

macromedia[®] **FLASH[™] 5**

Guida di riferimento di ActionScript



Marchi

Macromedia, il logo Macromedia, il logo Made With Macromedia, Authorware, Backstage, Director, Extreme 3D e Fontographer sono marchi registrati di Macromedia, Inc. Afterburner, AppletAce, Authorware Interactive Studio, Backstage, Backstage Designer, Backstage Desktop Studio, Backstage Enterprise Studio, Backstage Internet Studio, DECK II, Director Multimedia Studio, Doc Around the Clock, Extreme 3D, Flash, FreeHand, FreeHand Graphics Studio, Lingo, Macromedia xRes, MAGIC, Power Applets, Priority Access, SoundEdit, Shockwave, Showcase, Tools to Power Your Ideas e Xtra sono marchi di Macromedia, Inc. Altri nomi di prodotti, logo, progetti, titoli, termini o frasi che appaiono nella presente pubblicazione possono essere marchi registrati, marchi di servizio o denominazioni commerciali di Macromedia, Inc. o di altre società e potrebbero essere registrati in alcune giurisdizioni.

Dichiarazione di non responsabilità Apple

APPLE COMPUTER, INC. NON FORNISCE ALCUNA GARANZIA, ESPLICITA O IMPLICITA, PER IL PACCHETTO SOFTWARE ACCLUSO, LA SUA COMMERCIALIZZABILITÀ O IDONEITÀ PER SCOPI SPECIFICI. POICHÉ L'ESCLUSIONE DELLE GARANZIE IMPLICITE NON È CONSENTITA DA ALCUNI PAESI, POTREBBE NON ESSERE APPLICABILE. LA PRESENTE GARANZIA CONFERISCE ALL'UTENTE PARTICOLARI DIRITTI. ALTRI DIRITTI POSSONO ESSERE RITENUTI VALIDI, A SECONDA DELLO PAESE.

Copyright © 2000 Macromedia, Inc. Tutti i diritti riservati. Nessuna parte del presente manuale può essere copiata, fotocopiata, riprodotta, tradotta o convertita in qualsiasi formato elettronico o meccanico senza la previa autorizzazione scritta di Macromedia, Inc. Numero parte ZFL50M200IT

Riconoscimenti

Manager progetto: Erick Vera

Autori: Jody Bleye, Mary Burger, Louis Dobrozensky, Stephanie Gowin, Marcelle Taylor e Judy Walthers Von Alten

Revisori: Peter Fenczik, Rosana Francescato, Ann Szabla

Multimedia: George Brown, John "Zippy" Lehnus e Noah Zilberberg

Design documentazione stampata e Guida in linea: Chris Basmajian e Noah Zilberberg

Produzione: Chris Basmajian e Rebecca Godbois

Manager progetto per la localizzazione: Yuko Yagi

Produzione localizzazione: Masayo "Noppe" Noda e Bowne Global Solutions

Speciali ringraziamenti a: Max D'Ambrosio, Jeremy Clark, Brian Dister e l'intero team Flash Development, Michael Dominguez, Margaret Dumas, William Harding, Sherri Harte, Yoshika Hedberg, Tim Hussey, Kipling Inscore, Alyn Kelley, Pete Santangeli, Denise Seymour e l'intero team Flash QA, Cyn Taylor e Eric Wittman

Prima edizione: Settembre 2000

Macromedia, Inc.
600 Townsend St.
San Francisco, CA 94103

SOMMARIO

INTRODUZIONE

Guida introduttiva	17
Novità di ActionScript di Flash 5	17
Differenze tra ActionScript e JavaScript	18
Modifica del testo	19
Sintassi del punto	19
Tipi di dati	19
Variabili locali	19
Funzioni definite dall'utente	19
Oggetti predefiniti	20
Azioni clip	20
Nuove azioni	20
Clip intelligenti	20
Debugger	20
Supporto XML	21
Ottimizzazione	21
Uso della Guida di Flash per le azioni	21

CAPITOLO 1

Introduzione ad ActionScript	23
Informazioni sulla creazione di script in ActionScript	24
Informazioni sulla progettazione e sul debug di script	25
Informazioni sullo scripting orientato agli oggetti	25
Informazioni sull'oggetto MovieClip	27
Flusso logico degli script	28
Controllo dell'esecuzione di ActionScript	30
Terminologia di ActionScript	32
Analisi di uno script di esempio	34
Uso del pannello Azioni	37
Modalità normale	37

Modalità esperto	39
Passaggio da una modalità di modifica all'altra	40
Uso di un editor esterno	41
Selezione delle opzioni del pannello Azioni	41
Evidenziazione e controllo della sintassi	43
Informazioni sull'evidenziazione degli errori	44
Assegnazione di azioni a oggetti	45
Assegnazione di azioni a fotogrammi	47

CAPITOLO 2

Creazione di script con ActionScript 49

Uso della sintassi di ActionScript	49
Informazioni sui tipi di dati	54
Informazioni sulle variabili	57
Uso di operatori per la gestione dei valori nelle espressioni	62
Uso delle azioni	69
Controllo del flusso negli script	71
Uso di funzioni predefinite	74
Creazione di funzioni personalizzate	76
Uso di oggetti predefiniti	79
Uso di oggetti personalizzati	83
Apertura di file Flash 4	86
Uso di Flash 5 per creare contenuto Flash 4	87

CAPITOLO 3

Creazione di contenuto interattivo con ActionScript . 89

Creazione di un puntatore del mouse personalizzato	90
Determinazione della posizione del mouse	92
Rilevamento dei tasti premuti	93
Creazione di un campo di testo scorrevole	95
Impostazioni dei valori dei colori	98
Creazione dei controlli audio	100
Rilevamento della presenza di collisioni	104

CAPITOLO 4

Uso dei clip filmato 107

Informazioni sulle linee temporali multiple	108
Informazioni sulle relazioni gerarchiche delle linee temporali	110

Scambio di messaggi tra linee temporali	112
Informazioni sui percorsi target assoluti o relativi	115
Specifica dei percorsi target	119
Uso di azioni e metodi per il controllo delle linee temporali	122
Informazioni sulle differenze tra metodi e azioni	123
Uso di più metodi o azioni sulla stessa linea temporale	124
Assegnazione di un'azione o di un metodo	125
Caricamento e scaricamento di filmati aggiuntivi	125
Modifica della posizione e dell'aspetto di un clip filmato	126
Trascinamento di clip filmato	127
Duplicazione ed eliminazione di clip filmato	128
Associazione di clip filmato	128
Creazione di clip intelligenti	129
Definizione dei parametri clip	130

CAPITOLO 5

Integrazione di Flash nelle applicazioni Web 137

Invio di variabili a e caricamento di variabili da un file remoto	137
Uso di loadVariables, getURL e loadMovie	141
Informazioni su XML	142
Uso dell'oggetto XML	143
Uso dell'oggetto XMLSocket	146
Creazione di moduli	147
Creazione di un modulo di ricerca	148
Uso di variabili nei moduli	149
Verifica dei dati immessi	150
Invio di messaggi a e da Flash Player	151
Uso di fscommand	152
Informazioni sui metodi Flash Player	154

CAPITOLO 6

Risoluzione dei problemi relativi ad ActionScript 155

Indicazioni per la creazione di codice e la risoluzione dei problemi	156
Uso del Debugger	158
Attivazione del debug in un filmato	158
Informazioni sulla barra di stato	160
Informazioni sulla lista di visualizzazione	160
Visualizzazione e modifica di variabili	160
Uso della lista di controllo	162

Visualizzazione delle proprietà dei filmati e modifica delle proprietà modificabili	162
Uso della finestra Output	164
Uso di Elenca oggetti	164
Uso di Elenca variabili	165
Uso di trace	166

CAPITOLO 7

Dizionario di ActionScript.	167
-------------------------------------	-----

Voci di esempio per la maggior parte degli elementi di ActionScript . .	168
Voci di esempio per gli oggetti.	169
Contenuto del dizionario	169
— (decremento)	182
++ (incremento)	183
! (NOT logico)	184
!= (diseguaglianza)	185
% (modulo)	186
%= (assegnazione modulo)	186
& (AND bit a bit).	187
&& (cortocircuito AND)	187
&= (assegnazione AND bit a bit)	188
() (parentesi tonde)	188
— (meno).	189
* (moltiplicazione)	190
*= (assegnazione moltiplicazione)	191
, (virgola)	191
. (operatore punto)	192
?: (condizionale)	193
/ (divisione).	193
// (delimitatore di commento).	194
/* (delimitatore di commento)	195
/= (assegnazione divisione)	195
[] (operatore di accesso matrice)	196
^(XOR bit a bit)	197
^= (assegnazione XOR bit a bit)	197
{ (operatore di inizializzazione degli oggetti).	198
(OR bit a bit)	199
(OR)	200
= (assegnazione OR bit a bit)	200
~ (NOT bit a bit)	201

+	(addizione)	201
+=	(assegnazione addizione)	202
<	(minore di)	203
<<	(spostamento a sinistra bit a bit)	204
<<=	(spostamento a sinistra bit a bit e assegnazione)	205
<=	(minore o uguale a)	205
<>	(disuguaglianza)	206
=	(assegnazione)	207
-=	(assegnazione negazione)	207
==	(uguaglianza)	208
>	(maggiore di)	209
>=	(maggiore o uguale a)	209
>>	(spostamento a destra bit a bit)	210
>>=	(spostamento a destra bit a bit e assegnazione)	211
>>>	(spostamento a destra senza segno bit a bit)	212
>>>=	(spostamento a destra senza segno bit a bit e assegnazione)	213
add		214
_alpha		214
and		215
Array (oggetto)		215
Array.concat		217
Array.join		218
Array.length		219
Array.pop		219
Array.push		220
Array.reverse		220
Array.shift		221
Array.splice		221
Array.sort		222
Array.splice		223
Array.toString		224
Array.unshift		225
Boolean (funzione)		225
Boolean (oggetto)		225
Boolean.toString		226
Boolean.valueOf		227
break		227
call		228
chr		228
Color (oggetto)		228

Color.setRGB	229
Color.getTransform	230
Color.setRGB	230
Color.setTransform	231
continue	232
_currentframe	233
Date (oggetto)	233
Date.getDate	237
Date.getDay	237
Date.getFullYear	237
Date.getHours	238
Date.getMilliseconds	238
Date.getMinutes	238
Date.getMonth	239
Date.getSeconds	239
Date.getTime	239
Date.getTimezoneOffset	240
Date.getUTCDate	240
Date.getUTCDay	241
Date.getUTCFullYear	241
Date.getUTCHours	241
Date.getUTCMilliseconds	242
Date.getUTCMinutes	242
Date.getUTCMonth	242
Date.getUTCSeconds	243
Date.getYear	243
Date.setDate	243
Date.setFullYear	244
Date.setHours	244
Date.setMilliseconds	245
Date.setMinutes	245
Date.setMonth	245
Date.setSeconds	246
Date.setTime	246
Date.setUTCDate	246
Date.setUTCFullYear	247
Date.setUTCHours	247
Date.setUTCMilliseconds	248
Date.setUTCMinutes	248
Date.setUTCMonth	248

Date.setUTCSeconds	249
Date.setYear	249
Date.toString	250
Date.UTC	250
delete	251
do... while	253
_droptarget	253
duplicateMovieClip	254
else	255
eq (uguale; specifico per stringhe)	256
escape	256
eval	257
evaluate	258
_focusrect	258
for	258
for..in	260
_framesloaded	261
fscommand	262
function	262
ge (maggiore di o uguale a; specifico per stringhe)	263
getProperty	264
getTimer	264
getURL	265
getVersion	266
gotoAndPlay	266
gotoAndStop	267
gt (maggiore di; specifico per stringhe)	267
_height	268
_highquality	268
if	269
ifFrameLoaded	269
#include	270
Infinity	270
int	271
isFinite	271
isNaN	272
Key (oggetto)	272
Key.BACKSPACE	274
Key.CAPSLOCK	274
Key.CONTROL	274

Key.DELETEKEY	275
Key.DOWN	275
Key.END	275
Key.ENTER	276
Key.ESCAPE	276
Key.getAscii	276
Key.getCode	277
Key.HOME	277
Key.INSERT	277
Key.isDown	278
Key.isToggled	278
Key.LEFT	278
Key.PGDN	279
Key.PGUP	279
Key.RIGHT	279
Key.SHIFT	280
Key.SPACE	280
Key.TAB	280
Key.UP	280
le (minore di o uguale a; specifico per stringhe)	281
length	281
_level	282
loadMovie	283
loadVariables	284
lt (minore di; specifico per stringhe)	285
Math (oggetto)	286
Math.abs	288
Math.acos	288
Math.asin	289
Math.atan	289
Math.atan2	289
Math.ceil	290
Math.cos	290
Math.E	291
Math.exp	291
Math.floor	292
Math.log	292
Math.LOG2E	292
Math.LOG10E	293
Math.LN2	293

Math.LN10	294
Math.max	294
Math.min	294
Math.PI	295
Math.pow	295
Math.random	296
Math.round	296
Math.sin	296
Math.sqrt	297
Math.SQRT1_2	297
Math.SQRT2	297
Math.tan	298
maxscroll	298
mbchr	299
mblength	299
mbord	299
mbsubstring	300
Mouse (oggetto)	300
Mouse.hide	301
Mouse.show	301
MovieClip (oggetto)	302
MovieClip.attachMovie	304
MovieClip.duplicateMovieClip	304
MovieClip.getBounds	305
MovieClip.getBytesLoaded	305
MovieClip.getBytesTotal	306
MovieClip.getURL	306
MovieClip.globalToLocal	307
MovieClip.gotoAndPlay	308
MovieClip.gotoAndStop	308
MovieClip.hitTest	308
MovieClip.loadMovie	310
MovieClip.loadVariables	310
MovieClip.localToGlobal	311
MovieClip.nextFrame	312
MovieClip.play	312
MovieClip.prevFrame	313
MovieClip.removeMovieClip	313
MovieClip.startDrag	313
MovieClip.stop	314

MovieClip.stopDrag	314
MovieClip.swapDepths	315
MovieClip.unloadMovie	315
_name	316
NaN	316
ne (non uguale; specifico per stringhe)	316
new	317
newline	318
nextFrame	318
nextScene	318
not	319
null	319
Number (funzione)	320
Number (oggetto)	320
Number.MAX_VALUE	322
Number.MIN_VALUE	323
Number.NaN	323
Number.NEGATIVE_INFINITY	323
Number.POSITIVE_INFINITY	324
Number.toString	324
Number.valueOf	324
Object (oggetto)	325
Object.toString	326
Object.valueOf	326
onClipEvent	326
on(mouseEvent)	328
or	329
ord	330
_parent	330
parseFloat	331
parseInt	332
play	333
prevFrame	333
prevScene	334
print	334
printAsBitmap	335
_quality	337
random	338
removeMovieClip	338
return	339

_root	339
_rotation	340
scroll	341
Selection (oggetto)	341
Selection.getBeginIndex	342
Selection.getCaretIndex	342
Selection.getEndIndex	343
Selection.getFocus	343
Selection.setFocus	344
Selection.setSelection	344
set	344
setProperty	346
Sound (oggetto)	346
Sound.attachSound	348
Sound.getPan	348
Sound.getTransform	348
Sound.getVolume	349
Sound.setPan	349
Sound.setTransform	350
Sound.setVolume	353
Sound.start	353
Sound.stop	354
_soundbuftime	354
startDrag	355
stop	355
stopAllSounds	356
stopDrag	356
String (funzione)	357
" " (delimitatore di stringa)	358
String (oggetto)	358
String.charAt	360
String.charCodeAt	360
String.concat	361
String.fromCharCode	361
String.indexOf	361
String.lastIndexOf	362
String.length	362
String.slice	362
String.split	363
String.substr	363

String.substring	364
String.toLowerCase	364
String.toUpperCase	365
substring	365
_target	365
targetPath	366
tellTarget	366
this	367
toggleHighQuality	368
_totalframes	369
trace	369
typeof	370
unescape	371
unloadMovie	371
updateAfterEvent	372
_url	373
var	373
_visible	373
void	374
while	374
_width	375
with	376
_x	379
XML (oggetto)	379
XML.appendChild	382
XML.attributes	382
XML.childNodes	383
XML.cloneNode	383
XML.createElement	384
XML.createTextNode	384
XML.docTypeDecl	385
XML.firstChild	386
XML.hasChildNodes	386
XML.insertBefore	387
XML.lastChild	387
XML.load	387
XML.loaded	388
XML.nextSibling	389
XML.nodeName	389
XML.nodeType	390

XML.nodeValue	390
XML.onLoad	390
XML.parentNode	391
XML.parseXML	392
XML.previousSibling	392
XML.removeNode	393
XML.send	393
XML.sendAndLoad	393
XML.status	394
XML.toString	395
XML.xmlDecl	395
XMLSocket (oggetto)	396
XMLSocket.close	399
XMLSocket.connect	399
XMLSocket.onClose	400
XMLSocket.onConnect	401
XMLSocket.onXML	402
XMLSocket.close	403
_xmouse	404
_xscale	404
_y	405
_ymouse	406
_yscale	406

APPENDICE A

Precedenza e associatività degli operatori 407

Lista operatori 407

APPENDICE B

Tasti della tastiera e valori dei codici tasto 411

Lettere dalla A alla Z e numeri standard da 0 a 9 412

Tasti sul tastierino numerico 414

Tasti funzione 414

Altri tasti 416

APPENDICE C

Messaggi di errore 419

INDICE ANALITICO 423



INTRODUZIONE

Guida introduttiva

.....

ActionScript è un linguaggio di scripting di Flash. È possibile usare ActionScript sia per controllare gli oggetti nei filmati Flash e creare così elementi di navigazione e interattivi, che per estendere le funzionalità di Flash e creare filmati e applicazioni Web altamente interattive.

Novità di ActionScript di Flash 5

ActionScript di Flash 5 offre nuove interessanti funzioni per la creazione di siti Web interattivi e coinvolgenti che contengono giochi sofisticati, moduli, questionari e sistemi in tempo reale come quelli di conversazione.

Le nuove funzioni e convenzioni sintattiche di ActionScript di Flash 5 lo rendono simile al linguaggio di programmazione JavaScript. In questo manuale sono introdotti i concetti di programmazione di base quali funzioni, variabili, istruzioni, operatori, espressioni condizionali e cicli. Il capitolo 7 di questo manuale, “Dizionario di ActionScript”, contiene una descrizione dettagliata per ogni elemento di ActionScript.

Lo scopo del presente manuale esula dall'insegnare la programmazione in generale, a tal fine sono disponibili numerose risorse che forniscono informazioni sui concetti generali di programmazione e sul linguaggio JavaScript.

L'Associazione europea dei produttori di computer (ECMA) ha stipulato il documento ECMA-262 derivato da JavaScript che rappresenta lo standard internazionale per il linguaggio JavaScript. ActionScript è basato sulla specifica ECMA-262 disponibile all'indirizzo <http://www.ecma.ch>.

Altra documentazione e articoli utili alla comprensione di ActionScript sono disponibili nel JavaScript Developer Central di Netscape DevEdge Online (<http://developer.netscape.com/tech/javascript/index.html>). Il miglior materiale di riferimento è il manuale Core JavaScript Guide all'indirizzo <http://developer.netscape.com/docs/manuals/js/core/jsguide/index.htm>.

Differenze tra ActionScript e JavaScript

Non occorre conoscere JavaScript per usare e apprendere ActionScript. Tuttavia una certa familiarità con JavaScript semplifica l'apprendimento di ActionScript. Di seguito sono elencate alcune delle differenze tra ActionScript e JavaScript:

- ActionScript non supporta oggetti specifici per determinati browser quali Document, Window e Anchor.
- ActionScript non supporta completamente tutti gli oggetti predefiniti di JavaScript.
- ActionScript supporta costrutti sintattici non consentiti in JavaScript (ad esempio le azioni `tellTarget` e `ifFrameLoaded` e la sintassi della barra rovesciata).
- ActionScript non supporta alcuni costrutti sintattici di JavaScript, ad esempio le etichette `switch`, `continue`, `try`, `catch`, `throw` e `statement`.
- ActionScript non supporta la funzione di costruzione `Function` di JavaScript.
- In ActionScript l'azione `eval` può solo fare riferimento a variabili.
- In JavaScript `toString` di `undefined` fornisce `undefined` come risultato. In Flash 5, per compatibilità con Flash 4, `toString` di `undefined` fornisce " " come risultato.
- In JavaScript la valutazione di `undefined` in un contesto numerico restituisce `NaN`. In Flash 5, per compatibilità con Flash 4, la valutazione di `undefined` restituisce 0.
- ActionScript non supporta Unicode; supporta invece ISO-8859-1 e il set di caratteri Shift-JIS.

Modifica del testo

È possibile immettere il testo degli script direttamente nel pannello Azioni in Modalità esperto. Alternativamente è possibile scegliere elementi da un menu a comparsa o da una lista come in Flash 4.

Sintassi del punto

È possibile usare questa sintassi per accedere e impostare le proprietà e i metodi degli oggetti, inclusi le istanze di clip filmato e le variabili. La sintassi del punto sostituisce quella della barra inclinata usata Flash 4 che, sebbene ancora supportata da Flash Player, non è consigliata.

Tipi di dati

ActionScript di Flash 5 supporta i seguenti tipi di dati: stringa, numerico, booleano, oggetto e clip filmato. I tipi di dati multipli consentono di usare diversi tipi di informazioni in ActionScript. Ad esempio, è possibile creare matrici e matrici associative.

Variabili locali

È possibile dichiarare variabili locali che non sono più valide alla fine di una lista di azioni o di una chiamata di funzione. Ciò consente di gestire meglio la memoria e riusare nomi di variabili. In Flash 4 le variabili erano permanenti, ossia anche le variabili temporanee come i contatori di ciclo rimanevano valide fino al completamento del filmato.

Funzioni definite dall'utente

È possibile definire funzioni con parametri che restituiscono valori permettendo in tal modo di riusare blocchi di codice negli script. In Flash 4 era possibile riusare codice tramite l'azione `call`, ma non era possibile passare parametri o ritornare valori.

Oggetti predefiniti

È possibile usare gli oggetti predefiniti per accedere a informazioni specifiche e gestirle. Di seguito sono elencati alcuni degli oggetti predefiniti di Flash:

- L'oggetto `Math` offre un insieme esauriente di costanti e funzioni matematiche incorporate quali `E` (costante di Eulero), `cos` (coseno) e `atan` (arcotangente).
- L'oggetto `Date` consente di ottenere informazioni sulla data e l'ora del sistema su cui è in esecuzione Flash Player.
- L'oggetto `Sound` consente di aggiungere audio a un filmato e controllarlo durante la riproduzione. Ad esempio, è possibile regolare il volume (`setVolume`) o il bilanciamento `setPan`).
- L'oggetto `Mouse` consente di nascondere il cursore standard in modo da poterne usare uno personalizzato.
- L'oggetto `MovieClip` consente di controllare i clip filmato senza usare azioni wrapper aggiuntive quali `tellTarget`. È possibile chiamare un metodo quale `play`, `loadMovie` o `duplicateMovieClip` da un nome di istanza tramite la sintassi del punto (ad esempio `myMovieClip.play()`).

Azioni clip

È possibile usare l'azione `onClipEvent` per assegnare azioni direttamente alle istanze di clip filmato sullo stage. L'azione `onClipEvent` gestisce eventi quali `load`, `enterFrame`, `mouseMove` e `data` che consentono di creare effetti interattivi avanzati.

Nuove azioni

È possibile usare nuove azioni quali `do..while` e `for` per creare cicli complessi. Altre nuove azioni sono state inserite come metodi dell'oggetto `MovieClip`; ad esempio `getBounds`, `attachMovie`, `hitTest`, `swapDepths` e `globalToLocal`.

Clip intelligenti

Ai clip intelligenti sono associati script che gli sviluppatori possono modificare senza usare il pannello Azioni. A tali clip intelligenti è possibile passare valori usando i parametri di clip definibili nella libreria.

Debugger

Il Debugger consente di visualizzare e modificare i valori di proprietà e variabili durante la riproduzione di un filmato in modalità di prova filmato, nella versione autonoma di Flash Player o in un browser Web, permettendo in tal modo di individuare eventuali problemi nel codice ActionScript.

Supporto XML

L'oggetto predefinito XML consente di convertire codice ActionScript in documenti XML e passarli ad applicazioni lato server oppure di caricare documenti XML in un filmato Flash e interpretarli. L'oggetto predefinito XMLSocket consente di creare una connessione continua a un server per passare dati XML per applicazioni in tempo reale.

Ottimizzazione

Flash 5 esegue semplici ottimizzazioni del codice ActionScript per migliorare le prestazioni e contenere la dimensione del file. Di conseguenza, nella maggior parte dei casi, il codice binario di ActionScript prodotto da Flash 5 ha dimensioni inferiori rispetto a quello generato da Flash 4.

Uso della Guida di Flash per le azioni

Flash 5 contiene una guida sensibile al contesto per ogni azione disponibile nel pannello Azioni. In tal modo durante la creazione di script è possibile ottenere informazioni sulle azioni che si stanno usando.

Per accedere alla guida per le azioni:

- 1** Nel pannello Azioni selezionare un'azione nella lista nel riquadro a sinistra.
- 2** Fare clic sul pulsante ? nella parte superiore del pannello.

L'argomento relativo all'azione verrà visualizzato nel browser.

CAPITOLO 1

Introduzione ad ActionScript

ActionScript, il linguaggio di scripting di Flash, consente di aggiungere interattività a un filmato. È possibile configurare il filmato in modo che gli eventi causati dall'utente, quali la selezione di pulsanti o la pressione di tasti, attivino gli script che comunicano al filmato l'azione da eseguire. Ad esempio, è possibile creare uno script che indica a Flash di caricare filmati diversi in Flash Player a seconda del pulsante di navigazione selezionato dall'utente.

ActionScript è uno strumento che consente di creare un filmato che si comporta esattamente come si desidera. Non occorre conoscere tutti gli usi possibili dello strumento per iniziare a creare script. È sufficiente avere un obiettivo chiaro per cominciare a creare script che contengono semplici azioni. Mentre si apprendono nuovi elementi del linguaggio, è possibile incorporarli per eseguire operazioni più complesse.

In questo capitolo viene presentato ActionScript come linguaggio di scripting orientato agli oggetti e viene fornita una panoramica dei termini relativi ad ActionScript. Viene inoltre analizzato uno script di esempio in modo che sia possibile concentrarsi sul quadro generale.

In questo capitolo viene introdotto anche il pannello Azioni, tramite il quale è possibile creare script selezionando elementi di ActionScript o immettendo testo nella finestra dello script.

Informazioni sulla creazione di script in ActionScript

È possibile iniziare a creare semplici script senza conoscere approfonditamente ActionScript. Tutto ciò che occorre è un obiettivo, definito il quale è solo questione di selezionare le azioni corrette. Il miglior modo per imparare a usare ActionScript è creare uno script. Di seguito viene descritto come associare uno script a un pulsante che modifica la visibilità di un clip filmato.

Per modificare la visibilità di un clip filmato:

- 1 Scegliere Finestra > Librerie comuni > Pulsanti, quindi scegliere Finestra > Librerie comuni > Clip filmato. Posizionare un pulsante e un clip filmato sullo stage.
- 2 Selezionare l'istanza del clip filmato sullo stage, quindi scegliere Finestra > Panelli > Istanza.
- 3 Nel campo Nome immettere **testMC**.
- 4 Selezionare il pulsante sullo stage, quindi scegliere Finestra > Azioni per aprire il pannello Azioni.
- 5 Nel pannello Azioni oggetto fare clic sulla categoria Azioni per aprirla.
- 6 Fare doppio clic sull'azione `setProperty` per aggiungerla alla lista delle azioni.
- 7 Dal menu a comparsa Proprietà scegliere `_visible` (visibilità).
- 8 Per il parametro Target immettere **testMC**.
- 9 Per il parametro Valore immettere **0**.

Il codice generato dovrebbe assomigliare al seguente:

```
on (release) {  
    setProperty ("testMC", _visible, false);  
}
```

- 10 Scegliere Controlli > Prova filmato, quindi fare clic sul pulsante per nascondere il clip filmato.

ActionScript è un linguaggio di scripting orientato agli oggetti in quanto le azioni controllano gli oggetti quando si verifica un determinato evento. In questo script l'evento è il rilascio del pulsante del mouse, l'oggetto è l'istanza del clip filmato MC e l'azione è `setProperty`. Quando l'utente seleziona il pulsante visualizzato sullo schermo, l'evento `release` attiva uno script che imposta la proprietà `_visible` dell'oggetto MC su `false` e rende l'oggetto invisibile.

È possibile usare il pannello Azioni per imparare a creare semplici script. Per sfruttare appieno le potenzialità di ActionScript, è importante conoscere il linguaggio, ossia i concetti, gli elementi e le regole che il linguaggio usa per organizzare le informazioni e creare filmati interattivi.

In questa sezione vengono descritti il flusso di lavoro di ActionScript, i concetti fondamentali dello scripting orientato agli oggetti, gli oggetti di Flash e il flusso logico degli script. Viene inoltre spiegato dove risiedono gli script in un filmato Flash.

Informazioni sulla progettazione e sul debug di script

Quando si creano script per un intero filmato, la quantità e la varietà di script può essere considerevole. Decidere quali azioni usare, come strutturare efficacemente gli script e dove posizionarli richiede una progettazione approfondita e un'esauritiva fase di prova, soprattutto al crescere della complessità del filmato.

Prima di iniziare a creare script, definire l'obiettivo da raggiungere e stabilire che cosa si desidera realizzare. Questa fase è importante quanto sviluppare le storyboard per il progetto e, solitamente, richiede lo stesso impegno. Per iniziare, delineare che cosa succederà nel filmato, come nell'esempio seguente.

- Creazione dell'intero sito usando Flash.
- I visitatori del sito dovranno fornire il loro nome, che verrà riusato in tutto il sito nei messaggi.
- Il sito includerà una barra di navigazione mobile con pulsanti che consentono di passare alle varie sezioni disponibili all'interno del sito.
- Quando si seleziona un pulsante, la nuova sezione apparirà con effetto dissolvenza al centro dello stage.
- Una scena conterrà un modulo di contatto con il nome dell'utente già inserito.

Una volta stabilito l'obiettivo, è possibile creare gli oggetti necessari e scrivere gli script per controllare tali oggetti.

La creazione di script che funzionano come desiderato richiede tempo, spesso più di un ciclo di scrittura, prova e debug. L'approccio migliore consiste nell'iniziare con script semplici ed eseguire prove frequenti. Quando una porzione dello script funziona correttamente, scegliere Salva con nome per salvare una versione del file (ad esempio Filmato01.fla) e iniziare a scrivere la parte successiva. Questo approccio consentirà di identificare efficientemente gli errori e di garantire la stabilità del codice ActionScript anche quando si creano script più complessi.

Informazioni sullo scripting orientato agli oggetti

Nello scripting orientato agli oggetti le informazioni vengono organizzate in gruppi detti *classi*. È possibile creare più istanze di una classe, dette *oggetti*, da usare negli script. È possibile usare le classi predefinite di ActionScript o crearne di personalizzate.

Quando si crea una classe, si definiscono tutte le *proprietà* (caratteristiche) e tutti i *metodi* (comportamenti) di ciascun oggetto creato, proprio come si definiscono gli oggetti del mondo reale. Ad esempio, una persona è caratterizzata da proprietà come sesso, altezza e colore di capelli e metodi come parlare, camminare e lanciare. In questo esempio “persona” è una classe e ogni singolo individuo è un oggetto o un'istanza della classe.

Gli oggetti in ActionScript possono contenere dati o essere rappresentati graficamente sullo stage come clip filmato. Tutti i clip filmato sono istanze della classe predefinita MovieClip. Ogni istanza di un clip filmato contiene tutte le proprietà (ad esempio `_height`, `_rotation`, `_totalframes`) e tutti i metodi (ad esempio `gotoAndPlay`, `loadMovie`, `startDrag`) della classe MovieClip.

Per definire una classe, creare una funzione speciale detta *funzione di costruzione*. Le funzioni di costruzione per le classi predefinite sono già definite. Ad esempio, se si desiderano informazioni su un ciclista del filmato, è possibile creare una funzione di costruzione, `Biker`, con le proprietà `time` e `distance` e il metodo `rate`, che indica la velocità a cui viaggia il ciclista:

```
function Biker(t, d) {
    this.time = t;
    this.distance = d;
}
function Speed() {
    return this.time / this.distance;
}
Biker.prototype.rate = Speed;
```

È possibile quindi creare copie, cioè istanze, della classe. Il codice seguente crea due istanze dell'oggetto `Biker`, `emma` e `hamish`.

```
emma = new Biker(30, 5);
hamish = new Biker(40, 5);
```

Le istanze possono anche comunicare tra loro. Per l'oggetto `Biker` è possibile creare un metodo chiamato `shove` che consente a un ciclista di spingere un altro ciclista. L'istanza `emma` può quindi invocare il metodo `shove` se `hamish` si avvicina troppo. Per passare le informazioni a un metodo, usare dei parametri (argomenti). Ad esempio, il metodo `shove` potrebbe richiedere i parametri *who* e *howFar*. In questo esempio `emma` spinge `hamish` e lo sposta di 10 pixel:

```
emma.shove(hamish, 10);
```

Nello scripting orientato agli oggetti le classi possono scambiarsi proprietà e metodi in base a un ordine specifico. Questa funzionalità è detta *ereditarietà*. È possibile usare l'ereditarietà per aumentare o ridefinire le proprietà e i metodi di una classe. Una classe che eredita proprietà o metodi da un'altra classe è detta *sottoclasse*. Una classe che passa proprietà o metodi a un'altra classe è detta *superclasse*. Una classe può essere sia una sottoclasse che una superclasse.

Informazioni sull'oggetto MovieClip

Le classi predefinite di ActionScript sono dette *oggetti*. Ogni oggetto consente di accedere a un determinato tipo di informazioni. Ad esempio, l'oggetto Date dispone di metodi (quali `getFullYear` e `getMonth`) che consentono di leggere le informazioni dall'orologio di sistema. L'oggetto Sound dispone di metodi (quali `setVolume` e `setPan`) che consentono di regolare l'audio in un filmato. L'oggetto MovieClip dispone di metodi che consentono di controllare le istanze di clip filmato (quali `play`, `stop` e `getURL`) e sia leggere che impostare informazioni sulle loro proprietà (quali `_alpha`, `_framesloaded` e `_visible`).

I clip filmato sono gli oggetti più importanti di un filmato Flash in quanto includono linee temporali che funzionano indipendentemente l'una dall'altra. Ad esempio, se la linea temporale principale ha un solo fotogramma e un clip filmato in quel fotogramma comprende 10 fotogrammi, ogni fotogramma nel clip filmato verrà comunque riprodotto. Ciò consente alle istanze di agire come oggetti indipendenti in grado di comunicare tra loro.

Le istanze di clip filmato hanno un nome univoco in modo che sia possibile identificarle come target quando si esegue un'azione. Ad esempio, sullo stage potrebbero essere disponibili più istanze (ad esempio `leftClip` e `rightClip`), ma si desidera riprodurre solo una alla volta. Per riprodurre una determinata istanza, è necessario identificarla tramite il relativo nome per poterle assegnare un'azione. Nell'esempio seguente il nome del clip filmato è `leftClip`:

```
leftClip.play();
```

I nomi delle istanze consentono inoltre di duplicare, rimuovere e trascinare clip filmato durante la riproduzione di un filmato. Nell'esempio seguente l'istanza `cartItem` viene duplicata per riempire il carrello con il numero di articoli acquistati:

```
onClipEvent(load) {  
    do {  
        duplicateMovieClip("cartItem", "cartItem" + i, i);  
        i = i + 1;  
    } while (i <= numberItemsPur);  
}
```

I valori delle proprietà dei clip filmato possono essere impostati e recuperati dinamicamente tramite ActionScript. La modifica e la lettura di queste proprietà può cambiare l'aspetto e l'identità di un clip filmato e rappresenta lo strumento per creare contenuto interattivo. Ad esempio, lo script seguente usa l'azione `setProperty` per impostare la trasparenza (impostazione alfa) dell'istanza `navigationBar` su 10:

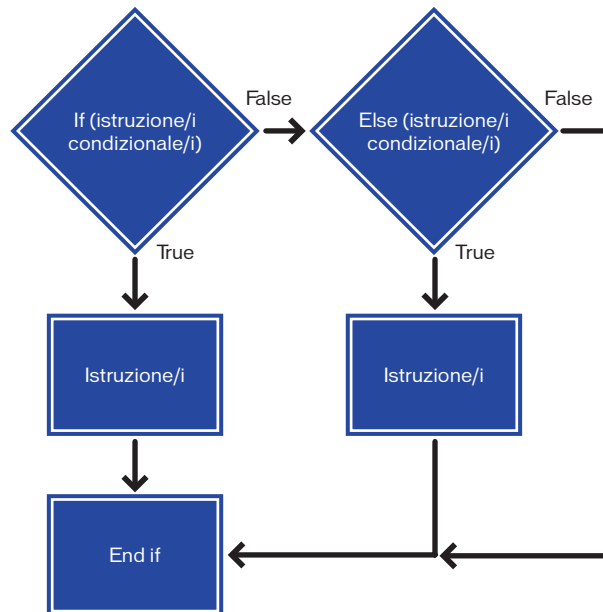
```
setProperty("navigationBar", _alpha, 10);
```

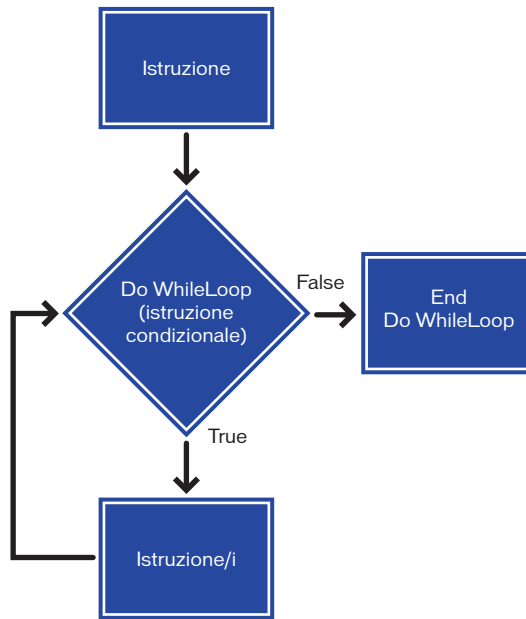
Per ulteriori informazioni sugli altri tipi di oggetti, vedere “Uso di oggetti predefiniti” a pagina 79.

Flusso logico degli script

ActionScript segue un flusso logico. Flash esegue le istruzioni ActionScript a partire dalla prima e continuando in ordine fino a quando non raggiunge l'istruzione finale o un'istruzione che ordina ad ActionScript di spostarsi in un punto diverso.

Alcune delle azioni che spostano il flusso di ActionScript in una posizione diversa dall'istruzione successiva sono le istruzioni `if`, le ripetizioni cicliche `do...while` e l'azione `return`.





L'istruzione `if` è detta istruzione condizionale o “salto logico”, in quanto controlla il flusso di uno script in base alla valutazione di una certa condizione. Ad esempio, il codice seguente verifica se il valore della variabile `number` è minore o uguale a 10. Se il controllo restituisce `true` (ad esempio se il valore di `number` è 5), la variabile `alert` viene impostata e ne viene visualizzato il valore in un campo di testo di input.

```

if (number <= 10) {
    alert = "The number is less than or equal to 10";
}
  
```

È inoltre possibile aggiungere istruzioni `else` per creare un'istruzione condizionale più complessa. Nell'esempio seguente se la condizione restituisce `true` (ad esempio se il valore di `number` è 3), viene eseguita l'istruzione tra il primo gruppo di parentesi graffe e la variabile `alert` viene impostata sulla prima stringa. Se la condizione restituisce `false` (ad esempio se il valore di `number` è 30), il primo blocco di codice viene saltato e viene eseguita l'istruzione tra parentesi graffe dopo l'istruzione `else`.

```

if (number <= 10) {
    alert = "The number is less than or equal to 10";
} else {
    alert = "The number is greater than 10";
}
  
```

Per ulteriori informazioni, consultare “Uso di istruzioni if” a pagina 71.

Le ripetizioni cicliche eseguono un'azione un determinato numero di volte o fino a quando viene soddisfatta una certa condizione. Nell'esempio seguente un clip filmato viene duplicato cinque volte:

```
i = 0;
do {
    duplicateMovieClip ("myMovieClip", "newMovieClip" + i, i);
    newName = eval("newMovieClip" + i);
    setProperty(newName, _x, getProperty("myMovieClip", _x) + (i *
5));
    i = i + 1;
} while (i <= 5);
```

Per ulteriori informazioni, consultare “Ripetizione di un'azione” a pagina 72.

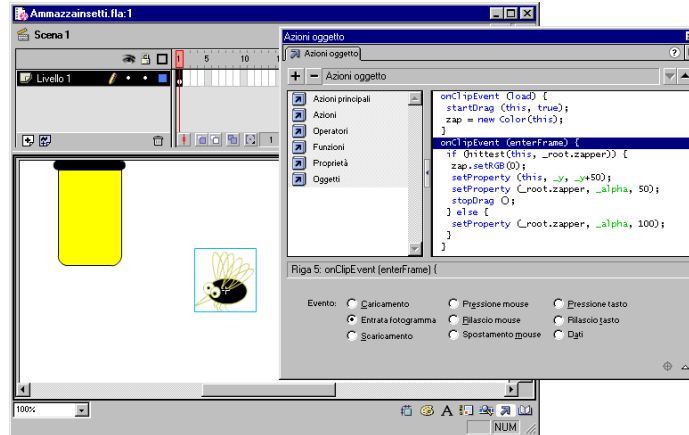
Controllo dell'esecuzione di ActionScript

Per creare uno script si usa il pannello Azioni per associare lo script a un fotogramma sulla linea temporale principale o di un clip filmato, a un pulsante oppure a un clip filmato sullo stage.

Flash esegue le azioni in momenti diversi a seconda dell'elemento a cui sono associate.

- Le azioni associate a un fotogramma vengono eseguite quando l'indicatore di riproduzione entra nel fotogramma.
- Le azioni associate a un pulsante sono incluse in un gestore `on`.
- Le azioni associate a un clip filmato sono incluse in un gestore `onClipEvent`.

Le azioni `onClipEvent` e `on` sono chiamate gestori perché gestiscono un evento. Un evento è un'azione che avviene, ad esempio lo spostamento del mouse, la pressione di un tasto o il caricamento di un clip filmato. Le azioni associate ai clip filmato e ai pulsanti vengono eseguite quando si verifica l'evento specificato dal gestore. È possibile associare più gestori a un oggetto se si desidera che le azioni vengano eseguite quando si verificano eventi diversi. Per ulteriori informazioni, vedere il capitolo 3 “Creazione di contenuto interattivo con ActionScript”.



Diversi gestori `onClipEvent` associati a un clip filmato sullo stage.

Terminologia di ActionScript

Come tutti i linguaggi di scripting, ActionScript usa una terminologia particolare in base a regole di sintassi specifiche. Nella lista seguente, in ordine alfabetico, sono descritti brevemente i termini di ActionScript più importanti. Questi termini e la sintassi che li regola sono trattati approfonditamente nel capitolo 2 “Creazione di script con ActionScript”

Azioni: istruzioni che indicano a un filmato di eseguire una determinata operazione durante la riproduzione. Ad esempio, `gotoAndStop` sposta l'indicatore di riproduzione su determinato fotogramma o etichetta. Nel presente manuale i termini *azione* e *istruzione* sono intercambiabili.

Argomenti (detti anche parametri): segnaposto che consentono di passare valori a funzioni. Ad esempio, la seguente funzione `welcome` usa due valori che riceve come argomenti `firstName` e `hobby`:

```
function welcome(firstName, hobby) {  
    welcomeText = "Hello, " + firstName + "I see you enjoy " +  
    hobby;  
}
```

Classi: tipi di dati che è possibile creare per definire un nuovo tipo di oggetto. Per definire una classe di oggetti, è necessario creare una funzione di costruzione.

Costanti: elementi che non cambiano valore. Ad esempio, la costante `TAB` ha sempre lo stesso significato. Le costanti sono utili per confrontare valori.

Costruttori: funzioni che consentono di definire le proprietà e i metodi di una classe. Ad esempio, il codice seguente consente di creare una nuova classe `Circle` creando una funzione di costruzione `Circle`:

```
function Circle(x, y, radius){  
    this.x = x;  
    this.y = y;  
    this.radius = radius;  
}
```

Tipi di dati: insieme di valori e operazioni che possono essere eseguite con questi valori. I tipi di dati di ActionScript sono stringa, numerico, booleano (`true` e `false`), oggetto e clip filmato. Per ulteriori informazioni su questi elementi del linguaggio, vedere “Informazioni sui tipi di dati” a pagina 54.

Eventi: azioni che si verificano durante la riproduzione di un filmato. Ad esempio, eventi diversi vengono generati quando si carica un clip filmato, l'indicatore di riproduzione entra in un fotogramma, l'utente seleziona un pulsante o un clip filmato oppure l'utente digita sulla tastiera.

Espressioni: qualsiasi parte di un'istruzione che genera un valore. Ad esempio, `2 + 2` è un'espressione.

Funzioni: blocchi di codice riutilizzabile a cui è possibile passare argomenti (parametri) e che possono restituire un valore. Ad esempio, se alla funzione `getProperty` vengono passati il nome di una proprietà e il nome dell'istanza di un clip filmato, restituisce il valore della proprietà. La funzione `getVersion` restituisce la versione di Flash Player usata attualmente per riprodurre il filmato.

Gestori: azioni speciali che gestiscono un evento quale `mouseDown` o `load`. Ad esempio, `on` (`onMouseEvent`) e `onClipEvent` sono gestori di ActionScript.

Identificatori: nomi usati per indicare una variabile, una proprietà, un oggetto, una funzione o un metodo. Il primo carattere deve essere una lettera, un carattere di sottolineatura (`_`) o il simbolo del dollaro (`$`). Ogni carattere successivo deve essere una lettera, un numero, un carattere di sottolineatura (`_`) o il simbolo del dollaro (`$`). Ad esempio, `firstName` è il nome di una variabile.

Istanze: oggetti che appartengono a una determinata classe. Ogni istanza di una classe contiene tutte le proprietà e tutti i metodi della classe. Tutti i clip filmato sono istanze con le proprietà (ad esempio `_alpha` e `_visible`) e i metodi (ad esempio `gotoAndPlay` e `getURL`) della classe `MovieClip`.

Nomi di istanze: nomi univoci che consentono di identificare istanze di un clip filmato negli script. Ad esempio, nella libreria esiste un simbolo master di nome `counter` dal quale sono state create due istanze nel filmato chiamate `scorePlayer1` e `scorePlayer2`. Il codice seguente imposta la variabile `score` all'interno di ciascuna istanza del clip filmato usando i nomi delle istanze:

```
_root.scorePlayer1.score += 1  
_root.scorePlayer2.score -= 1
```

Parole chiave: parole riservate che hanno un significato speciale. Ad esempio, `var` è una parola chiave usata per dichiarare le variabili locali.

Metodi: funzioni assegnate a un oggetto. Dopo aver assegnato una funzione, è possibile chiamarla come metodo dell'oggetto. Ad esempio, nel codice seguente `clear` diventa un metodo dell'oggetto `controller`:

```
function Reset(){  
    x_pos = 0;  
    x_pos = 0;  
}  
controller.clear = Reset;  
controller.clear();
```

Oggetti: raccolte di proprietà. Ogni oggetto ha un proprio nome e valore. Gli oggetti consentono di accedere a un determinato tipo di informazioni. Ad esempio, l'oggetto `Date` predefinito fornisce informazioni provenienti dall'orologio di sistema.

Operatori: termini che calcolano un nuovo valore in base a uno o più valori. Ad esempio, l'operatore di addizione (+) consente di sommare due o più valori per produrre un nuovo valore.

Percorsi target: indirizzi gerarchici dei nomi delle istanze di clip filmato, delle variabili e degli oggetti di un filmato. È possibile assegnare un nome a un'istanza di clip filmato nel pannello Istanza. La linea temporale principale si chiama sempre `_root`. È possibile usare un percorso target per dirigere un'azione su un clip filmato oppure per ottenere o impostare il valore di una variabile. Ad esempio, l'istruzione seguente è il percorso target della variabile `volume` all'interno del clip filmato `stereoControl`:

```
_root.stereoControl.volume
```

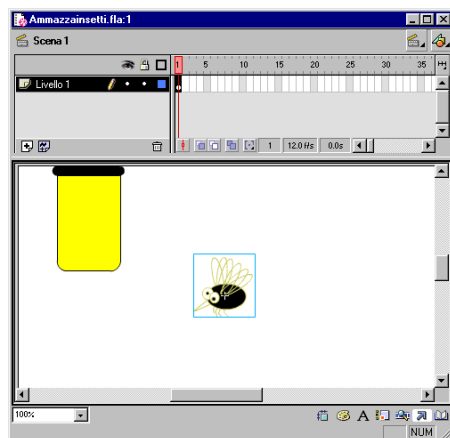
Proprietà: attributi che definiscono un oggetto. Ad esempio, `_visible` è una proprietà di tutti i clip filmato che definisce se il clip filmato è visibile o nascosto.

Variabili: identificatori che contengono i valori di qualsiasi tipo di dati. Le variabili possono essere create, modificate e aggiornate. È possibile accedere ai valori in esse contenuti e usarli negli script. Nell'esempio seguente gli identificatori a sinistra del segno di uguale sono variabili:

```
x = 5;  
name = "Lolo";  
customer.address = "66 7th Street";  
c = new Color(mcinstanceName);
```

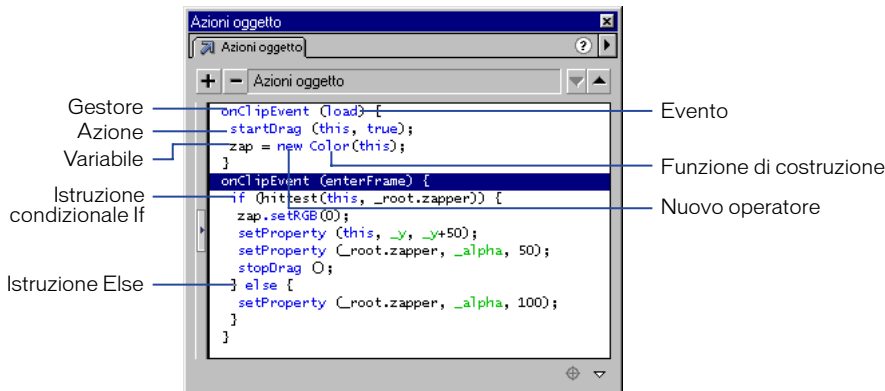
Analisi di uno script di esempio

In questo filmato di esempio, quando l'utente trascina l'insetto nell'ammazzainsetti, l'insetto diventa nero e cade mentre l'ammazzainsetti lampeggia. Il filmato è composto da un fotogramma e contiene due oggetti, l'istanza del clip filmato dell'insetto e l'istanza del clip filmato dell'ammazzainsetti. Ogni clip filmato contiene un fotogramma.



Le istanze dei clip filmato dell'insetto e dell'ammazzainsetti sullo stage nel fotogramma 1.

Il filmato contiene solo uno script, che è associato all'istanza bug, come illustrato nel pannello Azioni oggetto sottostante.



Il pannello Azioni oggetto con lo script associato all'istanza bug.

Entrambi gli oggetti devono essere clip filmato in modo che sia possibile assegnare loro nomi di istanze nel pannello Istanza e controllarli tramite ActionScript. Il nome dell'istanza dell'insetto è bug, mentre il nome dell'istanza dell'ammazzainsetti è zapper. Nello script si fa riferimento all'insetto con `this` in quanto lo script è associato all'insetto e la parola riservata `this` si riferisce all'oggetto che la invoca.

Lo script include due gestori `onClipEvent` con due eventi diversi: `load` e `enterFrame`. Le azioni nell'istruzione `onClipEvent(load)` vengono eseguite solo una volta, al caricamento del filmato. Le azioni nell'istruzione `onClipEvent(enterFrame)` vengono eseguite ogni volta che l'indicatore di riproduzione entra in un fotogramma. Anche nei filmati composti da un solo fotogramma, l'indicatore di riproduzione entra nel fotogramma più volte e lo script viene eseguito ripetutamente. Di seguito sono descritte le azioni che si verificano all'interno di ciascun gestore `onClipEvent`.

onClipEvent(load) Un'azione `startDrag` rende il clip filmato dell'insetto trascinabile. Viene creata un'istanza dell'oggetto `Color` tramite l'operatore `new` e la funzione di costruzione `Color`, `Color`, quindi tale istanza viene assegnata alla variabile `zap`:

```
onClipEvent (load) {
    startDrag (this, true);
    zap = new Color(this);
}
```

onClipEvent(enterFrame) Un'istruzione condizionale `if` valuta un'azione `hitTest` per verificare se l'istanza dell'insetto (`this`) tocca l'istanza dell'ammazzainsetti (`_root.zapper`). La valutazione può produrre due risultati: `true` o `false`:

```
onClipEvent (enterFrame) {  
    if (hitTest(_target, _root.zapper)) {  
        zap.setRGB(0);  
        setProperty (_target, _y, _y+50);  
        setProperty (_root.zapper, _alpha, 50);  
        stopDrag ();  
    } else {  
        setProperty (_root.zapper, _alpha, 100);  
    }  
}
```

Se l'azione `hitTest` restituisce `true`, l'oggetto `zap` creato dall'evento `load` verrà usato per impostare il colore dell'insetto su nero. La proprietà `y` dell'insetto (`_y`) è impostata su un valore pari a quello corrente più 50, in modo che l'insetto cada. La trasparenza dell'ammazzainsetti (`_alpha`) è impostata su 50 in modo che diventi opaco. L'azione `stopDrag` impedisce l'ulteriore trascinamento dell'insetto.

Se l'azione `hitTest` restituisce `false`, verrà eseguita l'azione dopo l'istruzione `else` e il valore `_alpha` dell'ammazzainsetti verrà impostato su 100. In tal modo l'ammazzainsetti lampeggia mentre il valore `_alpha` passa dallo stato iniziale (100) allo stato "ammazzato" (50) e quindi di nuovo allo stato iniziale. L'azione `hitTest` restituisce `false` e le istruzioni `else` vengono eseguite dopo che l'insetto è stato ammazzato ed è caduto.

Per riprodurre il filmato, consultare la *Guida di Flash*.

Uso del pannello Azioni

Il pannello Azioni consente di creare e modificare le azioni relative a un oggetto o a un fotogramma usando due modalità di modifica diverse. È possibile selezionare azioni predefinite dalla lista nel riquadro a sinistra, trascinare e rilasciare azioni e usare i pulsanti per eliminare o riordinare le azioni. In Modalità normale è possibile creare azioni usando i campi dei parametri (argomenti) che richiedono gli argomenti appropriati. In Modalità esperto è possibile creare e modificare le azioni direttamente in una casella di testo, come se si creassero script in un editor di testo.

Per visualizzare il pannello Azioni:

Scegliere Finestra > Azioni.

La selezione di un'istanza di un pulsante o di un clip filmato rende attivo il pannello Azioni. Il titolo del pannello Azioni diventa Azioni oggetto se si seleziona un pulsante o un clip filmato e Azioni fotogramma se si seleziona un fotogramma.

Per selezionare una modalità di modifica:

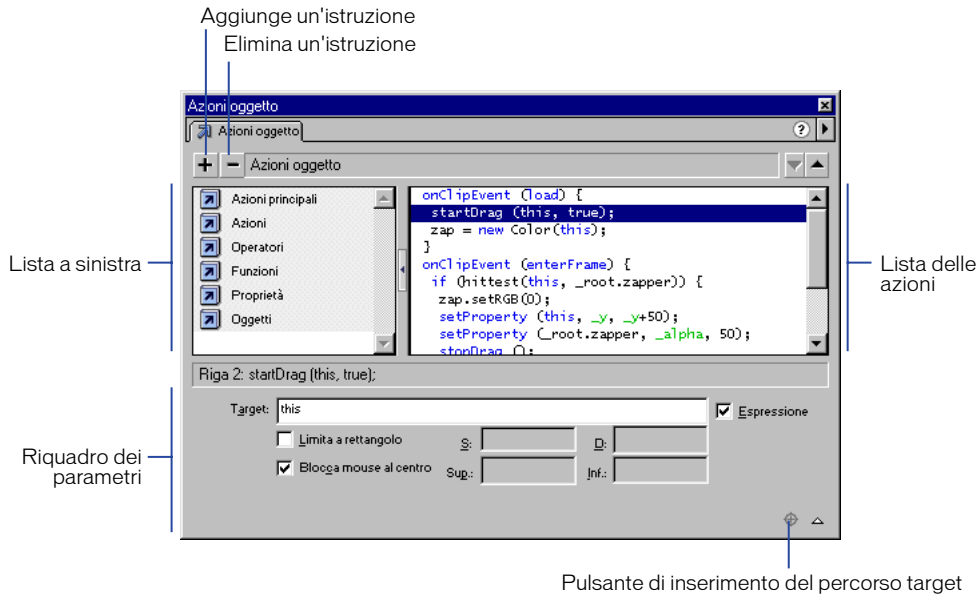
- 1 Nel pannello Azioni, fare clic sulla freccia nell'angolo superiore destro per visualizzare il menu a comparsa.
- 2 Scegliere Modalità normale o Modalità esperto dal menu a comparsa.

Ogni script mantiene la propria modalità. Ad esempio, è possibile creare lo script di un'istanza di un pulsante in Modalità normale e lo script di un'istanza di un altro pulsante in Modalità esperto. Quando si passa da un pulsante selezionato all'altro, cambia la modalità del pannello.

Modalità normale

In Modalità normale è possibile creare azioni selezionandole dalla lista nel riquadro a sinistra. La lista nel riquadro a sinistra contiene le categorie Azioni principali, Azioni, Operatori, Funzioni, Proprietà e Oggetti. La categoria Azioni principali a sua volta contiene le azioni di Flash più semplici ed è disponibile solo in Modalità normale. Le azioni selezionate sono elencate sul lato destro del pannello, nella lista delle azioni. È possibile aggiungere, eliminare o modificare l'ordine delle istruzioni delle azioni, nonché immettere parametri (argomenti) per le azioni negli appositi campi situati nella parte inferiore del pannello.

In Modalità normale è possibile usare i controlli del pannello Azioni per eliminare o modificare l'ordine delle istruzioni nella lista Azioni. Questi controlli sono particolarmente utili per la gestione delle azioni relative a fotogrammi o pulsanti che contengono diverse istruzioni.



Il pannello Azioni in Modalità normale.

Per selezionare un'azione:

- 1 Fare clic su una categoria Azioni nella lista nel riquadro a sinistra per visualizzare le azioni corrispondenti.
- 2 Fare doppio clic sull'azione desiderata oppure trascinarla nella finestra dello script.

Per usare i campi dei parametri:

- 1 Fare clic sul pulsante a forma di freccia rivolta verso il basso nell'angolo inferiore destro del pannello Azioni per visualizzare i campi
- 2 Selezionare l'azione desiderata e immettere i nuovi valori nei campi dei parametri per modificare i parametri delle azioni esistenti.

Per inserire il percorso target di un clip filmato:

- 1 Fare clic sul pulsante a forma di mirino nell'angolo inferiore destro del pannello Azioni per visualizzare la finestra di dialogo Inserisci percorso target.
- 2 Selezionare un clip filmato dalla lista di visualizzazione.

Per spostare un'istruzione all'interno della lista:

- 1 Selezionare l'istruzione desiderata dalla lista delle azioni.
- 2 Fare clic sul pulsante Freccia su o Freccia giù.

Per eliminare un'azione:

- 1 Selezionare l'istruzione desiderata dalla lista delle azioni.
- 2 Fare clic sul pulsante di eliminazione (–).

Per modificare i parametri delle azioni esistenti:

- 1 Selezionare l'istruzione desiderata dalla lista delle azioni.
- 2 Immettere nuovi valori nei campi dei parametri.

Per ridimensionare la lista nel riquadro a sinistra o a destra, eseguire una delle operazioni descritte.

- Trascinare la barra di divisione verticale tra le due liste.
- Fare doppio clic sulla barra di divisione per comprimere la lista nel riquadro a sinistra. Fare di nuovo doppio clic sulla barra per visualizzare nuovamente la lista.
- Fare clic sul pulsante Freccia sinistra o Freccia destra sulla barra di divisione per espandere o comprimere la lista.

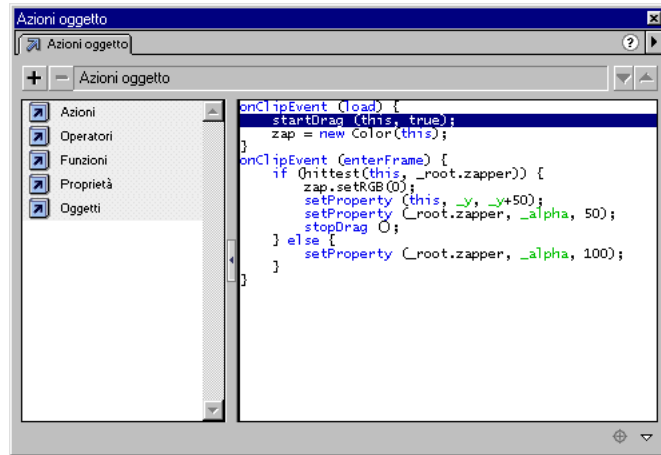
Quando la lista nel riquadro a sinistra è nascosta, è sempre possibile accedere agli elementi in essa contenuti tramite il pulsante di aggiunta (+) nella parte superiore sinistra del pannello Azioni.

Modalità esperto

In Modalità esperto, è possibile creare azioni immettendo codice ActionScript nella casella di testo sul lato destro del pannello o selezionando le azioni desiderate dalla lista nel riquadro a sinistra. È possibile modificare le azioni, immettere parametri per le azioni oppure eliminare le azioni direttamente nella casella di testo, come se si creassero script in un editor di testo.

La Modalità esperto consente agli utenti esperti di ActionScript di modificare gli script con un editor di testo, come farebbero con JavaScript o VBScript. Questa modalità differisce dalla Modalità normale nei seguenti aspetti.

- Se si seleziona un elemento dal menu a comparsa Aggiungi o dalla lista nel riquadro a sinistra, l'elemento viene inserito nella casella di testo.
- Non viene visualizzato alcun campo di parametri.
- Il pulsante di aggiunta (+) è l'unico pulsante attivo nell'area pulsanti.
- I pulsanti Freccia su e Freccia giù rimangono disattivati.



Il pannello Azioni in Modalità esperto

Passaggio da una modalità di modifica all'altra

Il passaggio da una modalità di modifica all'altra durante la creazione di uno script può cambiare la formattazione dello script. Per questo motivo si consiglia di usare un'unica modalità di modifica per script.

Quando si passa dalla Modalità normale alla Modalità esperto, vengono mantenuti il rientro e la formattazione. Sebbene sia possibile convertire gli script con errori dalla Modalità normale alla Modalità esperto, prima di esportare gli script è necessario correggere gli errori.

Il passaggio dalla Modalità esperto alla Modalità normale è leggermente più complesso.

- Quando si passa alla Modalità normale, Flash riformatta lo script ed elimina tutti gli spazi vuoti e i rientri aggiunti.
- Se si passa alla Modalità normale e poi di nuovo alla Modalità esperto, Flash riformatta lo script in base al suo aspetto in Modalità normale.
- Non è possibile esportare o convertire nella Modalità normale gli script creati in Modalità esperto che contengono errori. Un tentativo di conversione genererà un messaggio di errore.

Per passare da una modalità di modifica all'altra:

Scegliere Modalità normale o Modalità esperto dal menu a comparsa accessibile nell'angolo superiore destro del pannello Azioni. Un segno di spunta indica la modalità selezionata.

Per impostare la modalità di modifica predefinita:

- 1 Scegliere Modifica > Preferenze.
- 2 Selezionare la scheda Generale
- 3 Nell'area Pannello Azioni selezionare Modalità normale o Modalità esperto dal menu a comparsa.

Uso di un editor esterno

Sebbene la Modalità esperto del pannello Azioni offra un maggiore controllo durante la modifica di codice `JavaScript`, è possibile scegliere di modificare uno script al di fuori di Flash. È quindi possibile usare l'azione `include` per aggiungere gli script creati nell'editor esterno a uno script di Flash.

Ad esempio, l'istruzione seguente importa un file di script:

```
#include "externalfile.as"
```

Il testo del file di script sostituisce l'azione `include`. Quando si esporta il filmato, il file di testo deve essere disponibile.

Per aggiungere gli script creati in un editor esterno a uno script di Flash:

- 1 Trascinare `include` dalla lista nel riquadro a sinistra nella finestra dello script.
- 2 Immettere il percorso del file esterno nella casella Percorso.

Il percorso deve essere relativo al file FLA. Ad esempio, se Filmato.fla e fileesterno.as sono nella stessa cartella, il percorso sarà fileesterno.as. Se fileesterno.as è in una sottocartella chiamata script, il percorso sarà script/fileesterno.as.

Selezione delle opzioni del pannello Azioni

Il pannello Azioni consente di creare e modificare gli script in modi diversi. Nella finestra dello script è possibile cambiare la dimensione del carattere. È possibile importare un file di testo contenente codice `JavaScript` nel pannello Azioni ed esportare azioni come file di testo; trovare e sostituire del testo in uno script e usare l'evidenziazione della sintassi per rendere gli script più facili da leggere e semplificare l'identificazione degli errori. Il pannello Azioni visualizza evidenziazioni di avvertenza per segnalare errori di sintassi e incompatibilità di versione di Flash Player. Inoltre evidenzia gli elementi *obsoleti* di `JavaScript`, di cui è sconsigliato l'uso.

Queste opzioni del pannello Azioni sono disponibili sia in Modalità normale che in Modalità esperto a meno che non sia specificato diversamente.

Per cambiare la dimensione del carattere nella finestra dello script:

- 1 Dal menu a comparsa accessibile nell'angolo superiore destro del pannello Azioni scegliere Dimensione carattere.
- 2 Selezionare Piccolo, Medio o Grande.

Per importare un file di testo contenente codice ActionScript:

- 1 Dal menu a comparsa accessibile nell'angolo superiore destro del pannello Azioni scegliere Importa da file.
- 2 Selezionare il file di testo contenente il codice ActionScript, quindi fare clic su Apri.

Nota: gli script con errori di sintassi possono essere importati solo in Modalità esperto. In Modalità normale verrà visualizzato un messaggio di errore.

Per esportare le azioni come file di testo:

- 1 Dal menu a comparsa accessibile nell'angolo superiore destro del pannello Azione scegliere Esporta come file.
- 2 Selezionare una posizione in cui salvare il file, quindi fare clic su Salva.

Per stampare le azioni:

- 1 Dal menu a comparsa accessibile nell'angolo superiore destro del pannello Azioni scegliere Stampa.

Verrà visualizzata la finestra di dialogo Stampa.

- 2 Scegliere le opzioni e fare clic su OK.

Nota: il file stampato non comprenderà le informazioni sul file Flash di origine. Si consiglia di includere queste informazioni in un'azione comment nello script.

Per cercare del testo in uno script, scegliere un'opzione dal menu a comparsa del pannello Azioni:

- Scegliere Vai alla riga per spostarsi su una riga specifica dello script.
- Scegliere Trova per cercare del testo.
- Scegliere Trova di nuovo per cercare l'occorrenza successiva del testo.

- Scegliere Sostituisci per trovare il testo e sostituirlo.

In Modalità esperto il comando Sostituisci consente di effettuare la ricerca nell'intero testo di uno script. In Modalità normale il comando Sostituisci consente di trovare e sostituire testo solo nel campo dei parametri di ogni azione. Ad esempio, in Modalità normale non è possibile sostituire tutte le azioni `gotoAndPlay` con `gotoAndStop`.

Nota: usare il comando Trova o Sostituisci per effettuare una ricerca nella lista delle azioni corrente. Per eseguire una ricerca nel testo di tutti gli script di un filmato, usare Esplora filmato. Per ulteriori informazioni, consultare *Guida all'uso di Flash*.

Evidenziazione e controllo della sintassi

La funzione di evidenziazione della sintassi identifica certi elementi di ActionScript con colori specifici. Ciò consente di evitare errori di sintassi, quale l'uso non corretto di lettere maiuscole o minuscole nelle parole chiave. Ad esempio, se la parola chiave `typeof` viene scritta `typeOf`, non sarà colorata di blu indicando la presenza di un errore. Quando la funzione di evidenziazione della sintassi è attivata, il testo viene evidenziato in base alle seguenti regole.

- Le parole chiave e gli identificatori predefiniti (ad esempio `gotoAndStop`, `play` e `stop`) sono evidenziati in blu.
- Le proprietà sono evidenziate in verde.
- I commenti sono evidenziati in magenta.
- Le stringhe racchiuse tra virgolette sono evidenziate in grigio.

Per attivare o disattivare la funzione di evidenziazione della sintassi:

Scegliere Sintassi colorata dal menu a comparsa accessibile nell'angolo superiore destro del pannello Azioni. Un segno di spunta indica che l'opzione è attivata. Tutti gli script del filmato saranno evidenziati.

Si consiglia di verificare la correttezza sintattica di uno script prima di esportare un filmato. Gli errori vengono riportati nella finestra Output. È possibile esportare un filmato che contiene script non corretti. Tuttavia un messaggio di avvertenza indicherà che gli script contenenti errori non sono stati esportati.

Per verificare la correttezza sintattica dello script corrente:

Scegliere Controlla sintassi dal menu a comparsa accessibile nell'angolo superiore destro del pannello Azioni.

Informazioni sull'evidenziazione degli errori

In Modalità normale gli errori di sintassi sono visualizzati su sfondo rosso nella finestra dello script in modo che siano facilmente identificabili. Se si sposta il puntatore del mouse su un'azione la cui sintassi non è corretta, viene visualizzata una descrizione contenente il messaggio di errore associato all'azione. Quando si seleziona l'azione, il messaggio di errore viene visualizzato anche nel titolo del riquadro dell'area dei parametri.

In Modalità normale tutte le incompatibilità di esportazione di ActionScript sono visualizzate su sfondo giallo nella finestra dello script. Ad esempio, se la versione di esportazione di Flash Player è impostata su Flash 4, il codice ActionScript che è supportato solo da Flash 5 Player viene evidenziato in giallo. La versione di esportazione è impostata nella finestra di dialogo Impostazioni pubblicazione.

Tutte le azioni obsolete sono visualizzate su sfondo verde nella lista nel riquadro a sinistra. Le azioni obsolete vengono evidenziate solo quando la versione di esportazione di Flash è impostata su Flash 5.

Per impostare la versione di esportazione di Flash Player:

- 1 Scegliere File > Impostazioni pubblicazione.
- 2 Fare clic sulla scheda Flash.
- 3 Scegliere una versione di esportazione dal menu a comparsa Versione.

Nota: non è possibile disattivare la funzione di evidenziazione degli errori di sintassi.

Per mostrare l'evidenziazione della sintassi obsoleta:

Scegliere Mostra sintassi obsoleta dal menu a comparsa del pannello Azioni.

Per la lista completa di tutti i messaggi di errore, consultare l'appendice C “Messaggi di errore”

Assegnazione di azioni a oggetti

È possibile assegnare un'azione a un pulsante o a un clip filmato in modo che l'azione venga eseguita quando l'utente seleziona il pulsante o vi posiziona sopra il puntatore oppure quando il clip filmato viene caricato o raggiunge un determinato fotogramma. Quando si assegna l'azione a un'istanza del pulsante o del clip filmato, le altre istanze del simbolo rimangono inalterate. Per assegnare un'azione a un fotogramma, vedere “Assegnazione di azioni a fotogrammi” a pagina 47.

Quando si assegna un'azione a un pulsante, è necessario annidare l'azione in un gestore `on(mouse event)` e specificare gli eventi del mouse o della tastiera che innescano l'azione. Quando si assegna un'azione a un pulsante in Modalità normale, il gestore `on(mouse event)` viene automaticamente inserito.

Quando si assegna un'azione a un clip filmato, è necessario annidare l'azione in un gestore `onClipEvent` e specificare l'evento del clip che innesci l'azione. Quando si assegna un'azione a un clip filmato in Modalità normale, il gestore `on(mouse event)` viene automaticamente inserito.

Di seguito è descritto come assegnare azioni a oggetti tramite il pannello Azioni in Modalità normale.

Dopo aver assegnato un'azione, scegliere Controlli > Prova filmato per verificarne il funzionamento. La maggior parte delle azioni non funziona in modalità di modifica.

Per assegnare un'azione a un pulsante o a un clip filmato:

- 1 Selezionare il pulsante o l'istanza del clip filmato, quindi scegliere Finestra > Azioni.

Se l'elemento selezionato non è un pulsante, un'istanza di un clip filmato o un fotogramma oppure se sono stati selezionati più oggetti, il pannello Azioni risulterà disattivato.

- 2 Scegliere Modalità normale dal menu a comparsa accessibile nell'angolo superiore destro del pannello Azioni oggetto.

- 3 Per assegnare un'azione, eseguire una delle operazioni descritte.
- Fare clic sulla cartella Azioni nella lista nel riquadro a sinistra del pannello Azioni. Fare doppio clic sull'azione desiderata per aggiungerla alla lista delle azioni sul lato destro del pannello.
 - Trascinare un'azione dalla lista nel riquadro a sinistra nella lista nel riquadro a destra.
 - Fare clic sul pulsante di aggiunta (+) e scegliere l'azione desiderata dal menu a comparsa.
 - Usare i tasti di scelta rapida riportati accanto a ogni azione nel menu a comparsa.



Selezione di un oggetto dalla lista nel riquadro a sinistra in Modalità normale

- 4 Nei campi dei parametri nella parte inferiore del pannello selezionare i parametri richiesti dall'azione.

I parametri variano a seconda dell'azione selezionata. Per ulteriori informazioni sui parametri richiesti da ciascuna azione, consultare il capitolo 7 “Dizionario di ActionScript”. Per inserire un percorso target per un clip filmato nel campo di parametri, fare clic sul pulsante a forma di mirino nell'angolo inferiore destro del pannello Azioni. Per ulteriori informazioni, consultare il capitolo 4 “Uso dei clip filmato”.

- 5 Ripetere i punti 3 e 4 per assegnare altre azioni in base alle necessità.

Per eseguire la prova di un'azione oggetto:

Scegliere Controlli > Prova filmato.

Assegnazione di azioni a fotogrammi

Affinché un filmato esegua un'azione quando raggiunge un fotogramma chiave, assegnare un'azione fotogramma al fotogramma chiave. Ad esempio, per creare una ripetizione ciclica nella linea temporale tra i fotogrammi 20 e 10, aggiungere l'azione fotogramma seguente al fotogramma 20:

```
gotoAndPlay (10);
```

Si consiglia di posizionare le azioni fotogramma in un livello separato. I fotogrammi con azioni sono contrassegnati da una piccola *a* nella linea temporale.



Una “a” in un fotogramma chiave indica un'azione fotogramma.

Dopo aver assegnato un'azione, scegliere Controlli > Prova filmato per verificarne il funzionamento. La maggior parte delle azioni non funziona in modalità di modifica.

Di seguito è descritto come assegnare azioni fotogramma tramite il pannello Azioni in Modalità normale. Per informazioni sull'assegnazione di un'azione a un pulsante o a un clip filmato, vedere “Assegnazione di un'azione o di un metodo” a pagina 125.

Per assegnare un'azione a un fotogramma chiave:

- 1 Selezionare un fotogramma chiave nella linea temporale, quindi scegliere Finestra > Azioni.

Se il fotogramma selezionato non è un fotogramma chiave, l'azione verrà assegnata al fotogramma chiave precedente. Se l'elemento selezionato non è un fotogramma o se sono stati selezionati più fotogrammi chiave, il pannello Azioni risulterà disattivato.

- 2 Scegliere Modalità normale dal menu a comparsa accessibile nell'angolo superiore destro del pannello Azioni fotogramma.
- 3 Per assegnare un'azione, eseguire una delle operazioni descritte.
 - Fare clic sulla cartella Azioni nella lista nel riquadro a sinistra del pannello Azioni. Fare doppio clic sull'azione desiderata per aggiungerla alla lista delle azioni sul lato destro del pannello.
 - Trascinare un'azione dalla lista nel riquadro a sinistra nella lista nel riquadro a destra.
 - Fare clic sul pulsante di aggiunta (+) e scegliere l'azione desiderata dal menu a comparsa.
 - Usare i tasti di scelta rapida riportati accanto a ogni azione nel menu a comparsa.
 - Nei campi dei parametri nella parte inferiore del pannello selezionare i parametri richiesti dall'azione.
- 4 Per assegnare azioni aggiuntive, selezionare un altro fotogramma chiave e ripetere il punto 3.

Per eseguire la prova di un'azione fotogramma:

Scegliere Controlli > Prova filmato.

CAPITOLO 2

Creazione di script con ActionScript

Per la creazione di script con ActionScript è possibile scegliere il livello di dettaglio da usare. Per usare azioni semplici, è possibile creare gli script scegliendo opzioni dai menu e dalle liste del pannello Azioni in Modalità normale. Se tuttavia si desidera usare il linguaggio ActionScript per la creazione di script più articolati, è necessario acquisire familiarità con il funzionamento del linguaggio.

Come altri linguaggi di creazione di script, ActionScript è costituito da componenti quali oggetti e funzioni predefinite e consente di creare oggetti e funzioni personalizzate. ActionScript dispone di regole proprie per la sintassi, definisce parole chiave riservate, fornisce diversi operatori e consente l'uso di variabili per la memorizzazione e il recupero di informazioni.

La sintassi e lo stile del linguaggio ActionScript sono molto simili a quelli di JavaScript. Flash 5 converte automaticamente gli script ActionScript creati con versioni precedenti di Flash.

Uso della sintassi di ActionScript

ActionScript dispone di regole di grammatica e punteggiatura che determinano i caratteri e le parole significativi e l'ordine in cui è possibile scrivere tali caratteri e parole. Ad esempio, in italiano il punto viene usato per terminare una frase. In ActionScript un'istruzione viene terminata da un punto e virgola.

Le seguenti regole generali sono valide per tutti gli elementi ActionScript. Per la maggior parte dei termini di ActionScript esistono requisiti individuali. Per le regole relative a un termine specifico, consultare la voce corrispondente nel capitolo 7 “Dizionario di ActionScript”.

Sintassi del punto

In ActionScript un punto (.) indica le proprietà o i metodi associati a un oggetto o a un clip filmato. Il punto consente inoltre di identificare il percorso target di un clip filmato o di una variabile. Un'espressione con questo tipo di sintassi inizia con il nome dell'oggetto o del clip filmato seguito da un punto e termina con la proprietà, il metodo o la variabile da specificare.

Ad esempio, la proprietà `_x` del clip filmato indica la posizione sull'asse *x* del clip filmato nello stage. L'espressione `ballMC._x` indica la proprietà `_x` dell'istanza di clip filmato `ballMC`.

Come altro esempio supponiamo che `submit` sia una variabile impostata nel clip filmato `form` annidato nel clip filmato `shoppingCart`. L'espressione `shoppingCart.form.submit = true` imposta la variabile `submit` dell'istanza `form` su `true`.

Lo stesso meccanismo consente di fare riferimento a un metodo di un oggetto o di un clip filmato. Ad esempio, il metodo `play` dell'istanza `ballMC` sposta l'indicatore di riproduzione nella linea temporale di `ballMC`, come nell'istruzione seguente:

```
ballMC.play();
```

La sintassi del punto usa inoltre due alias speciali, `_root` e `_parent`. L'alias `_root` fa riferimento alla linea temporale principale. È possibile usare l'alias `_root` per creare un percorso target assoluto. Ad esempio, l'istruzione seguente chiama la funzione `buildGameBoard` nel clip filmato `functions` della linea temporale principale:

```
_root.functions.buildGameBoard();
```

È possibile usare l'alias `_parent` per indicare il clip filmato in cui è annidato il clip filmato corrente. È inoltre possibile usare `_parent` per creare un percorso target relativo. Se ad esempio il clip filmato `dog` è annidato nel clip filmato `animal`, la seguente istruzione eseguita per l'istanza `dog` richiede l'arresto di `animal`:

```
_parent.stop();
```

Consultare il capitolo 4 “Uso dei clip filmato”.

Sintassi della barra inclinata

In Flash 3 e 4 la barra inclinata indicava il percorso target di un clip filmato o di una variabile. Questa sintassi è supportata da Flash 5 Player, ma ne è sconsigliato l'uso. Tale sintassi prevede l'uso della barra inclinata al posto del punto per indicare il percorso di un clip filmato o di una variabile. Per indicare una variabile è necessario anteporre due punti alla variabile:

```
myMovieClip/childMovieClip:myVariable
```

È possibile scrivere lo stesso percorso target tramite l'uso del punto, come nell'esempio seguente:

```
myMovieClip.childMovieClip.myVariable
```

La sintassi della barra inclinata in genere veniva usata con l'azione `tellTarget`, il cui uso è ora sconsigliato.

