash Lite 1.0.

Uso

String(expression)

Operandi

expression Un'espressione che consente di convertire in una stringa.

Descrizione

Funzione; restituisce una rappresentazione sotto forma di stringa del parametro specificato, come descritto nell'elenco seguente:

- Se *expression* è un numero, la stringa restituita è la rappresentazione sotto forma di testo del numero.
- Se *expression* è una stringa, viene restituita la stringa *expression*.
- Se *expression* è un valore booleano, la stringa restituita è *true* o *false*.
- Se *expression* è un clip filmato, viene restituito il percorso target del clip filmato mediante notazione a barra (/).

Esempio

L'esempio seguente imposta *birthYearNum* su 1976, lo converte in stringa mediante la funzione *String()*, quindi lo confronta alla stringa "1976" mediante l'operatore *eq*.

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

substring()

Disponibilità

Flash Lite 1.0.

Uso

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

Operandi

string La stringa da cui estrarre la nuova stringa.

index Il numero del primo carattere da estrarre.

count Il numero di caratteri da includere nella stringa estratta, escluso il carattere *index*.

Descrizione

Funzione; estrae parte di una stringa. Questa funzione è a base uno, mentre i metodi della classe String sono a base zero.

Esempio

L'esempio seguente estrae i primi cinque caratteri dalla stringa "Hello World":

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

tellTarget()

Disponibilità

Flash Lite 1.0.

Uso

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

Operandi

target Una stringa che specifica il percorso di destinazione della linea temporale da controllare.

statement(s) Le istruzioni da eseguire se la condizione restituisce true.

Descrizione

Funzione; applica le istruzioni specificate nel parametro *statement(s)* alla linea temporale specificata nel parametro *target*. La funzione `tellTarget()` è utile per i controlli di navigazione. Assegnare `tellTarget()` ai pulsanti che arrestano o avviano i clip filmato in qualche altro punto dello stage. Inoltre, è possibile fare in modo che i clip filmato si spostino in un fotogramma particolare della clip. Ad esempio, è possibile assegnare `tellTarget()` a dei pulsanti che arrestano o avviano dei clip filmato sullo stage o che richiedano ai clip filmato di passare a un fotogramma particolare.

Esempio

Nell'esempio seguente, `tellTarget()` controlla l'istanza di clip filmato `ball` nella linea temporale principale. Il fotogramma 1 dell'istanza di `ball` è vuoto e su di esso è presente un'azione `stop()` in modo che non sia visibile sullo stage. Quando l'utente preme il tasto 5, `tellTarget()` comunica all'indicatore di riproduzione in `ball` di passare al fotogramma 2, in cui inizia l'animazione.

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

toggleHighQuality()

Disponibilità

Flash Lite 1.0.

Uso

```
toggleHighQuality()
```

Operandi

Nessuno.

Descrizione

Funzione; attiva e disattiva l'antialiasing in Flash Lite. L'antialiasing smussa i bordi degli oggetti ma rallenta la riproduzione dei file SWF. La funzione agisce su tutti i file SWF in Flash Lite.

Esempio

Ad esempio, il codice seguente può essere applicato a un pulsante che, una volta premuto, attiva e disattiva l'antialiasing:

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

trace()

Disponibilità

Flash Lite 1.0.

Uso

`trace(expression)`

Operandi

expression Un'espressione da valutare. Quando un file SWF viene aperto nello strumento di creazione Flash (mediante il comando Prova filmato), il valore del parametro *expression* viene visualizzato nel pannello Output.

Descrizione

Funzione; valuta l'espressione e visualizza il risultato nel pannello Output nella modalità di prova.

Utilizzare questa funzione per registrare le note di programmazione o visualizzare i messaggi nel pannello Output durante la prova di un file SWF. Utilizzare il parametro *expression* per verificare l'esistenza di una condizione o per visualizzare i valori nel pannello Output. La funzione `trace()` è simile alla funzione `alert` di JavaScript.

Il comando Ometti azioni Trace nelle impostazioni di pubblicazione consente di rimuovere le funzioni `trace()` dal file SWF esportato.

Esempio

L'esempio seguente utilizza la funzione `trace()` per osservare il comportamento di un ciclo `while`:

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

unloadMovie()

Disponibilità

Flash Lite 1.0.

Uso

```
unloadMovie(target)
```

Operandi

target Il percorso target di un clip filmato.

Descrizione

Funzione; rimuove da Flash Lite un clip filmato caricato mediante [loadMovie\(\)](#), [loadMovieNum\(\)](#) o [duplicateMovieClip\(\)](#).

Esempio

Quando l'utente preme il tasto 3, il codice seguente risponde scaricando il clip filmato `draggable_mc` nella linea temporale principale e caricando `movie.swf` nel livello 4 dell'impilamento dei documenti:

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

Quando l'utente preme il tasto 3, l'esempio seguente scarica il filmato caricato in precedenza nel livello 4:

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

Vedere anche

[loadMovie\(\)](#)

unloadMovieNum()

Disponibilità

Flash Lite 1.0.

Uso

```
unloadMovieNum(level)
```

Operandi

level Il livello (*_levelN*) di un filmato caricato.

Descrizione

Funzione; rimuove da Flash Lite un clip filmato caricato mediante [loadMovie\(\)](#), [loadMovieNum\(\)](#) o [duplicateMovieClip\(\)](#).

Normalmente, Flash Lite viene chiuso dopo aver visualizzato un file SWF. La funzione `unloadMovieNum()` consente di agire su diversi file SWF contemporaneamente e di passare da uno all'altro senza caricare un altro documento HTML.

Vedere anche

[loadMovieNum\(\)](#)

Questa sezione descrive le proprietà riconosciute da Macromedia Flash Lite 1.x di Adobe. Le voci sono elencate in ordine alfabetico, senza tener conto del carattere di sottolineatura () iniziale. Le proprietà sono riepilogate nella tabella seguente:

Proprietà	Descrizione
<code>/</code> (Barra)	Specifica o restituisce un riferimento alla linea temporale principale.
<code>_alpha</code>	Restituisce il valore della trasparenza alfa di un clip filmato.
<code>_currentframe</code>	Restituisce il numero del fotogramma sulla linea temporale in cui si trova l'indicatore di riproduzione.
<code>_focusrect</code>	Specifica se attorno al pulsante o al campo di testo attualmente attivo viene visualizzato un rettangolo giallo.
<code>_framesloaded</code>	Restituisce il numero di fotogrammi che sono stati caricati da un file SWF caricato in modo dinamico.
<code>_height</code>	Specifica l'altezza del clip filmato, espressa in pixel.
<code>_highquality</code>	Specifica il livello di antialiasing applicato al file SWF corrente.
<code>_level</code>	Restituisce un riferimento alla linea temporale principale di <code>_levelN</code> . Utilizzare la funzione <code>loadMovieNum()</code> per caricare i file SWF in Flash Lite Player prima di indicarli come destinazione mediante la proprietà <code>_level</code> . È anche possibile servirsi di <code>_levelN</code> per indicare come destinazione un file SWF al livello assegnato da <i>N</i> .
<code>maxscroll</code>	Indica il numero di riga della prima riga in alto del testo visibile all'interno di un campo di testo scorrevole in cui è visibile anche l'ultima riga in basso.
<code>_name</code>	Restituisce il nome dell'istanza di un clip filmato. Si applica solo ai clip filmato e non alla linea temporale principale.
<code>_rotation</code>	Restituisce la rotazione del clip filmato, espressa in pixel, rispetto alla posizione originale.

Proprietà	Descrizione
<code>scroll</code>	Controlla la visualizzazione delle informazioni in un campo di testo associato a una variabile. La proprietà <code>scroll</code> definisce il punto in cui il campo di testo inizia la visualizzazione del contenuto; una volta impostata, Flash Lite Player la aggiorna man mano che l'utente scorre il campo di testo.
<code>_target</code>	Restituisce il percorso target dell'istanza del clip filmato.
<code>_totalframes</code>	Restituisce il numero totale di fotogrammi in un clip filmato.
<code>_visible</code>	Indica se un clip filmato è visibile.
<code>_width</code>	Restituisce la larghezza del clip filmato, espressa in pixel.
<code>_x</code>	Contiene un numero intero che imposta la coordinata x di un clip filmato.
<code>_xscale</code>	Imposta la scala orizzontale (<i>percentuale</i>) del clip filmato applicata dal punto di registrazione del clip filmato.
<code>_y</code>	Contiene un numero intero che imposta la coordinata y di un clip filmato relativa alle coordinate locali del clip filmato principale.
<code>_yscale</code>	Imposta la scala verticale (<i>percentuale</i>) del clip filmato applicata dal punto di registrazione del clip filmato.

/ (Barra)

Disponibilità

Flash Lite 1.0.

Uso

/

/targetPath

/:varName

Descrizione

Identificatore; specifica o restituisce un riferimento alla linea temporale principale. La funzionalità fornita da questa proprietà è simile a quella fornita dalla proprietà `_root` in Flash 5.

Esempio

Per specificare una variabile su una linea temporale, utilizzare la sintassi della barra (/) combinata con i due punti (:).

Esempio 1: la variabile `car` che si trova sulla linea temporale principale:

```
/:car
```

Esempio 2: la variabile `car` nell'istanza di clip filmato `mc1` che si trova sulla linea temporale principale:

```
/mc1/:car
```

Esempio 3: la variabile `car` nell'istanza di clip filmato `mc2` nidificata nell'istanza di clip filmato `mc1` che si trova sulla linea temporale principale:

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

Esempio 4: la variabile `car` nell'istanza di clip filmato `mc2` che si trova sulla linea temporale corrente:

```
mc2/:car
```

_alpha

Disponibilità

Flash Lite 1.0.

Uso

```
my_mc._alpha
```

Proprietà; il valore della trasparenza alfa del clip filmato specificato dalla variabile `my_mc`.

I valori validi sono quelli compresi tra 0 (completamente trasparente) e 100 (completamente opaco), che è il valore predefinito. Gli oggetti in un clip filmato con `_alpha` impostato su 0 sono attivi, anche se sono invisibili. Ad esempio, è possibile fare clic su un pulsante in un clip filmato la cui proprietà `_alpha` è impostata su 0.

Esempio

Il codice seguente per un gestore di eventi pulsante imposta sul 30% la proprietà `_alpha` del clip filmato `my_mc` quando l'utente fa clic sul pulsante:

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

`_currentframe`

Disponibilità

Flash Lite 1.0.

Uso

```
my_mc._currentframe
```

Descrizione

Proprietà (sola lettura); restituisce il numero del fotogramma sulla linea temporale in cui si trova l'indicatore di riproduzione specificato dalla variabile *my_mc*.

Esempio

Nell'esempio seguente vengono utilizzate la proprietà `_currentframe` e la funzione `gotoAndStop()` per impostare l'avanzamento dell'indicatore di riproduzione del clip filmato *my_mc* di cinque fotogrammi avanti rispetto alla posizione corrente:

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

Vedere anche

[gotoAndStop\(\)](#)

`_focusrect`

Disponibilità

Flash Lite 1.0.

Uso

```
_focusrect = Boolean;
```

Descrizione

Proprietà (globale); specifica se attorno al pulsante o al campo di testo attualmente attivo viene visualizzato un rettangolo giallo. Il valore predefinito `true` visualizza un rettangolo giallo attorno al pulsante o al campo di testo attualmente attivo quando l'utente preme il tasto freccia su o giù sul proprio telefono o dispositivo portatile per spostarsi tra gli oggetti di un file SWF. Specificare `false` per fare in modo che il rettangolo giallo non venga visualizzato.

Esempio

L'esempio seguente disattiva la visualizzazione nell'applicazione del rettangolo giallo attorno all'elemento attivo:

```
_focusrect = false;
```

_framesloaded

Disponibilità

Flash Lite 1.0.

Uso

my_mc:_framesloaded

Descrizione

Proprietà (sola lettura); il numero di fotogrammi che sono stati caricati da un file SWF caricato in modo dinamico. Questa proprietà è utile per determinare se il contenuto di un fotogramma specifico e tutti i fotogrammi che lo precedono siano stati caricati e siano disponibili a livello locale nel browser. È utile anche come monitor durante lo scaricamento di file SWF di grandi dimensioni. Ad esempio, è possibile visualizzare un messaggio che indica all'utente che il caricamento del file SWF continua finché non è terminato il caricamento di uno specifico fotogramma nel file SWF.

Esempio

Nell'esempio seguente viene utilizzata la proprietà `_framesloaded` per avviare un file SWF dopo che sono stati caricati tutti i fotogrammi. Se non vengono caricati tutti i fotogrammi, la proprietà `_xscale` del `loader` dell'istanza del clip filmato viene aumentata proporzionalmente per creare una barra di avanzamento.

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

_height

Disponibilità

Flash Lite 1.0.

Uso

my_mc._height

Descrizione

Proprietà (sola lettura); l'altezza del clip filmato, espressa in pixel.

Esempio

Il seguente esempio di codice di gestore di eventi imposta l'altezza di un clip filmato quando l'utente fa clic sul pulsante del mouse:

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

_highquality

Disponibilità

Flash Lite 1.0.

Uso

_highquality

Descrizione

Proprietà (globale); specifica il livello di antialiasing applicato al file SWF corrente. Specificare 2 per applicare l'antialiasing di massima qualità. Specificare 1 per applicare un antialiasing di elevata qualità. Specificare 0 per impedire l'antialiasing.

Esempio

L'istruzione seguente applica l'antialiasing di elevata qualità al file SWF corrente:

```
_highquality = 1;
```

Vedere anche

[toggleHighQuality\(\)](#)

_level

Disponibilità

Flash Lite 1.0.

Uso

`_levelN`

Descrizione

Identificatore; un riferimento alla linea temporale principale di `_levelN`. Utilizzare la funzione `loadMovieNum()` per caricare i file SWF in Flash Lite Player prima di indicarli come destinazione mediante la proprietà `_level`. È anche possibile servirsi di `_levelN` per indicare come destinazione un file SWF al livello assegnato da *N*.

Il file SWF inizialmente caricato in un'istanza di Flash Lite Player viene automaticamente caricato in `_level0`. Il file SWF in `_level0` imposta la frequenza dei fotogrammi, il colore di sfondo e le dimensioni dei fotogrammi per tutti i file SWF che vengono caricati successivamente. I file vengono quindi ordinati nei livelli con numeri più alti, sopra il file SWF in `_level0`.

È necessario assegnare un livello a ciascun file SWF che viene caricato in Flash Lite Player mediante la funzione `loadMovieNum()`. I livelli possono essere assegnati in qualsiasi ordine. Se si assegna un livello che contiene già un file SWF (compreso `_level0`), il file che si trova in quel livello viene scaricato e sostituito dal nuovo file SWF.

Esempio

L'esempio seguente carica un file SWF nel livello 1, quindi arresta in corrispondenza del fotogramma 6 l'indicatore di riproduzione del file SWF caricato:

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

// almeno 1 fotogramma dopo
tellTarget(_level1) {
    gotoAndStop(6);
}
```

Vedere anche

[loadMovie\(\)](#)

maxscroll

Disponibilità

Flash Lite 1.1

Uso

variable_name:maxscroll

Descrizione

Proprietà (sola lettura); indica il numero di riga della prima riga in alto del testo visibile all'interno di un campo di testo scorrevole in cui è visibile anche l'ultima riga in basso. La proprietà `maxscroll` agisce in combinazione con la proprietà `scroll` per controllare il modo in cui vengono visualizzate le informazioni in un campo di testo. La proprietà può essere recuperata, ma non modificata.

Esempio

Il codice seguente, che scorre il campo `myText` verso il basso di una riga quando l'utente preme il tasto 8, effettua una verifica rispetto a `maxscroll` prima di scorrere:

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

Vedere anche

[scroll](#)

_name

Disponibilità

Flash Lite 1.0.

Uso

`my_mc:_name`

Descrizione

Proprietà; il nome di istanza del clip filmato specificato da `my_mc`. Si applica solo ai clip filmato e non alla linea temporale principale.

Esempio

L'esempio seguente visualizza il nome del clip filmato `bigRose` nel pannello Output sotto forma di stringa:

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

_rotation

Disponibilità

Flash Lite 1.0.

Uso

`my_mc:_rotation`

Descrizione

Proprietà; la rotazione del clip filmato, espressa in pixel, rispetto alla posizione originale. I valori da 0 a 180 rappresentano la rotazione oraria, quelli da 0 a -180 la rotazione antioraria. I valori esterni a questo intervallo vengono aggiunti o sottratti a 360 per ottenere un valore interno all'intervallo. Ad esempio, l'istruzione `my_mc:_rotation = 450` è uguale a `my_mc:_rotation = 90`.

Esempio

L'esempio seguente ruota il clip filmato `my_mc` di 15 gradi in senso orario quando l'utente preme il tasto 2:

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

scroll

Disponibilità

Flash Lite 1.1.

Uso

textFieldVariableName:scroll

Descrizione

Proprietà; controlla la visualizzazione delle informazioni in un campo di testo associato a una variabile. La proprietà `scroll` definisce il punto in cui il campo di testo inizia la visualizzazione del contenuto; una volta impostata, Flash Lite Player la aggiorna man mano che l'utente scorre il campo di testo. È possibile utilizzare la proprietà `scroll` per creare un campo di testo scorrevole o per dirigere un utente a uno specifico brano di un testo particolarmente lungo.

Esempio

Il codice seguente scorre il campo di testo `myText` verso l'alto di una riga ogni volta che l'utente preme il tasto 2:

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

Vedere anche

[maxscroll](#)

_target

Disponibilità

Flash Lite 1.0.

Uso

my_mc:_target

Descrizione

Proprietà (sola lettura); restituisce il percorso target dell'istanza di clip filmato specificata da *my_mc*.

_totalframes

Disponibilità

Flash Lite 1.0.

Uso

my_mc:_totalframes

Descrizione

Proprietà (sola lettura); restituisce il numero totale di fotogrammi del clip filmato *my_mc*.

Esempio

Il codice seguente carica mySWF.swf nel livello 1 e, 25 fotogrammi più avanti, verifica se è stato caricato:

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

// 25 fotogrammi più avanti nella linea temporale principale
if (_level1._framesloaded >= _level1._totalframes) {
    tellTarget("_level1/") {
        gotoAndStop("myLabel");
    }
} else {
    // ripetizione ciclica...
}
```

_visible

Disponibilità

Flash Lite 1.0.