Nota: L'azione `with` è preferibile all'azione `tellTarget` data la sua maggior compatibilità con la sintassi del punto. Per ulteriori informazioni, consultare le singole voci corrispondenti nel capitolo 7 "Dizionario di ActionScript".

Parentesi graffe

Le istruzioni ActionScript sono raggruppate in blocchi tramite parentesi graffe (`{ }`), come nello script seguente:

```
on(release) {  
    myDate = new Date();  
    currentMonth = myDate.getMonth();  
}
```

Consultare "Uso delle azioni" a pagina 69.

Punti e virgola

Un'istruzione ActionScript termina con un punto e virgola. Se si omette il punto e virgola finale, lo script viene comunque compilato correttamente. Ad esempio, le seguenti istruzioni terminano con punti e virgola:

```
column = passedDate.getDay();  
row    = 0;
```

È possibile scrivere le stesse istruzioni senza i punti e virgola finali:

```
column = passedDate.getDay()  
row    = 0
```

Parentesi

Per definire una funzione, racchiudere gli eventuali argomenti tra parentesi tonde:

```
function myFunction (name, age, reader){  
    ...  
}
```

Per chiamare una funzione racchiudere gli argomenti passati alla funzione tra parentesi tonde, come nell'esempio seguente:

```
myFunction ("Steve", 10, true);
```

È inoltre possibile usare le parentesi tonde per ridefinire l'ordine di precedenza di ActionScript o per semplificare la leggibilità delle istruzioni ActionScript. Consultare "Precedenza degli operatori" a pagina 63.

È inoltre possibile usare le parentesi tonde per valutare un'espressione posta sul lato sinistro di un punto in una sintassi del punto. Ad esempio, nell'istruzione seguente le parentesi determinano la valutazione dell'espressione `new color(this)` in modo che venga creato un nuovo oggetto colore:

```
onClipEvent(enterFrame) {  
    (new Color(this)).setRGB(0xffffffff);  
}
```

Senza l'uso delle parentesi sarebbe necessario aggiungere al codice un'istruzione apposita per valutare l'espressione:

```
onClipEvent(enterFrame) {  
    myColor = new Color(this);  
    myColor.setRGB(0xffffffff);  
}
```

Lettere maiuscole e minuscole

In ActionScript soltanto le parole chiave differenziano tra maiuscole e minuscole. Negli altri casi è possibile usare indifferentemente lettere maiuscole o minuscole. Ad esempio, le seguenti istruzioni sono equivalenti:

```
cat.hilite = true;  
CAT.hilite = true;
```

È tuttavia consigliabile applicare convenzioni fisse di uso delle maiuscole, quali quelle seguite nel presente manuale. Ciò semplifica l'identificazione dei nomi delle funzioni e delle variabili durante la lettura del codice ActionScript.

Se le parole chiave usate non rispettano le maiuscole e le minuscole, lo script presenterà errori. Se Sintassi colorata è attivata nel pannello Azioni, le parole chiave scritte correttamente, rispettando maiuscole e minuscole, appaiono in blu. Per ulteriori informazioni, vedere "Parole chiave" a pagina 53 e "Evidenziazione e controllo della sintassi" a pagina 43.

Commenti

Nel pannello Azioni usare l'istruzione `comment` per aggiungere osservazioni a un'azione fotogramma o pulsante ed evidenziare lo scopo dell'azione. I commenti consentono inoltre di comunicare informazioni ad altri sviluppatori quando si forniscono esempi del lavoro svolto o quando si opera in gruppo.

Quando si sceglie l'azione `comment` vengono inseriti nello script i caratteri `//`. Se si aggiungono commenti durante la creazione, anche la comprensione di uno script semplice può risultare ulteriormente semplificata.

```
on(release) {  
    // Crea un nuovo oggetto Date  
    myDate = new Date();  
    currentMonth = myDate.getMonth();  
    // Converte il numero del mese nel relativo nome  
    monthName = calcMonth(currentMonth);  
    year = myDate.getFullYear();  
    currentDate = myDate.getDat ();  
}
```

I commenti appaiono in rosa nella finestra dello script. I commenti possono avere una lunghezza qualsiasi, non influiscono sulla dimensione del file esportato e non devono seguire le regole della sintassi di ActionScript o delle parole chiave.

Parole chiave

In ActionScript le parole chiave hanno usi specifici nel linguaggio e non sono utilizzabili come nomi di variabili, funzioni o etichette. La tabella seguente elenca tutte le parole chiave di ActionScript:

<code>break</code>	<code>for</code>	<code>new</code>	<code>var</code>
<code>continue</code>	<code>function</code>	<code>return</code>	<code>void</code>
<code>delete</code>	<code>if</code>	<code>this</code>	<code>while</code>
<code>else</code>	<code>in</code>	<code>typeof</code>	<code>with</code>

Per ulteriori informazioni su una parola chiave specifica, vedere la voce corrispondente nel capitolo 7 “Dizionario di ActionScript”.

Costanti

Una costante è una proprietà il cui valore non viene mai modificato. Le costanti sono elencate in lettere maiuscole nella lista nel riquadro a sinistra del pannello Azioni e nel capitolo 7 “Dizionario di ActionScript”.

Ad esempio, le costanti `BACKSPACE`, `ENTER`, `QUOTE`, `RETURN`, `SPACE` e `TAB` sono proprietà dell'oggetto `Key` e fanno riferimento ai tasti della tastiera. Per verificare se l'utente preme il tasto Invio, usare la seguente istruzione:

```
if(keycode() == Key.ENTER) {  
    alert = "Are you ready to play?"  
    controlMC.gotoAndStop(5);  
}
```

Informazioni sui tipi di dati

Un tipo di dati descrive il tipo di informazioni che possono essere contenute in una variabile o in un elemento di ActionScript. Esistono due categorie di tipi di dati: tipi di dati di base e tipi di dati puntatore. I tipi di dati di base (stringa, numerico e booleano) hanno un valore costante, per cui possono contenere il valore effettivo dell'elemento che rappresentano. I tipi di dati puntatore (clip filmato e oggetto) hanno valori variabili, per cui contengono riferimenti al valore effettivo dell'elemento. In determinate situazioni, le variabili contenenti tipi di dati di base funzionano diversamente da quelle contenenti tipi di dati puntatore. Consultare “Uso di variabili in uno script” a pagina 60.

Ogni tipo di dati dispone di regole proprie. I tipi di dati sono elencati di seguito. Sono compresi riferimenti a ulteriori informazioni relative ad alcuni tipi di dati.

String

Una stringa è una sequenza di caratteri quali lettere, numeri e caratteri di punteggiatura. Per inserire stringhe in un'istruzione ActionScript è necessario racchiuderle tra virgolette singole o doppie. Le stringhe vengono gestite come caratteri e non come variabili. Ad esempio, nell'istruzione seguente `"L7"` è una stringa:

```
favoriteBand = "L7";
```

È possibile usare l'operatore di addizione (+) per *concatenare* (unire) due stringhe. In ActionScript gli spazi all'inizio o alla fine della stringa sono considerati parte integrante della stringa. La seguente espressione comprende uno spazio dopo la virgola:

```
greeting = "Welcome, " + firstName;
```

Sebbene in ActionScript la distinzione tra maiuscole e minuscole non sia applicata nei riferimenti a variabili, nomi di istanza ed etichette di fotogrammi, viene applicata ai valori di tipo stringa. Ad esempio, le due istruzioni seguenti inseriscono elementi di testo distinti nelle variabili campo di testo specificate, in quanto **"Hello"** e **"HELLO"** sono stringhe.

```
invoice.display = "Hello";  
invoice.display = "HELLO";
```

Per inserire virgolette nelle stringhe, farle precedere da un carattere barra rovesciata (`\`). Questa operazione è detta assegnazione di una sequenza di escape a un carattere. In ActionScript alcuni caratteri possono essere rappresentati soltanto tramite le relative sequenze di escape. La tabella seguente elenca tutti i caratteri escape di ActionScript:

Sequenza di escape	Carattere
<code>\b</code>	Backspace (ASCII 8)
<code>\f</code>	Avanzamento pagina (ASCII 12)
<code>\n</code>	Avanzamento riga (ASCII 10)
<code>\r</code>	Ritorno a capo (ASCII 13)
<code>\t</code>	Tabulazione (ASCII 9)
<code>\"</code>	Virgolette doppie
<code>\'</code>	Virgolette semplici
<code>\\</code>	Barra rovesciata
<code>\000 - \377</code>	Byte specificato in formato ottale
<code>\x00 - \xFF</code>	Byte specificato in formato esadecimale
<code>\u0000 - \uFFFF</code>	Carattere Unicode a 16 bit specificato in esadecimale

Number

Il tipo di dati numerico corrisponde a un numero in virgola mobile e a doppia precisione. È possibile gestire i valori numerici tramite gli operatori aritmetici di addizione (+), sottrazione (-), moltiplicazione (*), divisione (/), modulo (%), incremento (++) e decremento (--). È inoltre possibile gestire i valori numerici tramite i metodi dell'oggetto predefinito `Math`. Nell'esempio seguente il metodo `sqrt` (radice quadrata) viene usato per restituire la radice quadrata del numero 100:

```
Math.sqrt(100);
```

Consultare “Operatori numerici” a pagina 64.

Boolean

Un valore booleano è un valore che risulta `true` (vero) o `false` (falso). In ActionScript i valori `true` e `false` vengono convertiti in 1 e 0 quando appropriato. I valori booleani vengono usati spesso in congiunzione con gli operatori logici in istruzioni di ActionScript che eseguono confronti per il controllo del flusso di uno script. Ad esempio, nello script seguente il filmato viene riprodotto se la variabile `password` è `true`:

```
onClipEvent(enterFrame) {  
    if ((userName == true) && (password == true)){  
        play();  
    }  
}
```

Consultare “Uso di istruzioni if” a pagina 71 e “Operatori logici” a pagina 65.

Object

Un oggetto è un insieme di proprietà. Ogni proprietà è provvista di un nome e di un valore. Il valore di una proprietà può essere qualsiasi tipo di dati Flash, compreso il tipo di dati oggetto. Di conseguenza è possibile definire oggetti all'interno di altri oggetti, ossia annidarli. Per specificare gli oggetti e le relative proprietà, usare l'operatore punto (`.`). Ad esempio, nel codice seguente `hoursWorked` è una proprietà di `weeklyStats`, che è a sua volta una proprietà di `employee`:

```
employee.weeklyStats.hoursWorked
```

È possibile usare gli oggetti predefiniti di ActionScript per accedere a informazioni specifiche e gestirle. Ad esempio, l'oggetto `Math` dispone di metodi che eseguono operazioni matematiche sui numeri passati all'oggetto stesso. Questo esempio usa il metodo `sqrt`:

```
squareRoot = Math.sqrt(100);
```

L'oggetto ActionScript `MovieClip` dispone di metodi che consentono di controllare istanze del simbolo clip filmato nello stage. Questo esempio usa i metodi `play` e `nextFrame`:

```
mcInstanceName.play();  
mc2InstanceName.nextFrame();
```

È inoltre possibile creare oggetti personalizzati e organizzare le informazioni del filmato. Per aggiungere interattività a un filmato tramite ActionScript è necessario disporre di varie informazioni: ad esempio un nome utente, la velocità di una palla, il nome delle voci in un carrello, il numero di fotogrammi caricati, il codice postale dell'utente o l'ultimo tasto premuto. La creazione di oggetti personalizzati consente di organizzare le informazioni in gruppi, di semplificare gli script e di riusare gli script. Per ulteriori informazioni, consultare “Uso di oggetti personalizzati” a pagina 83.

Clip filmato

I clip filmato sono simboli che consentono la riproduzione di animazioni in un filmato Flash. Il tipo di dati clip filmato è l'unico tipo di dati che fa riferimento a un elemento grafico. Tale tipo di dati consente il controllo dei simboli clip filmato tramite i metodi dell'oggetto MovieClip. È possibile chiamare i metodi tramite l'operatore (.), come nell'esempio seguente:

```
myClip.startDrag(true);  
parentClip.childClip.getURL( "http://www.macromedia.com/support/"  
+ product);
```

Informazioni sulle variabili

Una variabile è un contenitore di informazioni. Il contenitore resta invariato, ma il contenuto può variare. Modificando il valore di una variabile durante la riproduzione di un filmato è possibile registrare e salvare informazioni sulle azioni eseguite dall'utente, registrare valori che cambiano durante la riproduzione o verificare la validità di una condizione.

Quando si definisce una variabile per la prima volta è opportuno assegnarle sempre un valore noto. Questa operazione è detta inizializzazione di una variabile e viene spesso eseguita nel primo fotogramma del filmato. L'inizializzazione di variabili semplifica la gestione e il confronto dei valori della variabile durante la riproduzione del filmato.

Le variabili possono contenere qualsiasi tipo di dati: numerico, stringa, booleano, oggetto o clip filmato. Il tipo di dati contenuto nella variabile determina la modalità di modifica del valore della variabile, quando questa viene assegnata in uno script.

I tipi di informazioni memorizzabili in una variabile includono un URL, un nome utente, il risultato di un'operazione matematica, il numero di volte che si è verificato un evento e lo stato di selezione di un pulsante. A ogni istanza di filmato e clip filmato è associato il proprio insieme di variabili e a ogni variabile è assegnato il proprio valore che è indipendente da quello delle variabili in altri filmati o clip filmato.

Assegnazione di un nome a una variabile

Il nome di una variabile deve soddisfare le seguenti regole:

- Il nome deve essere un identificatore
- Il nome non può essere una parola chiave o un valore booleano (`true` o `false`).
- Il nome deve essere univoco entro la propria area di validità Consultare "Assegnazione di un'area di validità a una variabile" a pagina 58.

Assegnazione di un tipo a una variabile

In Flash non è necessario definire esplicitamente il tipo di valore contenuto da una variabile, sia esso un numero, una stringa o un altro tipo di dati. Il tipo di dati della variabile viene determinato automaticamente al momento dell'assegnazione:

```
x = 3;
```

Nell'espressione `x = 3` Flash valuta l'elemento a destra dell'operatore e lo riconosce come tipo di dati numerico. Una successiva assegnazione può convertire il tipo di dati di `x`. Ad esempio, `x = "hello"` assegna a `x` il tipo di dati stringa. Una variabile alla quale non è stato assegnato un valore ha come tipo di dati `undefined`.

In ActionScript, quando un'espressione lo richiede, i tipi di dati vengono convertiti automaticamente. Ad esempio, quando si passa un valore all'azione `trace`, `trace` converte automaticamente il valore in stringa e lo invia alla finestra Output. Nelle espressioni con operatori, ActionScript converte i tipi di dati a seconda delle necessità. Ad esempio, quando viene usato con una stringa, l'operatore `+` prevede che anche l'altro operando sia una stringa:

```
"Next in line, number " + 7
```

ActionScript converte il numero `7` nella stringa `"7"` e lo concatena alla fine della prima stringa. Il risultato è la stringa seguente:

```
"Next in line, number 7"
```

Durante il debug degli script è spesso utile determinare il tipo di dati di un'espressione o di una variabile, per individuare il motivo di determinati funzionamenti. Tale operazione è eseguibile tramite l'operatore `typeof`, come nell'esempio seguente:

```
trace(typeof(variableName));
```

Per convertire una stringa in un valore numerico usare la funzione `Number`. Per convertire un valore numerico in una stringa usare la funzione `String`. Consultare le singole voci corrispondenti nel capitolo 7 “Dizionario di ActionScript” a pagina 167.

Assegnazione di un'area di validità a una variabile

L'area di validità di una variabile è la porzione di codice in cui la variabile è nota e può essere oggetto di riferimenti. In ActionScript le variabili possono essere di tipo globale o locale. Una variabile globale è condivisa tra tutte le linee temporali. Una variabile locale è disponibile soltanto all'interno del blocco di codice di appartenenza (delimitato dalle parentesi graffe).

L'istruzione `var` consente di dichiarare una variabile locale in uno script. Ad esempio, le variabili `i` e `j` vengono spesso usate come contatori di ciclo. Nell'esempio seguente `i` viene usata come variabile locale ed esiste esclusivamente all'interno della funzione `makeDays`:

```
function makeDays(){
    var i
    for( i = 0; i < monthArray[month]; i++ ) {

        _root.Days.attachMovie( "DayDisplay", i, i + 2000 );

        _root.Days[i].num = i + 1;
        _root.Days[i]._x = column * _root.Days[i]._width;
        _root.Days[i]._y = row * _root.Days[i]._height;

        column = column + 1;

        if (column == 7 ) {

            column = 0;
            row = row + 1;
        }
    }
}
```

Le variabili locali consentono inoltre di evitare conflitti di denominazione, che possono generare errori nel filmato. Se ad esempio si usa come variabile locale `name` è possibile memorizzare nella variabile un nome utente in un contesto e un'istanza di clip filmato in un altro contesto. Poiché tali variabili vengono eseguite in aree di validità diverse non si verificheranno conflitti.

È sempre consigliabile usare variabili locali nel corpo di una funzione, in modo che la funzione possa essere eseguita come entità di codice indipendente. Una variabile locale è modificabile soltanto all'interno del blocco di codice di appartenenza. Se un'espressione di una funzione usa una variabile globale è possibile che un elemento esterno alla funzione modifichi il valore della variabile, con conseguente modifica della funzione stessa.

Dichiarazione di variabili

Per dichiarare variabili globali, usare l'azione `setVariables` o l'operatore di assegnazione (`=`). Entrambi i metodi producono lo stesso risultato.

Per dichiarare variabili locali, usare l'azione `var` nel corpo di una funzione. L'area di validità delle variabili locali corrispondente al blocco, al di fuori del quale non sono più valide. La validità delle variabili locali non dichiarate all'interno di un blocco termina alla fine dello script di appartenenza.

Nota: Anche l'azione `call` crea una nuova area di validità delle variabili locali per lo script chiamato. All'uscita dallo script chiamato, l'area di validità delle variabili locali termina. Questa operazione è tuttavia sconsigliata, in quanto l'azione `call` è stata sostituita dall'azione `with`, più compatibile con la sintassi del punto.

Per verificare il valore di una variabile usare l'azione `trace` per inviarne il valore alla finestra `Output`. Ad esempio, `trace(hoursWorked)` invia il valore della variabile `hoursWorked` alla finestra `Output` in modalità di prova filmato. In modalità di prova filmato è inoltre possibile verificare e impostare i valori delle variabili nella finestra `Debugger`. Per ulteriori informazioni, consultare il capitolo 6 “Risoluzione dei problemi di ActionScript”.

Uso di variabili in uno script

Prima di usare una variabile in un'espressione è necessario dichiararla nello script. Se si usa una variabile non dichiarata, come nell'esempio seguente, il valore della variabile sarà `undefined` e lo script genererà un errore:

```
getURL(myWebSite);  
myWebSite = "http://www.shrimpmeat.net";
```

L'istruzione che dichiara la variabile `myWebSite` deve precederne l'uso, in modo che la variabile dell'azione `getURL` sia sostituibile da un valore.

È possibile modificare più volte il valore di una variabile in uno script. Il tipo di dati contenuto in una variabile determina come e quando essa viene modificata. I tipi di dati di base, quali stringhe e numeri, vengono passati come valore. Ciò significa che il contenuto effettivo della variabile viene passato alla variabile.

Nell'esempio seguente `x` è impostato su 15 e tale valore viene copiato in `y`. Quando `x` diventa 30 il valore di `y` resta 15, in quanto `y` non ricerca il proprio valore in `x` ma contiene il valore di `x` copiato.

```
var x = 15;  
var y = x;  
var x = 30;
```

In quest'altro esempio, la variabile `in` contiene il valore di base 9. Il valore effettivo viene passato alla funzione `sqr` che restituisce il valore 3:

```
function sqr(x){  
    return x * x;  
}  
  
var in = 9;  
var out = sqr(in);
```

Il valore della variabile `in` non cambia.

Poiché il tipo di dati oggetto può contenere una quantità di dati talmente estesa e complessa, una variabile con tale tipo di dati non contiene il valore effettivo, ma soltanto un riferimento al valore. Questo riferimento equivale a un alias che punta al contenuto della variabile. Quando la variabile richiede il proprio valore, il riferimento recupera il contenuto e restituisce la risposta senza trasferire il valore alla variabile.

Il seguente è un esempio di passaggio per riferimento:

```
var myArray = ["tom", "dick"];
var newArray = myArray;
myArray[1] = "jack";
trace(newArray);
```

Il codice precedente crea un oggetto Array di nome `myArray` che contiene due elementi. La variabile `newArray` viene creata e le viene assegnato un riferimento a `myArray`. Quando il secondo elemento di `myArray` viene modificato, ha effetto su tutte le variabili che contengono un riferimento a tale elemento. L'azione `trace` invia quindi `["tom", "jack"]` alla finestra Output.

Nell'esempio seguente `myArray` contiene un oggetto Array che viene passato alla funzione `zeroArray` come riferimento. La funzione `zeroArray` converte il contenuto della matrice in `myArray`.

```
function zeroArray (array){
    var i;
    for (i=0; i < array.length; i++) {
        array[i] = 0;
    }
}

var myArray = new Array();
myArray[0] = 1;
myArray[1] = 2;
myArray[2] = 3;

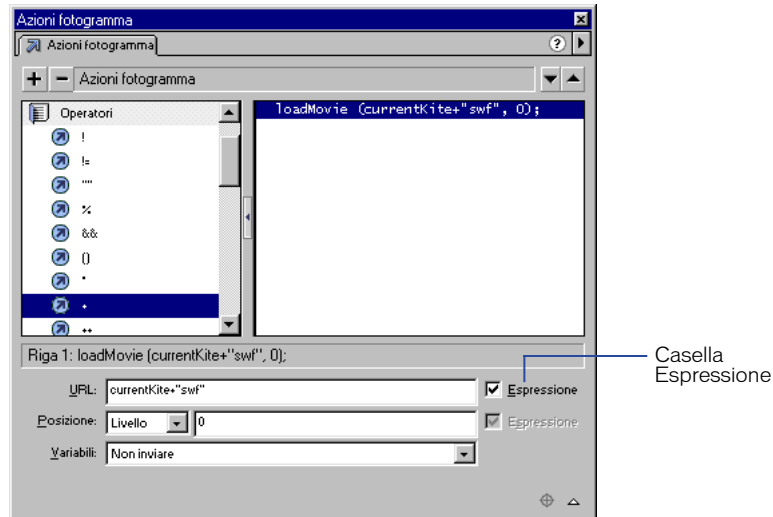
var out = zeroArray(myArray)
```

La funzione `zeroArray` accetta come argomento un oggetto Array e imposta su 0 tutti gli elementi di tale matrice. La funzione può modificare la matrice in quanto viene passata come riferimento.

I riferimenti a tutti gli oggetti diversi dai clip filmato sono detti *riferimenti fissi* in quanto se un oggetto è destinatario di un riferimento non può essere eliminato. Un riferimento a un clip filmato è un tipo di riferimento speciale detto *riferimento removibile*. I riferimenti removibili non richiedono l'esistenza dell'oggetto destinatario del riferimento. Se un clip filmato viene eliminato con un'azione quale `removeMovieClip`, i riferimenti a tale clip non saranno più operativi.

Uso di operatori per la gestione dei valori nelle espressioni

Un'espressione è qualsiasi istruzione valutabile da Flash che restituisce un valore. È possibile creare un'espressione combinando operatori e valori oppure chiamando una funzione. Quando si scrive un'espressione nel pannello Azioni in Modalità normale, verificare che la casella Espressioni nel riquadro dei parametri sia selezionata. In caso contrario il campo conterrà il valore letterale di una stringa.



Gli operatori sono caratteri che specificano la modalità di combinazione, confronto o modifica dei valori di un'espressione. Gli elementi sui quali ha effetto l'operatore sono detti *operandi*. Ad esempio, nell'istruzione seguente l'operatore + somma un valore numerico al valore della variabile `foo`; `foo` e `3` sono gli operandi:

`foo + 3`

La presente sezione descrive le regole generali relative agli operatori comuni. Per ulteriori informazioni su ogni operatore citato di seguito e su operatori speciali non appartenenti alle categorie descritte, consultare il capitolo 7 “Dizionario di ActionScript”.

Precedenza degli operatori

Quando nella stessa istruzione vengono usati due o più operatori, alcuni operatori hanno la precedenza su altri. In ActionScript una gerarchia precisa specifica la precedenza degli operatori da eseguire. Ad esempio, la moltiplicazione viene sempre eseguita prima dell'addizione; tuttavia gli elementi tra parentesi hanno la precedenza sulla moltiplicazione. In assenza di parentesi, nell'esempio seguente la prima operazione eseguita è la moltiplicazione:

```
total = 2 + 4 * 3;
```

Il risultato è 14.

Se l'operazione di addizione è racchiusa tra parentesi, viene eseguita prima della moltiplicazione:

```
total = (2 + 4) * 3;
```

Il risultato è 18.

Per una tabella contenente tutti gli operatori e l'ordine di precedenza, consultare l'appendice B “Precedenza e associatività degli operatori”.

Associatività degli operatori

Quando due o più operatori hanno lo stesso livello di precedenza, la loro associatività determina l'ordine di esecuzione. L'associatività può essere da sinistra a destra o da destra a sinistra. Ad esempio l'operatore di moltiplicazione ha un'associatività da sinistra a destra. Di conseguenza le due istruzioni che seguono sono equivalenti:

```
total = 2 * 3 * 4;  
total = (2 * 3) * 4;
```

Per una tabella contenente tutti gli operatori e l'ordine di associatività, consultare l'appendice B “Precedenza e associatività degli operatori”.

Operatori numerici

Gli operatori numerici consentono di eseguire operazioni di somma, sottrazione, moltiplicazione, divisione e altre operazioni aritmetiche. Le parentesi e il segno meno sono operatori aritmetici. La tabella seguente elenca gli operatori numerici di ActionScript.

Operatore	Operazione eseguita
+	Addizione
*	Moltiplicazione
/	Divisione
%	Modulo
-	Sottrazione
++	Incremento
--	Decremento

Operatori di confronto

Gli operatori di confronto confrontano i valori delle espressioni e restituiscono un valore booleano (`true` o `false`). Tali operatori vengono usati soprattutto nei cicli e nelle istruzioni condizionali. Nell'esempio seguente se la variabile `score` è 100 viene caricato un determinato filmato; in caso contrario viene caricato un altro filmato:

```
if (score == 100){  
    loadMovie("winner.swf", 5);  
} else {  
    loadMovie("loser.swf", 5);  
}
```

La tabella seguente elenca gli operatori di confronto di ActionScript.

Operatore	Operazione eseguita
<	Minore di
>	Maggiore di
<=	Minore o uguale a
>=	Maggiore o uguale a

Operatori stringa

Quando viene applicato a stringhe, l'operatore `+` ha un effetto particolare, in quanto determina il concatenamento tra i due operandi stringa. Ad esempio, la seguente istruzione aggiunge:

```
"Congratulations," a "Donna!":  
"Congratulations, " + "Donna!"
```

Il risultato è `"Congratulations, Donna!"`. Se soltanto uno degli operandi dell'operatore `+` è una stringa, l'altro operando viene convertito automaticamente in stringa.

Anche gli operatori di confronto `>`, `>=`, `<` e `<=` funzionano in modo particolare se applicati a stringhe. Questi operatori confrontano due stringhe per determinare qual è la prima in ordine alfabetico. Gli operatori di confronto confrontano le stringhe soltanto se entrambi gli operandi sono stringhe. Se soltanto un operando è una stringa, ActionScript converte entrambi gli operandi in numeri ed esegue un confronto numerico.

Nota: Le funzionalità dei tipi di dati di ActionScript in Flash 5 consentono l'uso degli stessi operatori su tipi di dati diversi. Non è più necessario usare gli operatori stringa di Flash 4 (ad esempio `eq`, `ge` e `lt`), salvo se si desidera esportare il filmato come Flash 4.

Operatori logici

Gli operatori logici confrontano valori booleani (`true` e `false`) e restituiscono un terzo valore booleano. Se ad esempio entrambi gli operandi restituiscono `true`, l'operatore logico AND (`&&`) restituisce `true`. Se uno o entrambi gli operandi restituiscono `true`, l'operatore logico OR (`||`) restituisce `false`. Gli operatori logici vengono spesso usati insieme agli operatori di confronto per determinare la condizione di un'azione `if`. Ad esempio, se nello script seguente entrambe le espressioni sono `true`, verrà eseguita l'azione `if`:

```
if ((i > 10) && (_framesloaded > 50)){  
    play()  
}
```

La tabella seguente elenca gli operatori logici di ActionScript.

Operatore	Operazione eseguita
<code>&&</code>	AND logico
<code> </code>	OR logico
<code>!</code>	NOT logico

Operatori bit a bit

Gli operatori bit a bit gestiscono internamente i numeri in virgola mobile convertendoli in interi a 32 bit, più semplici da usare. L'operazione bit a bit eseguita dipende dall'operatore, ma tutte le operazioni bit a bit valutano separatamente ogni cifra di un numero in virgola mobile per calcolare un nuovo valore.

La tabella seguente elenca gli operatori bit a bit di ActionScript.

Operatore	Operazione eseguita
&	AND bit a bit
	OR bit a bit
^	XOR bit a bit
-	NOT bit a bit
<<	Spostamento a sinistra
>>	Spostamento a destra
>>>	Spostamento a destra con inserimento zero

Operatori di uguaglianza e assegnazione

È possibile usare l'operatore di uguaglianza (==) per determinare se i valori o le identità di due operandi sono uguali. Il confronto restituisce un valore booleano (true o false). Se gli operandi sono stringhe, valori numerici o valori booleani, vengono confrontati in base al valore. Se gli operandi sono oggetti o matrici, vengono confrontati in base al riferimento.

L'operatore di assegnazione (=) consente di assegnare un valore a una variabile, come nell'esempio seguente:

```
password = "Sk8tEr";
```

È inoltre possibile usare questo operatore per assegnare più variabili nella stessa espressione. Nell'istruzione seguente il valore di b viene assegnato alle variabili c e d:

```
a = b = c = d;
```

Inoltre gli operatori di assegnazione composti consentono di combinare più operazioni. Gli operatori composti vengono eseguiti su entrambi gli operandi e assegnano il nuovo valore al primo operando. Ad esempio, le seguenti due istruzioni sono equivalenti:

```
x += 15;  
x = x + 15;
```

La tabella seguente elenca gli operatori di uguaglianza e assegnazione di ActionScript.

Operatore	Operazione eseguita
==	Uguaglianza
!=	Disuguaglianza
=	Assegnazione
+=	Addizione e assegnazione
-=	Sottrazione e assegnazione
*=	Moltiplicazione e assegnazione
%=	Modulo e assegnazione
/=	Divisione e assegnazione
<<=	Spostamento a sinistra bit a bit e assegnazione
>>=	Spostamento a destra bit a bit e assegnazione
>>>=	Spostamento a destra con inserimento zero e assegnazione
^=	XOR bit a bit e assegnazione
=	OR bit a bit e assegnazione
&=	AND bit a bit e assegnazione

Operatori punto e accesso matrice

È possibile usare l'operatore punto (.) e l'operatore accesso matrice ([]) per accedere a qualsiasi proprietà oggetto predefinita o personalizzata di ActionScript, comprese le proprietà di un clip filmato.

L'operatore punto richiede a sinistra il nome di un oggetto e a destra il nome di una proprietà o di una variabile. Il nome della proprietà o della variabile non può essere una stringa o una variabile che restituisce una stringa, ma deve essere un identificatore. Gli esempi seguenti illustrano l'uso dell'operatore punto:

```
year.month = "June";  
year.month.day = 9;
```

L'operatore punto e l'operatore accesso matrice hanno la stessa funzione, ma l'operatore punto richiede come proprietà un identificatore, mentre l'operatore accesso matrice valuta il contenuto in un nome, quindi accede il valore della proprietà con tale nome. Ad esempio, le due linee di codice seguenti accedono alla stessa variabile `velocity` nel clip filmato `rocket`:

```
rocket.velocity;  
rocket["velocity"];
```

L'operatore accesso matrice consente di impostare e recuperare dinamicamente nomi di istanza e variabili. Ad esempio, nel codice seguente l'espressione contenuta nell'operatore `[]` viene valutata, quindi il risultato viene usato come nome della variabile del clip filmato `name` da accedere:

```
name["mc" + i ]
```

Se si ha familiarità con la sintassi di ActionScript di Flash 4, lo stesso risultato è ottenibile tramite la funzione `eval`, come nell'esempio seguente:

```
eval("mc" & i);
```

L'operatore accesso matrice può essere usato anche sul lato sinistro di un'istruzione di assegnazione, consentendo l'impostazione dinamica di nomi di istanza, variabile e oggetto, come nell'esempio seguente:

```
name[index] = "Gary";
```

Questa operazione è equivalente alla seguente sintassi di ActionScript di Flash 4:

```
Set Variable: "name:" & index = "Gary"
```

L'operatore accesso matrice può essere annidato in se stesso, per simulare matrici a più dimensioni.

```
chessboard[row][column]
```

Ciò equivale alla seguente sintassi:

```
eval("chessboard/" & row & ":" & column)
```

Nota: Per creare codice ActionScript compatibile con Flash 4 Player è possibile usare l'azione `eval` con l'operatore `add`.

Uso delle azioni

Le azioni sono le istruzioni (o comandi) di ActionScript. Più azioni assegnate allo stesso fotogramma o oggetto creano uno script. Le azioni possono essere indipendenti, come nelle istruzioni seguenti:

```
swapDepths("mc1", "mc2");
gotoAndPlay(15);
```

Oppure è possibile annidare le azioni, inserendone una all'interno di un'altra, in modo che si influenzino tra di loro. Nell'esempio seguente l'azione `if` determina quando eseguire l'azione `gotoAndPlay`:

```
if (i >= 25) {
    gotoAndPlay(10);
}
```

Le azioni possono spostare l'indicatore di riproduzione nella linea temporale (`gotoAndPlay`), controllare il flusso di uno script creando cicli (`do while`) o elementi di logica condizionale (`if`) oppure creare nuove funzioni e variabili (`function` e `setVariable`). La tabella seguente elenca tutte le azioni di ActionScript.

Azioni				
break	evaluate	include	print	stopDrag
call	for	loadMovie	printAsBitmap	swapDepths
commento	for...in	loadVariables	removeMovieClip	tellTarget
continue	fsCommand	nextFrame nextScene	return	toggleHighQuality
delete	function	on	setVariable	stopDrag
do...while	getURL	onClipEvent	setProperty	trace
duplicateMovieClip	gotoAndPlay gotoAndStop	play	startDrag	unloadMovie
else	if	prevFrame	stop	var
else if	ifFrameLoaded	prevScene	stopAllSounds	while

Per la sintassi ed esempi d'uso di ogni azione, consultare le singole voci nel capitolo 7 “Dizionario di ActionScript”.

Nota: Nel presente manuale il termine ActionScript *action* (azione) è sinonimo del termine JavaScript *statement* (istruzione).

Creazione di un percorso target

Per usare un'azione per controllare un clip filmato o un filmato caricato, è necessario specificarne il nome e l'indirizzo, ovvero il *percorso target*. Le seguenti azioni richiedono come argomenti uno o più percorsi target:

- loadMovie
- loadVariables
- unloadMovie
- setProperty
- startDrag
- duplicateMovieClip
- removeMovieClip
- print
- printAsBitmap
- tellTarget

Ad esempio, l'azione loadMovie richiede gli argomenti *URL*, *Location* e *Variables*. *URL* è l'indirizzo Web del filmato da caricare. *Location* è il percorso target in cui verrà caricato il filmato.

```
loadMovie(URL, Location, Variables);
```

Nota: L'argomento *Variables* non è necessario in questo esempio.

L'istruzione seguente carica l'URL `http://www.mySite.com/myMovie.swf` nell'istanza `bar` della linea temporale principale `_root`; `_root.bar` è il percorso target:

```
loadMovie("http://www.mySite.com/myMovie.swf", _root.bar);
```

In ActionScript è possibile identificare un clip filmato in base al nome dell'istanza. Ad esempio, nell'istruzione seguente la proprietà `_alpha` del clip filmato `star` è impostata su una visibilità del 50%:

```
star._alpha = 50;
```

Per assegnare un nome di istanza a un clip filmato:

- 1 Selezionare il clip filmato sullo stage.
- 2 Scegliere Finestra > Pannelli > Istanza.
- 3 Specificare un nome di istanza nel campo Nome.

Per identificare un filmato caricato:

Usare `_levelX` dove *X* è il numero di livello specificato nell'azione loadMovie che ha caricato il filmato.

Ad esempio, un filmato caricato nel livello 5 ha il nome di istanza `_level5`. Nell'esempio seguente un filmato viene caricato nel livello 5 e la relativa visibilità viene impostata su `false`:

```
onClipEvent(load) {  
    loadMovie("myMovie.swf", 5);  
}  
onClipEvent(enterFrame) {  
    _level5._visible = false;  
}
```

Per specificare il percorso target di un filmato:

Fare clic sul pulsante a forma di mirino nel pannello Azioni e selezionare un clip filmato nella lista visualizzata.

Per ulteriori informazioni sulla creazione di percorsi target, consultare il capitolo 4 “Uso dei clip filmato”.

Controllo del flusso negli script

ActionScript usa le azioni `if`, `for`, `while`, `do...while` e `for...in` per eseguire un'azione in base all'esistenza di una condizione.

Uso di istruzioni if

Le istruzioni che verificano se una condizione è vera o falsa iniziano con il termine `if`. Se la condizione è soddisfatta, ActionScript esegue la seguente istruzione. Se la condizione non è soddisfatta, ActionScript passa all'istruzione successiva al di fuori del blocco di codice.

Per ottimizzare le prestazioni del codice è opportuno verificare prima le condizioni più probabili.

Le seguenti istruzioni verificano varie condizioni. Il termine `else if` specifica verifiche alternative da effettuare se le condizioni precedenti non risultano vere.

```
if ((password == null) || (email == null)){  
    gotoAndStop("reject");  
} else {  
    gotoAndPlay("startMovie");  
}
```

Ripetizione di un'azione

ActionScript può ripetere un'azione per un determinato numero di volte o quando una condizione specifica risulta soddisfatta. Per creare cicli usare le azioni `while`, `do...while`, `for`, e `for...in`.

Per ripetere un'azione finché una condizione viene soddisfatta:

Usare l'istruzione `while`.

Un ciclo `while` valuta l'espressione e, se l'espressione è `true`, esegue il codice nel corpo del ciclo. Dopo l'esecuzione di tutte le istruzioni nel corpo l'espressione viene nuovamente valutata. Nell'esempio seguente il ciclo viene eseguito per quattro volte:

```
i = 4
while (i > 0) {
    myMC.duplicateMovieClip("newMC" + i, i );
    i --;
}
```

È possibile usare l'istruzione `do...while` per creare un tipo di ciclo analogo al ciclo `while`. In un ciclo `do...while` l'espressione viene valutata alla fine del blocco di codice, per cui il ciclo viene sempre eseguito almeno una volta, come nell'esempio seguente:

```
i = 4
do {
    myMC.duplicateMovieClip("newMC" +i, i );
    i --;
} while (i > 0);
```

Per ripetere un'azione tramite un contatore incorporato:

Usare l'istruzione `for`.

Nella maggior parte dei cicli un contatore verifica il numero di ripetizioni dell'esecuzione. È possibile dichiarare una variabile, quindi scrivere un'istruzione che ne incrementa o decrementa il valore ad ogni esecuzione del ciclo. Nell'azione `for` il contatore e l'istruzione che incrementa il contatore fanno parte dell'azione, come nel codice seguente:

```
for (i = 4; i > 0; i--){
    myMC.duplicateMovieClip("newMC" + i, i + 10);
}
```

Per eseguire un ciclo tra gli elementi secondari di un clip filmato o di un oggetto:

Usare l'istruzione `for...in`.

Gli elementi secondari sono altri clip filmato, funzioni, oggetti e variabili. Nell'esempio seguente `trace` viene usata per stampare i risultati nella finestra Output:

```
myObject = { name:'Joe', age:25, city:'San Francisco' };
for (propertyName in myObject) {
    trace("myObject has the property: " + propertyName + ", with the
value: " + myObject[propertyName]);
}
```

Questo esempio produce i seguenti risultati nella finestra Output:

```
myObject has the property: name, with the value: Joe
myObject has the property: age, with the value: 25
myObject has the property: city, with the value: San Francisco
```

Se si richiede che lo script esegua un'iterazione su un determinato tipo di elemento secondario (ad esempio soltanto su clip filmato secondari) usare l'istruzione `for...in` unitamente all'operatore `typeof`.

```
for (name in myMovieClip) {
    if (typeof (myMovieClip[name]) == "movieclip") {
        trace("I have a movie clip child named " + name);
    }
}
```

Nota: L'istruzione `for...in` esegue le iterazioni sulle proprietà degli oggetti della catena di prototipi dell'oggetto base dell'iterazione. Se il prototipo di un oggetto secondario è `parent`, `for...in` esegue iterazioni anche sulle proprietà di `parent`. Consultare "Creazione dell'ereditarietà" a pagina 84.

Per ulteriori informazioni su ciascuna azione, consultare le singole voci nel capitolo 7 "Dizionario di ActionScript".

Uso di funzioni predefinite

Una funzione è un blocco di codice ActionScript riutilizzabile in qualsiasi punto di un filmato. Se a una funzione vengono passati valori specifici, detti argomenti, la funzione eseguirà le operazioni su tali valori. Una funzione può inoltre restituire valori. Flash dispone di funzioni predefinite che consentono l'accesso a determinate informazioni e l'esecuzione di attività particolari, ad esempio rilevare la presenza di collisioni (`hitTest`), determinare il valore dell'ultimo tasto premuto (`keyCode`) e determinare il numero della versione di Flash Player che riproduce il filmato (`getVersion`).

Chiamata di una funzione

È possibile chiamare una funzione in qualsiasi linea temporale da qualsiasi linea temporale, compreso un filmato caricato. Ogni funzione ha caratteristiche proprie e alcune funzioni richiedono il passaggio di determinati valori. Se viene passato un numero di argomenti superiore a quello richiesto dalla funzione, i valori aggiuntivi vengono ignorati. Se non viene passato un argomento obbligatorio, agli argomenti vuoti viene assegnato il tipo di dati `undefined` che può generare errori durante l'esportazione di uno script. È possibile chiamare solo funzioni che si trovano in un fotogramma già raggiunto dall'indicatore di riproduzione.

La tabella seguente elenca le funzioni predefinite di Flash:

Boolean	getTimer	isFinite	newline	scroll
escape	getVersion	isNaN	Number	String
eval	globalToLocal	keyCode	parseFloat	targetPath
false	hitTest	localToGlobal	parseInt	true
getProperty	int	maxscroll	random	unescape

Nota: L'uso delle funzioni stringa è obsoleto e tali funzioni non sono elencate nella tabella precedente.

Per chiamare una funzione in Modalità esperto:

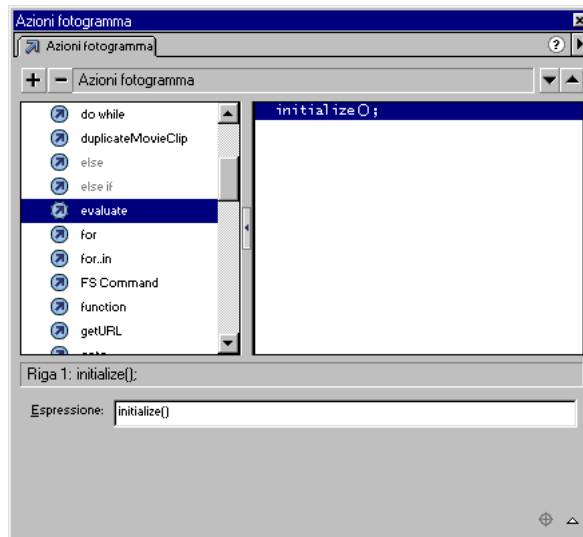
Usare il nome della funzione. Passare gli argomenti richiesti racchiudendoli tra parentesi.

Nell'esempio seguente viene chiamata la funzione `initialize` che non richiede argomenti:

```
initialize();
```

Per chiamare una funzione in Modalità normale:

Usare l'azione `evaluate`. Immettere il nome della funzione e gli argomenti richiesti nel campo `Espressione`.



Uso dell'azione `evaluate` per chiamare una funzione in Modalità normale

Per chiamare una funzione in un'altra linea temporale, usare un percorso target. Ad esempio, per chiamare la funzione `calculateTax` dichiarata nell'istanza `functionsMovieClip` usare il seguente percorso:

```
_root.functionsMovieClip.calculateTax(total);
```

Nota: Passare gli argomenti racchiudendoli tra parentesi.

Per ulteriori informazioni su ciascuna funzione, comprese le funzioni stringa obsolete, consultare le singole voci nel capitolo 7 “Dizionario di ActionScript”.

Creazione di funzioni personalizzate

È possibile definire funzioni per eseguire una serie di istruzioni sui valori passati. Le funzioni possono inoltre restituire valori. Una volta definita, una funzione può essere chiamata da qualsiasi linea temporale, inclusa la linea temporale di un filmato caricato.

Una funzione può essere considerata come una scatola nera: viene chiamata con dei valori di input (argomenti), esegue alcune operazioni, quindi produce un output (restituisce un valore). È consigliabile, quando si scrive una funzione, inserire commenti rilevanti relativi ai valori di input e di output e allo scopo della funzione stessa. In tal modo non è necessario che chi usa la funzione ne conosca nei dettagli il funzionamento.

Definizione di una funzione

Come le variabili, le funzioni sono associate al clip filmato che le definisce. Quando una funzione viene ridefinita, la nuova definizione sostituisce quella esistente.

Per definire una funzione usare l'azione `function` seguita dal nome della funzione, dagli argomenti da passare alla funzione e dalle istruzioni `ActionScript` che indicano le operazioni eseguite dalla funzione.

Nel seguente esempio è definita la funzione di nome `Circle` con l'argomento `radius`:

```
function Circle(radius) {  
    this.radius = radius;  
    this.area = Math.PI * radius * radius;  
}
```

Nota: La parola chiave `this`, usata nel corpo della funzione, è un riferimento al clip filmato a cui la funzione appartiene.

È possibile definire una funzione anche creando un *valore letterale di funzione*, ossia una funzione senza nome, dichiarata in un'espressione anziché in un'istruzione. È possibile usare un valore letterale di funzione per definire una funzione, restituire il valore corrispondente e assegnarlo a una variabile all'interno di un'espressione, come nell'esempio seguente:

```
area = (function () {return Math.PI * radius *radius;})(5);
```

Passaggio di argomenti a una funzione

Gli argomenti sono gli elementi sui quali viene eseguito il codice della funzione. Nel presente manuale i termini *argomento* e *parametro* sono intercambiabili. Ad esempio la seguente funzione riconosce gli argomenti `initials` e `finalScore`.

```
function fillOutScorecard(initials, finalScore) {  
    scorecard.display = initials;  
    scorecard.score = finalScore;  
}
```

Quando la funzione viene chiamata è necessario che gli argomenti richiesti siano passati alla funzione. La funzione sostituisce i valori passati agli argomenti elencati nella definizione della funzione. Nell'esempio seguente `scorecard` è il nome dell'istanza di un clip filmato, mentre `display` e `score` sono campi di testo di input nell'istanza. La seguente chiamata di funzione assegna alla variabile `display` il valore "JEB" e alla variabile `score` il valore 45000:

```
fillOutScorecard("JEB", 45000);
```

L'argomento `initials` nella funzione `fillOutScorecard` è simile a una variabile locale, esiste durante la chiamata della funzione e scompare al momento dell'uscita dalla funzione. Se si omettono argomenti in una chiamata di funzione, gli argomenti omessi vengono passati come `undefined`. Se in una chiamata di funzione si forniscono argomenti aggiuntivi non richiesti dalla dichiarazione della funzione, questi vengono ignorati.

Uso di variabili locali in una funzione

Le variabili locali sono strumenti utili per organizzare il codice e renderlo più comprensibile. Quando una funzione usa variabili locali può nascondere le proprie variabili a tutti gli altri script del filmato. Le variabili locali hanno come area di validità il corpo della funzione e scompaiono all'uscita dalla funzione. Gli argomenti passati a una funzione vengono considerati come variabili locali.

Nota: Se si modificano variabili globali in una funzione, documentare le modifiche tramite commenti nello script.

Restituzione di valori da una funzione

È possibile usare l'azione `return` affinché una funzione restituisca un valore. L'azione `return` interrompe la funzione sostituendo ad essa il valore dell'azione `return`. Se Flash non incontra un'azione `return` prima della fine di una funzione, viene restituita una stringa vuota. Ad esempio, la seguente funzione restituisce il quadrato dell'argomento `x`:

```
function sqr(x) {  
    return x * x;  
}
```

Alcune funzioni eseguono una serie di operazioni senza restituire un valore. Ad esempio, la funzione seguente inizializza un gruppo di variabili globali:

```
function initialize() {  
    boat_x = _root.boat._x;  
    boat_y = _root.boat._y;  
    car_x = _root.car._x;  
    car_y = _root.car._y;  
}
```

Chiamata di una funzione

Per chiamare una funzione tramite il pannello Azioni in Modalità normale, usare l'azione *evaluate*. Passare gli argomenti richiesti racchiudendoli tra parentesi. È possibile chiamare una funzione in qualsiasi linea temporale da qualsiasi linea temporale, compreso un filmato caricato. Ad esempio, l'istruzione seguente chiama la funzione *sqr* nel clip filmato *MathLib* della linea temporale principale, passa alla funzione l'argomento 3 e memorizza il risultato nella variabile *temp*:

```
var temp = _root.MathLib.sqr(3);
```

In Flash 4 per simulare la chiamata di una funzione è possibile creare uno script in un fotogramma dopo la fine del filmato, quindi chiamarlo passando il nome dell'etichetta del fotogramma all'azione *call*. Ad esempio, uno script che inizializza variabili e si trova in un fotogramma con etichetta *initialize* può essere chiamato con la sintassi seguente:

```
call("initialize");
```

Questo tipo di script non è una funzione vera e propria, in quanto non può accettare argomenti né restituire un valore. L'azione *call* è ancora disponibile in Flash 5, ma il suo uso è sconsigliato.

Uso di oggetti predefiniti

È possibile usare gli oggetti predefiniti di Flash per accedere a determinati tipi di informazioni. La maggior parte degli oggetti predefiniti dispone di *metodi* (funzioni assegnate a un oggetto) che è possibile chiamare affinché restituiscano un valore o eseguano un'azione. Ad esempio, l'oggetto Date restituisce informazioni provenienti dall'orologio di sistema mentre l'oggetto Sound consente il controllo degli elementi audio del filmato.

Alcuni oggetti predefiniti dispongono di proprietà con valori leggibili. Ad esempio, l'oggetto Key dispone di valori costanti corrispondenti ai tasti della tastiera. Ogni oggetto dispone di caratteristiche e funzionalità proprie utilizzabili nel filmato.

Di seguito sono elencati gli oggetti predefiniti di Flash:

- Array
- Boolean
- Color
- Date
- Tasto
- Math
- MovieClip
- Number
- Object
- Selection
- Audio
- String
- XML
- XMLSocket

Le istanze dei clip filmato vengono rappresentate come oggetti in ActionScript. È possibile chiamare i metodi predefiniti dei clip filmato seguendo le stesse regole valide per chiamare i metodi di qualsiasi altro oggetto ActionScript.

Per ulteriori informazioni su ciascun oggetto, consultare le singole voci nel capitolo 7 “Dizionario di ActionScript”.

Creazione di un oggetto

È possibile creare un oggetto in due modi: tramite l'operatore `new` e tramite l'operatore di inizializzazione degli oggetti (`{}`). L'operatore `new` consente di creare un oggetto da una classe oggetto predefinita o da una classe oggetto personalizzata. L'operatore di inizializzazione degli oggetti (`{}`) consente di creare un oggetto di tipo generico `Object`.

L'operatore `new` deve essere usato insieme a una funzione di costruzione. Una funzione di costruzione è una funzione il cui unico scopo consiste nel creare un determinato tipo di oggetto. Gli oggetti predefiniti di `ActionScript` sono di fatto funzioni di costruzione predefinite. Il nuovo oggetto *crea un'istanza* dell'oggetto, a cui assegna tutte le proprietà e i metodi dell'oggetto originale. Questa operazione è simile al trascinamento di un clip filmato dalla libreria allo stage in un filmato. Ad esempio, le seguenti istruzioni creano un'istanza di un oggetto `Date`:

```
currentDate = new Date();
```

È possibile accedere ai metodi di alcuni oggetti predefiniti senza creare istanze di tali oggetti. Ad esempio, l'istruzione seguente chiama il metodo `random` dell'oggetto `Math`.

```
Math.random();
```

Ogni oggetto che richiede una funzione di costruzione dispone di un elemento corrispondente nella lista nel riquadro a sinistra del pannello Azioni, ad esempio `new Color`, `new Date`, `new String` e così via.

Per creare un oggetto tramite l'operatore `new` in Modalità normale:

- 1 Scegliere `setVariable`.
- 2 Immettere un nome di variabile nel campo Nome.
- 3 Immettere `new Object`, `new Color` e così via nel campo Valore. Immettere tra parentesi gli argomenti richiesti dalla funzione di costruzione.
- 4 Selezionare la casella Espressione del campo Valore.

Se non si seleziona la casella Espressione, l'intero valore sarà interpretato come un valore stringa.

Nel codice seguente l'oggetto `c` viene creato tramite la funzione di costruzione `Color`:

```
c = new Color(this);
```

Nota: Un nome di oggetto è una variabile alla quale è assegnato il tipo di dati oggetto.

Per accedere a un metodo in Modalità normale:

- 1 Selezionare l'azione `evaluate`.
- 2 Immettere il nome dell'oggetto nel campo Espressione.
- 3 Immettere una proprietà dell'oggetto nel campo Espressione.

Per usare l'operatore di inizializzazione degli oggetti {} in Modalità normale:

- 1 Selezionare l'azione `setVariable`.
- 2 Immettere un nome nel campo Variabile che corrisponderà al nome del nuovo oggetto.
- 3 Immettere le coppie nome di proprietà e valore separate da due punti all'interno dell'operatore di inizializzazione degli oggetti (`{}`).

Ad esempio, nell'istruzione seguente i nomi delle proprietà sono `radius` e `area` e i valori sono `5` e il valore di un'espressione:

```
myCircle = {radius: 5, area:(pi * radius * radius)};
```

Le parentesi determinano la valutazione dell'espressione. Il valore restituito viene assegnato alla variabile `area`.

È inoltre possibile annidare inizializzatori di oggetti e matrici, come nella seguente istruzione:

```
newObject = {name: "John Smith", projects: ["Flash",  
"Dreamweaver"]};
```

Per ulteriori informazioni su ciascun oggetto, consultare le singole voci nel capitolo 7 “Dizionario di ActionScript”.

Accesso alle proprietà di un oggetto

Usare l'operatore punto (`.`) per accedere al valore delle proprietà di un oggetto. Specificare il nome dell'oggetto a sinistra del punto e il nome della proprietà a destra. Ad esempio, nell'istruzione seguente `myObject` è l'oggetto mentre `name` è la proprietà:

```
myObject.name
```

Per assegnare un valore a una proprietà in Modalità normale, usare l'azione `setVariable`:

```
myObject.name = "Allen";
```

Per modificare il valore di una proprietà, assegnare un nuovo valore come nell'esempio seguente:

```
myObject.name = "Homer";
```

È inoltre possibile accedere alle proprietà di un oggetto tramite l'operatore di accesso matrice (`[]`). Consultare “Operatori punto e accesso matrice” a pagina 67.

Chiamata dei metodi di un oggetto

È possibile chiamare il metodo di un oggetto tramite l'operatore punto seguito dal metodo. Nell'esempio seguente viene chiamato il metodo `setVolume` dell'oggetto `Sound`.

```
s = new Sound(this);  
s.setVolume(50);
```

Per chiamare il metodo di un oggetto predefinito in Modalità normale, usare l'azione `evaluate`.

Uso dell'oggetto MovieClip

I metodi dell'oggetto predefinito `MovieClip` consentono di controllare istanze del simbolo clip filmato nello stage. Il codice dell'esempio seguente richiede la riproduzione dell'istanza `dateCounter`:

```
dateCounter.play();
```

Per ulteriori informazioni sull'oggetto `MovieClip`, consultare le singole voci nel capitolo 7 “Dizionario di ActionScript”.

Uso dell'oggetto Array

L'oggetto `Array` è un oggetto predefinito di ActionScript di uso comune che memorizza i propri dati in proprietà identificate da numeri invece che da nomi. Il nome di un elemento della matrice è detto *indice*. Questo oggetto è utile per la memorizzazione e l'accesso di determinati tipi di dati, quali liste di studenti o una sequenza di azioni in un gioco.

L'assegnazione di elementi dell'oggetto `Array` avviene con modalità analoghe a quelle per le proprietà di qualsiasi oggetto:

```
move[1] = "a2a4";  
move[2] = "h7h5";  
move[3] = "b1c3";  
...  
move[100] = "e3e4";
```

Per accedere al secondo elemento della matrice, usare l'espressione `move[2]`.

L'oggetto `Array` dispone della proprietà predefinita `length` corrispondente al numero di elementi della matrice. Quando viene assegnato un elemento dell'oggetto `Array` e l'indice dell'elemento è un intero positivo che soddisfa l'uguaglianza `index >= length`, `length` viene automaticamente aggiornata a `index + 1`.

Uso di oggetti personalizzati

La creazione di oggetti personalizzati definendo le proprietà e i metodi consente di organizzare i dati all'interno degli script in modo da semplificarne l'archiviazione e l'accesso. Una volta creato un oggetto master o classe, è possibile usare o creare istanze dell'oggetto in un filmato. Ciò consente di riusare parti di codice e di limitare le dimensioni dei file.

Un oggetto è un tipo di dati complesso, contenente zero o più proprietà. Ogni proprietà, come una variabile, è provvista di un nome e di un valore. Le proprietà sono associate all'oggetto e contengono valori modificabili e recuperabili, il cui tipo può essere stringa, numerico, booleano, oggetto, clip filmato o `undefined`. Le proprietà seguenti sono di tipi di dati diversi:

```
customer.name = "Jane Doe"
customer.age = 30
customer.member = true
customer.account.currentRecord = 000609
customer.mcInstanceName._visible = true
```

La proprietà di un oggetto può a sua volta essere un oggetto. Nella riga 4 dell'esempio precedente `account` è una proprietà dell'oggetto `customer` e `currentRecord` è una proprietà dell'oggetto `account`. Il tipo di dati della proprietà `currentRecord` è numerico.

Creazione di un oggetto

È possibile usare l'operatore `new` per creare un oggetto da una funzione di costruzione. A una funzione di costruzione è sempre assegnato lo stesso nome del tipo di oggetto che crea. Ad esempio, una funzione di costruzione che crea un oggetto `Account` verrà denominata `Account`. La seguente istruzione crea un nuovo oggetto dalla funzione chiamata `MyConstructorFunction`:

```
new MyConstructorFunction (argument1, argument2, ... argumentN);
```

Quando viene chiamata la funzione `MyConstructorFunction`, Flash passa l'argomento nascosto `this`, che è un riferimento all'oggetto che `MyConstructorFunction` sta creando. Quando si definisce una funzione di costruzione `this` consente di far riferimento agli oggetti che verranno creati dalla funzione di costruzione stessa. Ad esempio, la seguente funzione di costruzione crea un cerchio:

```
function Circle(radius) {
    this.radius = radius;
    this.area = Math.PI * radius * radius;
}
```

Le funzioni di costruzione vengono usate comunemente per definire i metodi di un oggetto.

```
function Area() {  
    this.circleArea = Math.PI * radius * radius;  
}
```

Per usare un oggetto in uno script è necessario assegnarlo a una variabile. Per creare un nuovo oggetto cerchio con raggio 5 usare l'operatore `new`, che consente di creare l'oggetto e assegnarlo alla variabile locale `myCircle`:

```
var myCircle = new Circle(5);
```

Nota: Gli oggetti hanno la stessa area di validità della variabile alla quale sono assegnati. Consultare "Assegnazione di un'area di validità a una variabile" a pagina 58.

Creazione dell'ereditarietà

Tutte le funzioni dispongono di una proprietà `prototype` che viene creata automaticamente al momento della definizione della funzione. Quando si usa una funzione di costruzione per creare un nuovo oggetto, tutte le proprietà e i metodi della proprietà `prototype` della funzione di costruzione diventano metodi e proprietà della proprietà `__proto__` del nuovo oggetto. La proprietà `prototype` indica i valori predefiniti delle proprietà degli oggetti creati con tale funzione. Il passaggio di valori tramite le proprietà `__proto__` e `prototype` è detto ereditarietà.

L'ereditarietà si basa su una gerarchia precisa. Quando si chiama una proprietà o un metodo di un oggetto, ActionScript verifica se tale elemento esiste nell'oggetto. Se l'elemento non esiste, ActionScript ricerca i dati necessari nella proprietà `__proto__` dell'oggetto (`object.__proto__`). Se la proprietà chiamata non è una proprietà dell'oggetto `__proto__`, ActionScript ricerca in `object.__proto__.__proto__`.

I metodi vengono spesso associati a un oggetto assegnandoli alla proprietà `prototype` dell'oggetto. La procedura seguente descrive come definire un metodo di esempio.

- 1 Definire la funzione di costruzione `Circle` nel modo seguente:

```
function Circle(radius) {  
    this.radius = radius  
}
```

- 2 Definire il metodo `area` dell'oggetto `Circle`. Il metodo `area` calcola l'area del cerchio. Per definire il metodo `area` e impostare la proprietà `area` dell'oggetto prototipo del cerchio è possibile usare un valore letterale di funzione, nel modo seguente:

```
Circle.prototype.area = function () {  
    return Math.PI * this.radius * this.radius  
}
```

3 Creare un'istanza dell'oggetto `Circle` nel modo seguente:

```
var myCircle = new Circle(4);
```

4 Chiamare il metodo `area` del nuovo oggetto `myCircle` nel modo seguente:

```
var myCircleArea = myCircle.area()
```

ActionScript ricerca nell'oggetto `myCircle` il metodo `area`. Poiché l'oggetto non dispone di un metodo `area`, il metodo `area` viene ricercato nell'oggetto prototipo `Circle.prototype`. ActionScript trova il metodo ed esegue la chiamata.

In alternativa è possibile associare un metodo a un oggetto associandolo a ogni singola istanza dell'oggetto, come nell'esempio seguente:

```
function Circle(radius) {  
    this.radius = radius  
    this.area = function() {  
        return Math.PI * this.radius * this.radius  
    }  
}
```

Questa tecnica è tuttavia sconsigliata. L'uso dell'oggetto `prototype` è più efficiente, in quanto è sufficiente una sola definizione di `area` e tale definizione viene automaticamente copiata in tutte le istanze create dalla funzione `Circle`.

La proprietà `prototype` è supportata da Flash Player versione 5 e successive. Per ulteriori informazioni, consultare il capitolo 7 “Dizionario di ActionScript”.

Apertura di file Flash 4

ActionScript ha subito importanti modifiche con la versione di Flash 5 diventando un linguaggio orientato agli oggetti con tipi di dati multipli e che usa la sintassi del punto. ActionScript di Flash 4 disponeva di un solo vero tipo di dati: stringa. Il linguaggio usava diversi tipi di operatori nelle espressioni per indicare se il valore doveva essere considerato come stringa o come valore numerico. In Flash 5 è possibile usare un solo insieme di operatori per tutti i tipi di dati.

Quando in Flash 5 si apre un file creato in Flash 4, le espressioni di ActionScript vengono convertite automaticamente per renderle compatibili con la nuova sintassi di Flash 5. Nel codice ActionScript appaiono le seguenti conversioni di tipi di dati e operatori:

- L'operatore `=` veniva usato in Flash 4 per esprimere l'uguaglianza numerica. In Flash 5 `==` è l'operatore di uguaglianza e `=` è l'operatore di assegnazione. Gli operatori `=` dei file in Flash 4 vengono convertiti automaticamente in `==`.
- Flash esegue automaticamente le conversioni dei tipi di dati per garantire il funzionamento corretto degli operatori. A causa dell'introduzione di tipi di dati multipli, gli operatori seguenti hanno assunto un nuovo significato:
`+`, `==`, `!=`, `<>`, `<`, `>`, `>=`, `<=`
- In ActionScript di Flash 4 tali operatori erano sempre operatori numerici. In Flash 5 gli operatori funzionano in modo diverso a seconda dei tipi di dati degli operandi. Per evitare differenze semantiche nei file importati, viene applicata la funzione `Number` a tutti gli operandi di tali operatori. Ai numeri costanti non viene applicata la funzione `Number` in quanto sono ovviamente valori numerici.
- In Flash 4 la sequenza di escape `\n` generava un carattere di ritorno a capo (ASCII 13). In Flash 5, per osservanza dello standard ECMA-262, `\n` genera un carattere di avanzamento riga (ASCII 10). Una sequenza di escape `\n` dei file FLA di Flash 4 viene automaticamente convertita in `\r`.
- L'operatore `&` veniva usato in Flash 4 per l'addizione di stringhe. In Flash 5 `&` è l'operatore AND bit a bit. Il nuovo operatore di addizione stringhe è `add`. Gli operatori `&` dei file in Flash 4 vengono convertiti automaticamente in operatori `add`.
- Molte funzioni di Flash 4, quali `Get Timer`, `Set Variable`, `Stop` e `Play` non richiedevano parentesi finali. Per garantire una sintassi coerente, la funzione di Flash 5 `getTimer` e tutte le azioni richiedono la presenza di parentesi finali. Tali parentesi vengono aggiunte automaticamente durante la conversione.

- Quando la funzione `getProperty` viene eseguita su un clip filmato che non esiste, in Flash 5 viene restituito il valore `undefined` e non il valore 0. Inoltre, in ActionScript di Flash 5, `undefined == 0` è `false`. Per la conversione dai file Flash 4 il problema viene risolto introducendo la funzione `Number` nei confronti di uguaglianza. Nell'esempio seguente `Number` impone la conversione di `undefined` in 0 per consentire la riuscita del confronto:

```
getProperty("clip", _width) == 0
Number(getProperty("clip", _width)) == Number(0)
```

Nota: Se in ActionScript di Flash 4 sono state usate parole chiave di Flash 5 come nomi di variabili, Flash 5 genererà un errore di sintassi. Per ovviare, rinominare tutte le occorrenze delle variabili. Consultare “Parole chiave” a pagina 53.

Uso di Flash 5 per creare contenuto Flash 4

Se si usa Flash 5 per creare codice per Flash 4 Player (esportando nel formato Flash 4) non sarà possibile avvalersi di tutte le nuove funzionalità disponibili in ActionScript di Flash 5, ma solo di alcune. ActionScript di Flash 4 dispone di un solo tipo di dati di base, usato sia per la gestione dei dati numerici che per la gestione dei dati stringa. Quando si crea un filmato per Flash 4 Player è necessario usare gli operatori stringa obsoleti, disponibili nella categoria Operatori stringa nella lista nel riquadro a sinistra.

Le seguenti funzionalità di Flash 5 sono utilizzabili quando si esporta il filmato nel formato Flash 4 SWF:

- Operatore di accesso matrice e agli oggetti (`[]`).
- Operatore punto (`.`).
- Operatori logici, operatori di assegnazione, operatore decremento prima dell'operazione e operatori incremento/decremento dopo l'operazione.
- Operatore modulo (`%`), tutti i metodi e le proprietà dell'oggetto `Math`.

Questi operatori e funzioni non sono integrati in Flash 4 Player. Flash 5 esporta tali elementi come approssimazioni seriali. Ciò significa che i risultati sono soltanto approssimati. Inoltre, data l'inclusione di approssimazioni seriali nel file SWF, tali funzioni occupano più spazio nei file Flash 4 SWF che nei file Flash 5 SWF.

- Azioni `for`, `while`, `do while`, `break` e `continue`.
- Azioni `print` e `printAsBitmap`.

Le seguenti funzionalità di Flash 5 non sono utilizzabili nei filmati esportati nel formato Flash 4 SWF:

- Funzioni personalizzate
- Supporto XML
- Variabili locali
- Oggetti predefiniti (ad eccezione di Math)
- Azioni dei clip filmato
- Tipi di dati multipli
- `eval` con sintassi del punto (ad esempio `eval("_root.movieclip.variable")`)
- `return`
- `new`
- `delete`
- `typeof`
- `for..in`
- `keyCode`
- `targetPath`
- `escape`
- `globalToLocal` e `localToGlobal`
- `hitTest`
- `isFinite` e `isNaN`
- `parseFloat` e `parseInt`
- `tunescape`
- `_xmouse` e `_ymouse`
- `_quality`

CAPITOLO 3

Creazione di contenuto interattivo con ActionScript

.....

Un filmato interattivo coinvolge il pubblico. Usando la tastiera, il mouse o entrambi, gli utenti possono infatti passare a parti diverse dei filmati, spostare oggetti, immettere informazioni in moduli, fare clic su pulsanti ed eseguire molte altre operazioni interattive.

È possibile creare filmati interattivi impostando script che vengono eseguiti quando si verificano determinati eventi. Gli eventi che possono attivare uno script si verificano quando l'indicatore di riproduzione raggiunge un fotogramma, un clip filmato viene caricato o scaricato oppure l'utente fa clic su un pulsante o preme certi tasti sulla tastiera. ActionScript viene usato per creare script che comunicano a Flash quale azione eseguire quando si verifica l'evento.

Le azioni principali seguenti sono modi comuni di controllare la navigazione e l'interazione dell'utente in un filmato:

- Riproduzione e interruzione di filmati
- Regolazione della qualità di visualizzazione di un filmato
- Interruzione di tutti i suoni
- Passaggio a un dato fotogramma o scena
- Passaggio a un URL diverso
- Verifica del caricamento di un fotogramma
- Caricamento e scaricamento di filmati addizionali

Per ulteriori informazioni su queste azioni, consultare *Guida all'uso di Flash*.

Per creare un contenuto interattivo più complesso, è necessario avere padronanza delle seguenti tecniche:

- Creazione di un puntatore del mouse personalizzato
- Determinazione della posizione del mouse
- Rilevamento dei tasti premuti
- Creazione di un campo di testo scorrevole
- Impostazioni dei valori dei colori
- Creazione dei controlli audio
- Rilevamento della presenza di collisioni

Creazione di un puntatore del mouse personalizzato

Per nascondere il puntatore del mouse standard (cioè la rappresentazione su schermo del puntatore del mouse), è necessario usare il metodo `hide` dell'oggetto `Mouse` predefinito. Per usare un clip filmato come puntatore del mouse personalizzato, è necessario usare l'azione `startDrag`.



Azioni associate a un clip filmato per creare un puntatore del mouse personalizzato

Per creare un puntatore del mouse personalizzato:

- 1 Creare un clip filmato da usare come puntatore del mouse personalizzato.
- 2 Selezionare l'istanza del clip filmato sullo stage.
- 3 Scegliere Finestra > Azioni per aprire il pannello Azioni oggetto.

- 4 Nella lista nel riquadro a sinistra, selezionare Oggetti, quindi Mouse e trascinare `hide` nella finestra di script a destra.

Il codice generato dovrebbe assomigliare al seguente:

```
onClipEvent(load){  
    Mouse.hide();  
}
```

- 5 Nella lista nel riquadro a sinistra, selezionare Azioni e trascinare `startDrag` nella finestra di script a destra.

- 6 Selezionare la casella di controllo Blocca mouse al centro.

Il codice generato dovrebbe assomigliare al seguente:

```
onClipEvent(load){  
    Mouse.hide()  
    startDrag(this, true);  
}
```

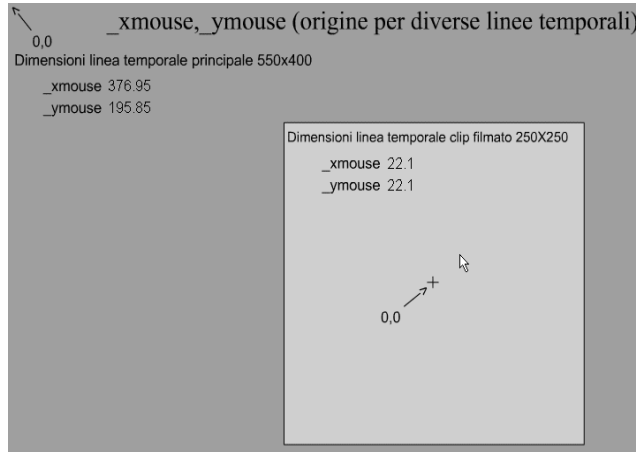
- 7 Scegliere Controlli > Prova filmato per usare il puntatore del mouse personalizzato.

I pulsanti funzionano anche quando si usa un puntatore del mouse personalizzato. Si consiglia di posizionare il puntatore del mouse personalizzato sul livello superiore della linea temporale in modo che sia in primo piano rispetto ai pulsanti e ad altri oggetti man mano che si sposta il mouse nel filmato.

Per ulteriori informazioni sui metodi dell'oggetto `Mouse`, consultare le voci corrispondenti nel capitolo 7 “Dizionario di `ActionScript`”.

Determinazione della posizione del mouse

È possibile usare le proprietà `_xmouse` e `_ymouse` per individuare la posizione del puntatore o cursore del mouse in un filmato. Ogni linea temporale comprende le proprietà `_xmouse` e `_ymouse` che restituiscono la posizione del mouse rispetto al relativo sistema di coordinate.



Le proprietà `_xmouse` e `_ymouse` all'interno della linea temporale principale e alla linea temporale di un clip filmato

L'istruzione seguente potrebbe essere posizionata su una linea temporale qualsiasi nel filmato `_level0` per restituire la posizione `_xmouse` all'interno della linea temporale principale:

```
x_pos = _root._xmouse;
```

Per determinare la posizione del mouse all'interno di un clip filmato, è possibile usare il nome dell'istanza del clip filmato. Ad esempio, l'istruzione seguente potrebbe essere posizionata su una linea temporale qualsiasi nel filmato `_level0` per restituire la posizione `_ymouse` nell'istanza `myMovieClip`:

```
y_pos = _root.myMovieClip._ymouse
```

È inoltre possibile determinare la posizione del mouse all'interno di un clip filmato usando le proprietà `_xmouse` e `_ymouse` in un'azione clip, come nell'esempio seguente:

```
onClipEvent(enterFrame){
    xmousePosition = _xmouse;
    ymousePosition = _ymouse;
}
```

Le variabili `x_pos` e `y_pos` sono usate come contenitori per memorizzare i valori delle posizioni del mouse. È possibile usare queste variabili in qualsiasi script del filmato. Nell'esempio seguente i valori di `x_pos` e `y_pos` vengono aggiornati ogni volta che l'utente sposta il mouse.

```
onClipEvent(mouseMove){  
    x_pos = _root._xmouse;  
    y_pos = _root._ymouse;  
}
```

Per ulteriori informazioni sulle proprietà `_xmouse` e `_ymouse`, consultare le voci corrispondenti nel capitolo 7 “Dizionario di ActionScript”.

Rilevamento dei tasti premuti

È possibile usare i metodi dell'oggetto `Key` predefinito per rilevare l'ultimo tasto premuto dall'utente. L'oggetto `Key` non richiede una funzione di costruzione. Per usarne i metodi è sufficiente chiamare l'oggetto stesso come nell'esempio seguente:

```
Key.getCode();
```

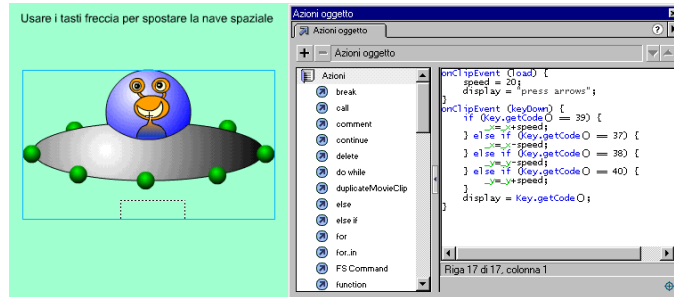
È possibile ottenere i codici di tasto virtuale o i valori ASCII dei tasti premuti:

- Per ottenere il codice di tasto virtuale dell'ultimo tasto premuto, usare il metodo `getCode`.
- Per ottenere il valore ASCII dell'ultimo tasto premuto, usare il metodo `getAscii`.

Il codice di tasto virtuale viene assegnato a ciascun tasto fisico di una tastiera. Ad esempio, il codice di tasto virtuale del tasto freccia sinistra è 37. Usando il codice di tasto virtuale si garantisce che i controlli del filmato sono gli stessi su tutte le tastiere indipendentemente dal linguaggio o dalla piattaforma.

I valori ASCII (American Standard Code for Information Interchange) sono assegnati ai primi 127 caratteri di ogni set di caratteri e forniscono informazioni sul carattere visualizzato sullo schermo. Ad esempio, la lettera “A” e la lettera “a” hanno valori ASCII diversi.

Un uso comune di `Key.getCode` è in un gestore `onClipEvent`. Passando `keyDown` come parametro, il gestore ordina ad `ActionScript` di rilevare il valore dell'ultimo tasto premuto solo quando il tasto viene effettivamente premuto. Questo esempio usa `Key.getCode` in un'istruzione `if` per creare i controlli di navigazione per la nave spaziale.



Per creare i controlli della tastiera per un filmato:

- 1 Decidere quali tasti usare e determinare i codici dei tasti virtuali usando uno degli approcci seguenti:
 - Consultare la lista dei codici tasto nell'appendice B “Tasti della tastiera e valori dei codici tasto”.
 - Usare una costante dell'oggetto `Key`. Nel pannello Azioni, nella lista nel riquadro a sinistra, selezionare Oggetti, quindi `Key`. Le costanti sono elencate tutte in maiuscolo.
 - Assegnare l'azione clip seguente, quindi scegliere Controlli > Prova filmato e premere il tasto desiderato:


```
onClipEvent(keyDown) {
    trace(Key.getCode());
}
```
- 2 Selezionare un clip filmato sullo stage.
- 3 Scegliere Finestra > Azioni.
- 4 Fare doppio clic sull'azione `onClipEvent` nella categoria Azioni della lista nel riquadro sinistro.
- 5 Scegliere l'evento `Key down` nel riquadro dei parametri.
- 6 Fare doppio clic sull'azione `if` nella categoria Azioni della lista nel riquadro sinistro.
- 7 Fare clic sul parametro Condizione, quindi selezionare Oggetti, quindi `Key` e `getCode`.

- 8 Fare doppio clic sull'operatore di uguaglianza (==) nella categoria Operatori della lista nel riquadro sinistro.
- 9 Immettere il codice di tasto virtuale a destra dell'operatore di uguaglianza.

Il codice generato dovrebbe assomigliare al seguente:

```
onClipEvent(keyDown) {  
    if (Key.getCode() == 32) {  
    }  
}
```

- 10 Selezionare un'azione da eseguire se viene premuto il tasto corretto.

Ad esempio, l'azione seguente causa lo spostamento della linea temporale principale al fotogramma successivo quando viene premuta la barra spaziatrice (32):

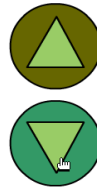
```
onClipEvent(keyDown) {  
    if (Key.getCode() == 32) {  
        nextFrame();  
    }  
}
```

Per ulteriori informazioni sui metodi dell'oggetto Key, consultare le voci corrispondenti nel capitolo 7 “Dizionario di ActionScript”.

Creazione di un campo di testo scorrevole

È possibile usare le proprietà `scroll` e `maxscroll` per creare un campo di testo scorrevole.

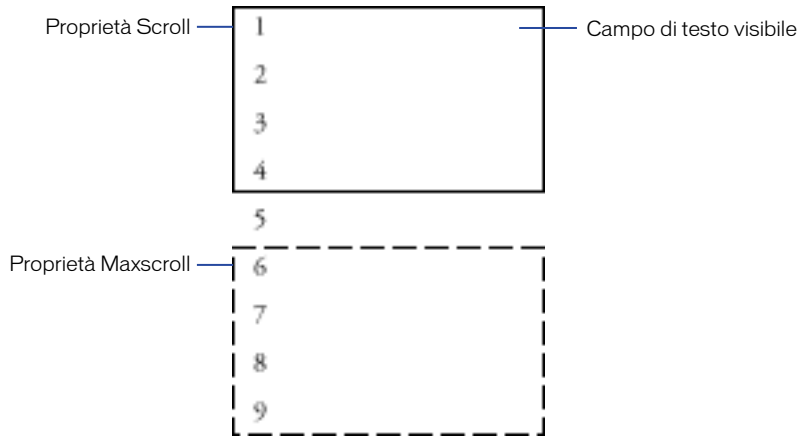
Proinde cum venabere, licebit, aucte
Ridebis, et licet rideas. Ego ille quen
Iam undique silvae et solitudo ipsum
Proinde cum venabere, licebit, aucte
Ridebis, et licet rideas. Ego ille quen
Iam undique silvae et solitudo ipsum
Proinde cum venabere, licebit, aucte
Ridebis. et licet rideas. Ego ille quen



Nel pannello Opzioni testo è possibile assegnare una variabile a qualsiasi campo di testo impostato su Testo di input o Testo dinamico. Il campo di testo funziona come una finestra che visualizza il valore della variabile.