Uso

my_mc:_visible

Descrizione

Proprietà; un valore booleano che indica se il clip filmato specificato da *my_mc* è visibile. I clip filmato non visibili (quelli con la proprietà *_visible* impostata su *false*) sono disabilitati.

Ad esempio, non è possibile fare clic su un pulsante presente in un clip filmato con *_visible* impostato su *false*. I clip filmato sono sempre visibili a meno che non siano stati esplicitamente resi invisibili in questo modo.

Esempio

Il codice seguente disattiva il clip filmato `my_mc` quando l'utente preme il tasto 3 e lo attiva quando preme il tasto 4:

```
on(keyPress "3") {  
    my_mc:_visible = 0;  
}  
  
on(keyPress "4") {  
    my_mc:_visible = 1;  
}
```

`_width`

Disponibilità

Flash Lite 1.0.

Uso

`my_mc:_width`

Descrizione

Proprietà; la larghezza del clip filmato, espressa in pixel.

Esempio

L'esempio seguente imposta le proprietà della larghezza di un clip filmato quando l'utente preme il tasto 5:

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

`_X`

Disponibilità

Flash Lite 1.0.

Uso

`my_mc:_X`

Descrizione

Proprietà; un numero intero che imposta la coordinata x di un clip filmato (rappresentato in questo caso da `my_mc`) relativa alle coordinate locali del clip filmato principale. Se un clip filmato si trova sulla linea temporale principale, le sue coordinate considerano l'angolo superiore sinistro dello stage come punto (0, 0).

Se è all'interno di un altro clip filmato che contiene delle trasformazioni, il clip filmato utilizza il sistema di coordinate locali del clip filmato che lo racchiude. Ad esempio, se un clip filmato viene ruotato di 90 gradi in senso antiorario, i clip filmato secondari ereditano una coordinata ruotata di 90 gradi in senso antiorario. Le coordinate del clip filmato sono relative al punto di registrazione.

Esempio

L'esempio seguente modifica la posizione orizzontale del clip filmato `my_mc` quando l'utente preme il tasto 6:

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

Vedere anche

[_xscale](#), [_y](#), [_yscale](#)

_xscale

Disponibilità

Flash Lite 1.0.

Uso

`my_mc:_xscale`

Descrizione

Proprietà; imposta la scala orizzontale (*percentuale*) del clip filmato applicata dal punto di registrazione del clip filmato. Il punto di registrazione predefinito è (0, 0).

Se si modifica in scala il sistema di coordinate locale, vengono cambiate anche le impostazioni delle proprietà `_x` e `_y`, che sono definite in pixel. Ad esempio, quando il clip filmato principale viene modificato in scala del 50%, se si imposta la proprietà `_x` un oggetto nel clip filmato viene spostato per la metà dei pixel per cui verrebbe spostato se il filmato fosse al 100% delle proprie dimensioni.

Esempio

L'esempio seguente modifica la scala orizzontale del clip filmato `my_mc` quando l'utente preme il tasto 7:

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

Vedere anche

[_x](#), [_y](#), [_yscale](#)

_y

Disponibilità

Flash Lite 1.0.

Uso

`my_mc:_y`

Descrizione

Proprietà; un numero intero che imposta la coordinata *y* di un clip filmato (rappresentato in questo caso da *my_mc*) relativa alle coordinate locali del clip filmato principale. Se un clip filmato si trova sulla linea temporale principale, le sue coordinate considerano l'angolo superiore sinistro dello stage come punto (0, 0).

Se è all'interno di un altro clip filmato che contiene delle trasformazioni, il clip filmato utilizza il sistema di coordinate locali del clip filmato. Ad esempio, se un clip filmato viene ruotato di 90 gradi in senso antiorario, i clip filmato secondari ereditano una coordinata ruotata di 90 gradi in senso antiorario. Le coordinate del clip filmato sono relative al punto di registrazione.

Esempio

Il codice seguente imposta la coordinata *y* del clip filmato `my_mc` 10 pixel sotto la coordinata (0, 0) del relativo clip principale quando l'utente preme il tasto 1:

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

Vedere anche

[_x](#), [_xscale](#), [_yscale](#)

_yscale

Disponibilità

Flash Lite 1.0.

Uso

`my_mc:_yscale`

Descrizione

Proprietà; imposta la scala verticale (*percentuale*) del clip filmato applicata dal punto di registrazione del clip filmato. Il punto di registrazione predefinito è (0, 0).

Se si modifica in scala il sistema di coordinate locale, vengono cambiate anche le impostazioni delle proprietà `_x` e `_y`, che sono definite in pixel. Ad esempio, quando il clip filmato principale viene modificato in scala del 50%, se si imposta la proprietà `_y` un oggetto nel clip filmato viene spostato per la metà dei pixel per cui verrebbe spostato se il filmato fosse al 100% delle proprie dimensioni.

Esempio

L'esempio seguente modifica al 10% la scala verticale del clip filmato `my_mc` quando l'utente preme il tasto 1:

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

Vedere anche

[_x](#), [_xscale](#), [_y](#)

Questa sezione descrive la sintassi e l'uso delle istruzioni ActionScript per Macromedia Flash Lite 1.x di Adobe. Le istruzioni sono elementi di linguaggio che eseguono o specificano un'azione, e sono riepilogate nella tabella seguente:

Istruzione	Descrizione
<code>break</code>	Indica a Flash Lite di saltare il resto del corpo del ciclo, interrompere la ripetizione ciclica ed eseguire l'istruzione che segue l'istruzione ciclica.
<code>case</code>	Definisce una condizione per l'istruzione <code>switch</code> . Le istruzioni nel parametro <code>statements</code> vengono eseguite se il parametro <code>expression</code> che segue la parola chiave <code>case</code> è uguale al parametro <code>expression</code> dell'istruzione <code>switch</code> .
<code>continue</code>	Passa oltre tutte le istruzioni rimanenti nel ciclo più interno e avvia l'iterazione successiva del ciclo come se il controllo fosse arrivato normalmente alla fine del ciclo.
<code>do..while</code>	Esegue le istruzioni, quindi valuta la condizione in un ciclo fintanto che la condizione è <code>true</code> .
<code>else</code>	Specifica le istruzioni da eseguire se la condizione nell'istruzione <code>if</code> restituisce <code>false</code> .
<code>else if</code>	Valuta una condizione e specifica le istruzioni da eseguire se la condizione nell'istruzione <code>if</code> iniziale restituisce <code>false</code> .
<code>for</code>	Valuta l'espressione <code>init</code> (initialize) una volta, quindi inizia una sequenza di ripetizione ciclica per mezzo della quale, fintanto che <code>condition</code> restituisce <code>true</code> , <code>statement</code> viene eseguita, quindi viene valutata l'espressione successiva.
<code>if</code>	Valuta una condizione per determinare l'azione successiva da eseguire in un file SWF. Se la condizione restituisce <code>true</code> , Flash esegue le istruzioni che seguono la condizione all'interno delle parentesi graffe (<code>{}</code>). Se la condizione è <code>false</code> , Flash ignora le istruzioni all'interno delle parentesi graffe ed esegue le istruzioni dopo le parentesi.

Istruzione	Descrizione
switch	Come nel caso dell'istruzione <code>if</code> , l'istruzione <code>switch</code> prova una condizione ed esegue le istruzioni se la condizione restituisce il valore <code>true</code> .
while	Prova un'espressione ed esegue ripetutamente in un ciclo un'istruzione o una serie di istruzioni fintanto che l'espressione è <code>true</code> .

break

Disponibilità

Flash Lite 1.0.

Uso

`break`

Parametri

Nessuno.

Descrizione

Istruzione; viene visualizzata all'interno di un ciclo (`for`, `do..while` o `while`) oppure in un blocco di istruzioni associate a un caso particolare all'interno di un'istruzione `switch`.

L'istruzione `break` indica a Flash Lite di saltare il resto del corpo del ciclo, interrompere la ripetizione ciclica ed eseguire l'istruzione che segue l'istruzione ciclica. Quando si utilizza l'istruzione `break`, l'interprete ActionScript salta il resto delle istruzioni nel blocco `case` e passa alla prima istruzione l'istruzione `switch` corrispondente. Utilizzare questa istruzione per uscire da una serie di cicli nidificati.

Esempio

L'esempio seguente utilizza l'istruzione `break` per uscire da un ciclo altrimenti infinito:

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

Vedere anche

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

case

Disponibilità

Flash Lite 1.0.

Uso

case expression: statements

Parametri

expression Qualsiasi espressione.

statements Qualsiasi istruzione.

Descrizione

Istruzione; definisce una condizione per l'istruzione `switch`. Le istruzioni nel parametro *statements* vengono eseguite se il parametro *expression* che segue la parola chiave `case` è uguale al parametro *expression* dell'istruzione `switch`.

Se si utilizza l'istruzione `case` al di fuori di un'istruzione `switch`, si verifica un errore e lo script non viene compilato.

Esempio

Nell'esempio seguente, se il parametro `myNum` restituisce 1, viene eseguita l'istruzione `trace()` che segue `case 1`; se il parametro `myNum` restituisce 2, viene eseguita l'istruzione `trace()` che segue `case 2`; e così via. Se nessuna espressione `case` corrisponde al parametro `number`, viene eseguita l'istruzione `trace()` che segue la parola chiave `default`.

```
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")
}
```

Nell'esempio seguente, se nel primo gruppo case non è presente alcuna istruzione break, se il numero è 1 sia A che B compaiono nel pannello Output:

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

Vedere anche

[switch](#)

continue

Disponibilità

Flash Lite 1.0.

Uso

`continue`

Parametri

Nessuno.

Descrizione

Istruzione; passa oltre tutte le istruzioni rimanenti nel ciclo più interno e avvia l'iterazione successiva del ciclo come se il controllo fosse arrivato normalmente alla fine del ciclo. Non ha alcun effetto al di fuori di un ciclo.

- In un ciclo `while`, `continue` indica all'interprete di Flash di ignorare il resto del corpo del ciclo e di passare all'inizio del ciclo, dove viene provata la condizione:
- In un ciclo `do..while`, `continue` indica all'interprete di Flash di ignorare il resto del corpo del ciclo e di passare alla fine del ciclo, dove viene provata la condizione:
- In un ciclo `for`, `continue` indica all'interprete di Flash di ignorare il resto del corpo del ciclo e di passare alla valutazione dell'espressione del ciclo `for` dopo l'operazione:

Esempio

In un ciclo `while`, `continue` indica a Flash Lite di ignorare il resto del corpo del ciclo e di passare all'inizio del ciclo, dove viene provata la condizione:

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

In un ciclo `do..while`, `continue` indica a Flash Lite di ignorare il resto del corpo del ciclo e di passare alla fine del ciclo, dove viene provata la condizione:

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

In un ciclo `for`, `continue` fa in modo che il resto del corpo del ciclo venga ignorato.

Nell'esempio seguente, se il modulo `i % 3` è uguale a 0, l'istruzione `trace(i)` viene ignorata:

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

Vedere anche

[do..while](#), [for](#), [while](#)

do..while

Disponibilità

Flash Lite 1.0.

Uso

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

Parametri

statement(s) Le istruzioni da eseguire fintanto che il parametro *condition* restituisce true.

condition La condizione da valutare.

Descrizione

Istruzione; esegue le istruzioni, quindi valuta la condizione in un ciclo fintanto che la condizione è true.

Esempio

L'esempio seguente incrementa la variabile di indice fintanto che il suo valore è inferiore a 10:

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

Vedere anche

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

else

Disponibilità

Flash Lite 1.0.

Uso

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

Parametri

condition Espressione che restituisce `true` o `false`.

t-statement(s) Le istruzioni da eseguire se la condizione restituisce `true`.

f-statement(s) Una serie alternativa di istruzioni da eseguire se la condizione restituisce `false`.

Descrizione

Istruzione; specifica le istruzioni da eseguire se la condizione nell'istruzione `if` restituisce `false`.

Esempio

L'esempio seguente mostra l'uso dell'istruzione `else` con una condizione. Un esempio reale conterrebbe del codice che specifica un'azione da eseguire in base alla condizione.

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

Vedere anche

[if](#)

else if

Disponibilità

Flash Lite 1.0.

Uso

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

Parametri

condition Espressione che restituisce `true` o `false`.

statement(s) Una serie di istruzioni da eseguire se la condizione specificata nell'istruzione `if` è `false`.

Descrizione

Istruzione; valuta una condizione e specifica le istruzioni da eseguire se la condizione nell'istruzione `if` iniziale restituisce `false`. Se la condizione `else if` restituisce un valore `true`, l'interprete Flash esegue le istruzioni che seguono la condizione `else if` tra parentesi graffe (`{}`). Se la condizione `else if` è `false`, Flash ignora le istruzioni all'interno delle parentesi graffe ed esegue le istruzioni dopo le parentesi. Utilizzare l'istruzione `else if` per creare una struttura logica ad albero negli script.

Esempio

L'esempio seguente utilizza le istruzioni `else if` per verificare se ogni lato dell'oggetto rientra in un limite specifico:

```
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");
}
```

Vedere anche

[if](#)

for

Disponibilità

Flash Lite 1.0.

Uso

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

Parametri

init Un'espressione da valutare prima dell'inizio della sequenza di ripetizione ciclica; solitamente si tratta di un'espressione di assegnazione.

condition Espressione che restituisce `true` o `false`. La condizione viene valutata prima di ciascuna iterazione del ciclo; il ciclo si interrompe quando la condizione restituisce `false`.

next Un'espressione da valutare dopo ciascuna iterazione del ciclo; solitamente un'espressione di assegnazione mediante l'operatore di incremento (`++`) o decremento (`--`).

statement(s) Una o più istruzioni da eseguire nel ciclo.

Descrizione

Istruzione; un costrutto di ciclo che valuta l'espressione *init* (initialize) una volta, quindi inizia una sequenza di ripetizione ciclica per mezzo della quale, fintanto che *condition* restituisce `true`, *statement* viene eseguita, quindi viene valutata l'espressione successiva.

Alcune proprietà non possono essere enumerate dall'istruzione `for` o `for..in`; ad esempio, le proprietà di clip filmato come `_x` e `_y`.

Esempio

L'esempio seguente utilizza il ciclo `for` per sommare i numeri da 1 a 100:

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

Vedere anche

[++ \(incremento\)](#), [-- \(decremento\)](#), [do..while](#), [while](#)

if

Disponibilità

Flash Lite 1.0.

Uso

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

Parametri

condition Espressione che restituisce true o false.

statement(s) Le istruzioni da eseguire se la condizione restituisce true.

Descrizione

Istruzione; valuta una condizione per determinare l'azione successiva da eseguire in un file SWF. Se la condizione restituisce true, Flash esegue le istruzioni che seguono la condizione all'interno delle parentesi graffe ({}). Se la condizione è false, Flash ignora le istruzioni all'interno delle parentesi graffe ed esegue le istruzioni dopo le parentesi. Utilizzare l'istruzione else if per creare una struttura logica ad albero negli script.

Esempio

Nell'esempio seguente, la condizione tra parentesi restituisce la variabile nome per verificare se contiene il valore letterale "Erica". In caso affermativo, la funzione play() viene eseguita.

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

switch

Disponibilità

Flash Lite 1.0.