Ogni variabile associata a un campo di testo comprende le proprietà `scroll` e `maxscroll`. È possibile usare queste proprietà per scorrere il testo in un campo di testo. La proprietà `scroll` restituisce il numero della riga visibile superiore in un campo di testo. È possibile sia impostare che recuperare il valore di questa proprietà. La proprietà `maxscroll` restituisce il numero della riga visibile superiore in un campo di testo quando l'ultima riga di testo è visibile. Questa proprietà può essere letta, ma non impostata.

Ad esempio, se un campo di testo è lungo quattro righe e in esso è contenuta la variabile *speech*, che occupa nove righe, è possibile visualizzare solo una parte di questa variabile alla volta (identificata dalla casella a linee continue):

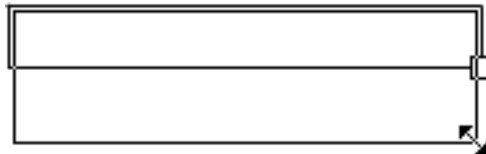


È possibile accedere a queste proprietà usando la sintassi del punto, come nell'esempio seguente:

```
textFieldVariable.scroll
myMovieClip.textFieldVariable.scroll
textFieldVariable.maxscroll
myMovieClip.textFieldVariable.maxscroll
```

Per creare un campo di testo scorrevole:

- 1 Trascinare un campo di testo sullo stage.
- 2 Scegliere Finestra > Pannelli > Opzioni testo.
- 3 Scegliere Testo di input dal menu a comparsa.
- 4 Immettere il nome della variabile **text** nel campo Variabile.
- 5 Trascinare l'angolo inferiore destro del campo di testo per ridimensionare il campo.



- 6 Scegliere Finestra > Azioni.

- 7 Selezionare il fotogramma 1 nella linea temporale principale e assegnare un'azione `set variable` che imposta il valore di `text`.

Nel campo non verrà visualizzato alcun testo fino a quando non si imposta la variabile. Pertanto, sebbene si possa assegnare questa azione a un fotogramma, pulsante o clip filmato qualsiasi, si consiglia di assegnarla al fotogramma 1 sulla linea temporale principale, come illustrato di seguito:



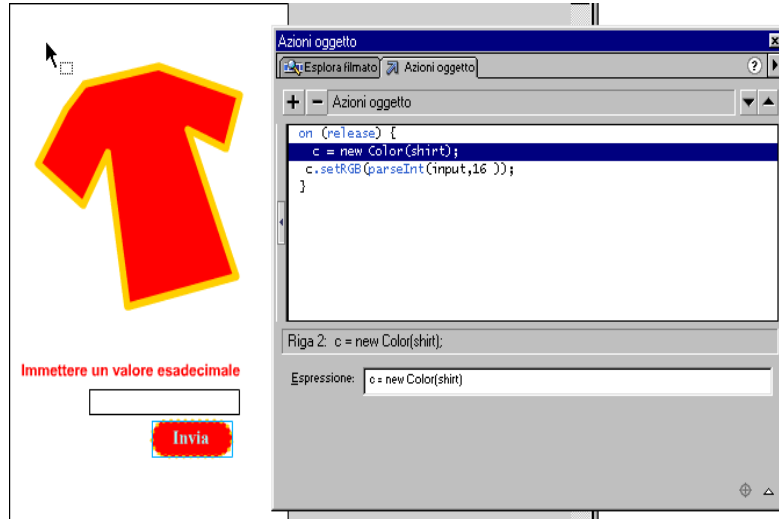
- 8 Scegliere Finestra > Librerie comuni > Pulsanti e trascinare un pulsante sullo stage.
- 9 Premere Alt (Windows) oppure Opzione (Macintosh) e trascinare il pulsante per creare una copia.
- 10 Selezionare il pulsante superiore e scegliere Finestra > Azioni.
- 11 Nel pannello Azioni, trascinare l'azione `set variables` dalla lista nel riquadro a sinistra nella finestra di script a destra.
- 12 Immettere `text.scroll` nella casella Variabile.
- 13 Immettere `text.scroll -1` nella casella Valore e selezionare la casella di controllo Espressione.
- 14 Selezionare il pulsante Freccia giù e assegnare l'azione `set variables` seguente:

```
text.scroll = text.scroll+1;
```
- 15 Scegliere Controlli > Prova filmato per osservare il funzionamento del campo di testo scorrevole.

Per ulteriori informazioni sulle proprietà `scroll` e `maxscroll`, consultare le voci corrispondenti nel capitolo 7 “Dizionario di ActionScript”.

Impostazioni dei valori dei colori

È possibile usare i metodi dell'oggetto Color predefinito per modificare il colore di un clip filmato. Il metodo `setRGB` assegna valori RGB (red, green, blue) esadecimali all'oggetto, mentre il metodo `setTransform` imposta la percentuale e i valori di offset per i componenti rosso, verde, blu e alfa (trasparenza) di un colore. L'esempio seguente usa `setRGB` per cambiare il colore di un oggetto in base all'input dell'utente.



L'azione pulsante crea un oggetto Color e cambia il colore della maglietta in base all'input dell'utente.

Per usare l'oggetto Color, è necessario creare un'istanza dell'oggetto e applicarla a un clip filmato.

Per impostare il valore del colore di un clip filmato:

- 1 Selezionare un clip filmato sullo stage, quindi scegliere Finestra > Pannelli > Istanza.
- 2 Immettere il nome dell'istanza **colorTarget** nella casella Nome.
- 3 Trascinare un campo di testo sullo stage.
- 4 Scegliere Finestra > Pannelli > Opzioni testo e assegnare al campo di testo il nome di variabile **input**.
- 5 Trascinare un pulsante sullo stage e selezionarlo.
- 6 Scegliere Finestra > Azioni.

- 7 Trascinare l'azione `set variable` dalla lista nel riquadro a sinistra nella finestra di script a destra.
- 8 Nella casella Variabile immettere `c`.
- 9 Nella lista nel riquadro a sinistra, selezionare Oggetti, quindi Color e trascinare `new Color` nella casella Valore.
- 10 Selezionare la casella di controllo Espressione.
- 11 Fare clic sul pulsante a forma di mirino per inserire un percorso target e selezionare `colorTarget`. Fare clic su OK.

Il codice nella finestra di script dovrebbe assomigliare al seguente:

```
on(release) {  
    c = new Color(colorTarget);  
}
```

- 12 Trascinare l'azione `evaluate` dalla lista nel riquadro a sinistra nella finestra di script a destra.
- 13 Immettere `c` nella casella Espressione.
- 14 Nella categoria Oggetti della lista nel riquadro a sinistra, selezionare Color, quindi trascinare `setRGB` nella casella Espressione.
- 15 Selezionare Funzioni e trascinare `parseInt` nella casella Espressione.

Il codice generato dovrebbe assomigliare al seguente:

```
on(release) {  
    c = new Color(colorTarget);  
    c.setRGB(parseInt(string, radix));  
}
```

- 16 Per l'argomento della stringa `parseInt` immettere `input`.

La stringa di cui occorre eseguire l'analisi sintattica è il valore immesso nel campo di testo modificabile.

- 17 Per l'argomento della radice `parseInt` immettere `16`.

La radice è la base del sistema numerico di cui occorre eseguire l'analisi sintattica. In questo caso 16 è la base del sistema esadecimale usato dall'oggetto Color. Il codice generato dovrebbe assomigliare al seguente:

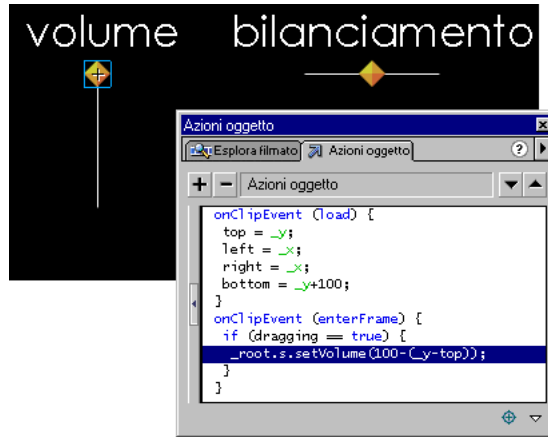
```
on(release) {  
    c = new Color(colorTarget);  
    c.setRGB(parseInt(input, 16));  
}
```

- 18 Scegliere Controlli > Prova filmato per cambiare il colore del clip filmato.

Per ulteriori informazioni sui metodi dell'oggetto Color, consultare le voci corrispondenti nel capitolo 7 “Dizionario di ActionScript”.

Creazione dei controlli audio

Per controllare i suoni in un filmato, usare l'oggetto Sound predefinito. Per usare i metodi dell'oggetto Sound, è necessario dapprima creare un nuovo oggetto Sound. Quindi è possibile usare il metodo `attachSound` per inserire un suono della libreria in un filmato mentre il filmato è in esecuzione. Il metodo `setVolume` dell'oggetto Sound controlla il volume, mentre il metodo `setPan` regola il bilanciamento destro e sinistro dell'audio. L'esempio seguente usa `setVolume` e `setPan` per creare i controlli del volume e del bilanciamento che l'utente può regolare.



Quando l'utente trascina il cursore del volume, viene chiamato il metodo `setVolume`.

Per associare un suono a una linea temporale:

- 1 Scegliere File > Importa per importare un suono.
- 2 Selezionare il suono nella libreria e scegliere Concatenamento dal menu Opzioni.
- 3 Selezionare Esporta questo simbolo e assegnare al suono l'identificatore **mySound**.
- 4 Selezionare il fotogramma 1 nella linea temporale principale, quindi scegliere Finestra > Azioni.
- 5 Trascinare l'azione `set variable` dalla lista nel riquadro a sinistra nella finestra di script a destra.

- 6 Immettere `s` nella casella Valore.
- 7 Nella lista nel riquadro a sinistra, selezionare Oggetti, quindi Sound e trascinare `new Sound` nella casella Valore.

Il codice generato dovrebbe assomigliare al seguente:

```
s = new Sound();
```

- 8 Fare doppio clic sull'azione `evaluate` nella lista nel riquadro sinistro.
- 9 Immettere `s` nella casella Espressione.
- 10 Nella categoria Oggetti della lista nel riquadro a sinistra, selezionare Sound, quindi trascinare `attachSound` nella casella Espressione.
- 11 Immettere **“mySound”** nell'argomento ID di `attachSound`.
- 12 Fare doppio clic sull'azione `evaluate` nella lista nel riquadro sinistro.
- 13 Immettere `s` nella casella Espressione.
- 14 Nella categoria Oggetti selezionare Sound, quindi trascinare `start` nella casella Espressione.

Il codice generato dovrebbe assomigliare al seguente:

```
s = new Sound();  
s.attachSound("mySound");  
s.start();
```

- 15 Scegliere Controlli > Prova filmato per sentire il suono.

Per creare un controllo scorrevole del volume:

- 1 Trascinare un pulsante sullo stage.
- 2 Selezionare il pulsante e scegliere Inserisci > Converti in simbolo. Scegliere il comportamento del clip filmato.

In questo modo si crea un clip filmato con il pulsante nel primo fotogramma.
- 3 Selezionare il clip filmato e scegliere Modifica > Modifica simbolo.
- 4 Selezionare il pulsante e scegliere Finestra > Azioni.

5 Immettere le azioni seguenti:

```
on (press) {  
    startDrag ("", false, left, top, right, bottom);  
    dragging = true;  
}  
on (release, releaseOutside) {  
    stopDrag ();  
    dragging = false;  
}
```

I parametri `left`, `top`, `right` e `bottom` di `startDrag` sono variabili impostate in un'azione clip.

6 Scegliere **Modifica > Modifica filmato** per tornare alla linea temporale principale.

7 Selezionare il clip filmato sullo stage.

8 Immettere le azioni seguenti:

```
onClipEvent (load) {  
    top=_y;  
    left=_x;  
    right=_x;  
    bottom=_y+100;  
}  
  
onClipEvent(enterFrame){  
    if (dragging==true){  
        _root.s.setVolume(100-(_y-top));  
    }  
}
```

9 Scegliere **Controlli > Prova filmato** per usare il cursore del volume.

Per creare un controllo scorrevole del bilanciamento:

1 Trascinare un pulsante sullo stage.

2 Selezionare il pulsante e scegliere **Inserisci > Converti in simbolo**. Scegliere la proprietà del clip filmato.

3 Selezionare il clip filmato e scegliere **Modifica > Modifica simbolo**.

4 Selezionare il pulsante e scegliere **Finestra > Azioni**.

5 Immettere le azioni seguenti:

```
on (press) {  
    startDrag ("", false, left, top, right, bottom);  
    dragging = true;  
}  
on (release, releaseOutside) {  
    stopDrag ();  
    dragging = false;  
}
```

I parametri `left`, `top`, `right` e `bottom` di `startDrag` sono variabili impostate in un'azione clip.

6 Scegliere Modifica > Modifica filmato per tornare alla linea temporale principale.

7 Selezionare il clip filmato sullo stage.

8 Immettere le azioni seguenti:

```
onClipEvent(load){  
    top=_y;  
    bottom=_y;  
    left=_x-50;  
    right=_x+50;  
    center=_x;  
}  
  
onClipEvent(enterFrame){  
    if (dragging==true){  
        _root.s.setPan((_x-center)*2);  
    }  
}
```

9 Scegliere Controlli > Prova filmato per usare il cursore del bilanciamento.

Per ulteriori informazioni sui metodi dell'oggetto `Sound`, consultare le voci corrispondenti nel capitolo 7 “Dizionario di `ActionScript`”.

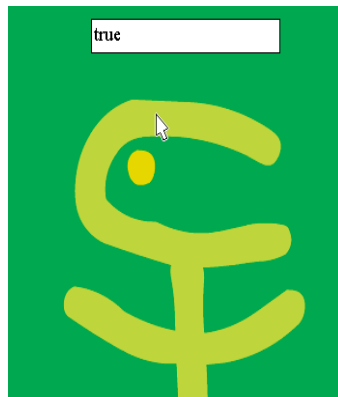
Rilevamento della presenza di collisioni

È possibile usare il metodo `hitTest` dell'oggetto `MovieClip` per rilevare la presenza di collisioni in un filmato. Il metodo `hitTest` verifica se un oggetto è entrato in collisione con un clip filmato e restituisce un valore booleano (`true` o `false`). È possibile usare i parametri del metodo `hitTest` per specificare le coordinate x e y di un'area attiva sullo stage o usare il percorso target di un altro clip filmato come area attiva.

Ogni clip filmato di un filmato è un'istanza dell'oggetto `MovieClip`. In questo modo è possibile chiamare i metodi dell'oggetto da qualsiasi istanza, come nell'esempio seguente:

```
myMovieClip.hitTest(target);
```

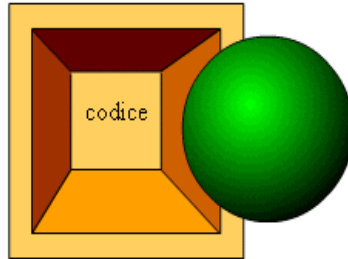
È possibile usare il metodo `hitTest` per verificare la presenza di una collisione di un clip filmato e un punto singolo.



“True” viene visualizzato nel campo di testo quando il puntatore è sull'area colorata.

È possibile usare il metodo `hitTest` per verificare la presenza di una collisione tra due clip filmato.

true



“True” viene visualizzato nel campo di testo quando i due filmati si toccano.

Per rilevare la collisione tra un clip filmato e un punto sullo stage:

- 1 Selezionare un clip filmato sullo stage.
- 2 Scegliere Finestra > Azioni per aprire il pannello Azioni oggetto.
- 3 Fare doppio clic su `trace` nella categoria Azioni della lista nel riquadro sinistro.

- 4 Selezionare la casella di controllo Espressione e immettervi:

```
trace (this.hitTest(_root._xmouse, _root._ymouse, true);
```

In questo esempio si usano le proprietà `_xmouse` e `_ymouse` come coordinate *x* e *y* dell'area attiva e i risultati vengono inviati alla finestra Output in modalità di prova filmato. È inoltre possibile impostare un campo di testo sullo stage in modo che visualizzi i risultati o usi i risultati in un'istruzione `if`.

- 5 Scegliere Controlli > Prova filmato e spostare il mouse sul clip filmato per provare la funzione di collisione.

Per rilevare la collisione di due clip filmato:

- 1** Trascinare due clip filmato sullo stage e assegnare loro i nomi di istanza **mcHitArea** e **mcDrag**.
- 2** Creare un campo di testo sullo stage e immettere **status** nella casella Variabile del pannello Opzioni testo.
- 3** Selezionare **mcHitArea** e scegliere Finestra > Azioni.
- 4** Fare doppio clic su **evaluate** nella lista nel riquadro sinistro.
- 5** Immettere il codice seguente nella casella Espressione selezionando gli elementi dalla lista nel riquadro sinistro:

```
_root.status=this.hitTest(_root.mcDrag);
```
- 6** Selezionare l'azione **onClipEvent** nella finestra di script e scegliere **enterFrame** come evento.
- 7** Selezionare **mcDrag** e scegliere Finestra > Azioni.
- 8** Fare doppio clic su **startDrag** nella lista nel riquadro sinistro.
- 9** Selezionare la casella di controllo Blocca mouse al centro.
- 10** Selezionare l'azione **onClipEvent** nella finestra di script e scegliere l'evento **Mouse down**.
- 11** Fare doppio clic su **stopDrag** nella lista nel riquadro sinistro.
- 12** Selezionare l'azione **onClipEvent** nella finestra di script e scegliere l'evento **Mouse up**.
- 13** Scegliere Controlli > Prova filmato e trascinare il clip filmato per provare la funzione di collisione.

Per ulteriori informazioni sul metodo `hitTest`, vedere la voce corrispondente nel capitolo 7 “Dizionario di ActionScript”.

CAPITOLO 4

Uso dei clip filmato

.....

Un clip filmato è un mini filmato Flash che dispone di una linea temporale e di proprietà specifiche. Un simbolo clip filmato della libreria può essere usato più volte in un filmato Flash; ogni uso è detto *istanza* del clip filmato. È possibile annidare i clip filmato uno nell'altro. Per distinguere le diverse istanze, è possibile assegnare un nome a ogni istanza.

Nella linea temporale di un clip filmato è possibile posizionare qualsiasi oggetto, compresi altri clip filmato. Anche i filmati caricati in Flash Player usando `LoadMovie` sono mini filmati Flash. I clip filmato, i filmati caricati e la linea temporale principale di un filmato Flash sono oggetti che dispongono di proprietà e metodi che è possibile gestire tramite ActionScript per creare animazioni complesse e non lineari con un contenuto altamente interattivo.

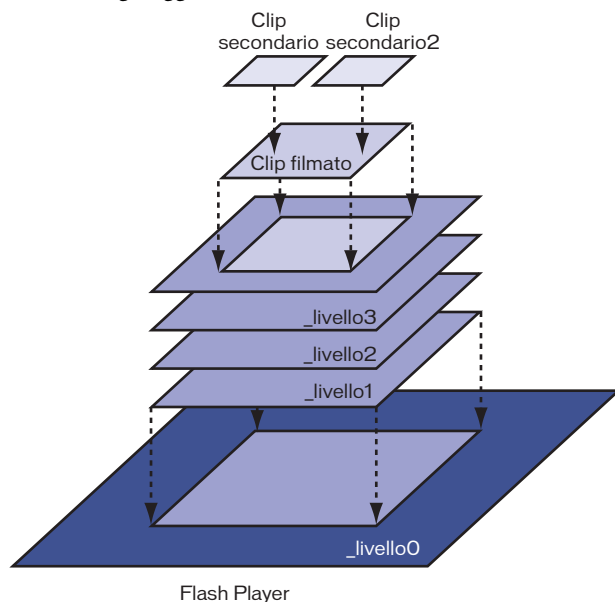
È possibile controllare i clip filmato tramite le azioni e i metodi dell'oggetto `MovieClip`. È possibile associare le azioni e i metodi a fotogrammi o pulsanti di un clip filmato (azioni fotogramma o pulsante) o a un'istanza di clip filmato specifica (azioni clip filmato). Le azioni di un clip filmato consentono di controllare qualsiasi linea temporale di un filmato. Per controllare una linea temporale, è necessario identificarla tramite un percorso target. Il percorso target indica la posizione della linea temporale nel filmato.

È inoltre possibile trasformare un clip filmato in un clip intelligente, ossia un clip filmato dotato di codice ActionScript che è possibile riprogrammare senza usare il pannello Azioni. I clip intelligenti consentono ai programmatori di passare con facilità gli oggetti con codice ActionScript complesso ai creatori di contenuto.

Informazioni sulle linee temporali multiple

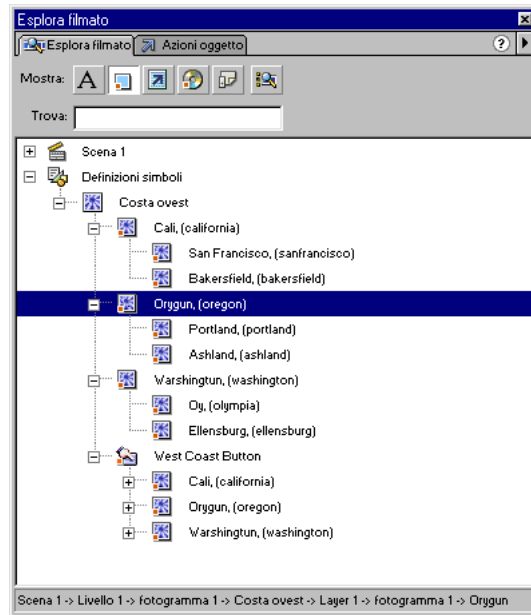
Ogni filmato Flash include una linea temporale principale situata nel livello 0 di Flash Player. È possibile usare l'azione `loadMovie` per caricare altri filmati Flash (file SWF) in Flash Player in qualsiasi livello superiore al livello 0 (ad esempio nel livello 1, 2, 15). Ogni filmato caricato in un livello di Flash Player include una linea temporale.

I filmati Flash di qualsiasi livello possono contenere istanze di clip filmato nelle proprie linee temporali. Ogni istanza di clip filmato include a sua volta una linea temporale e può contenere altri clip filmato che includono a loro volta linee temporali. Le linee temporali dei clip filmato e dei livelli in Flash Player presentano una struttura gerarchica che consente di organizzare e controllare facilmente gli oggetti del filmato.



Gerarchia dei livelli e dei clip filmato in Flash Player.

In Flash, questa gerarchia di livelli e di clip filmato è detta *lista di visualizzazione*. È possibile visualizzare la lista di visualizzazione in Esplora filmato durante la creazione di codice e contenuto in Flash. È inoltre possibile visualizzare la lista di visualizzazione nel Debugger durante la riproduzione del filmato in modalità di prova filmato, nella versione autonoma di Flash Player o in un browser Web.



Esplora filmato visualizza la gerarchia delle linee temporali detta lista di visualizzazione.

Le linee temporali di un filmato Flash sono oggetti che dispongono delle caratteristiche (proprietà) e delle funzionalità (metodi) dell'oggetto predefinito MovieClip. Tra le linee temporali esistono relazioni specifiche a seconda della loro posizione nella lista di visualizzazione. Le linee temporali annidate in altre linee temporali vengono influenzate dalle modifiche apportate alla linea temporale in cui si trovano. Ad esempio, se `portland` è un elemento secondario di `oregon` e si modifica la proprietà `_xscale` di `oregon`, il ridimensionamento verrà applicato anche a `portland`.

Le linee temporali possono inoltre scambiarsi messaggi. Ad esempio, un'azione eseguita sull'ultimo fotogramma di un clip filmato potrebbe richiedere a un altro clip filmato di avviare la riproduzione.

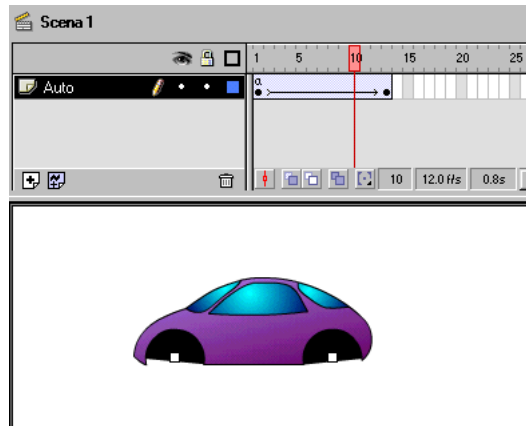
Informazioni sulle relazioni gerarchiche delle linee temporali

Quando si posiziona un'istanza di clip filmato sulla linea temporale di un altro clip filmato, solo uno dei due simboli clip filmato conterrà l'istanza dell'altro clip filmato, ovvero il primo clip filmato sarà l'*elemento secondario*, mentre il secondo clip filmato sarà l'*elemento principale*. La linea temporale principale di un filmato Flash è l'elemento principale di tutti i clip filmato che si trovano sul suo livello.

Le relazioni principale-secondario dei clip filmato sono gerarchiche. Per comprendere questa struttura gerarchica, si consideri la struttura gerarchica di un computer: l'unità disco rigido contiene una directory o cartella principale e le relative sottodirectory. La directory principale è simile alla linea temporale di un filmato Flash, in quanto è l'elemento principale di tutti gli altri elementi. Le sottodirectory sono simili ai clip filmato e possono essere usate per organizzare il contenuto.

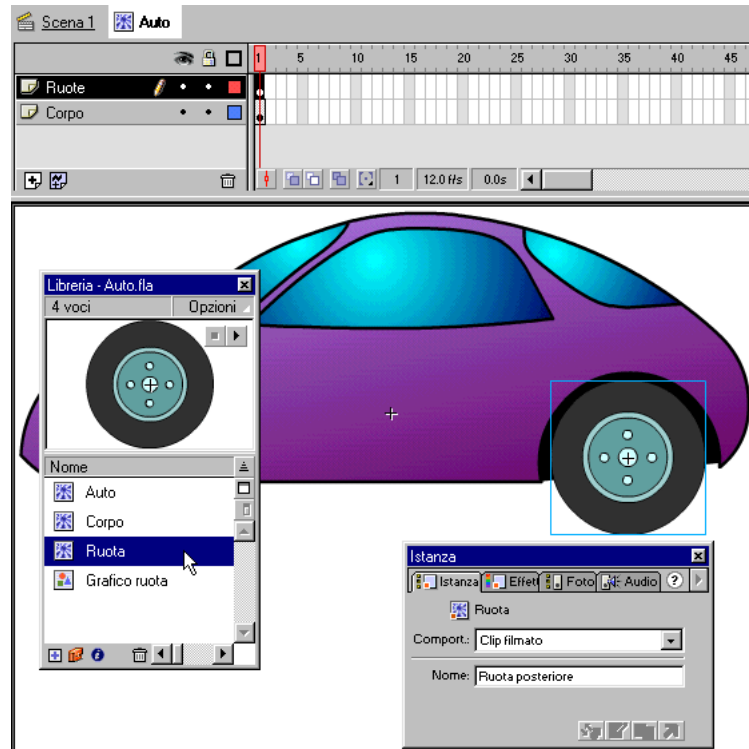
Analogamente, è possibile usare la gerarchia dei clip filmato in Flash per organizzare gli oggetti visivi associati, in genere in modo simile al comportamento degli oggetti nel mondo reale. Le modifiche effettuate in un clip filmato principale vengono apportate anche ai relativi elementi secondari.

Ad esempio, è possibile creare un filmato Flash in cui un'auto si muove attraverso lo stage. A tal fine è possibile usare un simbolo clip filmato per rappresentare l'auto e definire un'interpolazione movimento per muoverla sullo stage.



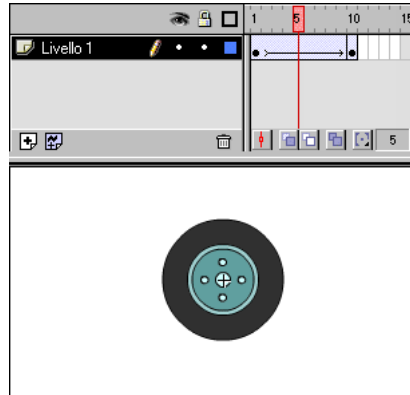
L'interpolazione movimento sposta il clip filmato dell'auto sulla linea temporale principale.

L'auto viene visualizzata di lato, con due ruote visibili. Quando l'auto è in movimento, è desiderabile aggiungere la rotazione delle ruote. È quindi necessario creare un clip filmato per una ruota dell'auto e quindi creare due istanze del clip filmato, denominate `frontWheel` e `backWheel`. Occorre quindi posizionare le ruote sulla linea temporale del clip filmato dell'auto, anziché sulla linea temporale principale. In quanto elementi secondari di `car`, `frontWheel` e `backWheel` vengono influenzati dalle modifiche apportate a `car`. Ciò significa che questi due elementi si muoveranno insieme all'auto mentre questa si muove sullo stage.



Le istanze `frontWheel` e `backWheel` sono posizionate nella linea temporale del clip filmato `car`.

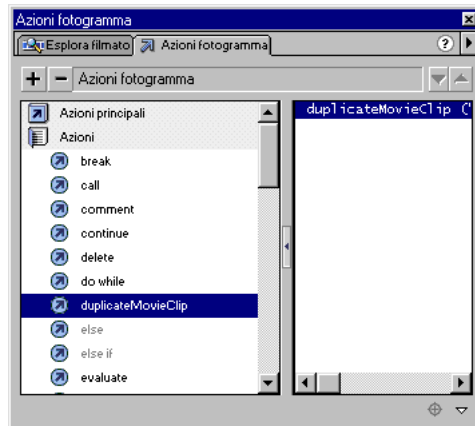
Per muovere le ruote, è possibile definire un'interpolazione movimento che ruoti il simbolo ruota in modo che entrambe le istanze girino. Anche dopo la modifica, le istanze `frontWheel` e `backWheel` saranno comunque influenzate dall'interpolazione del clip filmato principale `car`: le ruote gireranno, ma si muoveranno sullo stage insieme al clip filmato principale `car`.



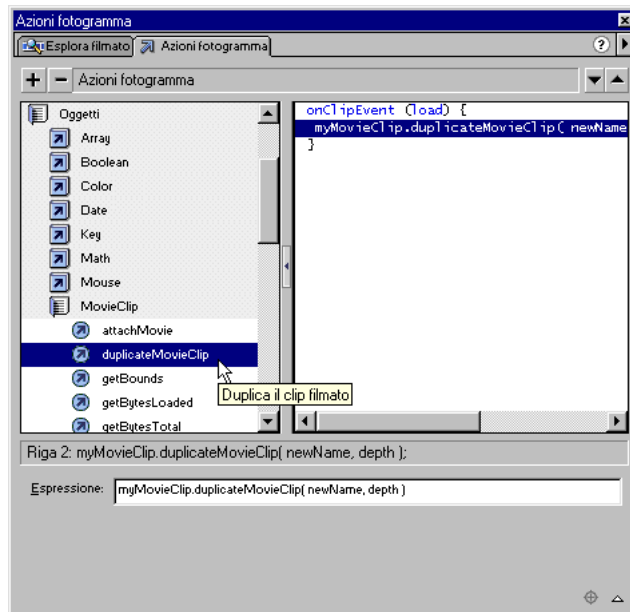
Scambio di messaggi tra linee temporali

È possibile inviare messaggi da una linea temporale all'altra. Una linea temporale contiene l'azione, detta *controller*, e l'altra riceve l'azione, detta *target*. È possibile associare un'azione a un fotogramma o a un pulsante in una linea temporale oppure, se la linea temporale è un clip filmato, allo stesso clip filmato.

Per identificare le linee temporali, è possibile usare le azioni della categoria Azioni oppure i metodi dell'oggetto MovieClip della categoria Oggetti nel pannello Azioni. Ad esempio, è possibile usare l'azione `duplicateMovieClip` per identificare e creare copie di istanze di clip filmato durante la riproduzione di un filmato.



Per identificare una linea temporale è possibile usare le azioni della categoria Azioni.



Per identificare una linea temporale è possibile usare i metodi dell'oggetto MovieClip.

Per eseguire più azioni sullo stesso target, è possibile usare l'azione `with`. Simile all'istruzione JavaScript `with`, l'azione `ActionScript with` è un wrapper che consente di identificare una sola volta la linea temporale e quindi di eseguire una serie di azioni su tale clip, senza dover indirizzare in ogni azione la linea temporale.

Per eseguire più azioni sullo stesso target, è inoltre possibile usare l'azione `tellTarget`.

Per comunicare tra linee temporali diverse, eseguire le operazioni descritte.

- Specificare un nome di istanza per il clip filmato target.

Per assegnare un nome a un'istanza di clip filmato, usare il pannello Istanza (Finestra > Pannelli > Istanza). Le linee temporali caricate nei livelli usano il proprio numero di livello come nome di istanza, ad esempio `_level6`.

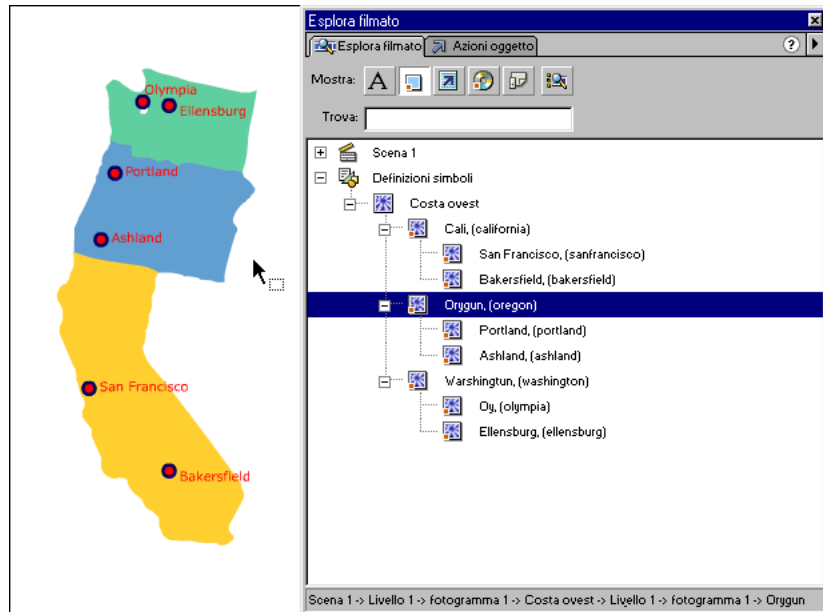
- Immettere il percorso target del nome dell'istanza nel pannello Azioni.

È possibile specificare il percorso target manualmente oppure usare la finestra di dialogo Inserisci percorso target per identificare un clip filmato. Consultare “Specifica dei percorsi target” a pagina 119.

Nota: Durante la riproduzione, è necessario che la linea temporale del clip filmato da identificare si trovi sullo stage.

Informazioni sui percorsi target assoluti o relativi

Il percorso target è l'indirizzo della linea temporale che si desidera identificare. In Flash, la lista di visualizzazione delle linee temporali è simile alla struttura gerarchica dei file e delle cartelle di un server Web.



Esplora filmato visualizza la lista di visualizzazione dei clip filmato in modalità di creazione.

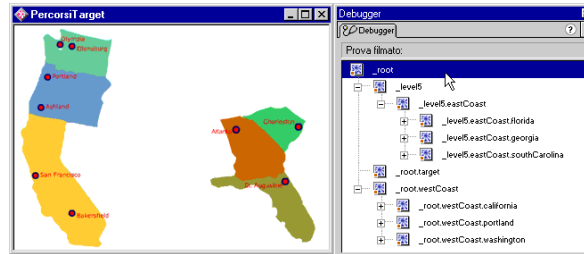
Come in un server Web, l'indirizzo di ogni linea temporale in Flash può essere definito in due modi: tramite un percorso assoluto o un percorso relativo. Il percorso assoluto di un'istanza rimane invariato, indipendentemente dalla linea temporale che chiama l'azione; ad esempio, il percorso assoluto dell'istanza `california` è sempre `_level0.westCoast.california`. Il percorso relativo varia a seconda della posizione da cui viene chiamato; ad esempio, il percorso relativo di `california` chiamato da `sanfrancisco` è `_parent`, ma se è chiamato da `portland`, sarà `_parent._parent.california`.

Nota: Per ulteriori informazioni su Esplora filmato, consultare *Guida all'uso di Flash*.

Il percorso assoluto inizia con il nome del livello in cui viene caricato il filmato e continua nella lista di visualizzazione fino a raggiungere l'istanza target.

Il primo filmato aperto in Flash Player viene caricato nel livello 0. A ogni filmato caricato aggiuntivo è necessario assegnare un numero di livello. Il nome target di un livello è `_levelX`, dove `X` è il numero di livello in cui viene caricato il filmato. Ad esempio, il primo filmato aperto in Flash Player è denominato `_level0`, il filmato caricato nel livello 3 è denominato `_level3`.

Nell'esempio seguente due filmati sono stati caricati nel lettore, `TargetPaths.swf` nel livello 0 e `EastCoast.swf` nel livello 5. I livelli sono indicati nel Debugger, con il livello 0 indicato come `_root`.



Il Debugger visualizza i percorsi assoluti di tutte le linee temporali nella lista di visualizzazione in modalità di prova filmato.

Il percorso assoluto di un'istanza rimane invariato, indipendentemente dal fatto che la chiamata venga eseguita da un'azione di un'istanza nello stesso livello o in un livello diverso. Ad esempio, il percorso assoluto dell'istanza `bakersfield` nel livello 0 espresso nella sintassi del punto è sempre:

```
_level0.california.bakersfield
```

Nella sintassi della barra inclinata, i punti del percorso assoluto sono sostituiti da barre inclinate, come indicato di seguito:

```
_level0/california/bakersfield
```

Per comunicare tra filmati in livelli diversi, è necessario usare il nome del livello nel percorso target. Ad esempio, l'istanza `portland` indirizza l'istanza `atlanta` nel modo seguente:

```
_level5.georgia.atlanta
```

Nella sintassi del punto, è possibile usare l'alias `_root` per fare riferimento alla linea temporale principale del livello corrente. Per la linea temporale principale o `_level0`, l'alias `_root` sostituisce `_level0` quando viene indirizzato da un clip in `_level0`. Per un filmato caricato in `_level5`, `_root` equivale a `_level5` quando viene indirizzato da un clip filmato nel livello 5. Ad esempio, un'azione chiamata dall'istanza `southcarolina` potrebbe usare il seguente percorso assoluto per identificare l'istanza `florida`:

```
_root.eastCoast.florida
```

Nella sintassi della barra inclinata, è possibile usare il carattere / per fare riferimento alla linea temporale principale del livello corrente, come nell'esempio seguente:

```
/eastCoast/florida
```

Nella sintassi del punto, in modalità assoluta o relativa, è possibile usare le stesse regole per il percorso target per identificare una variabile in una linea temporale o la proprietà di un oggetto. Ad esempio, l'istruzione seguente imposta il nome della variabile nel modulo dell'istanza sul valore "Gilbert":

```
_root.form.name = "Gilbert";
```

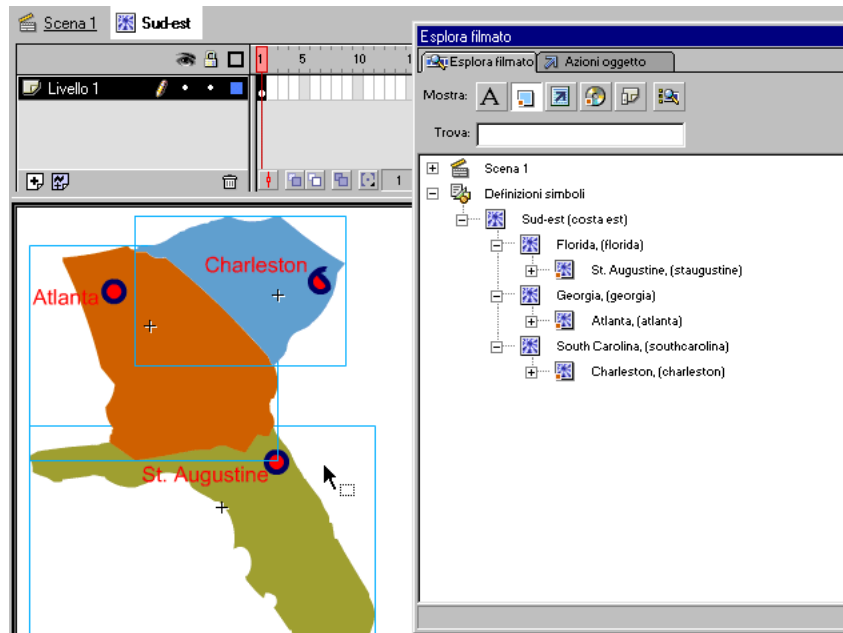
Nella sintassi della barra inclinata, in modalità assoluta o relativa, è possibile identificare una variabile in una linea temporale facendo precedere il nome della variabile da due punti (:), come nell'esempio seguente:

```
/form:name = "Gilbert";
```

Il percorso relativo dipende dalla relazione tra la linea temporale del controller e la linea temporale target. È possibile usare un percorso relativo per riusare le azioni dal momento che la stessa azione può definire linee temporali diverse a seconda della posizione in cui viene inserita. I percorsi relativi possono indirizzare i target solo all'interno del proprio livello di Flash Player e non possono indirizzare filmati caricati in altri livelli. Ad esempio, non è possibile usare un percorso relativo in un'azione posizionata in `_level0` che identifica una linea temporale che si trova in `_level5`.

Nella sintassi del punto, è possibile usare la parola chiave `this` in un percorso target relativo per fare riferimento alla linea temporale corrente. È possibile usare l'alias `_parent` in un percorso target relativo per indicare la linea temporale principale della linea temporale corrente. L'alias `_parent` può essere usato più volte per passare al livello superiore nella gerarchia dei clip filmato all'interno dello stesso livello di Flash Player. Ad esempio, `_parent._parent` indirizza un clip filmato di due livelli verso l'alto nella gerarchia.

Nell'esempio seguente ogni città (charleston, atlanta e staugustine) è un elemento secondario di un'istanza di stato e ogni stato (southcarolina, georgia e florida) è un elemento secondario dell'istanza eastCoast.



Esplora filmato visualizza le relazioni principale-secondario dei clip filmato.

Un'azione della linea temporale dell'istanza charleston potrebbe usare il seguente percorso target per identificare l'istanza southcarolina:

`_parent`

Per identificare l'istanza eastCoast da un'azione in charleston, è possibile usare il percorso relativo seguente:

`_parent._parent`

Nella sintassi della barra inclinata, è possibile usare due puntini (..) per passare al livello superiore nella gerarchia. Per identificare eastCoast da un'azione inclusa in charleston, è possibile usare il percorso seguente:

`../..`

Per identificare l'istanza atlanta da un'azione nella linea temporale di charleston, è possibile usare il percorso relativo seguente nella sintassi del punto:

`_parent._parent.georgia.atlanta`

I percorsi relativi sono utili se si desidera riusare gli script. Ad esempio, è possibile associare uno script a un clip filmato che ingrandisce al 150% il clip filmato del livello superiore, nel modo seguente:

```
onClipEvent (load) {  
    _parent._xscale = 150;  
    _parent._yscale = 150;  
}
```

È quindi possibile riusare lo script posizionandolo nella linea temporale di qualsiasi clip filmato.

Per ulteriori informazioni sull'indirizzamento e la sintassi del punto, consultare “Creazione di script con ActionScript” a pagina 49.

Per ulteriori informazioni sulle sintassi del punto e della barra inclinata, consultare “Uso della sintassi di ActionScript” a pagina 49.

Specifica dei percorsi target

Per controllare un clip filmato o un filmato caricato, è necessario specificare un target tramite un percorso target. Per identificare un clip filmato, è necessario che questo disponga di un nome di istanza. È possibile specificare un target in molti modi diversi:

- Immettere un percorso target usando il pulsante di inserimento nel pannello Azioni e la finestra di dialogo Inserisci percorso target.
- Immettere manualmente il percorso target del clip filmato nello script.
- Creare un'espressione usando un riferimento a un clip filmato o le funzioni predefinite `targetPath` ed `eval`.

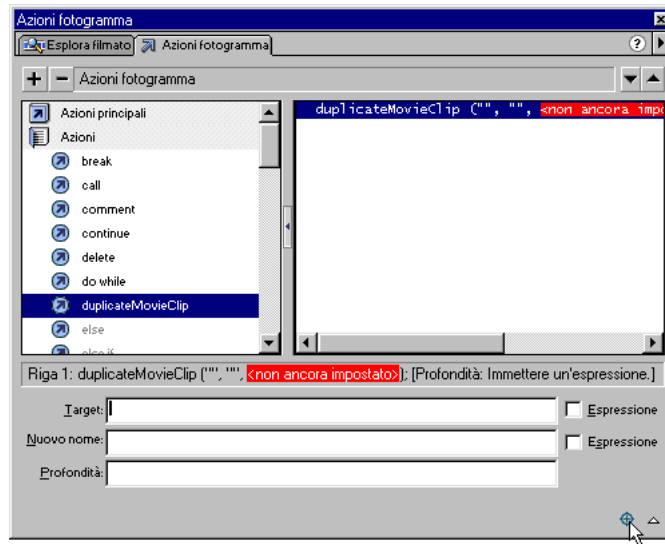
Per inserire un percorso target usando la finestra di dialogo Inserisci percorso target:

- 1 Selezionare il fotogramma, l'istanza di clip filmato o di pulsante alla quale assegnare l'azione.

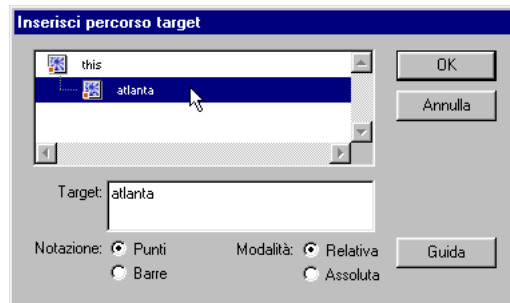
Questa sarà la linea temporale del controller.

- 2 Scegliere Finestra > Azioni per visualizzare il pannello Azioni.
- 3 Nella lista nel riquadro a sinistra, scegliere un'azione dalla categoria Azioni o un metodo dalla categoria MovieClip nella categoria Oggetti.
- 4 Fare clic sul campo Target o nella posizione nello script per inserire il percorso target.

- 5 Fare clic sul pulsante a forma di mirino nell'angolo inferiore destro del pannello Azioni per visualizzare la finestra di dialogo Inserisci percorso target.



- 6 Nella finestra di dialogo Inserisci percorso target, scegliere una notazione tra Punti (impostazione predefinita) o Barre.



- 7 Scegliere Assoluta o Relativa come modalità del percorso target.
Consultare “Informazioni sui percorsi target assoluti o relativi” a pagina 115.
- 8 Specificare il target eseguendo una delle operazioni descritte:
- Selezionare un clip filmato nella lista di visualizzazione.
 - Immettere il target manualmente nel campo Target usando un percorso assoluto o relativo con la sintassi del punto.
- 9 Fare clic su OK.

Per inserire un percorso target manualmente:

Eseguire i punti da 1 a 4 indicati in precedenza e immettere un percorso target assoluto o relativo nel pannello Azioni.

Per usare un'espressione come percorso target:

- 1 Eseguire i punti da 1 a 4 indicati in precedenza.
- 2 Eseguire una delle operazioni descritte:
 - Immettere manualmente un riferimento come percorso target. Il riferimento viene valutato per determinare il percorso target. È possibile usare un riferimento come parametro dell'azione `with`. Nell'esempio seguente la variabile `index` viene valutata e moltiplicata per 2. Il valore risultante viene usato come nome del clip filmato nell'istanza `Block` di cui eseguire la riproduzione:

```
with (Board.Block[index*2]) {  
    play();  
}
```

- Nella categoria Funzioni nella lista nel riquadro a sinistra, scegliere la funzione `targetPath`.

La funzione `targetPath` converte un riferimento a un clip filmato in una stringa che può essere usata dalle azioni quali `tellTarget`.

Nell'esempio seguente la funzione `targetPath` converte il riferimento `Board.Block[index*2+1]` in una stringa:

```
tellTarget (targetPath (Board.Block[index*2+1])) {  
    play();  
}
```

L'esempio precedente equivale alla seguente sintassi della barra inclinata:

```
tellTarget ("Board/Block:" + index*2+1)) {  
    play();  
}
```

- Nella categoria Funzioni nella lista nel riquadro a sinistra, scegliere la funzione `eval`.

La funzione `eval` converte una stringa in un riferimento a un clip filmato che può essere usato come percorso target dalle azioni quali `with`.

Il seguente script valuta la variabile `i`, la aggiunge alla stringa `"cat"` e assegna il valore risultante alla variabile `x`. La variabile `x` diventa quindi un riferimento a un'istanza di clip filmato e può chiamare i metodi dell'oggetto `MovieClip`, come nell'esempio seguente:

```
x = eval ("cat" + i)
x.play()
```

È inoltre possibile usare la funzione `eval` per chiamare i metodi direttamente, come nell'esempio seguente:

```
eval ("cat" + i).play()
```

Uso di azioni e metodi per il controllo delle linee temporali

Alcune azioni e metodi dell'oggetto `MovieClip` consentono di *identificare* o eseguire operazioni su un clip filmato o un livello caricato. Ad esempio, l'azione `setProperty` imposta una proprietà (ad esempio `_width`) di una linea temporale su un valore (ad esempio `100`). Alcuni metodi dell'oggetto `MovieClip` duplicano la funzione di tutte le azioni che identificano le linee temporali. Sono inoltre disponibili metodi aggiuntivi, quali `hitTest` e `swapDepths`. Indipendentemente dalla scelta di un'azione o di un metodo, è necessario che la linea temporale target sia caricata in Flash Player quando viene chiamata l'azione o il metodo.

Le seguenti azioni sono indirizzate a clip filmato: `loadMovie`, `unloadMovie`, `setProperty`, `startDrag`, `duplicateMovieClip` e `removeMovieClip`. Per usare queste azioni, è necessario immettere un percorso target nel parametro Target dell'azione per indicarne il destinatario. Alcune di queste azioni sono in grado di indirizzare sia clip filmato che livelli, mentre altre sono in grado di indirizzare solo clip filmato.

I seguenti metodi dell'oggetto `MovieClip` sono in grado di controllare i clip filmato o i livelli caricati e non esistono azioni equivalenti: `attachMovie`, `getBounds`, `getBytesLoaded`, `getBytesTotal`, `globalToLocal`, `localToGlobal`, `hitTest` e `swapDepths`.

Se un'azione e un metodo offrono funzioni simili, è possibile scegliere di controllare i clip filmato tramite l'azione o il metodo. La scelta dipende dalle preferenze e dalla familiarità dell'utente nella creazione di script con `ActionScript`.

Per ulteriori informazioni sui metodi dell'oggetto `MovieClip` e sulle singole azioni, consultare il capitolo 7 “Dizionario di `ActionScript`” a pagina 167.

Informazioni sulle differenze tra metodi e azioni

Per usare un metodo, è possibile chiamarlo usando il percorso target del nome dell'istanza seguito da un punto e quindi dal nome del metodo e dagli argomenti, come nelle istruzioni seguenti:

```
myMovieClip.play();
parentClip.childClip.gotoAndPlay(3);
```

Nella prima istruzione il metodo `play` provoca la riproduzione dell'istanza `myMovieClip`. Nella seconda istruzione il metodo `gotoAndPlay` porta l'indicatore di riproduzione di `childClip` (un elemento secondario dell'istanza `parentClip`) sul fotogramma 3 e ne avvia la riproduzione.

Le azioni che controllano una linea temporale hanno un parametro `Target` che specifica il percorso target. Ad esempio, nello script seguente l'azione `startDrag` indirizza l'istanza `customCursor` e la rende mobile:

```
on(press){
    startDrag("customCursor");
}
```

Quando si usa un metodo, questo viene chiamato alla fine del percorso target. Ad esempio, l'istruzione seguente esegue la stessa funzione `startDrag`:

```
customCursor.startDrag();
```

Le istruzioni create usando i metodi dell'oggetto `MovieClip` sono in genere più brevi, dal momento che non richiedono l'azione `tellTarget`. Si sconsiglia l'uso dell'azione `tellTarget` in quanto non è compatibile con lo standard ECMA-262.

Ad esempio, per indicare al clip filmato `myMovieClip` di iniziare la riproduzione usando i metodi dell'oggetto `MovieClip`, è necessario usare il codice seguente:

```
myMovieClip.play();
```

Il codice seguente restituisce gli stessi risultati usando l'azione `tellTarget`:

```
tellTarget ("myMovieClip") {
    play();
}
```

Uso di più metodi o azioni sulla stessa linea temporale

È possibile usare l'azione `with` per indirizzare una sola volta un clip filmato e quindi eseguire una serie di azioni su tale clip. L'azione `with` è applicabile a tutti gli oggetti ActionScript (ad esempio Array, Color e Sound), non solo per i clip filmato. L'azione `tellTarget` è simile all'azione `with`. Tuttavia si sconsiglia l'uso dell'azione `tellTarget` in quanto non è applicabile a tutti gli oggetti ActionScript e non è compatibile con lo standard ECMA-262.

L'azione `with` richiede un oggetto come un parametro. L'oggetto specificato viene aggiunto alla fine del percorso target corrente. Tutte le azioni annidate in un'azione `with` vengono eseguite rispetto al nuovo percorso target o *area di validità*. Ad esempio, nello script seguente nella linea temporale principale, all'azione `with` viene passato l'oggetto `donut.hole` per modificare le proprietà di `hole`:

```
with (donut.hole){  
    _alpha = 20;  
    _xscale = 150;  
    _yscale = 150;  
}
```

È come se le istruzioni all'interno dell'azione `with` fossero chiamate dalla linea temporale dell'istanza `hole`.

Nell'esempio seguente è evidente il vantaggio di usare l'azione `with` e i metodi dell'oggetto `MovieClip` per eseguire diverse istruzioni:

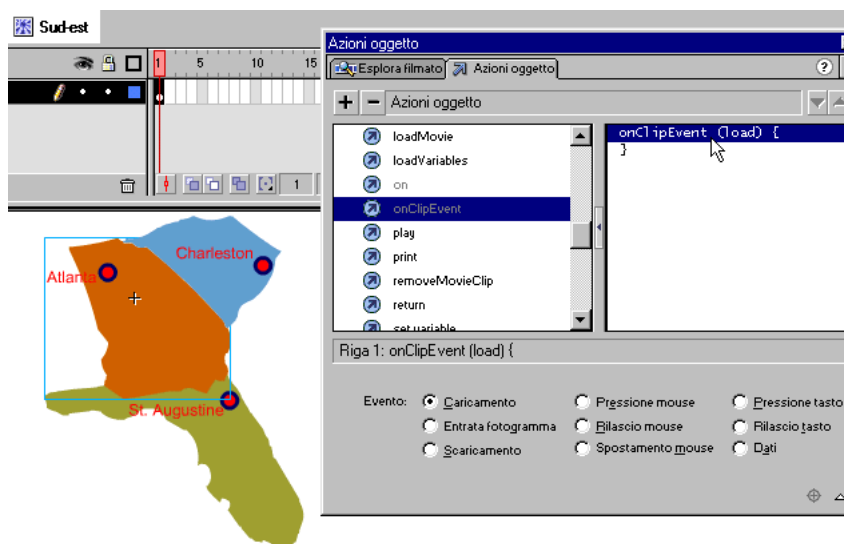
```
with (myMovieClip) {  
    _x -= 10;  
    _y += 10;  
    gotoAndPlay(3);  
}
```

Per ulteriori informazioni sull'azione `tellTarget`, consultare *Guida all'uso di Flash*.

Assegnazione di un'azione o di un metodo

È possibile assegnare azioni e metodi a un pulsante o a un fotogramma in una linea temporale o a un'istanza di clip filmato.

Per assegnare un'azione o un metodo a un'istanza di clip filmato, è necessario usare un gestore `onClipEvent`. Tutte le azioni associate all'istanza sono annidate in un gestore `onClipEvent` e vengono eseguite dopo l'attivazione del gestore. L'azione `onClipEvent` viene attivata da eventi della linea temporale (quale il caricamento di un filmato) o da eventi causati dall'utente (quali il clic con il mouse o la pressione di tasti). Ad esempio, `onClipEvent(mouseMove)` attiva un'azione ogni volta che l'utente muove il mouse.



L'azione `onClipEvent` è assegnata a un'istanza sullo stage. Gli eventi `onClipEvent` sono elencati nel riquadro dei parametri del pannello Azioni.

Caricamento e scaricamento di filmati aggiuntivi

È possibile usare l'azione o il metodo `loadMovie` per riprodurre filmati aggiuntivi senza chiudere Flash Player o per passare da un filmato all'altro senza caricare un'altra pagina HTML. `loadMovie` consente inoltre di inviare variabili a uno script CGI che genera un file SWF come output CGI. Quando si carica un filmato, è possibile specificare come target un livello o un clip filmato in cui caricare il filmato.

L'azione e il metodo `unloadMovie` rimuovono un filmato caricato in precedenza con `loadMovie`. Lo scaricamento esplicito dei filmati con `unloadMovie` assicura il passaggio regolare da un filmato all'altro e può ridurre la quantità di memoria richiesta da Flash Player. Usare l'azione `loadMovie` per eseguire una delle operazioni descritte:

- Riproduzione di una sequenza di pubblicità di intestazione costituite da file SWF tramite l'inserimento di un'azione `loadMovie` alla fine di ogni file SWF per caricare il filmato successivo
- Sviluppo di un'interfaccia a opzioni multiple che consente all'utente di scegliere tra diversi file SWF
- Creazione di un'interfaccia di navigazione con controlli di navigazione nel livello 0 che consentono di caricare altri livelli. Il caricamento di livelli genera passaggi più fluidi rispetto al caricamento di nuove pagine HTML in un browser.

Modifica della posizione e dell'aspetto di un clip filmato

Per modificare le proprietà di un clip filmato durante la riproduzione, è possibile usare l'azione `setProperty` o creare un'istruzione che assegna un valore a una proprietà. Se si carica un filmato in un target, il filmato caricato eredita le proprietà del clip filmato identificato. Dopo aver caricato il filmato, è possibile modificarne tali proprietà.

I valori di alcune proprietà, dette proprietà *di sola lettura*, possono essere letti ma non impostati. È possibile creare istruzioni per impostare le proprietà, a condizione che non siano di sola lettura. L'istruzione seguente imposta la proprietà `_alpha` dell'istanza di clip filmato `wheel`, elemento secondario dell'istanza `car`:

```
car.wheel._alpha = 50;
```

È inoltre possibile creare istruzioni che estraggono il valore di una proprietà di un clip filmato. Ad esempio, l'istruzione seguente estrae il valore della proprietà `_xmouse` nella linea temporale principale e imposta la proprietà `_x` dell'istanza `customCursor` su tale valore:

```
onClipEvent(enterFrame){  
    customCursor._x = _root._xmouse;  
}
```

È inoltre possibile usare la funzione `getProperty` per recuperare le proprietà dei clip filmato.

Le proprietà `_x`, `_y`, `_rotation`, `_xscale`, `_yscale`, `_height`, `_width`, `_alpha` e `_visible` sono influenzate dalle trasformazioni effettuate nell'elemento principale del clip filmato e modificano il clip filmato e tutti gli elementi secondari del clip. Le proprietà `_focusrect`, `_highquality`, `_quality` e `_soundbuftime` sono di tipo globale e sono relative solo alla linea temporale di livello 0. Tutte le altre proprietà sono relative a ogni clip filmato o livello caricato. La tabella seguente elenca tutte le proprietà dei clip filmato:

Proprietà			
<code>_alpha</code>	<code>_highquality</code>	<code>_totalframes</code>	<code>_xscale</code>
<code>_currentframe</code>	<code>_name</code>	<code>_url</code>	<code>_y</code>
<code>_droptarget</code>	<code>_quality</code>	<code>_visible</code>	<code>_ymouse</code>
<code>_focusrect</code>	<code>_rotation</code>	<code>_width</code>	<code>_yscale</code>
<code>_framesloaded</code>	<code>_soundbuftime</code>	<code>_x</code>	
<code>_height</code>	<code>_target</code>	<code>_xmouse</code>	

Trascinamento di clip filmato

L'azione o il metodo `startDrag` consente di trascinare un clip filmato durante la riproduzione di un filmato. Ad esempio, è possibile consentire il trascinamento di un clip filmato nel caso di giochi, funzioni di trascinamento della selezione, interfacce personalizzabili, barre di scorrimento e cursori.

È possibile trascinare un clip filmato fino a quando viene interrotto esplicitamente con `stopDrag` o fino a quando viene identificato un altro clip filmato con `startDrag`. È possibile trascinare un solo clip filmato alla volta.

Per creare funzionalità di trascinamento della selezione più complesse, è possibile valutare la proprietà `_droptarget` del clip filmato che viene trascinato. Ad esempio, è possibile verificare nella proprietà `_droptarget` se il filmato è stato trascinato in un clip filmato specifico (quale il clip filmato del “cestino”) e quindi attivare un'altra azione. Consultare “Uso di istruzioni if” a pagina 71 e “Uso di operatori per la gestione dei valori nelle espressioni” a pagina 62.

Duplicazione ed eliminazione di clip filmato

È possibile creare o eliminare istanze di clip filmato durante la riproduzione di un filmato usando, rispettivamente, `duplicateMovieClip` o `removeMovieClip`. L'azione e il metodo `duplicateMovieClip` creano dinamicamente una nuova istanza del clip filmato assegnando ad essa un nuovo nome e una profondità. I clip filmato duplicati iniziano sempre dal fotogramma 1 anche se il clip filmato originale si trovava in un fotogramma diverso al momento della duplicazione e sono sempre sovrapposti a tutti i clip filmato predefiniti posizionati nella linea temporale. Le variabili non vengono copiate nel clip filmato duplicato.

Per eliminare un clip filmato creato con `duplicateMovieClip`, usare `removeMovieClip`. I clip filmato duplicati vengono eliminati anche quando viene eliminato il clip filmato principale.

Associazione di clip filmato

È possibile recuperare una copia di un clip filmato da una libreria e riprodurlo come parte di un filmato usando il metodo `attachMovie`. Questo metodo carica un altro clip filmato nel clip filmato corrente e lo riproduce durante l'esecuzione del filmato.

Per usare il metodo `attachMovie`, è necessario specificare un nome univoco per il clip filmato che viene associato nella finestra di dialogo Proprietà di concatenamento del simbolo.

Per assegnare un nome a un clip filmato per la condivisione:

- 1 Nella libreria dei filmati, selezionare il clip filmato da associare.
- 2 Nella finestra Libreria, scegliere Concatenamento dal menu Opzioni.
- 3 Per Concatenamento, scegliere Esporta questo simbolo.
- 4 Nella finestra di dialogo Proprietà di concatenamento del simbolo, in Identificatore immettere un nome per il clip filmato. È necessario che il nome sia diverso dal nome del simbolo contenuto nella libreria.
- 5 Fare clic su OK.

Per assegnare un clip filmato a un altro clip filmato:

- 1 Nel pannello Azioni, specificare il target al quale si desidera associare un clip filmato.
- 2 Nella lista nel riquadro a sinistra, selezionare l'oggetto `MovieClip` e quindi selezionare il metodo `attachMovie`.

3 Impostare i seguenti argomenti:

- Per `idName`, specificare il nome dell'identificatore immesso nella finestra di dialogo Proprietà di concatenamento del simbolo.
- Per `newName`, immettere un nome di istanza per il clip associato in modo che sia possibile identificarlo.
- Per `depth`, immettere il livello nel quale il filmato duplicato verrà associato al clip filmato. Ogni filmato associato dispone di un proprio ordine di impilamento, con il livello 0 come livello del filmato di origine. I clip filmato associati sono sempre sovrapposti al clip filmato originale.

Ad esempio:

```
myMovieClip.attachMovie("calif", "california", 10 );
```

Creazione di clip intelligenti

I clip intelligenti sono clip filmato con parametri clip definiti che è possibile modificare. Questi parametri vengono quindi passati alle azioni del clip intelligente che modificano il comportamento del clip.

Per creare un clip intelligente, assegnare parametri clip a un simbolo clip filmato della libreria. Nei clip intelligenti è possibile creare istruzioni ActionScript che eseguono operazioni sui parametri clip, in modo analogo all'uso degli argomenti in una definizione di funzione. È possibile selezionare un'istanza di clip intelligente sullo stage e modificare i valori dei parametri nel pannello Parametri clip. Durante la riproduzione, i valori impostati nel pannello vengono inviati al clip intelligente prima dell'esecuzione delle azioni nel filmato.

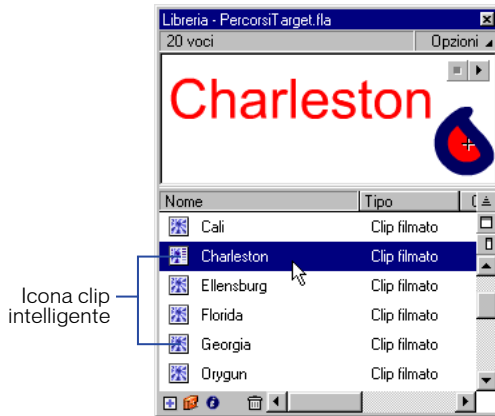
I clip intelligenti consentono ai programmatori di passare elementi Flash complessi ai creatori di contenuto. I programmatori possono creare azioni nei clip intelligenti con variabili che controllano il clip e il filmato. I creatori di contenuti possono quindi modificare i valori di tali variabili nel pannello Parametri clip senza dover aprire il pannello Azioni.

È possibile usare i clip intelligenti per creare elementi di interfaccia, quali pulsanti di scelta, menu a comparsa, descrizioni comandi, sondaggi, giochi e mondi immaginari. Un clip intelligente è un clip filmato che può essere riusato in modo diverso senza dover modificare i relativi script.

È inoltre possibile creare un'interfaccia personalizzata in Flash per il pannello Parametri clip per facilitare il compito dei creatori di contenuto responsabili della personalizzazione del clip.

Definizione dei parametri clip

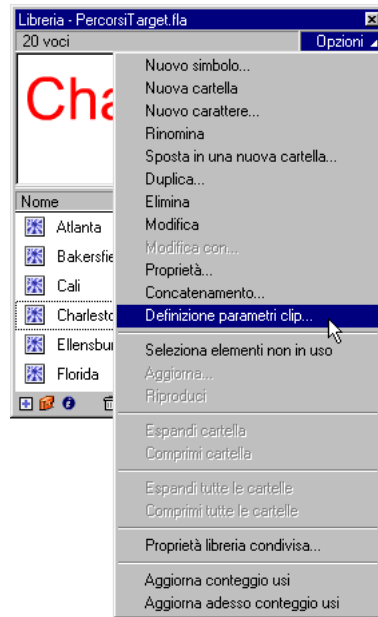
I parametri clip sono dati che vengono passati a un clip filmato quando viene caricato in un filmato. È possibile definire i parametri clip durante la creazione del codice del filmato. È possibile quindi usare questi parametri nelle azioni per modificare l'aspetto e il comportamento del clip intelligente durante la riproduzione del filmato. Un'icona speciale nella finestra Libreria indica un clip filmato con parametri clip definiti.



Per definire i parametri clip per un clip filmato:

- 1 Selezionare un simbolo clip filmato nella libreria dei filmati ed eseguire una delle operazioni descritte per visualizzare la finestra di dialogo Parametri clip:
 - In Windows, fare clic con il pulsante destro del mouse o, in Macintosh, premere Ctrl e scegliere Definizione parametri clip dal menu di scelta rapida.

- Scegliere Definizione parametri clip dal menu Opzioni nell'angolo superiore destro della finestra Libreria.



2 Usare i controlli della finestra di dialogo Parametri clip nel modo seguente:

- Fare clic sul pulsante + per aggiungere una nuova coppia nome/valore o parametri aggiuntivi per una coppia nome/valore selezionata.
- Fare clic sul pulsante - per eliminare una coppia nome/valore.
- Usare i pulsanti freccia per modificare l'ordine dei parametri nella lista.
- Selezionare un campo facendo doppio clic su di esso e quindi immettere un valore.

3 Sotto Nome immettere un identificatore univoco per il parametro.

4 Sotto Tipo scegliere dal menu a comparsa il tipo di dati che saranno contenuti nel parametro:

- Selezionare Predefinito per usare un valore stringa o numerico.
- Selezionare Matrice per usare una lista dinamica di elementi che può ingrandirsi o ridursi.
- Selezionare Oggetto per dichiarare numerosi elementi correlati con nomi e valori, ad esempio un oggetto punto con coordinate x e y .
- Selezionare Lista per limitare la selezione a diverse scelte, quali `true` o `false` oppure Red, Green o Blue.

- 5 Sotto Valore selezionare dal menu a comparsa il valore predefinito che sarà contenuto nel parametro.
- 6 Se si desidera usare un'interfaccia personalizzata per il pannello Parametri clip, eseguire una delle operazioni descritte:
 - Immettere un percorso relativo per il file SWF dell'interfaccia personalizzata nel campo Collegamento all'interfaccia utente personalizzata.
 - Fare clic sulla cartella Collegamento all'interfaccia utente personalizzata e individuare il file SWF dell'interfaccia personalizzata.

Consultare “Creazione di un'interfaccia personalizzata” a pagina 134.

- 7 In Descrizione immettere la descrizione della funzione di ogni parametro che verrà visualizzata nel pannello Parametri clip.

Includere tutte le informazioni che si desidera fornire agli utenti che useranno il clip intelligente. Ad esempio, una spiegazione dei metodi che sono stati definiti.

- 8 Scegliere Blocca in istanza per impedire agli utenti di rinominare i parametri nel pannello Parametri clip.

Si consiglia di mantenere bloccati i nomi di parametro.

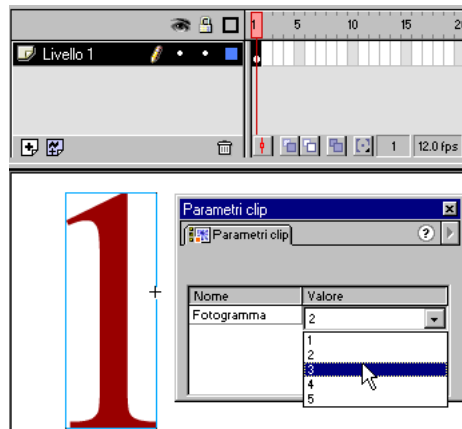
- 9 Fare clic su OK.

Impostazione dei parametri clip

In un clip intelligente è possibile creare azioni che usano i parametri definiti per modificare il comportamento del clip intelligente. Si consideri un semplice esempio. Se si definisce un parametro clip con il nome `Frame`, è possibile creare lo script seguente nel clip intelligente che usa il parametro `Frame`:

```
onClipEvent(load){  
    gotoAndStop(Frame);  
}
```

È quindi possibile selezionare il clip intelligente sullo stage e impostare il valore per il parametro *Frame* nel pannello Parametri clip per cambiare il fotogramma da riprodurre.



Per impostare i parametri clip di un clip intelligente:

- 1 Selezionare un'istanza di clip intelligente sullo stage.

I clip intelligenti sono clip filmato e, di conseguenza, solo il primo fotogramma verrà visualizzato in modalità di creazione.

- 2 Scegliere Finestra > Pannelli > Parametri clip per visualizzare il pannello Parametri clip.

- 3 Nel pannello Parametri clip, eseguire una delle operazioni descritte:

- Fare doppio clic sul campo Valore per selezionarlo e immettere un valore per ogni parametro.

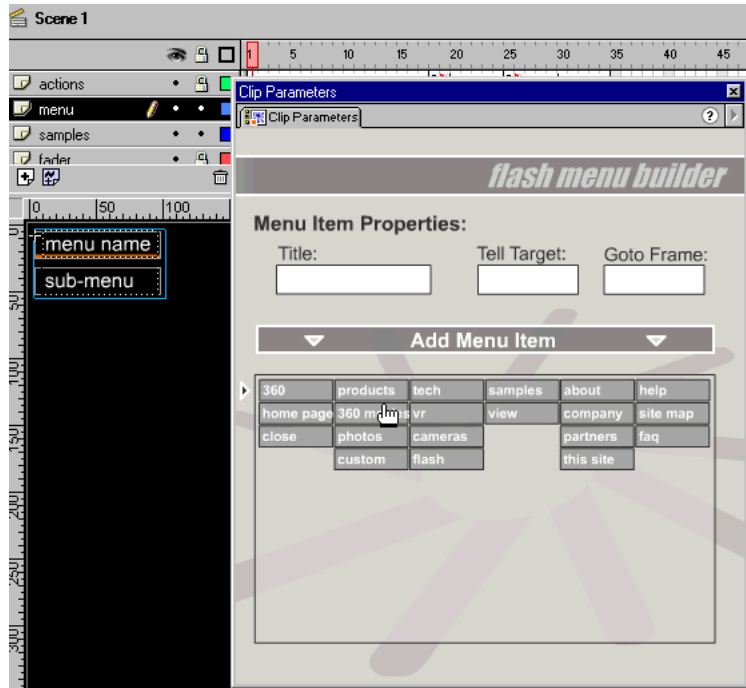
Se il parametro è stato definito come Lista, verrà visualizzato un menu a comparsa.

- Se è stata definita un'interfaccia personalizzata, usare gli elementi di interfaccia forniti.

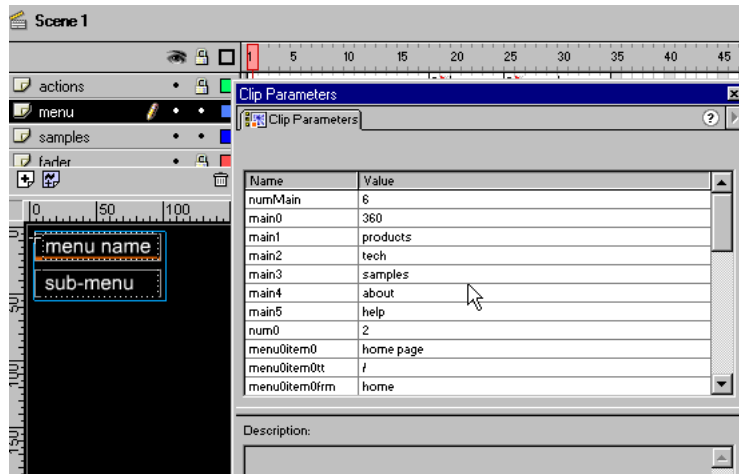
- 4 Scegliere Controlli > Prova filmato per verificare la modifica nel comportamento del clip intelligente.

Creazione di un'interfaccia personalizzata

Un'interfaccia personalizzata è un filmato Flash che consente di immettere valori da passare al clip intelligente. L'interfaccia personalizzata sostituisce l'interfaccia del pannello Parametri clip.



Pannello Parametri clip con un filmato di interfaccia personalizzata.



Lo stesso clip intelligente senza l'interfaccia personalizzata nel pannello Parametri clip.

Tutti i valori immessi usando un'interfaccia personalizzata vengono passati dal pannello Parametri clip al clip intelligente attraverso un intermediario, o clip filmato di scambio, nell'interfaccia personalizzata. Il nome di istanza del clip filmato di scambio deve essere `xch`. Se nella finestra di dialogo Definizione parametri clip è selezionata un'interfaccia personalizzata, l'istanza di clip intelligente passa i parametri definiti al clip filmato `xch` e tutti i nuovi valori immessi nell'interfaccia personalizzata vengono copiati nel clip filmato `xch` e passati al clip intelligente.

Il clip `xch` deve essere posizionato nella linea temporale principale del filmato di interfaccia e deve essere sempre caricato. Il clip filmato `xch` deve contenere solo i valori da passare al clip intelligente e non può contenere oggetti grafici, altri clip filmato o istruzioni ActionScript, in quanto è semplicemente un contenitore attraverso il quale vengono passati i valori. Attraverso il clip `xch` è possibile trasferire oggetti di primo livello, quali Array e Object. Tuttavia, non è possibile passare oggetti Array o Object annidati.

Per creare un'interfaccia personalizzata per un clip intelligente:

- 1** Scegliere File > Nuovo per creare un nuovo filmato Flash.
- 2** Scegliere Inserisci > Nuovo simbolo per creare il clip filmato di scambio.
- 3** Creare un nuovo livello denominato “Clip di scambio”.
- 4** Con il nuovo livello “Clip di scambio” selezionato, trascinare il clip filmato di scambio dalla finestra Libreria sullo stage nel fotogramma 1.
- 5** Selezionare il clip filmato di scambio sullo stage, scegliere Finestra > Pannelli > Istanza e immettere il nome `xch`.
- 6** Creare gli elementi di interfaccia che consentono di impostare i parametri clip. Ad esempio, menu a comparsa, pulsanti di scelta o voci di menu trascinabili.
- 7** Per copiare i valori delle variabili e degli oggetti nell'istanza `xch`, usare l'azione `set variable`.

Ad esempio, se un pulsante viene usato come un elemento di interfaccia, al pulsante potrebbe essere associata un'azione che imposta il valore della variabile `vertical` e lo passa a `xch`, come nell'esempio seguente:

```
on (release){  
    _root.xch.vertical = true;  
}
```

- 8** Esportare il filmato come file SWF.

Per usare il file SWF dell'interfaccia personalizzata con un clip intelligente, è necessario collegarlo nella finestra di dialogo Definizione parametri clip nella libreria che contiene il clip intelligente. Si consiglia di salvare il file SWF nella stessa directory del file FLA contenente il clip intelligente. Se il clip intelligente viene riusato in un altro file o viene passato a un altro sviluppatore, è necessario che il clip intelligente e il file SWF dell'interfaccia personalizzata vengano mantenuti negli stessi percorsi relativi.

CAPITOLO 5

Integrazione di Flash nelle applicazioni Web

I filmati Flash possono inviare informazioni a e caricare informazioni da file remoti. Per inviare e caricare variabili, usare l'azione `loadVariables` o `getURL`. Per caricare un filmato Flash Player da una posizione remota, usare l'azione `loadMovie`. Per inviare e caricare dati XML, usare l'oggetto XML o XMLSocket. È possibile strutturare i dati XML usando i metodi dell'oggetto XML predefinito.

È inoltre possibile creare moduli Flash composti da tipici elementi di interfaccia, quali campi di testo e menu a comparsa, per raccogliere i dati da inviare a un'applicazione lato server.

Per estendere Flash in modo che possa inviare e ricevere messaggi dall'ambiente host del filmato, ad esempio Flash Player o una funzione JavaScript in un browser Web, è possibile usare `fscommand` e i metodi di Flash Player.

Invio di variabili a e caricamento di variabili da un file remoto

Un filmato Flash è una finestra che consente di acquisire e visualizzare informazioni, quasi come una pagina HTML. Diversamente dalle pagine HTML, i filmati Flash possono rimanere caricati nel browser ed essere continuamente aggiornati in base alle nuove informazioni senza dovere essere riaggiornati. È possibile usare le azioni Flash e i metodi degli oggetti per inviare informazioni a e ricevere informazioni da script lato server, file di testo e file XML.

Gli script lato server possono richiedere informazioni specifiche a un database e passarle dal database a un filmato Flash e viceversa. Gli script lato server possono essere creati in numerosi linguaggi diversi, Perl, ASP (Microsoft Active Server Pages) e PHP sono tra i più comuni.

La memorizzazione di informazioni in un database e il conseguente accesso consentono di creare un contenuto dinamico e personalizzato per il filmato. Ad esempio, è possibile creare una bacheca di messaggi, profili personali per gli utenti o un carrello della spesa contenente ciò che l'utente ha acquistato in modo che sia possibile determinare le preferenze dell'utente.

È possibile usare diverse azioni ActionScript e metodi degli oggetti per passare informazioni a e da un filmato. Ogni azione e metodo usa un protocollo per trasferire le informazioni e richiede che queste siano formattate in un certo modo.

Le seguenti azioni usano il protocollo HTTP o HTTPS per inviare informazioni in formato con codifica URL: `getURL`, `loadVariables`, `loadMovie`.

I seguenti metodi usano il protocollo HTTP o HTTPS per inviare informazioni in formato XML: `XML.send`, `XML.load`, `XML.sendAndLoad`.

I seguenti metodi creano e usano una connessione tramite socket TCP/IP per inviare informazioni in formato XML: `XMLSocket.connect`, `XMLSocket.send`.

Informazioni sulla sicurezza

Durante la riproduzione di un filmato Flash in un browser Web è possibile caricare dati nel filmato solo da un file che si trova su un server nello stesso sottodominio. In questo modo si impedisce che i filmati Flash possano scaricare informazioni dai server di altri utenti.

Per determinare il sottodominio di un URL che consiste di uno o due componenti, usare l'intero dominio.

Dominio	Sottodominio
<code>http://macromedia</code>	<code>macromedia</code>
<code>http://macromedia.com</code>	<code>macromedia.com</code>

Per determinare il sottodominio di un URL che consiste di più di due componenti, rimuovere l'ultimo livello.