Uso

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

Parametri

expression Qualsiasi espressione numerica.

caseClause Una parola chiave `case` seguita da un'espressione, due punti e un gruppo di istruzioni da eseguire se l'espressione corrisponde al parametro *expression* dello `switch`.

defaultClause Una parola chiave `default` opzionale seguita dalle istruzioni da eseguire se nessuna delle espressioni `case` corrisponde al parametro *expression* dello `switch`.

Descrizione

Istruzione; crea una struttura ad albero per le istruzioni `ActionScript`. Come nel caso dell'istruzione `if`, l'istruzione `switch` prova una condizione ed esegue le istruzioni se la condizione restituisce il valore `true`.

Le istruzioni `switch` contengono un'opzione di riserva di nome `default`. Se nessun'altra istruzione è `true`, viene eseguita l'istruzione `default`.

Esempio

Nell'esempio seguente, se il parametro `myNum` restituisce 1, viene eseguita l'istruzione `trace()` che segue `case 1`; se il parametro `myNum` restituisce 2, viene eseguita l'istruzione `trace()` che segue `case 2`; e così via. Se nessuna espressione `case` corrisponde al parametro `number`, viene eseguita l'istruzione `trace()` che segue la parola chiave `default`.

```
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")
}
```

Nell'esempio seguente, il primo gruppo `case` non contiene un'istruzione `break`, pertanto se il numero è 1 sia A che B compaiono nel pannello `Output`:

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

Vedere anche

[case](#)

while

Disponibilità

Flash Lite 1.0.

Uso

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

Parametri

condition L'espressione che viene valutata ogni volta che viene eseguita l'istruzione `while`.

statement(s) Le istruzioni da eseguire quando la condizione restituisce `true`.

Descrizione

Istruzione; prova un'espressione ed esegue ripetutamente in un ciclo un'istruzione o una serie di istruzioni fintanto che l'espressione è `true`.

Prima che venga eseguito il blocco di istruzioni, la condizione viene verificata; se la verifica restituisce `true`, il blocco viene eseguito. Se la condizione è `false`, il blocco di istruzioni viene saltato e viene eseguita la prima istruzione che segue il blocco di istruzioni dell'istruzione `while`.

La ripetizione ciclica viene solitamente utilizzata per eseguire un'azione finché la variabile `counter` è inferiore al valore specificato. Alla fine di ogni ciclo, la variabile `counter` viene incrementata finché non viene raggiunto il valore specificato. A quel punto, la condizione non è più `true` e il ciclo termina.

L'istruzione `while` esegue le serie di operazioni seguenti. Ogni ripetizione dal punto 1 al punto 4 viene definita *iterazione* del ciclo. La condizione viene verificata all'inizio di ogni iterazione:

1. L'espressione *condition* viene valutata.
2. Se *condition* restituisce `true` o un valore che viene convertito nel valore booleano `true` (ad esempio, un numero diverso da zero), passare al punto 3.

Altrimenti, l'istruzione `while` viene completata e l'esecuzione viene ripresa in corrispondenza della prima istruzione dopo il ciclo `while`.

3. Eseguire il blocco di istruzioni *statement(s)*.
4. Andare al punto 1.

Esempio

L'esempio seguente esegue un ciclo fintanto che il valore della variabile di indice i è inferiore a 10:

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

Vedere anche

[continue](#), [do..while](#), [for](#)

Questa sezione descrive la sintassi e l'uso degli operatori ActionScript per Macromedia Flash Lite 1.x di Adobe. Tutte le voci sono elencate in ordine alfabetico. Tuttavia, alcuni operatori sono simboli e sono pertanto ordinati alfabeticamente in base alla relativa descrizione testuale. Gli operatori descritti in questa sezione sono riepilogati nella tabella seguente:

Operatore	Descrizione
<code>add</code> (concatenazione stringhe)	Concatena (combina) due o più stringhe.
<code>+=</code> (assegnazione addizione)	Assegna a <i>expression1</i> il valore di <i>expression1</i> + <i>expression2</i> .
<code>and</code>	Esegue un'operazione AND logica.
<code>=</code> (assegnazione)	Assegna il valore di <i>expression2</i> (l'operando a destra) alla variabile o alla proprietà in <i>expression1</i> .
<code>/*</code> (commento a blocchi)	Indica una o più righe di commento di script. Tutti i caratteri racchiusi tra il tag di inizio commento (<code>/*</code>) e il tag di fine commento (<code>*/</code>) vengono considerati come commenti e ignorati dall'interprete di ActionScript.
<code>,</code> (virgola)	Valuta <i>expression1</i> e successivamente <i>expression2</i> , quindi restituisce il valore di <i>expression2</i> .
<code>//</code> (commento)	Indica l'inizio di un commento di script. Tutti i caratteri racchiusi tra il delimitatore (<code>//</code>) e il carattere di fine riga vengono considerati commenti e ignorati dall'interprete di ActionScript.
<code>?:</code> (condizionale)	Indica a Flash Lite di valutare <i>expression1</i> e, se il valore di <i>expression1</i> è true, l'operatore restituisce il valore di <i>expression2</i> ; altrimenti restituisce il valore di <i>expression3</i> .
<code>--</code> (decremento)	Sottrae 1 da <i>expression</i> . Il formato di decremento prima dell'operazione (<code>--expression</code>) sottrae 1 da <i>expression</i> e restituisce il risultato sotto forma di numero. Il formato post-decremento dell'operatore (<code>expression--</code>) sottrae 1 da <i>expression</i> e restituisce il valore iniziale di <i>expression</i> (il valore prima della sottrazione).

Operatore	Descrizione
/ (divisione)	Divide <i>expression1</i> per <i>expression2</i> .
/= (assegnazione divisione)	Assegna a <i>expression1</i> il valore di <i>expression1</i> / <i>expression2</i> .
. (punto)	Utilizzato per spostarsi nelle gerarchie dei clip filmato e accedere ai clip filmato nidificati (secondari), alle variabili e alle proprietà.
++ (incremento)	Aggiunge 1 a <i>expression</i> . Il valore di <i>expression</i> può essere una variabile, un elemento di un array o la proprietà di un oggetto. Il formato di incremento prima dell'operazione (<i>++expression</i>) aggiunge 1 a <i>expression</i> e restituisce il risultato sotto forma di numero. Il formato post-decremento dell'operatore (<i>expression++</i>) aggiunge 1 a <i>expression</i> e restituisce il valore iniziale di <i>expression</i> (il valore prima dell'addizione).
&& (AND logico)	Valuta <i>expression1</i> (l'espressione a sinistra dell'operatore) e restituisce <i>false</i> se l'espressione equivale a <i>false</i> . Se <i>expression1</i> restituisce <i>true</i> , viene valutata <i>expression2</i> (l'espressione a destra dell'operatore). Se <i>expression2</i> restituisce <i>true</i> , il risultato finale è <i>true</i> ; in caso contrario, è <i>false</i> .
! (NOT logico)	Inverte il valore booleano di una variabile o espressione. Se <i>expression</i> è una variabile con un valore convertito o assoluto <i>true</i> , il valore di <i>!expression</i> è <i>false</i> . Se l'espressione <i>x && y</i> restituisce <i>false</i> , l'espressione <i>!(x && y)</i> restituisce <i>true</i> .
(OR logico)	Valuta <i>expression1</i> ed <i>expression2</i> . Il risultato è <i>true</i> se almeno una delle espressioni restituisce <i>true</i> ; il risultato è <i>false</i> solo se entrambe le espressioni restituiscono <i>false</i> . È possibile utilizzare l'operatore logico OR con un numero qualsiasi di operandi; se uno degli operandi restituisce <i>true</i> , il risultato è <i>true</i> .
% (modulo)	Calcola il resto di <i>expression1</i> diviso <i>expression2</i> . Se un operando di <i>expression</i> non è un valore numerico, l'operatore modulo tenta di convertirlo in un numero.
%= (assegnazione modulo)	Assegna a <i>expression1</i> il valore di <i>expression1</i> % <i>expression2</i> .
*= (assegnazione moltiplicazione)	Assegna a <i>expression1</i> il valore di <i>expression1</i> * <i>expression2</i> .
* (moltiplicazione)	Moltiplica due espressioni numeriche.
+ (addizione numerica)	Aggiunge le espressioni numeriche.
== (uguaglianza numerica)	Verifica l'uguaglianza; se <i>expression1</i> è uguale a <i>expression2</i> , il risultato è <i>true</i> .

Operatore	Descrizione
> (maggiore di numerico)	Esegue un confronto tra due espressioni e determina se <i>expression1</i> è maggiore di <i>expression2</i> ; in caso affermativo, l'operatore restituisce <i>true</i> . Se <i>expression1</i> è minore di o uguale a <i>expression2</i> , l'operatore restituisce <i>false</i> .
>= (maggiore di o uguale a numerico)	Confronta due espressioni e determina se <i>expression1</i> è maggiore di o uguale a <i>expression2</i> (<i>true</i>) o se <i>expression1</i> è minore di <i>expression2</i> (<i>false</i>).
<> (disuguaglianza numerica)	Verifica la disuguaglianza; se <i>expression1</i> è uguale a <i>expression2</i> , il risultato è <i>false</i> .
< (minore di numerico)	Esegue un confronto tra due espressioni e determina se <i>expression1</i> è minore di <i>expression2</i> ; in caso affermativo, l'operatore restituisce <i>true</i> . Se <i>expression1</i> è maggiore di o uguale a <i>expression2</i> , l'operatore restituisce <i>false</i> .
<= (minore di o uguale a numerico)	Confronta due espressioni e determina se <i>expression1</i> è minore di o uguale a <i>expression2</i> . In caso affermativo, l'operatore restituisce <i>true</i> ; altrimenti, restituisce <i>false</i> .
() (parentesi)	Raggruppa uno o più parametri ed esegue una valutazione sequenziale delle espressioni, oppure racchiude uno o più parametri e li trasmette come tali a una funzione fuori delle parentesi.
" " (delimitatore di stringa)	Se utilizzate prima e dopo una sequenza di zero o più caratteri, le virgolette indicano che tali caratteri hanno un valore letterale e che devono essere considerati come una <i>stringa</i> , non una variabile, un valore numerico o un altro elemento di <i>ActionScript</i> .
eq (uguaglianza stringhe)	Confronta due espressioni per verificarne l'uguaglianza e restituisce <i>true</i> se la rappresentazione sotto forma di stringa di <i>expression1</i> è uguale alla rappresentazione sotto forma di stringa di <i>expression2</i> ; in caso contrario, restituisce <i>false</i> .
gt (stringa maggiore di)	Confronta la rappresentazione sotto forma di stringa di <i>expression1</i> con la rappresentazione sotto forma di stringa di <i>expression2</i> e restituisce <i>true</i> se <i>expression1</i> è maggiore di <i>expression2</i> ; in caso contrario, restituisce <i>false</i> .
ge (stringa maggiore di o uguale a)	Confronta la rappresentazione sotto forma di stringa di <i>expression1</i> con la rappresentazione sotto forma di stringa di <i>expression2</i> e restituisce <i>true</i> se <i>expression1</i> è maggiore di o uguale a <i>expression2</i> ; in caso contrario, restituisce <i>false</i> .
ne (disuguaglianza di stringa)	Confronta le rappresentazioni sotto forma di stringa di <i>expression1</i> con <i>expression2</i> e restituisce <i>true</i> se <i>expression1</i> non è uguale a <i>expression2</i> ; in caso contrario, restituisce <i>false</i> .

Operatore	Descrizione
<code>lt</code> (stringa minore di)	Confronta la rappresentazione sotto forma di stringa di <i>expression1</i> con la rappresentazione sotto forma di stringa di <i>expression2</i> e restituisce <code>true</code> se <i>expression1</i> è minore di <i>expression2</i> ; in caso contrario, restituisce <code>false</code> .
<code>le</code> (stringa minore di o uguale a)	Confronta la rappresentazione sotto forma di stringa di <i>expression1</i> con la rappresentazione sotto forma di stringa di <i>expression2</i> e restituisce <code>true</code> se <i>expression1</i> è minore di o uguale a <i>expression2</i> ; in caso contrario, restituisce <code>false</code> .
<code>-</code> (sottrazione)	Utilizzato per negare o sottrarre.
<code>-=</code> (assegnazione sottrazione)	Assegna a <i>expression1</i> il valore di <i>expression1</i> - <i>expression2</i> .

add (concatenazione stringhe)

Disponibilità

Flash Lite 1.0.

Uso

```
string1 add string2
```

Operandi

string1, *string2* Stringhe.

Descrizione

Operatore; concatena (combina) due o più stringhe.

Esempio

L'esempio seguente combina due valori di stringa per produrre la stringa *catalog*.

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

Vedere anche

`+` (addizione numerica)

+= (assegnazione addizione)

Disponibilità

Flash Lite 1.0.

Uso

expression1 += *expression2*

Operandi

expression1, *expression2* Numeri di stringhe.

Descrizione

Operatore (assegnazione aritmetica composta); assegna a *expression1* il valore di *expression1* + *expression2*. Ad esempio, le due istruzioni seguenti danno lo stesso risultato:

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

Tutte le regole dell'operatore di addizione (+) si applicano anche all'operatore di assegnazione addizione (+=).

Esempio

L'esempio seguente utilizza l'operatore di assegnazione addizione (+=) per aumentare il valore di x del valore di y:

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

Vedere anche

[+ \(addizione numerica\)](#)

and

Disponibilità

Flash Lite 1.0.

Uso

condition1 and *condition2*

Operandi

condition1, *condition2* Condizioni o espressioni che restituiscono true o false.

Descrizione

Operatore; esegue un'operazione AND logica.

Esempio

L'esempio seguente utilizza l'operatore `and` per verificare se un giocatore ha vinto. La variabile `turns` e la variabile `score` vengono aggiornate quando un giocatore comincia il proprio turno o realizza dei punti durante il gioco. Lo script seguente mostra la scritta "You Win the Game!" nel pannello Output quando il punteggio del giocatore raggiunge 75 o più punti in tre o meno turni.

```
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!
```

Vedere anche

[&& \(AND logico\)](#)

= (assegnazione)

Disponibilità

Flash Lite 1.0.

Uso

expression1 = *expression2*

Operandi

expression1 Una variabile o una proprietà.

expression2 Un valore.

Descrizione

Operatore; assegna il valore di *expression2* (l'operando a destra) alla variabile o alla proprietà in *expression1*.

Esempio

L'esempio seguente utilizza l'operatore di assegnazione (=) per assegnare un valore numerico alla variabile `weight`:

```
weight = 5;
```

L'esempio seguente utilizza l'operatore di assegnazione (=) per assegnare un valore di stringa alla variabile `greeting`:

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

/* (commento a blocchi)

Disponibilità

Flash Lite 1.0

Uso

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

Operandi

comment Qualsiasi carattere.

Descrizione

Delimitatore commenti; indica una o più righe di commento di script. Tutti i caratteri racchiusi tra il tag di inizio commento (*/**) e il tag di fine commento (**/*) vengono considerati come commenti e ignorati dall'interprete di ActionScript.

Utilizzare il delimitatore commenti *//* per identificare i commenti su una sola riga. Utilizzare il delimitatore commenti */** per identificare i commenti su più righe successive. Quando si utilizza questo formato di delimitazione, se si omette il tag di fine commento (**/*) o si cerca di nidificare i commenti, viene restituito un messaggio di errore.

Dopo che è stato utilizzato un tag di inizio commento (*/**), il primo tag di fine commento (**/*) determina la fine del commento, indipendentemente dal numero di tag di inizio commento (*/**) situati tra il primo tag di inizio e il tag di fine.

Vedere anche

[// \(commento\)](#)

, (virgola)

Disponibilità

Flash Lite 1.0.

Uso

expression1, *expression2*

Operandi

expression1, *expression2* Numeri o espressioni che restituiscono numeri.

Descrizione

Operatore; valuta *expression1* e successivamente *expression2*, quindi restituisce il valore di *expression2*.

Esempio

L'esempio seguente utilizza l'operatore virgola (,) senza l'operatore parentesi () e mostra come l'operatore virgola restituisce solo il valore della prima espressione senza l'operatore parentesi ():

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

L'esempio seguente utilizza l'operatore virgola (,) con l'operatore parentesi () e mostra come l'operatore virgola restituisce il valore dell'ultima espressione se utilizzato con l'operatore parentesi ():

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

L'esempio seguente utilizza l'operatore virgola (,) senza l'operatore parentesi () e mostra come l'operatore virgola valuta in sequenza tutte le espressioni ma restituisce il valore della prima espressione. La seconda espressione, z++, viene valutata e z viene incrementato di uno.

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

L'esempio seguente è identico all'esempio precedente a eccezione dell'aggiunta dell'operatore parentesi () e mostra nuovamente come l'operatore virgola (,), se utilizzato con l'operatore parentesi (), restituisce il valore dell'ultima espressione nella serie:

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

Vedere anche

[for, \(\) \(parentesi\)](#)

// (commento)

Disponibilità

Flash Lite 1,0

Uso

```
// comment
```

Operandi

comment Qualsiasi carattere.

Descrizione

Delimitatore di commento; indica l'inizio di un commento di script. Tutti i caratteri racchiusi tra il delimitatore (`//`) e il carattere di fine riga vengono considerati commenti e ignorati dall'interprete di ActionScript.

Esempio

L'esempio seguente utilizza i delimitatori commenti per identificare la prima, terza, quinta e settima riga come commenti:

```
// Registra la posizione X del clip filmato della palla.  
ballX = ball._x;  
// Registra la posizione Y del clip filmato della palla.  
ballY = ball._y;  
// Registra la posizione X del clip filmato della mazza.  
batX = bat._x;  
// Registra la posizione Y del clip filmato della mazza.  
batY = bat._y;
```

Vedere anche

[/* \(commento a blocchi\)](#)

?: (condizionale)

Disponibilità

Flash Lite 1.0.

Uso

expression1 ? *expression2* : *expression3*

Operandi

expression1 Espressione che restituisce un valore booleano; solitamente un'espressione di confronto, come `x < 5`.

expression2, *expression3* Valori di qualsiasi tipo.

Descrizione

Operatore; indica a Flash di valutare *expression1* e, se il valore di *expression1* è `true`, restituisce il valore di *expression2*; altrimenti restituisce il valore di *expression3*.

Esempio

L'esempio seguente assegna il valore della variabile `x` alla variabile `z`, poiché `expression1` restituisce `true`:

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

-- (decremento)

Disponibilità

Flash Lite 1.0.

Uso

`--expression`

`expression--`

Operandi

Nessuno.

Descrizione

Operatore (aritmetico); operatore unario di decremento prima e dopo l'operazione, che sottrae 1 da `expression`. Il formato di decremento prima dell'operazione (`--expression`) sottrae 1 da `expression` e restituisce il risultato sotto forma di numero. Il formato di decremento dopo l'operazione (`expression--`) sottrae 1 da `expression` e restituisce il valore iniziale di `expression` (il valore prima della sottrazione).

Esempio

L'esempio seguente mostra il decremento dell'operatore prima dell'operazione, decrementando `aWidth` a 2 (`aWidth - 1 = 2`) e restituendo il risultato come `bWidth`:

```
aWidth = 3;
bWidth = --aWidth;
// Il valore bWidth è uguale a 2.
```

L'esempio seguente mostra il decremento dell'operatore dopo l'operazione, decrementando `aWidth` a 2 (`aWidth - 1 = 2`) e restituendo il valore originale di `aWidth` come `bWidth`:

```
aWidth = 3;
bWidth = aWidth--;
// Il valore bWidth è uguale a 3.
```

/ (divisione)

Disponibilità

Flash Lite 1.0.

Uso

expression1 / *expression2*

Operandi

expression1, *expression2* Numeri o espressioni che restituiscono numeri.

Descrizione

Operatore (aritmetico); divide *expression1* per *expression2*. Il risultato della divisione corrisponde a un numero a virgola mobile e doppia precisione.

Esempio

L'istruzione seguente divide il numero a virgola mobile 22.0 per 7.0 e visualizza il risultato nel pannello Output:

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

Il risultato è un numero a virgola mobile: 3.1429.

/= (assegnazione divisione)

Disponibilità

Flash Lite 1.0.

Uso

expression1 /= *expression2*

Operandi

expression1, *expression2* Numeri o espressioni che restituiscono numeri.

Descrizione

Operatore (assegnazione aritmetica composta); assegna a *expression1* il valore di *expression1* / *expression2*. Ad esempio, le due istruzioni seguenti sono equivalenti:

```
x /= y
```

```
x = x / y
```

Esempio

L'esempio seguente utilizza l'operatore /= con variabili e numeri:

```
x = 10;
y = 2;
x /= y;
// L'espressione x ora contiene il valore 5.
```

. (punto)

Disponibilità

Flash Lite 1.0.

Uso

instancename.variable

instancename.childinstance.variable

Operandi

instancename Il nome dell'istanza di un clip filmato.

childinstance Un'istanza di clip filmato secondario, o nidificato, rispetto a un altro clip filmato.

variable Una variabile sulla linea temporale del nome dell'istanza di clip filmato specificato.

Descrizione

Operatore; utilizzato per spostarsi nelle gerarchie dei clip filmato e accedere ai clip filmato nidificati (secondari), alle variabili e alle proprietà.

Esempio

L'esempio seguente identifica il valore corrente della variabile `hairColor` nel clip filmato

`person_mc:`

`person_mc.hairColor`

Equivale alla seguente sintassi di notazione della barra:

`/person_mc:hairColor`

Vedere anche

[/ \(Barra\)](#)

++ (incremento)

Disponibilità

Flash Lite 1.0.

Uso

`++expression`

`expression++`

Operandi

Nessuno.

Descrizione

Operatore (aritmetico); operatore unario di incremento prima e dopo l'operazione, che aggiunge 1 a *expression*. Il valore di *expression* può essere una variabile, un elemento di un array o la proprietà di un oggetto. Il formato di incremento prima dell'operazione (`++expression`) aggiunge 1 a *expression* e restituisce il risultato sotto forma di numero. Il formato di incremento dopo l'operazione (`expression++`) aggiunge 1 a *expression* e restituisce il valore iniziale di *expression* (il valore prima dell'addizione).

Esempio

L'esempio seguente utilizza ++ come operatore di incremento dopo l'operazione per far eseguire cinque volte un ciclo `while`:

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

L'esempio seguente utilizza ++ come operatore di incremento prima dell'operazione:

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

Questo script mostra il risultato seguente nel pannello Output:

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

L'esempio seguente utilizza ++ come operatore di incremento dopo l'operazione:

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

Questo script mostra il risultato seguente nel pannello Output:

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

&& (AND logico)

Disponibilità

Flash Lite 1.0.

Uso

```
expression1 && expression2
```

Operandi

expression1, *expression2* Valori booleani o espressioni che convertono in valori booleani.

Descrizione

Operatore (logico); esegue un'operazione booleana sui valori di un'espressione o di entrambe. L'operatore valuta *expression1* (l'espressione a sinistra dell'operatore) e restituisce *false* se l'espressione equivale a *false*. Se *expression1* restituisce *true*, viene valutata *expression2* (l'espressione a destra dell'operatore). Se *expression2* restituisce *true*, il risultato finale è *true*; in caso contrario, è *false*.

Esempio

L'esempio seguente utilizza l'operatore && per eseguire una prova che determini se un giocatore ha vinto. La variabile `turns` e la variabile `score` vengono aggiornate quando un giocatore comincia il proprio turno o realizza dei punti durante il gioco. Lo script seguente mostra la scritta "You Win the Game!" nel pannello Output quando il punteggio del giocatore raggiunge 75 o più punti in tre o meno turni.

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

L'esempio seguente verifica che una posizione *x* immaginaria sia compresa in un intervallo:

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

! (NOT logico)

Disponibilità

Flash Lite 1.0.

Uso

!expression

Operandi

Nessuno.

Descrizione

Operatore (logico); inverte il valore booleano di una variabile o espressione. Se *expression* è una variabile con un valore convertito o assoluto *true*, il valore di *!expression* è *false*.

Se l'espressione *x && y* restituisce *false*, l'espressione *!(x && y)* restituisce *true*.

Le espressioni seguenti mostrano il risultato dell'uso dell'operatore **!**:

!true restituisce *false*

!false restituisce *true*

Esempio

Nell'esempio seguente, la variabile *happy* è impostata su *false*. La condizione *if* restituisce la condizione *!happy*, e se la condizione è *true*, la funzione `trace()` invia una stringa al pannello Output.

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

|| (OR logico)

Disponibilità

Flash Lite 1.0.

Uso

expression1 || *expression2*

Operandi

expression1, *expression2* Valori booleani o espressioni che convertono in valori booleani.

Descrizione

Operatore (logico); valuta *expression1* ed *expression2*. Il risultato è *true* se almeno una delle espressioni restituisce *true*; il risultato è *false* solo se entrambe le espressioni restituiscono *false*. È possibile utilizzare l'operatore logico OR con un numero qualsiasi di operandi; se uno degli operandi restituisce *true*, il risultato è *true*.

Con le espressioni non booleane, l'operatore OR logico fa in modo che Flash Lite valuti l'espressione a sinistra; se può essere convertita in *true*, il risultato è *true*. Altrimenti, valuta l'espressione a destra e il risultato è il valore di tale espressione.

Esempio

Uso 1: l'esempio seguente utilizza l'operatore || in un'istruzione *if*. La seconda espressione restituisce *true*, quindi il risultato finale è *true*:

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

% (modulo)

Disponibilità

Flash Lite 1.0.

Uso

expression1 % *expression2*

Operandi

expression1, *expression2* Numeri o espressioni che restituiscono numeri.

Descrizione

Operatore (aritmetico); calcola il resto di *expression1* diviso per *expression2*. Se un operando di *expression* non è un valore numerico, l'operatore modulo tenta di convertirlo in un numero. *expression* può essere un numero o una stringa convertibile in un valore numerico.

Quando si imposta Flash Lite 1.0 o 1.1 come destinazione, il compilatore Flash espande l'operatore % nel file SWF pubblicato mediante la formula seguente:

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

L'esecuzione di questa approssimazione può non essere veloce e precisa come nelle versioni di Flash Player che supportano a livello nativo l'operatore modulo.

Esempio

L'esempio numerico seguente utilizza l'operatore modulo (%):

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

%= (assegnazione modulo)

Disponibilità

Flash Lite 1.0.

Uso

expression1 %= *expression2*

Operandi

expression1, *expression2* Numeri o espressioni che restituiscono numeri.

Descrizione

Operatore (assegnazione aritmetica composta); assegna a *expression1* il valore di *expression1* % *expression2*. Ad esempio, le due espressioni seguenti sono equivalenti:

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

Esempio

l'esempio seguente assegna il valore 4 alla variabile x:

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

Vedere anche

[% \(modulo\)](#)

*= (assegnazione moltiplicazione)

Disponibilità

Flash Lite 1.0.

Uso

```
expression1 *= expression2
```

Operandi

expression1, *expression2* Numeri o espressioni che restituiscono numeri.

Descrizione

Operatore (assegnazione aritmetica composta); assegna a *expression1* il valore di *expression1* * *expression2*.

Ad esempio, le due espressioni seguenti sono equivalenti:

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

Esempio

Uso 1: l'esempio seguente assegna il valore 50 alla variabile x:

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

Uso 2: la seconda e terza riga dell'esempio seguente calcolano le espressioni a destra del segno di uguale (=) e assegnano i risultati a x e y:

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

* (moltiplicazione)

Disponibilità

Flash Lite 1.0.

Uso

expression1 * *expression2*

Operandi

expression1, *expression2* Espressioni numeriche.

Descrizione

Operatore (aritmetico); moltiplica due espressioni numeriche. Se entrambe le espressioni sono numeri interi, il prodotto è un numero intero. Se almeno una delle due espressioni è un numero a virgola mobile, il prodotto è un numero a virgola mobile.

Esempio

Uso 1: l'istruzione seguente moltiplica gli interi 2 e 3:

```
2 * 3
```

Il risultato è un numero intero: 6.

Uso 2: l'istruzione moltiplica i numeri a virgola mobile 2.0 e 3.1416:

```
2.0 * 3.1416
```

Il risultato è un numero a virgola mobile: 6.2832.

+ (addizione numerica)

Disponibilità

Flash Lite 1.0.

Uso

expression1 + *expression2*

Operandi

expression1, *expression2* Numeri.

Descrizione

Operatore; aggiunge le espressioni numeriche. L'operatore + è solo numerico; non può essere utilizzato per la concatenazione di stringhe.

Se entrambe le espressioni sono numeri interi, la somma è un numero intero; se almeno una delle due espressioni è un numero a virgola mobile, la somma è un numero a virgola mobile.

Esempio

L'esempio seguente aggiunge ai numeri interi 2 e 3; il numero intero risultante 5 viene visualizzato nel pannello Output:

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

L'esempio seguente aggiunge i numeri a virgola mobile 2.5 e 3.25; il risultato è il numero a virgola mobile 5.75 e viene visualizzato nel pannello Output:

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

Vedere anche

[add \(concatenazione stringhe\)](#)

== (uguaglianza numerica)

Disponibilità

Flash Lite 1.0.

Uso

```
expression1 == expression2
```

Operandi

expression1, *expression2* Numeri, valori booleani o variabili.

Descrizione

Operatore (confronto); prova l'uguaglianza ed è l'esatto opposto dell'operatore <>. Se *expression1* è uguale a *expression2*, il risultato è `true`. Come nel caso dell'operatore <>, la definizione di *uguale* dipende dai tipi di dati confrontati:

- I numeri e i valori booleani vengono confrontati in base al valore.
- Le variabili vengono confrontate in base al riferimento.

Esempio

Gli esempi seguenti restituiscono i valori `true` e `false`:

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

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

Vedere anche

[eq \(uguaglianza stringhe\)](#)

> (maggiore di numerico)

Disponibilità

Flash Lite 1.0.

Uso

expression1 > *expression2*

Operandi

expression1, *expression2* Numeri o espressioni che restituiscono numeri.

Operatore (confronto); esegue un confronto tra due espressioni e determina se *expression1* è maggiore di *expression2*; in caso affermativo, l'operatore restituisce *true*. Se *expression1* è minore di o uguale a *expression2*, l'operatore restituisce *false*.

Esempio

Gli esempi seguenti restituiscono i valori *true* e *false* per i confronti numerici:

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

Vedere anche

[gt \(stringa maggiore di\)](#)

>= (maggiore di o uguale a numerico)

Disponibilità

Flash Lite 1.0.

Uso

expression1 >= *expression2*

Operandi

expression1, *expression2* Numero intero o numeri a virgola mobile.

Descrizione

Operatore (confronto); esegue un confronto tra due espressioni e determina se *expression1* è maggiore di o uguale a *expression2* (*true*) o se *expression1* è minore di *expression2* (*false*)

Esempio

Gli esempi seguenti restituiscono i valori true e false:

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

Vedere anche

[ge \(stringa maggiore di o uguale a\)](#)

⟷ (disuguaglianza numerica)

Disponibilità

Flash Lite 1.0.

Uso

```
expression1 <> expression2
```

Operandi

expression1, *expression2* Numeri, valori booleani o variabili.

Descrizione

Operatore (uguaglianza); prova la disuguaglianza ed è l'esatto opposto dell'operatore di uguaglianza (==). Se *expression1* è uguale a *expression2*, il risultato è false. Come nel caso dell'operatore di uguaglianza (==), la definizione di *uguale* dipende dai tipi di dati confrontati:

- I numeri e i valori booleani vengono confrontati in base al valore.
- Le variabili vengono confrontate in base al riferimento.

Esempio

Gli esempi seguenti restituiscono i valori true e false:

```
trees = 7;
B = "7";

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

Vedere anche

[ne \(disuguaglianza di stringa\)](#)

< (minore di numerico)

Disponibilità

Flash Lite 1.0.

Uso

expression1 < *expression2*

Operandi

expression1, *expression2* Numeri.

Descrizione

Operatore (confronto); esegue un confronto tra due espressioni e determina se *expression1* è minore di *expression2*; in caso affermativo, l'operatore restituisce `true`. Se *expression1* è minore o uguale a *expression2*, l'operatore restituisce `false`. L'operatore < (minore di) è un operatore numerico.

Esempio

Gli esempi seguenti mostrano i risultati `true` e `false` sia per i confronti numerici che per quelli di stringa:

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

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

Vedere anche

[lt \(stringa minore di\)](#)

<= (minore di o uguale a numerico)

Flash Lite 1.0.

Uso

expression1 <= *expression2*

Operandi

expression1, *expression2* Numeri.

Descrizione

Operatore (confronto); esegue un confronto tra due espressioni e determina se *expression1* è minore di o uguale a *expression2*. In caso affermativo, l'operatore restituisce `true`; altrimenti, restituisce `false`. Questo operatore viene utilizzato solo per i confronti numerici.

Esempio

Gli esempi seguenti restituiscono i valori true e false per i confronti numerici:

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

Vedere anche

[Le \(stringa minore di o uguale a\)](#)

() (parentesi)

Disponibilità

Flash Lite 1.0.

Uso

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

expression1, *expression2* Numeri, stringhe, variabili o testo.

parameter1, ..., *parameterN* Una serie di parametri da eseguire prima che i risultati vengano trasmessi come parametri alla funzione che si trova fuori dalle parentesi.

Descrizione

Operatore; raggruppa uno o più parametri ed esegue una valutazione sequenziale delle espressioni, oppure racchiude uno o più parametri e li trasmette come tali a una funzione fuori delle parentesi.

Uso 1: controlla l'ordine di esecuzione degli operatori nell'espressione. Le parentesi hanno la priorità sul normale ordine di precedenza e le espressioni racchiuse tra parentesi vengono valutate per prime. Quando le parentesi sono nidificate, il contenuto delle parentesi interne viene valutato prima del contenuto di quelle esterne.

Uso 2: valuta in sequenza una serie di espressioni separate da virgola, e restituisce il risultato dell'espressione finale.

Esempio

Uso 1: le istruzioni seguenti mostrano l'uso delle parentesi per controllare l'ordine con cui vengono eseguite le espressioni (il valore di ciascuna espressione viene visualizzato nel pannello Output):

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

Uso 1: le istruzioni seguenti mostrano l'uso delle parentesi per controllare l'ordine con cui vengono eseguite le espressioni (il valore di ciascuna espressione viene scritto nel file di registro):

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

" " (delimitatore di stringa)

Disponibilità

Flash Lite 1.0.

Uso

```
"text"
```

Operandi

text Zero o più caratteri.

Descrizione

Delimitatore di stringa; se utilizzate prima e dopo una sequenza di zero o più caratteri, le virgolette indicano che tali caratteri hanno un valore letterale e che devono essere considerati come una *stringa*, non una variabile, un valore numerico o un altro elemento di ActionScript.

Esempio

L'esempio seguente utilizza le virgolette (") per indicare che il valore della variabile `yourGuess` è il valore letterale di stringa "Prince Edward Island" e non il nome di una variabile. Il valore di `province` è una variabile, non un valore letterale; per determinare il valore di `province`, è necessario individuare il valore di `yourGuess`.