Dominio	Sottodominio
<code>http://x.y.macromedia.com</code>	<code>y.macromedia.com</code>
<code>http://www.macromedia.com</code>	<code>macromedia.com</code>

Il seguente diagramma mostra come Flash Player determina se consentire o meno una richiesta HTTP:

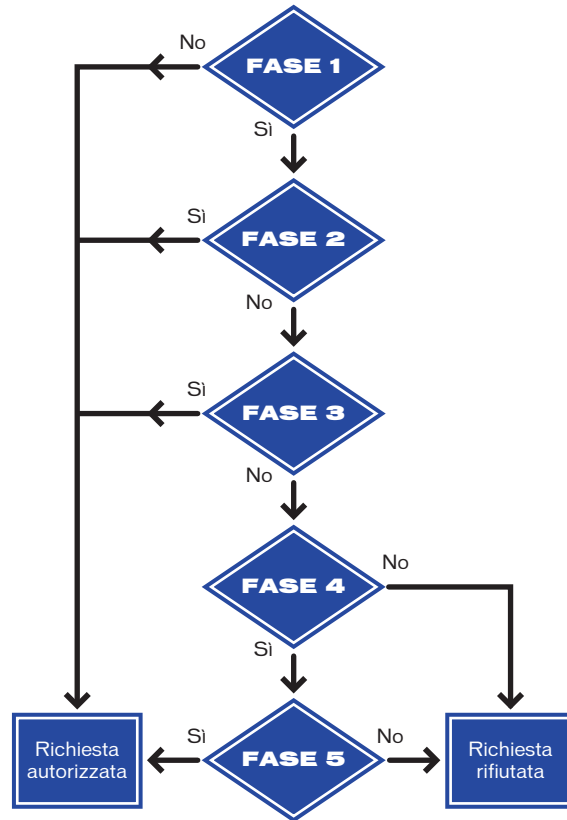
FASE 1
È questa richiesta per:
loadVariables,
xml.load,
xml.sendAndLoad,
o
xmlsocket.connect?

FASE 2
È questa
richiesta per un
URL relativo?

FASE 3
Il filmato richiesto
viene caricato da
un'unità locale?
(ossia l'URL inizia con
file: o res:)

FASE 4
L'URL richiesto
inizia con
http://,
https:// o
ftp://?

FASE 5
Il nome del dominio del
filmato che esegue la
richiesta corrisponde al
nome del dominio
dell'URL richiesto?



Quando si usa l'oggetto XMLSocket per creare una connessione a un server tramite socket, è necessario usare la porta 1024 o una superiore. Le porte con numero inferiore sono comunemente usate per Telnet, FTP, World Wide Web o Finger.

Flash sfrutta le funzioni di sicurezza dei browser standard, HTTP e HTTPS. Fondamentalmente Flash offre lo stesso livello di sicurezza del linguaggio HTML standard. Attenersi alle stesse regole valide per la creazione di siti Web HTML sicuri. Ad esempio, per supportare password sicure in Flash, è necessario definire l'autenticazione della password tramite richiesta a un server Web.

Per creare una password, usare un campo di testo per richiedere all'utente di immettere la password. Inviarla a un server in un'azione `loadVariables` o in un metodo `XML.sendAndLoad` usando un URL HTTPS con il metodo `POST`. Il server Web può quindi verificare se la password è valida. In questo modo la password non sarà mai disponibile nel file SWF.

Verifica dei dati caricati

Le azioni e i metodi che caricano dati in un filmato, a eccezione di `XMLSocket.send`, sono *asincroni*, ossia i risultati dell'azione vengono restituiti in un momento indeterminato.

Prima di potere usare i dati caricati in un filmato, occorre accertarsi che siano stati caricati. Ad esempio, non è possibile caricare variabili e manipolarne i valori all'interno dello stesso script. Nello script seguente non è possibile usare la variabile `lastFrameVisited` fino a quando non si è certi che la variabile sia stata caricata dal file `myData.txt`:

```
loadVariables("myData.txt", 0);
gotoAndPlay(lastFrameVisited);
```

Per ogni azione o metodo esiste una tecnica specifica da usare per verificare che i dati siano stati caricati. Se si usa l'azione `loadVariables` o `loadMovie`, è possibile caricare informazioni in un filmato clip target se usare l'evento `data` dell'azione `onClipEvent` per eseguire uno script. Se si usa l'azione `loadVariables` per caricare i dati, l'azione `onClipEvent(data)` viene eseguita quando viene caricata l'ultima variabile. Se si usa l'azione `loadMovie` per caricare i dati, l'azione `onClipEvent(data)` viene eseguita ogni volta che viene effettuato lo streaming di un frammento del filmato in Flash Player.

Ad esempio, la seguente azione associata a un pulsante carica le variabili dal file `myData.txt` nel clip filmato `loadTargetMC`:

```
on(release){
    loadVariables("myData.txt", _root.loadTargetMC);
}
```

Un'azione assegnata all'istanza `loadTargetMC` usa la variabile `lastFrameVisited` caricata dal file `myData.txt`. L'azione seguente verrà eseguita solo dopo che tutte le variabili, compresa `lastFrameVisited`, sono state caricate:

```
onClipEvent(data) {
    gotoAndPlay(lastFrameVisited);
}
```

Se si usano i metodi `XML.load` e `XMLSocket.connect`, è possibile definire un gestore che elabora i dati in arrivo. Un gestore è una proprietà dell'oggetto XML o `XMLSocket` alla quale si assegna una funzione che si è definita. I gestori vengono chiamati automaticamente al ricevimento delle informazioni. Per l'oggetto XML usare `XML.onLoad`. Per l'oggetto `XMLSocket` usare `XMLSocket.onConnect`.

Per ulteriori informazioni, vedere “Uso dell'oggetto XML” a pagina 143 e “Uso dell'oggetto XMLSocket” a pagina 146.

Uso di `loadVariables`, `getURL` e `loadMovie`

Le azioni `loadVariables`, `getURL` e `loadMovie` comunicano tutte con gli script lato server mediante il protocollo HTTP. Ogni azione invia tutte le variabili dalla linea temporale a cui è associata e gestisce la risposta nel modo seguente:

- `getURL` restituisce le informazioni alla finestra di un browser e non a Flash Player.
- `loadVariables` carica le variabili in una linea temporale specifica di Flash Player.
- `loadMovie` carica un filmato in un livello specifico di Flash Player.

Quando si usa l'azione `loadVariables`, `getURL` o `loadMovie`, è possibile specificare diversi argomenti:

- *URL* è il file in cui risiedono le variabili remote.
- *Location* è il livello o il target del filmato che riceve le variabili.

Per ulteriori informazioni su livelli e target, consultare “Informazioni sulle linee temporali multiple” a pagina 108.

Nota: l'azione `getURL` non richiede questo argomento.

- *Variables* imposta il metodo HTTP, GET o POST, in base al quale verranno inviate le variabili.

Ad esempio, se si desidera tenere traccia dei punteggi migliori di un gioco, è possibile memorizzare i punteggi su un server e usare un'azione `loadVariables` per caricarli nel filmato ogni volta che qualcuno gioca. L'azione dovrebbe assomigliare alla seguente:

```
loadVariables("http://www.mySite.com/scripts/high_score.php",  
_root.scoreClip, GET)
```

Questa azione carica le variabili dallo script PHP detto `high_score.php` nell'istanza del clip filmato `scoreClip` usando il metodo HTTP GET.

Le variabili caricate mediante l'azione `loadVariables` devono essere nel formato MIME standard `application/x-www-urlformencoded` (un formato standard usato dagli script CGI). Il file specificato nell'argomento URL dell'azione `loadVariables` deve riportare le coppie di variabili e valori in questo formato affinché Flash possa leggerle.

Il file può definire un numero qualsiasi di variabili. Le coppie di variabili e valori devono essere separate dal simbolo di "&" commerciale (&), mentre le parole all'interno di un valore devono essere separate dal segno più (+). Ad esempio, questa frase definisce diverse variabili:

```
highScore1=54000&playerName1=rockin+good&highScore2=53455&playerName2=bonehelmet&highScore3=42885&playerName3=soda+pop
```

Per ulteriori informazioni su `loadVariables`, `getURL` e `loadMovie`, consultare le voci corrispondenti nel capitolo 7 “Dizionario di ActionScript”.

Informazioni su XML

XML (*Extensible Markup Language*) si sta affermando come standard per lo scambio di dati strutturati tra le applicazioni Internet. È possibile integrare i dati in Flash con i server che usano la tecnologia XML per creare applicazioni sofisticate, ad esempio un sistema di conversazione o un sistema di mediazione.

In XML, come in HTML, è possibile usare tag per *contrassegnare*, o definire, una porzione del corpo de testo. In HTML è possibile usare tag predefiniti per indicare l'aspetto del testo in un browser Web (ad esempio, il tag `` indica che il testo è in grassetto). In XML occorre definire tag che identificano il tipo di un dato (ad esempio `<password>VerySecret</password>`). XML separa la definizione della struttura delle informazioni dal relativo aspetto. In questo modo un documento XML può essere usato più volte in ambienti diversi.

Ogni tag XML è detto *nodo* o elemento. Ogni nodo è caratterizzato da un tipo (1 indica un elemento XML oppure 3 indica un nodo di testo) e ogni elemento può avere degli attributi. Un nodo annidato in un altro nodo è detto *secondario* o *childNode*. Questa struttura ad albero gerarchica dei nodi, simile a DOM JavaScript, è detta DOM (Document Object Model, modello a oggetti del documento) XML e coincide con la struttura degli elementi in un browser Web.

Nell'esempio seguente `<PORTFOLIO>` è il nodo principale. Questo nodo non ha attributi e contiene il *childNode* `<HOLDING>` con gli attributi `SYMBOL`, `QTY`, `PRICE` e `VALUE`:

```
<PORTFOLIO>
  <HOLDING SYMBOL="RICH"
    QTY="75"
    PRICE="245.50"
    VALUE="18412.50" />
</PORTFOLIO>
```

Uso dell'oggetto XML

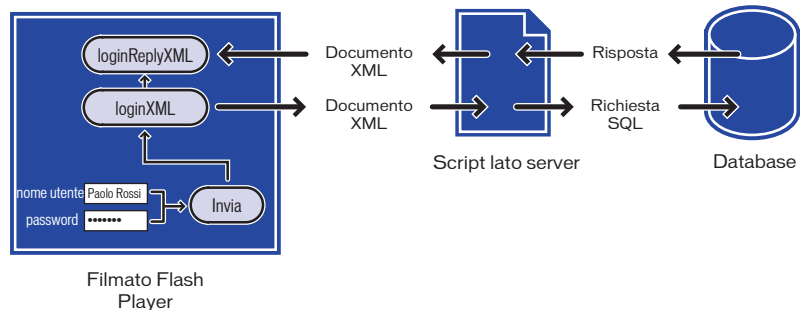
È possibile usare i metodi dell'oggetto XML ActionScript (ad esempio `appendChild`, `removeNode` e `insertBefore`) per strutturare i dati XML in Flash da inviare a un server e per manipolare e interpretare i dati XML scaricati.

È possibile usare i metodi dell'oggetto XML seguenti per inviare a e caricare dati XML da un server mediante il metodo HTTP POST:

- `load` scarica i dati XML da un URL e li inserisce in un oggetto XML ActionScript.
- `send` passa un oggetto XML a un URL. Le informazioni restituite vengono inviate a un'altra finestra del browser.
- `sendAndLoad` invia un oggetto XML a un URL. Le informazioni restituite vengono inserite in un oggetto XML ActionScript.

Ad esempio, è possibile creare un sistema di mediazione per la negoziazione di titoli che memorizza tutte le informazioni (nomi degli utenti, password, ID delle sessioni, portafoglio titoli e informazioni sulle transazioni) in un database.

Lo script lato server che passa le informazioni tra Flash e il database legge e scrive i dati in formato XML. È possibile usare ActionScript per convertire le informazioni raccolte nel filmato Flash (ad esempio un nome utente e una password) in un oggetto XML e inviare quindi i dati allo script lato server come documento XML. È inoltre possibile usare ActionScript per caricare il documento XML restituito dal server in un oggetto XML da usare nel filmato.



Il flusso e la conversione dei dati tra un filmato Flash Player, un documento di scripting lato server e un database.

La convalida della password per il sistema di mediazione richiede due script: una funzione definita nel fotogramma 1 e uno script che crea e invia gli oggetti XML associati al pulsante INVIA nel modulo.

Quando gli utenti immettono le informazioni nei campi di testo del filmato Flash a cui sono associate le variabili `username` e `password`, le variabili devono essere convertite in XML prima che vengano passate al server. La prima sezione dello script carica le variabili in un nuovo oggetto XML di nome `loginXML`. Quando un utente preme il pulsante INVIA, l'oggetto `loginXML` viene convertito in una stringa XML e inviato al server.

Lo script seguente è associato al pulsante INVIA. Per capire il funzionamento dello script, leggere le righe di commento di ogni script contrassegnate dai caratteri `//`:

```
on (release) {  
    // A. Crea un documento XML con un elemento LOGIN  
    loginXML = new XML();  
    loginElement = loginXML.createElement("LOGIN");  
    loginElement.attributes.username = username;  
    loginElement.attributes.password = password;  
    loginXML.appendChild(loginElement);  
  
    // B. Crea un oggetto XML per contenere la risposta del server  
    loginReplyXML = new XML();  
    loginReplyXML.onLoad = onLoginReply;  
  
    // C. Invia l'elemento LOGIN al server e  
    //     inserisce la risposta in loginReplyXML  
    loginXML.sendAndLoad("https://www.imxstocks.com/main.cgi",  
        loginReplyXML);  
}
```

La prima sezione dello script genera la stringa XML seguente quando l'utente preme il pulsante INVIA:

```
<LOGIN USERNAME="JeanSmith" PASSWORD="VerySecret" />
```

Il server riceve la stringa XML, genera una risposta XML e la rinvia al filmato Flash. Se la password viene accettata, il server risponde con il seguente messaggio:

```
<LOGINREPLY STATUS="OK" SESSION="rnr6f7vkj2oe14m7jkkycilb" />
```

Questa stringa XML include un attributo `SESSION` che contiene un ID della sessione univoco, generato a caso, che verrà usato in tutte le comunicazioni tra il client e il server per il resto della sessione. Se la password viene rifiutata, il server risponde con il seguente messaggio:

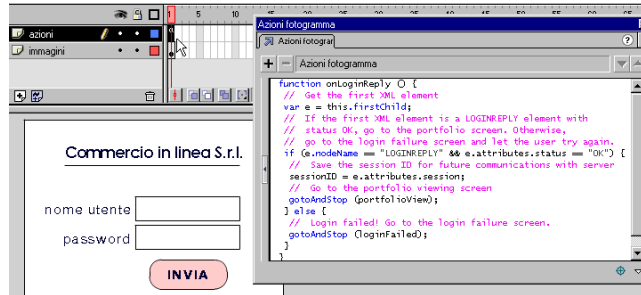
```
<LOGINREPLY STATUS="FAILED" />
```

Il nodo XML `LOGINREPLY` deve essere caricato in un oggetto XML vuoto nel filmato Flash. La seguente istruzione crea l'oggetto XML `loginReplyXML` per ricevere il nodo XML:

```
// B. Crea un oggetto XML per contenere la risposta del server  
loginReplyXML = new XML();  
loginReplyXML.onLoad = onLoginReply;
```


La seconda istruzione assegna la funzione `onLoginReply` al gestore `loginReplyXML.onLoad`.

L'elemento XML `LOGINREPLY` arriva in modo asincrono, come i dati da un'azione `loadVariables`, e viene caricato nell'oggetto `loginReplyXML`. Quando arrivano i dati, viene chiamato il metodo `onLoad` dell'oggetto `loginReplyXML`. È necessario definire la funzione `onLoginReply` e assegnarla al gestore `loginReplyXML.onLoad` in modo che quest'ultimo possa elaborare l'elemento `LOGINREPLY`. La funzione `onLoginReply` è assegnata al fotogramma che contiene il pulsante **Invio**.



La funzione `onLoginReply` è definita nel primo fotogramma del filmato.

La funzione `onLoginReply` è definita nel primo fotogramma del filmato. Per capire il funzionamento dello script, leggere le righe di commento di ogni script contrassegnate dai caratteri `//`:

```
function onLoginReply() {  
    // Ottiene il primo elemento XML  
    var e = this.firstChild;  
    // Se il primo elemento XML è un elemento LOGINREPLY il cui  
    // stato è OK, va alla schermata di visualizzazione del  
    // portafoglio. In caso contrario va alla schermata  
    // di login non riuscito e consente all'utente di riprovare.  
    if (e.nodeName == "LOGINREPLY" && e.attributes.status == "OK") {  
        // Salva l'ID della sessione per future comunicazioni  
        // con il server  
        sessionId = e.attributes.session;  
        // Va alla schermata di visualizzazione del portafoglio  
        gotoAndStop("portfolioView");  
    } else {  
        // Login non riuscito Va alla schermata di login  
        // non riuscito.  
        gotoAndStop("loginFailed");  
    }  
}
```

La prima riga di questa funzione, `var e = this.firstChild`, usa la parola chiave `this` per fare riferimento all'oggetto XML `loginReplyXML` che è stato appena caricato tramite XML dal server. È possibile usare `this` perché `onLoginReply` è stato chiamato come `loginReplyXML.onLoad`. Quindi anche se `onLoginReply` sembra una funzione semplice, in realtà si comporta come un metodo di `loginReplyXML`.

Per inviare il nome utente e la password come XML al server e per caricare una risposta XML nel filmato Flash, è possibile usare il metodo `sendAndLoad` come nell'esempio seguente:

```
// C. Invia l'elemento LOGIN al server e
//    inserisce la risposta in loginReplyXML
loginXML.sendAndLoad("https://www.imexstocks.com/main.cgi",
loginReplyXML);
```

Per ulteriori informazioni sui metodi XML, consultare le voci corrispondenti nel capitolo 7 “Dizionario di ActionScript”.

Nota: questo script è solo un esempio e viene fornita nessuna garanzia sul livello di sicurezza fornito. Se si desidera implementare un sistema sicuro protetto mediante password, accertarsi di avere una conoscenza approfondita della sicurezza di rete.

Uso dell'oggetto XMLSocket

ActionScript fornisce un oggetto `XMLSocket` predefinito che consente di stabilire una connessione continua al server. Una connessione tramite socket consente al server di inviare le informazioni al client non appena queste sono disponibili. Senza una connessione continua il server deve attendere una richiesta HTTP. Questa connessione aperta elimina i problemi di latenza ed è comunemente usata per le applicazioni in tempo reale, ad esempio i sistemi di conversazione. I dati vengono inviati attraverso la connessione tramite socket sotto forma di stringa e devono essere in formato XML. È possibile usare l'oggetto XML per strutturare i dati.

Per creare una connessione tramite socket, è necessario creare un'applicazione lato server che attenda la richiesta della connessione tramite socket e invii una risposta al filmato Flash. Questo tipo di applicazione lato server può essere scritto in un linguaggio di programmazione quale Java.

È possibile usare i metodi `connect` e `send` dell'oggetto `XMLSocket` ActionScript per trasferire dati XML a e da un server attraverso una connessione tramite socket. Il metodo `connect` stabilisce una connessione tramite socket con una porta del server Web. Il metodo `send` passa un oggetto XML al server specificato nella connessione tramite socket.

Quando si chiama il metodo `connect` dell'oggetto `XMLSocket`, Flash Player stabilisce una connessione TCP/IP al server e la mantiene aperta fino a quando non si verifica una delle seguenti condizioni:

- Viene chiamato il metodo `close` dell'oggetto `XMLSocket`.
- Non esistono più riferimenti all'oggetto `XMLSocket`.
- Viene chiuso Flash Player.
- La connessione viene interrotta (ad esempio, il modem viene scollegato).

Nell'esempio seguente viene creata una connessione tramite socket XML e vengono inviati dati dall'oggetto XML `myXML`. Per capire il funzionamento dello script, leggere le righe di commento di ogni script contrassegnate dai caratteri `//`:

```
// Crea un nuovo oggetto XMLSocket
sock = new XMLSocket();
// Chiama il metodo di connessione corrispondente per stabilire
// una connessione con la porta 1024 del server all'URL
sock.connect("http://www.myserver.com", 1024);
// Definisce una funzione da assegnare all'oggetto sock che
// gestisce la risposta del server. Se la connessione viene
// stabilita, invia l'oggetto myXML. Se la connessione non
// riesce, visualizza un messaggio di errore in un campo di testo.
function onSockConnect(success){
    if (success){
        sock.send(myXML);
    } else {
        msg="There has been an error connecting to "+serverName;
    }
}
// Assegna la funzione onSockConnect alla proprietà onConnect
sock.onConnect = onSockConnect;
```

Per ulteriori informazioni, vedere la voce `XMLSocket` nel capitolo 7 “Dizionario di ActionScript”.

Creazione di moduli

I moduli Flash forniscono un tipo di interattività avanzata: una combinazione di pulsanti, filmati e campi di testo che consentono di passare informazioni a un'altra applicazione su un server locale o remoto. Tutti gli elementi comuni di un modulo, quali i pulsanti di scelta, le caselle di riepilogo a discesa e le caselle di controllo, possono essere creati come filmati o pulsanti con l'aspetto del progetto globale del proprio sito Web. L'elemento più comune di un modulo è il campo di testo di input.

Tra i tipi comuni di moduli che usano tali elementi di interfaccia figurano le interfacce dei sistemi di conversazione, i moduli d'ordine e le interfacce di ricerca. Ad esempio, un modulo Flash può raccogliere le informazioni sull'indirizzo e inviarle a un'altra applicazione che le compila in un messaggio di posta elettronica o in un file di database. Anche un singolo campo di testo è considerato un modulo e può essere usato per raccogliere l'input dell'utente e visualizzare risultati.

I moduli richiedono due componenti principali: gli elementi dell'interfaccia Flash che formano il modulo e un'applicazione lato server o uno script lato client per elaborare le informazioni immesse dall'utente. Le operazioni seguenti descrivono la procedura generale per la creazione di un modulo in Flash.

Per creare un modulo:

- 1 Posizionare gli elementi di interfaccia nel filmato usando il layout desiderato.
È possibile usare gli elementi di interfaccia della libreria comune Pulsanti o crearne di nuovi.
- 2 Nel pannello Opzioni testo impostare i campi di testo su Testo di input e assegnare a ciascun campo un nome di variabile univoco.
Per ulteriori informazioni sulla creazione di campi di testo modificabili, consultare *Guida all'uso di Flash*.
- 3 Assegnare un'azione che invii, carichi o invii e carichi i dati.

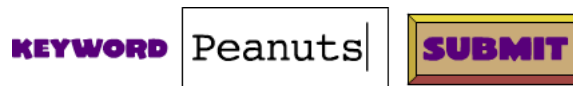
Creazione di un modulo di ricerca

Un esempio di un semplice modulo è un campo di ricerca con un pulsante Invia. Quale introduzione alla creazione di moduli, l'esempio seguente fornisce le istruzioni per creare un'interfaccia di ricerca usando un'azione `getURL`. Immettendo le informazioni richieste gli utenti possono inviare una parola chiave a un motore di ricerca su un server Web remoto.

Per creare un semplice modulo di ricerca:

- 1 Creare un pulsante per l'invio dei dati immessi.
- 2 Creare un'etichetta, un campo di testo vuoto e un'istanza del pulsante sullo stage.

La schermata dovrebbe assomigliare alla seguente:



- 3 Selezionare il campo di testo e scegliere Finestra > Pannelli > Opzioni testo.

4 Nel pannello Opzioni testo impostare le opzioni seguenti:

- Scegliere Testo di input dal menu a comparsa.
- Selezionare Bordo/Sf.
- Specificare un nome di variabile.

Nota: i singoli motori di ricerca potrebbero richiedere un nome di variabile specifico. Per informazioni, visitare il sito Web del motore di ricerca.

5 Sullo stage selezionare il pulsante e scegliere Finestra > Azioni.

Verrà visualizzato il pannello Azioni oggetto.

Nota: un segno di spunta accanto ad Azioni nel menu Finestra indica che il pannello è aperto.

6 Trascinare l'azione `getURL` dalla lista nel riquadro a sinistra nella finestra di script a destra.

7 Nel riquadro dei parametri impostare le opzioni seguenti:

- Per URL, immettere l'URL del motore di ricerca.
- Per Finestra, selezionare `_blank`. Verrà aperta una nuova finestra che visualizza i risultati della ricerca.
- Per Variabili, selezionare Invia con GET.

8 Per provare il modulo, scegliere File > Anteprima pubblicazione > HTML.

Uso di variabili nei moduli

È possibile usare variabili in un modulo per memorizzare l'input dell'utente. Per impostare le variabili, usare i campi di testo modificabili o assegnare azioni ai pulsanti negli elementi di interfaccia. Ad esempio, ogni elemento di un menu a comparsa è un pulsante con un'azione che imposta una variabile che indica l'elemento selezionato. È possibile assegnare un nome di variabile a un campo di testo di input. Il campo di testo funziona come una finestra che visualizza il valore della variabile.

Quando si passano informazioni a e da uno script lato server, le variabili nel filmato Flash devono corrispondere alle variabili nello script. Ad esempio, se lo script richiede una variabile di nome `password`, al campo di testo in cui gli utenti immettono la password deve essere assegnato il nome di variabile `password`.

Alcuni script richiedono variabili nascoste, ovvero variabili che l'utente non vede mai. Per creare una variabile nascosta in Flash, è possibile impostare una variabile in un fotogramma del clip filmato che contiene gli altri elementi del modulo. Le variabili nascoste vengono inviate allo script lato server insieme alle altre variabili impostate sulla linea temporale che contiene l'azione che invia il modulo.

Verifica dei dati immessi

Per un modulo che passa le variabili a un'applicazione su un server Web, è necessario verificare che gli utenti immettano le informazioni corrette. Ad esempio, si desidera impedire che gli utenti immettano testo in un campo che deve contenere un numero di telefono. Usare una serie di azioni `set variable` unitamente a `for` e `if` per valutare i dati immessi.

L'azione di esempio seguente controlla se i dati immessi rappresentano un numero e se il formato del numero è `###-###-####`. Se i dati sono validi, verrà visualizzato un messaggio che conferma che il numero di telefono immesso è valido. Se i dati non sono validi, verrà visualizzato un messaggio che comunica che il numero di telefono immesso non è valido.

Per usare questo script in un filmato, creare due campi di testo sullo stage e scegliere Testo di input per ciascun campo nel pannello Opzioni testo. Assegnare la variabile `phoneNumber` a un campo di testo e la variabile `message` all'altro. Associare l'azione seguente a un pulsante sullo stage accanto ai campi di testo:

```
on (release) {
  valid = validPhoneNumber(phoneNumber);
  if (valid) {
    message = "Good, this is a valid phone number!";
  } else {
    message = "This phone number is invalid!";
  }
  function isdigit(ch) {
    return ch.length == 1 && ch >= '0' && ch <= '9';
  }
  function validPhoneNumber(phoneNumber) {
    if (phoneNumber.length != 12) {
      return false;
    }
    for (var index = 0; index < 12; index++) {
      var ch = phoneNumber.charAt(index);
      if (index == 3 || index == 7) {
        if (ch != "-") {
          return false;
        }
      } else if (!isdigit(ch)) {
        return false;
      }
    }
    return true;
  }
}
```

Per inviare i dati, creare un pulsante con un'azione simile alla seguente. Sostituire gli argomenti `getURL` con argomenti appropriati per il filmato.

```
on (release) {  
    if (valid) {  
        getURL("http://www.webserver.com", "_self", "GET");  
    }  
}
```

Per ulteriori informazioni su queste istruzioni `ActionScript`, consultare `set`, `for` e `if` nel capitolo 7 “Dizionario di `ActionScript`” a pagina 167”.

Invio di messaggi a e da Flash Player

Per inviare messaggi da un filmato Flash all'ambiente host (ad esempio un browser Web, un filmato Director o la versione autonoma di Flash Player), è possibile usare l'azione `fscommand`. In questo modo è possibile estendere il filmato usando le funzioni dell'host. Ad esempio, è possibile passare un'azione `fscommand` a una funzione JavaScript in una pagina HTML che apre una nuova finestra del browser con proprietà specifiche.

Per controllare un filmato in Flash Player tramite linguaggi di scripting per browser Web, quali JavaScript, VBScript e Microsoft JScript, è possibile usare i metodi Flash Player, ossia funzioni che inviano messaggi da un ambiente host al filmato Flash. Ad esempio, è possibile avere un collegamento in una pagina HTML che sposta il filmato Flash a un fotogramma specifico.

Uso di fscommand

Usare l'azione `fscommand` per inviare un messaggio al programma che ospita Flash Player. L'azione `fscommand` ha due parametri: *command* e *arguments*. Per inviare un messaggio alla versione autonoma di Flash Player, è necessario usare i comandi e gli argomenti predefiniti. Ad esempio, l'azione seguente imposta il lettore autonomo in modo che ridimensioni il filmato e lo ingrandisca a schermo intero quando si rilascia il pulsante:

```
on(release){  
    fscommand("fullscreen", "true");  
}
```

La tabella seguente contiene i valori che è possibile specificare per i parametri *command* e *arguments* dell'azione `fscommand` per controllare un filmato riprodotto nel lettore autonomo (proiettori compresi):

<i>command</i>	<i>arguments</i>	Scopo
quit	Nessuno	Chiude il proiettore.
fullscreen	true o false	Se si specifica true, Flash Player viene impostato in modalità a schermo intero. Se si specifica false, il lettore torna alla visualizzazione dei menu normale.
allowscale	true o false	Se si specifica false, il lettore viene impostato in modo tale che il filmato sia sempre visualizzato nelle dimensioni originali e non venga mai ridimensionato. Se si specifica true, il filmato viene ridimensionato per coincidere con le dimensioni del lettore.
showmenu	true o false	Se si specifica true, viene attivato l'intero gruppo delle voci di menu di scelta rapida. Se si specifica false, tutte le voci dei menu di scelta rapida, a eccezione di Informazioni su Flash Player, vengono disattivate.
exec	Percorso dell'applicazione	Esegue un'applicazione dal proiettore.

Per usare `fscommand` per inviare un messaggio a un linguaggio di scripting quale JavaScript in un browser Web, è possibile passare due argomenti qualsiasi nei parametri *command* e *arguments*. Questi argomenti possono essere stringhe o espressioni e saranno usati in una funzione JavaScript che “cattura” o gestisce l'azione `fscommand`.

Un'azione `fscommand` chiama la funzione JavaScript `nomefilmato_DoFSCCommand` nella pagina HTML che incorpora il filmato Flash, dove *nomefilmato* è il nome di Flash Player assegnato dall'attributo NAME del tag EMBED o dall'attributo ID del tag OBJECT. Se a Flash Player viene assegnato il nome `myMovie`, la funzione JavaScript chiamata è `myMovie_DoFSCCommand`.

Per usare l'azione `fscommand` per aprire da un filmato Flash una finestra che contiene un messaggio nella pagina HTML tramite JavaScript:

- 1 Nella pagina HTML che incorpora il filmato Flash aggiungere il seguente codice JavaScript:

```
function theMovie_DoFSCommand(command, args) {  
    if (command == "messagebox") {  
        alert(args);  
    }  
}
```

Se si pubblica il filmato usando Flash con il modello `FSCommand` nelle impostazioni di pubblicazione HTML, questo codice viene inserito automaticamente. Gli attributi `NAME` e `ID` del filmato saranno il nome del file. Ad esempio, per il file `myMovie.fl` gli attributi verrebbero impostati su `myMovie`. Per ulteriori informazioni sulla pubblicazione, consultare *Guida all'uso di Flash*.

- 2 Nel filmato Flash aggiungere l'azione `fscommand` a un pulsante:

```
fscommand("messagebox", "This is a message box invoked from  
within Flash.")
```

È inoltre possibile usare espressioni per l'azione `fscommand` e gli argomenti, come nell'esempio seguente:

```
fscommand("messagebox", "Hello, " & name & ", welcome to our  
Web site!")
```

- 3 Scegliere **File > Anteprima pubblicazione > HTML** per provare il filmato.

L'azione `fscommand` può inviare messaggi a Macromedia Director che vengono interpretati da Lingo come stringhe, eventi o codice Lingo eseguibile. Se il messaggio è una stringa o un evento, è necessario scrivere il codice Lingo per riceverlo dall'azione `fscommand` ed eseguire un'azione in Director. Per ulteriori informazioni, visitare il Centro di assistenza Director all'indirizzo <http://www.macromedia.com/support/director>.

In Visual Basic, Visual C++ e altri programmi che supportano i controlli ActiveX, `fscommand` invia un evento VB con due stringhe che possono essere gestite nel linguaggio di programmazione dell'ambiente. Per ulteriori informazioni, usare le parole chiave `Flash method` per eseguire una ricerca nel Centro di assistenza Flash all'indirizzo <http://www.macromedia.com/support/flash>.

Informazioni sui metodi Flash Player

È possibile usare i metodi Flash Player per controllare un filmato in Flash Player dai linguaggi di scripting per browser Web, quali JavaScript e VBScript. Come nel caso di altri metodi è possibile usare i metodi Flash Player per inviare chiamate ai filmati Flash Player da un ambiente di scripting diverso da ActionScript. Ogni metodo ha un nome e la maggior parte dei metodi richiede il passaggio di argomenti. Un argomento specifica un valore su cui agisce il metodo. Il calcolo eseguito da alcuni metodi restituisce un valore che può essere usato dall'ambiente di scripting.

Esistono due tecnologie diverse che consentono la comunicazione tra il browser e Flash Player: LiveConnect (Netscape Navigator 3.0 o versione successiva su Windows 95/98/2000/NT o Power Macintosh) e ActiveX (Microsoft Internet Explorer 3.0 o versione successiva su Windows 95/98/2000/NT). Sebbene le tecniche di scripting siano simili per tutti i browser e i linguaggi, per i controlli ActiveX sono disponibili proprietà ed eventi supplementari.

Per ulteriori informazioni e una lista completa dei metodi di scripting per Flash Player, usare le parole chiave `Flash method` per eseguire una ricerca nelCentro di assistenza Flash all'indirizzo <http://www.macromedia.com/support/flash>.

CAPITOLO 6

Risoluzione dei problemi relativi ad ActionScript

La complessità di alcune azioni, specie se combinate tra loro, può rendere particolarmente complessi anche i filmati Flash. Come in qualsiasi linguaggio di programmazione, è inoltre possibile scrivere codice ActionScript non corretto, che genera errori negli script. L'uso di tecniche di creazione codice adeguate semplifica la risoluzione dei problemi relativi al filmato in caso di imprevisti.

Flash dispone di vari strumenti che consentono la prova dei filmati in modalità di prova filmato o in un browser Web. Il Debugger visualizza una lista gerarchica dei clip filmato attualmente caricati in Flash Player e consente di visualizzare e modificare i valori delle variabili durante la riproduzione del filmato. In modalità di prova filmato, la finestra Output visualizza i messaggi di errore e liste di variabili e oggetti. È inoltre possibile usare l'azione `trace` negli script per inviare note di programmazione e valori di espressioni alla finestra Output.

Indicazioni per la creazione di codice e la risoluzione dei problemi.

Se durante la creazione di script si seguono procedure di creazione di codice appropriate, è possibile ridurre il numero di errori di programmazione (bug). Usare le indicazioni seguenti per ridurre al minimo l'insorgenza di problemi e per risolverli rapidamente nel caso si verificano.

Uso di procedure di creazione codice adeguate

È consigliabile salvare più versioni del filmato durante la creazione del codice. Scegliere File > Salva con nome per salvare una versione con nome diverso ogni mezz'ora. Sarà così possibile individuare l'introduzione di un errore risalendo fino al primo file in cui il problema non si presentava. Questo approccio consente di disporre sempre di una versione funzionante, anche se un file risultasse danneggiato.

Un'altra buona norma è quella di eseguire prove fin dall'inizio, frequentemente e su tutte le piattaforme di destinazione, in modo da individuare i problemi alla nascita. Scegliere Controlli > Prova filmato per eseguire il filmato in modalità di prova filmato ogni volta che si apporta una modifica importante o prima di salvare una versione. In modalità di prova filmato il filmato viene eseguito in una versione del lettore autonomo.

Se il filmato è destinato alla visualizzazione sul Web, è opportuno provarlo anche in un browser. In determinate situazioni (ad esempio per lo sviluppo di un'intranet) è possibile che il browser e la piattaforma usati dai destinatari del filmato siano già noti. Tuttavia, quando si sviluppa un sito Web, è opportuno provare il filmato in tutti i browser e su tutte le piattaforme possibili.

È consigliabile osservare le seguenti procedure di creazione codice:

- Usare l'azione `trace` per inviare commenti alla finestra Output Consultare “Uso di trace” a pagina 166.
- Usare l'azione `comment` per includere istruzioni che vengono visualizzate esclusivamente nel pannello Azioni. Consultare “Commenti” a pagina 53.
- Usare convenzioni di denominazione coerenti per identificare gli elementi di uno script. Ad esempio è consigliabile evitare l'uso di spazi nei nomi. Assegnare alle variabili e alle funzioni nomi che iniziano con una lettera minuscola e usare lettere maiuscole per le iniziali delle parole interne (`myVariableName`, `myFunctionName`). Assegnare alle funzioni di costruzione nomi che iniziano con lettere maiuscole (`MyConstructorFunction`). È importante scegliere uno stile riconoscibile e adottarlo in modo coerente.
- Per le variabili usare nomi significativi che indichino il tipo di contenuto della variabile. Ad esempio, una variabile contenente informazioni sull'ultimo pulsante selezionato può essere chiamata `lastButtonPressed`. Un nome quale `foo` non semplificherebbe l'identificazione del contenuto della variabile.

- Per rintracciare i valori delle variabili usare i campi di testo modificabili nei livelli di guida, anziché il Debugger.
- Usare Prova filmato in modalità di modifica filmato per accedere alla lista di visualizzazione e a tutte le azioni di un filmato. Vedere Guida di Flash.
- Usare l'azione `for...in` per eseguire cicli tra le proprietà dei clip filmato, compresi i clip filmato secondari. È possibile usare l'azione `for...in` con l'azione `trace` per inviare una lista di proprietà alla finestra Output. Consultare “Ripetizione di un'azione” a pagina 72.

Uso di un elenco di verifica per la risoluzione dei problemi

Come in ogni ambiente di sviluppo è possibile identificare errori comunemente commessi durante la creazione di script. L'elenco di verifica seguente è un buon punto di partenza per individuare il problema relativo al filmato.

- Verificare che sia attiva la modalità di prova filmato.
In modalità di creazione codice funzionano soltanto azioni fotogramma e pulsante semplici (ad esempio `gotoAndPlay` e `stop`). Per attivare tali azioni, scegliere Controlli > Attiva azioni fotogramma semplici oppure Controlli > Attiva pulsanti semplici.
- Verificare che non siano presenti azioni fotogramma in conflitto tra loro su più livelli.
- Se si usa il pannello Azioni in Modalità normale, verificare che l'istruzione sia impostata come espressione.
Se si passa un'espressione a un'azione senza aver selezionato la casella di controllo Espressione, il valore verrà passato come stringa. Consultare “Uso di operatori per la gestione dei valori nelle espressioni” a pagina 62.
- Verificare che non siano presenti più elementi di ActionScript con lo stesso nome.
È consigliabile assegnare a ogni variabile, funzione, oggetto e proprietà un nome univoco, ad eccezione delle variabili locali che devono avere nomi univoci soltanto all'interno dell'area di validità e vengono spesso riusate come contatori. Consultare “Assegnazione di un'area di validità a una variabile” a pagina 58.

Per ulteriori suggerimenti sulla risoluzione dei problemi relativi a un filmato Flash, visitare il Centro di assistenza Flash all'indirizzo <http://www.macromedia.com/support/flash>.

Uso del Debugger

Il Debugger consente di identificare gli errori in un filmato durante l'esecuzione in Flash Player. È possibile consultare la lista di visualizzazione di clip filmato e filmati caricati e modificare i valori delle variabili e proprietà inserendo valori appropriati. È quindi possibile tornare agli script e modificarli per ottenere i risultati desiderati. Per usare il Debugger è necessario eseguire Flash Debug Player, una versione speciale di Flash Player.

Flash Debug Player viene installato automaticamente con l'applicazione di creazione codice Flash 5. Questo programma consente di scaricare nel Debugger dell'applicazione di creazione codice Flash la lista di visualizzazione, i nomi delle variabili e delle proprietà con i relativi valori.

Per visualizzare il Debugger:

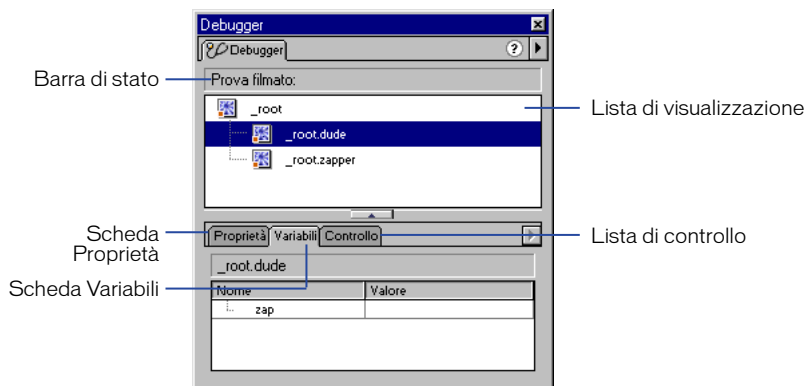
Scegliere Finestra > Debugger.

Il Debugger viene aperto con lo stato inattivo. Fino a quando Flash Player non invia un comando, nella lista di visualizzazione non appare alcun dato.

Per attivare il Debugger in modalità di prova filmato:

Scegliere Controlli > Debug filmato.

Il Debugger viene aperto con lo stato attivo.



Attivazione del debug in un filmato

Quando si esporta un filmato Flash Player è possibile scegliere di attivare il debug nel filmato e di creare una password di debug. Se non si attiva il debug, il Debugger non verrà avviato.

Come in JavaScript o HTML, le variabili di ActionScript sul client possono essere visualizzate dall'utente. Per memorizzare variabili in modo sicuro è necessario inviarle a un'applicazione su un server, anziché memorizzarle nel filmato.

È tuttavia possibile che lo sviluppatore del filmato Flash non desideri rivelare altre informazioni riservate, quali la struttura del clip filmato. Per consentire soltanto a utenti selezionati la visualizzazione dei filmati con Flash Debug Player, è possibile pubblicare il filmato con una password di debug.

Per attivare il debug e creare una password:

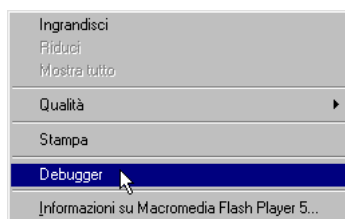
- 1 Scegliere File > Impostazioni pubblicazione.
- 2 Fare clic sulla scheda Flash.
- 3 Selezionare Debug consentito.
- 4 Per impostare una password, immetterla nella casella Password.

Se non si specifica la password non sarà possibile scaricare informazioni nel Debugger. Se il campo password viene lasciato vuoto, non viene attivata la protezione tramite password.

Per attivare il Debugger in un browser Web:

- 1 In Windows, fare clic con il pulsante destro del mouse o, in Macintosh, premere Ctrl e fare clic per aprire il menu di scelta rapida di Flash Debug Player.
- 2 Scegliere Debugger.

Nota: Il Debugger consente il monitoraggio di un solo filmato alla volta. Per poter usare il Debugger è necessario che Flash sia in esecuzione.



Menu di scelta rapida di Flash Debug Player

Informazioni sulla barra di stato

Quando attivata, la barra di stato del Debugger visualizza l'URL o il percorso locale del file del filmato. Flash Player è implementato in modi diversi a seconda dell'ambiente di riproduzione. La barra di stato del Debugger visualizza il tipo di Flash Player in cui è eseguito il filmato:

- Modalità di prova filmato
- Lettore autonomo
- Plug-in di Netscape

Il plug-in di Netscape viene usato con Netscape Navigator su Windows e Macintosh e con Microsoft Internet Explorer su Macintosh.

- Controllo ActiveX

Il controllo ActiveX è usato con Internet Explorer su Windows.

Informazioni sulla lista di visualizzazione

Quando il Debugger è attivo, contiene una lista di visualizzazione dei clip filmato aggiornata in tempo reale. È possibile espandere e ridurre le visualizzazioni per visualizzare tutti i clip filmato presenti attualmente nello stage. Quando vengono aggiunti o rimossi clip filmato dal filmato, la lista di visualizzazione riporta immediatamente le modifiche. È possibile ridimensionare la lista di visualizzazione spostando la barra di divisione orizzontale o trascinandola dall'angolo inferiore destro.

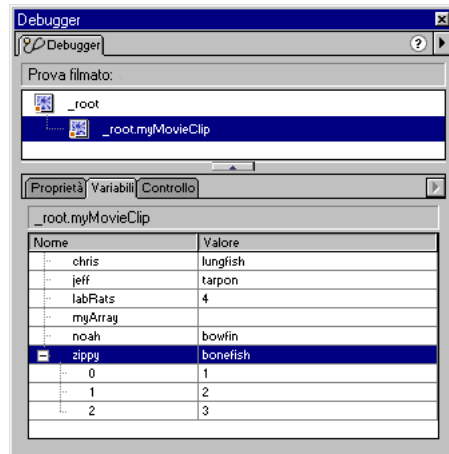
Visualizzazione e modifica di variabili

La scheda Variabili del Debugger visualizza i nomi e i valori delle variabili presenti nel filmato. Se si modifica il valore di una variabile nella scheda Variabili, è possibile osservarne l'effetto direttamente nel filmato durante la sua esecuzione. Ad esempio, per provare il rilevamento della presenza di collisioni in un gioco è possibile immettere il valore di una variabile in modo da posizionare una palla nel punto appropriato accanto a una parete.

Per visualizzare una variabile:

- 1 Selezionare il clip filmato contenente la variabile dalla lista di visualizzazione.
- 2 Fare clic sulla scheda Variabili.

La lista di visualizzazione viene aggiornata automaticamente durante la riproduzione del filmato. Se un clip filmato viene rimosso dal filmato in corrispondenza di un fotogramma specifico, tale clip viene rimosso anche dalla lista di visualizzazione nel Debugger, ossia vengono rimossi sia il nome che il valore della variabile.



Per modificare il valore di una variabile:

Selezionare il valore, quindi immetterne uno nuovo.

Il valore deve essere una costante (ad esempio "Hello", 3523, o "http://www.macromedia.com") e non un'espressione (ad esempio $x + 2$ o `eval("name: " + i)`). Il valore può essere una stringa, ossia qualsiasi valore racchiuso tra virgolette (" "), un valore numerico o un valore booleano (`true` o `false`).

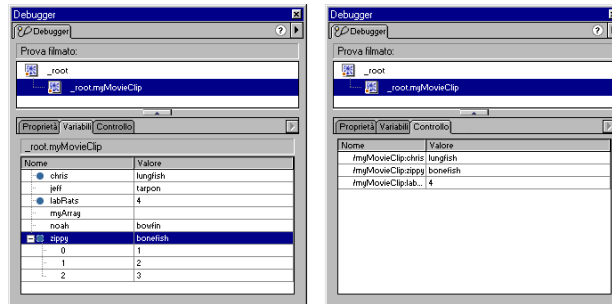
Le variabili Object e Array sono visualizzate nella scheda Variabili. Per visualizzare le proprietà e i valori di tali variabili, fare clic sul pulsante di aggiunta (+). Tuttavia nei campi dei valori non è possibile immettere valori per le variabili Object o Array (ad esempio `{name: "I am an object"}` o `[1, 2, 3]`).

Nota: Per inviare in output il valore di un'espressione in modalità di prova filmato, usare l'azione `trace`. Consultare "Uso di trace" a pagina 166.

Uso della lista di controllo

Per monitorare in modo organizzato un insieme di variabili critiche, è possibile contrassegnare le variabili affinché vengano visualizzate nella lista di controllo. La lista di controllo visualizza il percorso assoluto della variabile e il suo valore. È inoltre possibile immettere un nuovo valore della variabile nella lista di controllo.

È possibile aggiungere alla lista di controllo soltanto variabili e non proprietà né funzioni.



Variabili contrassegnate da includere nella lista di controllo e variabili nella lista di controllo.

Per aggiungere variabili alla lista di controllo, eseguire una delle operazioni descritte.

- Nella scheda Variabili, fare clic con il pulsante destro del mouse (Windows) o premere Controllo e fare clic (Macintosh) su una variabile selezionata, quindi scegliere Controllo dal menu di scelta rapida. La variabile viene affiancata da un punto blu.
- Nella scheda Controllo, fare clic con il pulsante destro del mouse (Windows) o premere Controllo e fare clic (Macintosh) e scegliere Aggiungi dal menu di scelta rapida. Immettere il nome e il valore della variabile nei campi.

Per rimuovere variabili dalla lista di controllo:

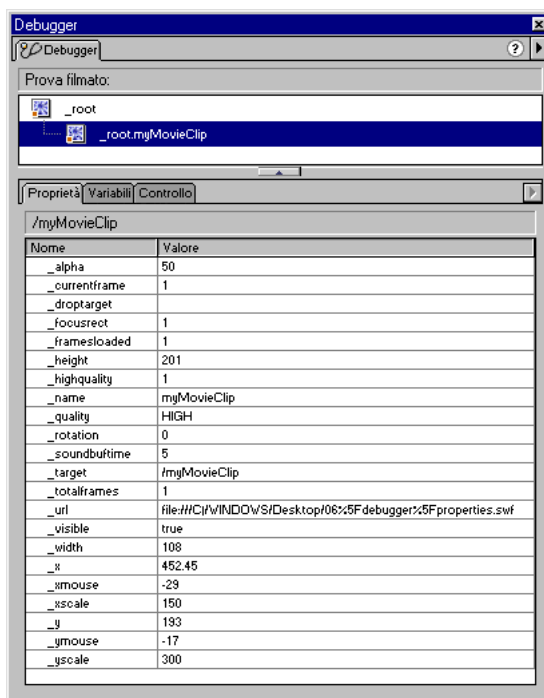
Nella scheda Controllo, fare clic con il pulsante destro del mouse (Windows) o premere Controllo e fare clic e scegliere Rimuovi dal menu di scelta rapida.

Visualizzazione delle proprietà dei filmati e modifica delle proprietà modificabili

La scheda Proprietà della finestra Debugger visualizza i valori di tutte le proprietà dei clip filmato nello stage. È possibile modificare il valore di una proprietà e osservarne l'effetto direttamente nel filmato durante la sua esecuzione. Alcune proprietà del clip filmato sono di sola lettura e non possono essere modificate.

Per visualizzare le proprietà di un clip filmato:

- 1 Selezionare un clip filmato dalla lista di visualizzazione.
- 2 Fare clic sulla scheda Proprietà.



Per modificare il valore di una proprietà:

Selezionare il valore, quindi immetterne uno nuovo.

Il valore deve essere una costante (ad esempio 50 o "clearwater") e non un'espressione (ad esempio $x + 50$). Il valore può essere una stringa, ossia qualsiasi valore racchiuso tra virgolette (" "), un valore numerico o un valore booleano (true o false). Tuttavia nel Debugger non è possibile immettere valori di tipo oggetto o matrice (ad esempio `{id: "rogue"}` o `[1, 2, 3]`).

Per ulteriori informazioni, vedere "String" a pagina 54 e "Uso di operatori per la gestione dei valori nelle espressioni" a pagina 62.

Nota: Per inviare in output il valore di un'espressione in modalità di prova filmato, usare l'azione `trace`. Consultare "Uso di trace" a pagina 166.

Uso della finestra Output

In modalità di prova filmato, la finestra Output visualizza informazioni che agevolano l'identificazione di problemi relativi al filmato. Alcune informazioni, quali gli errori di sintassi, vengono visualizzate automaticamente. Altre informazioni sono visualizzabili tramite i comandi Elenca oggetti ed Elenca variabili. Consultare “Uso di Elenca oggetti” a pagina 164 e “Uso di Elenca variabili” a pagina 165.

L'azione `trace` usata negli script consente di inviare informazioni specifiche alla finestra Output durante l'esecuzione del filmato. Tali informazioni possono includere note sullo stato del filmato o sul valore di un'espressione. Consultare “Uso di `trace`” a pagina 166.

Per visualizzare la finestra Output:

- 1 Se il filmato non è eseguito in modalità di prova filmato, scegliere Controlli > Prova filmato.
- 2 Scegliere Finestra > Output.
Viene visualizzata la finestra Output.

Nota: Se in uno script sono presenti errori di sintassi, la finestra Output appare automaticamente.

- 3 Per elaborare il contenuto della finestra Output usare il menu Opzioni:
 - Scegliere Opzioni > Copia per copiare il contenuto della finestra Output negli Appunti.
 - Scegliere Opzioni > Cancella per cancellare il contenuto della finestra.
 - Scegliere Opzioni > Salva su file per salvare il contenuto della finestra in un file di testo.
 - Scegliere Opzioni > Stampa per stampare il contenuto della finestra.

Uso di Elenca oggetti

In modalità di prova filmato, il comando Elenca oggetti visualizza il livello, il fotogramma, il tipo di oggetto (forma, clip filmato o pulsante) e il percorso target di un'istanza di clip filmato in una lista gerarchica. Questo comando è particolarmente utile per ottenere il percorso target e il nome dell'istanza corretti. A differenza di quanto accade nel Debugger, la lista non viene aggiornata automaticamente durante la riproduzione del filmato: ogni volta che si desidera inviare informazioni alla finestra Output è necessario scegliere il comando Elenca oggetti.

Per visualizzare la lista degli oggetti in un filmato:

- 1 Se il filmato non è eseguito in modalità di prova filmato, scegliere Controlli > Prova filmato.
- 2 Scegliere Debug > Elenca oggetti.

Nella finestra Output vengono elencati tutti gli oggetti sullo stage, come nell'esempio seguente:

```
Layer #0: Frame=3
Movie Clip: Frame=1 Target=_root.MC
  Shape:
    Movie Clip: Frame=1 Target=_root.instance3
      Shape:
        Button:
          Movie Clip: Frame=1 Target=_root.instance3.instance2
            Shape:
```

Nota: Il comando Elenca oggetti non elenca tutti gli oggetti di dati ActionScript. Nel presente contesto per oggetto si intende una forma o un simbolo nello stage.

Uso di Elenca variabili

In modalità di prova filmato, il comando Elenca variabili elenca tutte le variabili che si trovano attualmente nel filmato. Questo comando è particolarmente utile per ottenere il percorso target e il nome della variabile. A differenza di quanto accade nel Debugger la lista non viene aggiornato automaticamente durante la riproduzione del filmato: ogni volta che si desidera inviare informazioni alla finestra Output è necessario scegliere il comando Elenca variabili.

Per visualizzare la lista delle variabili in un filmato:

- 1 Se il filmato non è eseguito in modalità di prova filmato, scegliere Controlli > Prova filmato.
- 2 Scegliere Debug > comando Elenca variabili.

Nella finestra Output vengono elencate tutte le variabili presenti nel filmato, come nell'esempio seguente:

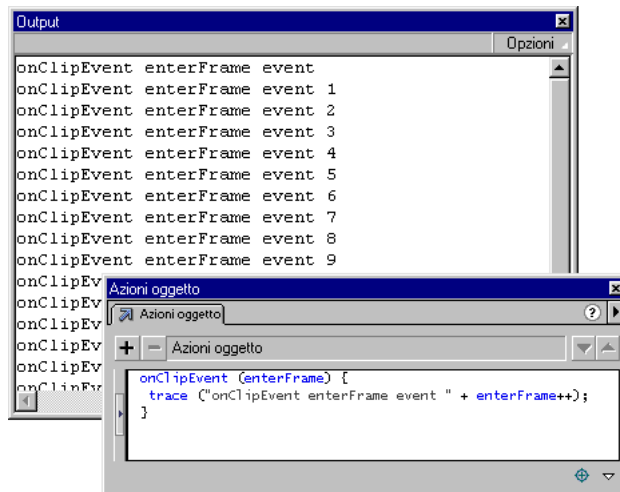
```
Level #0:
  Variable _root.country = "Sweden"
  Variable _root.city = "San Francisco"
Movie Clip: Target=""
Variable _root.instance1.firstName = "Rick"
```

Uso di trace

L'azione `trace` usata negli script consente di inviare informazioni alla finestra Output. Ad esempio, quando si prova un filmato o una scena è possibile inviare alla finestra note di programmazione oppure risultati specifici in corrispondenza della scelta di un pulsante o della riproduzione di un fotogramma. L'azione `trace` è simile all'istruzione JavaScript `alert`.

Quando si usa l'azione `trace` negli script è possibile usare espressioni come argomenti. Il valore di un'espressione viene visualizzato nella finestra Output in modalità di prova filmato, come nell'esempio seguente:

```
onClipEvent(enterFrame){  
    trace("onClipEvent enterFrame " + enterFrame++)  
}
```



L'azione `trace` restituisce valori che vengono visualizzati nella finestra Output.

CAPITOLO 7

Dizionario di ActionScript

Questa sezione della *Guida di riferimento di ActionScript* descrive la sintassi e l'uso degli elementi di ActionScript in Flash 5 e versioni successive. Le voci descritte in questa guida coincidono con quelle descritte nel Dizionario di ActionScript della Guida in linea. Per usare gli esempi in uno script, copiarne il testo dal Dizionario di ActionScript della Guida in linea, quindi incollarlo nel pannello Azioni in Modalità esperto.

Nel dizionario sono elencati tutti gli elementi di ActionScript, quali operatori, parole chiave, istruzioni, azioni, proprietà, funzioni, oggetti e metodi. Per una panoramica generale delle voci del dizionario, vedere “Contenuto del dizionario” a pagina 169. Le tabelle contenute in questa sezione offrono un buon punto di partenza per la ricerca di metodi o operatori simbolici di cui non si conosce la classe oggetto.

ActionScript è conforme allo standard ECMA-262, la specifica elaborata dall'Associazione europea dei produttori di computer, se non specificato diversamente.

Il dizionario contiene due tipi di voci:

- Singole voci relative a operatori, parole chiave, funzioni, variabili, proprietà, metodi e istruzioni
- Voci relative agli oggetti che forniscono informazioni dettagliate sugli oggetti predefiniti

Le informazioni fornite nelle voci di esempio consentono di interpretare la struttura e le convenzioni usate nei due tipi di voci.

Voci di esempio per la maggior parte degli elementi di ActionScript

Di seguito sono descritte le convenzioni usate per gli elementi di ActionScript diversi dagli oggetti.

Titolo della voce

Tutte le voci sono elencate in ordine alfabetico senza tenere conto dell'uso delle maiuscole, dei caratteri di sottolineatura iniziali e così via.

Sintassi

La sezione “Sintassi” fornisce la sintassi corretta per l'uso degli elementi di ActionScript nel codice. Nella sintassi, il formato della porzione di codice è *carattere codice* e il formato degli argomenti obbligatori è *carattere codice corsivo*. Gli argomenti opzionali sono racchiusi tra parentesi.

Argomenti

Questa sezione descrive gli argomenti elencati nella sintassi.

Descrizione

Questa sezione identifica l'elemento (ad esempio un operatore, un metodo, una funzione o altro elemento) e ne descrive le modalità d'uso.

Lettore

Questa sezione indica le versioni del lettore compatibili con l'elemento. Ciò non si riferisce alla versione di Flash usata per creare il contenuto. Se, ad esempio, si sta creando del contenuto per Flash 4 Player tramite lo strumento Flash 5, non è possibile usare gli elementi di ActionScript disponibili solo per Flash 5 Player.

Dopo l'introduzione di ActionScript di Flash 5, alcuni elementi di ActionScript di Flash 4 e versioni precedenti sono diventati obsoleti. Gli elementi obsoleti sono ancora supportati da Flash 5 Player, ma si consiglia di usare i nuovi elementi di Flash 5.

Le funzionalità degli operatori sono state significativamente ampliate in Flash 5, nel senso che, oltre ad aver introdotto numerosi operatori matematici nuovi, alcuni operatori già esistenti sono ora in grado di gestire tipi di dati aggiuntivi. Per garantire la coerenza del tipo di dati, i file Flash 4 vengono modificati automaticamente al momento dell'importazione in ambiente di creazione Flash 5, senza che tali modifiche incidano sulla funzionalità dello script originale. Per ulteriori informazioni, vedere le voci + (addizione), < (minore di), > (maggiore di), <= (minore o uguale a), >= (maggiore o uguale a), != (disuguaglianza) e = (uguaglianza).

Esempio

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

Vedere anche

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

Voci di esempio per gli oggetti

Di seguito sono descritte le convenzioni usate per gli oggetti predefiniti di ActionScript. Gli oggetti sono elencati in ordine alfabetico con tutti gli altri elementi del dizionario.

Titolo della voce

Il titolo della voce è costituito dal nome dell'oggetto ed è seguito da un paragrafo contenente informazioni di carattere generale sull'oggetto.

Tabelle di riepilogo di metodi e proprietà

Tutte le voci relative agli oggetti contengono una tabella che elenca tutti i metodi associati all'oggetto. Le eventuali proprietà (spesso costanti) di cui dispone l'oggetto sono elencate in una tabella aggiuntiva. A tutti i metodi e le proprietà elencati nelle tabelle corrispondono delle voci di dizionario specifiche che seguono la voce dell'oggetto.

Funzione di costruzione

Se è necessario usare una funzione di costruzione per accedere ai metodi e alle proprietà dell'oggetto, tale funzione è descritta in fondo alla voce dell'oggetto. Nella descrizione della funzione sono contenuti tutti gli elementi standard tipici delle voci di dizionario, quali la descrizione della sintassi e così via.

Lista dei metodi e delle proprietà

I metodi e le proprietà di ciascun oggetto sono elencati in ordine alfabetico dopo la voce dell'oggetto.

Contenuto del dizionario

Tutte le voci del dizionario sono elencate in ordine alfabetico. Gli operatori che sono simboli, tuttavia, vengono elencati secondo l'ordine ASCII. Inoltre, i metodi associati a un oggetto sono elencati insieme al nome dell'oggetto corrispondente, ad esempio, il metodo `abs` dell'oggetto `Math` sarà elencato come `Math.abs`.

Le due tabelle seguenti facilitano la ricerca di tali elementi. La prima tabella elenca gli operatori simbolici nell'ordine in cui vengono presentati nel dizionario. La seconda tabella elenca tutti gli altri elementi di ActionScript.

Nota: per l'ordine di precedenza e l'associatività degli operatori, consultare l'appendice A.

Operatori simbolici

--	(decremento)
++	incremento
!	(NOT logico)
!=	(diseguaglianza)
%	(modulo)
%=	(assegnazione modulo)
&	(AND bit a bit)
&&	(cortocircuito AND)
&=	(assegnazione AND bit a bit)
()	(parentesi tonde)
-	(meno)
*	(moltiplicazione)
*=	(assegnazione moltiplicazione)
,	(virgola)
.	(punto)
? :	(condizionale)
/	(divisione)
//	(delimitatore di commento)
/*	(delimitatore di commento)
/=	(assegnazione divisione)
[]	(accesso matrice)
^	(XOR bit a bit)
^=	(assegnazione XOR bit a bit)
{ }	(operatore di inizializzazione degli oggetti)
	(OR bit a bit)
	(OR logico)
=	(assegnazione OR bit a bit)

Operatori simbolici	
-	(NOT bit a bit)
+	(addizione)
+=	(assegnazione addizione)
<	(minore di)
<<	(spostamento a sinistra bit a bit)
<<=	(spostamento a sinistra bit a bit e assegnazione)
<=	(minore o uguale a)
◇	(disuguaglianza)
=	(assegnazione)
-=	(assegnazione negazione)
==	(uguaglianza)
>	(maggiore di)
>=	(maggiore o uguale a)
>>	(spostamento a destra bit a bit)
>>=	(spostamento a destra bit a bit e assegnazione)
>>>	(spostamento a destra senza segno bit a bit)
>>>=	(spostamento a destra senza segno bit a bit e assegnazione)

La tabella seguente elenca tutti gli elementi di ActionScript che non sono operatori simbolici.

Elemento di ActionScript	Vedere la voce
abs	"Math.abs" a pagina 288
acos	"Math.acos" a pagina 288
add	"add" a pagina 214
and	"and" a pagina 215
_alpha	"_alpha" a pagina 214
appendChild	"XML.appendChild" a pagina 382
Array	"Array (oggetto)" a pagina 215

Elemento di ActionScript	Vedere la voce
asin	"Math.asin" a pagina 289
atan	"Math.atan" a pagina 289
atan2	"Math.atan2" a pagina 289
attachMovie	"MovieClip.attachMovie" a pagina 304
attachSound	"Sound.attachSound" a pagina 348
attributes	"XML.attributes" a pagina 382
BACKSPACE	"Key.BACKSPACE" a pagina 274
Boolean	"Boolean (funzione)" a pagina 225, "Boolean (oggetto)" a pagina 225
break	"break" a pagina 227
call	"call" a pagina 228
CAPSLOCK	"Key.CAPSLOCK" a pagina 274
ceil	"Math.ceil" a pagina 290
charAt	"String.charAt" a pagina 360
charCodeAt	"String.charCodeAt" a pagina 360
childNodes	"XML.childNodes" a pagina 383
chr	"chr" a pagina 228
cloneNode	"XML.cloneNode" a pagina 383
close	"XMLSocket.close" a pagina 399
Color	"Color (oggetto)" a pagina 228
concat	"Array.concat" a pagina 217, "String.concat" a pagina 361
connect	"XMLSocket.connect" a pagina 399
funzione di costruzione	Array, Boolean, Color, Date, Number, Object, Sound, String, XML, XMLSocket
continue	"continue" a pagina 232
CONTROL	"Key.CONTROL" a pagina 274
cos	"Math.cos" a pagina 290
createElement	"XML.createElement" a pagina 384
createTextNode	"XML.createTextNode" a pagina 384

Elemento di ActionScript	Vedere la voce
<code>_currentframe</code>	" <code>_currentframe</code> " a pagina 233
<code>Date</code>	" <code>Date</code> (oggetto)" a pagina 233
<code>delete</code>	" <code>delete</code> " a pagina 251
<code>DELETEKEY</code>	" <code>Key.DELETEKEY</code> " a pagina 275
<code>docTypeDecl</code>	" <code>XML.docTypeDecl</code> " a pagina 385
<code>do...while</code>	" <code>do... while</code> " a pagina 253
<code>DOWN</code>	" <code>Key.DOWN</code> " a pagina 275
<code>_droptarget</code>	" <code>_droptarget</code> " a pagina 253
<code>duplicateMovieClip</code>	" <code>duplicateMovieClip</code> " a pagina 254, " <code>MovieClip.duplicateMovieClip</code> " a pagina 304
<code>E</code>	" <code>Math.E</code> " a pagina 291
<code>else</code>	" <code>else</code> " a pagina 255
<code>END</code>	" <code>Key.END</code> " a pagina 275
<code>ENTER</code>	" <code>Key.ENTER</code> " a pagina 276
<code>eq</code>	" <code>eq</code> (uguale; specifico per stringhe)" a pagina 256
<code>escape</code> (funzione)	" <code>escape</code> " a pagina 256
<code>ESCAPE</code> (costante)	" <code>Key.ESCAPE</code> " a pagina 276
<code>eval</code>	" <code>eval</code> " a pagina 257
<code>evaluate</code>	" <code>evaluate</code> " a pagina 258
<code>exp</code>	" <code>Math.exp</code> " a pagina 291
<code>firstChild</code>	" <code>XML.firstChild</code> " a pagina 386
<code>floor</code>	" <code>Math.floor</code> " a pagina 292
<code>_focusrect</code>	" <code>_focusrect</code> " a pagina 258
<code>for</code>	" <code>for</code> " a pagina 258
<code>for.. in</code>	" <code>for..in</code> " a pagina 260
<code>_framesloaded</code>	" <code>_framesloaded</code> " a pagina 261
<code>fromCharCode</code>	" <code>String.fromCharCode</code> " a pagina 361
<code>fscommand</code>	" <code>fscommand</code> " a pagina 262
<code>function</code>	" <code>function</code> " a pagina 262

Elemento di ActionScript	Vedere la voce
ge	"ge (maggiore di o uguale a; specifico per stringhe)" a pagina 263
getAscii	"Key.getAscii" a pagina 276
getBeginIndex	"Selection.getBeginIndex" a pagina 342
getBounds	"MovieClip.getBounds" a pagina 305
getBytesLoaded	"MovieClip.getBytesLoaded" a pagina 305
getBytesTotal	"MovieClip.getBytesTotal" a pagina 306
getCaretIndex	"Selection.getCaretIndex" a pagina 342
getCode	"Key.getCode" a pagina 277
getDate	"Date.getDate" a pagina 237
getDay	"Date.getDay" a pagina 237
getEndIndex	"Selection.getEndIndex" a pagina 343
getFocus	"Selection.getFocus" a pagina 343
getFullYear	"Date.getFullYear" a pagina 237
getHours	"Date.getHours" a pagina 238
getMilliseconds	"Date.getMilliseconds" a pagina 238
getMinutes	"Date.getMinutes" a pagina 238
getMonth	"Date.getMonth" a pagina 239
getPan	"Sound.getPan" a pagina 348
getProperty	"getProperty" a pagina 264
getRGB	"Color.setRGB" a pagina 230
getSeconds	"Date.getSeconds" a pagina 239
getTime	"Date.getTime" a pagina 239
getTimer	"getTimer" a pagina 264
getTimezoneOffset	"Date.getTimezoneOffset" a pagina 240
getTransform	"Color.getTransform" a pagina 230, "Sound.getTransform" a pagina 348
getURL	"getURL" a pagina 265, "MovieClip.getURL" a pagina 306
getUTCDate	"Date.getUTCDate" a pagina 240

Elemento di ActionScript	Vedere la voce
getUTCDay	"Date.getUTCDay" a pagina 241
getUTCFullYear	"Date.getUTCFullYear" a pagina 241
getUTCHours	"Date.getUTCHours" a pagina 241
getUTCMilliseconds	"Date.getUTCMilliseconds" a pagina 242
getUTCMinutes	"Date.getUTCMinutes" a pagina 242
getUTCMonth	"Date.getUTCMonth" a pagina 242
getUTCSeconds	"Date.getUTCSeconds" a pagina 243
getVersion	"getVersion" a pagina 266
getVolume	"Sound.getVolume" a pagina 349
getYear	"Date.getYear" a pagina 243
globalToLocal	"MovieClip.globalToLocal" a pagina 307
gotoAndPlay	"gotoAndPlay" a pagina 266, "MovieClip.gotoAndPlay" a pagina 308
gotoAndStop	"gotoAndStop" a pagina 267, "MovieClip.gotoAndStop" a pagina 308
gt	"gt (maggiore di; specifico per stringhe)" a pagina 267
hasChildNodes	"XML.hasChildNodes" a pagina 386
_height	"_height" a pagina 268
hide	"Mouse.hide" a pagina 301
_highquality	"_highquality" a pagina 268
hitTest	"MovieClip.hitTest" a pagina 308
HOME	"Key.HOME" a pagina 277
if	"if" a pagina 269
ifFrameLoaded	"ifFrameLoaded" a pagina 269
#include	"#include" a pagina 270
indexOf	"String.indexOf" a pagina 361
Infinity	"Infinity" a pagina 270
INSERT	"Key.INSERT" a pagina 277
insertBefore	"XML.insertBefore" a pagina 387

Elemento di ActionScript	Vedere la voce
int	"int" a pagina 271
isDown	"Key.isDown" a pagina 278
isFinite	"isFinite" a pagina 271
isNaN	"isNaN" a pagina 272
isToggled	"Key.isToggled" a pagina 278
join	"Array.join" a pagina 218
Key	"Key (oggetto)" a pagina 272
lastChild	"XML.lastChild" a pagina 387
lastIndexOf	"String.indexOf" a pagina 361
le	"le (minore di o uguale a; specifico per stringhe)" a pagina 281
LEFT	"Key.LEFT" a pagina 278
length	"length" a pagina 281, "Array.length" a pagina 219, "String.length" a pagina 362
LN2	"Math.LN2" a pagina 293
LN10	"Math.LN10" a pagina 294
load	"XML.load" a pagina 387
loaded	"XML.loaded" a pagina 388
loadMovie	"loadMovie" a pagina 283, "MovieClip.loadMovie" a pagina 310
loadVariables	"loadVariables" a pagina 284, "MovieClip.loadVariables" a pagina 310
localToGlobal	"MovieClip.localToGlobal" a pagina 311
log	"Math.log" a pagina 292
LOG2E	"Math.LOG2E" a pagina 292
LOG10E	"Math.LOG10E" a pagina 293
lt	"le (minore di o uguale a; specifico per stringhe)" a pagina 281
Math	"Math (oggetto)" a pagina 286
max	"Math.max" a pagina 294
maxscroll	"maxscroll" a pagina 298

Elemento di ActionScript	Vedere la voce
MAX_VALUE	"Number.MAX_VALUE" a pagina 322
mbchr	"mbchr" a pagina 299
mblength	"mblength" a pagina 299
mbord	"mbord" a pagina 299
mbsubstring	"mbsubstring" a pagina 300
min	"Math.min" a pagina 294
MIN_VALUE	"Number.MIN_VALUE" a pagina 323
Mouse	"Mouse (oggetto)" a pagina 300
MovieClip	"MovieClip (oggetto)" a pagina 302
_name	"_name" a pagina 316
NaN	"NaN" a pagina 316, "Number.NaN" a pagina 323
ne	"ne (non uguale; specifico per stringhe)" a pagina 316
NEGATIVE_INFINITY	"Number.NEGATIVE_INFINITY" a pagina 323
new (operatore)	"new" a pagina 317
newline	"newline" a pagina 318
nextFrame	"nextFrame" a pagina 318, "MovieClip.nextFrame" a pagina 312
nextScene	"nextScene" a pagina 318
nextSibling	"XML.nextSibling" a pagina 389
nodeName	"XML.nodeName" a pagina 389
nodeType	"XML.nodeType" a pagina 390
nodeValue	"XML.nodeValue" a pagina 390
not	"not" a pagina 319
null	"null" a pagina 319
Number	"Number (funzione)" a pagina 320, "Number (oggetto)" a pagina 320
Object	"Object (oggetto)" a pagina 325
On	"on(mouseEvent)" a pagina 328
onClipEvent	"onClipEvent" a pagina 326

Elemento di ActionScript	Vedere la voce
onClose	"XMLSocket.onClose" a pagina 400
onConnect	"XMLSocket.onConnect" a pagina 401
OnLoad	"XML.onLoad" a pagina 390
onXML	"XMLSocket.onXML" a pagina 402
or (OR logico)	"or" a pagina 329
ord	"ord" a pagina 330
_parent	"_parent" a pagina 330
parentNode	"XML.parentNode" a pagina 391
parseFloat	"parseFloat" a pagina 331
parseInt	"parseInt" a pagina 332
parseXML	"XML.parseXML" a pagina 392
PGDN	"Key.PGDN" a pagina 279
PGUP	"Key.PGUP" a pagina 279
PI	"Math.PI" a pagina 295
play	"play" a pagina 333, "MovieClip.play" a pagina 312
pop	"Array.pop" a pagina 219
POSITIVE_INFINITY	"Number.POSITIVE_INFINITY" a pagina 324
pow	"Math.pow" a pagina 295
prevFrame	"prevFrame" a pagina 333, "MovieClip.prevFrame" a pagina 313
previousSibling	"XML.previousSibling" a pagina 392
prevScene	"prevScene" a pagina 334
print	"print" a pagina 334
printAsBitmap	"printAsBitmap" a pagina 335
push	"Array.push" a pagina 220
_quality	"_quality" a pagina 337
random	"random" a pagina 338, "Math.random" a pagina 296
removeMovieClip	"removeMovieClip" a pagina 338, "MovieClip.removeMovieClip" a pagina 313

Elemento di ActionScript	Vedere la voce
removeNode	"XML.removeNode" a pagina 393
return	"return" a pagina 339
reverse	"Array.reverse" a pagina 220
RIGHT	"Key.RIGHT" a pagina 279
_root	"_root" a pagina 339
_rotation	"_rotation" a pagina 340
round	"Math.round" a pagina 296
scroll	"scroll" a pagina 341
Selection	"Selection (oggetto)" a pagina 341
send	"XML.send" a pagina 393, "XMLSocket.close" a pagina 403
sendAndLoad	"XML.sendAndLoad" a pagina 393
set	"set" a pagina 344
setDate	"Date.setDate" a pagina 243
setFocus	"Selection.setFocus" a pagina 344
setFullYear	"Date.setFullYear" a pagina 244
setHours	"Date.setHours" a pagina 244
setMilliseconds	"Date.setMilliseconds" a pagina 245
setMinutes	"Date.setMinutes" a pagina 245
setMonth	"Date.setMonth" a pagina 245
setPan	"Sound.setPan" a pagina 349
setProperty	"setProperty" a pagina 346
setRGB	"Color.setRGB" a pagina 230
setSeconds	"Date.setSeconds" a pagina 246
setSelection	"Selection.setSelection" a pagina 344
setTime	"Date.setTime" a pagina 246
setTransform	"Color.setTransform" a pagina 231, "Sound.setTransform" a pagina 350
setUTCDate	"Date.setUTCDate" a pagina 246

Elemento di ActionScript	Vedere la voce
setUTCFullYear	"Date.setUTCFullYear" a pagina 247
setUTCHours	"Date.setUTCHours" a pagina 247
setUTCMilliseconds	"Date.setUTCMilliseconds" a pagina 248
setUTCMinutes	"Date.setUTCMinutes" a pagina 248
setUTCMonth	"Date.setUTCMonth" a pagina 248
setUTCSeconds	"Date.setUTCSeconds" a pagina 249
setVolume	"Sound.setVolume" a pagina 353
setYear	"Date.setYear" a pagina 249
shift (metodo)	"Array.shift" a pagina 221
SHIFT (costante)	"Key.SHIFT" a pagina 280
show	"Mouse.show" a pagina 301
sin	"Math.sin" a pagina 296
slice	"Array.splice" a pagina 221, "String.slice" a pagina 362
sort	"Array.sort" a pagina 222
Sound	"Sound (oggetto)" a pagina 346
_soundbuftime	"_soundbuftime" a pagina 354
SPACE	"Key.SPACE" a pagina 280
splice	"Array.splice" a pagina 223
split	"String.split" a pagina 363
sqrt	"Math.sqrt" a pagina 297
SQRT1_2	"Math.SQRT1_2" a pagina 297
SQRT2	"Math.SQRT2" a pagina 297
start	"Sound.start" a pagina 353
startDrag	"startDrag" a pagina 355, "MovieClip.startDrag" a pagina 313
status	"XML.status" a pagina 394
stop	"stop" a pagina 355, "MovieClip.stop" a pagina 314, "Sound.stop" a pagina 354
stopAllSounds	"stopAllSounds" a pagina 356

Elemento di ActionScript	Vedere la voce
stopDrag	"stopDrag" a pagina 356, "MovieClip.stopDrag" a pagina 314
String	"String (funzione)" a pagina 357, "String (oggetto)" a pagina 358, "" " (delimitatore di stringa)" a pagina 358
substr	"String.substr" a pagina 363
substring	"substring" a pagina 365, "String.substring" a pagina 364
swapDepths	"MovieClip.swapDepths" a pagina 315
TAB	"Key.TAB" a pagina 280
tan	"Math.tan" a pagina 298
_target	"_target" a pagina 365
targetPath	"targetPath" a pagina 366
tellTarget	"tellTarget" a pagina 366
this	"this" a pagina 367
toggleHighQuality	"toggleHighQuality" a pagina 368
toLowerCase	"String.toLowerCase" a pagina 364
toString	"Array.toString" a pagina 224, "Boolean.toString" a pagina 226, "Date.toString" a pagina 250, "Number.toString" a pagina 324, "Object.toString" a pagina 326, "XML.toString" a pagina 395
_totalframes	"_totalframes" a pagina 369
toUpperCase	"String.toUpperCase" a pagina 365
trace	"trace" a pagina 369
typeof	"typeof" a pagina 370
unescape	"unescape" a pagina 371
unloadMovie	"unloadMovie" a pagina 371, "MovieClip.unloadMovie" a pagina 315
unshift	"Array.shift" a pagina 221
UP	"Key.UP" a pagina 280
updateAfterEvent	"updateAfterEvent" a pagina 372
_url	"_url" a pagina 373
UTC	"Date.UTC" a pagina 250

Elemento di ActionScript	Vedere la voce
valueOf	"Boolean.valueOf" a pagina 227, "Number.valueOf" a pagina 324, "Object.valueOf" a pagina 326
var	"var" a pagina 373
_visible	"_visible" a pagina 373
void	"void" a pagina 374
while	"while" a pagina 374
_width	"_width" a pagina 375
with	"with" a pagina 376
_x	"_x" a pagina 379
XML	"XML (oggetto)" a pagina 379
xmlDecl	"XML.xmlDecl" a pagina 395
XMLSocket	"XMLSocket (oggetto)" a pagina 396
_xmouse	"_xmouse" a pagina 404
_xscale	"_xscale" a pagina 404
_y	"_y" a pagina 405
_ymouse	"_ymouse" a pagina 406
_yscale	"_yscale" a pagina 406

-- (decremento)

Sintassi

```
--espressione
espressione--
```

Argomenti

espressione Variabile, numero, elemento di una matrice o proprietà di un oggetto.

Descrizione

Operatore unario di decremento prima dell'operazione e dopo l'operazione che sottrae 1 dal valore di *espressione*. L'operatore di decremento prima dell'operazione (*--espressione*) sottrae 1 dal valore di *espressione* e restituisce il risultato. L'operatore di decremento dopo l'operazione (*espressione--*) sottrae 1 dal valore di *espressione* e restituisce il valore iniziale di *espressione* (il risultato prima della sottrazione).