```
yourGuess = "Prince Edward Island";

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

eq (uguaglianza stringhe)

Disponibilità

Flash Lite 1.0.

Uso

expression1 eq *expression2*

Operandi

expression1, *expression2* Numeri, stringhe o variabili.

Descrizione

Operatore (confronto); esegue un confronto tra due espressioni per verificarne l'uguaglianza e restituisce *true* se la rappresentazione sotto forma di stringa di *expression1* è uguale alla rappresentazione sotto forma di stringa di *expression2*; in caso contrario, restituisce *false*.

Esempio

Gli esempi seguenti restituiscono i valori *true* e *false*:

```
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)
```

Vedere anche

[== \(uguaglianza numerica\)](#)

gt (stringa maggiore di)

Disponibilità

Flash Lite 1.0.

Uso

expression1 gt *expression2*

Operandi

expression1, *expression2* Numeri, stringhe o variabili.

Descrizione

Operatore (confronto); esegue un confronto tra la rappresentazione sotto forma di stringa di *expression1* e la rappresentazione sotto forma di stringa di *expression2* e restituisce *true* se *expression1* è maggiore di *expression2*; in caso contrario, restituisce *false*. Le stringhe vengono confrontate in ordine alfabetico; le cifre precedono le lettere e tutte le maiuscole precedono tutte le minuscole.

Esempio

Gli esempi seguenti restituiscono i valori *true* e *false*:

```
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)
```

Vedere anche

> (maggiore di numerico)

ge (stringa maggiore di o uguale a)

Disponibilità

Flash Lite 1.0.

Uso

expression1 ge *expression2*

Operandi

expression1, *expression2* Numeri, stringhe o variabili.

Descrizione

Operatore (confronto); esegue un confronto tra la rappresentazione sotto forma di stringa di *expression1* e la rappresentazione sotto forma di stringa di *expression2* e restituisce *true* se *expression1* è maggiore di o uguale a *expression2*; in caso contrario, restituisce *false*. Le stringhe vengono confrontate in ordine alfabetico; le cifre precedono le lettere e tutte le maiuscole precedono tutte le minuscole.

Esempio

Gli esempi seguenti restituiscono i valori true e false:

```
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)
```

Vedere anche

[>= \(maggiore di o uguale a numerico\)](#)

ne (diseguaglianza di stringa)

Disponibilità

Flash Lite 1.0.

Uso

expression1 ne *expression2*

Operandi

expression1, *expression2* Numeri, stringhe o variabili.

Descrizione

Operatore (confronto); esegue un confronto tra le rappresentazioni sotto forma di stringa di *expression1* e *expression2* e restituisce true se *expression1* non è uguale a *expression2*; in caso contrario, restituisce false.

Esempio

Gli esempi seguenti restituiscono i valori true e false:

```
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)
```

Vedere anche

[<> \(diseguaglianza numerica\)](#)

lt (stringa minore di)

Disponibilità

Flash Lite 1.0.

Uso

expression1 lt *expression2*

Operandi

expression1, *expression2* Numeri, stringhe o variabili.

Descrizione

Operatore (confronto); esegue un confronto tra la rappresentazione sotto forma di stringa di *expression1* e la rappresentazione sotto forma di stringa di *expression2* e restituisce *true* se *expression1* è minore di *expression2*; in caso contrario, restituisce *false*. Le stringhe vengono confrontate in ordine alfabetico; le cifre precedono le lettere e tutte le maiuscole precedono tutte le minuscole.

Esempio

Gli esempi seguenti mostrano l'output di vari confronti di stringa. Nell'ultima riga, *lt* non restituisce un errore quando si confronta una stringa a un numero intero, poiché la sintassi di ActionScript 1.0 tenta di convertire il tipo di dati interi in una stringa e restituisce *false*.

```
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)
```

Vedere anche

[< \(minore di numerico\)](#)

le (stringa minore di o uguale a)

Disponibilità

Flash Lite 1.0.

Uso

expression1 le *expression2*

Operandi

expression1, *expression2* Numeri, stringhe o variabili.

Descrizione

Operatore (confronto); esegue un confronto tra la rappresentazione sotto forma di stringa di *expression1* e la rappresentazione sotto forma di stringa di *expression2* e restituisce true se *expression1* è minore di o uguale a *expression2*; in caso contrario, restituisce false. Le stringhe vengono confrontate in ordine alfabetico; le cifre precedono le lettere e tutte le maiuscole precedono tutte le minuscole.

Esempio

Gli esempi seguenti mostrano l'output di vari confronti di stringa:

```
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)
```

Vedere anche

[<= \(minore di o uguale a numerico\)](#)

- (sottrazione)

Disponibilità

Flash Lite 1.0.

Uso

(Negazione) *-expression*

(Sottrazione) *expression1 - expression2*

Operandi

expression1, *expression2* Numeri o espressioni che restituiscono numeri.

Descrizione

Operatore (aritmetico); utilizzato per la negazione e la sottrazione.

Uso 1: quando viene utilizzato per la negazione, inverte il segno dell'espressione numerica.

Uso 2: quando viene utilizzato per la sottrazione, esegue una sottrazione aritmetica su due espressioni numeriche, sottraendo *expression2* da *expression1*. Se entrambe le espressioni sono numeri interi, la differenza è un numero intero. Se almeno una delle due espressioni è un numero a virgola mobile, la differenza è un numero a virgola mobile.

Esempio

Uso 1: l'istruzione seguente inverte il segno dell'espressione $2 + 3$:

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

Uso 2: l'istruzione seguente sottrae il numero intero 2 dal numero intero 5:

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

Il risultato è un numero intero: 3.

Uso 3: l'istruzione seguente sottrae il numero a virgola mobile 1.5 dal numero a virgola mobile 3.25:

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

Il risultato è un numero a virgola mobile: 1,75.

-= (assegnazione sottrazione)

Disponibilità

Flash Lite 1.0.

Uso

expression1 -= expression2

Operandi

expression1, *expression2* Numeri o espressioni che restituiscono numeri.

Descrizione

Operatore (assegnazione aritmetica composta); assegna a *expression1* il valore di *expression1 - expression2*. Non viene restituito alcun valore.

Ad esempio, le due istruzioni seguenti sono equivalenti:

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

Le espressioni stringa devono essere convertite in numeri; altrimenti viene restituito il valore -1.

Esempio

Uso 1: l'esempio seguente utilizza l'operatore -= per sottrarre 10 da 5 e assegnare il risultato alla variabile x:

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

Uso 2: l'esempio seguente mostra il modo in cui le stringhe vengono convertite in numeri:

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

Elementi di linguaggio specifici di Flash Lite

Questa sezione descrive le variabili e le funzionalità di piattaforma riconosciute da Macromedia Flash Lite 1.1 di Adobe, oltre ai comandi Flash Lite che è possibile eseguire mediante le funzioni `fscommand()` e `fscommand2()`. La funzionalità descritta in questa sezione è specifica di Flash Lite.

Il contenuto di questa sezione è riepilogato nella tabella seguente:

Elemento del linguaggio	Descrizione
<code>_capCompoundSound</code>	Indica se Flash Lite può elaborare i dati audio composti.
<code>_capEmail</code>	Indica se il client Flash Lite può inviare messaggi e-mail mediante il comando <code>GetURL()</code> di ActionScript.
<code>_capLoadData</code>	Indica se l'applicazione host può caricare dinamicamente dati aggiuntivi mediante delle chiamate alle funzioni <code>loadMovie()</code> , <code>loadMovieNum()</code> , <code>loadVariables()</code> e <code>loadVariablesNum()</code> .
<code>_capMFi</code>	Indica se il dispositivo è in grado di riprodurre i dati audio nel formato audio MFi (Melody Format per i-mode).
<code>_capMIDI</code>	Indica se il dispositivo è in grado di riprodurre i dati audio nel formato audio MIDI (Musical Instrument Digital Interface).
<code>_capMMS</code>	Indica se Flash Lite è in grado di inviare messaggi MMS (Multimedia Messaging Service) mediante il comando <code>GetURL()</code> di ActionScript.
<code>_capMP3</code>	Indica se il dispositivo è in grado di riprodurre i dati audio nel formato audio MP3 (MPEG Audio Layer 3).
<code>_capSMAF</code>	Indica se il dispositivo è in grado di riprodurre i file multimediali in formato SMAF (Synthetic Music Mobile Application Format).
<code>_capSMS</code>	Indica se Flash Lite è in grado di inviare messaggi SMS (Short Message Service) mediante il comando <code>GetURL()</code> di ActionScript.
<code>_capStreamSound</code>	Indica se il dispositivo può riprodurre audio in streaming (sincronizzato).

Elemento del linguaggio	Descrizione
<code>_cap4WayKeyAS</code>	Indica se Flash Lite esegue le espressioni ActionScript associate ai gestori di eventi di tastiera associati ai tasti freccia Su, Giù, Pag su e Pag giù.
<code>\$version</code>	Contiene il numero di versione di Flash Lite.
<code>fscommand()</code>	Una funzione utilizzata per eseguire il comando <code>Launch</code> (descritto di seguito).
<code>Launch</code>	(L'unico comando supportato per <code>fscommand()</code>) Consente al file SWF di comunicare con Flash Lite o l'ambiente host (ad esempio, il sistema operativo di un telefono o di un altro dispositivo).
<code>fscommand2()</code>	Una funzione utilizzata per eseguire i comandi di questa tabella, eccetto <code>fscommand()</code> .
<code>Escape</code>	Codifica una stringa arbitraria in un formato sicuro per i trasferimenti in rete.
<code>FullScreen</code>	Imposta le dimensioni dell'area di visualizzazione da utilizzare per il rendering.
<code>GetBatteryLevel</code>	Restituisce il livello corrente della batteria.
<code>GetDateDay</code>	Restituisce il giorno della data corrente sotto forma di valore numerico.
<code>GetDateMonth</code>	Restituisce il mese della data corrente sotto forma di valore numerico.
<code>GetDateWeekday</code>	Restituisce il numero del giorno della data corrente sotto forma di valore numerico.
<code>GetDateYear</code>	Restituisce un valore numerico a quattro cifre che corrisponde all'anno della data corrente.
<code>GetDevice</code>	Imposta un parametro che identifica il dispositivo su cui è in esecuzione Flash Lite.
<code>GetDeviceID</code>	Imposta un parametro che rappresenta l'identificatore univoco del dispositivo (ad esempio, il numero di serie).
<code>GetFreePlayerMemory</code>	Restituisce la quantità di memoria heap, espressa in KB, attualmente disponibile per Flash Lite.
<code>GetLanguage</code>	Imposta un parametro che identifica la lingua attualmente utilizzata dal dispositivo.

Elemento del linguaggio	Descrizione
GetLocaleLongDate	Imposta un parametro su una stringa che rappresenta la data corrente in formato esteso e formattata in base alla versione localizzata attualmente definita.
GetLocaleShortDate	Imposta un parametro su una stringa che rappresenta la data corrente in formato abbreviato e formattata in base alla versione localizzata attualmente definita.
GetLocaleTime	Imposta un parametro su una stringa che rappresenta l'orario corrente, formattato in base alla versione localizzata attualmente definita.
GetMaxBatteryLevel	Restituisce il livello massimo della batteria del dispositivo.
GetMaxSignalLevel	Restituisce il livello di potenza massimo del segnale.
GetMaxVolumeLevel	Restituisce il livello massimo del volume del dispositivo, sotto forma di valore numerico.
GetNetworkConnectStatus	Restituisce un valore che indica lo stato di connessione corrente alla rete.
GetNetworkName	Imposta un parametro sul nome della rete corrente.
GetNetworkRequestStatus	Restituisce un valore che indica lo stato della richiesta HTTP più recente.
GetNetworkStatus	Restituisce un valore che indica lo stato di rete per il telefono (ovvero se è registrata una rete e se il telefono è attualmente in roaming).
GetPlatform	Imposta un parametro che identifica la piattaforma corrente, che descrive in generale la classe del dispositivo. Per i dispositivi con sistemi operativi aperti, l'identificatore di solito indica il nome e la versione del sistema operativo.
GetPowerSource	Restituisce un valore che indica se l'alimentazione è attualmente fornita da una batteria o da una fonte di alimentazione esterna.
GetSignalLevel	Restituisce la potenza corrente del segnale, sotto forma di valore numerico.
GetTimeHours	Restituisce l'ora dell'orario corrente del giorno, nel formato a 24 ore, sotto forma di valore numerico.
GetTimeMinutes	Restituisce il minuto dell'orario corrente del giorno, sotto forma di valore numerico.
GetTimeSeconds	Restituisce il secondo dell'orario corrente del giorno, sotto forma di valore numerico.

Elemento del linguaggio	Descrizione
GetTimeZoneOffset	Imposta il parametro su un numero di minuti compreso tra il fuso orario locale e l'ora universale (UTC).
GetTotalPlayerMemory	Restituisce la quantità totale di memoria heap, espressa in kilobyte, disponibile per Flash Lite.
GetVolumeLevel	Restituisce il livello corrente del volume del dispositivo, sotto forma di valore numerico.
Quit	Indica a Flash Lite Player di interrompere la riproduzione e uscire.
ResetSoftKeys	Ripristina le impostazioni originali dei tasti virtuali.
SetInputTextType	Specifica la modalità di apertura dei campi di testo di input.
SetQuality	Imposta la qualità di rendering dell'animazione.
SetSoftKeys	Mappa nuovamente i tasti virtuali sinistro e destro del dispositivo, a condizione che siano accessibili e modificabili.
StartVibrate	Avvia la vibrazione del telefono.
StopVibrate	Interrompe la vibrazione del telefono, se è attualmente in funzione.
Unescape	Decodifica una stringa arbitraria che è stata codificata per essere sicura per i trasferimenti in rete, e la riporta al normale formato non codificato.

Funzionalità

Questa sezione descrive le funzionalità e le variabili di piattaforma riconosciute da Macromedia Flash Lite 1.1. Le voci sono elencate in ordine alfabetico, senza tener conto del carattere di sottolineatura (_) iniziale.

_capCompoundSound

Disponibilità

Flash Lite 1.1.

Uso

_capCompoundSound

Descrizione

Variabile numerica; indica se Flash Lite può elaborare i dati audio composti. In tal caso, questa variabile è definita e ha il valore 1; in caso contrario, è undefined.

Ad esempio, un file Flash singolo può contenere lo stesso suono rappresentato in formato sia MIDI che MFi. Il lettore riproduce i dati nel formato appropriato in base al formato supportato dal dispositivo. Questa variabile definisce se Flash Lite Player supporta questa funzione per il microtelefono corrente.

Nell'esempio seguente, useCompoundSound viene impostato su 1 in Flash Lite 1.1, ma viene lasciato undefined in Flash Lite 1.0:

```
useCompoundSound = _capCompoundSound;
```

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

_capEmail

Disponibilità

Flash Lite 1.1.

Uso

`_capEmail`

Descrizione

Variabile numerica; indica se il client Flash Lite può inviare messaggi e-mail mediante il comando `GetURL()` di ActionScript. In tal caso, questa variabile è definita e ha il valore 1; in caso contrario, è `undefined`.

Esempio

Se l'applicazione host è in grado di inviare messaggi e-mail mediante il comando `GetURL()` di ActionScript, l'esempio seguente imposta `canEmail` su 1:

```
canEmail = _capEmail;
```

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

_capLoadData

Disponibilità

Flash Lite 1.1.