Letttore

Flash 4 o versione successiva.

Esempio

L'operatore di decremento prima dell'operazione cambia il valore di x in 2 ($x - 1 = 2$) e restituisce il risultato come y :

```
x = 3;
```

```
y = --x
```

L'operatore di decremento dopo l'operazione cambia il valore di x in 2 ($x - 1 = 2$) e restituisce il valore originale ($x = 3$) come risultato y :

```
Se x = 3;
```

```
y = x--
```

++ (incremento)

Sintassi

++espressione

espressione++

Argomenti

espressione Variabile, numero, elemento di una matrice o proprietà di un oggetto.

Descrizione

Operatore unario di incremento prima dell'operazione e dopo l'operazione che aggiunge 1 al valore di *espressione*. L'operatore di incremento prima dell'operazione (*++espressione*) aggiunge 1 al valore di *espressione* e restituisce il risultato. L'operatore di incremento dopo l'operazione (*espressione++*) aggiunge 1 al valore di *espressione* e restituisce il valore iniziale di *espressione* (il risultato prima dell'addizione).

L'operatore di incremento prima dell'operazione incrementa il valore di x a 2 ($x + 1 = 2$) e restituisce il risultato come y :

```
x = 1;
```

```
y = ++x
```

L'operatore di incremento dopo l'operazione incrementa il valore di x a 2 ($x + 1 = 2$) e restituisce il valore originale ($x = 1$) come risultato y :

```
x = 1;
```

```
y = x++
```

Letttore

Flash 4 o versione successiva.

Esempio

L'esempio seguente usa ++ come operatore di incremento prima dell'operazione con un'istruzione while.

```
i = 0
while(i++ < 5){
// Questa sezione verrà eseguita cinque volte
}
```

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

```
var a = [];
var i = 0;
while (i < 10) {
    a.push(++i);
}
trace(a.join());
```

Il risultato stampato di questo script è il seguente:

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

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

```
var a = [];
var i = 0;
while (i < 10) {
    a.push(i++);
}
trace(a.join());
```

Il risultato stampato di questo script è il seguente:

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

! (NOT logico)

Sintassi

!espressione

Argomenti

espressione Variabile o espressione valutata.

Descrizione

Operatore (logico); inverte il valore booleano di una variabile o di un'espressione. Se *espressione* è una variabile con un valore assoluto o convertito true, *!variabile* il valore di *! espressione* è false. Se l'espressione *x && y* restituisce false, l'espressione *!(x && y)* restituisce true. Questo operatore è uguale all'operatore not usato in Flash 4.

Lettore

Flash 4 o versione successiva.

Esempio

Nell'esempio seguente la variabile `happy` è impostata su `false`, la condizione `if` valuta la condizione `!happy`, e se la condizione è `true`, `trace` invia una stringa alla finestra `Output`.

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

L'esempio seguente illustra i risultati dell'operatore `!`:

`! true` restituisce `false`

`! false` restituisce `true`

!=(diseguaglianza)

Sintassi

espressione1 `!=` *espressione2*

Argomenti

espressione1, *espressione2* Numeri, stringhe, valori booleani, variabili, oggetti, matrici o funzioni.

Descrizione

Operatore (eguaglianza); verifica l'esatto opposto dell'operatore `==`. Se *espressione1* è uguale a *espressione2*, il risultato è `false`. Come nel caso dell'operatore `==`, la definizione di *uguale* dipende dal tipo di dati che vengono confrontati.

- Numeri, stringhe e valori booleani vengono confrontati come valore.
- Variabili, oggetti, matrici e funzioni vengono confrontati come riferimento.

Letture

Flash 5 o versione successiva.

Esempio

L'esempio seguente illustra i risultati dell'operatore `!=`:

`5 != 8` restituisce `true`

`5 != 5` restituisce `false`

L'esempio seguente illustra l'uso dell'operatore `!=` in un'istruzione `if`:

```
a = "David";
b = "Fool"
if (a != b)
    trace("David is not a fool");
```

Vedere anche

`==` (uguaglianza) a pagina 208

% (modulo)

Sintassi

espressione1 % *espressione2*

Argomenti

espressione1, *espressione2* Numeri, numeri interi, numeri in virgola mobile o stringhe che vengono convertite in valori numerici.

Descrizione

Operatore (aritmetico); calcola il resto di *espressione1* diviso per *espressione2*. Se *espressione* contiene argomenti che non sono valori numerici, l'operatore modulo tenta di convertirli in numeri.

Letto

Flash 4 o versione successiva. Nei file Flash 4, l'operatore % viene espanso nel file SWF come operatore $x - \text{int}(x/y) * y$ e il risultato potrebbe non essere veloce e preciso come in ambiente Flash 5 Player.

Esempio

Di seguito è riportato un esempio numerico dell'uso dell'operatore % :

12 % 5 restituisce 2

4,3 % 2,1 restituisce 0,1

%= (assegnazione modulo)

Sintassi

espressione1 %= *espressione2*

Argomenti

espressione1, *espressione2* Numeri interi e variabili.

Descrizione

Operatore (assegnazione); assegna a *espressione1* il valore di *espressione1* % *espressione2*.

Letto

Flash 4 o versione successiva.

Esempio

L'esempio seguente illustra l'uso dell'operatore %= con variabili e numeri:

$x \text{ \%} = y$ equivale a $x = x \text{ \%} y$

Se $x = 14$ e $y = 5$ allora

$x \text{ \%} = 5$ restituisce 4

Vedere anche

“% (modulo)” a pagina 186

& (AND bit a bit)

Sintassi

espressione1 & *espressione2*

Argomenti

espressione1, *espressione2* Un numero.

Descrizione

Operatore (bit a bit); converte *espressione1* ed *espressione2* in interi a 32 bit senza segno ed esegue un'operazione AND booleana su ciascun bit degli argomenti convertiti in interi. Il risultato è un numero intero a 32 bit senza segno.

Letture

Flash 5 o versione successiva. In Flash 4 l'operatore & veniva usato per concatenare le stringhe. In Flash 5 & è l'operatore AND bit a bit e gli operatori add e + concatenano le stringhe. I file Flash 4 che usano l'operatore & vengono aggiornati automaticamente per l'uso di add al momento dell'importazione in ambiente di creazione di codice di Flash 5.

&& (cortocircuito AND)

Sintassi

espressione1 && *espressione2*

Argomenti

espressione1, *espressione2* Numeri, stringhe, variabili o funzioni.

Descrizione

Operatore (logico); esegue un'operazione booleana sui valori di una o entrambe le espressioni. Determina la valutazione da parte dell'interprete Flash di *espressione1* (l'espressione posta a sinistra) e il risultato è *false* se l'espressione restituisce *false*. Se *espressione1* restituisce *true*, *espressione2* (l'espressione posta a destra) viene valutata. Se *espressione2* restituisce *true*, il risultato finale è *true*; in caso contrario, sarà *false*.

Letture

Flash 4 o versione successiva.

Esempio

L'esempio seguente assegna alle variabili `winner` e `loser` i valori delle espressioni valutate per eseguire una verifica:

```
winner = (chocolateEggs >=10) && (jellyBeans >=25);
loser = (chocolateEggs <=1) && (jellyBeans <= 5);
if (winner) {
    alert = "You Win the Hunt!";
    if (loser) {
        alert = "Now THAT'S Unhappy Hunting!";
    }
} else {
    alert = "We're all winners!";
}
```

&= (assegnazione AND bit a bit)

Sintassi

espressione1 &= *espressione2*

Argomenti

espressione1, *espressione2* Numeri interi e variabili.

Descrizione

Operatore (assegnazione bit a bit); assegna a *espressione1* il valore di *espressione1* & *espressione2*.

Lettore

Flash 5 o versione successiva.

Esempio

L'esempio seguente illustra l'uso dell'operatore &= con variabili e numeri:

`x &= y` equivale a `x = x & y`

Se `x = 15` e `y = 9` allora

`x &= 9` restituisce 9

Vedere anche

“& (AND bit a bit)” a pagina 187

() (parentesi tonde)

Sintassi

(espressione1, espressione2);

funzione(chiamataFunzione1, ..., chiamataFunzioneN);

Argomenti

espressione1, *espressione2* Numeri, stringhe, variabili o testo.

funzione La funzione da eseguire sul contenuto racchiuso tra parentesi tonde.

chiamataFunzione1...chiamataFunzioneN Una serie di funzioni da eseguire prima di passare il risultato alla funzione fuori dalle parentesi tonde.

Descrizione

Operatore (generale); esegue un'operazione di raggruppamento su uno o più argomenti oppure delimita uno o più argomenti, quindi passa i risultati sotto forma di parametro a una funzione posta all'esterno delle parentesi tonde.

Uso 1: esegue un'operazione di raggruppamento su una o più espressioni in modo da controllare l'ordine di esecuzione degli operatori nell'espressione. Questo operatore ridefinisce l'ordine di precedenza automatico e determina che vengano valutate per prime le espressioni racchiuse tra parentesi. Nel caso di parentesi annidate, Flash valuta prima il contenuto delle parentesi più interne, quindi il contenuto delle parentesi più esterne.

Uso 2: delimita uno o più argomenti per poi passarli sotto forma di parametri alla funzione posta all'esterno delle parentesi tonde.

Lettore

Flash 4 o versione successiva.

Esempio

(Uso 1) Le seguenti istruzioni illustrano l'uso delle parentesi tonde per controllare l'ordine di valutazione delle espressioni. Il risultato è visualizzato ad ogni istruzione.

```
(2 + 3) * (4 + 5)
45
2 + (3 * (4 + 5))
29
2 + (3 * 4) + 5
19
```

(Uso 2) L'esempio seguente illustra l'uso delle parentesi tonde con una funzione:

```
getDate();
invoice(item, amount);
```

Vedere anche

"with" a pagina 376

- (meno)

Sintassi

(Negazione) *-espressione*

(Sottrazione) *espressione1 - espressione2*

Argomenti

espressione1, *espressione2* Un numero.

Descrizione

Operatore (aritmetico); usato per la negazione o la sottrazione. Quando viene usato per la negazione, inverte il segno dell'*espressione* numerica. Quando viene usato per la sottrazione, esegue un'operazione aritmetica su due espressioni numeriche, ossia sottrae *espressione2* da *espressione1*. Se le due espressioni sono costituite da numeri interi, la differenza sarà un numero intero. Se una o entrambe le espressioni sono costituite da numeri in virgola mobile, la differenza sarà un numero in virgola mobile.

Lettore

Flash 4 o versione successiva.

Esempio

(Negazione): questa istruzione inverte il segno dell'espressione $2 + 3$:

$-(2 + 3)$

Il risultato è -5.

(Sottrazione): questa istruzione sottrae il numero intero 2 dal numero intero 5:

$5 - 2$

Il risultato è 3, un numero intero.

(Sottrazione): questa istruzione sottrae il numero in virgola mobile 1,5 dal numero in virgola mobile 3,25:

$put\ 3,25 - 1,5$

Il risultato è 1,75, un numero in virgola mobile.

***** (moltiplicazione)

Sintassi

espressione1 * *espressione2*

Argomenti

espressione1, *espressione2* Numeri interi o in virgola mobile.

Descrizione

Operatore (aritmetico); moltiplica due espressioni numeriche. Se le due espressioni sono costituite da numeri interi, il prodotto sarà un numero intero. Se una o entrambe le espressioni sono costituite da numeri in virgola mobile, il prodotto sarà un numero in virgola mobile.

Lettore

Flash 4 o versione successiva.

Esempio

Questa istruzione moltiplica tra loro i numeri interi 2 e 3:

`2 * 3`

Il risultato è 6, un numero intero.

Esempio

Questa istruzione moltiplica tra loro i numeri in virgola mobile 2,0 e 3,1416:

`2.0 * 3.1416`

Il risultato è 6,2832, un numero in virgola mobile.

***= (assegnazione moltiplicazione)**

Sintassi

espressione1 *= *espressione2*

Argomenti

espressione1, *espressione2* Numeri interi, in virgola mobile o stringhe.

Descrizione

Operatore (assegnazione); assegna a *espressione1* il valore di *espressione1***espressione2*.

Lettore

Flash 4 o versione successiva.

Esempio

L'esempio seguente illustra l'uso dell'operatore *=con variabili e numeri:

`x *= y` equivale a `x = x * y`

Se `x = 5` e `y = 10` allora

`x *= 10` restituisce 50

Vedere anche

“* (moltiplicazione)” a pagina 190

, (virgola)

Sintassi

espressione1 % *espressione2*

Argomenti

espressione Numero, variabile, stringa, elemento di matrice o altri dati.

Descrizione

Operatore; ordina a Flash di valutare *espressione1*, quindi *espressione2* e di restituire il valore di *espressione2*. Questo operatore si usa principalmente con l'istruzione del ciclo `for`.

Lettore

Flash 4 o versione successiva.

Esempio

Nel codice seguente viene usato l'operatore virgola:

```
var a=1, b=2, c=3;
```

Ciò equivale a scrivere quanto segue:

```
var a=1;  
var b=2;  
var c=3;
```

. (operatore punto)

Sintassi

oggetto.proprietà_o_metodo

nomeistanza.variabile

nomeistanza.istanzasecondaria.variabile

Argomenti

oggetto Istanza di un oggetto. Alcuni oggetti richiedono l'uso della funzione di costruzione per creare le relative istanze. Può trattarsi di un oggetto predefinito di ActionScript o di un oggetto personalizzato. Questo argomento viene sempre posto a sinistra dell'operatore punto (`.`).

proprietà_o_metodo Il nome di una proprietà o di un metodo associato all'oggetto. I metodi e le proprietà validi per gli oggetti predefiniti sono elencati nelle relative tabelle di riepilogo di metodi e proprietà. Questo argomento viene sempre posto a destra dell'operatore punto (`.`).

nomeistanza Il nome di un'istanza di clip filmato.

istanzasecondaria Istanza di un clip filmato secondario del clip filmato principale.

variabile Variabile in un clip filmato.

Descrizione

Operatore; usato per esplorare la gerarchia dei clip filmati in modo da avere accesso ai clip filmati secondari annidati, alle variabili o alle proprietà. L'operatore punto consente anche di verificare o impostare le proprietà di un oggetto, eseguirne un metodo oppure creare una struttura di dati.

Lettore

Flash 4 o versione successiva.

Vedere anche

“[] (operatore di accesso matrice)” a pagina 196

Esempio

Questa istruzione identifica il valore corrente della variabile `hairColor` in base al clip filmato `person`:

```
person.hairColor
```

Ciò equivale alla seguente sintassi di Flash 4:

```
/person:hairColor
```

Esempio

Il codice seguente illustra come l'operatore punto possa essere usato per creare la struttura di una matrice:

```
account.name = "Gary Smith";  
account.address = "123 Main St ";  
account.city = "Any Town";  
account.state = "CA";  
account.zip = "12345";
```

?: (condizionale)

Sintassi

espressione1 ? *espressione2* : *espressione3*

Argomenti

espressione1 Espressione che restituisce un valore booleano, di norma un'espressione di confronto.

espressione2, *espressione3* Valori di qualsiasi tipo.

Descrizione

Operatore (condizionale); indica a Flash di valutare *espressione1*, e di restituire il valore di *espressione2* se *espressione1* è true; in caso contrario, di restituire il valore di *espressione3*.

Lettore

Flash 4 o versione successiva.

/ (divisione)

Sintassi

espressione1 / *espressione2*

Argomenti

espressione Un numero.

Descrizione

Operatore (aritmetico); divide *espressione1* per *espressione2*. Gli argomenti dell'espressione e i risultati dell'operazione di divisione, vengono valutati ed espressi come numeri in virgola mobile a doppia precisione.

Lettore

Flash 4 o versione successiva.

Esempio

Questa istruzione divide il numero in virgola mobile 22,0 per 7,0, quindi visualizza il risultato nella finestra Output:

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

Il risultato è 3,1429, un numero in virgola mobile.

// (delimitatore di commento)

Sintassi

```
// commento
```

Argomenti

commento Testo che non fa parte del codice e che deve essere ignorato dall'interprete.

Descrizione

Commento; identifica l'inizio di un commento nello script. Il testo racchiuso tra il delimitatore di commento // e il carattere di fine riga viene riconosciuto come commento e quindi ignorato dall'interprete di ActionScript.

Lettore

Flash 1 o versione successiva.

Esempio

In questo script le barre inclinate dei delimitatori di commento consentono di identificare la prima, la terza, la quinta e la settima riga come righe di commento:

```
// Imposta la posizione X del clip filmato ball
ball = getProperty(ball._x);
// Imposta la posizione Y del clip filmato ball
ball = getProperty(ball._y);
// Imposta la posizione X del clip filmato kitty
kitty = getProperty(kitty._x);
// Imposta la posizione Y del clip filmato kitty
kitty_y = getProperty(kitty._y);
```

Vedere anche

“/* (delimitatore di commento)” a pagina 195

/* (delimitatore di commento)

Sintassi

```
/* commento */  
/*  
* commento  
* commento  
*/
```

Argomenti

commento Testo.

Descrizione

Commento; indica una o più righe di commento nello script. Il testo racchiuso tra il tag di apertura del commento */** e il tag di chiusura del commento **/* viene riconosciuto come commento e quindi ignorato dall'interprete di ActionScript. La prima sintassi consente di identificare singole righe di commento, mentre la seconda sintassi consente di identificare commenti disposti su più righe consecutive. Se il tag di chiusura **/* viene omissso quando si usa questo tipo di delimitatore di commento, ActionScript restituisce un messaggio di errore.

Lettore

Flash 5 o versione successiva.

Vedere anche

“// (delimitatore di commento)” a pagina 194

/= (assegnazione divisione)

Sintassi

```
espressione1 /= espressione2
```

Argomenti

espressione1, espressione2 Numeri interi, in virgola mobile o stringhe.

Descrizione

Operatore (assegnazione); assegna a *espressione1* il valore di *espressione1 / espressione2*.

Lettore

Flash 4 o versione successiva.

Esempio

L'esempio seguente illustra l'uso dell'operatore */=* con variabili e numeri:

```
x /= y equivale a x = x / y  
x = 10;  
y = 2;  
x /= y;  
// x adesso contiene il valore 5
```

[] (operatore di accesso matrice)

Sintassi

nomeMatrice["a0", "a1", ... "aN"];

oggetto[*valore1*, *valore2*, ... *valoreN*];

Argomenti

nomeMatrice Il nome di una matrice.

a0, *a1*, ... *aN* Elementi in una matrice.

valore1, *valore2*, ... *valoreN* Nomi di proprietà.

Descrizione

Operatore; crea un nuovo oggetto iniziando le proprietà specificate negli argomenti oppure inizializza una nuova matrice con gli elementi (*a0*) specificati negli argomenti.

Il prototipo dell'oggetto creato è l'oggetto `Object` generico. L'uso di questo operatore equivale a chiamare `new Object` e assegnare valori alle proprietà tramite l'operatore di assegnazione. Rappresenta inoltre un'alternativa all'uso dell'operatore `new` che consente di creare gli oggetti in modo veloce e semplice.

Letture

Flash 4 o versione successiva.

Esempio

Gli esempi di codice seguenti rappresentano due modi diversi di creare un nuovo oggetto `Array` vuoto.

```
myArray = [];  
myArray = new Array();
```

Il seguente è un esempio di matrice semplice:

```
myArray = ["red", "orange", "yellow", "green", "blue", "purple"]  
myArray[0]="red"  
myArray[1]="yellow"  
myArray[2]="green"  
myArray[3]="blue"  
myArray[4]="purple"
```

^(XOR bit a bit)

Sintassi

espressione1 ^ *espressione2*

Argomenti

espressione1, *espressione2* Un numero.

Descrizione

Operatore (bit a bit); converte *espressione1* ed *espressione2* in numeri interi a 32 bit senza segno, quindi restituisce 1 nella posizione di ciascun bit dove il bit corrispondente in *espressione1* o *espressione2*, non entrambi, equivale a 1.

Letture

Flash 5 o versione successiva.

Esempio

15 ^ 9 restituisce 6
(1111 ^ 1001 = 0110)

^= (assegnazione XOR bit a bit)

Sintassi

espressione1 ^= *espressione2*

Argomenti

espressione1, *espressione2* Numeri interi e variabili.

Descrizione

Operatore (assegnazione composta); assegna a *espressione1* il valore di *espressione1* ^ *espressione2*.

Letture

Flash 5 o versione successiva.

Esempio

Il seguente è un esempio di operazione ^=:

```
// 15 notazione decimale = 1111 notazione binaria
x = 15;
// 9 notazione decimale = 1001 notazione binaria
x ^= y;
restituisce
x ^ y (0110 in notazione binaria)
```

L'esempio seguente illustra l'uso dell'operatore ^=con variabili e numeri:

```
x ^= y equivale a x = x ^ y
Se x = 15 e y = 9 allora
15 ^= 9 restituisce 6
```

Vedere anche

“(XOR bit a bit)” a pagina 197

{} (operatore di inizializzazione degli oggetti)

Sintassi

```
oggetto {nome1: valore1,  
        nome1: valore2,  
        ...  
        nomeN: valoreN };
```

Argomenti

oggetto L'oggetto da creare.

nome1, nome2, ... nomeN Il nome della proprietà.

valore1, valore2, ... valoreN Il valore corrispondente per ciascuna proprietà *nome*.

Descrizione

Operatore; crea un nuovo oggetto e lo inizializza con la coppia di proprietà *nome* e *valore* specificata. Il prototipo dell'oggetto creato è l'oggetto `Object` generico. L'uso di questo operatore equivale a chiamare `new Object` e assegnare valori alle coppie di proprietà tramite l'operatore di assegnazione. Rappresenta inoltre un'alternativa all'uso dell'operatore `new` che consente di creare gli oggetti in modo veloce e semplice.

Letture

Flash 5 o versione successiva.

Esempio

Il codice seguente illustra come creare un oggetto vuoto tramite l'operatore di inizializzazione degli oggetti e usando `new Object`.

```
object = {};  
object = new Object();
```

Il codice seguente crea un oggetto `account` e inizializza le proprietà `name`, `address`, `city`, `state`, `zip` e `balance`.

```
account = { name: "John Smith",  
            address: "123 Main Street",  
            city: "Blossomville",  
            state: "California",  
            zip: "12345",  
            balance: "1000" };
```

L'esempio seguente illustra come si possano annidare gli operatori di inizializzazione di matrici e oggetti l'uno nell'altro.

```
person = { name: "Peter Piper",  
            children: [ "Jack", "Jill", "Moe", ] };
```

L'esempio seguente illustra un uso diverso delle informazioni riportate nell'esempio precedente, producendo lo stesso risultato.

```
person = new Person();
person.name = 'John Smith';
person.children = new Array();
person.children[0] = 'Jack';
person.children[1] = 'Jill';
person.children[2] = 'Moe';
```

Vedere anche

“[] (operatore di accesso matrice)” a pagina 196

“new” a pagina 317

“Object (oggetto)” a pagina 325

| (OR bit a bit)

Sintassi

espressione1 | *espressione2*

Argomenti

espressione1, *espressione2* Un numero.

Descrizione

Operatore (bit a bit); converte *espressione1* ed *espressione2* in numeri interi a 32 bit senza segno e restituisce 1 nella posizione di ciascun bit dove il bit corrispondente di *espressione1* o *espressione2* equivale a 1.

Letture

Flash 5 o versione successiva.

Esempio

Il seguente è un esempio di operazione OR bit a bit. Si noti che 15 equivale al numero binario 1111.

```
// 15 notazione decimale = 1111 notazione binaria
x = 15;
// 9 notazione decimale = 1001 notazione binaria
y = 9;
// x | y = valore binario
z = x | y;
z = 15
```

Il precedente esempio può essere rappresentato nel modo seguente:

```
15 | 9 restituisce 15
(1111 | 0011 = 1111)
```

|| (OR)

Sintassi

espressione1 || *espressione2*

Argomenti

espressione1, *espressione2* Un valore booleano o un'espressione che restituisce un valore booleano.

Descrizione

Operatore (logico); valuta *espressione1* ed *espressione2*. Il risultato è `true` se una o entrambe le espressioni restituiscono `true`; il risultato è `false` solo se entrambe le espressioni restituiscono `false`.

Con espressioni non booleane, l'operatore OR logico indica a Flash di valutare l'espressione posta a sinistra. Se questa può essere convertita in `true` il risultato è `true`. In caso contrario, verrà valutata l'espressione posta a destra e il risultato sarà il valore di tale espressione.

Letture

Flash 4 o versione successiva.

Esempio

L'esempio seguente usa l'operatore || in un'istruzione `if`:

```
want = true;
need = true;
love = false;
if (want || need || love){
    trace("two out of 3 ain't bad");
}
```

|= (assegnazione OR bit a bit)

Sintassi

espressione1 |= *espressione2*

Argomenti

espressione1, *espressione2* Numeri interi e variabili.

Descrizione

Operatore (assegnazione); assegna a *espressione1* il valore di *espressione1* | *espressione2*.

Letture

Flash 5 o versione successiva.

Esempio

L'esempio seguente illustra l'uso dell'operatore |= con variabili e numeri:

```
x |= y equivale a x = x | y  
Se x = 15 e y = 9 allora  
x |= 9 restituisce 15
```

Vedere anche

“| (OR bit a bit)” a pagina 199

~ (NOT bit a bit)

Sintassi

~ espressione

Argomenti

espressione Un numero.

Descrizione

Operatore (bit a bit); converte *espressione* in un intero a 32 bit senza segno, quindi inverte i bit. In altre parole, cambia il segno di un numero e sottrae 1.

Un'operazione NOT bit a bit cambia il segno di un numero e sottrae 1.

Letture

Flash 5 o versione successiva.

Esempio

La seguente è una rappresentazione numerica di un'operazione NOT bit a bit eseguita su una variabile:

```
~a, restituisce -1 se a = 0 e restituisce -2 se a = 1, ossia:  
~0=-1 e ~1=-2
```

+ (addizione)

Sintassi

espressione1 + espressione2

Argomenti

espressione1, espressione2 Numeri interi, numeri, numeri in virgola mobile o stringhe.

Descrizione

Operatore; addiziona espressioni numeriche o concatena le stringhe. Se un'espressione è una stringa, tutte le altre espressioni vengono convertite in stringhe e concatenate.

Se entrambe le espressioni sono numeri interi, la loro somma sarà un numero intero; se una o entrambe le espressioni sono numeri in virgola mobile, la loro somma sarà un numero in virgola mobile.

Letttore

Flash 4; Flash 5 o versione successiva. In Flash 5, + è un operatore numerico o un operatore di concatenazione delle stringhe a seconda del tipo di dati dell'argomento. In Flash 4 + è solo un operatore numerico. I file di Flash 4 importati in ambiente di creazione di Flash 5 vengono sottoposti a un processo di conversione per mantenere l'integrità del tipo di dati. Il primo esempio seguente illustra il processo di conversione.

Esempio

L'esempio seguente illustra la conversione di un file Flash 4 che contiene un'operazione numerica che richiede un confronto del tipo.

File Flash 4:

```
x + y
```

File Flash 5 convertito:

```
Number(x) + Number(y)
```

La seguente istruzione addiziona gli interi 2 e 3, quindi visualizza nella finestra Output il numero intero 5 risultante:

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

La seguente istruzione addiziona i numeri in virgola mobile 2,5 e 3,25, quindi visualizza nella finestra Output il numero in virgola mobile 5,7500 risultante:

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

La seguente istruzione concatena due stringhe e visualizza il risultato, "today is my birthday," nella finestra Output:

```
"today is my" + "birthday"
```

Vedere anche

"add" a pagina 214

+= (assegnazione addizione)

Sintassi

```
espressione1 += espressione2
```

Argomenti

espressione1, *espressione2* Numeri interi, in virgola mobile o stringhe.

Descrizione

Operatore (assegnazione composta); assegna a *espressione1* il valore di *espressione1* + *espressione2*. Questo operatore consente anche di concatenare le stringhe.

Letttore

Flash 4 o versione successiva.

Esempio

Di seguito è riportato un esempio numerico dell'uso dell'operatore +=:

```
x += y equivale a x = x + y
Se x = 5 e y = 10 allora
x += 10 restituisce 15
```

L'esempio seguente illustra l'uso dell'operatore += con un'espressione stringa:

```
x = "My name is"
x += "Mary"
```

Il risultato di tale codice è il seguente:

```
"My name is Mary"
```

Vedere anche

“+ (addizione)” a pagina 201

< (minore di)

Sintassi

espressione1 < *espressione2*

Argomenti

espressione1, *espressione2* Numeri o stringhe.

Descrizione

Operatore (confronto); confronta due espressioni e determina se *espressione1* è minore di *espressione2* (true) oppure se *espressione1* è maggiore o uguale a *espressione2* (false). Le espressioni stringa vengono valutate e confrontate in base al numero di caratteri che compongono la stringa.

Letture

Flash 4; Flash 5 o versione successiva. In Flash 5 < è un operatore di confronto in grado di gestire diversi tipi di dati. In Flash 4 < è un operatore numerico. I file di Flash 4 importati in ambiente di creazione di Flash 5 vengono sottoposti a un processo di conversione per mantenere l'integrità del tipo di dati. Il primo esempio seguente illustra il processo di conversione.

Esempio

L'esempio seguente illustra la conversione di un file Flash 4 che contiene un'operazione numerica che richiede un confronto del tipo.

File Flash 4:

```
x < y
```

File Flash 5 convertito:

```
Number(x) < Number(y)
```

L'esempio seguente illustra due casi in cui i valori restituiti sono `true` e `false` per numeri e stringhe:

```
3 < 10 o "Al" < "Jack" restituisce true
10 < 3 o "Jack" < "Al" restituisce false
```

« (spostamento a sinistra bit a bit)

Sintassi

espressione1 << *espressione2*

Argomenti

espressione1 Numero, stringa o espressione da spostare a sinistra.

espressione2 Numero, stringa o espressione che viene convertita in un numero intero da 0 a 31.

Descrizione

Operatore (bit a bit); converte *espressione1* ed *espressione2* in interi a 32 bit, quindi sposta a sinistra tutti i bit contenuti in *espressione1* di un numero di posizioni corrispondente al numero intero risultante dalla conversione di *espressione2*. Se al termine dell'operazione rimangono delle posizioni vuote, al loro posto viene inserito uno 0. Lo spostamento di un valore a sinistra di 1 posizione equivale alla sua moltiplicazione per 2.

Letture

Flash 5 o versione successiva.

Esempio

L'esempio seguente sposta l'intero 1 a sinistra di dieci bit.

```
x = 1 << 10
```

Il risultato di questa operazione è $x = 1024$. Ciò è dovuto al fatto che il numero decimale 1 equivale al numero binario 1, il numero binario 1 spostato a sinistra di 10 bit equivale al numero binario 10000000000 e quest'ultimo equivale al numero decimale 1024.

L'esempio seguente sposta l'intero 7 a sinistra di otto bit.

```
x = 7 << 8
```

Il risultato di questa operazione è $x = 1792$. Ciò è dovuto al fatto che il numero decimale 7 equivale al numero binario 111, il numero binario 111 spostato a sinistra di 8 bit equivale al numero binario 11100000000 e quest'ultimo equivale al numero decimale 1792.

Vedere anche

“>>= (spostamento a destra bit a bit e assegnazione)” a pagina 211

<<= (spostamento a sinistra bit a bit e assegnazione)

Sintassi

espressione1 <<= *espressione2*

Argomenti

espressione1 Numero, stringa o espressione da spostare a sinistra.

espressione2 Numero, stringa o espressione che viene convertita in un numero intero da 0 a 31.

Descrizione

Operatore (assegnazione composta); esegue un'operazione di spostamento a sinistra bit a bit e memorizza il contenuto come risultato in *espressione1*.

Lettore

Flash 5 o versione successiva.

Esempio

Le due espressioni seguenti si equivalgono:

A <<= B

A = (A << B)

Vedere anche

“<< (spostamento a sinistra bit a bit)” a pagina 204

“>>= (spostamento a destra bit a bit e assegnazione)” a pagina 211

<= (minore o uguale a)

Sintassi

espressione1 <= *espressione2*

Argomenti

espressione1, *espressione2* Numeri o stringhe.

Descrizione

Operatore (confronto); confronta due espressioni e determina se *espressione1* è minore o uguale a *espressione2* (true) oppure se *espressione1* è maggiore di *espressione2* (false).

Lettore

Flash 4; Flash 5 o versione successiva. In Flash 5 <= è un operatore di confronto in grado di gestire diversi tipi di dati. In Flash 4 <= è un operatore numerico. I file di Flash 4 importati in ambiente di creazione di Flash 5 vengono sottoposti a un processo di conversione per mantenere l'integrità del tipo di dati. Il primo esempio seguente illustra il processo di conversione.

Esempio

L'esempio seguente illustra la conversione di un file Flash 4 che contiene un'operazione numerica che richiede un confronto del tipo.

File Flash 4:

```
x <= y
```

File Flash 5 convertito:

```
Number(x) <= Number(y)
```

L'esempio seguente illustra due casi in cui i valori restituiti sono `true` e `false` per numeri e stringhe:

```
5 <= 10 o "Al" <= "Jack" restituisce true  
10 <= 5 o "Jack" <= "Al" restituisce false
```

⌵ (disuguaglianza)

Sintassi

espressione1 <> *espressione2*

Argomenti

espressione1, *espressione2* Numeri, stringhe, valori booleani, variabili, oggetti, matrici o funzioni.

Descrizione

Operatore (eguaglianza); verifica l'esatto opposto dell'operatore `==`. Se *espressione1* è uguale a *espressione2*, il risultato è `false`. Come nel caso dell'operatore `==`, la definizione di *uguale* dipende dal tipo di dati che vengono confrontati.

- Numeri, stringhe e valori booleani vengono confrontati come valore.
- Variabili, oggetti, matrici e funzioni vengono confrontati come riferimento.

Questo operatore è diventato obsoleto in Flash 5 ed è consigliato l'uso del nuovo operatore `!=`.

Lettore

Flash 2 o versione successiva.

Vedere anche

`!=(disuguaglianza)` a pagina 185

= (assegnazione)

Sintassi

espressione1 = *espressione2*

Argomenti

espressione1 Variabile, elemento di una matrice o proprietà di un oggetto.

espressione2 Un valore di qualsiasi tipo.

Descrizione

Operatore (assegnazione); assegna il tipo di valore di *espressione2* (l'argomento posto a destra) alla variabile, all'elemento di matrice o alla proprietà in *espressione1*.

Letture

Flash 4; Flash 5 o versione successiva. In Flash 5 = è un operatore di assegnazione e l'operatore == viene usato per valutare l'uguaglianza. In Flash 4 = è un operatore di uguaglianza numerica. I file di Flash 4 importati in ambiente di creazione di Flash 5 vengono sottoposti a un processo di conversione per mantenere l'integrità del tipo di dati. Il primo esempio seguente illustra il processo di conversione.

Esempio

L'esempio seguente illustra la conversione di un file Flash 4 che contiene un'operazione numerica che richiede un confronto del tipo.

File Flash 4:

```
x = y
```

File Flash 5 convertito:

```
Number(x) == Number(y)
```

L'esempio seguente usa l'operatore di assegnazione per assegnare il tipo di dati numero alla variabile *x*.

```
x = 5
```

L'esempio seguente usa l'operatore di assegnazione per assegnare il tipo di dati stringa alla variabile *x*.

```
x = "hello"
```

-= (assegnazione negazione)

Sintassi

espressione1 -= *espressione2*

Argomenti

espressione1, *espressione2* Numeri interi, in virgola mobile o stringhe.

Descrizione

Operatore (assegnazione composta); assegna a *espressione1* il valore di *espressione1* - *espressione2*.

Letture

Flash 4 o versione successiva.

Esempio

L'esempio seguente illustra l'uso dell'operatore `-=` con variabili e numeri:

```
x -= y equivale a x = x - y  
Se x = 5 e y = 10 allora  
x -= 10 restituisce -5
```

== (uguaglianza)

Sintassi

espressione1 == *espressione2*

Argomenti

espressione1, *espressione2* Numeri, stringhe, valori booleani, variabili, oggetti, matrici o funzioni.

Descrizione

Operatore (uguaglianza); verifica l'uguaglianza tra due espressioni. Il risultato è `true` se le espressioni sono uguali.

La definizione di *uguale* dipende dal tipo di dati dell'argomento:

- Numeri, stringhe e valori booleani vengono confrontati come valore e vengono considerati uguali se il loro valore corrisponde. Ad esempio, due stringhe sono uguali se sono costituite dallo stesso numero di caratteri.
- Variabili, oggetti, matrici e funzioni vengono confrontati come riferimento. Due variabili sono uguali se fanno riferimento allo stesso oggetto, matrice o funzione. Due matrici distinte non vengono mai considerate uguali, anche se sono costituite dallo stesso numero di elementi.

Letture

Flash 5 o versione successiva.

Esempio

L'esempio seguente usa l'operatore `==` in un'istruzione `if`:

```
a = "David" , b = "David";  
if (a == b)  
trace("David is David");
```


> (maggiore di)

Sintassi

espressione1 > *espressione2*

Argomenti

espressione1, *espressione2* Numeri interi, in virgola mobile o stringhe.

Descrizione

Operatore (confronto); confronta due espressioni e determina se *espressione1* è maggiore di *espressione2* (*true*) oppure se *espressione1* è minore o uguale a *espressione2* (*false*).

Letture

Flash 4; Flash 5 o versione successiva. In Flash 5 > è un operatore di confronto in grado di gestire diversi tipi di dati. In Flash 4 > è un operatore numerico. I file di Flash 4 importati in ambiente di creazione di Flash 5 vengono sottoposti a un processo di conversione per mantenere l'integrità del tipo di dati. L'esempio seguente illustra il processo di conversione.

Esempio

L'esempio seguente illustra la conversione di un file Flash 4 che contiene un'operazione numerica che richiede un confronto del tipo.

File Flash 4:

```
x > y
```

File Flash 5 convertito:

```
Number(x) > Number(y)
```

>= (maggiore o uguale a)

Sintassi

espressione1 >= *espressione2*

Argomenti

espressione1, *espressione2* Stringhe, numeri interi o in virgola mobile.

Descrizione

Operatore (confronto); confronta due espressioni e determina se *espressione1* è maggiore o uguale a *espressione2* (*true*) oppure se *espressione1* è minore di *espressione2* (*false*).

Letture

Flash 4; Flash 5 o versione successiva. In Flash 5 >= è un operatore di confronto in grado di gestire diversi tipi di dati. In Flash 4 >= è un operatore numerico. I file di Flash 4 importati in ambiente di creazione di Flash 5 vengono sottoposti a un processo di conversione per mantenere l'integrità del tipo di dati. L'esempio seguente illustra il processo di conversione.

Esempio

L'esempio seguente illustra la conversione di un file Flash 4 che contiene un'operazione numerica che richiede un confronto del tipo.

File Flash 4:

$x \geq y$

File Flash 5 convertito:

$\text{Number}(x) \geq \text{Number}(y)$

» (spostamento a destra bit a bit)

Sintassi

espressione1 >> *espressione2*

Argomenti

espressione1 Numero, stringa o espressione da spostare a destra.

espressione2 Numero, stringa o espressione che viene convertita in un numero intero da 0 a 31.

Descrizione

Operatore (bit a bit); converte *espressione1* ed *espressione2* in interi a 32 bit, quindi sposta a destra tutti i bit contenuti in *espressione1* di un numero di posizioni corrispondente al numero intero risultante dalla conversione di *espressione2*. I bit spostati a destra oltre la cifra meno significativa, vengono scartati. Per mantenere invariato il segno originale di *espressione*, in corrispondenza dei bit a sinistra viene inserito uno 0 se il bit più significativo (ovvero il bit più a sinistra) di *espressione1* è 0, mentre viene inserito il numero 1 se il bit più significativo è 1. Lo spostamento di un valore a destra di 1 posizione equivale a dividere tale valore per 2 e a scartare quindi il resto.

Lettore

Flash 5 o versione successiva.

Esempio

L'esempio seguente converte il numero 65535 in un intero a 32 bit, quindi lo sposta a destra di otto bit.

$x = 65535 \gg 8$

Il risultato di tale operazione è il seguente:

$x = 255$

Ciò è dovuto al fatto che il numero decimale 65535 equivale al numero binario 1111111111111111 (*sedici 1*), il numero binario 1111111111111111 spostato a destra di otto bit equivale al numero binario 11111111 e quest'ultimo equivale al numero decimale 255. Il bit più significativo è 0 perché gli interi sono a 32 bit, quindi il bit da inserire è 0.

L'esempio seguente converte -1 in un intero a 32 bit, quindi lo sposta a destra di un bit.

```
x = -1 >> 1
```

Il risultato di tale operazione è il seguente:

```
x = -1
```

Ciò è dovuto al fatto che il numero decimale -1 equivale al numero binario 111111111111111111111111111111 (trentadue 1), lo spostamento a destra di un bit determina lo scarto del bit meno significativo (il bit all'estrema destra) e l'inserimento del numero 1 in corrispondenza del bit più significativo. Ne risulta il numero binario 111111111111111111111111111111 (*trentadue 1*) che rappresenta il numero intero a 32 bit -1.

Vedere anche

“>>= (spostamento a destra bit a bit e assegnazione)” a pagina 211

>>= (spostamento a destra bit a bit e assegnazione)

Sintassi

espressione1 =>> *espressione2*

Argomenti

espressione1 Numero, stringa o espressione da spostare a sinistra.

espressione2 Numero, stringa o espressione che viene convertita in un numero intero da 0 a 31.

Descrizione

Operatore (assegnazione composta); esegue un'operazione di spostamento a destra bit a bit e memorizza il contenuto come risultato in *espressione1*.

Lettore

Flash 5 o versione successiva.

Esempio

Le due espressioni seguenti si equivalgono:

```
A >>= B
```

```
A = (A >> B)
```

Il seguente codice commentato usa l'operatore bit a bit `>>=` e rappresenta anche un esempio dell'uso di tutti gli operatori bit a bit.

```
function convertToBinary(number)
{
    var result = "";
    for (var i=0; i<32; i++) {
        // Estrae il bit meno significativo usando
        // l'operatore AND bit a bit
        var lsb = number & 1;
        // Aggiunge questo bit alla stringa di risultato
        result = (lsb ? "1" : "0") + result;
        // Sposta il numero a destra di un bit, per accedere
        // al bit successivo
        number >>= 1;
    }
    return result;
}
convertToBinary(479)
// Restituisce la stringa
000000000000000000000000011101111
// La stringa precedente è la rappresentazione binaria
// del numero decimale 479.
```

Vedere anche

“<< (spostamento a sinistra bit a bit)” a pagina 204

>>> (spostamento a destra senza segno bit a bit)

Sintassi

espressione1 >>> *espressione2*

Argomenti

espressione1 Numero, stringa o espressione da spostare a destra.

espressione2 Numero, stringa o espressione che viene convertita in un numero intero da 0 a 31.

Descrizione

Operatore (bit a bit); la sua funzione è analoga a quella dell'operatore di spostamento a destra bit a bit (`>>`) tranne per il fatto che non mantiene il segno originale di *espressione* perché in corrispondenza dei bit a sinistra viene sempre inserito uno 0.

Letture

Flash 5 o versione successiva.

Esempio

L'esempio seguente converte -1 in un intero a 32 bit, quindi lo sposta a destra di un bit.

```
x = -1 >>> 1
```

Il risultato di tale operazione è il seguente:

$x = 2147483647$

Ciò è dovuto al fatto che il numero decimale -1 equivale al numero binario 11111111111111111111111111111111 (*trentadue 1*) e che lo spostamento a destra (senza segno) di un bit determina lo scarto del bit meno significativo (il bit all'estrema destra) e l'inserimento di uno 0 in corrispondenza del bit più significativo (il bit all'estrema sinistra). Il risultato è il seguente:

numero binario 01111111111111111111111111111111

che rappresenta il numero intero a 32 bit 2147483647.

Vedere anche

">>= (spostamento a destra bit a bit e assegnazione)" a pagina 211

>>>= (spostamento a destra senza segno bit a bit e assegnazione)

Sintassi

espressione1>>> = *espressione2*

Argomenti

espressione1 Numero, stringa o espressione da spostare a sinistra.

espressione2 Numero, stringa o espressione che viene convertita in un numero intero da 0 a 31.

Descrizione

Operatore (assegnazione composta); esegue un'operazione di spostamento a destra bit a bit senza segno e memorizza il contenuto come risultato in *espressione1*.

Lettore

Flash 5 o versione successiva.

Esempio

Le due espressioni seguenti si equivalgono:

A >>>= B

A = (A >>> B)

Vedere anche

">>> (spostamento a destra senza segno bit a bit)" a pagina 212

">>= (spostamento a destra bit a bit e assegnazione)" a pagina 211

add

Sintassi

stringa1 add *stringa2*

Argomenti

stringa1, *stringa2* Una stringa.

Descrizione

Operatore; concatena due o più stringhe. L'operatore `add` sostituisce l'operatore `&` di Flash 4; i file Flash 4 che usano l'operatore `&` al momento dell'importazione in ambiente di creazione codice di Flash 5, vengono convertiti automaticamente in modo che usino l'operatore `add` per la concatenazione di stringhe. L'uso dell'operatore `add` in Flash 5 è diventato obsoleto e si consiglia l'uso dell'operatore `+` in fase di creazione di codice per Flash 5 Player. Usare l'operatore `add` per concatenare le stringhe quando si crea del contenuto per Flash 4 Player o versioni precedenti.

Lettore

Flash 4 o versione successiva.

Vedere anche

“+ (addizione)” a pagina 201

_alpha

Sintassi

```
nomeistanza._alpha  
nomeistanza._alpha = valore;
```

Argomenti

nomeistanza Il nome di un'istanza di clip filmato.

valore Un numero da 0 a 100 che specifica la trasparenza alfa.

Descrizione

Proprietà; imposta o recupera la trasparenza alfa (*valore*) del clip filmato. I valori validi sono compresi tra 0 (trasparenza completa) e 100 (opacità completa). In un clip filmato in cui la proprietà `_alpha` è impostata su 0, gli oggetti sono attivi anche se sono invisibili. Ad esempio, un pulsante in un filmato in cui la proprietà `_alpha` è impostata su 0, può comunque essere premuto.

Lettore

Flash 4 o versione successiva.

Esempio

L'istruzione seguente imposta la proprietà `_alpha` del clip filmato `star` su 30% quando il pulsante viene premuto.

```
on(release) {  
    setProperty(star._alpha = 30);  
}
```

oppure

```
on(release) {  
    star._alpha = 30;  
}
```

and

Sintassi

condizione1 and condizione2

Argomenti

condizione1, condizione2 Condizioni o espressioni che restituiscono `true` o `false`.

Descrizione

Operatore; esegue un'operazione di AND logico in ambiente Flash 4 Player. Se entrambe le espressioni restituiscono `true` l'intera espressione è `true`.

Letture

Flash 4 o versione successiva. Questo operatore è diventato obsoleto in Flash 5 ed è consigliato l'uso del nuovo operatore `&&`.

Vedere anche

"`&&` (cortocircuito AND)" a pagina 187

Array (oggetto)

L'oggetto Array consente di accedere alle matrici e di gestirle. Una matrice è un oggetto le cui proprietà sono identificate da un numero che ne rappresenta la posizione all'interno della matrice. Tale numero è detto anche indice. Tutte le matrici iniziano automaticamente da zero, il che significa che il primo elemento nella matrice è `[0]`, il secondo è `[1]` e così via.. Nell'esempio seguente, `myArray` contiene i mesi dell'anno identificati da un numero.

```
myArray[0] = "January"  
myArray[1] = "February"  
myArray[2] = "March"  
myArray[3] = "April"
```

Per creare un oggetto Array, usare la funzione di costruzione `new Array()`. Per accedere agli elementi della matrice, usare l'operatore di accesso matrice `[]`.

Riepilogo dei metodi validi per l'oggetto Array

Metodo	Descrizione
<code>concat</code>	Concatena gli argomenti e li restituisce come nuova matrice.
<code>join</code>	Restituisce una stringa che contiene tutti gli elementi della matrice.
<code>pop</code>	Rimuove l'ultimo elemento della matrice e ne restituisce il valore corrispondente.
<code>push</code>	Aggiunge uno o più elementi in fondo alla matrice e ne restituisce la nuova dimensione.
<code>reverse</code>	Inverte la direzione di una matrice.
<code>shift</code>	Rimuove il primo elemento da una matrice e ne restituisce il valore corrispondente.
<code>slice</code>	Estrae una sezione da una matrice e la restituisce come nuova matrice.
<code>sort</code>	Ordina una matrice in posizione.
<code>splice</code>	Aggiunge e/o rimuove gli elementi di una matrice.
<code>toString</code>	Restituisce un valore stringa che rappresenta gli elementi dell'oggetto Array.
<code>unshift</code>	Aggiunge uno o più elementi all'inizio della matrice e ne restituisce la nuova dimensione.

Riepilogo delle proprietà valide per l'oggetto Array

Proprietà	Descrizione
<code>length</code>	Restituisce le dimensioni della matrice.

Funzione di costruzione dell'oggetto Array

Sintassi

```
new Array();  
new Array(lunghezza);  
new Array(elemento0, elemento1, elemento2,...elementoN);
```

Argomenti

lunghezza Numero intero che indica il numero di elementi della matrice. Nel caso di elementi non contigui, la dimensione della matrice corrisponde al numero di indice dell'ultimo elemento della matrice più 1. Per ulteriori informazioni, vedere la proprietà `Array.length`.

elemento0...elementoN Lista di due o più valori arbitrari. Tali valori possono essere numeri, nomi o altri elementi specificati in una matrice. Al primo elemento della matrice corrisponde sempre il numero di indice o posizione 0.

Descrizione

Funzione di costruzione; consente di accedere agli elementi di una matrice e di gestirli. Le matrici iniziano automaticamente da zero e gli elementi vengono indicizzati per numero ordinale.

Se non vengono specificati argomenti, viene creata una matrice di dimensione zero.

Lettores

Flash 5 o versione successiva.

Esempio

L'esempio seguente crea un nuovo oggetto Array la cui dimensione iniziale è uguale a 0.

```
myArray = new Array();
```

L'esempio seguente crea il nuovo oggetto Array A-Team, con una dimensione iniziale di 4.

```
A-Team = new Array("Jody", "Mary", "Marcelle", "Judy");
```

Gli elementi iniziali della matrice A-Team sono disposti nel modo seguente:

```
myArray[0] = "Jody"  
myArray[1] = "Mary"  
myArray[2] = "Marcelle"  
myArray[3] = "Judy"
```

Vedere anche

"Array.length" a pagina 219

Array.concat

Sintassi

```
nomeMatrice.concat(valore0, valore1, ... valoreN);
```

Argomenti

valore0, ... valoreN Numeri, elementi o stringhe da concatenare in una nuova matrice.

Descrizione

Metodo; concatena gli elementi specificati negli argomenti, se esistenti, quindi crea e restituisce una nuova matrice. Se negli argomenti è specificata una matrice, vengono concatenati gli elementi di tale matrice e non la matrice stessa.

Lettores

Flash 5 o versione successiva.

Esempio

Il codice seguente concatena due matrici:

```
alpha = new Array("a","b","c");
numeric = new Array(1,2,3);
alphaNumeric=alpha.concat(numeric); // Crea la matrice
["a","b","c",1,2,3]
```

Il codice seguente concatena tre matrici:

```
num1=[1,3,5];
num2=[2,4,6];
num3=[7,8,9];
nums=num1.concat(num2,num3) // Crea la matrice [1,3,5,2,4,6,7,8,9]
```

Array.join

Sintassi

```
nomeMatrice.join();
nomeMatrice.join(separatore);
```

Argomenti

separatore Carattere o stringa che separa gli elementi della matrice nella stringa che viene restituita. Se questo argomento viene omissso, il separatore predefinito usato sarà la virgola.

Descrizione

Metodo; converte in stringa gli elementi della matrice, li concatena, inserisce tra di essi il separatore specificato e restituisce la stringa risultante.

Lettore

Flash 5 o versione successiva.

Esempio

L'esempio seguente crea una matrice con tre elementi, quindi li unisce per tre volte inserendo tra gli elementi il separatore predefinito, quindi una virgola seguita da uno spazio e infine il segno più.

```
a = new Array("Earth","Moon","Sun")
// Assegna "Earth,Moon,Sun" a myVar1
myVar1=a.join();
// Assegna "Earth, Moon, Sun" a myVar2
myVar2=a.join(", ");
// Assegna "Earth + Moon + Sun" a myVar3
myVar3=a.join(" + ");
```

Array.length

Sintassi

`nomeMatrice.length;`

Argomenti

Nessuno.

Descrizione

Proprietà; contiene la dimensione della matrice. Questa proprietà viene aggiornata automaticamente quando alla matrice vengono aggiunti nuovi elementi. Durante l'assegnazione `nomeMatrice[indice] = valore`; se `indice` è un numero e `indice+1` è un valore maggiore di quello della proprietà `length`, la proprietà `length` viene aggiornata al valore di `indice+ 1`.

Lettore

Flash 5 o versione successiva.

Esempio

Il codice seguente illustra come la proprietà `length` viene aggiornata.

```
// La lunghezza iniziale è 0
myArray = new Array();
// .length di myArray diventa 1
myArray[0] = 'a'
// .length di myArray diventa 2
myArray[1] = 'b'
// .length di myArray diventa 10
myArray[9] = 'c'
```

Array.pop

Sintassi

`nomeMatrice.pop();`

Argomenti

Nessuno.

Descrizione

Metodo; rimuove l'ultimo elemento della matrice e restituisce il valore di tale elemento.

Lettore

Flash 5 o versione successiva.

Esempio

Il codice seguente crea la matrice `myPets` che contiene quattro elementi, quindi rimuove l'ultimo di essi.

```
myPets = ["cat", "dog", "bird", "fish"];
popped = myPets.pop();
```

Array.push

Sintassi

nomeMatrice.push(*valore*,...);

Argomenti

valore Uno o più valori da aggiungere in fondo alla matrice.

Descrizione

Metodo; aggiunge uno o più elementi in fondo alla matrice e ne restituisce la nuova dimensione.

Lettore

Flash 5 o versione successiva.

Esempio

Il codice seguente crea la matrice `myPets` che contiene due elementi, quindi aggiunge ad essa due elementi. Quando il codice viene eseguito, al termine `pushed` contiene 4 elementi.

```
myPets = ["cat", "dog"];
pushed = myPets.push("bird", "fish")
```

Array.reverse

Sintassi

nomeMatrice.reverse();

Argomenti

Nessuno.

Descrizione

Metodo; inverte la matrice in posizione.

Lettore

Flash 5 o versione successiva.

Esempio

L'esempio seguente illustra l'uso del metodo `Array.reverse`:

```
var numbers = [1, 2, 3, 4, 5, 6];
trace(numbers.join())
numbers.reverse()
trace(numbers.join())
```

Output:

```
1,2,3,4,5,6
6,5,4,3,2,1
```

Array.shift

Sintassi

```
nomeMatrice.shift();
```

Argomenti

Nessuno.

Descrizione

Metodo; rimuove il primo elemento della matrice e lo restituisce.

Lettore

Flash 5 o versione successiva.

Esempio

Il codice seguente crea la matrice `myPets` e quindi rimuove da essa il primo elemento:

```
myPets = ["cat", "dog", "bird", "fish"];  
shifted = myPets.shift();
```

Il valore restituito è `cat`.

Vedere anche

“`Array.pop`” a pagina 219

“`Array.unshift`” a pagina 225

Array.splice

Sintassi

```
nomeMatrice.splice(inizio, fine);
```

Argomenti

inizio Un numero che specifica l'indice del punto di inizio della porzione. Se *inizio* è un numero negativo, il punto di inizio è specificato a partire dal fondo della matrice, dove -1 è l'ultimo elemento.

fine Un numero che specifica l'indice del punto finale della porzione. Se questo argomento viene omissso, la porzione include tutti gli elementi dall'inizio fino alla fine della matrice. Se *fine* è un numero negativo, il punto finale è specificato a partire dal fondo della matrice, dove -1 è l'ultimo elemento.

Descrizione

Metodo; estrae una porzione o una sottostringa della matrice e la restituisce come nuova matrice senza modificare quella originale. La matrice restituita contiene l'elemento identificato da *inizio* e tutti gli elementi tra questo e l'elemento identificato da *fine* che è escluso.

Lettore

Flash 5 o versione successiva.

Array.sort

Sintassi

```
nomeMatrice.sort();  
nomeMatrice.sort(funzordinamento);
```

Argomenti

funzordinamento Funzione di confronto facoltativa usata per determinare il tipo di ordinamento. Dati gli argomenti A e B, la funzione di ordinamento specificata deve restituire i seguenti valori:

- -1 se nella sequenza ordinata A precede B
- 0 se A = B
- 1 se nella sequenza ordinata A segue B

Descrizione

Metodo; ordina la matrice in posizione senza farne una copia. Se viene omesso l'argomento *funzordinamento*, Flash ordina gli elementi in posizione usando l'operatore di confronto <.

Lettore

Flash 5 o versione successiva.

Esempio

Il seguente esempio usa il metodo *Array.sort* senza specificare l'argomento *funzordinamento*.

```
var fruits = ["oranges", "apples", "strawberries",  
             "pineapples", "cherries"];  
trace(fruits.join())  
fruits.sort()  
trace(fruits.join())
```

Output:

```
oranges,apples,strawberries,pineapples,cherries  
apples,cherries,oranges,pineapples,strawberries
```

L'esempio seguente usa il metodo `Array.sort` e specifica la funzione di ordinamento.

```
var passwords = [
    "gary:foo",
    "mike:bar",
    "john:snafu",
    "steve:yuck",
    "daniel:1234"
];
function order (a, b) {
    // Le voci da ordinare sono nel formato
    // nome:password
    // Ordina usando solo la porzione del nome
    // della voce come chiave.
    var name1 = a.split(':')[0];
    var name2 = b.split(':')[0];
    if (name1 < name2) {
        return -1;
    } else if (name1 > name2) {
        return 1;
    } else {
        return 0;
    }
}
for (var i=0; i< password.length; i++) {
    trace (passwords.join());
}
passwords.sort(order);
trace ("Sorted:")
for (var i=0; i< password.length; i++) {
    trace (passwords.join());
}
```

Output:

```
daniel:1234
gary:foo
john:snafu
mike:bar
steve:yuck
```

Array.splice

Sintassi

nomeMatrice.splice(inizio, numeroeliminati, valore0, valore1...valoreN);

Argomenti

inizio L'indice dell'elemento della matrice in corrispondenza del quale ha inizio l'operazione di inserimento o eliminazione.

numeroeliminati Il numero di elementi da eliminare. Tale numero include l'elemento specificato dall'argomento *inizio*. Se non viene specificato un valore per *numeroeliminati*, il metodo elimina tutti i valori, dall'elemento *inizio* all'ultimo elemento della matrice.

valore Zero o più valori da inserire nella matrice in corrispondenza del punto di inserimento specificato dall'argomento *inizio*. È un argomento opzionale.

Descrizione

Metodo; aggiunge e/o rimuove gli elementi di una matrice. Questo metodo modifica la matrice stessa senza farne una copia.

Lettore

Flash 5 o versione successiva.

Array.toString

Sintassi

```
nomeMatrice.toString();
```

Argomenti

Nessuno.

Descrizione

Metodo; restituisce un valore stringa che rappresenta gli elementi dell'oggetto Array specificato. Ciascun elemento della matrice, a partire dall'indice 0 fino ad arrivare all'indice *nomeMatrice.length-1*, viene convertito in stringa concatenata separata da virgola.

Lettore

Flash 5 o versione successiva.

Esempio

L'esempio seguente crea *myArray* e la converte in stringa.

```
myArray = new Array();  
myArray[0] = 1;  
myArray[1] = 2;  
myArray[2] = 3;  
myArray[3] = 4;  
myArray[4] = 5;  
  
trace(myArray.toString())
```

Output:

1,2,3,4,5

Array.unshift

Sintassi

nomeMatrice.unshift(valore1, valore2, ... valoreN);

Argomenti

valore1, ... valoreN Uno o più numeri, elementi o variabili da inserire all'inizio della matrice.

Descrizione

Metodo; aggiunge uno o più elementi all'inizio della matrice e ne restituisce la nuova dimensione.

Lettore

Flash 5 o versione successiva.

Boolean (funzione)

Sintassi

Boolean(espressione);

Argomenti

espressione Variabile, numero o stringa da convertire in valore booleano.

Descrizione

Funzione; converte l'argomento specificato in valore booleano e lo restituisce.

Lettore

Flash 5 o versione successiva.

Boolean (oggetto)

L'oggetto Boolean è un semplice oggetto wrapper con la stessa funzionalità dell'oggetto Boolean standard di JavaScript. L'oggetto Boolean consente di recuperare il tipo di dati di base o la rappresentazione in formato stringa dell'oggetto Boolean.

Riepilogo dei metodi validi per l'oggetto Boolean

Metodo	Descrizione
<code>toString</code>	Restituisce la rappresentazione in formato stringa (<code>true</code>) o (<code>false</code>) dell'oggetto Boolean.
<code>valueOf</code>	Restituisce il tipo di valore di base dell'oggetto Boolean specificato.

Funzione di costruzione dell'oggetto Boolean

Sintassi

```
new Boolean();  
new Boolean(x);
```

Argomenti

x Numero, stringa, valore booleano, oggetto, clip filmato o altra espressione.
È un argomento opzionale.

Descrizione

Funzione di costruzione; crea un'istanza dell'oggetto Boolean. Se viene omissa l'argomento *x*, l'oggetto Boolean viene inizializzato con il valore `false`. Se l'argomento *x* viene specificato, il metodo valuta l'argomento e restituisce il risultato come valore booleano in base alle seguenti regole di conversione del tipo.

- Se *x* è un numero, la funzione restituisce `true` se *x* non equivale a 0 oppure `false` se *x* è un altro numero.
- Se *x* è un valore booleano, la funzione restituisce *x*.
- Se *x* è un oggetto o un clip filmato, la funzione restituisce `true` se *x* non equivale a `null`; altrimenti la funzione restituisce `false`.
- Se *x* è una stringa, la funzione restituisce `true` se `Number(x)` non equivale a 0; altrimenti la funzione restituisce `false`.

Nota: Per mantenere la compatibilità con Flash 4, la gestione delle stringhe in relazione all'oggetto Boolean non è conforme allo standard ECMA 262.

Letture

Flash 5 o versione successiva.

Boolean.toString

Sintassi

```
Boolean.toString();
```

Argomenti

Nessuno.

Descrizione

Metodo; restituisce una rappresentazione in formato stringa, `true` o `false`, dell'oggetto Boolean.

Letture

Flash 5 o versione successiva.

Boolean.valueOf

Sintassi

```
Boolean.toString();
```

Argomenti

Nessuno.

Descrizione

Metodo; restituisce il tipo di valore di base dell'oggetto Boolean specificato e converte l'oggetto wrapper Boolean in tale tipo di valore di base.

Letture

Flash 5 o versione successiva.

break

Sintassi

```
break;
```

Argomenti

Nessuno.

Descrizione

Azione; viene usata all'interno di un ciclo (`for`, `for...in`, `do...while` o `while`). L'azione `break` indica a Flash di ignorare il resto del corpo del ciclo, interromperne l'iterazione ed eseguire l'istruzione che segue l'istruzione del ciclo. L'azione `break` consente di uscire da una serie di cicli annidati.

Letture

Flash 4 o versione successiva.

Esempio

L'esempio seguente usa l'azione `break` per uscire da un ciclo altrimenti infinito.

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

call

Sintassi

```
call(fotogramma);
```

Argomenti

fotogramma Il nome o il numero del fotogramma da chiamare nel contesto dello script.

Descrizione

Azione; passa il contesto dallo script corrente allo script associato al fotogramma che viene chiamato. Le variabili locali non esisteranno più al termine dell'esecuzione dello script.

Letttore

Flash 4 o versione successiva. Questa azione è diventata obsoleta in Flash 5 ed è consigliato l'uso della nuova azione `function`.

Vedere anche

“`function`” a pagina 262

chr

Sintassi

```
chr(numero);
```

Argomenti

numero Il numero di codice ASCII da convertire in carattere.

Descrizione

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

Letttore

Flash 4 o versione successiva. Questa funzione è diventata obsoleta in Flash 5 ed è consigliato l'uso del nuovo metodo `String.fromCharCode`.

Esempio

L'esempio seguente converte il numero 65 nella lettera “A”.

```
chr(65) = "A"
```

Vedere anche

“`String.fromCharCode`” a pagina 361

Color (oggetto)

L'oggetto `Color` consente di impostare e recuperare il valore del colore RGB e la trasformazione dei colori dei clip filmato. L'oggetto `Color` è supportato da Flash 5 Player e versioni successive.

Prima di chiamare i metodi dell'oggetto `Color` è necessario crearne un'istanza usando la funzione di costruzione `new Color()` .

Riepilogo dei metodi validi per l'oggetto `Color`

Metodo	Descrizione
<code>getRGB</code>	Restituisce il valore RGB numerico impostato dall'ultima chiamata di <code>setRGB</code> .
<code>getTransform</code>	Restituisce i dati relativi alla trasformazione impostati dall'ultima chiamata di <code>setTransform</code> .
<code>setRGB</code>	Imposta la rappresentazione in formato esadecimale del valore RGB per l'oggetto <code>Color</code> .
<code>setTransform</code>	Imposta la trasformazione dei colori per l'oggetto <code>Color</code> .

Funzione di costruzione per l'oggetto `Color`

Sintassi

```
new Color(target);
```

Argomenti

target Il nome del clip filmato a cui viene applicato il nuovo colore.

Descrizione

Funzione di costruzione; crea un oggetto `Color` per il clip filmato specificato dall'argomento *target*.

Letture

Flash 5 o versione successiva.

Esempio

L'esempio seguente crea un nuovo oggetto `Color` di nome `myColor`, per il filmato `myMovie`.

```
myColor = new Color(myMovie);
```

`Color.setRGB`

Sintassi

```
myColor.getRGB();
```

Argomenti

Nessuno.

Descrizione

Metodo; restituisce i valori numerici impostati dall'ultima chiamata di `setRGB`.

Lettore

Flash 5 o versione successiva.

Esempio

Il codice seguente recupera il valore RGB in formato stringa esadecimale:

```
value = (getRGB()).toString(16);
```

Vedere anche

“Color.setRGB” a pagina 230

Color.getTransform

Sintassi

```
oggettoColore.getTransform();
```

Argomenti

Nessuno.

Descrizione

Metodo; restituisce i valori di trasformazione impostati dall'ultima chiamata di `setTransform`.

Lettore

Flash 5 o versione successiva.

Vedere anche

“Color.setTransform” a pagina 231

Color.setRGB

Sintassi

```
oggettoColore.setRGB(0xRRGGBB);
```

Argomenti

0xRRGGBB Il colore esadecimale o RGB da impostare. *RR*, *GG* e *BB* sono costituiti ciascuno da due cifre esadecimali che specificano l'offset di ciascun componente del colore.

Descrizione

Metodo; specifica un colore RGB per l'oggetto Color. La chiamata di questo metodo ridefinisce tutte le precedenti impostazioni definite dal metodo `setTransform`.

Lettore

Flash 5 o versione successiva.

Esempio

L'esempio seguente imposta il valore del colore RGB per il clip filmato `myMovie`.

```
myColor = new Color(myMovie);  
myColor.setRGB(0x993366);
```

Vedere anche

“Color.setTransform” a pagina 231

Color.setTransform

Sintassi

```
oggettoColore.setTransform(oggettoTrasformazioneColore);
```

Argomenti

oggettoTrasformazioneColore Un oggetto creato tramite la funzione di costruzione dell'oggetto Object generico, specificando i valori di trasformazione dei colori per i parametri. L'oggetto di trasformazione dei colori deve avere i parametri *ra*, *rb*, *ga*, *gb*, *ba*, *bb*, *aa*, *ab* descritti di seguito.

Descrizione

Metodo; imposta i dati relativi alla trasformazioni dei colori per un oggetto Color. L'argomento *oggettoTrasformazioneColore* è un oggetto che viene creato a partire dall'oggetto generico Object con parametri che specificano la percentuale e i valori di offset dei componenti rosso, verde, blu e alfa (trasparenza) di un colore, immessi in formato *0xRRGGBBAA*.

I parametri relativi a tale oggetto sono definiti di seguito:

- *ra* è la percentuale del componente rosso (da -100 a 100).
- *rb* è l'offset del componente rosso (da -255 a 255).
- *ga* è la percentuale del componente verde (da -100 a 100).
- *gb* è l'offset del componente verde (da -255 a 255).
- *ba* è la percentuale del componente blu (da -100 a 100).
- *bb* è l'offset del componente blu (da -255 a 255).
- *aa* è la percentuale di alfa (da -100 a 100).
- *ab* è l'offset di alfa (da -255 a 255).

Il codice seguente consente di creare un oggetto per la trasformazione dei colori:

```
myColorTransform = new Object();  
myColorTransform.ra = 50;  
myColorTransform.rb = 244;  
myColorTransform.ga = 40;  
myColorTransform.gb = 112;  
myColorTransform.ba = 12;  
myColorTransform.bb = 90;  
myColorTransform.aa = 40;  
myColorTransform.ab = 70;
```

È possibile usare anche la sintassi seguente:

```
myColorTransform = { ra: '50', rb: '244', ga: '40', gb: '112', ba: '12', bb: '90', aa: '40', ab: '70' }
```

Letture

Flash 5 o versione successiva.

Esempio

L'esempio seguente illustra il processo di creazione di un nuovo oggetto Color per un filmato, tramite la creazione di un oggetto per la trasformazione dei colori con i parametri descritti sopra, usando la funzione di costruzione Object e passando quindi l'oggetto per la trasformazione a un oggetto Color tramite il metodo setTransform.

```
// Crea un oggetto Color di nome myColor per il target myMovie
myColor = new Color(myMovie);
// Crea un oggetto di trasformazione colore di nome
// myColorTransform usando l'oggetto generico Object
myColorTransform = new Object;
// Imposta i valori per myColorTransform
myColorTransform = { ra: '50', rb: '244', ga: '40', gb: '112', ba: '12', bb: '90', aa: '40', ab: '70' }

// Associa l'oggetto di trasformazione colore con l'oggetto Color
// creato per myMovie
myColor.setTransform(myColorTransform);
```

continue

Sintassi

```
continue;
```

Argomenti

Nessuno.

Descrizione

Azione; si usa in vari tipi di istruzione di ciclo.

In un ciclo while l'azione continue indica a Flash di ignorare il resto del corpo del ciclo e di passare all'inizio del ciclo, dove la condizione è verificata.

In un ciclo do...while l'azione continue indica a Flash di ignorare il resto del corpo del ciclo e di passare alla fine del ciclo, dove la condizione è verificata.

In un ciclo for l'azione continue indica a Flash di ignorare il resto del corpo del ciclo e di passare alla valutazione dell'espressione finale del ciclo for.

In un ciclo for...in l'azione continue indica a Flash di ignorare il resto del corpo del ciclo e di tornare all'inizio del ciclo, dove viene elaborato il valore successivo nell'enumerazione.

Lettore

Flash 4 o versione successiva.

Vedere anche

“do... while” a pagina 253

“for” a pagina 258

“for..in” a pagina 260

“while” a pagina 374

_currentframe

Sintassi

nomeistanza._currentframe

Argomenti

nomeistanza Il nome di un'istanza di clip filmato.

Descrizione

Proprietà (sola lettura); restituisce il numero del fotogramma in cui si trova attualmente l'indicatore di riproduzione nella linea temporale.

Lettore

Flash 4 o versione successiva.

Esempio

L'esempio seguente usa _currentframe per indicare a un filmato di spostarsi cinque fotogrammi oltre il fotogramma che contiene l'azione:

```
gotoAndStop(_currentframe + 5);
```

Date (oggetto)

L'oggetto Date consente di recuperare i valori di data e ora relativi all'ora standard (ora di Greenwich, detta attualmente tempo universale coordinato UTC) o relativi al sistema operativo in cui si sta eseguendo Flash Player. Prima di chiamare i metodi dell'oggetto Date è necessario creare un'istanza dell'oggetto Date usando la funzione di costruzione.

L'oggetto Date richiede Flash 5 Player.

I metodi dell'oggetto Date non sono statici, ma sono validi solo per la singola istanza dell'oggetto Date specificata quando viene chiamato il metodo.

Riepilogo dei metodi validi per l'oggetto Date

Metodo	Descrizione
<code>getDate</code>	Restituisce il giorno del mese dell'oggetto Date specificato in base all'ora locale.
<code>getDay</code>	Restituisce il giorno del mese dell'oggetto Date specificato in base all'ora locale.
<code>getFullYear</code>	Restituisce l'anno in formato a quattro cifre dell'oggetto Date specificato in base all'ora locale.
<code>getHours</code>	Restituisce l'ora dell'oggetto Date specificato in base all'ora locale.
<code>getMilliseconds</code>	Restituisce i millesimi di secondo dell'oggetto Date specificato in base all'ora locale.
<code>getMinutes</code>	Restituisce i minuti dell'oggetto Date specificato in base all'ora locale.
<code>getMonth</code>	Restituisce il mese dell'oggetto Date specificato in base all'ora locale.
<code>getSeconds</code>	Restituisce i secondi dell'oggetto Date specificato in base all'ora locale.
<code>getTime</code>	Restituisce il numero di millesimi di secondo dalla mezzanotte dell'1 gennaio 1970, tempo universale coordinato UTC, per l'oggetto Date specificato.
<code>getTimezoneOffset</code>	Restituisce la differenza, in minuti, tra l'ora locale del computer e il tempo universale coordinato UTC.
<code>getUTCDate</code>	Restituisce il giorno del mese (data) dell'oggetto Date specificato in base al tempo universale coordinato UTC.
<code>getUTCDay</code>	Restituisce il giorno della settimana dell'oggetto Date specificato in base al tempo universale coordinato UTC.
<code>getUTCFullYear</code>	Restituisce l'anno in formato a quattro cifre dell'oggetto Date specificato in base al tempo universale coordinato UTC.
<code>getUTCHours</code>	Restituisce l'ora dell'oggetto Date specificato in base al tempo universale coordinato UTC.
<code>getUTCMilliseconds</code>	Restituisce i millesimi di secondo dell'oggetto Date specificato in base al tempo universale coordinato UTC.
<code>getUTCMinutes</code>	Restituisce i minuti dell'oggetto Date specificato in base al tempo universale coordinato UTC.
<code>getUTCMonth</code>	Restituisce il mese dell'oggetto Date specificato in base al tempo universale coordinato UTC.

Metodo	Descrizione
<code>getUTCSeconds</code>	Restituisce i secondi dell'oggetto <code>Date</code> specificato in base al tempo universale coordinato UTC.
<code>getYear</code>	Restituisce l'anno dell'oggetto <code>Date</code> specificato in base al tempo locale.
<code>setDate</code>	Imposta il giorno del mese dell'oggetto <code>Date</code> specificato in base al tempo locale.
<code>setFullYear</code>	Imposta l'anno nel formato a quattro cifre per un oggetto <code>Date</code> in base all'ora locale.
<code>setHours</code>	Imposta le ore per un oggetto <code>Date</code> in base all'ora locale.
<code>setMilliseconds</code>	Imposta i millesimi di secondo per un oggetto <code>Date</code> in base all'ora locale.
<code>setMinutes</code>	Imposta i minuti per un oggetto <code>Date</code> in base all'ora locale.
<code>setMonth</code>	Imposta il mese per un oggetto <code>Date</code> in base all'ora locale.
<code>setSeconds</code>	Imposta i secondi per un oggetto <code>Date</code> in base all'ora locale.
<code>setTime</code>	Imposta la data, in millesimi di secondo, per l'oggetto <code>Date</code> specificato.
<code>setUTCDate</code>	Imposta la data dell'oggetto <code>Date</code> specificato in base al tempo universale coordinato UTC.
<code>setUTCFullYear</code>	Imposta l'anno nel formato a quattro cifre dell'oggetto <code>Date</code> specificato in base al tempo universale coordinato UTC.
<code>setUTCHours</code>	Imposta l'ora dell'oggetto <code>Date</code> specificato in base al tempo universale coordinato UTC.
<code>setUTCMilliseconds</code>	Imposta i millesimi di secondo dell'oggetto <code>Date</code> specificato in base al tempo universale coordinato UTC.
<code>setUTCMinutes</code>	Imposta i minuti dell'oggetto <code>Date</code> specificato in base al tempo universale coordinato UTC.
<code>setUTCMonth</code>	Imposta il mese rappresentato dall'oggetto <code>Date</code> specificato in base al tempo universale coordinato UTC.
<code>setUTCSeconds</code>	Imposta i secondi dell'oggetto <code>Date</code> specificato in base al tempo universale coordinato UTC.
<code>setYear</code>	Imposta l'anno dell'oggetto <code>Date</code> specificato in base all'ora locale.
<code>toString</code>	Restituisce un valore stringa che rappresenta la data e l'ora memorizzate nell'oggetto <code>Date</code> specificato.
<code>Date.UTC</code>	Restituisce il numero di millesimi di secondo tra la mezzanotte dell'1 gennaio 1970, tempo universale coordinato UTC, e l'ora specificata.

Funzione di costruzione per l'oggetto Date

Sintassi

```
new Date();  
  
new Date(anno [, mese [, data [, ora [, minuti [, secondi [,  
millisecondi ]]]]] ) ;
```

Argomenti

anno Valore da 0 a 99 che indica l'anno dal 1900 al 1999; per definire altri intervalli usare il formato esteso a quattro cifre.

mese Numero intero da 0 (gennaio) a 11 (dicembre). È un argomento opzionale.

data Numero intero da 1 a 31. È un argomento opzionale.

ora Numero intero da 0 (mezzanotte) a 23.

minuti Numero intero da 0 a 59. È un argomento opzionale.

secondi Numero intero da 0 a 59. È un argomento opzionale.

millisecondi Numero intero da 0 a 999. È un argomento opzionale.

Descrizione

Oggetto; crea un nuovo oggetto Date che contiene la data e l'ora correnti.

Lettore

Flash 5 o versione successiva.

Esempio

L'esempio seguente recupera la data e l'ora correnti.

```
now = new Date();
```

L'esempio seguente crea un nuovo oggetto Date per il compleanno di Gary, il 7 agosto 1974.

```
gary_birthday = new Date (74, 7, 7);
```

L'esempio seguente crea un nuovo oggetto Date, concatena i valori restituiti dall'esecuzione dei metodi dell'oggetto Date `getMonth`, `getDate` e `getFullYear`, quindi li visualizza nel campo di testo specificato dalla variabile `dateTextField`.

```
myDate = new Date();  
dateTextField = (mydate.getMonth() + "/" + myDate.getDate() + "/"  
+ mydate.getFullYear());
```

Date.getDate

Sintassi

oggettoData.getDate();

Argomenti

Nessuno.

Descrizione

Metodo; restituisce il giorno del mese (un intero da 1 a 31) dell'oggetto Date specificato in base all'ora locale.

Lettore

Flash 5 o versione successiva.

Date.getDay

Sintassi

oggettoData.getDay();

Argomenti

Nessuno.

Descrizione

Metodo; restituisce il giorno del mese (0 corrisponde a domenica, 1 a lunedì e così via) dell'oggetto Date specificato in base all'ora locale. L'ora locale è determinata dal sistema operativo in cui si esegue Flash Player.

Lettore

Flash 5 o versione successiva.

Date.getFullYear

Sintassi

oggettoData.getFullYear();

Argomenti

Nessuno.

Descrizione

Metodo; restituisce l'anno per esteso (un numero in formato a quattro cifre, ad esempio 2000) dell'oggetto Date specificato, in base all'ora locale. L'ora locale è determinata dal sistema operativo in cui si esegue Flash Player.

Lettore

Flash 5 o versione successiva.

Esempio

L'esempio seguente usa la funzione di costruzione per creare un nuovo oggetto `Date` e invia il valore restituito dal metodo `getFullYear` alla finestra `Output`.

```
myDate = new Date();  
trace(myDate.getFullYear());
```

Date.getHours

Sintassi

```
oggettoData.getHours();
```

Argomenti

Nessuno.

Descrizione

Metodo; restituisce l'ora (un intero da 0 a 23) dell'oggetto `Date` specificato in base all'ora locale. L'ora locale è determinata dal sistema operativo in cui si esegue `Flash Player`.

Lettore

Flash 5 o versione successiva.

Date.getMilliseconds

Sintassi

```
oggettoData.getMilliseconds();
```

Argomenti

Nessuno.

Descrizione

Metodo; restituisce i millesimi di secondo (un intero da 0 a 999) dell'oggetto `Date` specificato in base all'ora locale. L'ora locale è determinata dal sistema operativo in cui si esegue `Flash Player`.

Lettore

Flash 5 o versione successiva.

Date.getMinutes

Sintassi

```
oggettoData.getMinutes();
```

Argomenti

Nessuno.

Descrizione

Metodo; restituisce i minuti (un intero da 0 a 59) dell'oggetto Date specificato in base all'ora locale. L'ora locale è determinata dal sistema operativo in cui si esegue Flash Player.

Lettore

Flash 5 o versione successiva.

Date.getMonth

Sintassi

```
oggettoData.getMonth();
```

Argomenti

Nessuno.