Uso

`_capLoadData`

Descrizione

Variabile numerica; indica se l'applicazione host può caricare dinamicamente dati aggiuntivi mediante delle chiamate alle funzioni `loadMovie()`, `loadMovieNum()`, `loadVariables()` e `loadVariablesNum()`. In tal caso, questa variabile è definita e ha il valore 1; in caso contrario, è `undefined`.

Esempio

Se l'applicazione host è in grado di effettuare il caricamento dinamico di filmati e variabili, l'esempio seguente imposta `iCanLoad` su 1:

```
canLoad = _capLoadData;

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

_capMFi

Disponibilità

Flash Lite 1.1.

Uso

`_capMFi`

Descrizione

Variabile numerica; indica se il dispositivo è in grado di riprodurre i dati audio nel formato audio MFi (Melody Format per i-mode). In tal caso, questa variabile è definita e ha il valore 1; in caso contrario, è undefined.

Esempio

Se il dispositivo è in grado di riprodurre i dati audio MFi, l'esempio seguente imposta `canMFi` su 1:

```
canMFi = _capMFi;

if (canMFi == 1) {
    // invia i pulsanti del clip filmato al fotogramma con i pulsanti
    // che attivano l'audio degli eventi
    tellTarget("buttons") {
        gotoAndPlay(2);
    }
}
```

_capMIDI

Disponibilità

Flash Lite 1.1.

Uso

_capMIDI

Descrizione

Variabile numerica; indica se il dispositivo è in grado di riprodurre i dati audio nel formato audio MIDI (Musical Instrument Digital Interface). In tal caso, questa variabile è definita e ha il valore 1; in caso contrario, è undefined.

Esempio

Se il dispositivo è in grado di riprodurre i dati audio MIDI, l'esempio seguente imposta canMidi su 1:

```
canMIDI = _capMIDI;
```

```
if (canMIDI == 1) {  
    // invia i pulsanti del clip filmato al fotogramma con i pulsanti  
    // che attivano l'audio degli eventi  
    tellTarget("buttons") {  
        gotoAndPlay(2);  
    }  
}
```

_capMMS

Disponibilità

Flash Lite 1.1.

Uso

_capMMS

Descrizione

Variabile numerica; indica se Flash Lite è in grado di inviare messaggi MMS (Multimedia Messaging Service) mediante il comando `GetURL()` di ActionScript. In tal caso, questa variabile è definita e ha il valore 1; in caso contrario, è undefined.

Esempio

L'esempio seguente imposta `canMMS` su 1 in Flash Lite 1.1, ma lo lascia undefined in Flash Lite 1.0 (tuttavia, non tutti i telefoni dotati di Flash Lite 1.1 sono in grado di inviare messaggi MMS, pertanto questo codice dipende sempre dall'apparecchio):

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

_capMP3

Disponibilità

Flash Lite 1.1.

Uso

`_capMP3`

Descrizione

Variabile numerica; indica se il dispositivo è in grado di riprodurre i dati audio nel formato audio MP3 (MPEG Audio Layer 3). In tal caso, questa variabile è definita e ha il valore 1; in caso contrario, è undefined.

Esempio

Se il dispositivo è in grado di riprodurre i dati audio MP3, l'esempio seguente imposta `canMP3` su 1:

```
canMP3 = _capMP3;
if (canMP3 == 1) {
    tellTarget("soundClip") {
        gotoAndPlay(2);
    }
}
```

_capSMAF

Disponibilità

Flash Lite 1.1.

Uso

_capSMAF

Descrizione

Variabile numerica; indica se il dispositivo è in grado di riprodurre i file multimediali in formato SMAF (Synthetic Music Mobile Application Format). In tal caso, questa variabile è definita e ha il valore 1; in caso contrario, è undefined.

Esempio

L'esempio seguente imposta canSMAF su 1 in Flash Lite 1.1, ma lo lascia undefined in Flash Lite 1.0 (tuttavia, non tutti i telefoni dotati di Flash Lite 1.1 sono in grado di inviare messaggi SMAF, pertanto questo codice dipende sempre dall'apparecchio):

```
canSMAF = _capSMAF;
```

```
if (canSMAF) {  
    // invia i pulsanti del clip filmato al fotogramma con i pulsanti  
    // che attivano l'audio degli eventi  
    tellTarget("buttons") {  
        gotoAndPlay(2);  
    }  
}
```

_capSMS

Disponibilità

Flash Lite 1.1.

Uso

_capSMS

Descrizione

Variabile numerica; indica se Flash Lite è in grado di inviare messaggi *Short Message Service* (SMS) mediante il comando `GetURL()` di ActionScript. In tal caso, questa variabile è definita e ha il valore 1; in caso contrario, è undefined.

Esempio

L'esempio seguente imposta `canSMS` su 1 in Flash Lite 1.1, ma lo lascia undefined in Flash Lite 1.0 (tuttavia, non tutti i telefoni dotati di Flash Lite 1.1 sono in grado di inviare messaggi SMS, pertanto questo codice dipende sempre dall'apparecchio):

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

_capStreamSound

Disponibilità

Flash Lite 1.1.

Uso

`_capStreamSound`

Descrizione

Variabile numerica; indica se il dispositivo può riprodurre audio in streaming (sincronizzato). In tal caso, questa variabile è definita e ha il valore 1; in caso contrario, è undefined.

Esempio

L'esempio seguente riproduce l'audio in streaming se `canStreamSound` è abilitato:

```
on(press) {
    canStreamSound = _capStreamSound;
    if (canStreamSound) {
        // riproduci audio in streaming in un clip filmato con questo pulsante
        tellTarget("music") {
            gotoAndPlay(2);
        }
    }
}
```

_cap4WayKeyAS

Disponibilità

Flash Lite 1.1.

Uso

_cap4WayKeyAS

Descrizione

Variabile numerica; indica se Flash Lite esegue le espressioni ActionScript associate ai gestori di eventi di tastiera associati ai tasti freccia Su, Giù, Pag su e Pag giù. Questa variabile è definita e ha un valore 1 solo quando l'applicazione host utilizza la modalità di navigazione a quattro direzioni per spostarsi tra i controlli di Flash (pulsanti e campi di testo di input). In caso contrario, questa variabile è undefined.

Quando uno dei tasti a quattro direzioni viene premuto, se il valore di questa variabile è 1, Flash Lite cerca innanzi tutto il gestore per il tasto. Se non ne viene trovato nessuno, viene utilizzata la navigazione mediante i controlli di Flash. Tuttavia, se viene trovato un gestore di eventi, per il tasto non viene eseguita alcuna azione di navigazione. Ad esempio, se viene trovato un gestore relativo alla pressione del tasto freccia Giù, l'utente non può navigare.

Esempio

L'esempio seguente imposta `canUse4Way` su 1 in Flash Lite 1.1, ma lo lascia undefined in Flash Lite 1.0 (tuttavia, non tutti i telefoni dotati di Flash Lite 1.1 supportano i tasti a quattro direzioni, pertanto questo codice dipende sempre dall'apparecchio):

```
canUse4Way = _cap4WayKeyAS;
if (canUse4Way == 1) {
    msg = "Use your directional joypad 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

Disponibilità

Flash Lite 1.1.

Uso

```
$version
```

Descrizione

Variabile stringa; contiene il numero di versione di Flash Lite. In realtà, sono indicati un numero di versione principale, un numero di versione minore, il numero della build e un numero di build interno, che generalmente è 0 in tutte le versioni rilasciate al pubblico.

Il numero di versione principale riportato per tutti i prodotti Flash Lite 1.x è 5. Flash Lite 1.0 ha il numero di versione minore 1; Flash Lite 1.1 ha il numero di versione minore 2.

Esempio

In Flash Lite Player 1.1, il codice seguente imposta il valore di `myVersion` su "5, 2, 12, 0":

```
myVersion = $version;
```

fscommand()

Disponibilità

Flash Lite 1.1.

Uso

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

Parametri

"Launch" L'identificatore del comando. Il comando `Launch` è l'unico comando che viene eseguito dalla funzione `fscommand()`.

"*percorso-applicazione, arg1, arg2, ..., argn*" Il nome dell'applicazione che si sta avviando e i relativi parametri, separati da virgole.

Descrizione

Funzione; consente al file SWF di comunicare con Flash Lite o l'ambiente host (ad esempio, il sistema operativo di un telefono o di un altro dispositivo).

Vedere anche

[fscommand2\(\)](#)

Launch

Disponibilità

Flash Lite 1.1.

Uso

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

Parametri

"Launch" L'identificatore del comando. In Flash Lite, si utilizza la funzione `fscommand()` solo per eseguire il comando `Launch`.

"percorso-applicazione, *arg1*, *arg2*, ..., *argn*" Il nome dell'applicazione che si sta avviando e i relativi parametri, separati da virgole.

Descrizione

Comando eseguito mediante la funzione `fscommand()`; avvia un'altra applicazione sul dispositivo. Il nome dell'applicazione da avviare e i relativi parametri vengono passati in un unico argomento.

NOTA

Questa funzione dipende dal sistema operativo.

Questo comando è supportato solo quando Flash Lite Player è in esecuzione in modalità autonoma, e non quando è in esecuzione nel contesto di un'altra applicazione (ad esempio, come plug-in di un browser).

Esempio

L'esempio seguente apre `wap.yahoo.com` nel browser servizi/Web sui telefoni della serie 60:

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

Vedere anche

[fscommand2\(\)](#)

fscommand2()

Disponibilità

Flash Lite 1.1.

Uso

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

Parametri

command Una stringa trasmessa all'applicazione host per qualsiasi uso, o un comando trasmesso a Flash Lite.

parametro1...parametroN Un elenco delimitato da virgole di stringhe passate come parametri al comando specificato da *command*.

Descrizione

Funzione; consente al file SWF di comunicare con Flash Lite o l'ambiente host (ad esempio, il sistema operativo di un telefono o di un altro dispositivo). Il valore restituito da `fscommand2()` dipende dal comando specifico.

La funzione `fscommand2()` è simile alla funzione `fscommand()`, con le seguenti differenze:

- La funzione `fscommand2()` può ricevere un numero arbitrario di argomenti.
- Flash Lite esegue `fscommand2()` immediatamente, mentre `fscommand()` viene eseguito alla fine del fotogramma in corso di elaborazione.
- La funzione `fscommand2()` restituisce un valore che può essere utilizzato per riportare la corretta esecuzione, il fallimento o il risultato del comando.

Le stringhe e le espressioni che vengono passate alla funzione come comandi e parametri sono descritte nelle tabelle di questa sezione.

Tali tabelle presentano le tre colonne illustrate di seguito:

- La colonna Comando mostra il parametro letterale di stringa che identifica il comando.
- La colonna Parametri indica quali tipi di valori è necessario passare per gli eventuali parametri aggiuntivi.
- La colonna Valore restituito descrive i valori restituiti previsti.

Esempio

Nel resto di questa sezione sono presentati gli esempi per ogni comando specifico eseguibile mediante la funzione `fscommand2()`.

Vedere anche

[fscommand\(\)](#)

Escape

Disponibilità

Flash Lite 1.1.

Descrizione

Codifica una stringa arbitraria in un formato sicuro per i trasferimenti in rete. Sostituisce ogni carattere non alfanumerico con una sequenza esadecimale di escape (`%XX o %XX%XX` nel caso dei caratteri multibyte).

Comando	Parametri	Valore restituito
"Escape"	<p><i>original</i> Stringa da codificare in un formato sicuro per gli URL.</p> <p><i>encoded</i> Stringa codificata risultante.</p> <p>Questi parametri sono costituiti da nomi di variabili o valori costanti di stringa (ad esempio, "Encoded_String").</p>	<p>0: esito negativo.</p> <p>1: esito positivo.</p>

Esempio

L'esempio seguente mostra la conversione di una stringa di esempio nella relativa forma codificata:

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

Vedere anche

[Unescape](#)

FullScreen

Disponibilità

Flash Lite 1.1.

Descrizione

Imposta le dimensioni dell'area di visualizzazione da utilizzare per il rendering. Le dimensioni possono essere equivalenti o inferiori allo schermo intero.

Questo comando è supportato solo quando Flash Lite è in esecuzione in modalità autonoma, e non quando è in esecuzione nel contesto di un'altra applicazione (ad esempio, come plug-in di un browser).

Comando	Parametri	Valore restituito
"FullScreen"	<i>size</i> Una variabile definita o un valore costante di stringa, con uno dei seguenti valori: <code>true</code> (schermo intero) o <code>false</code> (inferiore allo schermo intero). Qualunque altro valore viene considerato <code>false</code> .	-1: non supportato. 0: supportato.

Esempio

L'esempio seguente tenta di impostare l'area di visualizzazione a schermo intero. Se il valore restituito è diverso da 0, l'indicatore di riproduzione viene inviato al fotogramma con l'etichetta `smallScreenMode`:

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

GetBatteryLevel

Disponibilità

Flash Lite 1.1.

Descrizione

Restituisce il livello corrente della batteria. Si tratta di un valore numerico compreso tra 0 e il valore massimo restituito dalla variabile `GetMaxBatteryLevel`.

Comando	Parametri	Valore restituito
"GetBatteryLevel"	Nessuno.	-1: non supportato. Altri valori numerici: il livello corrente della batteria.

Esempio

L'esempio seguente imposta la variabile `battLevel` sul livello corrente della batteria:

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

Vedere anche

[GetMaxBatteryLevel](#)

GetDateDay

Disponibilità

Flash Lite 1.1.

Descrizione

Restituisce il giorno della data corrente. Si tratta di un valore numerico (senza 0 iniziale). I giorni validi sono compresi tra 1 e 31.

Comando	Parametri	Valore restituito
"GetDateDay"	Nessuno.	-1: non supportato. Da 1 a 31: il giorno del mese.

Esempio

L'esempio seguente raccoglie le informazioni relative alla data e crea una stringa completa relativa alla data:

```
today = fscommand2("GetDateDay");
weekday = fscommand2("GetDateWeekday");
thisMonth = fscommand2("GetDateMonth");
thisYear = fscommand2("GetDateYear");
when = weekday add ", " add ThisMonth add " " add today add ", " add
    thisYear;
```

Vedere anche

[GetDateMonth](#), [GetDateWeekday](#), [GetDateYear](#)

GetDateMonth

Disponibilità

Flash Lite 1.1.

Descrizione

Restituisce il mese della data corrente sotto forma di valore numerico (senza 0 iniziale).

Comando	Parametri	Valore restituito
"GetDateMonth"	Nessuno.	-1: Non supportato. Da 1 a 12: il numero del mese corrente.

Esempio

L'esempio seguente raccoglie le informazioni relative alla data e crea una stringa completa relativa alla data:

```
today = fscommand2("GetDateDay");
weekday = fscommand2("GetDateWeekday");
thisMonth = fscommand2("GetDateMonth");
thisYear = fscommand2("GetDateYear");
when = weekday add ", " add thisMonth add " " add today add ", " add
    thisYear;
```

Vedere anche

[GetDateDay](#), [GetDateWeekday](#), [GetDateYear](#)

GetDateWeekday

Disponibilità

Flash Lite 1.1.

Descrizione

Restituisce un valore numerico che corrisponde al nome del giorno della data corrente, sotto forma di valore numerico.

Comando	Parametri	Valore restituito
"GetDateWeekday"	Nessuno.	-1: non supportato. 0: domenica. 1: lunedì. 2: martedì. 3: mercoledì. 4: giovedì. 5: venerdì. 6: sabato.

Esempio

L'esempio seguente raccoglie le informazioni relative alla data e crea una stringa completa relativa alla data:

```
today = fscommand2("GetDateDay");  
weekday = fscommand2("GetDateWeekday");  
thisMonth = fscommand2("GetDateMonth");  
thisYear = fscommand2("GetDateYear");  
when = weekday add ", " add thisMonth add " " add today add ", " add  
    thisYear;
```

Vedere anche

[GetDateDay](#), [GetDateMonth](#), [GetDateYear](#)

GetDateYear

Restituisce un valore numerico a quattro cifre che corrisponde all'anno della data corrente.

Comando	Parametri	Valore restituito
"GetDateYear"	Nessuno.	-1: Non supportato. Da 0 a 9999: l'anno corrente.

Disponibilità

Flash Lite 1.1.

Esempio

L'esempio seguente raccoglie le informazioni relative alla data e crea una stringa completa relativa alla data:

```
today = fscommand2("GetDateDay");
weekday = fscommand2("GetDateWeekday");
thisMonth = fscommand2("GetDateMonth");
thisYear = fscommand2("GetDateYear");
when = weekday add ", " add thisMonth add " " add today add ", " add
    thisYear;
```

Vedere anche

[GetDateDay](#), [GetDateMonth](#), [GetDateWeekday](#)

GetDevice

Imposta un parametro che identifica il dispositivo su cui è in esecuzione Flash Lite. Questo identificatore corrisponde generalmente al nome del modello.

Comando	Parametri	Valore restituito
"GetDevice"	<i>device</i> Stringa che riceve l'identificatore del dispositivo. Può trattarsi del nome di una variabile o di un valore di stringa che contiene il nome di una variabile.	-1: non supportato. 0: supportato.

Disponibilità

Flash Lite 1.1.

Esempio

Il codice seguente assegna l'identificatore del dispositivo alla variabile `statusdevice`, quindi aggiorna un campo di testo con il nome generico del dispositivo:

Segono alcuni risultati di esempio e i dispositivi a cui corrispondono:

```
D506i Un telefono Mitsubishi 506i.
DFOMA1 Un telefono Mitsubishi FOMA1.
F506i Un telefono Fujitsu 506i.
FFOMA1 Un telefono Fujitsu FOMA1.
N506i Un telefono NEC 506i.
```

NFOMA1 Un telefono NEC FOMA1.

Nokia3650 Un telefono Nokia 3650.

p506i Un telefono Panasonic 506i.

PFOMA1 Un telefono Panasonic FOMA1.

SH506i Un telefono Sharp 506i.

SHFOMA1 Un telefono Sharp FOMA1.

S0506i Un telefono Sony 506i.

```
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 "S0506i":
    /:myText += "device: Sony 506i" add newline;
    break;
}
```

GetDeviceID

Imposta un parametro che rappresenta l'identificatore univoco del dispositivo (ad esempio, il numero di serie).

Comando	Parametri	Valore restituito
"GetDeviceID"	<i>id</i> Una stringa che riceve l'identificatore del dispositivo. Può trattarsi del nome di una variabile o di un valore di stringa che contiene il nome di una variabile.	-1: non supportato. 0: supportato.

Disponibilità

Flash Lite 1.1.

Esempio

L'esempio seguente assegna l'identificatore univoco alla variabile `deviceID`:

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

GetFreePlayerMemory

Restituisce la quantità di memoria heap, espressa in KB, attualmente disponibile per Flash Lite.

Comando	Parametri	Valore restituito
"GetFreePlayerMemory"	Nessuno.	-1: non supportato. 0 o valore positivo: kilobyte di memoria heap disponibili.

Disponibilità

Flash Lite 1.1.

Esempio

L'esempio seguente imposta lo stato in modo che sia uguale alla quantità di memoria libera:

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

Vedere anche

[GetTotalPlayerMemory](#)

GetLanguage

Disponibilità

Flash Lite 1.1.

Imposta un parametro che identifica la lingua attualmente utilizzata dal dispositivo. La lingua viene restituita sotto forma di una stringa di una variabile che viene passata in base al nome.

Comando	Parametri	Valore restituito
"GetLanguage"	<p><i>language</i> Stringa che riceve il codice di lingua. Può trattarsi del nome di una variabile o di un valore di stringa che contiene il nome di una variabile. Il valore restituito può essere uno dei seguenti:</p> <ul style="list-style-type: none">cs: ceco (Czech).da: danese (Danish).de: tedesco (German).en-UK: inglese britannico o internazionale (UK or international English).en-US: inglese americano (US English).es: spagnolo (Spanish).fi: finlandese (Finnish).fr: francese (French).hu: ungherese (Hungarian).it: italiano (Italian).ja: giapponese (Japanese).ko: coreano (Korean).nl: olandese (Dutch).no: norvegese (Norwegian).pl: polacco (Polish).pt: portoghese (Portuguese).ru: russo (Russian).sv: svedese (Swedish).tr: turco (Turkish).xu: lingua non determinata.zh-CN: cinese semplificato (Simplified Chinese).zh-TW: cinese tradizionale (Traditional Chinese).	<p>-1: non supportato. 0: supportato.</p>

NOTA

Quando i telefoni giapponesi sono impostati per visualizzare l'inglese, per *language* viene restituito `en_US`.

Esempio

L'esempio seguente assegna il codice di lingua alla variabile `language`, quindi aggiorna un campo di testo con la lingua riconosciuta da Flash Lite Player:

```
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

Disponibilità

Flash Lite 1.1.

Descrizione

Imposta un parametro su una stringa che rappresenta la data corrente in formato esteso e formattata in base alla versione localizzata attualmente definita.

Comando	Parametri	Valore restituito
"GetLocaleLongDate"	<p><i>longdate</i> Variabile di stringa che riceve la forma estesa dalla data corrente (ad esempio, "October 16, 2004" o "16 October 2004").</p> <p>Può trattarsi del nome di una variabile o di un valore di stringa che contiene il nome di una variabile.</p> <p>Il valore restituito in <i>longdate</i> è una stringa a più caratteri e a lunghezza variabile. La formattazione vera e propria dipende dal dispositivo e dalle impostazioni internazionali.</p>	<p>-1: non supportato. 0: supportato.</p>

Esempio

L'esempio seguente tenta di restituire la forma estesa della data corrente nella variabile `longDate`. Inoltre, imposta il valore di `status` per riportare se è stato possibile completare l'operazione.

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

Vedere anche

[GetLocaleShortDate](#), [GetLocaleTime](#)

GetLocaleShortDate

Disponibilità

Flash Lite 1.1.

Descrizione

Imposta un parametro su una stringa che rappresenta la data corrente in formato abbreviato e formattata in base alla versione localizzata attualmente definita.

Comando	Parametri	Valore restituito
"GetLocaleShortDate"	<i>shortdate</i> Variabile di stringa che riceve la forma abbreviata dalla data corrente (ad esempio, "10/16/2004" o "16-10-2004"). Può trattarsi del nome di una variabile o di un valore di stringa che contiene il nome di una variabile. Il valore restituito in <i>shortdate</i> è una stringa a più caratteri e a lunghezza variabile. La formattazione vera e propria dipende dal dispositivo e dalla versione localizzata.	-1: non supportato. 0: supportato.

Esempio

L'esempio seguente tenta di ottenere la forma abbreviata della data corrente nella variabile `shortDate`. Inoltre, imposta il valore di `status` per riportare se è stato in grado di completare l'operazione.

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

Vedere anche

[GetLocaleLongDate](#), [GetLocaleTime](#)

GetLocaleTime

Disponibilità

Flash Lite 1.1.

Descrizione

Imposta un parametro su una stringa che rappresenta l'orario corrente, formattato in base alla versione localizzata attualmente definita.

Comando	Parametri	Valore restituito
"GetLocaleTime"	<i>time</i> Variabile di stringa che riceve il valore dell'orario corrente (ad esempio, "6:10:44 PM" o "18:10:44 "). Può trattarsi del nome di una variabile o di un valore di stringa che contiene il nome di una variabile. Il valore restituito in <i>time</i> è una stringa a più caratteri e a lunghezza variabile. La formattazione vera e propria dipende dal dispositivo e dalla versione localizzata.	-1: non supportato. 0: supportato.

Esempio

L'esempio seguente tenta di ottenere l'ora locale corrente nella variabile *time*. Inoltre, imposta il valore di *status* per riportare se è stato possibile completare l'operazione.

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

Vedere anche

[GetLocaleLongDate](#), [GetLocaleShortDate](#)

GetMaxBatteryLevel

Disponibilità

Flash Lite 1.1.

Descrizione

Restituisce il livello massimo della batteria del dispositivo. Si tratta di un valore numerico maggiore di 0.

Comando	Parametri	Valore restituito
"GetMaxBatteryLevel"	Nessuno.	-1: non supportato. Altri valori: il livello massimo della batteria.

Esempio

L'esempio seguente imposta la variabile `maxBatt` sul livello massimo della batteria:

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

GetMaxSignalLevel

Disponibilità

Flash Lite 1.1.

Descrizione

Restituisce il livello di potenza massimo del segnale. Si tratta di un valore numerico maggiore di 0.

Comando	Parametri	Valore restituito
"GetMaxSignalLevel"	Nessuno.	-1: non supportato. Altri valori numerici: il livello massimo del segnale.

Esempio

L'esempio seguente assegna la potenza massima del segnale alla variabile `sigStrengthMax`:

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

GetMaxVolumeLevel

Disponibilità

Flash Lite 1.1.

Descrizione

Restituisce il livello massimo del volume del dispositivo, sotto forma di valore numerico.

Comando	Parametri	Valore restituito
"GetMaxVolumeLevel"	Nessuno.	-1: non supportato. Altri valori: il livello massimo del volume.

Esempio

L'esempio seguente imposta la variabile `maxvolume` sul livello di volume massimo del dispositivo:

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

Vedere anche

[GetVolumeLevel](#)

GetNetworkConnectStatus

Disponibilità

Flash Lite 1.1.

Descrizione

Restituisce un valore che indica lo stato di connessione corrente alla rete.

Comando	Parametri	Valore restituito
"GetNetworkConnectStatus"	Nessuno.	-1: non supportato. 0: è attualmente presente una connessione di rete attiva. 1: il dispositivo sta tentando di connettersi alla rete. 2: non è attualmente presente alcuna connessione di rete attiva. 3: la connessione di rete è sospesa. 4: non è possibile determinare la connessione di rete.

Esempio

L'esempio seguente assegna lo stato della connessione di rete alla variabile `connectstatus`, quindi utilizza un'istruzione `switch` per aggiornare un campo di testo con lo stato della connessione:

```
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

Disponibilità

Flash Lite 1.1.

Descrizione

Imposta un parametro sul nome della rete corrente.

Comando	Parametri	Valore restituito
"GetNetworkName"	<i>networkName</i> Una stringa che rappresenta il nome di rete. Può trattarsi del nome di una variabile o di un valore di stringa che contiene il nome di una variabile. Se la rete è registrata ed è possibile determinarne il nome, <i>networkname</i> viene impostato sul nome della rete; in caso contrario, viene impostato sulla stringa vuota.	-1: non supportato. 0: non è registrata alcuna rete. 1: la rete è registrata, ma il nome di rete non è conosciuto. 2: la rete è registrata e il nome di rete è conosciuto.

Esempio

L'esempio seguente assegna il nome della rete corrente alla variabile `myNetName` e un valore di status alla variabile `netNameStatus`:

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

GetNetworkRequestStatus

Disponibilità

Flash Lite 1.1.

Descrizione

Restituisce un valore che indica lo stato della richiesta HTTP più recente.

Comando	Parametri	Valore restituito
"GetNetworkRequestStatus"	Nessuno.	-1: il comando non è supportato. 0: è presente una richiesta in sospeso ("pending"), è stata stabilita una connessione di rete, è stato risolto il nome host del server ed è stata effettuata una connessione al server. 1: è presente una richiesta in sospeso ("pending") e si sta stabilendo una connessione di rete. 2: è presente una richiesta in sospeso ("pending") ma non è ancora stata stabilita una connessione di rete. 3: è presente una richiesta in sospeso ("pending"), è stata stabilita una connessione di rete e si sta risolvendo il nome host del server. 4: la richiesta non è riuscita a causa di un errore di rete. 5: la richiesta non è riuscita a causa di un errore nella connessione al server. 6: il server ha restituito un errore HTTP (ad esempio, 404). 7: la richiesta non è riuscita a causa di un errore nell'accesso al server DNS o nella risoluzione del nome del server. 8: la richiesta è stata completata correttamente. 9: la richiesta non è riuscita a causa di un timeout. 10: la richiesta non è ancora stata effettuata.

Esempio

L'esempio seguente assegna lo stato della richiesta HTTP più recente alla variabile `requeststatus`, quindi utilizza un'istruzione `switch` per aggiornare un campo di testo con lo stato:

```
requeststatus = fscCommand2("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

Disponibilità

Flash Lite 1.1.

Descrizione

Restituisce un valore che indica lo stato di rete per il telefono (ovvero se è registrata una rete e se il telefono è attualmente in roaming).

Comando	Parametri	Valore restituito
"GetNetworkStatus"	Nessuno.	-1: il comando non è supportato. 0: non è registrata alcuna rete. 1: sulla rete domestica. 2: sulla rete domestica estesa. 3: roaming in corso (lontano dalla rete domestica).

Esempio

L'esempio seguente assegna lo stato della connessione di rete alla variabile `networkstatus`, quindi utilizza un'istruzione `switch` per aggiornare un campo di testo con lo stato:

```
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

Disponibilità

Flash Lite 1.1.

Descrizione

Imposta un parametro che identifica la piattaforma corrente, che descrive in generale la classe del dispositivo. Per i dispositivi con sistemi operativi aperti, l'identificatore di solito indica il nome e la versione del sistema operativo.

Comando	Parametri	Valore restituito
"GetPlatform"	<i>platform</i> Stringa che riceve l'identificatore della piattaforma. Può trattarsi del nome di una variabile o di un valore di stringa che contiene il nome di una variabile.	-1: non supportato. 0: supportato.

Esempio

Il codice seguente assegna l'identificatore della piattaforma alla variabile `statusplatform`, quindi aggiorna un campo di testo con il nome generico della piattaforma:

Gli esempi seguenti illustrano alcuni risultati possibili di `myPlatform` e le classi di dispositivo che rappresentano:

506i Un telefono 506i.

FOMA1 Un telefono FOMA1.

Symbian6.1_s60.1 Un telefono Symbian 6.1, serie 60 versione 1.