Descrizione

Metodo; restituisce il mese (0 corrisponde a gennaio, 1 a febbraio e così via) dell'oggetto Date specificato in base all'ora locale. L'ora locale è determinata dal sistema operativo in cui si esegue Flash Player.

Lettore

Flash 5 o versione successiva.

Date.getSeconds

Sintassi

```
oggettoData.getSeconds();
```

Argomenti

Nessuno.

Descrizione

Metodo; restituisce i secondi (un intero da 0 a 59) dell'oggetto Date specificato in base all'ora locale. L'ora locale è determinata dal sistema operativo in cui si esegue Flash Player.

Lettore

Flash 5 o versione successiva.

Date.getTime

Sintassi

```
oggettoData.getTime();
```

Argomenti

Nessuno.

Descrizione

Metodo; restituisce il numero di millesimi di secondo (un intero da 0 a 999) dalla mezzanotte dell'1 gennaio 1970, tempo universale coordinato UTC, per l'oggetto Date specificato. Questo metodo consente di rappresentare un istante specifico in fase di confronto tra due o più ore definite in base a fusi orari diversi.

Letture

Flash 5 o versione successiva.

Date.getTimezoneOffset

Sintassi

```
oggettoData.getTimezoneOffset();
```

Argomenti

Nessuno.

Descrizione

Metodo; restituisce la differenza, in minuti, tra l'ora locale del computer e il tempo universale coordinato UTC.

Letture

Flash 5 o versione successiva.

Esempio

L'esempio seguente restituisce la differenza tra l'ora legale locale di San Francisco e il tempo universale coordinato UTC. L'ora legale viene considerata nel valore restituito solo se la data specificata nell'oggetto Date ricade nel periodo in cui viene applicata l'ora legale.

```
new Date().getTimezoneOffset();
```

Il risultato è il seguente:

```
420 (7 ore * 60 minuti/ore = 420 minuti)
```

Date.getUTCDate

Sintassi

```
oggettoData.getUTCDate();
```

Argomenti

Nessuno.

Descrizione

Metodo; restituisce il giorno del mese (data) dell'oggetto Date specificato in base al tempo universale coordinato UTC.

Letture

Flash 5 o versione successiva.

Date.getUTCDay

Sintassi

oggettoData.getUTCDate();

Argomenti

Nessuno.

Descrizione

Metodo; restituisce il giorno della settimana dell'oggetto Date specificato in base al tempo universale coordinato UTC.

Date.getUTCFullYear

Sintassi

oggettoData.getUTCFullYear();

Argomenti

Nessuno.

Descrizione

Metodo; restituisce l'anno in formato a quattro cifre dell'oggetto Date specificato in base al tempo universale coordinato UTC.

Lettore

Flash 5 o versione successiva.

Date.getUTCHours

Sintassi

oggettoData.getUTCHours();

Argomenti

Nessuno.

Descrizione

Metodo; restituisce l'ora dell'oggetto Date specificato in base al tempo universale coordinato UTC.

Lettore

Flash 5 o versione successiva.

Date.getUTCMilliseconds

Sintassi

oggettoData.getUTCMilliseconds();

Argomenti

Nessuno.

Descrizione

Metodo; restituisce i millesimi di secondo dell'oggetto Date specificato in base al tempo universale coordinato UTC.

Lettore

Flash 5 o versione successiva.

Date.getUTCMinutes

Sintassi

oggettoData.getUTCMinutes();

Argomenti

Nessuno.

Descrizione

Metodo; restituisce i minuti dell'oggetto Date specificato in base al tempo universale coordinato UTC.

Lettore

Flash 5 o versione successiva.

Date.getUTCMonth

Sintassi

oggettoData.getUTCMonth();

Argomenti

Nessuno.

Descrizione

Metodo; restituisce il mese dell'oggetto Date specificato in base al tempo universale coordinato UTC.

Lettore

Flash 5 o versione successiva.

Date.getUTCSeconds

Sintassi

oggettoData.getUTCSeconds();

Argomenti

Nessuno.

Descrizione

Metodo; restituisce i secondi dell'oggetto Date specificato in base al tempo universale coordinato UTC.

Lettore

Flash 5 o versione successiva.

Date.getYear

Sintassi

oggettoData.getYear();

Argomenti

Nessuno.

Descrizione

Metodo; restituisce l'anno dell'oggetto Date specificato in base al tempo locale. L'ora locale è determinata dal sistema operativo in cui si esegue Flash Player. L'anno è rappresentato dal numero in formato esteso meno 1900. Ad esempio, l'anno 2000 è rappresentato dal numero 100.

Lettore

Flash 5 o versione successiva.

Date.setDate

Sintassi

oggettoData.setDate(*data*);

Argomenti

data Numero intero da 1 a 31.

Descrizione

Metodo; restituisce il giorno del mese per l'oggetto Date specificato in base all'ora locale. L'ora locale è determinata dal sistema operativo in cui si esegue Flash Player.

Lettore

Flash 5 o versione successiva.

Date.setFullYear

Sintassi

oggettoData.setFullYear(*anno* [, *mese* [, *data*]]);

Argomenti

anno Numero in formato a quattro cifre che specifica l'anno. I numeri a due cifre non rappresentano gli anni. Ad esempio, 99 non rappresenta l'anno 1999, ma l'anno 99.

mese Numero intero da 0 (gennaio) a 11 (dicembre). È un argomento opzionale.

data Numero da 1 a 31. È un argomento opzionale.

Descrizione

Metodo; imposta l'anno dell'oggetto Date specificato in base all'ora locale. Se gli argomenti *mese* e *data* sono specificati, vengono impostati anch'essi in base all'ora locale. L'ora locale è determinata dal sistema operativo in cui si esegue Flash Player.

I risultati dei metodi `getUTCDay` e `getDay` possono cambiare in seguito alla chiamata di questo metodo.

Lettore

Flash 5 o versione successiva.

Date.setHours

Sintassi

oggettoData.setHours(*ora*);

Argomenti

ora Numero intero da 0 (mezzanotte) a 23.

Descrizione

Metodo; imposta l'ora per l'oggetto Date specificato in base all'ora locale. L'ora locale è determinata dal sistema operativo in cui si esegue Flash Player.

Lettore

Flash 5 o versione successiva.

Date.setMilliseconds

Sintassi

oggettoData.setMilliseconds(millisecondi);

Argomenti

millisecondi Numero intero da 0 a 999.

Descrizione

Metodo; imposta i millesimi di secondo per l'oggetto Date specificato in base all'ora locale. L'ora locale è determinata dal sistema operativo in cui si esegue Flash Player.

Lettore

Flash 5 o versione successiva.

Date.setMinutes

Sintassi

oggettoData.setMinutes(minuti);

Argomenti

minuti Numero intero da 0 a 59.

Descrizione

Metodo; imposta i minuti per l'oggetto Date specificato in base all'ora locale. L'ora locale è determinata dal sistema operativo in cui si esegue Flash Player.

Lettore

Flash 5 o versione successiva.

Date.setMonth

Sintassi

oggettoData.setMonth(mese [, data]);

Argomenti

mese Numero intero da 0 (gennaio) a 11 (dicembre).

data Numero intero da 1 a 31. È un argomento opzionale.

Descrizione

Metodo; imposta il mese per l'oggetto Date specificato in base all'ora locale. L'ora locale è determinata dal sistema operativo in cui si esegue Flash Player.

Lettore

Flash 5 o versione successiva.

Date.setSeconds

Sintassi

oggettoData.setSeconds(secondi);

Argomenti

secondi Numero intero da 0 a 59.

Descrizione

Metodo; imposta i secondi per l'oggetto Date specificato in base all'ora locale. L'ora locale è determinata dal sistema operativo in cui si esegue Flash Player.

Lettore

Flash 5 o versione successiva.

Date.setTime

Sintassi

oggettoData.setTime(millesimi);

Argomenti

millesimi Numero intero da 0 a 999.

Descrizione

Metodo; imposta la data, in millesimi di secondo, per l'oggetto Date specificato.

Lettore

Flash 5 o versione successiva.

Date.setUTCDate

Sintassi

oggettoData.setUTCDate(data);

Argomenti

data Numero intero da 1 a 31.

Descrizione

Metodo; imposta la data per l'oggetto Date specificato in base al tempo universale coordinato UTC. La chiamata di questo metodo non modifica gli altri campi dell'oggetto Date specificato, ma, se in seguito alla sua chiamata cambia il giorno della settimana, è possibile che i metodi `getUTCDay` e `getDay` riportino un nuovo valore.

Lettore

Flash 5 o versione successiva.

Date.setUTCFullYear

Sintassi

oggettoData.setUTCFullYear(*anno* [, *mese* [, *data*]])

Argomenti

anno L'anno in formato a quattro cifre, ad esempio 2000.

mese Numero intero da 0 (gennaio) a 11 (dicembre). È un argomento opzionale.

data Numero intero da 1 a 31. È un argomento opzionale.

Descrizione

Metodo; imposta l'anno dell'oggetto Date specificato (*oggettoData*) in base al tempo universale coordinato UTC.

Facoltativamente, questo metodo può anche impostare il mese e la data rappresentati dall'oggetto Date specificato. Non viene modificato nessun altro campo dell'oggetto Date. Se in seguito alla chiamata del metodo `setUTCFullYear` cambia il giorno della settimana, è possibile che i metodi `getUTCDay` e `getDay` riportino un nuovo valore.

Letture

Flash 5 o versione successiva.

Date.setUTCHours

Sintassi

oggettoData.setUTCHours(*ora* [, *minuti* [, *secondi* [, *millisecondi*]]));

Argomenti

ora Numero intero da 0 (mezzanotte) a 23.

minuti Numero intero da 0 a 59. È un argomento opzionale.

secondi Numero intero da 0 a 59. È un argomento opzionale.

millisecondi Numero intero da 0 a 999. È un argomento opzionale.

Descrizione

Metodo; imposta l'ora per l'oggetto Date specificato in base al tempo universale coordinato UTC.

Letture

Flash 5 o versione successiva.

Date.setUTCMilliseconds

Sintassi

oggettoData.setUTCMilliseconds(*millisecondi*);

Argomenti

millisecondi Numero intero da 0 a 999.

Descrizione

Metodo; imposta i millesimi di secondo per l'oggetto Date specificato in base al tempo universale coordinato UTC.

Lettore

Flash 5 o versione successiva.

Date.setUTCMinutes

Sintassi

oggettoData.setUTCMinutes(*minuti* [, *secondi* [, *millisecondi*]]);

Argomenti

minuti Numero intero da 0 a 59.

secondi Numero intero da 0 a 59. È un argomento opzionale.

millisecondi Numero intero da 0 a 999. È un argomento opzionale.

Descrizione

Metodo; imposta i minuti per l'oggetto Date specificato in base al tempo universale coordinato UTC.

Lettore

Flash 5 o versione successiva.

Date.setUTCMonth

Sintassi

oggettoData.setUTCMonth(*mese* [, *data*]);

Argomenti

mese Numero intero da 0 (gennaio) a 11 (dicembre).

data Numero intero da 1 a 31. È un argomento opzionale.

Descrizione

Metodo; imposta il mese e, facoltativamente, il giorno (la data) per l'oggetto Date specificato in base al tempo universale coordinato UTC. La chiamata di questo metodo non modifica gli altri campi dell'oggetto Date specificato, ma, se dopo aver specificato l'argomento *data* nella chiamata al metodo `setUTCMonth` cambia il giorno della settimana, è possibile che i metodi `getUTCDay` e `getDay` riportino un nuovo valore.

Lettore

Flash 5 o versione successiva.

Date.setUTCSeconds

Sintassi

```
oggettoData.setUTCSeconds(secondi [, millisecondi]);
```

Argomenti

secondi Numero intero da 0 a 59.

millisecondi Numero intero da 0 a 999. È un argomento opzionale.

Descrizione

Metodo; imposta i secondi per l'oggetto Date specificato in base al tempo universale coordinato UTC.

Lettore

Flash 5 o versione successiva.

Date.setYear

Sintassi

```
oggettoData.setYear(anno);
```

Argomenti

anno Numero in formato a quattro cifre, ad esempio 2000.

Descrizione

Metodo; imposta l'anno per l'oggetto Date specificato in base all'ora locale. L'ora locale è determinata dal sistema operativo in cui si esegue Flash Player.

Lettore

Flash 5 o versione successiva.

Date.toString

Sintassi

oggettoData.toString();

Argomenti

Nessuno.

Descrizione

Metodo; restituisce un valore stringa per rappresentare l'oggetto Date specificato in formato leggibile.

Lettore

Flash 5 o versione successiva.

Esempio

L'esempio seguente restituisce le informazioni contenute nell'oggetto `dateOfBirth` di tipo Date in formato stringa.

```
var dateOfBirth = new Date(74, 7, 7, 18, 15);  
trace (dateOfBirth.toString());
```

Output (ora solare Pacifico):

Mer 7 Ago 18:15:00 GMT-0700 1974

Date.UTC

Sintassi

Date.UTC(anno, mese [, data [, ora [, minuti [, secondi [, milliseconds]]]]]);

Argomenti

anno Numero in formato a quattro cifre, ad esempio 2000.

mese Numero intero da 0 (gennaio) a 11 (dicembre).

data Numero intero da 1 a 31. È un argomento opzionale.

ora Numero intero da 0 (mezzanotte) a 23.

minuti Numero intero da 0 a 59. È un argomento opzionale.

secondi Numero intero da 0 a 59. È un argomento opzionale.

milliseconds Numero intero da 0 a 999. È un argomento opzionale.

Descrizione

Metodo; restituisce il numero di millesimi di secondo tra la mezzanotte dell'1 gennaio 1970, tempo universale coordinato UTC, e l'ora specificata negli argomenti. Si tratta di un metodo statico che viene chiamato attraverso la funzione di costruzione dell'oggetto `Date` e non attraverso un oggetto `Date` specifico. Questo metodo consente di creare un oggetto `Date` che usa il tempo universale UTC, mentre la funzione di costruzione `Date` usa l'ora locale.

Lettore

Flash 5 o versione successiva.

Esempio

L'esempio seguente crea un nuovo oggetto `Date` di nome `gary_birthday` definito in base al tempo universale coordinato UTC. È lo stesso esempio usato per illustrare la funzione di costruzione `new Date()`, ma in base al tempo universale coordinato UTC.

```
gary_birthday = new Date(Date.UTC(1974, 7, 8));
```

delete

Sintassi

```
delete (riferimento);
```

Argomenti

riferimento Il nome della variabile o dell'oggetto da eliminare.

Descrizione

Operatore; distrugge l'oggetto o la variabile specificata come *riferimento* e restituisce `true` se l'oggetto viene eliminato correttamente; in caso contrario, restituisce `false`. Questo operatore consente di liberare la memoria usata dagli script. Tuttavia l'operatore `delete` viene solitamente usato come istruzione nel modo seguente:

```
delete x;
```

L'operazione potrebbe non riuscire e restituire il valore `false` se l'elemento definito da *riferimento* non esiste oppure se non è possibile eliminarlo, come nel caso degli oggetti e delle proprietà predefinite, nonché delle variabili dichiarate con `var`.

Lettore

Flash 5 o versione successiva.

Esempio

L'esempio seguente crea un oggetto, lo usa e infine, quando non è più necessario, lo elimina.

```
account = new Object();  
    account.name = 'Jon';  
    account.balance = 10000;
```

```
...
delete account;
```

L'esempio seguente elimina una proprietà di un oggetto:

```
// Crea il nuovo oggetto "account"
account = new Object();
// Assegna la proprietà name all'account
account.name = 'Jon';
// Cancella la proprietà
delete account.name;
```

Il seguente è un altro esempio di eliminazione di una proprietà di un oggetto:

```
// Crea un oggetto Array con lunghezza 0
array = new Array();
// Array.length diventa 1
array[0] = "abc";
// Aggiunge un altro elemento alla matrice; .Array.length diventa 2
array[1] = "def";
// Aggiunge un altro elemento alla matrice; .Array.length diventa 3
array[2] = "ghi";
// array[2] viene eliminato, ma Array.length rimane invariata
delete array[2];
```

L'esempio seguente illustra il comportamento dell'operatore `delete` sui riferimenti agli oggetti.

```
// Crea un nuovo oggetto e assegna la variabile ref1 come
// riferimento a tale oggetto
ref1 = new Object();
ref1.name = "Jody";
// Copia la variabile di riferimento in una nuova variabile
// ed elimina ref1
ref2 = ref1;
delete ref1;
```

Se la variabile `ref1` non fosse stata copiata in `ref2`, l'oggetto sarebbe stato eliminato al momento dell'eliminazione di `ref1` in quanto non sarebbero esistiti più riferimenti ad esso. Se si eliminasse la variabile `ref2`, non ci sarebbero altri riferimenti all'oggetto e, in tal modo, l'oggetto verrebbe distrutto rendendo disponibile la memoria da esso occupata.

Vedere anche

“var” a pagina 373

do... while

Sintassi

```
do {  
    istruzione;  
} while (condizione);
```

Argomenti

condizione La condizione da valutare.

istruzione L'istruzione da eseguire fino a quando *condizione* restituisce il valore true.

Descrizione

Azione; esegue le istruzioni, quindi valuta la condizione in un ciclo, finché la condizione risulta soddisfatta.

Lettore

Flash 4 o versione successiva.

Vedere anche

“break” a pagina 227

“continue” a pagina 232

_droptarget

Sintassi

nomeIstanzaMobile._droptarget

Argomenti

nomeIstanzaMobile Il nome dell'istanza di un clip filmato su cui è stata indirizzata un'azione `startDrag`.

Descrizione

Proprietà (sola lettura); restituisce il percorso assoluto usando la sintassi della barra inclinata dell'istanza di clip filmato su cui è stato rilasciato *nomeIstanzaMobile*. La proprietà `_droptarget` restituisce sempre un percorso che inizia con `/`. Per confrontare la proprietà `_droptarget` di un'istanza con un riferimento, usare la funzione `eval` che consente di convertire il valore restituito dalla notazione che usa la barra rovesciata in un riferimento.

Lettore

Flash 4 o versione successiva.

Esempio

L'esempio seguente valuta la proprietà `_droptarget` dell'istanza di clip filmato `garbage`, quindi usa la funzione `eval` per convertire il valore dalla notazione che usa la barra rovesciata in un riferimento che usa la sintassi del punto. Il riferimento `garbage` viene quindi confrontato con il riferimento all'istanza di clip filmato `trash`. Se i due riferimenti coincidono, la visibilità di `garbage` viene impostata su `false`. In caso contrario, l'istanza di `garbage` viene reimpostata sulla posizione originale.

```
if (eval(garbage._droptarget) == _root.trash) {  
    garbage._visible = false;  
} else {  
    garbage._x = x_pos;  
    garbage._y = y_pos;  
}
```

Le variabili `x_pos` e `y_pos` vengono impostate sul fotogramma 1 del filmato tramite lo script seguente:

```
x_pos = garbage._x;  
y_pos = garbage._y;
```

Vedere anche

“startDrag” a pagina 355

duplicateMovieClip

Sintassi

```
duplicateMovieClip(target, nuovonome, profondità);
```

Argomenti

target Il percorso target del filmato da duplicare.

nuovonome Identificatore univoco per il clip filmato duplicato.

profondità Il livello di profondità del clip filmato. Il livello di profondità è l'ordine di impilamento che determina l'aspetto di clip filmato e altri oggetti quando si sovrappongono. Al primo clip filmato creato o istanza trascinata sullo stage viene assegnato un livello di profondità pari a 0. È necessario assegnare un livello di profondità diverso a ogni clip filmato successivo o duplicato per impedire che sostituisca i filmati presenti sui livelli già occupati o il clip filmato originale.

Descrizione

Azione; crea un'istanza di clip filmato durante la riproduzione del filmato. La riproduzione dei clip filmato duplicati viene avviata sempre a partire dal fotogramma 1, a prescindere dal fotogramma su cui si trovava il clip filmato originale. Le variabili nel clip filmato principale non vengono copiate nel clip filmato duplicato. L'eliminazione del clip filmato principale comporta l'eliminazione del clip filmato duplicato. Per eliminare un'istanza di clip filmato creata con l'azione `duplicateMovieClip`, usare l'azione o metodo `removeMovieClip`.

Lettore

Flash 4 o versione successiva.

Esempio

L'istruzione seguente duplica l'istanza di clip filmato `flower` dieci volte. La variabile `i` viene usata per creare un nuovo nome di istanza e una profondità.

```
on(release) {  
    amount = 10;  
    while(amount>0) {  
        duplicateMovieClip (_root.flower, "mc" + i, i);  
        setProperty("mc" + i, _x, random(275));  
        setProperty("mc" + i, _y, random(275));  
        setProperty("mc" + i, _alpha, random(275));  
        setProperty("mc" + i, _xscale, random(50));  
        setProperty("mc" + i, _yscale, random(50));  
        i = i + 1;  
        amount = amount-1;  
    }  
}
```

Vedere anche

“`removeMovieClip`” a pagina 338

“`MovieClip.removeMovieClip`” a pagina 313

else

Sintassi

```
else { istruzione/i };
```

Argomenti

istruzioni Una serie di istruzioni alternative da eseguire se la condizione specificata nell'istruzione `if` è `false`.

Descrizione

Azione; specifica le azioni, le proposizioni, gli argomenti o altre istruzioni condizionali da eseguire se l'istruzione `if` iniziale restituisce `false`.

Lettore

Flash 4 o versione successiva.

Vedere anche

“`if`” a pagina 269

eq (uguale; specifico per stringhe)

Sintassi

espressione1 eq *espressione2*

Argomenti

espressione1, *espressione2* Numeri, stringhe o variabili.

Descrizione

Operatore di confronto; verifica l'uguaglianza di due espressioni e restituisce `true` se *espressione1* equivale a *espressione2*; altrimenti restituisce `false`.

Lettore

Flash 1 o versione successiva. Questo operatore è diventato obsoleto in Flash 5 ed è consigliato l'uso del nuovo operatore (uguaglianza) `==`.

Vedere anche

`"=="` (uguaglianza)" a pagina 208

escape

Sintassi

`escape(espressione)`;

Argomenti

espressione L'espressione da convertire in stringa e da codificare in un formato con codifica URL.

Descrizione

Funzione; converte l'argomento in una stringa in formato con codifica URL dove tutti i caratteri alfanumerici vengono sostituiti dalle sequenze esadecimali di `escape` precedute da `%`.

Lettore

Flash 5 o versione successiva.

Esempio

```
escape("Hello{[World]}");
```

Il risultato di tale codice è il seguente:

```
Hello%7B%5BWorld%5D%7D
```

Vedere anche

`"unescape"` a pagina 371

eval

Sintassi

```
eval(espressione);
```

Argomenti

espressione Una stringa contenente il nome di una variabile, una proprietà, un oggetto o un clip filmato da recuperare.

Descrizione

Funzione; accede alle variabili, proprietà, oggetti o clip filmato in base al loro nome. Se *espressione* è una variabile o una proprietà, viene restituito il valore corrispondente. Se *espressione* è un oggetto o un clip filmato, viene restituito un riferimento ad esso. Se l'elemento definito in *espressione* non è reperibile, viene restituito un valore indefinito.

In Flash 4, la funzione `eval` era usata per simulare le matrici, in Flash 5 si consiglia a tale scopo l'uso dell'oggetto `Array`.

Nota: L'azione `eval` di ActionScript non equivale alla funzione `eval` di JavaScript e non può essere usata per valutare le istruzioni.

Letttore

Flash 5 o versione successiva per avvalersi delle funzionalità complete. La funzione `eval` può essere usata in fase di esportazione in Flash 4 Player, ma è necessario usare la sintassi della barra rovesciata e, inoltre, è possibile accedere solo alle variabili, non alle proprietà o agli oggetti.

Esempio

L'esempio seguente usa `eval` per determinare il valore della variabile `x` e lo assegna a `y`.

```
x = 3;  
y = eval("x");
```

L'esempio seguente usa la funzione `eval` per fare riferimento all'oggetto clip filmato associato all'istanza di clip filmato `Ball` sullo stage.

```
eval("_root.Ball");
```

Vedere anche

"Array (oggetto)" a pagina 215

evaluate

Sintassi

istruzione;

Argomenti

Nessuno.

Descrizione

Azione; crea una nuova riga vuota e vi inserisce un ; per l'immissione di una sola istruzione di script tramite il campo Espressione nel pannello Azioni. Inoltre l'istruzione *evaluate* consente agli utenti che creano gli script in Modalità normale nel pannello Azioni di Flash 5 di chiamare le funzioni.

Lettore

Flash 5 o versione successiva.

_focusrect

Sintassi

_focusrect = *Booleano*;

Argomenti

Booleano *true* o *false*.

Descrizione

Proprietà (globale); specifica se visualizzare un rettangolo giallo attorno al pulsante attivo. Il valore predefinito è *true* (diverso da zero) e visualizza un rettangolo giallo attorno al pulsante o al campo di testo correntemente attivo quando l'utente preme il tasto TAB per spostarsi. Specificare *false* per visualizzare solo lo stato "su" del pulsante (qualora ve ne sia uno definito) quando l'utente si sposta.

Lettore

Flash 4 o versione successiva.

for

Sintassi

```
for(inizializzazione; condizione; passaggio); {  
  istruzione;  
}
```

Argomenti

inizializzazione Espressione da valutare prima di iniziare la sequenza di iterazioni del ciclo, in genere un'espressione di assegnazione. Per questo argomento è consentita anche un'istruzione *var*.

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

passaggio Espressione da valutare dopo ciascuna iterazione del ciclo, in genere un'espressione di assegnazione che usa gli operatori `++` (incremento) o `--` (decremento).

istruzione Un'istruzione all'interno del corpo del ciclo da eseguire.

Descrizione

Azione; una struttura di ciclo che valuta l'espressione *inizializzazione* una volta, quindi inizia una sequenza di iterazioni durante le quali, fino a quando *condizione* restituisce `true`, viene eseguita *istruzione* e viene valutata l'espressione *passaggio*.

Alcune proprietà non possono essere enumerate dall'azione `for` o `for..in`. È il caso, ad esempio, dei metodi predefiniti dell'oggetto `Array` (`Array.sort` e `Array.reverse`) che non sono inclusi nell'enumerazione di un oggetto `Array`, così come non sono enumerate le proprietà dei clip filmato, quali `_x` e `_y`.

Letture

Flash 5 o versione successiva.

Esempio

L'esempio seguente usa il ciclo `for` per inserire gli elementi in una matrice:

```
for(i=0; i<10; i++) {  
    array [i] = (i + 5)*10;  
}
```

Il risultato è la matrice seguente:

```
[50, 60, 70, 80, 90, 100, 110, 120, 130, 140]
```

L'esempio seguente usa il ciclo `for` per eseguire più volte la stessa azione. Nel codice seguente, il ciclo `for` è usato per addizionare i numeri da 1 a 100.

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

Vedere anche

“`++` (incremento)” a pagina 183
“`--` (decremento)” a pagina 182
“`for..in`” a pagina 260
“`var`” a pagina 373

for..in

Sintassi

```
for(variabileiterazione in oggetto){  
  istruzione; }
```

Argomenti

variabileiterazione Il nome di una variabile che agisce da elemento di iterazione e fa riferimento a ciascuna proprietà di un oggetto o elemento di una matrice.

oggetto Il nome di un oggetto su cui eseguire l'iterazione.

istruzione Istruzione da eseguire per ciascuna iterazione.

Descrizione

Azione; esegue un ciclo attraverso le proprietà di un oggetto o gli elementi di una matrice ed esegue *istruzione* per ciascuna proprietà.

Alcune proprietà non possono essere enumerate dall'azione `for` o `for..in`. È il caso, ad esempio, dei metodi predefiniti dell'oggetto Array (`Array.sort` e `Array.reverse`) che non sono inclusi nell'enumerazione di un oggetto Array, così come non sono enumerate le proprietà dei clip filmato, quali `_x` e `_y`.

Il costrutto `for...in` esegue le iterazioni sulle proprietà degli oggetti della catena di prototipi dell'oggetto base dell'iterazione. Se il prototipo dell'oggetto secondario è un oggetto principale, l'esecuzione dell'iterazione sulle proprietà dell'oggetto secondario con il ciclo `for...in` comporterà l'esecuzione dell'iterazione anche sulle proprietà dell'oggetto principale.

Lettore

Flash 5 o versione successiva.

Esempio

L'esempio seguente usa il ciclo "for..in" per eseguire l'iterazione sulle proprietà di un oggetto.

```
myObject = { name:'Tara', age:27, city:'San Francisco' };  
for (name in myObject) {  
  trace ("myObject." + name + " = " + myObject[name]);  
}
```

Il risultato di questo esempio è il seguente:

```
myObject.name = Tara  
myObject.age = 27  
myObject.city = San Francisco
```

L'esempio seguente usa l'operatore "typeof" con il ciclo "for..in" per eseguire l'iterazione su un tipo particolare di oggetto secondario.

```
for (name in myMovieClip) {  
  if (typeof (myMovieClip[name]) = "movieclip") {
```

```

        trace ("I have a movie clip child named " + name);
    }
}

```

L'esempio seguente enumera gli oggetti secondari di un clip filmato e sposta ciascuno di essi al fotogramma 2 nella rispettiva linea temporale. Il clip filmato `RadioButtonGroup` è un oggetto principale con diversi oggetti secondari: `_RedRadioButton_`, `_GreenRadioButton_` e `_BlueRadioButton_`.

```

for (var name in RadioButtonGroup) {
    RadioButtonGroup[name].gotoAndStop(2);
}

```

_framesloaded

Sintassi

nomeistanza.`_framesloaded`

Argomenti

nomeistanza Il nome dell'istanza di clip filmato da valutare.

Descrizione

Proprietà (sola lettura); il numero di fotogrammi che sono stati caricati da un filmato in streaming. Questa proprietà consente di determinare se il contenuto di un fotogramma specifico e di tutti i fotogrammi che lo precedono sono stati caricati e sono disponibili localmente nel browser dell'utente. È una proprietà utile per monitorare il processo di scaricamento di filmati di grandi dimensioni. È possibile, ad esempio, visualizzare un messaggio che informa gli utenti sull'avanzamento del processo di caricamento del filmato fino a raggiungimento di un fotogramma specifico.

Lettore

Flash 4 o versione successiva.

Esempio

L'esempio seguente usa la proprietà `_framesloaded` per coordinare l'inizio della riproduzione del filmato con il numero di fotogramma specificato.

```

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

```

fscommand

Sintassi

`fscommand(comando, argomenti);`

Argomenti

comando Una stringa passata all'applicazione host per un uso indeterminato.

argomenti Una stringa passata all'applicazione host per un uso indeterminato.

Descrizione

Azione; consente al filmato Flash di comunicare con il programma in cui si trova Flash Player. In un browser Web, l'azione `fscommand` chiama la funzione `nomefilmato_Dofsccommand` JavaScript nella pagina HTML che contiene il filmato Flash, dove `nomefilmato` è il nome di Flash Player assegnato dall'attributo `NAME` del tag `EMBED` o dalla proprietà `ID` del tag `OBJECT`. Se a Flash Player è stato assegnato il nome `Filmato`, la funzione JavaScript chiamata è `Filmato_Dofsccommand`.

Lettores

Flash 3 o versione successiva.

function

Sintassi

```
funzione nomefunzione ([argomento0, argomento1,...argomentoN]){  
  istruzione/i  
}
```

```
funzione ([argomento0, argomento1,...argomentoN]){  
  istruzione/i  
}
```

Argomenti

nomefunzione Il nome della nuova funzione.

argomento Zero o più stringhe, numeri o oggetti da passare a funzione.

istruzioni Zero o più istruzioni ActionScript definite per il corpo di funzione.

Descrizione

Azione; una serie di istruzioni che vengono definite per eseguire una determinata attività. È possibile *dichiarare*, o definire, una funzione in una posizione, quindi chiamarla da altri script in un filmato. Quando si definisce una funzione, è possibile anche specificarne gli argomenti. Gli argomenti sono dei segnaposto per i valori sui quali opererà la funzione. Ogni volta che si chiama una funzione è possibile passarle diversi argomenti, detti anche parametri.

L'azione `return` nelle *istruzioni* di una funzione determina la restituzione di un valore.

Uso 1: dichiara una funzione e specifica *nomefunzione*, *argomenti* e *istruzione/i*. Quando si chiama una funzione, si invoca la dichiarazione della funzione. È consentito il riferimento anticipato ad essa, ossia all'interno della stessa lista di azioni è possibile dichiarare una funzione dopo averla chiamata. La dichiarazione di una funzione sostituisce qualsiasi dichiarazione precedente della stessa. È possibile usare questa sintassi ogni qualvolta è consentita un'istruzione.

Uso 2: crea una funzione anonima e la restituisce. Questa sintassi si usa nelle espressioni e il suo uso è consigliato principalmente per l'inserimento di metodi negli oggetti.

Letto

Flash 5 o versione successiva.

Esempio

Uso 1. L'esempio seguente definisce la funzione `sqr` che accetta un argomento, quindi restituisce il valore `square(x*x)` dell'argomento. Se la funzione viene dichiarata e usata all'interno dello stesso script, la relativa dichiarazione può apparire dopo una chiamata alla funzione.

```
y=sqr(3);  
function sqr(x) {  
    return x*x;  
}
```

Uso 2. L'esempio seguente definisce l'oggetto `Circle`:

```
function Circle(radius) {  
    this.radius = radius;  
}
```

La seguente istruzione definisce una funzione anonima che calcola l'area del cerchio, quindi la associa come metodo all'oggetto `Circle`:

```
Circle.prototype.area = function () {return Math.PI * this.radius  
* this.radius}
```

ge (maggiore di o uguale a; specifico per stringhe)

Sintassi

espressione1 ge *espressione2*

Argomenti

espressione1, *espressione2* Numeri, stringhe o variabili.

Descrizione

Operatore (confronto); confronta *espressione1* ed *espressione2*, quindi restituisce `true` se *espressione1* è maggiore o uguale a *espressione2*; altrimenti restituisce `false`.

Lettore

Flash 4 o versione successiva. Questo operatore è diventato obsoleto in Flash 5 ed è consigliato l'uso del nuovo operatore `>=`.

Vedere anche

`">= (maggiore o uguale a)"` a pagina 209

getProperty

Sintassi

```
getProperty(nomeistanza, proprietà);
```

Argomenti

nomeistanza Il nome dell'istanza del clip filmato di cui si accede alla proprietà.

proprietà Proprietà di un clip filmato, ad esempio la coordinata *x* o *y*.

Descrizione

Funzione, restituisce il valore della *proprietà* specificata per l'istanza di clip filmato.

Lettore

Flash 4 o versione successiva.

Esempio

L'esempio seguente recupera la coordinata dell'asse orizzontale (*_x*) per il clip filmato `myMovie`.

```
getProperty(_root.myMovie_item._x);
```

getTimer

Sintassi

```
getTimer();
```

Argomenti

Nessuno.

Descrizione

Funzione; restituisce il numero di millesimi di secondo trascorsi dall'inizio della riproduzione del filmato.

Lettore

Flash 4 o versione successiva.

getURL

Sintassi

```
getURL(url [, finestra [, variabili]]);
```

Argomenti

url L'URL presso cui è reperibile il documento. Tale URL deve trovarsi nello stesso sottodominio dell'URL che contiene il filmato.

finestra Argomento opzionale che specifica la finestra o il frame HTML in cui caricare il documento. Immettere il nome di una finestra specifica o selezionare una dei seguenti nomi di destinazione riservati:

- *_self* indica il frame corrente nella finestra corrente.
- *_blank* indica una finestra nuova.
- *_parent* indica il frame principale rispetto a quello corrente.
- *_top* indica il frame di primo livello nella finestra corrente.

variabili Argomento opzionale che specifica un metodo per l'invio di variabili. Omettere questo argomento in assenza di variabili; in caso contrario, specificare se caricare le variabili tramite il metodo GET o POST. GET accoda le variabili in fondo all'URL e si usa in caso di un numero limitato di variabili. POST invia le variabili in un'intestazione HTTP separata e si usa in caso di lunghe stringhe di variabili.

Descrizione

Azione; carica un documento da un URL specifico in una finestra oppure passa le variabili a un'altra applicazione presso l'URL definito. Prima di provare questa azione, accertarsi che il percorso specificato per il file da caricare sia corretto. Per usare un URL assoluto, ad esempio, *http://www.server.com*, si deve disporre di una connessione di rete.

Lettore

Flash 2 o versione successiva. I metodi GET e POST sono disponibili solo per Flash 4 e versioni successive del lettore.

Esempio

Questo esempio carica un URL nuovo in una finestra vuota del browser. L'azione `getURL` usa la variabile `incomingAd` come parametro *url* in modo che sia possibile cambiare l'URL senza dover modificare il filmato Flash. I valori della variabile `incomingAd` vengono passati a Flash direttamente nel filmato tramite l'azione `loadVariables`.

```
on(release) {  
    getURL(incomingAd, "_blank");  
}
```

Vedere anche

“loadVariables” a pagina 284

“XML.send” a pagina 393

“XML.sendAndLoad” a pagina 393

“XMLSocket.close” a pagina 403

getVersion

Sintassi

```
getVersion();
```

Argomenti

Nessuno.

Descrizione

Funzione; restituisce una stringa contenente le informazioni relative alla versione di Flash Player e alla piattaforma in uso.

Non funziona in modalità di prova filmato e restituirà solo informazioni per Flash Player 5 o versioni successive.

Esempio

Il seguente è un esempio di stringa restituita dalla funzione getVersion:

```
WIN 5,0,17,0
```

Ciò significa che la piattaforma è Windows, il numero di versione principale di Flash Player è 5 e quello della versione secondaria è 17 (5.0r17).

Lettore

Flash 5 o versione successiva.

gotoAndPlay

Sintassi

```
gotoAndPlay(scena, fotogramma);
```

Argomenti

scena Il nome della scena a cui viene inviato l'indicatore di riproduzione.

fotogramma Il numero di fotogramma a cui viene inviato l'indicatore di riproduzione.

Descrizione

Azione; invia l'indicatore di riproduzione al fotogramma specificato in una scena e inizia la riproduzione da tale fotogramma. Se non viene specificata una scena, l'indicatore di riproduzione passa al fotogramma specificato all'interno della scena corrente.

Lettore

Flash 2 o versione successiva.

Esempio

Quando l'utente sceglie un pulsante a cui è assegnata un'azione `gotoAndPlay`, l'indicatore di riproduzione passa al fotogramma 16 e inizia la riproduzione.

```
on(release) {  
    gotoAndPlay(16);  
}
```

gotoAndStop

Sintassi

```
gotoAndStop(scena, fotogramma);
```

Argomenti

scena Il nome della scena a cui viene inviato l'indicatore di riproduzione.

fotogramma Il numero di fotogramma a cui viene inviato l'indicatore di riproduzione.

Descrizione

Azione; invia l'indicatore di riproduzione al fotogramma specificato in una scena e lo blocca in tale posizione. Se non viene specificata una scena, l'indicatore di riproduzione passa al fotogramma all'interno della scena corrente.

Lettore

Flash 2 o versione successiva.

Esempio

Quando l'utente sceglie un pulsante a cui è assegnata un'azione `gotoAndStop`, l'indicatore di riproduzione passa al fotogramma 5 e interrompe la riproduzione del filmato.

```
on(release) {  
    gotoAndStop(5);  
}
```

gt (maggiore di; specifico per stringhe)

Sintassi

```
espressione1 gt espressione2
```

Argomenti

espressione1, *espressione2* Numeri, stringhe o variabili.

Descrizione

Operatore (confronto); confronta *espressione1* ed *espressione2*, quindi restituisce `true` se *espressione1* è maggiore di *espressione2*; altrimenti restituisce `false`.

Lettore

Flash 4 o versione successiva. Questo operatore è diventato obsoleto in Flash 5 ed è consigliato l'uso del nuovo operatore >.

Vedere anche

"> (maggiore di)" a pagina 209

_height

Sintassi

```
nomeistanza._height  
nomeistanza._height = valore;
```

Argomenti

nomeistanza Il nome dell'istanza di un clip filmato la cui proprietà *_height* deve essere impostata o richiamata.

valore Numero intero che specifica l'altezza del filmato in pixel.

Descrizione

Proprietà; imposta e recupera l'altezza dello spazio occupato dal contenuto di un filmato. Nelle versioni precedenti di Flash, *_height* e *_width* erano proprietà di sola lettura, mentre in Flash 5 è possibile impostarle.

Lettore

Flash 4 o versione successiva.

Esempio

L'esempio di codice seguente imposta l'altezza e la larghezza di un clip filmato quando l'utente fa clic con il mouse.

```
onClipEvent(mouseDown) {  
    _width=200;  
    _height=200;  
}
```

_highquality

Sintassi

```
_highquality = valore;
```

Argomenti

valore Il livello di antialiasing applicato al filmato. Specificare 2 (Ottima) per applicare un livello di alta qualità con smussatura delle bitmap sempre attiva. Specificare 1 (Alta qualità) per applicare l'antialiasing; in tal modo le bitmap verranno smussate se il filmato non contiene animazioni. Specificare 0 (Bassa qualità) per impedire l'antialiasing.

Descrizione

Proprietà (globale); specifica il livello di antialiasing applicato al filmato corrente.

Lettore

Flash 4 o versione successiva.

Vedere anche

“_quality” a pagina 337

“toggleHighQuality” a pagina 368

if

Sintassi

```
if(condizione) {  
  
    istruzione;  
  
}
```

Argomenti

condizionale Espressione che restituisce true o false. Ad esempio, `if(nome == "Erica")` valuta la variabile `nome` per verificare se il nome è "Erica".

istruzioni Le istruzioni da eseguire se o quando la condizione restituisce true.

Descrizione

Azione; valuta una condizione per determinare l'azione successiva in un filmato. Se la condizione è soddisfatta, ovvero restituisce true, Flash esegue l'istruzione che segue. Usare `if` per creare codice condizionale negli script.

Lettore

Flash 4 o versione successiva.

Vedere anche

“else” a pagina 255

“for” a pagina 258

“for..in” a pagina 260

ifFrameLoaded

Sintassi

```
ifFrameLoaded(scena, fotogramma) {  
  
    istruzione;  
  
}  
  
ifFrameLoaded(fotogramma) {  
  
    istruzione;  
  
}
```

Argomenti

scena La scena alla quale viene inviata la richiesta.

fotogramma Il numero o l'etichetta del fotogramma da caricare prima di eseguire l'istruzione successiva.

Descrizione

Azione; verifica se il contenuto di un determinato fotogramma è disponibile localmente. Usare `ifFrameLoaded` per riprodurre una semplice animazione durante il caricamento del resto del filmato sul computer locale. La differenza tra `_framesloaded` e `ifFrameLoaded` è che `_framesloaded` consente di aggiungere le istruzioni `if` o `else`, mentre `ifFrameLoaded` consente di specificare un numero di fotogrammi in una sola istruzione.

Letttore

Flash 3 o versione successiva. L'azione `ifFrameLoaded` è diventata obsoleta in Flash 5 ed è consigliato l'uso di `_framesloaded`.

Vedere anche

"`_framesloaded`" a pagina 261

#include

Sintassi

```
#include "nomefile.as";
```

Argomenti

nomefile.as Il nome del file da includere; `.as` è l'estensione del file consigliata.

Descrizione

Azione; include il contenuto del file specificato nell'argomento quando il filmato viene provato, pubblicato o esportato. L'azione `#include` viene invocata durante la prova, la pubblicazione o l'esportazione e viene verificata quando viene eseguito un controllo della sintassi.

Letttore

N/D

Infinity

Sintassi

```
Infinity
```

Argomenti

Nessuno.

Descrizione

Variabile di primo livello; variabile predefinita a cui è assegnato il valore conforme allo standard ECMA-262 di infinito.

Letttore

Flash 5 o versione successiva.

int

Sintassi

`int(valore);`

Argomenti

valore Numero da arrotondare in intero.

Descrizione

Funzione; converte un numero decimale nel numero intero più vicino.

Lettore

Flash 4 o versione successiva. Questa funzione è diventata obsoleta in Flash 5 ed è consigliato l'uso del nuovo metodo `Math.floor`.

Vedere anche

"`Math.floor`" a pagina 292

isFinite

Sintassi

`isFinite(espressione);`

Argomenti

espressione Valore booleano, variabile o altra espressione da valutare.

Descrizione

Funzione di primo livello; valuta l'argomento e restituisce `true` se si tratta di un numero finito, `false` se si tratta di un valore infinito o infinito negativo. La presenza di un valore infinito o infinito negativo indica una condizione di errore matematico quale una divisione per 0.

Lettore

Flash 5 o versione successiva.

Esempio

I seguenti sono esempi di valori restituiti per `isFinite`:

`isFinite(56)` restituisce `true`

`isFinite(Number.POSITIVE_INFINITY)` restituisce `false`

`isNaN(Number.POSITIVE_INFINITY)` restituisce `false`

isNaN

Sintassi

`isNaN(espressione);`

Argomenti

espressione Valore booleano, variabile o altra espressione da valutare.

Descrizione

Funzione di primo livello; valuta l'argomento e restituisce `true` se il valore non è un numero (NaN) a indicare la presenza di errori matematici.

Letture

Flash 5 o versione successiva.

Esempio

L'esempio seguente illustra i valori restituiti da `isNaN`:

`isNaN("Tree")` restituisce `true`

`isNaN(56)` restituisce `false`

`isNaN(Number.POSITIVE_INFINITY)` restituisce `false`

Key (oggetto)

L'oggetto `Key` è un oggetto di primo livello a cui si può accedere senza usare una funzione di costruzione. I metodi validi per l'oggetto `Key` consentono di creare un'interfaccia che l'utente può controllare tramite una tastiera standard. Le proprietà dell'oggetto `Key` sono valori costanti che rappresentano i tasti usati di solito per controllare i giochi. Per un elenco completo dei valori dei codici tasto, consultare l'appendice B "Tasti della tastiera e valori dei codici tasto".

Esempio

```
onClipEvent (enterFrame) {  
    if(Key.isDown(Key.RIGHT)) {  
        setProperty ("", _x, _x+10);  
    }  
}  
or  
onClipEvent (enterFrame) {  
    if(Key.isDown(39)) {  
        setProperty("", _x, _x+10);  
    }  
}
```


Riepilogo dei metodi validi per l'oggetto Key

Metodo	Descrizione
<code>getAscii</code> ;	Restituisce il valore ASCII dell'ultimo tasto premuto.
<code>getCode</code> ;	Restituisce il codice di tasto virtuale dell'ultimo tasto premuto.
<code>isDown</code> ;	Restituisce <code>true</code> se viene premuto il tasto specificato nell'argomento.
<code>isToggled</code> ;	Restituisce <code>true</code> se Bloc Num o Bloc Maiusc sono attivati.

Riepilogo delle proprietà valide per l'oggetto Key

Tutte le proprietà valide per l'oggetto Key sono costanti.

Proprietà	Descrizione
<code>BACKSPACE</code>	Costante associata al valore del codice tasto per il tasto Backspace (9).
<code>CAPSLCK</code>	Costante associata al valore del codice tasto per il tasto Bloc Maiusc (20).
<code>CONTROL</code>	Costante associata al valore del codice tasto per il tasto Ctrl (17).
<code>DELETEKEY</code>	Costante associata al valore del codice tasto per il tasto Canc (46).
<code>DOWN</code>	Costante associata al valore del codice tasto per il tasto Freccia giù (40).
<code>END</code>	Costante associata al valore del codice tasto per il tasto Fine (35).
<code>ENTER</code>	Costante associata al valore del codice tasto per il tasto Invio (13).
<code>ESCAPE</code>	Costante associata al valore del codice tasto per il tasto Esc (27).
<code>HOME</code>	Costante associata al valore del codice tasto per il tasto Home (36).
<code>INSERT</code>	Costante associata al valore del codice tasto per il tasto Ins (45).
<code>LEFT</code>	Costante associata al valore del codice tasto per il tasto Freccia sinistra (37).
<code>PGDN</code>	Costante associata al valore del codice tasto per il tasto Pagina giù (34).
<code>PGUP</code>	Costante associata al valore del codice tasto per il tasto Pagina su (33).
<code>RIGHT</code>	Costante associata al valore del codice tasto per il tasto Freccia destra (39).

Proprietà	Descrizione
SHIFT	Costante associata al valore del codice tasto per il tasto Maiusc (16).
SPACE	Costante associata al valore del codice tasto per la Barra spaziatrice (32).
TAB	Costante associata al valore del codice tasto per il tasto Tab (9).
UP	Costante associata al valore del codice tasto per il tasto Freccia su (38).

Key.BACKSPACE

Sintassi

Key.BACKSPACE

Argomenti

Nessuno.

Descrizione

Proprietà; costante associata al valore del codice tasto per il tasto Backspace (9).

Lettore

Flash 5 o versione successiva.

Key.CAPSLOCK

Sintassi

Key.CAPSLOCK

Argomenti

Nessuno.

Descrizione

Proprietà; costante associata al valore del codice tasto per il tasto Bloc Maiusc (20).

Lettore

Flash 5 o versione successiva.

Key.CONTROL

Sintassi

Key.CONTROL

Argomenti

Nessuno.

Descrizione

Proprietà; costante associata al valore del codice tasto per il tasto Ctrl (17).

Lettore

Flash 5 o versione successiva.

Key.DELETEKEY

Sintassi

Key.DELETEKEY

Argomenti

Nessuno.

Descrizione

Proprietà; costante associata al valore del codice tasto per il tasto Canc (46).

Lettore

Flash 5 o versione successiva.

Key.DOWN

Sintassi

Key.DOWN

Argomenti

Nessuno.

Descrizione

Proprietà; costante associata al valore del codice tasto per il tasto Freccia giù (40).

Lettore

Flash 5 o versione successiva.

Key.END

Sintassi

Key.END

Argomenti

Nessuno.

Descrizione

Proprietà; costante associata al valore del codice tasto per il tasto Fine (35).

Lettore

Flash 5 o versione successiva.

Key.ENTER

Sintassi

`Key.ENTER`

Argomenti

Nessuno.

Descrizione

Proprietà; costante associata al valore del codice tasto per il tasto Invio (13).

Lettore

Flash 5 o versione successiva.

Key.ESCAPE

Sintassi

`Key.ESCAPE`

Argomenti

Nessuno.

Descrizione

Proprietà; costante associata al valore del codice tasto per il tasto Esc (27).

Lettore

Flash 5 o versione successiva.

Key.getAscii

Sintassi

`Key.getAscii();`

Argomenti

Nessuno.

Descrizione

Metodo; restituisce il codice ASCII dell'ultimo tasto premuto o rilasciato.

Lettore

Flash 5 o versione successiva.

Key.getCode

Sintassi

```
Key.getCode();
```

Argomenti

Nessuno.

Descrizione

Metodo; restituisce il valore del codice di tasto dell'ultimo tasto premuto. Per verificare la corrispondenza tra il valore del codice tasto restituito e il tasto virtuale di una tastiera standard, consultare l'appendice B "Tasti della tastiera e valori dei codici tasto".

Lettore

Flash 5 o versione successiva.

Key.HOME

Sintassi

```
Key.HOME
```

Argomenti

Nessuno.

Descrizione

Proprietà; costante associata al valore del codice tasto per il tasto Home (36).

Lettore

Flash 5 o versione successiva.

Key.INSERT

Sintassi

```
Key.INSERT
```

Argomenti

Nessuno.

Descrizione

Proprietà; costante associata al valore del codice tasto per il tasto Ins (45).

Lettore

Flash 5 o versione successiva.

Key.isDown

Sintassi

`Key.isDown(codicetasto);`

Argomenti

codicetasto Il valore del codice tasto assegnato a un tasto specifico oppure la proprietà dell'oggetto `Key` associata a un tasto specifico. Nell'appendice B "Tasti della tastiera e valori dei codici tasto" sono elencati tutti i codici tasto associati ai tasti di una tastiera standard.

Descrizione

Metodo; restituisce `true` se il tasto specificato in *codicetasto* viene premuto. In ambiente Macintosh, i valori del codice tasto per i tasti Bloc Maiusc e Bloc Num sono identici.

Lettore

Flash 5 o versione successiva.

Key.isToggled

Sintassi

`Key.isToggled(codicetasto)`

Argomenti

codicetasto Il codice tasto per Bloc Maiusc (20) o Bloc Num (144).

Descrizione

Metodo; restituisce `true` se Bloc Maiusc o Bloc Num sono attivati/disattivati. In ambiente Macintosh, i valori del codice tasto per questi tasti sono identici.

Lettore

Flash 5 o versione successiva.

Key.LEFT

Sintassi

`Key.LEFT`

Argomenti

Nessuno.

Descrizione

Proprietà; costante associata al valore del codice tasto per il tasto Freccia sinistra (37).

Lettore

Flash 5 o versione successiva.

Key.PGDN

Sintassi

Key.PGDN

Argomenti

Nessuno.

Descrizione

Proprietà; costante associata al valore del codice tasto per il tasto Pagina giù (34).

Lettore

Flash 5 o versione successiva.

Key.PGUP

Sintassi

Key.PGUP

Argomenti

Nessuno.

Descrizione

Proprietà; costante associata al valore del codice tasto per il tasto Pagina su (33).

Lettore

Flash 5 o versione successiva.

Key.RIGHT

Sintassi

Key.RIGHT

Argomenti

Nessuno.

Descrizione

Proprietà; costante associata al valore del codice tasto per il tasto Freccia destra (39).

Lettore

Flash 5 o versione successiva.

Key.SHIFT

Sintassi

Key.SHIFT

Argomenti

Nessuno.

Descrizione

Proprietà; costante associata al valore del codice tasto per il tasto Maiusc (16).

Lettore

Flash 5 o versione successiva.

Key.SPACE

Sintassi

Key.SPACE

Argomenti

Nessuno.

Descrizione

Proprietà; costante associata al valore del codice tasto per la Barra spaziatrice (32).

Lettore

Flash 5 o versione successiva.

Key.TAB

Sintassi

Key.TAB

Argomenti

Nessuno.

Descrizione

Proprietà; costante associata al valore del codice tasto per il tasto Tab (9).

Lettore

Flash 5 o versione successiva.

Key.UP

Sintassi

Key.UP

Argomenti

Nessuno.

Descrizione

Proprietà; costante associata al valore del codice tasto per il tasto Freccia su (38).

Lettore

Flash 5 o versione successiva.

le (minore di o uguale a; specifico per stringhe)

Sintassi

espressione1 le *espressione2*

Argomenti

espressione1, *espressione2* Numeri, stringhe o variabili.

Descrizione

Operatore (confronto); confronta *espressione1* ed *espressione2*, quindi restituisce *true* se *espressione1* è minore o uguale a *espressione2*; altrimenti restituisce *false*.

Lettore

Flash 4 o versione successiva. Questo operatore è diventato obsoleto in Flash 5 ed è consigliato l'uso del nuovo operatore <=.

Vedere anche

"<= (minore o uguale a)" a pagina 205

length

Sintassi

`length(espressione);`

`length(variabile);`

Argomenti

espressione Una stringa qualsiasi.

variabile Il nome di una variabile.

Descrizione

Funzione stringa; restituisce la lunghezza della stringa specificata o il nome della variabile.

Lettore

Flash 4 o versione successiva. Questa funzione, così come tutte le funzioni stringa, è diventata obsoleta in Flash 5. Per eseguire le stesse operazioni è consigliabile usare i metodi e la proprietà `length` dell'oggetto `String`.

Esempio

L'esempio seguente restituisce la lunghezza della stringa `Hello`.

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

Il risultato è 5.

Vedere anche

“ ” (delimitatore di stringa)” a pagina 358

“String.length” a pagina 362

_level

Sintassi

```
_levelN;
```

Argomenti

N Numero intero non negativo che specifica un livello di profondità. Per impostazione predefinita il valore di `_level` è 0, il filmato alla base della struttura gerarchica.

Descrizione

Proprietà; un riferimento alla linea temporale del filmato principale di *live1toN*. È necessario caricare i filmati tramite l'azione `loadMovie` prima di poterli identificare tramite la proprietà `_level`.

In Flash Player, ai filmati viene assegnato un numero in base all'ordine di caricamento. Il filmato che è stato caricato per primo, verrà caricato al livello inferiore, il livello 0. Il filmato che si trova nel livello 0 determina la frequenza dei fotogrammi, il colore di sfondo e le dimensioni del fotogramma per tutti i filmati caricati successivamente. I filmati verranno quindi impilati nei livelli superiori al di sopra del filmato caricato nel livello 0. Il livello in cui si trova un clip filmato è denominato anche livello di profondità o profondità.

Lettore

Flash 4 o versione successiva.

Esempio

L'esempio seguente interrompe la riproduzione della linea temporale del filmato nel livello 0.

```
_level0.stop();
```

L'esempio seguente invia la linea temporale del filmato nel livello 4 al fotogramma 5. Il filmato nel livello 4 deve essere stato caricato con un'azione `loadMovie`.

```
_level4.gotoAndStop(5);
```

Vedere anche

“loadMovie” a pagina 283

“MovieClip.swapDepths” a pagina 315

loadMovie

Sintassi

```
loadMovie(url [,posizione/target, variabili]);
```

Argomenti

url URL assoluto o relativo per il file SWF da caricare. Il percorso relativo deve essere relativo al file SWF. Tale URL deve trovarsi nello stesso sottodominio dell'URL che contiene il filmato. Per essere usati in Flash Player o verificati in modalità di prova filmato in ambiente di creazione Flash, tutti i file SWF devono essere memorizzati nella stessa cartella e i nomi dei file non possono includere specifiche della cartella o dell'unità disco.

target Argomento opzionale che specifica un clip filmato target che verrà sostituito dal filmato caricato. Il filmato caricato eredita le proprietà di posizione, rotazione e scala del clip filmato identificato. Se si specifica un *target* non occorre specificare anche la *posizione* (livello) del filmato target, in quanto tali valori si equivalgono.

posizione Argomento opzionale che specifica il livello nel quale viene caricato il filmato. Il filmato caricato eredita le proprietà di posizione, rotazione e scala del clip filmato identificato. Per caricare il nuovo filmato oltre a quelli già esistenti, specificare un livello che non sia occupato da un altro filmato. Per sostituire un filmato esistente con il filmato caricato, specificare un livello che sia attualmente occupato da un altro filmato. Per sostituire il filmato originale e scaricare tutti i livelli, caricare il nuovo filmato nel livello 0; in tal modo, il filmato caricato determinerà la frequenza dei fotogrammi, il colore di sfondo e le dimensioni del fotogramma per tutti gli altri filmati caricati.

variabili Argomento opzionale che specifica un metodo per l'invio delle variabili associate al filmato da caricare. L'argomento deve essere costituito dalla stringa "GET" o "POST." Omettere questo argomento in assenza di variabili; in caso contrario, specificare se caricare le variabili tramite il metodo GET o POST . GET accoda le variabili in fondo all'URL e si usa in caso di un numero limitato di variabili. POST invia le variabili in un'intestazione HTTP separata e si usa in caso di lunghe stringhe di variabili.

Descrizione

Azione; riproduce filmati aggiuntivi senza chiudere Flash Player. In genere Flash Player visualizza un solo filmato (file SWF), quindi si chiude. L'azione `loadMovie` consente di visualizzare più filmati contemporaneamente o di passare da un filmato all'altro senza dover caricare un altro documento HTML.

È possibile caricare i filmati nei livelli già occupati da file SWF. In questo caso, il nuovo filmato sostituirà il file SWF esistente. Se si carica un nuovo filmato nel livello 0, vengono scaricati tutti i livelli e il file nel livello 0 viene sostituito dal nuovo file. Usare l'azione `loadVariables` per mantenere il filmato attivo e aggiornare le variabili con i nuovi valori.

Usare l'azione `unloadMovie` per rimuovere i filmati caricati con l'azione `loadMovie`.

Lettore

Flash 3 o versione successiva.

Esempio

Nell'esempio seguente l'istruzione `loadMovie` è associata a un pulsante di navigazione etichettato `Products`. Sullo stage è presente un clip filmato invisibile con il nome di istanza `dropZone`. L'azione `loadMovie` usa questo clip filmato come parametro `target` per caricare i prodotti nel file SWF, nella posizione corretta sullo stage.

```
on(release) {  
    loadMovie("products.swf",_root.dropZone);  
}
```

Vedere anche

“`unloadMovie`” a pagina 371

“`_level`” a pagina 282

loadVariables

Sintassi

```
loadVariables (url ,posizione [, variabili]);
```

Argomenti

url URL assoluto o relativo in cui si trovano le variabili. L'host dell'URL deve trovarsi nello stesso sottodominio del filmato se vi si accede tramite un browser Web.

posizione Livello o target delle variabili. In Flash Player, ai filmati viene assegnato un numero in base all'ordine di caricamento. Il primo filmato viene caricato nel livello inferiore (livello 0). All'interno dell'azione `loadMovie` è necessario specificare un numero di livello per tutti i filmati successivi. È un argomento opzionale.

variabili Argomento opzionale che specifica un metodo per l'invio di variabili. Omettere questo argomento in assenza di variabili; in caso contrario, specificare se caricare le variabili tramite il metodo `GET` o `POST`. `GET` accoda le variabili in fondo all'URL e si usa in caso di un numero limitato di variabili. `POST` invia le variabili in un'intestazione HTTP separata e si usa in caso di lunghe stringhe di variabili.

Descrizione

Azione; legge i dati da un file esterno, quale un file di testo o un testo creato tramite script CGI, Active Server Pages (ASP) o Personal Home Page (PHP), e imposta i valori delle variabili in un filmato o in un clip filmato. Questa azione consente anche di aggiornare le variabili nel filmato attivo assegnando ad esse nuovi valori.

Il testo nell'URL specificato deve essere in formato MIME standard *application/x-www-urlencoded* (un formato standard usato dagli script CGI). Il filmato e le variabili da caricare devono risiedere nello stesso sottodominio. Non vi sono limitazioni sul numero di variabili che è possibile specificare. Ad esempio, questa frase definisce diverse variabili:

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

Letttore

Flash 4 o versione successiva.

Esempio

L'esempio seguente carica delle informazioni da un file di testo nei campi di testo della linea temporale principale (livello 0). I nomi delle variabili dei campi di testo devono corrispondere ai nomi delle variabili nel file data.txt.

```
on(release) {  
    loadVariables("data.txt", 0);  
}
```

Vedere anche

“getURL” a pagina 265

“MovieClip.loadMovie” a pagina 310

“MovieClip.loadVariables” a pagina 310

lt (minore di; specifico per stringhe)

Sintassi

espressione1 lt *espressione2*

Argomenti

espressione1, *espressione2* Numeri, stringhe o variabili.

Descrizione

Operatore (confronto); confronta *espressione1* ed *espressione2*, quindi restituisce *true* se *espressione1* è minore di *espressione2*; altrimenti restituisce *false*.

Letttore

Flash 4 o versione successiva. Questo operatore è diventato obsoleto in Flash 5 ed è consigliato l'uso del nuovo operatore < (minore di).

Vedere anche

“< (minore di)” a pagina 203

Math (oggetto)

L'oggetto Math è un oggetto di primo livello a cui si può accedere senza usare una funzione di costruzione.

I metodi e le proprietà di questo oggetto consentono di accedere e gestire le costanti e le funzioni matematiche. Tutte le proprietà e i metodi dell'oggetto Math sono statici e la loro chiamata richiede l'uso della sintassi

`Math.metodo(argomento)` o `Math.costante`. In ActionScript, le costanti sono definite come numeri in virgola mobile e doppia precisione secondo le specifiche IEEE-754.

L'oggetto Math è totalmente supportato in ambiente Flash 5 Player. In Flash 4 Player, i metodi dell'oggetto Math funzionano, ma sono emulati per approssimazione e la loro accuratezza può risultare inferiore rispetto alle funzioni matematiche non emulate supportate da Flash 5 Player.

Diversi metodi dell'oggetto Math richiedono il valore degli angoli in radianti come argomento. La seguente equazione consente di calcolare il valore in radianti o di passare semplicemente l'equazione (immettendo un valore corrispondente in gradi) come parametro.

Per calcolare il valore in radianti, usare la formula seguente:

```
radianti = Math.PI/180 * gradi
```

Nell'esempio seguente, l'equazione viene passata come argomento in modo da calcolare il seno di un angolo di 45 gradi:

```
Math.SIN(Math.PI/180 * 45) equivale a Math.SIN(.7854)
```

Riepilogo dei metodi validi per l'oggetto Math

Metodo	Descrizione
abs	Calcola il valore assoluto.
acos	Calcola l'arco coseno.
asin	Calcola l'arco seno.
atan	Calcola l'arcotangente.
atan2	Calcola l'angolo dall'asse x al punto.
ceil	Arrotonda un numero per eccesso all'intero più vicino.
cos	Calcola il coseno.
exp	Calcola un valore esponenziale.
floor	Arrotonda un numero per difetto all'intero più vicino.
log	Calcola il logaritmo naturale.
max	Restituisce il numero maggiore tra due interi.
min	Restituisce il numero minore tra due interi.
pow	Calcola x elevato alla y .
random	Restituisce un numero pseudo-casuale compreso tra 0,0 e 1,2.
round	Arrotonda al numero intero più vicino.
sin	Calcola il seno.
sqrt	Calcola la radice quadrata.
tan	Calcola la tangente.

Riepilogo delle proprietà valide per l'oggetto Math

Tutte le proprietà valide per l'oggetto Math sono costanti.

Proprietà	Descrizione
E	Costante di Eulero, base dei logaritmi naturali (circa 2,718).
LN2	Il logaritmo naturale di 2 (circa 0,693).
LOG2E	Logaritmo di e in base 2 (circa 1,442).
LN10	Il logaritmo naturale di 10 (circa 2,302).

Proprietà	Descrizione
LOG10E	Logaritmo di e in base 10 (circa 0,434).
PI	Il rapporto tra la circonferenza di un cerchio e il suo diametro (circa 3,14159).
SQRT1_2	Il reciproco della radice quadrata di $1/2$ (circa 0,707).
SQRT2	La radice quadrata di 2 (circa 1,414).

Math.abs

Sintassi

`Math.abs(x);`

Argomenti

x Un numero.

Descrizione

Metodo; calcola e restituisce il valore assoluto del numero specificato dall'argomento x .

Lettores

Flash 5 o versione successiva. In Flash 4 Player, i metodi e le proprietà dell'oggetto Math sono emulati per approssimazione e la loro accuratezza può risultare inferiore rispetto alle funzioni matematiche non emulate supportate da Flash 5 Player.

Math.acos

Sintassi

`Math.acos(x);`

Argomenti

x Un numero tra $-1,0$ e $1,0$.

Descrizione

Metodo; calcola e restituisce l'arco coseno del numero specificato dall'argomento x , in radianti.

Lettores

Flash 5 o versione successiva. In Flash 4 Player, i metodi e le proprietà dell'oggetto Math sono emulati per approssimazione e la loro accuratezza può risultare inferiore rispetto alle funzioni matematiche non emulate supportate da Flash 5 Player.

Math.asin

Sintassi

`Math.asin(x);`

Argomenti

x Un numero tra $-1,0$ e $1,0$.

Descrizione

Metodo; calcola e restituisce l'arco seno del numero specificato nell'argomento *x*, in radianti.

Lettore

Flash 5 o versione successiva. In Flash 4 Player, i metodi e le proprietà dell'oggetto Math sono emulati per approssimazione e la loro accuratezza può risultare inferiore rispetto alle funzioni matematiche non emulate supportate da Flash 5 Player.

Math.atan

Sintassi

`Math.atan(x);`

Argomenti

x Un numero.

Descrizione

Metodo; calcola e restituisce l'arcotangente del numero specificato nell'argomento *x*. Restituisce un valore compreso tra il valore di π greco negativo diviso per 2 e π greco positivo diviso per 2.

Lettore

Flash 5 o versione successiva. In Flash 4 Player, i metodi e le proprietà dell'oggetto Math sono emulati per approssimazione e la loro accuratezza può risultare inferiore rispetto alle funzioni matematiche non emulate supportate da Flash 5 Player.

Math.atan2

Sintassi

`Math.atan2(y, x);`

Argomenti

x Numero che specifica la coordinata *x* del punto.

y Numero che specifica la coordinata *y* del punto.

Descrizione

Metodo; calcola e restituisce l'arcotangente delle coordinate y/x in radianti. Il valore restituito rappresenta l'angolo opposto all'angolo retto di un triangolo rettangolo, dove x è la lunghezza del lato adiacente e y la lunghezza del lato opposto.

Lettore

Flash 5 o versione successiva. In Flash 4 Player, i metodi e le proprietà dell'oggetto Math sono emulati per approssimazione e la loro accuratezza può risultare inferiore rispetto alle funzioni matematiche non emulate supportate da Flash 5 Player.

Math.ceil

Sintassi

```
Math.ceil(x);
```

Argomenti

x Numero o espressione.

Descrizione

Metodo; restituisce il valore arrotondato per eccesso del numero o espressione specificata. Il valore arrotondato per eccesso di un numero corrisponde all'intero più vicino, maggiore o uguale al numero stesso.

Lettore

Flash 5 o versione successiva. In Flash 4 Player, i metodi e le proprietà dell'oggetto Math sono emulati per approssimazione e la loro accuratezza può risultare inferiore rispetto alle funzioni matematiche non emulate supportate da Flash 5 Player.

Math.cos

Sintassi

```
Math.cos(x);
```

Argomenti

x Un angolo misurato in radianti.

Descrizione

Metodo; restituisce il coseno (un valore compreso tra $-1,0$ e $1,0$) dell'angolo specificato dall'argomento x . L'angolo x deve essere specificato in radianti. Per calcolare un radiante, usare l'espressione fornita nell'introduzione all'oggetto Math.

Letttore

Flash 5 o versione successiva. In Flash 4 Player, i metodi e le proprietà dell'oggetto Math sono emulati per approssimazione e la loro accuratezza può risultare inferiore rispetto alle funzioni matematiche non emulate supportate da Flash 5 Player.

Esempio

Math.E

Sintassi

`Math.E`

Argomenti

Nessuno.

Descrizione

Costante matematica; costituisce la base per i logaritmi naturali, espressa come e . Il valore di e è uguale a circa 2,71828.

Letttore

Flash 5 o versione successiva. In Flash 4 Player, i metodi e le proprietà dell'oggetto Math sono emulati per approssimazione e la loro accuratezza può risultare inferiore rispetto alle funzioni matematiche non emulate supportate da Flash 5 Player.

Math.exp

Sintassi

`Math.exp(x);`

Argomenti

x Esponente; numero o espressione.

Descrizione

Metodo; restituisce il valore della base del logaritmo naturale (e) elevato alla potenza specificata nell'argomento x . La costante `Math.E` fornisce il valore di e .

Letttore

Flash 5 o versione successiva. In Flash 4 Player, i metodi e le proprietà dell'oggetto Math sono emulati per approssimazione e la loro accuratezza può risultare inferiore rispetto alle funzioni matematiche non emulate supportate da Flash 5 Player.

Math.floor

Sintassi

```
Math.floor(x);
```

Argomenti

x Numero o espressione.

Descrizione

Metodo; restituisce il valore arrotondato in difetto del numero o espressione specificata nell'argomento *x*. Il valore arrotondato in difetto è il numero intero più vicino, minore o uguale al numero o all'espressione specificata.

Lettore

Flash 5 o versione successiva. In Flash 4 Player, i metodi e le proprietà dell'oggetto Math sono emulati per approssimazione e la loro accuratezza può risultare inferiore rispetto alle funzioni matematiche non emulate supportate da Flash 5 Player.

Esempio

L'esempio seguente restituisce il valore 12.

```
Math.floor(12.5);
```

Math.log

Sintassi

```
Math.log(x);
```

Argomenti

x Numero o espressione con un valore maggiore di 0.

Descrizione

Metodo; restituisce il logaritmo naturale dell'argomento *x*.

Lettore

Flash 5 o versione successiva. In Flash 4 Player, i metodi e le proprietà dell'oggetto Math sono emulati per approssimazione e la loro accuratezza può risultare inferiore rispetto alle funzioni matematiche non emulate supportate da Flash 5 Player.

Math.LOG2E

Sintassi

```
Math.LOG2E
```

Argomenti

Nessuno.

Descrizione

Costante matematica; rappresenta il logaritmo in base 2 della costante e (`Math.E`), espresso come $\log_2 e$, con un valore uguale a circa 1,442695040888963387.

Lettore

Flash 5 o versione successiva. In Flash 4 Player, i metodi e le proprietà dell'oggetto `Math` sono emulati per approssimazione e la loro accuratezza può risultare inferiore rispetto alle funzioni matematiche non emulate supportate da Flash 5 Player.

Math.LOG10E

Sintassi

`Math.LOG10E`

Argomenti

Nessuno.

Descrizione

Costante matematica; rappresenta il logaritmo in base 10 della costante e (`Math.E`), espresso come $\log_{10} e$, con un valore uguale a circa 0.43429448190325181667.

Lettore

Flash 5 o versione successiva. In Flash 4 Player, i metodi e le proprietà dell'oggetto `Math` sono emulati per approssimazione e la loro accuratezza può risultare inferiore rispetto alle funzioni matematiche non emulate supportate da Flash 5 Player.

Math.LN2

Sintassi

`Math.LN2`

Argomenti

Nessuno.

Descrizione

Costante matematica; rappresenta il logaritmo naturale di 2, espresso come $\log_e 2$, con un valore uguale a circa 0,69314718055994528623.

Lettore

Flash 5 o versione successiva. In Flash 4 Player, i metodi e le proprietà dell'oggetto `Math` sono emulati per approssimazione e la loro accuratezza può risultare inferiore rispetto alle funzioni matematiche non emulate supportate da Flash 5 Player.

Math.LN10

Sintassi

`Math.LN10`

Argomenti

Nessuno.

Descrizione

Costante matematica; rappresenta il logaritmo naturale di 10, espresso come $\log_e 10$, con un valore uguale a circa 2,3025850929940459011.

Lettore

Flash 5 o versione successiva. In Flash 4 Player, i metodi e le proprietà dell'oggetto `Math` sono emulati per approssimazione e la loro accuratezza può risultare inferiore rispetto alle funzioni matematiche non emulate supportate da Flash 5 Player.

Math.max

Sintassi

`Math.max(x, y);`

Argomenti

x Numero o espressione.

y Numero o espressione.

Descrizione

Metodo; valuta *x* e *y*, quindi restituisce il valore maggiore.

Lettore

Flash 5 o versione successiva. In Flash 4 Player, i metodi e le proprietà dell'oggetto `Math` sono emulati per approssimazione e la loro accuratezza può risultare inferiore rispetto alle funzioni matematiche non emulate supportate da Flash 5 Player.

Math.min

Sintassi

`Math.min(x, y);`

Argomenti

x Numero o espressione.

y Numero o espressione.

Descrizione

Metodo; valuta *x* e *y*, quindi restituisce il valore minore.

Lettore

Flash 5 o versione successiva. In Flash 4 Player, i metodi e le proprietà dell'oggetto Math sono emulati per approssimazione e la loro accuratezza può risultare inferiore rispetto alle funzioni matematiche non emulate supportate da Flash 5 Player.

Math.PI

Sintassi

Math.PI

Argomenti

Nessuno.

Descrizione

Costante matematica; rappresenta il rapporto tra la circonferenza di un cerchio e il suo diametro, espresso come pi greco, con un valore pari a circa 3,14159265358979.

Lettore

Flash 5 o versione successiva. In Flash 4 Player, i metodi e le proprietà dell'oggetto Math sono emulati per approssimazione e la loro accuratezza può risultare inferiore rispetto alle funzioni matematiche non emulate supportate da Flash 5 Player.

Math.pow

Sintassi

Math.pow(*x* , *y*);

Argomenti

x Numero da elevare alla potenza.

y Numero che specifica la potenza a cui elevare il valore *x*.

Descrizione

Metodo; calcola e restituisce *x* elevato alla *y*, x^y .

Lettore

Flash 5 o versione successiva. In Flash 4 Player, i metodi e le proprietà dell'oggetto Math sono emulati per approssimazione e la loro accuratezza può risultare inferiore rispetto alle funzioni matematiche non emulate supportate da Flash 5 Player.

Math.random

Sintassi

`Math.random()`;

Argomenti

Nessuno.

Descrizione

Metodo; restituisce un numero pseudo-casuale compreso tra 0,0 e 1,2.

Letture

Flash 5 o versione successiva. In Flash 4 Player, i metodi e le proprietà dell'oggetto Math sono emulati per approssimazione e la loro accuratezza può risultare inferiore rispetto alle funzioni matematiche non emulate supportate da Flash 5 Player.

Vedere anche

“random” a pagina 338

Math.round

Sintassi

`Math.round(x)`;

Argomenti

x Un numero.

Descrizione

Metodo; arrotonda il valore dell'argomento *x*, per eccesso o per difetto, al numero intero più vicino e restituisce il valore corrispondente.

Letture

Flash 5 o versione successiva. In Flash 4 Player, i metodi e le proprietà dell'oggetto Math sono emulati per approssimazione e la loro accuratezza può risultare inferiore rispetto alle funzioni matematiche non emulate supportate da Flash 5 Player.

Math.sin

Sintassi

`Math.sin(x)`;

Argomenti

x Un angolo misurato in radianti.

Descrizione

Metodo; calcola e restituisce il seno dell'angolo specificato, in radianti. Per calcolare un radiante, usare l'espressione fornita nell'introduzione all'oggetto Math.

Lettore

Flash 5 o versione successiva. In Flash 4 Player, i metodi e le proprietà dell'oggetto Math sono emulati per approssimazione e la loro accuratezza può risultare inferiore rispetto alle funzioni matematiche non emulate supportate da Flash 5 Player.

Vedere anche

“Math (oggetto)” a pagina 286

Math.sqrt

Sintassi

```
Math.sqrt(x);
```

Argomenti

x Numero o espressione con un valore maggiore o uguale 0.

Descrizione

Metodo; calcola e restituisce la radice quadrata del numero specificato.

Lettore

Flash 5 o versione successiva. In Flash 4 Player, i metodi e le proprietà dell'oggetto Math sono emulati per approssimazione e la loro accuratezza può risultare inferiore rispetto alle funzioni matematiche non emulate supportate da Flash 5 Player.

Math.SQRT1_2

Sintassi

```
Math.SQRT1_2
```

Argomenti

Nessuno.

Descrizione

Costante matematica; rappresenta il reciproco della radice quadrata di un mezzo (1/2), con un valore uguale a circa 0,707106781186.

Lettore

Flash 5 o versione successiva. In Flash 4 Player, i metodi e le proprietà dell'oggetto Math sono emulati per approssimazione e la loro accuratezza può risultare inferiore rispetto alle funzioni matematiche non emulate supportate da Flash 5 Player.

Math.SQRT2

Sintassi

```
Math.SQRT2
```

Argomenti

Nessuno.

Descrizione

Costante matematica; rappresenta la radice quadrata di 2, con un valore uguale a circa 1,414213562373.

Lettore

Flash 5 o versione successiva. In Flash 4 Player, i metodi e le proprietà dell'oggetto Math sono emulati per approssimazione e la loro accuratezza può risultare inferiore rispetto alle funzioni matematiche non emulate supportate da Flash 5 Player.

Math.tan

Sintassi

```
Math.tan(x);
```

Argomenti

x Un angolo misurato in radianti.

Descrizione

Metodo; calcola e restituisce la tangente dell'angolo specificato. Per calcolare un radiante, usare l'espressione fornita nell'introduzione all'oggetto Math.

Lettore

Flash 5 o versione successiva. In Flash 4 Player, i metodi e le proprietà dell'oggetto Math sono emulati per approssimazione e la loro accuratezza può risultare inferiore rispetto alle funzioni matematiche non emulate supportate da Flash 5 Player.

maxscroll

Sintassi

```
nome_variabile.maxscroll = x
```

Argomenti

nome_variabile Il nome di una variabile associata a un campo di testo.

x Il numero della riga corrispondente al valore massimo consentito per la proprietà `scroll` basato sull'altezza del campo di testo. È un valore di sola lettura impostato da Flash.

Descrizione

Proprietà di sola lettura che opera in congiunzione con la proprietà `scroll` per controllare la visualizzazione delle informazioni in un campo di testo. È possibile recuperare il valore questa proprietà, ma non modificarla.

Lettore

Flash 4 o versione successiva.

Vedere anche

"scroll" a pagina 341

mbchr

Sintassi

`mbchr(numero);`

Argomenti

numero Il numero da convertire in un carattere a più byte.

Descrizione

Funzione stringa; converte un numero in codice ASCII in un carattere a più byte.

Letture

Flash 4 o versione successiva. Questa funzione è diventata obsoleta in Flash 5 ed è consigliato l'uso del nuovo metodo `String.fromCharCode`.

Vedere anche

“`String.fromCharCode`” a pagina 361

mblength

Sintassi

`mblength(stringa);`

Argomenti

stringa Una stringa.

Descrizione

Funzione stringa; restituisce la lunghezza della stringa di caratteri a più byte.

Letture

Flash 4 o versione successiva. Questa funzione è diventata obsoleta in Flash 5 ed è consigliato l'uso dei metodi dell'oggetto `String`.

mbord

Sintassi

`mbord(carattere);`

Argomenti

carattere Il carattere da convertire in un numero a più byte.

Descrizione

Funzione stringa; converte il carattere specificato in un numero a più byte.

Letture

Flash 4 o versione successiva. Questa funzione è diventata obsoleta in Flash 5 ed è consigliato l'uso del nuovo metodo `String.charCodeAt`.