Symbian7.0 Un telefono Symbian 7.0.

```
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

Disponibilità

Flash Lite 1.1.

Descrizione

Restituisce un valore che indica se l'alimentazione è attualmente fornita da una batteria o da una fonte di alimentazione esterna.

Comando	Parametri	Valore restituito
"GetPowerSource"	Nessuno.	-1: non supportato. 0: il dispositivo funziona a batteria. 1: il dispositivo funziona ad alimentazione esterna.

Esempio

L'esempio seguente imposta la variabile `myPower` per indicare la fonte di alimentazione oppure su -1 se non è stato possibile completare l'operazione:

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

GetSignalLevel

Disponibilità

Flash Lite 1.1.

Descrizione

Restituisce la potenza corrente del segnale, sotto forma di valore numerico.

Comando	Parametri	Valore restituito
"GetSignalLevel"	Nessuno.	-1: non supportato. Altri valori numerici: il valore corrente del segnale, compreso tra 0 e il valore massimo restituito da <code>GetMaxSignalLevel</code> .

Esempio

L'esempio seguente assegna il valore massimo del segnale alla variabile `sigLevel`:

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

GetTimeHours

Disponibilità

Flash Lite 1.1.

Descrizione

Restituisce l'ora dell'orario corrente del giorno, nel formato a 24 ore. Si tratta di un valore numerico (senza 0 iniziale).

Comando	Parametri	Valore restituito
"GetTimeHours"	Nessuno.	-1: non supportato. Da 0 a 23: l'ora corrente.

Esempio

L'esempio seguente imposta la variabile `hour` sulla porzione dell'ora dell'orario corrente del giorno, oppure su -1:

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

Vedere anche

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

GetTimeMinutes

Disponibilità

Flash Lite 1.1.

Descrizione

Restituisce il minuto dell'orario corrente del giorno. Si tratta di un valore numerico (senza 0 iniziale).

Comando	Parametri	Valore restituito
"GetTimeMinutes"	Nessuno.	-1: non supportato. Da 0 a 59: il minuto corrente.

Esempio

L'esempio seguente imposta la variabile `minutes` sulla porzione dei minuti dell'orario corrente del giorno, oppure su `-1`:

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

Vedere anche

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

GetTimeSeconds

Disponibilità

Flash Lite 1.1.

Descrizione

Restituisce il secondo dell'orario corrente del giorno. Si tratta di un valore numerico (senza 0 iniziale).

Comando	Parametri	Valore restituito
"GetTimeSeconds"	Nessuno.	-1: non supportato. Da 0 a 59: il secondo corrente.

Esempio

L'esempio seguente imposta la variabile `seconds` sulla porzione dei secondi dell'orario corrente del giorno, oppure su `-1`:

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

Vedere anche

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

GetTimeZoneOffset

Disponibilità

Flash Lite 1.1.

Descrizione

Imposta il parametro su un numero di minuti compreso tra il fuso orario locale e l'ora universale (UTC).

Comando	Parametri	Valore restituito
"GetTimeZoneOffset"	<i>timezoneOffset</i> Numero di minuti compresi tra il fuso orario locale e l'ora universale (UTC). Può trattarsi del nome di una variabile o di un valore di stringa che contiene il nome di una variabile. Viene restituito un valore numerico positivo o negativo, ad esempio: 540: ora standard del Giappone -420: ora legale del Pacifico	-1: non supportato. 0: supportato.

Esempio

L'esempio seguente assegna alla variabile `timezoneoffset` i minuti di sfalsamento rispetto all'ora universale e imposta `status` su 0 oppure imposta `status` su -1:

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

Vedere anche

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

GetTotalPlayerMemory

Disponibilità

Flash Lite 1.1.

Descrizione

Restituisce la quantità totale di memoria heap, espressa in kilobyte, disponibile per Flash Lite.

Comando	Parametri	Valore restituito
"GetTotalPlayerMemory"	Nessuno.	-1: non supportato. 0 o un valore positivo: kilobyte totali di memoria heap.

Esempio

L'esempio seguente imposta la variabile `status` sulla quantità totale di memoria heap:

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

Vedere anche

[GetFreePlayerMemory](#)

GetVolumeLevel

Disponibilità

Flash Lite 1.1.

Descrizione

Restituisce il livello corrente del volume del dispositivo, sotto forma di valore numerico.

Comando	Parametri	Valore restituito
"GetVolumeLevel"	Nessuno.	-1: non supportato. Altri valori numerici: il livello corrente del volume, compreso tra 0 e il valore restituito da <code>fscommand2("GetMaxVolumeLevel")</code> .

Esempio

L'esempio seguente assegna il valore corrente del volume alla variabile `volume`:

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

Vedere anche

[GetVolumeLevel](#)

Quit

Disponibilità

Flash Lite 1.1.

Descrizione

Indica a Flash Lite Player di interrompere la riproduzione e uscire.

Questo comando è supportato solo quando Flash Lite è in esecuzione in modalità autonoma, e non quando è in esecuzione nel contesto di un'altra applicazione (ad esempio, come plug-in di un browser).

Comando	Parametri	Valore restituito
"Quit"	Nessuno.	-1: non supportato.

Esempio

L'esempio seguente indica a Flash Lite di interrompere la riproduzione e uscire, se è in esecuzione in modalità autonoma:

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

ResetSoftKeys

Disponibilità

Flash Lite 1.1.

Descrizione

Ripristina le impostazioni originali dei tasti virtuali.

Questo comando è supportato solo quando Flash Lite è in esecuzione in modalità autonoma, e non quando è in esecuzione nel contesto di un'altra applicazione (ad esempio, come plug-in di un browser).

Comando	Parametri	Valore restituito
"ResetSoftKeys"	Nessuno.	-1: non supportato. 0: supportato.

Esempio

L'istruzione seguente ripristina le impostazioni originali dei tasti virtuali:

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

Vedere anche

[SetSoftKeys](#)

SetInputTextType

Disponibilità

Flash Lite 1.1.

Descrizione

Specifica la modalità di apertura dei campi di testo di input:

Comando	Parametri	Valore restituito
"SetInputTextType"	<i>variableName</i> Il nome del campo di testo di input. Può trattarsi del nome di una variabile o di un valore di stringa che contiene il nome di una variabile. <i>type</i> Uno dei seguenti valori: Numeric, Alpha, Alphanumeric, Latin, NonLatin o NoRestriction.	0: esito negativo. 1: esito positivo.

Flash Lite supporta la funzione di testo di input chiedendo all'applicazione host di avviare l'interfaccia generica di immissione del testo specifica per il dispositivo, spesso definita *elaboratore front-end o FEP* (Front-end Processor). Quando il comando `SetInputTextType` non viene utilizzato, il FEP viene aperto in modalità predefinita.

La tabella seguente mostra l'effetto di ogni modalità e le modalità che vengono sostituite:

Modalità specificata	Imposta il FEP su una di queste modalità a esclusione reciproca	Se non è supportata sul dispositivo corrente, apre il FEP in questa modalità
Numeric	Solo numeri (da 0 a 9)	Alphanumeric
Alpha	Solo caratteri alfanumerici (da A a Z, da a a z)	Alphanumeric
Alphanumeric	Solo caratteri alfanumerici (da 0 a 9, da A a Z, da a a z)	Latin
Latin	Solo caratteri latini (caratteri alfanumerici e punteggiatura)	NoRestriction

Modalità specificata	Imposta il FEP su una di queste modalità a esclusione reciproca	Se non è supportata sul dispositivo corrente, apre il FEP in questa modalità
NonLatin	Solo caratteri non latini (ad esempio, Kanji e Kana)	NoRestriction
NoRestriction	Modalità predefinita (non imposta limitazioni per il FEP)	

NOTA

Non tutti i telefoni cellulari supportano questi tipi di campi di testo di input. Per tale motivo, è necessario convalidare i dati di testo di input.

Esempio

La riga di codice seguente imposta il tipo di testo di input del campo associato alla variabile `input1` in modo che riceva i dati numerici:

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

SetQuality

Disponibilità

Flash Lite 1.1.

Descrizione

Imposta la qualità di rendering dell'animazione.

Comando	Parametri	Valore restituito
"SetQuality"	<i>quality</i> La qualità di rendering; deve essere "high", "medium" o "low".	-1: non supportato. 0: supportato.

Esempio

L'esempio seguente imposta la qualità di rendering su low (bassa):

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

SetSoftKeys

Disponibilità

Flash Lite 1.1.

Descrizione

Mappa nuovamente i tasti virtuali sinistro e destro del dispositivo, a condizione che siano accessibili e modificabili.

Una volta eseguito questo comando, se si preme il tasto sinistro viene generato un evento `PageUp`, mentre se si preme il tasto destro viene generato un evento `PageDown`. Il codice `ActionScript` associato agli eventi di pressione tasto `PageUp` e `PageDown` viene eseguito quando viene premuto il rispettivo tasto.

Questo comando è supportato solo quando Flash Lite è in esecuzione in modalità autonoma, e non quando è in esecuzione nel contesto di un'altra applicazione (ad esempio, come plug-in di un browser).

Comando	Parametri	Valore restituito
"SetSoftKeys"	<i>left</i> Testo da visualizzare per il tasto virtuale sinistro. <i>right</i> Testo da visualizzare per il tasto virtuale destro. Questi parametri sono costituiti da nomi di variabili o valori costanti di stringa (ad esempio, "Previous").	-1: non supportato. 0: supportato.

Esempio

L'esempio seguente specifica che il tasto virtuale sinistro sia contrassegnato come `Previous` e che il tasto virtuale destro come `Next`.

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

Vedere anche

[ResetSoftKeys](#)

StartVibrate

Disponibilità

Flash Lite 1.1.

Descrizione

Avvia la vibrazione del telefono. Se la vibrazione è già in corso, Flash Lite la interrompe prima di avviarne una nuova. Le vibrazioni si fermano anche quando la riproduzione dell'applicazione Flash viene interrotta o messa in pausa, o quando si esce da Flash Lite Player.

Comando	Parametri	Valore restituito
"StartVibrate"	<i>time_on</i> Quantità di tempo, espressa in millisecondi (fino a un massimo di 5 secondi), per cui la vibrazione è attiva. <i>time_off</i> Quantità di tempo, espressa in millisecondi (fino a un massimo di 5 secondi), per cui la vibrazione è disattivata. <i>repeat</i> Numero di volte (massimo 3) per cui la vibrazione viene ripetuta.	-1: non supportato. 0: la vibrazione è stata avviata. 1: si è verificato un errore e non è stato possibile avviare la vibrazione.

Esempio

L'esempio seguente tenta di avviare una sequenza di vibrazioni composta da 2,5 secondi di attività e 1 secondo di inattività e ripetuta due volte. Assegna un valore alla variabile `status`, che indica l'esito positivo o negativo.

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

Vedere anche

[StopVibrate](#)

StopVibrate

Disponibilità

Flash Lite 1.1.

Descrizione

Interrompe la vibrazione del telefono, se è attualmente in funzione.

Comando	Parametri	Valore restituito
"StopVibrate"	Nessuno.	-1: non supportato. 0: la vibrazione si è interrotta.

Esempio

L'esempio seguente chiama `StopVibrate` e salva il risultato (non supportato o vibrazione interrotta) nella variabile `status`:

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

Vedere anche

[StartVibrate](#)

Unescape

Disponibilità

Flash Lite 1.1.

Descrizione

Decodifica una stringa arbitraria che è stata codificata per essere sicura per i trasferimenti in rete, e la riporta al normale formato non codificato. Tutti i caratteri che sono in formato esadecimale, ovvero un simbolo di percentuale (%) seguito da due cifre esadecimali, vengono convertiti nella rispettiva forma decodificata.

Comando	Parametri	Valore restituito
"Unescape"	<i>original</i> Stringa da decodificare da un formato sicuro per gli URL a un formato normale. <i>decoded</i> Stringa decodificata risultante. (Questo parametro può essere il nome di una variabile o di un valore di stringa che contiene il nome di una variabile.)	0: esito negativo. 1: esito positivo.

Esempio

L'esempio seguente mostra la decodifica di una stringa codificata:

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

Vedere anche

[Escape](#)

Indice analitico

Simboli

! (NOT logico), operatore 98
" " (delimitatore di stringa), operatore 109
\$version, variabile 129
% (modulo), operatore 100
%= (assegnazione modulo), operatore 100
&& (AND logico), operatore 97
|| (OR logico), operatore 99
* (moltiplicazione), operatore 102
*= (assegnazione moltiplicazione), operatore 101
+ (addizione numerica), operatore 103
++ (incremento), operatore 96
+= (assegnazione addizione), operatore 87
, (virgola), operatore 90
-= (assegnazione sottrazione), operatore 116
. (punto), operatore 95
/ (barra - linea temporale principale), proprietà 54
/ (divisione), operatore 94
/* (commento a blocchi), operatore 89
// (commento), operatore 91
/= (divisione), operatore 94
< (minore di numerico), operatore 107

_capSMAF, variabile 126
_capSMS, variabile 126
_capStreamSound, variabile 127
_currentframe, proprietà 56
_focusrect, proprietà 56
_framesloaded, proprietà 57
_height, proprietà 58
_highquality, proprietà 58
_level, proprietà 59
_name, proprietà 61
_rotation, proprietà 61
_scroll, proprietà 62
_target, proprietà 62
_visible, proprietà 63
_width, proprietà 64
_x, proprietà 64
_xscale, proprietà 65
_y, proprietà 66
_yscale, proprietà 67
- (sottrazione), operatore 115
— (decremento), operatore 93

A

add (concatenazione stringhe), operatore 86
addizione numerica 103
_alpha, variabile 55
AND logico, operatore 97
AND, operatore 97
and, operatore 88
assegnazione divisione, operatore 94
assegnazione modulo 100
assegnazione sottrazione, operatore 116
assegnazione, operatore 89

B

break, istruzione 70

C

call 15

_cap4WayKeyAS, variabile 128

_capCompoundSound, variabile 121

_capEmail, variabile 122

_capLoadData, variabile 122

_capMFi, variabile 124

_capMMS, variabile 124

_capSMAF, variabile 126

_capSMS, variabile 126

_capStreamSound, variabile 127

case, istruzione 71

chr(), funzione 16

commenti

 block 89

 riga singola 91

commento a blocchi, operatore 89

concatenazione 86

condizioni 78

continue, istruzione 72

_currentframe, proprietà 56

D

delimitatore di stringa, operatore 109

disuguaglianza, operatore 106

divisione 94

do..while, istruzione 74

duplicateMovieClip(), funzione 17

E

e-mail, variabile di funzionalità 122

else if, istruzione 76

else, istruzione 75

eq (uguaglianza stringhe), operatore 110

eval(), funzione 18

F

_focusrect, proprietà 56

for, ciclo 77

for, istruzione 77

_framesloaded, proprietà 57

fsccommand(), comando 129

funzioni

 chr() 16

 duplicateMovieClip() 17

 eval() 18

 fsccommand() 129

 getProperty() 19

 getTimer() 20

 getURL() 20

 gotoAndPlay() 23

 gotoAndStop() 24

 iffFrameLoaded() 25

 int() 26

 length() 26

 loadMovie() 27

 loadMovieNum() 28

 loadVariables() 30

 loadVariablesNum() 31

 mbchr() 32

 mbsubstring() 35

 nextFrame() 35

 nextScene() 36

 Number() 37

 on() 38

 ord() 39

 play() 39

 prevFrame() 40

 prevScene() 41

 random() 41

 removeMovieClip() 42

 set() 43

 setProperty() 44

 stop() 45

 stopAllSounds() 45

 String() 46

 substring() 47

 tellTarget() 47

 toggleHighQuality() 48

 trace() 49

 unloadMovie() 50

 unloadMovieNum() 51

G

ge (stringa maggiore di o uguale a), operatore 111

getProperty(), funzione 19

getTimer(), funzione 20

getURL(), funzione 20

gotoAndPlay(), funzione 23

gotoAndStop(), funzione 24
gt (stringa maggiore di), operatore 110

H

_height, proprietà 58
_highquality, proprietà 58

I

identificatore della linea temporale principale 54
if, istruzione 78
iffFrameLoaded(), funzione 25
incremento, operatore 96
int(), funzione 26
istruzioni
 break 70
 case 71
 continue 72
 do..while 74
 else 75
 else if 76
 for 77
 if 78
 NOT logico 98
 switch 78
 while 80

L

le (stringa minore di o uguale a), operatore 114
length(), funzione 26
_level, proprietà 59
loadMovie(), funzione 27
loadMovieNum(), funzione 28
loadVariables(), funzione 30
loadVariablesNum(), funzione 31
lt (stringa minore di), operatore 113

M

maggiore di o uguale a, operatore 105
maggiore di, operatore 105
maxscroll, proprietà 60
mbchr(), funzione 32
mbsubstring(), funzione 35
messaggistica, variabili 124, 126
MFI, audio 123
MIDI, audio 124

minore di o uguale a, operatore 107
minore di, operatore 107
MMS, messaggi 124
modulo, operatore 100
moltiplicazione 102

N

_name, proprietà 61
ne (diseguaglianza stringhe), operatore 112
nextFrame(), funzione 35
nextScene(), funzione 36
NOT logico, operatore 98
NOT, operatore 98
Number(), funzione 37

O

on(), funzione 38
operatore condizionale 92
operatore di assegnazione addizione 87
operatori
 addizione numerica 103
 and 88
 AND logico 97
 assegnazione 89
 assegnazione addizione 87
 assegnazione divisione 94
 assegnazione modulo 100
 assegnazione sottrazione 116
 comment 91
 commento a blocchi 89
 condizionali 92
 delimitatore di stringa 109
 diseguaglianza numerica 106
 diseguaglianza stringhe 112
 divisione 94
 incremento 96
 maggiore di 105
 maggiore di o uguale a 105
 minore di numerico 107
 minore di o uguale a numerico 107
 modulo 100
 moltiplicazione 102
 NOT logico 98
 OR logico 99
 punto 95
 stringa maggiore di 110
 stringa maggiore di o uguale a 111

- stringa minore di 113
- stringa minore di o uguale a 114
- stringhe, concatenazione 86
- uguaglianza numerica 104
- uguaglianza stringhe 110
- virgola 90

OR logico, operatore 99

OR, operatore 99

ord(), funzione 39

P

play(), funzione 39

prevFrame(), funzione 40

prevScene(), funzione 41

proprietà

- _alpha 55
- _currentframe 56
- _focusrect 56
- _framesloaded 57
- _height 58
- _highquality 58
- _level 59
- _name 61
- _rotation 61
- _scroll 62
- _target 62
- _visible 63
- _width 64
- _x 64
- _xscale 65
- _y 66
- _yscale 67
- barra 54
- maxscroll 60
- scroll 62

punto, operatore 95

R

random(), funzione 41

removeMovieClip(), funzione 42

_rotation, proprietà 61

S

scroll, proprietà 62

set(), funzione 43

setProperty(), funzione 44

stop(), funzione 45

stopAllSounds(), funzioni 45

String(), funzione 46

stringa maggiore di o uguale a 111

stringa maggiore di, operatore 110

stringa minore di o uguale a 114

substring(), funzione 47

suono, variabili 121, 123, 124, 126, 127

switch, istruzione 78

T

_target, proprietà 62

tellTarget(), funzione 47

toggleHighQuality(), funzione 48

_totalframes, proprietà 63

trace(), funzione 49

U

uguaglianza stringhe, operatore 110

unloadMovie(), funzione 50

unloadMovieNum(), funzione 51

V

variabili

- \$version 129
- _alpha 55
- _cap4WayKeyAS 128
- _capCompoundSound 121
- _capEmail 122
- _capLoadData 122
- _capMFi 123
- _capMIDI 124
- _capMMS 124
- _capSMAF 126
- _capSMS 126
- _capStreamSound 127
- e-mail, funzionalità 122
- funzione di caricamento dati 122
- navigazione mediante i tasti freccia 128
- numero di versione di Flash Lite 129

variabili, messaggistica

- _capMMS 124
- _capSMS 126

variabili, suono
 _capCompoundSound 121
 _capMFi 123
 _capMIDI 124
 _capSMAF 126
 _capStreamSound 127
virgola, operatore 90
_visible, proprietà 63

W

while, ciclo 74
while, istruzione 80
_width, proprietà 64

X

_x, proprietà 64
_xscale, proprietà 65

Y

_y, proprietà 66
_yscale, proprietà 67