Vedere anche

“`String.fromCharCode`” a pagina 361

mbsubstring

Sintassi

`mbsubstring(valore, indice, numero);`

Argomenti

valore La stringa a più byte dalla quale estrarre una nuova stringa a più byte.

indice Il numero del primo carattere da estrarre.

numero Il numero di caratteri da includere nella stringa estratta, escluso il carattere specificato nell'argomento indice.

Descrizione

Funzione stringa; estrae una nuova stringa di caratteri a più byte da una stringa di caratteri a più byte.

Lettore

Flash 4 o versione successiva. Questa funzione è diventata obsoleta in Flash 5 ed è consigliato l'uso del nuovo metodo `string.substr`.

Vedere anche

“String.substr” a pagina 363

Mouse (oggetto)

Usare i metodi dell'oggetto Mouse per nascondere e mostrare il puntatore in un filmato. Per impostazione predefinita il puntatore è visibile, ma è possibile nascondarlo e usare un puntatore personalizzato creato come clip filmato.

Riepilogo dei metodi validi per l'oggetto Mouse

Metodo	Descrizione
<code>hide</code>	Nasconde il puntatore nel filmato.
<code>show</code>	Mostra il puntatore nel filmato.

Mouse.hide

Sintassi

`Mouse.hide();`

Argomenti

Nessuno.

Descrizione

Metodo; nasconde il puntatore in un filmato. Per impostazione predefinita il puntatore è visibile.

Lettore

Flash 5 o versione successiva.

Esempio

Il codice seguente, associato a un clip filmato nella linea temporale principale, nasconde il puntatore standard e imposta le posizioni *x* e *y* dell'istanza di clip filmato `customCursor` sulle posizioni *x* e *y* del mouse nella linea temporale principale:

```
onClipEvent(enterFrame){  
    Mouse.hide();  
    customCursorMC_x = _root._xmouse;  
    customCursorMC_y = _root._ymouse;  
}
```

Vedere anche

“_xmouse” a pagina 404

“_ymouse” a pagina 406

“Mouse.show” a pagina 301w

Mouse.show

Sintassi

`Mouse.show();`

Argomenti

Nessuno.

Descrizione

Metodo; rende visibile il puntatore in un filmato. Per impostazione predefinita il puntatore è visibile.

Lettore

Flash 5 o versione successiva.

Vedere anche

“_xmouse” a pagina 404

“_ymouse” a pagina 406

“Mouse.show” a pagina 301

MovieClip (oggetto)

I metodi dell'oggetto MovieClip forniscono le stesse funzionalità delle azioni standard applicabili a clip filmato. Vi sono inoltre metodi aggiuntivi che forniscono funzionalità non disponibili tramite le azioni standard elencate nella categoria Azioni del pannello Azioni. Non è necessario usare una funzione di costruzione per chiamare i metodi dell'oggetto MovieClip, bensì è possibile fare riferimento alle istanze di clip filmato per nome, usando la sintassi seguente:

```
anyMovieClip.play();  
anyMovieClip.gotoAndPlay(3);
```

Riepilogo dei metodi validi per l'oggetto MovieClip

Metodo	Descrizione
attachMovie	Associa un filmato nella libreria.
duplicateMovieClip	Duplica il clip filmato specificato.
getBounds	Restituisce i valori minimo e massimo delle coordinate x e y di un filmato rispetto a uno spazio di coordinate specificato.
getBytesLoaded	Restituisce il numero di byte caricati per il clip filmato specificato.
getBytesTotal	Restituisce la dimensione del clip filmato in byte.
getURL	Recupera un documento da un URL.
globalToLocal	Converte le coordinate di un punto da coordinate rispetto allo stage in coordinate locali rispetto al clip filmato specificato.
gotoAndPlay	Invia l'indicatore di riproduzione a un fotogramma specifico nel clip filmato e riproduce il filmato.
gotoAndStop	Invia l'indicatore di riproduzione a un fotogramma specifico nel clip filmato e interrompe la riproduzione del filmato.
hitTest	Restituisce true se il riquadro di limitazione del clip filmato specificato interseca il riquadro di limitazione del clip filmato target.
loadMovie	Carica il filmato specificato nel clip filmato.

Metodo	Descrizione
<code>loadVariables</code>	Carica le variabili da un URL o da un'altra posizione, nel clip filmato.
<code>localToGlobal</code>	Converte le coordinate di un punto da coordinate locali rispetto al clip filmato in coordinate globali rispetto allo stage.
<code>nextFrame</code>	Invia l'indicatore di riproduzione al fotogramma successivo del clip filmato.
<code>play</code>	Riproduce il clip filmato specificato.
<code>prevFrame</code>	Invia l'indicatore di riproduzione al fotogramma precedente del clip filmato.
<code>removeMovieClip</code>	Rimuove il clip filmato dalla linea temporale, se esso è stato creato tramite un'azione <code>duplicateMovieClip</code> o il metodo <code>attachMovie</code> .
<code>startDrag</code>	Rende un clip filmato mobile e ne avvia il trascinamento.
<code>stop</code>	Interrompe il filmato attualmente in riproduzione.
<code>stopDrag</code>	Interrompe l'azione di trascinamento in corso di qualunque clip filmato.
<code>swapDepths</code>	Scambia il livello di profondità del filmato specificato con quella del filmato che occupa il livello di profondità indicato.
<code>unloadMovie</code>	Rimuove un filmato caricato tramite il metodo <code>loadMovie</code> .

MovieClip.attachMovie

Sintassi

```
ClipFilmato.attachMovie(nomeId, nuovonome, profondità);
```

Argomenti

nomeId Il nome del filmato nella libreria da associare. Corrisponde al nome immesso nel campo Identificatore nella finestra di dialogo Proprietà di concatenamento del simbolo.

nuovonome Nome univoco dell'istanza del clip filmato da associare.

profondità Numero intero che specifica il livello di profondità in cui posizionare il filmato.

Descrizione

Metodo; crea una nuova istanza di un filmato nella libreria e la associa al filmato specificato in *ClipFilmato*. Usare l'azione o metodo `removeMovieClip` o `unloadMovie` per rimuovere un filmato associato tramite il metodo `attachMovie`.

Lettore

Flash 5 o versione successiva.

Vedere anche

“`removeMovieClip`” a pagina 338

“`unloadMovie`” a pagina 371

“`MovieClip.removeMovieClip`” a pagina 313

“`MovieClip.unloadMovie`” a pagina 315

MovieClip duplicateMovieClip

Sintassi

```
ClipFilmato.duplicateMovieClip(nuovonome, profondità);
```

Argomenti

nuovonome Identificatore univoco per il clip filmato duplicato.

profondità Un numero che specifica il livello di profondità in cui posizionare il filmato indicato.

Descrizione

Metodo; crea un'istanza del clip filmato specificato durante la riproduzione del filmato. La riproduzione dei clip filmato duplicati viene avviata sempre a partire dal fotogramma 1, a prescindere dal fotogramma su cui si trovava il clip filmato originale al momento della chiamata del metodo `duplicateMovieClip`. Le variabili nel clip filmato principale non vengono copiate nel clip filmato duplicato. L'eliminazione del clip filmato principale comporta l'eliminazione del clip filmato duplicato. I clip filmato aggiunti con il metodo `duplicateMovieClip` possono essere eliminati tramite l'azione o metodo `removeMovieClip`.

Lettore

Flash 5 o versione successiva.

Vedere anche

“removeMovieClip” a pagina 338

“MovieClip.removeMovieClip” a pagina 313

MovieClip.getBounds

Sintassi

```
ClipFilmato.getBounds(spazioCoordinateTarget);
```

Argomenti

spazioCoordinateTarget Il percorso target della linea temporale le cui coordinate verranno usate come punto di riferimento.

Descrizione

Metodo; restituisce i valori minimo e massimo delle coordinate *x* e *y* dell'oggetto MovieClip rispetto allo spazio di coordinate specificato nel relativo argomento. L'oggetto restituito conterrà le proprietà {xMin, xMax, yMin, yMax}. Usare i metodi `localToGlobal` e `globalToLocal` dell'oggetto MovieClip per convertire le coordinate locali del clip filmato in coordinate rispetto allo stage o viceversa, rispettivamente.

Lettore

Flash 5 o versione successiva.

Esempio

L'esempio seguente usa il metodo `getBounds` per ottenere il riquadro di limitazione dell'istanza `myMovieClip` rispetto allo spazio di coordinate del filmato principale.

```
myMovieClip.getBounds(_root);
```

Vedere anche

“MovieClip.globalToLocal” a pagina 307

“MovieClip.localToGlobal” a pagina 311

MovieClip.getBytesLoaded

Sintassi

```
ClipFilmato.getBytesLoaded();
```

Argomenti

Nessuno.

Descrizione

Metodo; restituisce il numero di byte caricati (di cui si è effettuato lo streaming) per l'oggetto `MovieClip` specificato. Poiché i clip filmato interni vengono caricati automaticamente, il valore restituito da questo metodo coincide con quello restituito dal metodo `MovieClip.getBytesTotal` se l'oggetto `MovieClip` specificato fa riferimento a un clip filmato interno. Questo metodo è indirizzato all'uso con filmati caricati. Il confronto tra il valore restituito da `getBytesLoaded` e il valore restituito da `getBytesTotal` fornisce la percentuale di un filmato esterno caricata.

Lettore

Flash 5 o versione successiva.

MovieClip.getBytesTotal

Sintassi

```
ClipFilmato.getBytesTotal();
```

Argomenti

Nessuno.

Descrizione

Metodo; restituisce la dimensione, in byte, dell'oggetto `MovieClip` specificato. Nel caso di clip filmato esterni (il filmato principale o un clip filmato che è stato caricato in un percorso target o un livello), il valore restituito corrisponde alla dimensione del file SWF.

Lettore

Flash 5 o versione successiva.

MovieClip.getURL

Sintassi

```
ClipFilmato.getURL(URL [,finestra, variabili]);
```

Argomenti

URL L'URL presso cui è reperibile il documento.

finestra Argomento opzionale che specifica il nome, il frame o l'espressione che indica la finestra o il frame HTML in cui caricare il documento. È inoltre possibile usare uno dei seguenti nomi riservati: `_self` indica il frame corrente nella finestra corrente, `_blank` indica una finestra nuova, `_parent` indica il frame principale rispetto a quello corrente, `_top` indica il frame di primo livello nella finestra corrente.

variabili Argomento opzionale che specifica un metodo per l'invio delle variabili associate al filmato da caricare. Omettere questo argomento in assenza di variabili; in caso contrario, specificare se caricare le variabili tramite il metodo GET o POST . GET accoda le variabili in fondo all'URL e si usa in caso di un numero limitato di variabili. POST invia le variabili in un'intestazione HTTP separata e si usa in caso di lunghe stringhe di variabili.

Descrizione

Metodo; carica un documento dall'URL specificato nella finestra indicata. Il metodo `getUrl` consente inoltre di passare variabili a un'altra applicazione definita presso l'URL tramite il metodo GET o POST.

Letttore

Flash 5 o versione successiva.

MovieClip.globalToLocal

Sintassi

```
ClipFilmato.globalToLocal(punto);
```

Argomenti

punto Il nome o l'identificatore di un oggetto creato tramite l'oggetto generico Object che specifica le coordinate *x* e *y* come proprietà.

Descrizione

Metodo; converte le coordinate dell'oggetto *punto* da coordinate rispetto allo stage (globali) a coordinate rispetto al clip filmato (locali).

Letttore

Flash 5 o versione successiva.

Esempio

L'esempio seguente converte le coordinate globali *x* e *y* dell'oggetto *point* in coordinate locali del clip filmato.

```
onClipEvent(mouseMove) {  
    point = new object();  
    point.x = _root._xmouse;  
    point.y = _root._ymouse;  
    globalToLocal(point);  
    _root.out = _xmouse + " === " + _ymouse;  
    _root.out2 = point.x + " === " + point.y;  
    updateAfterEvent();  
}
```

Vedere anche

“MovieClip.localToGlobal” a pagina 311

“MovieClip.getBounds” a pagina 305

MovieClip.gotoAndPlay

Sintassi

```
ClipFilmato.gotoAndPlay(fotogramma);
```

Argomenti

fotogramma Il numero di fotogramma a cui viene inviato l'indicatore di riproduzione.

Descrizione

Metodo; avvia la riproduzione del filmato a partire dal fotogramma specificato.

Lettore

Flash 5 o versione successiva.

MovieClip.gotoAndStop

Sintassi

```
ClipFilmato.gotoAndStop(fotogramma);
```

Argomenti

fotogramma Il numero di fotogramma a cui viene inviato l'indicatore di riproduzione.

Descrizione

Metodo; interrompe la riproduzione del filmato in corrispondenza del fotogramma specificato.

Lettore

Flash 5 o versione successiva.

MovieClip.hitTest

Sintassi

```
ClipFilmato.hitTest(x, y, indicatoreForma);
```

```
ClipFilmato.hitTest(target);
```

Argomenti

x La coordinata *x* dell'area di collisione sullo stage.

y La coordinata *y* dell'area di collisione sullo stage.

Le coordinate *x* e *y* sono definite nello spazio di coordinate globali.

target Il percorso target dell'area di collisione che potrebbe intersecare o sovrapporsi all'istanza specificata in *ClipFilmato*. Il *target* rappresenta di solito un pulsante o un campo per l'immissione di testo.

indicatoreForma Valore booleano che specifica se valutare per l'intera forma dell'istanza specificata (*true*) oppure solo il riquadro di limitazione (*false*). Questo argomento può essere specificato solo se l'area di collisione è identificata dalle coordinate *x* e *y*.

Descrizione

Metodo; valuta l'istanza specificata in *ClipFilmato* per verificare se si sovrappone o interseca con l'area di collisione identificata dall'argomento *target* o dalle coordinate *x* e *y*.

Nel primo caso confronta le coordinate *x* e *y* e la forma o il riquadro di limitazione dell'istanza specificata, a seconda dell'impostazione di *indicatoreForma*. Se *indicatoreForma* è impostato su *true*, viene valutata solo l'area correntemente occupata dall'istanza sullo stage, e se le coordinate *x* e *y* si sovrappongono in un punto qualsiasi, viene restituito il valore *true*. Ciò consente di determinare se il clip filmato si trova all'interno di un'area di collisione, o attiva, specificata.

Nel secondo caso valuta i riquadri di limitazione dell'istanza specificata e del *target*, quindi restituisce *true* se si sovrappongono o intersecano in un punto qualsiasi.

Letture

Flash 5 o versione successiva.

Esempio

L'esempio seguente usa il metodo *hitTest* con le proprietà *x_mouse* e *y_mouse* per determinare se il mouse si trova sul riquadro di limitazione del *target*.

```
if (hitTest( _root._xmouse, _root._ymouse, false));
```

L'esempio seguente usa il metodo *hitTest* per determinare se il clip filmato *ball* interseca o si sovrappone al clip filmato *square*.

```
if(_root.ball, hittest(_root.square)){  
trace("ball intersects square");  
}
```

Vedere anche

"*MovieClip.localToGlobal*" a pagina 311

"*MovieClip.globalToLocal*" a pagina 307

"*MovieClip.getBounds*" a pagina 305

MovieClip.loadMovie

Sintassi

```
ClipFilmato.loadMovie(url [,variabili]);
```

Argomenti

Argomenti

url URL assoluto o relativo per il file SWF da caricare. Il percorso relativo deve essere relativo al file SWF. Tale URL deve trovarsi nello stesso sottodominio dell'URL che contiene il filmato. Per essere usati in Flash Player o verificati in modalità di prova filmato in ambiente di creazione Flash, tutti i file SWF devono essere memorizzati nella stessa cartella e i nomi dei file non possono includere specifiche della cartella o dell'unità disco.

variabili Argomento opzionale che specifica un metodo per l'invio delle variabili associate al filmato da caricare. L'argomento deve essere costituito dalla stringa "GET" o "POST." Omettere questo argomento in assenza di variabili; in caso contrario, specificare se caricare le variabili tramite il metodo GET o POST . GET accoda le variabili in fondo all'URL e si usa in caso di un numero limitato di variabili. POST invia le variabili in un'intestazione HTTP separata e si usa in caso di lunghe stringhe di variabili.

Descrizione

Metodo; riproduce filmati aggiuntivi senza chiudere Flash Player. In genere Flash Player visualizza un solo filmato (file SWF), quindi si chiude. Il metodo `loadMovie` consente di visualizzare più filmati contemporaneamente o di passare da un filmato all'altro senza dover caricare un altro documento HTML.

Usare l'azione `unloadMovie` per rimuovere i filmati caricati con l'azione `loadMovie`.

Usare il metodo `loadVariables` per mantenere il filmato attivo e aggiornare le variabili con i nuovi valori.

Lettore

Flash 5 o versione successiva.

Vedere anche

"`MovieClip.loadVariables`" a pagina 310

"`MovieClip.unloadMovie`" a pagina 315

MovieClip.loadVariables

Sintassi

```
ClipFilmato.loadVariables(url, variabili);
```

Argomenti

url L'URL assoluto o relativo per il file esterno. L'host dell'URL deve trovarsi nello stesso sottodominio del clip filmato.

variabili Il metodo di recupero delle variabili. GET accoda le variabili in fondo all'URL e si usa in caso di un numero limitato di variabili. POST invia le variabili in un'intestazione HTTP separata e si usa in caso di lunghe stringhe di variabili.

Descrizione

Metodo; legge i dati da un file esterno e imposta i valori delle variabili in un filmato o clip filmato. Il file esterno può essere un file di testo creato tramite script CGI, Active Server Pages (ASP) o Personal Home Page (PHP) e può contenere un numero illimitato di variabili.

Questo metodo consente anche di aggiornare le variabili nel filmato attivo assegnando ad esse nuovi valori.

È necessario che il testo presso l'URL sia in formato MIME standard: *application/x-www-urlformencoded* (un formato standard usato dagli script CGI).

Lettore

Flash 5 o versione successiva.

Vedere anche

"MovieClip.loadMovie" a pagina 310

MovieClip.localToGlobal

Sintassi

```
ClipFilmato.localToGlobal(punto);
```

Argomenti

punto Il nome o l'identificatore di un oggetto creato tramite l'oggetto Object che specifica le coordinate *x* e *y* come proprietà.

Descrizione

Metodo; converte le coordinate dell'oggetto *punto* da coordinate rispetto al clip filmato (locali) in coordinate rispetto allo stage (globali).

Lettore

Flash 5 o versione successiva.

Esempio

L'esempio seguente converte le coordinate *x* e *y* dell'oggetto *punto* da coordinate del clip filmato (locali) in coordinate dello stage (globali). Le coordinate locali *x* e *y* vengono specificate tramite le proprietà *xmouse* e *ymouse* in modo da recuperare le coordinate *x* e *y* della posizione del mouse.

```
onClipEvent(mouseMove) {  
    point = new object();  
    point.x = _xmouse;  
    point.y = _ymouse;  
    _root.out3 = point.x + " === " + point.y;  
}
```

```
_root.out = _root._xmouse + " == " + _root._ymouse;  
localToGlobal(point);  
_root.out2 = point.x + " == " + point.y;  
updateAfterEvent();  
}
```

Vedere anche

“MovieClip.globalToLocal” a pagina 307

MovieClip.nextFrame

Sintassi

```
ClipFilmato.nextFrame();
```

Argomenti

Nessuno.

Descrizione

Metodo; invia l'indicatore di riproduzione al fotogramma successivo del clip filmato.

Lettore

Flash 5 o versione successiva.

MovieClip.play

Sintassi

```
ClipFilmato.play();
```

Argomenti

Nessuno.

Descrizione

Metodo; riproduce il clip filmato.

Lettore

Flash 5 o versione successiva.

MovieClip.prevFrame

Sintassi

```
ClipFilmato.prevFrame();
```

Argomenti

Nessuno.

Descrizione

Metodo; invia l'indicatore di riproduzione al fotogramma precedente e lo blocca in tale posizione.

Lettore

Flash 5 o versione successiva.

MovieClip.removeMovieClip

Sintassi

```
ClipFilmato.removeMovieClip();
```

Argomenti

Nessuno.

Descrizione

Metodo; rimuove un'istanza di clip filmato creata con l'azione `duplicateMovieclip` oppure con i metodi `duplicateMovieClip` o `attachMovie` dell'oggetto `MovieClip`.

Lettore

Flash 5 o versione successiva.

Vedere anche

“`MovieClip.loadMovie`” a pagina 310

“`MovieClip.attachMovie`” a pagina 304

MovieClip.startDrag

Sintassi

```
ClipFilmato.startDrag([bloccato, sinistra, destra, superiore, inferiore]);
```

Argomenti

bloccato Valore booleano che specifica se il clip filmato mobile è bloccato al centro rispetto alla posizione del mouse (*true*) oppure in corrispondenza del punto in cui l'utente ha fatto clic inizialmente nel clip filmato (*false*). È un argomento opzionale.

sinistra, superiore, destra, inferiore Valori relativi alle coordinate del filmato principale del clip filmato che definiscono un rettangolo di delimitazione per quest'ultimo. Si tratta di argomenti opzionali.

Descrizione

Metodo; consente il trascinamento da parte dell'utente del clip filmato specificato. Il filmato è mobile fino a quando viene interrotto esplicitamente tramite la chiamata del metodo `stopDrag` o fino a quando viene reso mobile un altro clip filmato. È possibile trascinare un solo clip filmato alla volta.

Lettore

Flash 5 o versione successiva.

Vedere anche

“`MovieClip.stopDrag`” a pagina 314

“`_droptarget`” a pagina 253

MovieClip.stop

Sintassi

```
ClipFilmato.stop();
```

Argomenti

Nessuno.

Descrizione

Metodo; interrompe il clip filmato attualmente in riproduzione.

Lettore

Flash 5 o versione successiva.

MovieClip.stopDrag

Sintassi

```
ClipFilmato.stopDrag();
```

Argomenti

Nessuno.

Descrizione

Metodo; interrompe l'azione di trascinamento implementata con il metodo `startDrag`. Un filmato è mobile fino alla chiamata del metodo `stopDrag` o fino a quando diventa mobile un altro filmato. È possibile trascinare un solo clip filmato alla volta.

Lettore

Flash 5 o versione successiva.

Vedere anche

“`_droptarget`” a pagina 253

“`MovieClip.startDrag`” a pagina 313

MovieClip.swapDepths

Sintassi

```
ClipFilmato.swapDepths(profondità);
```

```
ClipFilmato.swapDepths(target);
```

Argomenti

target L'istanza di clip filmato la cui profondità viene sostituita con quella dell'istanza specificata in *ClipFilmato*. Le due istanze devono avere lo stesso clip filmato principale.

profondità Un numero che specifica il livello di profondità in cui posizionare il filmato *ClipFilmato*.

Descrizione

Metodo; scambia la posizione di impilamento o *z* (livello di profondità) dell'istanza specificata con quella del filmato specificato nell'argomento *target* o con il filmato che occupa correntemente il livello *profondità* specificato nel relativo argomento. I due filmati devono avere lo stesso clip filmato principale. Lo scambio del livello di profondità tra i clip filmato implica lo spostamento di un filmato in primo o in secondo piano rispetto all'altro. Eventuali interpolazioni del filmato in corso quando viene eseguita la chiamata di questo metodo verranno interrotte.

Lettore

Flash 5 o versione successiva.

Vedere anche

“_level” a pagina 282

MovieClip.unloadMovie

Sintassi

```
ClipFilmato.unloadMovie();
```

Argomenti

Nessuno.

Descrizione

Metodo; rimuove un clip filmato caricato con i metodi `loadMovie` o `attachMovie` dell'oggetto `MovieClip`.

Lettore

Flash 5 o versione successiva.

Vedere anche

“`MovieClip.loadMovie`” a pagina 310

“`MovieClip.attachMovie`” a pagina 304

_name

Sintassi

```
nomeistanza._name  
nomeistanza._name = valore;
```

Argomenti

nomeistanza Il nome di un'istanza di un clip filmato la cui proprietà *_name* deve essere impostata o richiamata.

valore Una stringa che specifica un nuovo nome dell'istanza.

Descrizione

Proprietà; specifica il nome dell'istanza del clip filmato.

Lettore

Flash 4 o versione successiva.

NaN

Sintassi

```
NaN
```

Argomenti

Nessuno.

Descrizione

Variabile; una variabile predefinita con il valore IEEE-754 per NaN (non un numero).

Lettore

Flash 5 o versione successiva.

ne (non uguale; specifico per stringhe)

Sintassi

```
espressione1 ne espressione2
```

Argomenti

espressione1, *espressione2* Numeri, stringhe o variabili.

Descrizione

Operatore (confronto); confronta *espressione1* ed *espressione2*, quindi restituisce *true* se *espressione1* è diversa da *espressione2*; altrimenti restituisce *false*.

Lettore

Flash 4 o versione successiva. Questo operatore è diventato obsoleto in Flash 5 ed è consigliato l'uso del nuovo operatore `!=` (non uguale).

Vedere anche

`"!=(disuguaglianza)"` a pagina 185

new

Sintassi

```
new funzionecostruzione();
```

Argomenti

funzionecostruzione Una funzione seguita da un qualsiasi argomento opzionale tra parentesi. La funzione corrisponde di solito al nome del tipo di oggetto (ad esempio Array, Math, Number, Object) da creare.

Descrizione

Operatore; crea un nuovo oggetto, in un primo tempo anonimo, chiama la funzione identificata dall'argomento *funzionecostruzione*, passa qualsiasi argomento opzionale tra parentesi, quindi passa il nuovo oggetto creato come valore della parola chiave `this`. La funzione di costruzione può in seguito usare `this` per creare un'istanza del nuovo oggetto.

La proprietà `_prototype_` dell'oggetto della funzione di costruzione viene copiata nella proprietà `_proto_` del nuovo oggetto. Di conseguenza, il nuovo oggetto supporta tutti i metodi e le proprietà specificate nell'oggetto prototipo della funzione di costruzione.

Lettore

Flash 5 o versione successiva.

Esempio

L'esempio seguente crea gli oggetti `book1` e `book2` mediante l'uso del nuovo operatore.

```
function Book(name, price)
{
    this.name = name;
    this.price = price;
}
book1 = new Book("Confederacy of Dunces", 19.95);
book2 = new Book("The Floating Opera", 10.95);
```

Vedere anche

`"[] (operatore di accesso matrice)"` a pagina 196

`"{} (operatore di inizializzazione degli oggetti)"` a pagina 198

La sezione relativa al metodo di costruzione all'interno di una voce oggetto.

newline

Sintassi

`newline;`

Argomenti

Nessuno.

Descrizione

Costante; inserisce un carattere di ritorno a capo (`{ }`) aggiungendo una riga vuota nel codice ActionScript. La costante `newline` consente di creare spazio per le informazioni recuperate da una funzione o da un'azione nel codice.

Lettore

Flash 4 o versione successiva.

nextFrame

Sintassi

`nextFrame();`

Argomenti

Nessuno.

Descrizione

Azione; invia l'indicatore di riproduzione al successivo fotogramma e lo blocca in tale posizione.

Lettore

Flash 2 o versione successiva.

Esempio

Quando l'utente sceglie un pulsante al quale è assegnata un'azione `nextFrame`, l'indicatore di riproduzione viene inviato al successivo fotogramma.

```
on (release) {  
    nextFrame(5);  
}
```

nextScene

Sintassi

`nextScene();`

Argomenti

Nessuno.

Descrizione

Azione; invia l'indicatore di riproduzione al fotogramma 1 della successiva scena e lo blocca in tale posizione.

Lettore

Flash 2 o versione successiva.

Esempio

Questa azione è assegnata a un pulsante che, se premuto e rilasciato, invia l'indicatore di riproduzione al fotogramma 1 della successiva scena.

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

not

Sintassi

not espressione

Argomenti

espressione Qualsiasi variabile o altra espressione che è convertibile in un valore booleano.

Descrizione

Operatore; esegue un'operazione di NOT logico in ambiente Flash 4 Player.

Lettore

Flash 4 o versione successiva. Questo operatore è diventato obsoleto in Flash 5 ed è consigliato l'uso del nuovo operatore ! (NOT logico).

Vedere anche

“! (NOT logico)” a pagina 184

null

Sintassi

null

Argomenti

Nessuno.

Descrizione

Parola chiave; un valore speciale che può essere assegnato a variabili o restituito da una funzione se non è stato fornito alcun dato. Il valore *null* consente di rappresentare valori mancanti o valori senza un tipo di dati definito.

Lettore

Flash 5 o versione successiva.

Esempio

In un contesto numerico, `null` viene valutato come 0. I test di uguaglianza possono essere effettuati con `null`. In questa istruzione, il nodo di un albero binario non possiede nodi secondari a sinistra; il campo per l'elemento a sinistra può quindi essere impostato su `null`.

```
if (tree.left == null) {  
    tree.left = new TreeNode();  
}
```

Number (funzione)

Sintassi

`Number(espressione);`

Argomenti

espressione Stringa, valore booleano o altra espressione da convertire in un numero.

Descrizione

Funzione; converte l'argomento *x* in un numero e restituisce un valore nel modo seguente:

Se *x* è un numero, il valore restituito è *x*.

Se *x* è un valore booleano, il valore restituito è 1 se *x* è `true` e 0 se *x* è `false`.

Se *x* è una stringa, la funzione tenta di analizzare *x* come se fosse un numero decimale seguito da un valore esponente opzionale, ad esempio `1,57505e-3`.

Se *x* non è definito, il valore restituito è 0.

Questa funzione viene usata per convertire i file Flash 4 contenenti operatori obsoleti quando vengono importati in ambienti di creazione di Flash 5. Per ulteriori informazioni, vedere l'operatore `&`.

Lettore

Flash 4 o versione successiva.

Vedere anche

"Number (oggetto)" a pagina 320

Number (oggetto)

L'oggetto `Number` è un semplice oggetto wrapper per il tipo di dati numerico, ossia consente di manipolare i valori numerici di base con i metodi e le proprietà associati all'oggetto `Number`. La funzionalità fornita da questo oggetto è identica a quella dell'oggetto `Number` di JavaScript.

La funzione di costruzione `Number` deve essere usata per chiamare i metodi dell'oggetto `Number`; il suo uso, però, non è richiesto per chiamare le proprietà dell'oggetto `Number`. Gli esempi seguenti descrivono la sintassi per chiamare i metodi e le proprietà dell'oggetto `Number`:

Nell'esempio seguente viene chiamato il metodo `toString` dell'oggetto `Number`:

```
myNumber = new Number(1234);  
myNumber.toString();
```

Restituisce una stringa contenente la rappresentazione binaria del numero 1234.

Nell'esempio seguente viene chiamata la proprietà `MIN_VALUE` (detta anche costante) dell'oggetto `Number`:

```
smallest = Number.MIN_VALUE
```

Riepilogo dei metodi validi per l'oggetto `Number`

Metodo	Descrizione
<code>toString</code>	Restituisce una rappresentazione in formato stringa di un oggetto <code>Number</code> .
<code>valueOf</code>	Restituisce il valore di base di un oggetto <code>Number</code> .

Riepilogo delle proprietà valide per l'oggetto `Number`

Proprietà	Descrizione
<code>MAX_VALUE</code>	Costante equivalente al numero rappresentabile più elevato (doppia precisione IEEE-754). Questo numero è circa 1,7976931348623158e+308.
<code>MIN_VALUE</code>	Costante equivalente al numero rappresentabile più basso (doppia precisione IEEE-754). Questo numero è circa 5e-324.
<code>NaN</code>	Costante equivalente al valore per NaN (Not a Number).
<code>NEGATIVE_INFINITY</code>	Costante equivalente al valore infinito negativo.
<code>POSITIVE_INFINITY</code>	Costante equivalente al valore infinito positivo. Questo valore è lo stesso di quello della variabile globale <code>Infinity</code> .

Funzione di costruzione per l'oggetto Number

Sintassi

```
myNumber = new Number(valore);
```

Argomenti

valore Il valore numerico dell'oggetto Number che viene creato o un valore da convertire in un numero.

Descrizione

Funzione di costruzione; crea un nuovo oggetto Number. È necessario usare la funzione di costruzione Number quando vengono usati i metodi `toString` e `valueOf` dell'oggetto Number. Non usare una funzione di costruzione quando si usano le proprietà dell'oggetto Number. La funzione di costruzione `new Number` è principalmente usata come segnaposto. Un'istanza dell'oggetto Number non è uguale alla funzione Number che converte un argomento in un valore di base.

Lettore

Flash 5 o versione successiva.

Esempio

Nel codice seguente vengono costruiti nuovi oggetti Number.

```
n1 = new Number(3.4);  
n2 = new Number(-10);
```

Vedere anche

“Number (funzione)” a pagina 320

Number.MAX_VALUE

Sintassi

```
Number.MAX_VALUE
```

Argomenti

Nessuno.

Descrizione

Proprietà; il numero rappresentabile più elevato (doppia precisione IEEE-754). Questo numero è circa 1,79E+308.

Lettore

Flash 5 o versione successiva.

Number.MIN_VALUE

Sintassi

`Number.MIN_VALUE`

Argomenti

Nessuno.

Descrizione

Proprietà; il numero rappresentabile più basso (doppia precisione IEEE-754). Questo numero è circa $5e-324$.

Lettore

Flash 5 o versione successiva.

Number.NaN

Sintassi

`Number.NaN`

Argomenti

Nessuno.

Descrizione

Proprietà; il valore IEEE-754 che rappresenta NaN (Not A Number).

Lettore

Flash 5 o versione successiva.

Number.NEGATIVE_INFINITY

Sintassi

`Number.NEGATIVE_INFINITY`

Argomenti

Nessuno.

Descrizione

Proprietà; restituisce il valore IEEE-754 che rappresenta l'infinito negativo. Questo valore è lo stesso della variabile globale `Infinity`.

L'infinito negativo è un valore numerico speciale restituito quando un'operazione matematica o una funzione restituisce un valore negativo maggiore di quello rappresentabile.

Lettore

Flash 5 o versione successiva.

Number.POSITIVE_INFINITY

Sintassi

`Number.POSITIVE_INFINITY`

Argomenti

Nessuno.

Descrizione

Proprietà; restituisce il valore IEEE-754 che rappresenta l'infinito positivo. Questo valore è lo stesso della variabile globale `Infinity`.

L'infinito positivo è un valore numerico speciale restituito quando un'operazione matematica o una funzione restituisce un valore maggiore di quello rappresentabile.

Lettore

Flash 5 o versione successiva.

Number.toString

Sintassi

`Numero.toString(radice);`

Argomenti

radice Specifica la base numerica (da 2 a 36) da usare per la conversione da numero a stringa. Se non si specifica l'argomento *radice*, il valore predefinito è 10.

Descrizione

Metodo; restituisce una rappresentazione in formato stringa dell'oggetto `Number` specificato (*Numero*).

Lettore

Flash 5 o versione successiva.

Esempio

L'esempio seguente usa il metodo `Number.toString` con il valore dell'argomento *radice* pari a 2:

```
myNumber = new Number (1000);  
(1000).toString(2);
```

Restituisce una stringa contenente la rappresentazione binaria del numero 1000.

Number.valueOf

Sintassi

`Numero.valueOf();`

Argomenti

Nessuno.

Descrizione

Metodo; restituisce il tipo di valore di base dell'oggetto Number specificato e converte l'oggetto wrapper di Number in tale tipo di valore di base.

Lettore

Flash 5 o versione successiva.

Object (oggetto)

L'oggetto generico Object si trova alla radice della gerarchia di classi ActionScript. La funzionalità dell'oggetto Object generico rappresenta una piccola parte rispetto a quella fornita dall'oggetto Object di JavaScript.

L'oggetto Object generico richiede Flash 5 Player.

Riepilogo dei metodi per l'oggetto Object

Metodo	Descrizione
<code>toString</code>	Converte l'oggetto specificato in una stringa e la restituisce.
<code>valueOf</code>	Restituisce il valore di base di un oggetto Object.

Funzione di costruzione per l'oggetto Object

Sintassi

```
new Object();
```

```
new Object(valore);
```

Argomenti

valore Numero, valore booleano o stringa da convertire in un oggetto. È un argomento opzionale. Se *valore* non viene specificato, la funzione di costruzione crea un nuovo oggetto senza proprietà definite.

Descrizione

Funzione di costruzione; crea un nuovo oggetto Object.

Lettore

Flash 5 o versione successiva.

Vedere anche

“Sound.setTransform” a pagina 350

“Color.setTransform” a pagina 231

Object.toString

Sintassi

nomeOggetto.toString();

Argomenti

Nessuno.

Descrizione

Metodo; converte l'oggetto specificato in una stringa e la restituisce.

Lettore

Flash 5 o versione successiva.

Object.valueOf

Sintassi

nomeOggetto.valueOf();

Argomenti

Nessuno.

Descrizione

Metodo; restituisce il valore di base dell'oggetto specificato. Se l'oggetto non possiede alcun valore di base, viene restituito l'oggetto stesso.

Lettore

Flash 5 o versione successiva.

onClipEvent

Sintassi

```
onClipEvent(eventoFilmato){  
  ...  
}
```

Argomenti

L'elemento *eventoFilmato* è un evento di attivazione che esegue azioni assegnate a un'istanza di clip filmato. Per l'argomento *eventoFilmato* è possibile specificare qualsiasi dei valori seguenti.

- **load** L'azione viene attivata appena il clip filmato diventa un'istanza e appare nella linea temporale.
- **unload** L'azione viene attivata nel primo fotogramma dopo la rimozione del clip filmato dalla linea temporale. Le azioni associate all'evento clip filmato **Unload** vengono eseguite prima di qualsiasi azione associata al fotogramma interessato.

- **enterFrame** L'azione viene attivata con la riproduzione di ogni fotogramma, come avviene nelle azioni associate a un clip filmato. Le azioni associate all'evento di clip filmato `OnEnterFrame` vengono eseguite dopo qualsiasi azione associata ai fotogrammi interessati.
- **mouseMove** L'azione viene attivata a ogni spostamento del mouse. L'uso delle proprietà `_xmouse` e `_ymouse` consente di determinare la posizione corrente del mouse.
- **mouseDown** L'azione viene attivata quando il pulsante sinistro del mouse viene premuto.
- **mouseUp** L'azione viene attivata quando il pulsante sinistro del mouse viene rilasciato.
- **keyDown** L'azione viene attivata quando viene premuto un tasto. L'uso del metodo `Key.getCode` consente di recuperare informazioni sull'ultimo tasto premuto.
- **keyUp** L'azione viene attivata quando viene rilasciato un tasto. L'uso del metodo `Key.getCode` consente di recuperare informazioni sull'ultimo tasto premuto.
- **data** L'azione viene attivata al ricevimento di dati in un'azione `loadVariables` o `loadMovie`. Se specificato con un'azione `loadVariables`, l'evento `data` si verifica una sola volta, al caricamento dell'ultima variabile. Se specificato con un'azione `loadMovie`, l'evento `data` si verifica ripetutamente, al ricevimento di ogni sezione di dati.

Descrizione

Gestore; innesca le azioni definite per un'istanza specifica di un clip filmato.

Letture

Flash 5 o versione successiva.

Esempio

L'istruzione seguente include lo script da un file esterno quando l'istanza di clip filmato viene caricata e appare per la prima volta nella linea temporale.

```
onClipEvent(load) {
    #include "myScript.as"
}
```

L'esempio seguente usa `onClipEvent` con l'evento filmato `keyDown`. L'evento filmato `keyDown` viene solitamente usato insieme a uno o più metodi e proprietà associati all'oggetto `Key`. Nello script seguente, `key.getCode` viene usato per determinare il tasto premuto dall'utente; il valore restituito viene associato alle proprietà `RIGHT` o `LEFT` dell'oggetto `Key` e il filmato indirizzato di conseguenza.

```
onClipEvent(keyDown) {
    if (Key.getCode() == Key.RIGHT) {
        _parent.nextFrame();
    } else if (Key.getCode() == Key.LEFT){
```

```

        _parent.prevFrame();
    }

```

L'esempio seguente usa `onClipEvent` con l'evento filmato `mouseMove`. Le proprietà `xmouse` e `ymouse` segnalano la posizione del mouse.

```

onClipEvent(mouseMove) {
    stageX=_root.xmouse;
    stageY=_root.ymouse;
}

```

Vedere anche

“on(mouseEvent)” a pagina 328

“Key (oggetto)” a pagina 272

“_xmouse” a pagina 404

“_ymouse” a pagina 406

on(mouseEvent)

Sintassi

```

on(eventoMouse) {
    istruzione;
}

```

Argomenti

istruzione Le istruzioni da eseguire quando si verifica `eventoMouse`.

Gli argomenti validi per un'azione *mouseEvent* sono:

- `press` Il pulsante del mouse viene premuto mentre il puntatore si trova sopra il pulsante.
- `release` Il pulsante del mouse viene rilasciato mentre il puntatore si trova sopra il pulsante.
- `releaseOutside` Il pulsante del mouse viene rilasciato mentre il puntatore si trova al di fuori del pulsante.
- `rollOver` Il puntatore del mouse scorre sopra il pulsante.
- `rollOut` Il puntatore del mouse viene spostato fuori dall'area del pulsante.
- `dragOver` Mentre il puntatore si trova sopra il pulsante, il pulsante del mouse viene premuto e contemporaneamente il puntatore viene spostato fuori dall'area del pulsante, quindi riportato di nuovo sopra quest'ultimo.
- `dragOut` Mentre il puntatore si trova sopra il pulsante, il pulsante del mouse viene premuto, quindi il puntatore viene trascinato fuori dall'area del pulsante.

- `keyPress ("tasto")` Il *tasto* specificato viene premuto. L'argomento *tasto* viene specificato tramite uno dei codici tasto elencati nell'Appendice B, "Tasti della tastiera e valori dei codici tasto", o tramite una delle costanti tasto elencate in "Riepilogo delle proprietà valide per l'oggetto Key" a pagina 273.

Descrizione

Gestore; specifica l'evento mouse o la pressione di tasto che innesca un'azione.

Letture

Flash 2 o versione successiva.

Esempio

Nello script seguente, l'azione `startDrag` viene eseguita quando il mouse viene premuto e lo script condizionale viene eseguito quando viene rilasciato il pulsante del mouse e di conseguenza l'oggetto:

```
on(press) {
    startDrag("rabbit");
}
on(release) {
    if(getproperty("", _droptarget) == target) {
        setProperty ("rabbit", _x, _root.rabbit_x);
        setProperty ("rabbit", _y, _root.rabbit_y);
    } else {
        _root.rabbit_x = getProperty("rabbit", _x);
        _root.rabbit_y = getProperty("rabbit", _y);
        _root.target = "pasture";
    }
    trace(_root.rabbit_y);
    trace(_root.rabbit_x);
    stopDrag();
}
```

Vedere anche

"Key (oggetto)" a pagina 272

"onClipEvent" a pagina 326

or

Sintassi

condizione1 or condizione2

Argomenti

condizione1, condizione2 Espressione che restituisce true o false.

Descrizione

Operatore; valuta *condizione1* e *condizione2*; se una delle due espressioni risulta true, l'intera espressione viene valutata come true.

Letttore

Flash 4 o versione successiva. Questo operatore è diventato obsoleto in Flash 5 ed è consigliato l'uso del nuovo operatore `||`.

Vedere anche

`"|| (OR)"` a pagina 200

ord

Sintassi

```
ord(carattere);
```

Argomenti

carattere Il carattere da convertire in un numero di codice ASCII.

Descrizione

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

Letttore

Flash 4 o versione successiva. Questa funzione è diventata obsoleta in Flash 5 ed è consigliato l'uso dei metodi e delle proprietà dell'oggetto `String`.

Vedere anche

`"String (oggetto)"` a pagina 358

_parent

Sintassi

```
_parent.proprietà = x  
_parent._parent.proprietà = x
```

Argomenti

proprietà La proprietà che si specifica per il clip filmato corrente e principale.

x Il valore impostato per la proprietà. Questo argomento è opzionale e potrebbe non essere richiesto a seconda della proprietà.

Descrizione

Proprietà; specifica o restituisce un riferimento al clip filmato contenente il clip filmato corrente. Il clip filmato corrente è il clip filmato contenente lo script in corso di esecuzione. L'uso di `_parent` consente di specificare un percorso relativo.

Letttore

Flash 4 o versione successiva.

Esempio

Nell'esempio seguente, il clip filmato `desk` è l'elemento secondario del clip filmato `classroom`. Quando lo script seguente viene eseguito all'interno del clip filmato `desk`, l'indicatore di riproduzione passa al fotogramma 10 nella linea temporale del clip filmato `classroom`.

```
_parent.gotoAndStop(10);
```

Vedere anche

“_root” a pagina 339

“targetPath” a pagina 366

parseFloat

Sintassi

```
parseFloat(stringa);
```

Argomenti

stringa La stringa da analizzare sintatticamente e convertire in un numero in virgola mobile.

Descrizione

Funzione; converte una stringa in un numero in virgola mobile. La funzione esegue l'analisi sintattica e restituisce i numeri nella stringa finché non raggiunge un carattere che non è parte del numero iniziale. Se la stringa non inizia con un numero che può essere analizzato sintatticamente, `parseFloat` restituisce NaN o 0. Gli spazi vuoti prima di numeri interi validi vengono ignorati, così come i caratteri finali non numerici.

Letture

Flash 5 o versione successiva.

Esempio

Gli esempi seguenti descrivono l'uso di `parseFloat` per valutare diversi tipi di numeri:

```
parseFloat( "-2 " ) restituisce -2
```

```
parseFloat( "2,5 " ) restituisce 2,5
```

```
parseFloat( "3,5e6 " ) restituisce 3,5e6 o 3500000
```

```
parseFloat( "foobar " ) restituisce NaN
```

parseInt

Sintassi

`parseInt(espressione, radice);`

Argomenti

espressione Stringa, numero in virgola mobile o altra espressione da analizzare sintatticamente e convertire in numero intero.

radice Numero intero che rappresenta la radice (base) del numero da analizzare sintatticamente. I valori validi sono compresi tra 2 e 36. È un argomento opzionale.

Descrizione

Funzione; converte una stringa in un numero intero. Se non è possibile convertire la stringa specificata negli argomenti in un numero, la funzione restituisce NaN o 0. Se la stringa inizia con 0 o se si specifica una radice di 8 il numero viene interpretato come numero ottale. I numeri interi che iniziano con 0x vengono interpretati come numeri esadecimali. Gli spazi vuoti prima di numeri interi validi vengono ignorati, così come i caratteri finali non numerici.

Lettore

Flash 5 o versione successiva.

Esempio

Gli esempi seguenti descrivono l'uso di `parseInt` per valutare diversi tipi di numeri.

```
parseInt("3,5") restituisce 3,5
```

```
parseInt("bar") restituisce NaN
```

```
parseInt("4foo") restituisce 4
```

Esempio

Conversione esadecimale:

```
parseInt("0x3F8") restituisce 1016
```

```
parseInt("3E8", 16) restituisce 1000
```

Conversione binaria:

```
parseInt("1010", 2) restituisce 10 (la rappresentazione decimale del numero binario 1010)
```

Analisi sintattica di un numero ottale (in questo caso il numero ottale è identificato dalla radice 8):

```
parseInt("777", 8) restituisce 511 (la rappresentazione decimale del numero ottale 777)
```

play

Sintassi

`play();`

Argomenti

Nessuno.

Descrizione

Azione; sposta l'indicatore di riproduzione in avanti nella linea temporale.

Lettore

Flash 2 o versione successiva.

Esempio

Il codice seguente usa un'istruzione `if` per verificare il valore di un nome immesso dall'utente. Se l'utente immette `Steve`, viene chiamata l'azione `play` e l'indicatore di riproduzione si sposta in avanti lungo la linea temporale. Se l'utente immette un valore diverso da `Steve`, il filmato 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

Sintassi

`prevFrame();`

Argomenti

Nessuno.

Descrizione

Azione; invia l'indicatore di riproduzione al fotogramma precedente e lo blocca in tale posizione.

Lettore

Flash 2 o versione successiva.

Esempio

Quando l'utente sceglie un pulsante al quale è assegnata un'azione `prevFrame`, l'indicatore di riproduzione viene inviato al fotogramma precedente.

```
on(release) {
    prevFrame(5);
}
```

Vedere anche

“MovieClip.prevFrame” a pagina 313

prevScene

Sintassi

```
prevScene();
```

Argomenti

Nessuno.

Descrizione

Azione; invia l'indicatore di riproduzione al fotogramma 1 della scena precedente e lo blocca in tale posizione.

Lettore

Flash 2 o versione successiva.

Vedere anche

“nextScene” a pagina 318

print

Sintassi

```
print (target, "bmovie");  
print (target, "bmax");  
print (target, "bframe");
```

Argomenti

target Il nome dell'istanza di clip filmato da stampare. Per impostazione predefinita vengono stampati tutti i fotogrammi del filmato. Per stampare solo determinati fotogrammi, marcarli per la stampa associando loro un'etichetta di fotogramma #P nell'ambiente di creazione.

bmovie Indica il riquadro di delimitazione di un fotogramma specifico del filmato come area di stampa per tutti i fotogrammi stampabili nel filmato. Associare un'etichetta #b (nell'ambiente di creazione) per indicare il fotogramma il cui riquadro di delimitazione deve essere usato come area di stampa.

bmax Indica come area di stampa un riquadro di delimitazione composito formato da tutti i fotogrammi stampabili. Specificare l'argomento *bmax* quando i fotogrammi stampabili nel filmato sono di dimensioni diverse.

bframe Indica che il riquadro di delimitazione di ogni fotogramma stampabile deve essere usato come area di stampa per quel fotogramma. Ciò modifica l'area di stampa per ogni fotogramma e agisce sulla scala degli oggetti per adattarli all'area di stampa. L'uso di *bframe* consente, nel caso di oggetti di diverse dimensioni in ogni fotogramma, di riempire la pagina stampata con l'oggetto.

Descrizione

Azione; stampa il clip filmato *target* a seconda del modificatore di stampa specificato nell'argomento. Per stampare solo determinati fotogrammi nel filmato *target*, associare un'etichetta di fotogramma *#P* ai fotogrammi da stampare. Sebbene l'azione *print* offra una qualità di stampa più elevata rispetto all'azione *printAsBitmap*, essa non può essere impiegata per stampare filmati che usano trasparenze alfa o effetti colore speciali.

Se non si specifica un argomento per l'area di stampa, quest'ultima viene determinata, per impostazione predefinita, dalla dimensione dello stage del filmato caricato. Il filmato non eredita la dimensione dello stage del filmato principale. È possibile controllare l'area di stampa specificando gli argomenti *bmovie*, *bmax* o *bframe*.

Tutti gli elementi stampabili di un filmato devono essere completamente caricati prima dell'inizio della stampa.

La funzione di stampa di Flash Player supporta stampanti PostScript e non PostScript. Le stampanti non PostScript convertono i vettori in bitmap.

Letture

Flash 5 o versione successiva.

Esempio

L'esempio seguente stampa tutti i fotogrammi stampabili in *myMovie* con l'area di stampa definita dal riquadro di delimitazione del fotogramma a cui è associata l'etichetta di fotogramma *#b*.

```
print("myMovie", "bmovie");
```

L'esempio seguente stampa tutti i fotogrammi stampabili in *myMovie* con l'area di stampa definita dal riquadro di delimitazione di ogni fotogramma.

```
print("myMovie", "bmovie");
```

Vedere anche

"*printAsBitmap*" a pagina 335

printAsBitmap

Sintassi

```
printAsBitmap(target, "bmovie");
```

```
printAsBitmap(target, "bmax");
```

```
printAsBitmap(target, "bframe");
```

Argomenti

target Il nome dell'istanza di clip filmato da stampare. Per impostazione predefinita vengono stampati tutti i fotogrammi del filmato. Per stampare solo determinati fotogrammi, marcarli per la stampa associando loro un'etichetta di fotogramma *#P* nell'ambiente di creazione.

bmovie Indica il riquadro di delimitazione di un fotogramma specifico del filmato come area di stampa per tutti i fotogrammi stampabili nel filmato. Associare un'etichetta #b (nell'ambiente di creazione) per indicare il fotogramma il cui riquadro di delimitazione deve essere usato come area di stampa.

bmax Indica come area di stampa un riquadro di delimitazione composito formato da tutti i fotogrammi stampabili. Specificare l'argomento *bmax* quando i fotogrammi stampabili nel filmato sono di dimensioni diverse.

bframe Indica che il riquadro di delimitazione di ogni fotogramma stampabile deve essere usato come area di stampa per quel fotogramma. Ciò modifica l'area di stampa per ogni fotogramma e agisce sulla scala degli oggetti per adattarli all'area di stampa. L'uso di *bframe* consente, nel caso di oggetti di diverse dimensioni in ogni fotogramma, di riempire la pagina stampata con l'oggetto.

Descrizione

Azione; stampa il clip filmato *target* come bitmap. L'uso di `printAsBitmap` consente di stampare filmati contenenti fotogrammi con oggetti che usano trasparenza o effetti colore. L'azione `printAsBitmap` consente la stampa con la più alta risoluzione disponibile della stampante in modo da mantenere la massima definizione e qualità. Per calcolare la dimensione del file stampabile di un fotogramma da stampare come bitmap, moltiplicare la larghezza in pixel per l'altezza in pixel per la risoluzione della stampante.

Per una migliore qualità, se il filmato non contiene trasparenze alfa o effetti colore, si consiglia di usare l'azione `print`.

Per impostazione predefinita, l'area di stampa è determinata dalla dimensione dello stage del filmato caricato. Il filmato non eredita la dimensione dello stage del filmato principale. È possibile controllare l'area di stampa specificando gli argomenti *bmovie*, *bmax* o *bframe*.

Tutti gli elementi stampabili di un filmato devono essere completamente caricati prima dell'inizio della stampa.

La funzione di stampa di Flash Player supporta stampanti PostScript e non PostScript. Le stampanti non PostScript convertono i vettori in bitmap.

Letture

Flash 5 o versione successiva.

Vedere anche

“`print`” a pagina 334

`_quality`

Sintassi

```
_quality  
_quality = x;
```

Argomenti

`x` Una stringa che specifica uno dei valori seguenti:

LOW Bassa qualità di rendering. I grafici non sono sottoposti ad antialiasing e le bitmap non vengono smussate.

MEDIUM Media qualità di rendering. I grafici sono sottoposti ad antialiasing con una griglia di 2x2, ma le bitmap non vengono smussate. Questo valore è adatto a filmati che non contengono testo.

HIGH Alta qualità di rendering. I grafici sono sottoposti ad antialiasing con una griglia di 4x4 e le bitmap vengono smussate se il filmato è statico. Questa è l'impostazione predefinita per la qualità di rendering usata da Flash.

BEST Ottima qualità di rendering. I grafici sono sottoposti ad antialiasing con una griglia di 4x4 e le bitmap vengono sempre smussate.

Descrizione

Proprietà (globale); imposta o recupera la qualità di rendering usata per un filmato.

Lettore

Flash 5 o versione successiva.

Esempio

L'esempio seguente imposta il rendering per `oldQuality` su **HIGH**.

```
oldQuality = _quality  
_quality = "HIGH";
```

Vedere anche

`"_highquality"` a pagina 268

random

Sintassi

```
random();
```

Argomenti

valore Il numero intero più elevato per il quale `random` restituisce un valore.

Descrizione

Funzione; restituisce un numero intero casuale compreso tra 0 e il numero intero specificato nell'argomento *valore*.

Letture

Flash 4 Questa funzione è diventata obsoleta in Flash 5 ed è consigliato l'uso del nuovo metodo `Math.random`.

Esempio

Nell'esempio seguente `random` restituisce il valore 0, 1, 2, 3 o 4.

```
random(5);
```

Vedere anche

“`Math.random`” a pagina 296

removeMovieClip

Sintassi

```
removeMovieClip(target);
```

Argomenti

target Il percorso `target` di un'istanza di clip filmato creata con `duplicateMovieClip` o il nome dell'istanza di clip filmato creata con i metodi `attachMovie` o `duplicateMovie` dell'oggetto `MovieClip`.

Descrizione

Azione; elimina un'istanza di clip filmato creata con i metodi `attachMovie` o `duplicateMovieClip` dell'oggetto `MovieClip` oppure creata con l'azione `duplicateMovieClip`.

Letture

Flash 4 o versione successiva.

Vedere anche

“`duplicateMovieClip`” a pagina 254

“`MovieClip.duplicateMovieClip`” a pagina 304

“`MovieClip.attachMovie`” a pagina 304

“`MovieClip.removeMovieClip`” a pagina 313

return

Sintassi

```
return[espressione];  
return;
```

Argomenti

espressione Un tipo, stringa, numero, matrice o oggetto da valutare e restituire come valore della funzione. È un argomento opzionale.

Descrizione

Azione; specifica il valore restituito da una funzione. Quando viene eseguita l'azione return, l'elemento *espressione* viene valutato e restituito come valore della funzione. L'azione return interrompe l'esecuzione della funzione. Se l'istruzione return viene usata da sola o se Flash non incontra un'istruzione return durante l'azione ciclica, viene restituito null.

Letto

Flash 5 o versione successiva.

Esempio

L'esempio seguente illustra l'uso di return.

```
function sum(a, b, c){  
    return a + b + c;  
}
```

Vedere anche

“function” a pagina 262

_root

Sintassi

```
_root;  
_root.ClipFilmato;  
_root.azione;
```

Argomenti

ClipFilmato Il nome dell'istanza di un clip filmato.

azione Il valore impostato per la proprietà. Questo argomento è opzionale e potrebbe non essere richiesto a seconda della proprietà.

Descrizione

Proprietà; specifica o restituisce un riferimento alla linea temporale del filmato principale. Se un filmato ha più livelli, la linea temporale del filmato principale si trova sul livello contenente lo script in corso di esecuzione. Ad esempio, se uno script nel livello 1 valuta _root, restituirà il livello 1.

L'uso di `_root` è equivalente all'uso della notazione della barra inclinata “/” per specificare il percorso assoluto all'interno del livello corrente.

Lettore

Flash 4 o versione successiva.

Esempio

L'esempio seguente interrompe la riproduzione della linea temporale del livello contenente lo script in corso di esecuzione.

```
_root1.stop();
```

L'esempio seguente invia la linea temporale del livello corrente al fotogramma 3.

```
_root.gotoAndStop(3);
```

Vedere anche

“_parent” a pagina 330

“targetPath” a pagina 366

`_rotation`

Sintassi

```
nomeistanza._rotation
```

```
nomeistanza._rotation = intero
```

Argomenti

intero Il numero di gradi di rotazione del clip filmato.

nomeistanza Il clip filmato da ruotare.

Descrizione

Proprietà; specifica la rotazione del clip filmato in gradi.

Lettore

Flash 4 o versione successiva.

Esempio

scroll

Sintassi

nome_variabile.scroll = *x*

Argomenti

nome_variabile Il nome di una variabile associata a un campo di testo.

x Il numero della riga superiore visibile nel campo di testo. È possibile specificare questo valore o usare quello predefinito di 1. Il Flash Player aggiorna questo valore man mano che l'utente scorre il campo di testo verso l'alto e verso il basso.

Descrizione

Proprietà; controlla la visualizzazione delle informazioni in un campo di testo associato a una variabile. La proprietà `scroll` definisce il punto in cui inizia la visualizzazione del contenuto nel campo di testo; una volta impostata la proprietà, Flash Player la aggiorna man mano che l'utente scorre il campo di testo. La proprietà `scroll` è utile per indirizzare gli utenti verso un paragrafo specifico incluso in lunghi passaggi o per creare campi di testo scorrevoli. È possibile recuperare il valore questa proprietà e modificarla.

Lettore

Flash 4 o versione successiva.

Vedere anche

“`maxscroll`” a pagina 298

Selection (oggetto)

L'oggetto Selection consente di impostare e controllare il campo di testo modificabile correntemente attivo, ossia quello in cui il cursore dell'utente è correntemente posizionato. Gli indici per l'estensione della selezione iniziano da zero (la prima posizione è 0, la seconda è 1 e così via).

Non esiste un metodo di costruzione per l'oggetto Selection perché può essere attivo un solo campo alla volta.

Riepilogo dei metodi validi per l'oggetto Selection

Metodo	Descrizione
<code>getBeginIndex</code>	Restituisce l'indice all'inizio dell'estensione di selezione. Restituisce -1 in assenza di indice o di campo correntemente selezionato.
<code>getCaretIndex</code>	Restituisce la posizione corrente del cursore lampeggiante nell'estensione di selezione correntemente attiva. Restituisce -1 in assenza del cursore lampeggiante o di estensione di selezione correntemente attiva.
<code>getEndIndex</code>	Restituisce l'indice alla fine dell'estensione di selezione. Restituisce -1 in assenza di indice o di campo correntemente selezionato.
<code>getFocus</code>	Restituisce il nome della variabile per il campo di testo modificabile correntemente attivo. Restituisce null in assenza di campo di testo modificabile correntemente attivo.
<code>setFocus</code>	Rende attivo il campo di testo modificabile associato alla variabile specificata nell'argomento.
<code>setSelection</code>	Imposta gli indici iniziale e finale dell'estensione di selezione.

Selection.getBeginIndex

Sintassi

```
Selection.getBeginIndex();
```

Argomenti

Nessuno.

Descrizione

Metodo; restituisce l'indice all'inizio dell'estensione di selezione. In assenza di indice o di un campo attivo, il metodo restituisce -1. Gli indici per l'estensione della selezione iniziano da zero (la prima posizione è 0, la seconda è 1 e così via).

Lettore

Flash 5 o versione successiva.

Selection.getCaretIndex

Sintassi

```
Selection.getCaretIndex();
```

Argomenti

Nessuno.

Descrizione

Metodo; restituisce l'indice della posizione del cursore lampeggiante. In assenza di cursore lampeggiante visualizzato, il metodo restituisce -1. Gli indici per l'estensione della selezione iniziano da zero (la prima posizione è 0, la seconda è 1 e così via).

Lettore

Flash 5 o versione successiva.

Selection.getEndIndex

Sintassi

```
Selection.getEndIndex();
```

Argomenti

Nessuno.

Descrizione

Metodo; restituisce l'indice finale dell'estensione di selezione correntemente attiva. In assenza di indice o di estensione di selezione correntemente attiva, il metodo restituisce -1. Gli indici per l'estensione della selezione iniziano da zero (la prima posizione è 0, la seconda è 1 e così via).

Lettore

Flash 5 o versione successiva.

Selection.getFocus

Sintassi

```
Selection.getFocus();
```

Argomenti

Nessuno.

Descrizione

Metodo; restituisce il nome della variabile del campo di testo modificabile correntemente attivo. In assenza di campo di testo correntemente attivo, il metodo restituisce null.

Lettore

Flash 5 o versione successiva.

Esempio

Il codice seguente restituisce il nome della variabile.

```
_root.anyMovieClip.myTextField.
```

Selection.setFocus

Sintassi

```
Selection.setFocus(variabile);
```

Argomenti

variabile Una stringa che specifica il nome di una variabile associata al campo di testo in notazione del punto (.) o della barra inclinata (/).

Descrizione

Metodo; rende attivo il campo di testo modificabile associato alla *variabile* specificata.

Lettores

Flash 5 o versione successiva.

Selection.setSelection

Sintassi

```
Selection.setSelection(inizio, fine);
```

Argomenti

inizio L'indice iniziale dell'estensione di selezione.

fine L'indice finale dell'estensione di selezione.

Descrizione

Metodo; imposta l'estensione di selezione del campo di testo correntemente attivo. La nuova estensione di selezione inizia dall'indice specificato nell'argomento *inizio* e termina all'indice specificato nell'argomento *fine*. Gli indici per l'estensione della selezione iniziano da zero (la prima posizione è 0, la seconda è 1 e così via). Questo metodo non ha alcun effetto in assenza di campo di testo correntemente attivo.

Lettores

Flash 5 o versione successiva.

set

Sintassi

```
variabile = espressione;  
set(variabile, espressione);
```

Argomenti

variabile Il nome del contenitore in cui viene memorizzato il valore dell'argomento *espressione*.

espressione Il valore (o l'espressione riconducibile a un valore) assegnato alla variabile.

Descrizione

Azione; assegna un valore a una variabile. Una variabile è un contenitore di informazioni. Il contenitore resta invariato, ma il contenuto può variare. Modificando il valore di una variabile durante la riproduzione di un filmato, è possibile registrare e salvare informazioni sulle azioni svolte dall'utente, registrare valori modificati durante la riproduzione del filmato o verificare se una condizione è `true` o `false`.

Le variabili possono contenere numeri o stringhe di caratteri. A ogni filmato e clip filmato è associato il proprio insieme di variabili, in cui a ogni variabile è assegnato il proprio valore che è indipendente da quello delle variabili in altri filmati o clip filmato.

ActionScript è un linguaggio senza tipi, ossia non occorre definire in modo esplicito il tipo delle variabili per indicare se conterranno un numero o una stringa. Flash interpreta infatti il tipo di dati come un numero intero o una stringa a seconda del caso.

L'uso dell'istruzione `set` unitamente all'azione `call` consente di passare o restituire valori.

Lettore

Flash 4 o versione successiva.

Esempio

Questo esempio imposta la variabile `orig_x_pos` che memorizza la posizione originale sull'asse delle *x* del clip filmato `ship` in modo da poter riportare in seguito la nave nella posizione di partenza nel filmato.

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

Ciò equivale al seguente codice:

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

Vedere anche

“var” a pagina 373

“call” a pagina 228

setProperty

Sintassi

`setProperty(target, proprietà, espressione);`

Argomenti

target Il percorso per il nome dell'istanza di clip filmato di cui si imposta la proprietà.

proprietà La proprietà da impostare.

espressione Il valore impostato per la proprietà.

Descrizione

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

Lettore

Flash 4 o versione successiva.

Esempio

L'istruzione seguente imposta la proprietà `_alpha` del un clip filmato `star` su 30 % quando il pulsante viene premuto.

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

Vedere anche

“getProperty” a pagina 264

Sound (oggetto)

L'oggetto `Sound` consente di impostare e controllare i suoni in una determinata istanza di clip filmato o per l'intera linea temporale, se non si specifica un *target* quando si crea un nuovo oggetto `Sound`. Prima di chiamare i metodi dell'oggetto `Sound` è necessario crearne un'istanza usando la funzione di costruzione `new Sound()`.

L'oggetto `Sound` è supportato solo in ambiente Flash Player 5.

Riepilogo dei metodi validi per l'oggetto Sound

Metodo	Descrizione
<code>attachSound</code>	Associa il suono specificato nell'argomento.
<code>getPan</code>	Restituisce il valore della chiamata <code>setPan</code> precedente.
<code>getTransform</code>	Restituisce il valore della chiamata <code>setTransform</code> precedente.
<code>getVolume</code>	Restituisce il valore della chiamata <code>setVolume</code> precedente.
<code>setPan</code>	Imposta il bilanciamento destra/sinistra dell'audio.
<code>setTransform</code>	Imposta la trasformazione dell'audio.
<code>setVolume</code>	Imposta il livello del volume dell'audio.
<code>start</code>	Avvia la riproduzione di un suono dall'inizio o, se indicato, a partire dal punto impostato nell'argomento.
<code>stop</code>	Arresta il suono specificato o tutti i suoni in corso di riproduzione.

Funzione di costruzione per l'oggetto Sound

Sintassi

```
new Sound();  
new Sound(target);
```

Argomenti

target L'istanza del clip filmato a cui viene applicato l'oggetto Sound. È un argomento opzionale.

Descrizione

Metodo; crea un nuovo oggetto Sound per un clip filmato specificato. Se non si specifica un *target*, l'oggetto Sound controlla tutti i suoni nell'intera linea temporale.

Lettore

Flash 5 o versione successiva.

Esempio

```
GlobalSound = new Sound();  
MovieSound = new Sound(mymovie);
```

Sound.attachSound

Sintassi

```
Suono.attachSound("IDNome");
```

Argomenti

IDNome Il nome della nuova istanza del suono. Corrisponde al nome immesso nel campo Identificatore nella finestra di dialogo Proprietà di concatenamento del simbolo. Questo argomento deve essere racchiuso tra virgolette (" ").

Descrizione

Metodo; associa il suono specificato nell'argomento *IDNome* all'oggetto Sound specificato. Il suono deve essere nella libreria del filmato corrente e selezionato per l'esportazione nella finestra di dialogo Proprietà di concatenamento del simbolo. È necessario chiamare `Sound.start` per avviare la riproduzione del suono.

Lettore

Flash 5 o versione successiva.

Vedere anche

"Sound.start" a pagina 353

Sound.getPan

Sintassi

```
Suono.getPan();
```

Argomenti

Nessuno.

Descrizione

Metodo; restituisce il bilanciamento impostato nell'ultima chiamata `setPan` come numero intero compreso tra -100 e 100. Questa impostazione controlla il bilanciamento destra/sinistra dell'audio corrente e futuro di un filmato.

Questo metodo è cumulabile con il metodo `setVolume` o `setTransform`.

Lettore

Flash 5 o versione successiva.

Vedere anche

"Sound.setPan" a pagina 349

"Sound.setTransform" a pagina 350

Sound.getTransform

Sintassi

```
Suono.getTransform();
```

Argomenti

Nessuno.

Descrizione

Metodo; restituisce le informazioni sulla trasformazione dell'audio per l'oggetto Sound specificato impostato con l'ultima chiamata `setTransform`.

Lettore

Flash 5 o versione successiva.

Vedere anche

“`Sound.setTransform`” a pagina 350

Sound.getVolume

Sintassi

```
Suono.getVolume();
```

Argomenti

Nessuno.

Descrizione

Metodo; restituisce il livello del volume dell'audio come numero intero compreso tra 0 e 100, dove 0 corrisponde all'assenza di audio e 100 al volume massimo. L'impostazione predefinita è 100.

Lettore

Flash 5 o versione successiva.

Vedere anche

“`Sound.setVolume`” a pagina 353

Sound.setPan

Sintassi

```
Suono.setPan(bilanciamento);
```

Argomenti

bilanciamento Un numero intero che specifica il bilanciamento destra/sinistra dell'audio. L'intervallo di valori validi è compreso tra -100 e 100, dove -100 corrisponde all'uso del solo canale sinistro e 100 all'uso del solo canale destro; 0 distribuisce uniformemente l'audio sui due canali.

Descrizione

Metodo; determina come l'audio viene riprodotto nei canali destro e sinistro (altoparlanti). Per l'audio mono, *bilanciamento* indica da quale altoparlante (destro o sinistro) verrà emesso il suono.

Questo metodo è cumulabile con i metodi `setVolume` e `setTransform`. Inoltre la chiamata di questo metodo elimina e aggiorna le precedenti impostazioni di `setPan` e `setTransform`.

Lettore

Flash 5 o versione successiva.

Esempio

L'esempio seguente usa `setVolume` e `setPan` per controllare un oggetto `Sound` con il target specificato "u2".

```
onClipEvent(mouseDown) {  
    // Crea un nuovo oggetto Sound  
    s = new Sound(this);  
    // Collega un suono dalla libreria.  
    s.attachSound("u2");  
    // Imposta il volume al 50%  
    s.setVolume(50);  
    // Disattiva il canale destro  
    s.setPan(-100);  
    // Parte dopo 30 secondi dall'inizio della traccia e  
    // riproduce il suono 5 volte  
    s.start(30, 5);  
}
```

Vedere anche

"`Sound.setTransform`" a pagina 350

"`Sound.setVolume`" a pagina 353

Sound.setTransform

Sintassi

Suono.setTransform(oggettoTrasformazioneSuono);

Argomenti

oggettoTrasformazioneSuono Un oggetto creato con la funzione di costruzione per l'oggetto generico `Object`.

Descrizione

Metodo; imposta i dati relativi alla trasformazione dell'audio per un oggetto `Sound`. Questo metodo è cumulabile con i metodi `setVolume` e `setPan`. Inoltre la chiamata di questo metodo elimina e aggiorna tutte le precedenti impostazioni di `setPan` e `setVolume`. Questa chiamata può essere usata da utenti esperti che desiderano aggiungere effetti interessanti all'audio.

L'audio può usare un notevole quantitativo di spazio su disco e di memoria. Poiché l'audio stereo usa una quantità di dati doppia rispetto all'audio mono, si consiglia di usare un audio mono a 22 Khz/6 bit. È possibile usare il metodo `setTransform` per riprodurre l'audio mono in modalità stereo, quello stereo in modalità mono e per aggiungere effetti audio interessanti.

L'argomento `oggettoTrasformazioneSuono` è un oggetto creato con il metodo di costruzione dell'elemento `Object` generico, specificando i parametri che indicano come distribuire l'audio sui canali destro e sinistro (altoparlanti).

I parametri per `oggettoTrasformazioneSuono` sono i seguenti:

`ll` Un valore percentuale che indica il quantitativo di input sinistro da riprodurre nell'altoparlante sinistro (da -100 a 100).

`lr` Un valore percentuale che indica il quantitativo di input destro da riprodurre nell'altoparlante sinistro (da -100 a 100).

`rr` Un valore percentuale che indica il quantitativo di input destro da riprodurre nell'altoparlante destro (da -100 a 100).

`rl` Un valore percentuale che indica il quantitativo di input sinistro da riprodurre nell'altoparlante destro (da -100 a 100).

Il risultato netto dei parametri è rappresentato dalle formule seguenti:

$$\text{OutputSinistro} = \text{input sinistro} * ll + \text{input destro} * lr$$

$$\text{OutputDestro} = \text{input destro} * rr + \text{input sinistro} * rl$$

I valori di input sinistro o di input destro sono determinati dal tipo (stereo o mono) di audio del filmato.

Le impostazioni di trasformazione predefinite dell'audio stereo che divide uniformemente l'input tra gli altoparlanti destro e sinistro sono:

`ll = 100`

`lr = 0`

`rr = 100`

`rl = 0`

Le impostazioni di trasformazione predefinite dell'audio mono che riproduce tutto l'input audio nell'altoparlante sinistro sono:

`ll = 100`

`lr = 100`

`rr = 0`

`rl = 0`

Lettore

Flash 5 o versione successiva.

Esempio

L'esempio seguente crea un oggetto di trasformazione suono che riproduce entrambi i canali sinistro e destro in quello sinistro.

```
mySoundTransformObject = new Object
mySoundTransformObject.ll = 100
mySoundTransformObject.lr = 100
mySoundTransformObject.rr = 0
mySoundTransformObject.rl = 0
```

Il codice precedente crea un oggetto di trasformazione suono. Per applicarlo a un oggetto Sound, è necessario passare tale oggetto all'oggetto Sound usando il metodo `setTransform` come segue:

```
mySound.setTransform(mySoundTransformObject);
```

Gli esempi seguenti sono configurazioni che possono essere impostate tramite `setTransform`, ma non possono essere impostate tramite `setVolume` o `setPan`, anche se combinati.

Il codice seguente riproduce entrambi i canali sinistro e destro attraverso quello sinistro:

```
mySound.setTransform(soundTransformObjectLeft);
```

Nel codice precedente l'oggetto di trasformazione del suono, *soundTransformObjectLeft* ha i parametri seguenti:

```
ll = 100
lr = 100
rr = 0
rl = 0
```

Esempio

Il codice seguente riproduce un audio stereo in formato mono:

```
setTransform(soundTransformObjectMono);
```

Nel codice precedente l'oggetto di trasformazione del suono, *soundTransformObjectMono* ha i parametri seguenti:

```
ll = 50
lr = 50
rr = 50
rl = 50
```

Esempio

Il codice seguente riproduce metà della capacità del canale sinistro e il resto di quest'ultimo viene riprodotto sul canale destro:

```
setTransform(soundTransformObjectHalf);
```

Nel codice precedente l'oggetto di trasformazione del suono, *soundTransformObjectHalf* ha i parametri seguenti:

```
ll = 50
lr = 0
rr = 100
rl = 50
```


Vedere anche

“Funzione di costruzione per l'oggetto Object” a pagina 325

Sound.setVolume

Sintassi

```
Suono.setVolume(volume);
```

Argomenti

volume Un numero tra 0 e 100 che indica il livello del volume. 100 corrisponde al massimo volume e 0 all'assenza di audio. L'impostazione predefinita è 100.

Descrizione

Metodo; imposta il volume per l'oggetto Sound.

Questo metodo è cumulabile con i metodi `setPan` e `setTransform`.

Letture

Flash 5 o versione successiva.

Esempio

L'esempio seguente imposta il volume al 50% e quindi trasferisce l'audio dall'altoparlante sinistro a quello destro.

```
onClipEvent (load) {  
    i = -100;  
    s = new sound();  
    s.setVolume(50);  
}  
onClipEvent (enterFrame) {  
    S.setPan(i++);  
}
```

Vedere anche

“Sound.setPan” a pagina 349

“Sound.setTransform” a pagina 350

Sound.start

Sintassi

```
Suono.start();  
  
Suono.start([OffsetInSecondi, ciclo]);
```

OffsetInSecondi Argomento opzionale che consente di avviare la riproduzione del suono in un punto specifico. Ad esempio, se si desidera avviare la riproduzione di un suono della durata di 30 secondi in mezzo, specificare 15 per l'argomento *OffsetInSecondi*. Il suono non viene ritardato di 15 secondi, ma viene invece avviato all'altezza dell'indicatore dei 15 secondi.

ciclo Argomento opzionale che consente di specificare il numero di volte che un suono viene riprodotto.

Descrizione

Metodo; avvia la riproduzione dell'ultimo suono associato dall'inizio, in assenza di argomento specificato, o a partire dal punto indicato dall'argomento *OffsetInSeconds*.

Lettore

Flash 5 o versione successiva.

Vedere anche

“Sound.setPan” a pagina 349

“Sound.stop” a pagina 354

Sound.stop

Sintassi

```
Suono.stop();
```

```
Suono.stop(["IDNome"]);
```

Argomenti

IDNome Argomento opzionale che specifica l'interruzione di un determinato suono. L'argomento *IDNome* deve essere racchiuso tra virgolette (" ").

Descrizione

Metodo; interrompe tutti i suoni in corso di riproduzione, in assenza di argomento, o solo quello specificato nell'argomento *IDNome*.

Lettore

Flash 5 o versione successiva.

Vedere anche

“Sound.start” a pagina 353

_soundbuftime

Sintassi

```
_soundbuftime = intero;
```

Argomenti

intero Il numero di secondi prima dell'inizio dello streaming del filmato.

Descrizione

Proprietà (globale); stabilisce il numero di secondi di streaming dell'audio da accumulare nel buffer prima della riproduzione. Il valore predefinito è 5 secondi.

Lettore

Flash 4 o versione successiva.

startDrag

Sintassi

```
startDrag(target);  
startDrag(target,[bloccato]);  
startDrag(target [,bloccato [,sinistra , superiore , destra ,  
inferiore]]);
```

Argomenti

target Il percorso target del clip filmato da trascinare.

bloccato Valore booleano che specifica se il clip filmato mobile è bloccato al centro rispetto alla posizione del mouse (*true*) oppure in corrispondenza del punto in cui l'utente ha fatto clic inizialmente nel clip filmato (*false*). È un argomento opzionale.

sinistra, superiore, destra, inferiore Valori relativi alle coordinate del filmato principale del clip filmato che definiscono un rettangolo di delimitazione per quest'ultimo. Si tratta di argomenti opzionali.

Descrizione

Azione; rende il clip filmato *target* trascinabile durante la riproduzione del filmato. È possibile trascinare un solo clip filmato alla volta. Una volta eseguita un'operazione *startDrag*, il clip filmato rimane trascinabile fino alla chiamata esplicita di un'azione *stopDrag* o di un'azione *startDrag* per un altro clip filmato.

Esempio

Per creare un clip filmato che gli utenti possono collocare in qualsiasi posizione, associare le azioni *startDrag* e *stopDrag* a un pulsante interno al clip filmato, come segue:

```
on(press) {  
    startDrag("",true);  
}  
on(release) {  
    stopDrag();  
}
```

Vedere anche

“*stopDrag*” a pagina 356

“*_droptarget*” a pagina 253

stop

Sintassi

```
stop;
```

Argomenti

Nessuno.

Descrizione

Azione; interrompe il clip filmato attualmente in riproduzione. Questa azione viene comunemente usata per controllare i clip filmato tramite pulsanti.

Lettore

Flash 3 o versione successiva.

stopAllSounds

Sintassi

```
stopAllSounds();
```

Argomenti

Nessuno.

Descrizione

Azione; interrompe tutti i suoni in corso di riproduzione in un filmato senza arrestare l'indicatore di riproduzione. La riproduzione dei suoni in streaming ricomincia quando l'indicatore di riproduzione si sposta sui fotogrammi in cui si trovano.

Lettore

Flash 3 o versione successiva.

Esempio

Il codice seguente è applicabile a un pulsante che interrompe tutti i suoni di un filmato quando viene premuto.

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

Vedere anche

“Sound (oggetto)” a pagina 346

stopDrag

Sintassi

```
stopDrag();
```

Argomenti

Nessuno.

Descrizione

Azione; arresta l'operazione di trascinamento corrente.

Lettore

Flash 4 o versione successiva.

Esempio

Questa istruzione arresta l'azione di trascinamento sull'istanza `mc` quando l'utente rilascia il pulsante del mouse.

```
on(press) {  
    startDrag("mc");  
}  
on(release) {  
    stopdrag();  
}
```

Vedere anche

“startDrag” a pagina 355

“_droptarget” a pagina 253

String (funzione)

Sintassi

`String(espressione);`

Argomenti

espressione Numero, valore booleano, variabile o oggetto da convertire in stringa.

Descrizione

Funzione; restituisce una rappresentazione in formato stringa dell'argomento specificato, come segue:

Se *x* è un valore booleano, la stringa restituita è `true` o `false`.

Se *x* è un numero, la stringa restituita è l'equivalente rappresentazione decimale del numero.

Se *x* è una stringa, la stringa restituita è *x*.

Se *x* è un oggetto, il valore restituito è una rappresentazione in formato stringa dell'oggetto generata dalla chiamata della proprietà `string` per l'oggetto o dalla chiamata di `object.toString`, in assenza di tale proprietà.

Se *x* è un clip filmato, il valore restituito è il percorso target del clip filmato in notazione dalla barra inclinata (/).

Se *x* non è definito, il valore restituito è una stringa vuota.

Letture

Flash 3 o versione successiva.

Vedere anche

“Object.toString” a pagina 326

“Number.toString” a pagina 324

“String (oggetto)” a pagina 358

“ ” (delimitatore di stringa)” a pagina 358

" " (delimitatore di stringa)

Sintassi

`"testo"`

Argomenti

`testo` Qualsiasi testo.

Descrizione

Delimitatore di stringa; se usate prima e dopo una stringa, le virgolette indicano che si tratta di un letterale e non di una variabile, di un valore numerico o di un altro elemento di ActionScript.

Lettore

Flash 4 o versione successiva.

Esempio

Questa istruzione usa le virgolette per indicare che la stringa "Prince Edward Island" è una stringa letterale e non il valore di una variabile:

```
province = "Prince Edward Island"
```

Vedere anche

"String (oggetto)" a pagina 358

"String (funzione)" a pagina 357

String (oggetto)

L'oggetto String è un wrapper per il tipo di dati di base stringa, consentendo di usare i metodi e le proprietà dell'oggetto String per manipolare valori di tipo di base stringa. È possibile convertire il valore di qualsiasi oggetto in una stringa tramite la funzione `String()`.

Tutti i metodi dell'oggetto String sono generici, ad eccezione di `concat`, `fromCharCode`, `slice` e `substr`. Ciò implica che i metodi stessi chiamano `this.toString` prima di effettuare le relative operazioni e possono quindi essere usati con altri oggetti diversi da String.

È possibile chiamare qualsiasi metodo dell'oggetto String usando il metodo di costruzione `new String` o un valore letterale di tipo stringa. Se si specifica un letterale stringa, l'interprete di ActionScript lo converte automaticamente in un oggetto String temporaneo, chiama il metodo, quindi elimina l'oggetto creato. È inoltre possibile usare la proprietà `String.length` con un letterale stringa.

È importante non confondere un letterale stringa e un'istanza dell'oggetto String. Nell'esempio seguente, la prima riga di codice crea il letterale stringa `s1`, e la seconda riga di codice crea l'istanza `s2` dell'oggetto String.

```
s1 = "foo"  
s2 = new String("foo")
```

Si consiglia l'uso di letterali stringa a meno che non sia richiesto l'uso di un oggetto String, in quanto gli oggetti String possono presentare un comportamento non intuitivo.

Riepilogo dei metodi validi per l'oggetto String

Metodo	Descrizione
<code>charAt</code>	Restituisce un numero corrispondente alla posizione del carattere nella stringa.
<code>charCodeAt</code>	Restituisce il valore del carattere in una determinata posizione come numero intero a 16 bit compreso tra 0 e 65535.
<code>concat</code>	Concatena il testo di due stringhe e restituisce una nuova stringa.
<code>fromCharCode</code>	Restituisce una stringa composta dai caratteri specificati come argomenti.
<code>indexOf</code>	Ricerca nella stringa il valore specificato negli argomenti e ne restituisce l'indice. Se il valore è presente più volte, viene restituito l'indice della prima occorrenza. Se il valore è assente, viene restituito -1.
<code>lastIndexOf</code>	Restituisce l'ultima occorrenza della sottostringa all'interno della stringa che appare prima della posizione iniziale specificata nell'argomento; se la sottostringa non viene trovata, viene restituito -1.
<code>slice</code>	Estrae una sezione di una stringa e restituisce una nuova stringa.
<code>split</code>	Divide un oggetto String in una matrice di stringhe separando la stringa in sottostringhe.
<code>substr</code>	Restituisce il numero di caratteri specificato di una stringa, iniziando dalla posizione specificata nell'argomento.
<code>substring</code>	Restituisce i caratteri della stringa tra i due indici specificati come argomenti.
<code>toLowerCase</code>	Converte la stringa in caratteri minuscoli e restituisce il risultato.
<code>toUpperCase</code>	Converte la stringa in caratteri maiuscoli e restituisce il risultato.

Riepilogo delle proprietà valide per l'oggetto String

Proprietà	Descrizione
<code>length</code>	Restituisce la lunghezza della stringa.

Funzione di costruzione dell'oggetto String

Sintassi

```
new String(valore);
```

Argomenti

valore Il valore iniziale del nuovo oggetto String.

Descrizione

Funzione di costruzione; crea un nuovo oggetto String.

Lettore

Flash 5 o versione successiva.

Vedere anche

“String (funzione)” a pagina 357

“" " (delimitatore di stringa)” a pagina 358

String.charAt

Sintassi

```
Stringa.charAt(indice);
```

Argomenti

indice Il numero del carattere nella stringa da restituire.

Descrizione

Metodo; restituisce il carattere specificato dall'argomento *indice*. L'indice del primo carattere in una stringa è 0. Se *indice* non è un numero compreso tra 0 e `string.length - 1`, viene restituita una stringa vuota.

Lettore

Flash 5 o versione successiva.

String.charCodeAtAt

Sintassi

```
Stringa.charCodeAtAt(indice);
```

Argomenti

indice Il numero del carattere per il quale viene restituito il valore.

Descrizione

Metodo; restituisce il valore del carattere specificato da *indice*. Il valore restituito è un numero intero a 16 bit compreso tra 0 e 65535.

Questo metodo è simile a `string.charAt` eccetto che viene restituito il valore per il carattere in una data posizione invece che la sottostringa contenente il carattere.

Lettore

Flash 5 o versione successiva.

String.concat

Sintassi

```
Stringa.concat(valore0,...valoreN);
```

Argomenti

valore0,...*valoreN* Zero o più valori da concatenare.

Descrizione

Metodo; concatena i valori specificati e restituisce una nuova stringa. Se necessario, ogni argomento *valore* viene convertito in una stringa e aggiunto nell'ordine alla fine della stringa.

Lettore

Flash 5 o versione successiva.

String.fromCharCode

Sintassi

```
Stringa.fromCharCode(c1,c2,...cN);
```

Argomenti

c1,*c2*,...*cN* I caratteri da trasformare in una stringa.

Descrizione

Metodo; restituisce una stringa composta dai caratteri specificati come argomenti.

Lettore

Flash 5 o versione successiva.

String.indexOf

Sintassi

```
Stringa.indexOf(valore);
```

```
Stringa.indexOf (valore, inizio);
```

Argomenti

valore Numero intero o stringa che specifica la sottostringa da ricercare all'interno di *Stringa*.

inizio Numero intero che specifica il punto di inizio della sottostringa. È un argomento opzionale.

Descrizione

Metodo; ricerca nella stringa l'elemento *valore* specificato e ne restituisce la posizione della prima occorrenza. Se il valore non viene trovato, il metodo restituisce -1.

Lettore

Flash 5 o versione successiva.

String.lastIndexOf

Sintassi

```
Stringa.lastIndexOf(sottostringa);
```

```
Stringa.lastIndexOf(sottostringa, inizio);
```

Argomenti

sottostringa Numero intero o stringa che specifica la stringa da ricercare.

inizio Numero intero che specifica il punto di inizio della sottostringa. È un argomento opzionale.

Descrizione

Metodo; ricerca nella stringa di chiamata la *sottostringa* e restituisce l'indice dell'ultima occorrenza trovata. Se la *sottostringa* non è presente, il metodo restituisce -1.

Lettore

Flash 5 o versione successiva.

String.length

Sintassi

```
string.length
```

Argomenti

Nessuno.

Descrizione

Proprietà; restituisce il numero di caratteri nell'oggetto String specificato. L'indice dell'ultimo carattere per qualsiasi stringa *x* è *x.length-1*.

Lettore

Flash 5 o versione successiva.

String.slice

Sintassi

```
Stringa.slice(inizio, fine);
```

Argomenti

inizio Un numero che specifica l'indice del punto di inizio della porzione. Se *inizio* è un numero negativo, il punto di inizio è specificato a partire dalla fine della stringa, dove -1 è l'ultimo carattere.

fine Un numero che specifica l'indice del punto finale della porzione. Se *fine* non è specificato, la porzione include tutti i caratteri dall'inizio alla fine della stringa. Se *fine* è un numero negativo, il punto finale è specificato a partire dalla fine della stringa, dove -1 è l'ultimo carattere.

Descrizione

Metodo; estrae una porzione o una sottostringa dell'oggetto String specificato e la restituisce come nuova stringa senza modificare l'oggetto String originale. La stringa restituita include il carattere *inizio* e tutti i caratteri tra questo e il carattere identificato da *fine* che è escluso.

Lettore

Flash 5 o versione successiva.

String.split

Sintassi

```
Stringa.split(delimitatore);
```

Argomenti

delimitatore Il carattere usato per delimitare la stringa.

Descrizione

Metodo; divide l'oggetto String rompendo la stringa nel punto in cui viene trovato l'argomento *delimitatore*, quindi restituisce le sottostringhe in una matrice. Se non viene specificato un delimitatore, la matrice restituita contiene un solo elemento, la stringa stessa. Se il delimitatore è una stringa vuota, ogni carattere nell'oggetto String diventa un elemento nella matrice.

Lettore

Flash 5 o versione successiva.

String.substr

Sintassi

```
Stringa.substr(inizio, lunghezza);
```

Argomenti

inizio Numero intero che indica la posizione del primo carattere nella sottostringa che verrà creata. Se *inizio* è un numero negativo, il punto di inizio è specificato a partire dalla fine della stringa, dove -1 è l'ultimo carattere.

lunghezza Numero di caratteri nella sottostringa creata. Se *lunghezza* non è specificato, la sottostringa include tutti i caratteri dall'inizio alla fine della stringa.

Descrizione

Metodo; restituisce i caratteri in una stringa dall'indice specificato nell'argomento *inizio*, fino al numero di caratteri specificato nell'argomento *lunghezza*.

Lettore

Flash 5 o versione successiva.

String.substring

Sintassi

```
Stringa.substring(da, a);
```

Argomenti

da Numero intero che indica la posizione del primo carattere nella sottostringa che verrà creata. I valori validi per *da* sono compresi tra 0 e `string.length - 1`.

a Numero intero che corrisponde al valore +1 dell'indice dell'ultimo carattere nella sottostringa che verrà creata. I valori validi per *a* sono compresi tra 1 e `string.length`. Se l'argomento *a* non è specificato, la fine della sottostringa corrisponde con la fine della stringa. Se *da* è uguale a *a*, il metodo restituisce una stringa vuota. Se *da* è maggiore di *a*, gli argomenti vengono automaticamente scambiati prima di eseguire la funzione.

Descrizione

Metodo; restituisce una stringa costituita dai caratteri tra i punti specificati dagli argomenti *da* e *a*.

Lettore

Flash 5 o versione successiva.

String.toLowerCase

Sintassi

```
Stringa.toLowerCase();
```

Argomenti

Nessuno.

Descrizione

Metodo; restituisce una copia dell'oggetto String con tutti i caratteri maiuscoli convertiti in minuscoli.

Lettore

Flash 5 o versione successiva.

String.toUpperCase

Sintassi

Stringa.toUpperCase();

Argomenti

Nessuno.

Descrizione

Metodo; restituisce una copia dell'oggetto String con tutti i caratteri minuscoli convertiti in maiuscoli.

Lettore

Flash 5 o versione successiva.

substring

Sintassi

substring(*stringa*, *indice*, *numero*);

Argomenti

stringa La stringa da cui estrarre quella nuova.

indice Il numero del primo carattere da estrarre.

numero Il numero di caratteri da includere nella stringa estratta, escluso il carattere specificato nell'argomento indice.

Descrizione

Funzione di stringa; estrae parte di una stringa.

Lettore

Flash 4 o versione successiva. Questa funzione è diventata obsoleta in Flash 5.

Vedere anche

“String.substring” a pagina 364

_target

Sintassi

nomeistanza._target

Argomenti

nomeistanza Il nome di un'istanza di clip filmato.

Descrizione

Proprietà (sola lettura); restituisce il percorso target dell'istanza di clip filmato specificata nell'argomento *nomeistanza*.

Lettore

Flash 4 o versione successiva.

targetPath

Sintassi

```
targetPath(OggettoMovieClip);
```

Argomenti

OggettoMovieClip Riferimento (ad esempio, `_root` o `_parent`) al clip filmato per il quale viene recuperato il percorso target.

Descrizione

Funzione; restituisce una stringa contenente il percorso target di *OggettoMovieClip*. Il percorso target è restituito in notazione del punto. Per recuperare il percorso target in notazione della barra inclinata, usare la proprietà `_target`.

Lettore

Flash 5 o versione successiva.

Esempio

Gli esempi seguenti sono equivalenti. Il primo esempio usa la notazione del punto e il secondo quella della barra inclinata.

```
targetPath (Board.Block[index*2+1]) {  
    play();  
}
```

È equivalente a:

```
tellTarget ("Board/Block:" + (index*2+1)) {  
    play();  
}
```

Vedere anche

“eval” a pagina 257

tellTarget

Sintassi

```
tellTarget(target) {  
    istruzione;  
}
```

Argomenti

target Una stringa di percorso target che specifica la linea temporale da controllare.

istruzione Istruzioni applicate alla linea temporale target.

Descrizione

Azione; applica le istruzioni specificate nell'argomento *istruzione* alla linea temporale specificata nell'argomento *target*. L'azione `tellTarget` è utile per il controllo della navigazione. Assegnare `tellTarget` ai pulsanti per arrestare o avviare i clip filmato in altri punti dello stage. È inoltre possibile spostarsi in un particolare fotogramma all'interno di un clip filmato specifico. Ad esempio, è possibile assegnare `tellTarget` a un pulsante per arrestare o avviare clip filmato sullo stage o per inviare un clip filmato su un particolare fotogramma.

L'azione `tellTarget` è molto simile all'azione `with`, ad eccezione del fatto che `with` agisce su un clip filmato o altro oggetto come *target*, mentre `tellTarget` richiede un percorso *target* al clip filmato e non può controllare oggetti.

Lettores

Flash 3 o versione successiva. Questa azione è diventata obsoleta in Flash 5 ed è consigliato l'uso della nuova azione `with`.

Esempio

Questa istruzione `tellTarget` controlla l'istanza di clip filmato `ball` sulla linea temporale principale. Il fotogramma 1 del clip filmato è vuoto e ad esso è associata un'azione `stop` in modo che non sia visibile sullo stage. Quando il pulsante associato all'azione seguente viene premuto, `tellTarget` sposta l'indicatore di riproduzione del clip filmato `ball` al fotogramma 2 e riproduce l'animazione che inizia in quel punto.

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

Vedere anche

“with” a pagina 376

this

Sintassi

`this`

Argomenti

Nessuno.

Descrizione

Parola chiave; fa riferimento a un oggetto o a un'istanza di clip filmato. La parola chiave `this` ha lo stesso scopo e funzione in ActionScript e in JavaScript, con alcune funzionalità aggiuntive in ActionScript. In ActionScript, quando uno script viene eseguito, `this` fa riferimento all'istanza di clip filmato contenente lo script. Quando viene usato con una chiamata di metodo, `this` contiene un riferimento all'oggetto contenente il metodo eseguito.

Lettore

Flash 5 o versione successiva.

Esempio

Nell'esempio seguente, la parola chiave `this` fa riferimento all'oggetto `Circle`.

```
function Circle(radius){
    this.radius = radius;
    this.area = math.PI * radius * radius;
}
```

Nell'istruzione seguente assegnata a un fotogramma, la parola chiave `this` fa riferimento al clip filmato corrente.

```
// Imposta la proprietà alpha del filmato corrente su 20.
this._alpha = 20;
```

Nell'istruzione seguente interna a un gestore `onClipEvent`, la parola chiave `this` fa riferimento al clip filmato corrente.

```
// Al momento del caricamento del filmato, viene attivata
// un'azione startDrag per il clip filmato corrente.

onClipEvent (load) {
    startDrag (this, true);
}
```

Vedere anche

“new” a pagina 317

toggleHighQuality

Sintassi

```
toggleHighQuality();
```

Argomenti

Nessuno.

Descrizione

Azione; attiva/disattiva l'antialiasing in Flash Player. L'antialiasing smussa i bordi degli oggetti e rallenta la riproduzione del filmato. L'azione `toggleHighQuality` influenza tutti i filmati in Flash Player.

Lettore

Flash 2 o versione successiva.

Esempio

Il codice seguente è applicabile a un pulsante che attiva/disattiva l'antialiasing quando viene premuto.

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


Vedere anche

“_quality” a pagina 337

“_highquality” a pagina 268

_totalframes

Sintassi

nomeistanza._totalframes

Argomenti

nomeistanza Il nome del clip filmato da valutare.

Descrizione

Proprietà (sola lettura); valuta il clip filmato specificato nell'argomento *nomeistanza* e restituisce il numero totale di fotogrammi del filmato.

Lettore

Flash 4 o versione successiva.

trace

Sintassi

trace(*espressione*);

Argomenti

espressione Un'istruzione da valutare. Durante le prove del filmato, i risultati dell'argomento *espressione* vengono visualizzati nella finestra Output.

Descrizione

Azione; valuta *espressione* e visualizza i risultati nella finestra Output in modalità di prova filmato.

Usare la funzione *trace* per registrare note di programmazione o per visualizzare messaggi nella finestra Output durante le prove del filmato. Usare il parametro *espressione* per verificare se una condizione esiste o per visualizzare valori nella finestra Output. L'azione *trace* è simile alla funzione *alert* in JavaScript.

Lettore

Flash 4 o versione successiva.

Esempio

Questo esempio è tratto da un gioco in cui l'istanza di clip filmato trascinabile `rabbi` deve essere rilasciata su un target specifico. Un'istruzione condizionale valuta la proprietà `_droptarget` ed esegue diverse azioni a seconda di dove il clip filmato `rabbi` viene rilasciato. L'azione `trace` è usata alla fine dello script per valutare la posizione del clip filmato `rabbi` e visualizzare il risultato nella finestra Output. Se `rabbi` non si comporta come previsto (ad esempio, se viene rilasciato nel posto sbagliato), i valori inviati alla finestra Output dall'azione `trace` possono contribuire a identificare il problema nello script.

```
on(press) {
  rabbi.startDrag();
}
on(release) {
  if(eval(_droptarget) != target) {
    rabbi._x = rabbi_x;
    rabbi._y = rabbi_y;
  } else {
    rabbi_x = rabbi._x;
    rabbi_y = rabbi._y;
    target = "_root.pasture";
  }
  trace("rabbi_y = " + rabbi_y);
  trace("rabbi_x = " + rabbi_x);
  stopDrag();
}
```

typeof

Sintassi

`typeof(espressione);`

Argomenti

espressione Stringa, clip filmato, oggetto o funzione.

Descrizione

Operatore; un operatore unario collocato prima di un singolo argomento. Comporta la valutazione dell'*espressione*; il risultato è una stringa che indica se l'espressione è una stringa, un clip filmato, un oggetto o una funzione.

Letture

Flash 5 o versione successiva.

unescape

Sintassi

`unescape(x);`

Argomenti

x Una stringa con sequenze di escape esadecimali.

Descrizione

Funzione di primo livello; valuta l'argomento *x* come stringa, decodifica la stringa a partire da un formato con codifica URL (converte le sequenze esadecimali in caratteri ASCII) e restituisce la stringa.

Lettore

Flash 5 o versione successiva.

Esempio

L'esempio seguente illustra la conversione di una stringa senza sequenze di escape nella relativa stringa con sequenze di escape.

```
escape("Hello{[World]}");
```

Il risultato che utilizza le sequenze di escape è il seguente:

```
("Hello%7B%5BWorld%5D%7D");
```

Per ritornare al formato originale, usare `unescape`:

```
unescape("Hello%7B%5BWorld%5D%7D")
```

Il risultato è il seguente:

```
Hello{[World]}
```

unloadMovie

Sintassi

`unloadMovie(posizione);`

Argomenti

posizione Il livello di profondità o il clip filmato target dal quale scaricare il filmato.

Descrizione

Azione; rimuove da Flash Player un filmato precedentemente caricato tramite l'azione `loadMovie`.

Lettore

Flash 3 o versione successiva.

Esempio

L'esempio seguente scarica il filmato principale, lasciando libero lo stage.

```
unloadMovie(_root);
```

L'esempio seguente scarica il filmato al livello 15, quando l'utente preme il pulsante del mouse.

```
on(press) {  
    unloadMovie(_level15);  
}
```

Vedere anche

“loadMovie” a pagina 283

updateAfterEvent

Sintassi

```
updateAfterEvent(evento clip filmato);
```

Argomenti

Per *evento clip filmato* specificare uno dei seguenti valori:

- **mouseMove** L'azione viene attivata a ogni spostamento del mouse. L'uso delle proprietà `_xmouse` e `_ymouse` consente di determinare la posizione corrente del mouse.
- **mouseDown** L'azione viene attivata quando il pulsante sinistro del mouse viene premuto.
- **mouseUp** L'azione viene attivata quando il pulsante sinistro del mouse viene rilasciato.
- **keyDown** L'azione viene attivata quando viene premuto un tasto. L'uso del metodo `Key.getCode` consente di recuperare informazioni sull'ultimo tasto premuto.
- **keyUp** L'azione viene attivata quando viene rilasciato un tasto. L'uso del metodo `Key.getCode` consente di recuperare informazioni sull'ultimo tasto premuto.

Descrizione

Azione; aggiorna il video (indipendentemente dal numero di fotogrammi al secondo del filmato) dopo che l'evento clip specificato negli argomenti è stato completato. Questa azione non è elencata nel pannello Azioni di Flash. L'uso dell'azione `updateAfterEvent` con altre di trascinamento che specificano le proprietà `_x` e `_y` durante lo spostamento del mouse non produce lo sfarfallio dello schermo quando si trascinano oggetti.

Letture

Flash 5 o versione successiva.

Vedere anche

“onClipEvent” a pagina 326

_url

Sintassi

nomeistanza._url

Argomenti

nomeistanza Il clip filmato target.

Descrizione

Proprietà (sola lettura); recupera l'URL dal file SWF dal quale è stato scaricato il clip filmato.

Lettore

Flash 4 o versione successiva.

var

Sintassi

```
var nomeVariabile1 [= valore1] [...,nomeVariabileN [=valoreN]];
```

Argomenti

nomeVariabile Il nome della variabile da dichiarare.

valore Il valore che si sta assegnando alla variabile.

Descrizione

Azione; usata per dichiarare variabili locali. Se si dichiarano variabili locali all'interno di una funzione, le variabili vengono definite per la funzione e non sono più valide quando si esce dalla funzione. Se le variabili non sono dichiarate all'interno di un blocco e la lista delle azioni è stata eseguita con un'azione `call`, le variabili sono locali e non sono più valide quando si esce dalla lista corrente. Se le variabili non sono dichiarate all'interno di un blocco e la lista delle azioni corrente non è stata eseguita con un'azione `call`, le variabili non sono locali.

Lettore

Flash 5 o versione successiva.

_visible

Sintassi

nomeistanza._visible

nomeistanza._visible = *booleano*;

Argomenti

boolean Immettere il valore *true* o *false* per specificare se il clip filmato è visibile.

Descrizione

Proprietà; determina se il filmato specificato nell'argomento *nomeistanza* è visibile. I clip filmato non visibili (proprietà impostata su *false*) sono disattivati. Ad esempio, un pulsante in un clip filmato con la proprietà *_visible* impostata su *false* non può essere premuto.

Letture

Flash 4 o versione successiva.

void

Sintassi

```
void (espressione);
```

Argomenti

espressione Un'espressione di qualsiasi valore.

Descrizione

Operatore; un operatore unario che ignora il valore *espressione* e restituisce un valore non definito. L'operatore *void* è spesso usato per valutare un URL per provarne gli effetti secondari senza visualizzare l'espressione risultante nella finestra del browser. L'operatore *void* è inoltre usato in operazioni di confronto con l'operatore *==* per verificare se un valore non è definito.

Letture

Flash 5 o versione successiva.

while

Sintassi

```
while(condizione) {  
  istruzione/i;  
}
```

Argomenti

condizione L'istruzione valutata a ogni esecuzione dell'azione *while*. Se l'istruzione restituisce il valore *true*, viene eseguita l'espressione in *istruzione/i*.

istruzione/i L'espressione da eseguire se la condizione restituisce il valore *true*.

Descrizione

Azione; esegue un'istruzione o una serie di istruzioni ripetutamente in un ciclo fintanto che l'argomento *condizione* restituisce il valore *true*. Alla fine di ogni azione *while*, Flash riavvia il ciclo riverificando la condizione. Se la condizione restituisce *false* o 0, Flash salta alla prima istruzione dopo l'azione *while*.

I cicli sono generalmente usati per eseguire un'azione finché una variabile contatore è minore di un valore specificato. Alla fine di ogni ciclo il contatore viene incrementato fino al raggiungimento della soglia, quando *condizione* non è più *true* e il ciclo termina.

Letttore

Flash 4 o versione successiva.

Esempio

Questo esempio duplica cinque clip filmato sullo stage, ognuno con posizioni *x* e *y*, *xscale* e *yscale* e proprietà *_alpha* generati a caso per realizzare un effetto sparpagliato. La variabile *foo* è inizializzata con il valore 0. L'argomento *condizione* è impostato in modo che il ciclo *while* venga eseguito cinque volte, ossia finché il valore della variabile *foo* è minore di 5. All'interno del ciclo *while*, un clip filmato viene duplicato e *setProperty* usato per modificare le diverse proprietà del clip filmato duplicato. L'ultima istruzione del ciclo incrementa *foo* in modo che quando il valore raggiunge 5, l'argomento *condizione* restituisce il valore *false* e il ciclo non viene più eseguito.

```
on(release) {  
    foo = 0;  
    while(foo < 5) {  
        duplicateMovieClip("/flower", "mc" + foo, foo);  
        setProperty("mc" + foo, _x, random(275));  
        setProperty("mc" + foo, _y, random(275));  
        setProperty("mc" + foo, _alpha, random(275));  
        setProperty("mc" + foo, _xscale, random(200));  
        setProperty("mc" + foo, _yscale, random(200));  
        foo = foo + 1;  
    }  
}
```

Vedere anche

“do... while” a pagina 253

“continue” a pagina 232

_width

Sintassi

nomeistanza._width

nomeistanza._width = *valore*;

Argomenti

valore La larghezza del filmato in pixel.

nomeistanza Il nome dell'istanza di un clip filmato la cui proprietà *_width* deve essere impostata o richiamata.

Descrizione

Proprietà; imposta la larghezza del filmato. Nelle versioni precedenti di Flash, *_height* e *_width* erano proprietà di sola lettura, mentre in Flash 5 è possibile sia impostarle che accederle.

Letttore

Flash 4 come proprietà di sola lettura. In Flash 5 o versione successiva, questa proprietà può essere sia impostata che acceduta.

Esempio

L'esempio di codice seguente imposta le proprietà di altezza e larghezza di un clip filmato quando l'utente fa clic con il mouse.

```
onClipEvent(mouseDown) {  
    _width=200;  
    _height=200;  
}
```

Vedere anche

“_height” a pagina 268

with

x209E6 | IDS_ACTIONHELP_WITH, with, istruzione

Sintassi

```
with (oggetto) {  
    istruzione/i;  
}
```

Argomenti

oggetto Istanza di un oggetto ActionScript o di un clip filmato.

istruzione/i Un'azione o un gruppo di azioni racchiuso tra parentesi graffe.

Descrizione

Azione; modifica temporaneamente l'area di validità (o percorso target) usata per valutare espressioni e azioni nelle *istruzioni*. Dopo l'esecuzione dell'azione *with*, la catena dell'area di validità viene ripristinata allo stato originale.

L'elemento *oggetto* diventa il contesto in cui le proprietà, variabili e funzioni vengono lette. Ad esempio, se *oggetto* è *myArray* e due delle proprietà specificate sono *length* e *concat*, queste proprietà vengono automaticamente lette come *myArray.length* e *myArray.concat*. Come altro esempio, se *oggetto* è *state.california*, è come se qualsiasi azione o istruzione interna all'azione *with* fosse chiamata dall'interno dell'istanza *california*.

Per trovare il valore di un identificatore nelle *istruzioni*, ActionScript comincia all'inizio della catena dell'area di validità specificata da *oggetto* e ricerca l'identificatore a ogni livello della catena dell'area di validità seguendo un ordine specifico.

La catena dell'area di validità usata dall'azione *with* per risolvere gli identificatori inizia con la prima voce della lista e prosegue fino all'ultima, come segue:

- *Oggetto* a cui fa riferimento l'azione *with* più interna
- *Oggetto* a cui fa riferimento l'azione *with* più esterna

- Oggetto di attivazione; un oggetto temporaneo automaticamente creato quando viene chiamata una funzione che contiene le variabili locali chiamate nella funzione
- Clip filmato contenente lo script in esecuzione
- Oggetto globale; oggetti predefiniti come Math o String

In Flash 5 l'azione `with` sostituisce l'azione `tellTarget` diventata obsoleta. Si consiglia di usare `with` invece di `tellTarget` perché si tratta di un'estensione di ActionScript conforme allo standard ECMA-262. La differenza principale tra le azioni `with` e `tellTarget` consiste nel fatto che `with` richiede come argomento il riferimento a un clip filmato o a un altro oggetto, mentre `tellTarget` richiede una stringa di percorso `target` che identifica il clip filmato e non può agire su oggetti.

Per impostare una variabile dentro un'azione `with`, la variabile deve essere stata dichiarata all'esterno dell'azione `with` oppure è necessario definire il percorso completo della linea temporale sulla quale risiede la variabile. Se si imposta una variabile in un'azione `with` senza dichiararla, l'azione `with` ricercherà il valore seguendo la catena dell'area di validità. Se la variabile non esiste ancora, il nuovo valore verrà impostato sulla linea temporale da cui l'azione `with` è stata chiamata.

Esempio

L'esempio seguente imposta le proprietà `x` e `y` dell'istanza `someOtherMovieClip`, quindi ordina a `someOtherMovieClip` di passare al fotogramma 3 e di bloccarsi in tale posizione:

```
with (someOtherMovieClip) {
    _x = 50;
    _y = 100;
    gotoAndStop(3);
}
```

Il frammento di codice seguente rappresenta un'alternativa a quello precedente, senza l'uso dell'azione `with`.

```
someOtherMovieClip._x = 50;
someOtherMovieClip._y = 100;
someOtherMovieClip.gotoAndStop(3);
```

Questo codice può inoltre essere scritto usando l'azione `tellTarget`.

```
tellTarget ("someOtherMovieClip") {
    _x = 50;
    _y = 100;
    gotoAndStop(3);
}
```

L'azione `with` è utile per accedere contemporaneamente a più elementi all'interno di una catena di area di validità. Nell'esempio seguente, l'oggetto incorporato `Math` è posto all'inizio della catena di area di validità. L'impostazione di `Math` come oggetto predefinito risolve gli identificatori `cos`, `sin` e `PI` rispettivamente a `Math.cos`, `Math.sin` e `Math.PI`. Gli identificatori `a`, `x`, `y` e `r` non sono metodi o proprietà dell'oggetto `Math`, ma poiché esistono nell'area di validità dell'oggetto di attivazione per la funzione `polar` vengono risolti come le variabili locali corrispondenti.

```
function polar(r){
  var a, x, y
  with (Math) {
    a = PI * r * r
    x = r * cos(PI)
    y = r * sin(PI/2)
  }
  trace("area = " + a)
  trace("x = " + x)
  trace("y = " + y)
}
```

È possibile usare azioni `with` annidate per accedere a informazioni in più aree di validità. Nell'esempio seguente, l'istanza `fresno` e l'istanza `salinas` sono istanze secondarie dell'istanza `california`. L'istruzione imposta i valori `_alpha` di `fresno` e `salinas` senza modificare il valore `_alpha` di `california`.

```
with (california){
  with (fresno){
    _alpha = 20;
  }
  with (salinas){
    _alpha = 40;
  }
}
```

Vedere anche

“`tellTarget`” a pagina 366

_x

Sintassi

```
nomeistanza._x  
nomeistanza._x = intero
```

Argomenti

intero La coordinata *x* locale del filmato.

nomeistanza Il nome di un'istanza di clip filmato.

Descrizione

Proprietà; imposta la coordinata *x* del filmato rispetto alle coordinate locali del clip filmato principale. Se un clip filmato è nella linea temporale principale, allora il sistema di coordinate considera l'angolo superiore sinistro dello stage come il punto (0, 0). Se il clip filmato è all'interno di un altro clip filmato a cui sono state applicate trasformazioni, il clip si trova nel sistema di coordinate locali del clip che lo contiene. Ad esempio, se un clip filmato è ruotato di 90° in senso antiorario, il clip filmato secondario eredita un sistema di coordinate ruotato di 90° in senso antiorario. Le coordinate del clip filmato si riferiscono alla posizione del punto di registrazione.

Lettore

Flash 3 o versione successiva.

Vedere anche

“_y” a pagina 405
“_xscale” a pagina 404

XML (oggetto)

I metodi e le proprietà dell'oggetto XML consentono di caricare, analizzare sintatticamente, inviare, costruire e manipolare alberi di documenti XML.

Prima di chiamare i metodi dell'oggetto XML è necessario crearne un'istanza usando la funzione di costruzione `new XML()`.

L'oggetto XML è supportato da Flash Player 5 o versioni successive.

Riepilogo dei metodi validi per l'oggetto XML

Metodo	Descrizione
<code>appendChild</code>	Aggiunge un nodo alla fine della lista dei nodi secondari dell'oggetto specificato.
<code>cloneNode</code>	Duplica il nodo specificato e se richiesto duplica ricorsivamente tutti i nodi secondari.
<code>createElement</code>	Crea un nuovo elemento XML.
<code>createTextNode</code>	Crea un nuovo nodo di testo XML.
<code>hasChildNodes</code>	Restituisce <code>true</code> se il nodo specificato possiede nodi secondari; altrimenti restituisce <code>false</code> .
<code>insertBefore</code>	Inserisce un nodo davanti a uno esistente nella lista dei nodi secondari del nodo specificato.
<code>load</code>	Carica un documento (specificato dall'oggetto XML) da un URL.
<code>onLoad</code>	Una funzione di callback per <code>load</code> e <code>sendAndLoad</code> .
<code>parseXML</code>	Analizza sintatticamente un documento XML nell'albero dell'oggetto XML specificato.
<code>removeNode</code>	Rimuove il nodo specificato dal relativo nodo principale.
<code>send</code>	Invia l'oggetto XML specificato a un URL.
<code>sendAndLoad</code>	Invia l'oggetto XML specificato a un URL e carica la risposta del server in un altro oggetto XML.
<code>toString</code>	Converte il nodo specificato e ogni nodo secondario in testo XML.

Riepilogo delle proprietà valide per l'oggetto XML

Proprietà	Descrizione
<code>docTypeDecl</code>	Imposta e restituisce informazioni sulla dichiarazione DOCTYPE di un documento XML.
<code>firstChild</code>	Fa riferimento al primo nodo secondario nella lista per il nodo specificato.
<code>lastChild</code>	Fa riferimento all'ultimo nodo secondario nella lista per il nodo specificato.
<code>loaded</code>	Verifica se l'oggetto XML specificato è stato caricato.
<code>nextSibling</code>	Fa riferimento al nodo successivo nella lista dei nodi secondari del nodo principale.

Proprietà	Descrizione
nodeName	Restituisce il nome del tag di un elemento XML.
nodeType	Restituisce il tipo del nodo specificato (elemento XML o nodo di testo).
nodeValue	Restituisce il testo del nodo specificato se il nodo è un nodo di testo.
parentNode	Fa riferimento al nodo principale del nodo specificato.
previousSibling	Fa riferimento al nodo precedente nella lista dei nodi secondari del nodo principale.
status	Restituisce un codice numerico di stato che indica l'esito dell'operazione di analisi sintattica di un documento XML.
xmlDecl	Imposta e restituisce informazioni sulla dichiarazione del documento di un documento XML.

Riepilogo delle collezioni valide per l'oggetto XML

Metodo	Descrizione
attributes	Restituisce una matrice associativa contenente tutti gli attributi del nodo specificato.
childNodes	Restituisce una matrice contenente riferimenti ai nodi secondari del nodo specificato.

Funzione di costruzione dell'oggetto XML

Sintassi

```
new XML();
new XML(origine);
```

Argomenti

origine Il documento XML analizzato sintatticamente per creare il nuovo oggetto XML.

Descrizione

Funzione di costruzione; crea un nuovo oggetto XML. Prima di chiamare i metodi dell'oggetto XML è necessario crearne un'istanza usando il metodo di costruzione `new XML()`.

La prima sintassi crea un nuovo oggetto XML vuoto.

La seconda sintassi crea un nuovo oggetto XML analizzando sintatticamente il documento XML specificato nell'argomento *origine* e inserisce gli elementi dell'albero del documento XML risultante nell'oggetto XML appena creato.

Nota: i metodi `createElement` e `createTextNode` sono i metodi di ‘costruzione’ per creare gli elementi e i nodi di testo in un albero di documento XML.

Letttore

Flash 5 o versione successiva.

Esempio

L'esempio seguente crea un nuovo oggetto XML vuoto.

```
myXML = new XML();
```

Vedere anche

“XML.createTextNode” a pagina 384

“XML.createElement” a pagina 384

XML.appendChild

Sintassi

```
OggettoXML.appendChild(NodoSecondario);
```

Argomenti

NodoSecondario Il nodo secondario da aggiungere alla lista di nodi secondari dell'oggetto XML specificato.

Descrizione

Metodo; aggiunge il nodo secondario specificato alla lista di nodi secondari dell'oggetto XML. Il nodo secondario aggiunto viene collocato nella struttura ad albero una volta rimosso dal relativo nodo principale eventualmente esistente.

Letttore

Flash 5 o versione successiva.

Esempio

L'esempio seguente duplica l'ultimo nodo di `doc1` e lo aggiunge a `doc2`.

```
doc1 = new XML(src1);  
doc2 = new XML();  
node = doc1.lastChild.cloneNode(true);  
doc2.appendChild(node);
```

XML.attributes

Sintassi

```
OggettoXML.attributes;
```

Argomenti

Nessuno.

Descrizione

Collezione (lettura-scrittura); restituisce una matrice associativa contenente tutti gli attributi dell'oggetto XML specificato.

Letture

Flash 5 o versione successiva.

Esempio

L'esempio seguente visualizza i nomi degli attributi XML nella finestra Output.

```
str = "<mytag name=\"Val\"> item </mytag>";  
doc = new XML(str);  
y = doc.firstChild.attributes.name;  
    trace (y);  
doc.firstChild.attributes.order = "first";  
z = doc.firstChild.attributes.order  
    trace(z);
```

I valori seguenti vengono visualizzati nella finestra Output:

```
Val  
First
```

XML.childNodes

Sintassi

OggettoXML.childNodes;

Argomenti

Nessuno.

Descrizione

Collezione (sola lettura); restituisce una matrice dei nodi secondari dell'oggetto XML specificato. Ogni elemento nella matrice è un riferimento a un oggetto XML rappresentante un nodo secondario. Questa proprietà è di sola lettura e non può essere usata per manipolare nodi secondari. Usare i metodi `appendChild`, `insertBefore` e `removeNode` per manipolare i nodi secondari.

Questa collezione non è definita per i nodi di testo (`nodeType == 3`).

Letture

Flash 5 o versione successiva.

XML.cloneNode

Sintassi

OggettoXML.cloneNode(*profondità*);

Argomenti

profondità Valore booleano che specifica se i nodi secondari dell'oggetto XML specificato vengono ricorsivamente duplicati.

Descrizione

Metodo; costruisce e restituisce un nuovo nodo XML con tipo, nome, valore e attributi uguali a quelli dell'oggetto XML specificato. Se *profondità* è impostato su *true*, tutti i nodi secondari vengono ricorsivamente duplicati; il risultato è una copia esatta dell'albero di documento dell'oggetto originale.

Lettore

Flash 5 o versione successiva.

XML.createElement

Sintassi

```
OggettoXML.createElement(nome);
```

Argomenti

nome Il nome del tag dell'elemento XML creato.

Descrizione

Metodo; crea un nuovo elemento XML usando il nome specificato nell'argomento. In un primo tempo, il nuovo elemento non ha né nodi principali né nodi secondari. Il metodo restituisce un riferimento all'oggetto XML appena creato che rappresenta l'elemento. Questo metodo e `createTextNode` sono i metodi di costruzione per la creazione di nodi per un oggetto XML.

Lettore

Flash 5 o versione successiva.

XML.createTextNode

Sintassi

```
OggettoXML.createTextNode(testo);
```

Argomenti

testo Il testo usato per creare il nuovo nodo di testo.

Descrizione

Metodo; crea un nuovo nodo di testo XML con il testo specificato. In un primo tempo il nuovo nodo non ha nodi principali e, in quanto nodo di testo, non ha nodi secondari. Questo metodo restituisce un riferimento all'oggetto XML che rappresenta il nuovo nodo di testo. Questo metodo e `createElement` sono i metodi di costruzione per la creazione di nodi per un oggetto XML.

Lettore

Flash 5 o versione successiva.

XML.docTypeDecl

Sintassi

*Oggetto*XML.XMLdocTypeDecl;

Argomenti

Nessuno.

Descrizione

Proprietà; imposta e restituisce informazioni sulla dichiarazione DOCTYPE di un documento XML. Dopo la trasformazione del testo XML in oggetto XML tramite l'analisi sintattica, la proprietà `XML.docTypeDecl` dell'oggetto XML viene impostata sul testo della dichiarazione DOCTYPE del documento XML. Ad esempio, `<!DOCTYPE greeting SYSTEM "hello.dtd">`. Questa proprietà viene impostata usando il formato stringa della dichiarazione DOCTYPE, non un oggetto nodo XML.

La funzione di analisi sintattica XML di ActionScript non è una funzione di convalida. La dichiarazione DOCTYPE viene letta dalla funzione di analisi sintattica e memorizzata nella proprietà `docTypeDecl`, ma non viene effettuata alcuna convalida in base al DTD.

Se l'operazione di analisi sintattica non incontra alcuna dichiarazione DOCTYPE, `XML.docTypeDecl` viene impostato su un valore non definito. `XML.toString` genera il contenuto di `XML.docTypeDecl` immediatamente dopo la memorizzazione della dichiarazione XML in `XML.xmlDecl` e prima di qualsiasi altro testo nell'oggetto XML. Se `XML.docTypeDecl` non è definito, non viene generata alcuna dichiarazione DOCTYPE.

Letture

Flash 5 o versione successiva.

Esempio

L'esempio seguente usa `XML.docTypeDecl` per impostare la dichiarazione DOCTYPE per un oggetto XML.

```
OggettoXML.docTypeDecl = "<!DOCTYPE greeting SYSTEM  
\"hello.dtd\">";
```

Vedere anche

“`XML.toString`” a pagina 395

“`XML.xmlDecl`” a pagina 395

XML.firstChild

Sintassi

OggettoXML.firstChild;

Argomenti

Nessuno.

Descrizione

Proprietà (sola lettura); valuta l'oggetto XML specificato e fa riferimento al primo nodo secondario nella lista dei nodi secondari del nodo principale. Questa proprietà è `null` se il nodo non ha nodi secondari. Questa proprietà non è definita se il nodo è un nodo di testo. Questa proprietà è di sola lettura e non può essere usata per manipolare nodi secondari. Usare i metodi `appendChild`, `insertBefore` e `removeNode` per manipolare i nodi secondari.

Lettore

Flash 5 o versione successiva.

Vedere anche

“XML.appendChild” a pagina 382

“XML.insertBefore” a pagina 387

“XML.removeNode” a pagina 393

XML.hasChildNodes

Sintassi

OggettoXML.hasChildNodes();

Argomenti

Nessuno.

Descrizione

Metodo; valuta l'oggetto XML specificato e restituisce `true` se ci sono nodi secondari; altrimenti restituisce `false`.

Lettore

Flash 5 o versione successiva.

Esempio

L'esempio seguente usa le informazioni dall'oggetto XML in una funzione definita dall'utente.

```
if (rootNode.hasChildNodes()) {  
    myfunc (rootNode.firstChild);  
}
```

XML.insertBefore

Sintassi

OggettoXML.insertBefore(NodoSecondario, PrimaDelNodo);

Argomenti

NodoSecondario Il nodo da inserire.

PrimaDelNodo Il nodo prima del punto di inserzione per il *NodoSecondario*.

Descrizione

Metodo; inserisce un nuovo nodo secondario lista di nodi secondari dell'oggetto XML, prima di *PrimaDelNodo*.

Lettore

Flash 5 o versione successiva.

XML.lastChild

Sintassi

OggettoXML.lastChild;

Argomenti

Nessuno.

Descrizione

Proprietà (sola lettura); valuta l'oggetto XML e fa riferimento all'ultimo nodo secondario nella lista dei nodi secondari del nodo principale. Questo metodo restituisce *null* se il nodo non ha nodi secondari. Questa proprietà è di sola lettura e non può essere usata per manipolare nodi secondari. Usare i metodi *appendChild*, *insertBefore* e *removeNode* per manipolare i nodi secondari.

Lettore

Flash 5 o versione successiva.

Vedere anche

“XML.appendChild” a pagina 382

“XML.insertBefore” a pagina 387

“XML.removeNode” a pagina 393

XML.load

Sintassi

OggettoXML.load(url);

Argomenti

url L'URL dove si trova il documento XML da caricare. Tale URL deve trovarsi nello stesso sottodominio dell'URL che contiene il filmato.

Descrizione

Metodo; carica un documento XML dall'URL specificato e sostituisce il contenuto dell'oggetto XML specificato con i dati XML scaricati. Il processo di caricamento è asincrono; non termina subito dopo l'esecuzione del metodo `load`. Quando viene eseguita l'azione `load`, la proprietà `loaded` dell'oggetto XML viene impostata su `false`. Al termine dello scaricamento dei dati XML, la proprietà `loaded` viene impostata su `true` e viene chiamato il metodo `onLoad`. I dati XML non vengono analizzati sintatticamente fino alla fine dello scaricamento. Gli eventuali alberi XML contenuti nell'oggetto XML vengono rimossi.

È possibile specificare le proprie funzioni callback al posto del metodo `onLoad`.

Lettore

Flash 5 o versione successiva.

Esempio

L'esempio seguente illustra un semplice uso di `XML.load`.

```
doc = new XML();  
doc.load ("theFile.xml");
```

Vedere anche

“XML.onLoad” a pagina 390

“XML.loaded” a pagina 388

XML.loaded

Sintassi

*Oggetto*XML.loaded;

Argomenti

Nessuno.

Descrizione

Proprietà (sola lettura); indica se il processo di caricamento del documento avviato dalla chiamata `XML.load` è completato. Se il processo è completato, il metodo restituisce `true`; altrimenti restituisce `false`.

Lettore

Flash 5 o versione successiva.

Esempio

L'esempio seguente illustra l'uso di `XML.loaded` in un semplice script.

```
if (doc.loaded) {  
    gotoAndPlay(4)  
}
```

XML.nextSibling

Sintassi

OggettoXML.nextSibling;

Argomenti

Nessuno.

Descrizione

Proprietà (sola lettura); valuta l'oggetto XML e fa riferimento al nodo successivo nella lista dei nodi secondari del nodo principale. Questo metodo restituisce `null` se il nodo non ha un nodo successivo allo stesso livello. Questa proprietà è di sola lettura e non può essere usata per manipolare nodi secondari. Usare i metodi `appendChild`, `insertBefore` e `removeNode` per manipolare i nodi secondari.

Letttore

Flash 5 o versione successiva.

Vedere anche

“XML.appendChild” a pagina 382

“XML.insertBefore” a pagina 387

“XML.removeNode” a pagina 393

XML.nodeName

Sintassi

OggettoXML.nodeName;

Argomenti

Nessuno.

Descrizione

Proprietà; estrae o restituisce il nome del nodo dell'oggetto XML. Se l'oggetto XML è un elemento XML (`nodeType == 1`), `nodeName` è il nome del tag rappresentante il nodo nel file XML. Ad esempio, `TITLE` è il `nodeName` di un tag `HTML TITLE`. Se l'oggetto XML è un nodo di testo (`nodeType == 3`), `nodeName` è `null`.

Letttore

Flash 5 o versione successiva.

Vedere anche

“XML.nodeType” a pagina 390

XML.nodeType

Sintassi

OggettoXML.nodeType;

Argomenti

Nessuno.

Descrizione

Proprietà (sola lettura); estrae o restituisce un valore `nodeType`, dove 1 rappresenta un elemento XML e 3 un nodo di testo.

Lettore

Flash 5 o versione successiva.

Vedere anche

“XML.nodeValue” a pagina 390

XML.nodeValue

Sintassi

OggettoXML.nodeValue;

Argomenti

Nessuno.

Descrizione

Proprietà; restituisce il valore del nodo dell'oggetto XML. Se l'oggetto XML è un nodo di testo, `nodeType` è 3, `nodeValue` è il testo del nodo. Se l'oggetto XML è un elemento XML, `nodeValue` contiene il valore `null` ed è di sola lettura.

Lettore

Flash 5 o versione successiva.

Vedere anche

“XML.nodeType” a pagina 390

XML.onLoad

Sintassi

OggettoXML.onLoad(*successo*);

Argomenti

successo Valore booleano che indica se l'oggetto XML è stato caricato correttamente tramite l'azione `XML.load` o `XML.sendAndLoad`.

Descrizione

Metodo; chiamato da Flash Player quando un documento XML viene ricevuto dal server. Se il documento XML è ricevuto correttamente, l'argomento *successo* è *true*. Se il documento non è stato ricevuto o se si è verificato un errore nel ricevimento della risposta dal server, l'argomento *successo* è *false*.

L'implementazione predefinita di questo metodo non implica alcuna azione. Per ridefinire tale implementazione, è necessario assegnare una funzione contenente le proprie azioni.

Lettores

Flash 5 o versione successiva.

Esempio

L'esempio seguente crea un semplice filmato Flash per una semplice applicazione di vetrina e-commerce. Il metodo `sendAndLoad` viene usato per passare un elemento XML contenente il nome e la password dell'utente e il gestore `onLoad` per gestire la risposta dal server.

```
var myLoginReply = new XML();
myLoginReply.onLoad = myOnLoad;
myXML.sendAndLoad("http://www.samplestore.com/login.cgi",
                  myLoginReply);
function myOnLoad(success) {
    if (success) {
        if (e.firstChild.nodeName == "LOGINREPLY" &&
            e.firstChild.attributes.status == "OK") {
            gotoAndPlay("loggedIn")
        } else {
            gotoAndStop("loginFailed")
        }
    } else {
        gotoAndStop("connectionFailed")
    }
}
```

Vedere anche

“function” a pagina 262

“XML.load” a pagina 387

“XML.sendAndLoad” a pagina 393

XML.parentNode

Sintassi

OggettoXML.parentNode;

Argomenti

Nessuno.

Descrizione

Proprietà (sola lettura); fa riferimento al nodo principale dell'oggetto XML specificato o restituisce il valore `null` se il nodo non ha nodo principale. Questa proprietà è di sola lettura e non può essere usata per manipolare nodi secondari. Usare i metodi `appendChild`, `insertBefore` e `removeNode` per manipolare i nodi secondari.

Lettore

Flash 5 o versione successiva.

XML.parseXML

Sintassi

```
OggettoXML.parseXML(origine);
```

Argomenti

origine Il testo XML di cui eseguire l'analisi sintattica e da passare all'oggetto XML specificato.

Descrizione

Metodo; esegue l'analisi sintattica del testo XML specificato nell'argomento *origine*, quindi inserisce gli elementi dell'albero XML risultante nell'oggetto XML specificato. Gli eventuali alberi XML contenuti nell'oggetto XML vengono rimossi.

Lettore

Flash 5 o versione successiva.

XML.previousSibling

Sintassi

```
OggettoXML.previousSibling;
```

Argomenti**Descrizione**

Proprietà (sola lettura); valuta l'oggetto XML e fa riferimento al nodo precedente nella lista dei nodi secondari del nodo principale. Restituisce `null` se il nodo non ha un nodo precedente allo stesso livello. Questa proprietà è di sola lettura e non può essere usata per manipolare nodi secondari. Usare i metodi `appendChild`, `insertBefore` e `removeNode` per manipolare i nodi secondari.

Lettore

Flash 5 o versione successiva.

XML.removeNode

Sintassi

OggettoXML.removeNode();

Argomenti

Nessuno.

Descrizione

Metodo; rimuove l'oggetto XML specificato dal relativo nodo principale.

Lettore

Flash 5 o versione successiva.

XML.send

Sintassi

OggettoXML.send(url);

OggettoXML.send(url, finestra);

Argomenti

url L'URL di destinazione dell'oggetto XML specificato.

finestra La finestra del browser in cui visualizzare i dati restituiti dal server: *_self* indica il frame corrente nella finestra corrente, *_blank* indica una finestra nuova *_parent* indica il frame principale rispetto a quello corrente, *_top* indica il frame di primo livello nella finestra corrente.

Descrizione

Metodo; codifica l'oggetto XML specificato e lo trasforma in un documento XML, per poi inviarlo all'URL specificato tramite il metodo POST.

Lettore

Flash 5 o versione successiva.

XML.sendAndLoad

Sintassi

OggettoXML.sendAndLoad(url, oggettoXMLtarget);

Argomenti

url L'URL di destinazione dell'oggetto XML specificato. L'URL deve trovarsi nello stesso sottodominio dell'URL dal quale è stato scaricato il filmato.

oggettoXMLtarget Oggetto XML creato con la relativa funzione di costruzione che riceverà le informazioni restituite dal server.

Descrizione

Metodo; codifica l'oggetto XML specificato e lo trasforma in un documento XML, lo invia all'URL specificato tramite il metodo `POST`, scarica la risposta del server e quindi la carica nell'oggetto *oggettoXMLtarget* specificato negli argomenti. La procedura per il caricamento della risposta del server è analoga a quella usata dal metodo `load`.

Lettores

Flash 5 o versione successiva.

Vedere anche

"XML.load" a pagina 387

XML.status

Sintassi

OggettoXML.status;

Argomenti

Nessuno.

Descrizione

Proprietà; imposta e restituisce automaticamente un valore numerico che indica se l'analisi sintattica del documento XML e la sua trasformazione in oggetto XML sono state eseguite correttamente. La seguente lista contiene i codici numerici di stato con la relativa descrizione.

- 0 Nessun errore; l'analisi sintattica è stata completata correttamente.
- -2 Una sezione di CDATA non è stata terminata correttamente.
- -3 La dichiarazione XML non è stata terminata correttamente.
- -4 La dichiarazione DOCTYPE non è stata terminata correttamente.
- -5 Un commento non è stato terminato correttamente.
- -6 È presente un elemento XML non strutturalmente corretto.
- -7 Memoria esaurita.
- -8 Un valore attributo non è stato terminato correttamente.
- -9 È presente un tag di apertura senza tag di chiusura corrispondente.
- -10 È presente un tag di chiusura senza tag di apertura corrispondente.

Lettores

Flash 5 o versione successiva.

XML.toString

Sintassi

OggettoXML.toString();

Argomenti

Nessuno.

Descrizione

Metodo; valuta l'oggetto XML specificato, costruisce una rappresentazione testuale della struttura XML, includendo in essa il nodo, i nodi secondari e gli attributi, quindi restituisce il risultato in formato stringa.

Nel caso di oggetti XML di primo livello, ossia quelli creati con la funzione di costruzione, l'output del metodo `XML.toString` sarà costituito dalla dichiarazione XML del documento (memorizzata nella proprietà `XML.xmlDecl`), seguita dalla dichiarazione `DOCTYPE` del documento (memorizzata nella proprietà `XML.docTypeDecl`), quindi dalla rappresentazione testuale di tutti i nodi XML nell'oggetto. La dichiarazione XML non viene inclusa nell'output se la proprietà `XML.xmlDecl` non è definita. La dichiarazione `DOCTYPE` non viene inclusa nell'output se la proprietà `XML.docTypeDecl` non è definita.

Lettore

Flash 5 o versione successiva.

Esempio

Il codice seguente illustra l'uso del metodo `XML.toString`:

```
node = new XML("<h1>test</h1>");
trace(node.toString());
invia
<H1>test</H1>
alla finestra Output
```

Vedere anche

“`XML.xmlDecl`” a pagina 395

“`XML.docTypeDecl`” a pagina 385

XML.xmlDecl

Sintassi

OggettoXML.xmlDecl;

Argomenti

Nessuno.

Descrizione

Proprietà; imposta e restituisce informazioni sulla dichiarazione XML di un documento. Dopo aver eseguito l'analisi sintattica di un documento XML e averlo trasformato in oggetto XML, questa proprietà viene impostata usando il testo della dichiarazione XML del documento. Questa proprietà viene impostata usando il formato stringa della dichiarazione XML, non un oggetto nodo XML. Se in fase di analisi sintattica non viene rilevata alcuna dichiarazione XML, questa proprietà viene impostata su un valore indefinito. Il metodo `XML.toString` produce come output il contenuto della proprietà `XML.xmlDecl` prima di qualsiasi altro testo nell'oggetto XML. Se `XML.xmlDecl` contiene il tipo `undefined`, l'output non conterrà la dichiarazione XML.

Lettore

Flash 5 o versione successiva.

Esempio

L'esempio seguente usa la proprietà `XML.xmlDecl` per impostare la dichiarazione XML del documento per un oggetto XML.

```
OggettoXML.xmlDecl = "<?xml version=\"1.0\" ?>";
```

Vedere anche

`"XML.toString"` a pagina 395

`"XML.docTypeDecl"` a pagina 385

XMLSocket (oggetto)

L'oggetto `XMLSocket` implementa dei socket lato client che consentono al computer che esegue Flash Player di comunicare con un computer server identificato da un indirizzo IP o da un nome di dominio.

Uso dell'oggetto XMLSocket

Per usare l'oggetto `XMLSocket`, è necessario che sul computer server sia in esecuzione un daemon compatibile con il protocollo usato dall'oggetto stesso. Tale protocollo è il seguente:

- I messaggi XML vengono inviati attraverso una connessione socket in streaming TCP/IP full duplex.
- Ciascun messaggio XML è un documento XML completo, che termina con un byte a zero.
- È possibile inviare e ricevere un numero illimitato di messaggi XML attraverso una singola connessione `XMLSocket`.

L'oggetto `XMLSocket` è utile per le applicazioni client server che richiedono un intervallo di latenza basso, quali i sistemi di conversazione in tempo reale. In genere il funzionamento dei sistemi di conversazione tradizionali basati sul protocollo HTTP consiste nell'interrogare il server e quindi scaricare i nuovi messaggi tramite una richiesta HTTP. Il sistema di conversazione `XMLSocket`, al contrario, mantiene la connessione al server aperta in modo da consentire a quest'ultimo di inviare subito i messaggi in arrivo senza aspettare la richiesta da parte del client.

La configurazione di un server che comunichi con l'oggetto `XMLSocket` può risultare impegnativa. Se l'applicazione non richiede un'interazione in tempo reale, usare l'azione `loadVariables` o i metodi di connettività al server dell'oggetto XML basati sul protocollo HTTP disponibili in Flash (`XML.load`, `XML.sendAndLoad`, `XML.send`), anziché i metodi dell'oggetto `XMLSocket`.

Per poter usare i metodi dell'oggetto `XMLSocket`, è necessario innanzitutto creare un nuovo oggetto `XMLSocket` tramite la funzione di costruzione `new XMLSocket`.

Oggetto `XMLSocket` e sicurezza

Poiché l'oggetto `XMLSocket` stabilisce una connessione al server e la mantiene aperta, sono state definite restrizioni specifiche per ragioni di sicurezza.

- Il metodo `XMLSocket.connect` consente di effettuare la connessione solo ai numeri di porta TCP maggiori o uguali a 1024. Ne consegue, fra l'altro, che ai daemon server che comunicano con l'oggetto `XMLSocket` deve essere assegnato un numero di porta maggiore o uguale a 1024. I numeri di porta minori vengono spesso usati dai servizi di sistema quali FTP, Telnet e HTTP e questa restrizione esclude l'oggetto `XMLSocket` da tali porte. In tal modo viene ridotta la possibilità di accesso non autorizzato e di abuso di tali risorse.
- Il metodo `XMLSocket.connect` consente di effettuare la connessione solo ai computer all'interno dello stesso sottodominio in cui risiede il file SWF (il filmato). Questa restrizione non si applica ai filmati in esecuzione da un disco locale e corrisponde esattamente alle regole di sicurezza relative all'azione `loadVariables` e ai metodi `XML.sendAndLoad` e `XML.load`.

Riepilogo dei metodi validi per l'oggetto XMLSocket

Metodo	Descrizione
<code>close</code>	Chiude una connessione socket aperta.
<code>connect</code>	Stabilisce una connessione al server specificato.
<code>onClose</code>	Funzione di callback invocata quando viene chiusa una connessione XMLSocket.
<code>onConnect</code>	Funzione di callback invocata quando viene stabilita una connessione XMLSocket.
<code>onXML</code>	Funzione di callback invocata quando arriva dal server un oggetto XML.
<code>send</code>	Invia un oggetto XML al server.

Funzione di costruzione per l'oggetto XMLSocket

Sintassi

```
new XMLSocket();
```

Argomenti

Nessuno.

Descrizione

Funzione di costruzione; crea un nuovo oggetto XMLSocket. L'oggetto XMLSocket inizialmente non è collegato a un server. Ciò avviene all'atto della chiamata del metodo `XMLSocket.connect`.

Lettore

Flash 5 o versione successiva.

Esempio

```
myXMLSocket = new XMLSocket();
```

Vedere anche

“XMLSocket.connect” a pagina 399

XMLSocket.close

Sintassi

OggettoXMLSocket.close();

Argomenti

Nessuno.

Descrizione

Metodo; chiude la connessione specificata dall'oggetto XMLSocket.

Lettores

Flash 5 o versione successiva.

Vedere anche

“XMLSocket.connect” a pagina 399

XMLSocket.connect

Sintassi

OggettoXMLSocket.connect(host, porta);

Argomenti

host Il nome di dominio DNS completo o l'indirizzo IP nel formato *aaa.bbb.ccc.ddd*. È inoltre possibile specificare la parola chiave `null` per stabilire la connessione al server host su cui risiede il filmato.

porta Il numero di porta TCP sull'host tramite la quale si stabilisce la connessione. Il numero di porta deve essere maggiore o uguale a 1024.

Descrizione

Metodo; stabilisce una connessione all'host Internet indicato tramite la porta TCP specificata (numero maggiore o uguale a 1024), quindi restituisce `true` o `false` a seconda del risultato del tentativo di connessione. Se non si conosce il numero di porta del computer host per Internet, rivolgersi all'amministratore di rete. Se è in uso il plug-in Netscape di Flash o il controllo ActiveX, il sottodominio dell'host specificato nel relativo argomento e quello dell'host da cui si è scaricato il filmato devono corrispondere.

Se si specifica la parola chiave `null` come valore dell'argomento *host*, verrà contattato l'host su cui risiede il filmato che chiama il metodo `XMLSocket.connect`. Se, ad esempio, il filmato è stato scaricato dal sito `http://www.sito.com` e si è specificato il valore `null` per l'argomento *host*, ciò equivale ad aver immesso l'indirizzo IP del sito `www.sito.com`.

Se il metodo `XMLSocket.connect` restituisce il valore `true`, significa che la fase iniziale del processo di connessione è stata completata correttamente. In un secondo momento, viene invocato il metodo `XMLSocket.onConnect` al fine di determinare se la connessione finale è stata stabilita correttamente o se non è riuscita. Se il metodo `XMLSocket.connect` restituisce il valore `false`, significa che non è stato possibile stabilire una connessione.

Letture

Flash 5 o versione successiva.

Esempio

L'esempio seguente usa il metodo `XMLSocket.connect` per effettuare una connessione all'host su cui risiede il filmato, quindi usa l'azione `trace` per visualizzare il valore restituito che indica se la connessione è stata stabilita correttamente.

```
function myOnConnect(success) {
    if (success) {
        trace ("Connection succeeded!")
    } else {
        trace ("Connection failed!")
    }
}
socket = new XMLSocket()
socket.onConnect = myOnConnect
if (!socket.connect(null, 2000)) {
    trace ("Connection failed!")
}
```

Vedere anche

“function” a pagina 262

“XMLSocket.onConnect” a pagina 401

XMLSocket.onClose

Sintassi

OggettoXMLSocket.onClose();

Argomenti

Nessuno.

Descrizione

Metodo; funzione di callback invocata solo quando una connessione aperta viene chiusa dal server. L'implementazione predefinita di questo metodo non implica l'esecuzione di azioni. Per ridefinire tale implementazione, è necessario assegnare una funzione contenente le proprie azioni.

Letture

Flash 5 o versione successiva.

Vedere anche

“function” a pagina 262

“XMLSocket.onConnect” a pagina 401

XMLSocket.onConnect

Sintassi

OggettoXMLSocket.onConnect(successo);

Argomenti

successo Valore booleano che indica se la connessione socket è stata stabilita correttamente (true o false).

Descrizione

Metodo; una funzione di callback invocata da Flash Player quando una richiesta di connessione effettuata tramite il metodo `XMLSocket.connect` è stata stabilita o non è riuscita. Se la connessione è stata stabilita, il valore dell'argomento *successo* è true; se non è riuscita, il valore dell'argomento *successo* è false.

L'implementazione predefinita di questo metodo non implica l'esecuzione di azioni. Per ridefinire tale implementazione, è necessario assegnare una funzione contenente le proprie azioni.

Lettore

Flash 5 o versione successiva.

Esempio

Nell'esempio seguente viene specificata una funzione che sostituisce il metodo `onConnect` in una semplice applicazione di conversazione in linea.

La funzione controlla la finestra alla quale avrà accesso l'utente, a seconda che la connessione sia stata stabilita correttamente o meno. Se la connessione è stata stabilita correttamente, verrà visualizzata la finestra principale di conversazione nel fotogramma etichettato `startChat`. Se la connessione non è riuscita, verrà visualizzata la finestra contenente le informazioni per la risoluzione dei problemi nel fotogramma etichettato `connectionFailed`.

```
function myOnConnect(success) {  
    if (success) {  
        gotoAndPlay("startChat")  
    } else {  
        gotoAndStop("connectionFailed")  
    }  
}
```

Dopo aver creato un oggetto `XMLSocket` tramite la funzione di costruzione, lo script inserisce il metodo `onConnect` tramite l'operatore di assegnazione:

```
socket = new XMLSocket()  
socket.onConnect = myOnConnect
```

Infine, viene avviato il processo di connessione. Se `connect` restituisce `false`, il filmato viene inviato direttamente al fotogramma etichettato `connectionFailed` e il metodo `onConnect` non viene invocato. Se `connect` restituisce `true`, il filmato passa al fotogramma etichettato `waitForConnection`, ossia a una finestra di attesa. Il filmato rimane nel fotogramma etichettato `waitForConnection` fino a quando viene invocato il gestore `onConnect` in un arco di tempo che dipende dalla latenza di rete.

```
if (!socket.connect(null, 2000)) {  
    gotoAndStop("connectionFailed")  
} else {  
    gotoAndStop("waitForConnection")  
}
```

Vedere anche

“XMLSocket.connect” a pagina 399

“function” a pagina 262

XMLSocket.onXML

Sintassi

OggettoXMLSocket.onXML(oggetto);

Argomenti

oggetto Un'istanza dell'oggetto XML che contiene un documento XML di cui è stata eseguita l'analisi sintattica, proveniente da un server.

Descrizione

Metodo; una funzione di callback invocata da Flash Player quando l'oggetto XML specificato che contiene un documento XML viene ricevuto tramite una connessione XMLSocket aperta. La connessione XMLSocket consente di trasferire tra il client e il server un numero illimitato di documenti XML. Tutti i documenti terminano con un byte a zero. Nel momento in cui riceve il byte a zero, Flash Player esegue l'analisi sintattica di tutti i documenti XML ricevuti nell'intervallo trascorso dall'ultimo byte a zero fino a quel momento oppure, se si tratta del primo messaggio ricevuto, dal momento in cui è stata stabilita la connessione. Ogni blocco di documenti XML sottoposto ad analisi sintattica viene gestito come un singolo documento XML e viene passato al metodo `onXML`.

L'implementazione predefinita di questo metodo non implica l'esecuzione di azioni. Per ridefinire tale implementazione, è necessario assegnare una funzione contenente le proprie azioni.

Lettore

Flash 5 o versione successiva.

Esempio

La funzione seguente ridefinisce l'implementazione predefinita del metodo `onXML` in una semplice applicazione di conversazione in linea. La funzione `myOnXML` indica all'applicazione di conversazione in linea come riconoscere un singolo elemento XML, MESSAGE, nel formato seguente:

```
<MESSAGE USER="John" TEXT="Hello, my name is John!" />.
```

È necessario aver inserito il gestore `onXML` nell'oggetto `XMLSocket` nel modo seguente:

```
socket.onXML = myOnXML;
```

La funzione `displayMessage` rappresenta una funzione definita dall'utente per visualizzare il messaggio.

```
function myOnXML(doc) {  
    var e = doc.firstChild;  
    if (e != null && e.nodeName == "MESSAGE") {  
        displayMessage(e.attributes.user, e.attributes.text);  
    }  
}
```

Vedere anche

“function” a pagina 262

XMLSocket.close

Sintassi

```
OggettoXMLSocket.send(oggetto);
```

Argomenti

oggetto Oggetto XML o altri dati da trasmettere al server.

Descrizione

Metodo; converte l'oggetto XML o i dati specificati nell'argomento *oggetto* in una stringa che viene quindi trasmessa al server, seguita da un byte a zero. Se l'argomento *oggetto* contiene un oggetto XML, la stringa sarà la rappresentazione testuale XML dell'oggetto stesso. Questa operazione è asincrona, ossia, anche se restituisce subito un valore, i dati potrebbero essere trasmessi in un secondo momento. Il metodo `XMLSocket.send` non restituisce un valore che indica se i dati sono stati trasmessi correttamente.

Se l'oggetto *OggettoXMLSocket* non è collegato al server (tramite il metodo `XMLSocket.connect`), il metodo `XMLSocket.send` avrà esito negativo.

Lettore

Flash 5 o versione successiva.

Esempio

L'esempio seguente illustra il codice per specificare un nome utente e una password per inviare l'oggetto XML `myXML` al server.

```
var myXML = new XML();
var myLogin = myXML.createElement("login");
myLogin.attributes.username = usernameTextField;
myLogin.attributes.password = passwordTextField;
myXML.appendChild(myLogin);
myXMLSocket.send(myXML);
```

Vedere anche

“XMLSocket.connect” a pagina 399

_xmouse

Sintassi

nomeistanza.*_xmouse*

Argomenti

nomeistanza Il nome di un'istanza di clip filmato.

Descrizione

Proprietà (sola lettura); restituisce la coordinata *x* della posizione del mouse.

Letture

Flash 5 o versione successiva.

Vedere anche

“Mouse (oggetto)” a pagina 300

“_ymouse” a pagina 406

_xscale

Sintassi

nomeistanza.*_xscale*

nomeistanza.*_xscale* = *percentuale*;

Argomenti

percentuale Un valore che indica la percentuale per il ridimensionamento in orizzontale del filmato. Il valore predefinito è 100.

nomeistanza Il nome di un'istanza di clip filmato.

Descrizione

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

Il ridimensionamento del sistema di coordinate locale influisce sull'impostazione delle proprietà `_x` e `_y` che sono definite in pixel interi. Se, ad esempio, il clip filmato principale è ridimensionato al 50%, l'impostazione della proprietà `_x` sposta un oggetto nel clip filmato di un numero di pixel pari alla metà rispetto al numero corrispettivo per il filmato al 100%.

Lettore

Flash 4 o versione successiva.

Vedere anche

`"_xscale"` a pagina 404

`_y`

Sintassi

```
nomeistanza._y
```

```
nomeistanza._y = intero;
```

Argomenti

intero La coordinata *y* locale del clip filmato.

nomeistanza Il nome di un'istanza di clip filmato.

Descrizione

Proprietà; imposta la coordinata *y* del filmato rispetto alle coordinate locali del clip filmato principale. Se un clip filmato è nella linea temporale principale, allora il sistema di coordinate considera l'angolo superiore sinistro dello stage come il punto (0, 0). Se il clip filmato è all'interno di un altro clip filmato a cui sono state applicate trasformazioni, il clip si trova nel sistema di coordinate locali del clip che lo contiene. Ad esempio, se un clip filmato è ruotato di 90° in senso antiorario, il clip filmato secondario eredita un sistema di coordinate ruotato di 90° in senso antiorario. Le coordinate del clip filmato si riferiscono alla posizione del punto di registrazione.

Lettore

Flash 3 o versione successiva.

Vedere anche

`"yscale"` a pagina 406

_ymouse

Sintassi

nomeistanza._ymouse

Argomenti

nomeistanza Il nome di un'istanza di clip filmato.

Descrizione

Proprietà (sola lettura); restituisce la coordinata *y* della posizione del mouse.

Lettore

Flash 5 o versione successiva.

Vedere anche

“Mouse (oggetto)” a pagina 300

“_xmouse” a pagina 404

_yscale

Sintassi

nomeistanza._yscale

nomeistanza._yscale = percentuale;

Argomenti

percentuale Un valore che indica la percentuale per il ridimensionamento in verticale del filmato. Il valore predefinito è 100.

nomeistanza Il nome di un'istanza di clip filmato.

Descrizione

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

Il ridimensionamento del sistema di coordinate locale influisce sull'impostazione delle proprietà *_x* e *_y* che sono definite in pixel interi. Se, ad esempio, il clip filmato principale è ridimensionato al 50%, l'impostazione della proprietà *_x* sposta un oggetto nel clip filmato di un numero di pixel pari alla metà rispetto al numero corrispettivo per il filmato al 100%.

Lettore

Flash 4 o versione successiva.

Vedere anche

“_x” a pagina 379

“_y” a pagina 405

APPENDICE A

Precedenza e associatività degli operatori

Lista operatori

Questa tabella contiene la lista di tutti gli operatori di ActionScript in ordine di precedenza, dalla più alta alla più bassa, e ne specifica l'associatività.

Operatore	Descrizione	Associatività
Precedenza più alta		
+	Più unario	Da destra a sinistra
-	Meno unario	Da destra a sinistra
~	Complemento a uno bit a bit	Da destra a sinistra
!	NOT logico	Da destra a sinistra
not	NOT logico (stile Flash 4)	Da destra a sinistra
++	Incremento dopo l'operazione	Da sinistra a destra
--	Decremento dopo l'operazione	Da sinistra a destra
()	Chiamata di funzione	Da sinistra a destra
[]	Accesso all'elemento di una matrice	Da sinistra a destra
.	Accesso al membro di una struttura	Da sinistra a destra
++	Incremento prima dell'operazione	Da destra a sinistra
--	Decremento prima dell'operazione	Da destra a sinistra
new	Allocazione di un oggetto	Da destra a sinistra

Operatore	Descrizione	Associatività
delete	Disallocazione di un oggetto	Da destra a sinistra
typeof	Tipo do un oggetto	Da destra a sinistra
void	Restituzione di un valore non definito	Da destra a sinistra
*	Moltiplicazione	Da sinistra a destra
/	Divisione	Da sinistra a destra
%	Modulo	Da sinistra a destra
+	Addizione	Da sinistra a destra
add	Concatenazione di stringhe (precedentemente &)	Da sinistra a destra
-	Simbolo di sottrazione	Da sinistra a destra
<<	Spostamento a sinistra bit a bit	Da sinistra a destra
>>	Spostamento a destra bit a bit	Da sinistra a destra
>>>	Spostamento a destra bit a bit (senza segno)	Da sinistra a destra
<	Minore di	Da sinistra a destra
<=	Minore o uguale a	Da sinistra a destra
>	Maggiore di	Da sinistra a destra
>=	Maggiore o uguale a	Da sinistra a destra
lt	Minore o uguale a (versione per stringhe)	Da sinistra a destra
le	Minore o uguale a (versione per stringhe)	Da sinistra a destra
gt	Maggiore o uguale a (versione per stringhe)	Da sinistra a destra
ge	Maggiore o uguale a (versione per stringhe)	Da sinistra a destra
==	Uguale	Da sinistra a destra
!=	Non uguale	Da sinistra a destra
eq	Uguale (versione per stringhe)	Da sinistra a destra
ne	Non uguale (versione per stringhe)	Da sinistra a destra
&	AND bit a bit	Da sinistra a destra
^	XOR bit a bit	Da sinistra a destra
	OR bit a bit	Da sinistra a destra
&&	AND logico	Da sinistra a destra

Operatore	Descrizione	Associatività
and	AND logico (stile Flash 4)	Da sinistra a destra
	OR logico	Da sinistra a destra
oppure	OR logico (stile Flash 4)	Da sinistra a destra
?:	Condizionale	Da destra a sinistra
=	Assegnazione	Da destra a sinistra
*=, /=, %=, +=, -=, &=, =, ^=, <<=, >>=, >>="	Assegnazioni composte	Da destra a sinistra
,	Valutazioni multiple	Da sinistra a destra
Precedenza più bassa		

APPENDICE B

Tasti della tastiera e valori dei codici tasto

Le seguenti tabelle contengono la lista di tutti i tasti di una tastiera standard e i corrispondenti valori dei codici tasto usati per identificarli in `ActionScript`. Per ulteriori informazioni, vedere la descrizione dell'oggetto `Key` nel capitolo 7 “Dizionario di `ActionScript`”.

Lettere dalla A alla Z e numeri standard da 0 a 9.

Tasto della lettera o del numero	Codice del tasto
A	65
B	66
C	67
D	68
E	69
F	70
G	71
H	72
I	73
J	74
K	75
L	76
M	77
N	78
O	79
P	80
Q	81
R	82
S	83
T	84
U	85
V	86
W	87
X	88
Y	89
Z	90

Tasto della lettera o del numero	Codice del tasto
0	48
1	49
2	50
3	51
4	52
5	53
6	54
7	55
8	56
9	57

Tasti sul tastierino numerico

Tasto del tastierino numerico	Codice del tasto
0 sul tastierino numerico	96
1 sul tastierino numerico	97
2 sul tastierino numerico	98
3 sul tastierino numerico	99
4 sul tastierino numerico	100
5 sul tastierino numerico	101
6 sul tastierino numerico	102
7 sul tastierino numerico	103
8 sul tastierino numerico	104
9 sul tastierino numerico	105
Simbolo di moltiplicazione	106
Simbolo di addizione	107
Invio	108
Simbolo di sottrazione	109
Separatore decimale	110
Simbolo di divisione	111

Tasti funzione

Tasto funzione	Codice del tasto
F1	112
F2	113
F3	114
F4	115
F5	116
F6	117
F7	118

Tasto funzione	Codice del tasto
F8	119
F9	120
F10	121
F11	122
F12	123

Altri tasti

Tasto	Codice del tasto
Backspace	8
Tab	9
Canc	12
Invio	13
Maiusc	16
Ctrl	17
Alt	18
Bloc Maiusc	20
Esc	27
Barra spaziatrice	32
Pag Su	33
Pag Giù	34
Fine	35
Home	36
Freccia sinistra	37
Freccia su	38
Freccia destra	39
Freccia giù	40
Ins	45
Canc	46
Guida	47
Bloc Num	144
;;	186
= +	187
` _	189
/ ?	191
` ~	192

Tasto	Codice del tasto
[{	219
\	220
}]	221
“ ”	222

APPENDICE C

Messaggi di errore

.....

La tabella seguente contiene la lista dei messaggi di errore restituiti dal compilatore di Flash. Per ogni messaggio viene fornita una spiegazione per facilitare la risoluzione dei problemi relativi ai file dei filmati.

Messaggio di errore	Descrizione
Proprietà <proprietà> inesistente	È stata rilevata una proprietà che non esiste. Ad esempio, <code>x = _green</code> non è valido perché non esiste la proprietà <code>_green</code> .
L'operatore <operatore> deve essere seguito da un operando	È stato rilevato un operatore senza operando. Ad esempio, <code>x = 1 +</code> richiede un operando dopo l'operatore <code>+</code> . Un operatore è seguito da un operando non valido. Ad esempio, <code>trace(1+)</code> ; è sintatticamente errato.
Errore di sintassi	Questo messaggio viene visualizzato ogni volta che viene rilevato un errore di sintassi non specifico.
Nome di campo dopo l'operatore <code>'.'</code> non presente	È necessario specificare un nome di campo valido quando si usa la sintassi <i>object.field</i> .
<token>non presente	È stato rilevato un token non valido o imprevisto. Ad esempio, nella sintassi seguente il token <code>foo</code> non è valido. Il token mancante è <code>while</code> . <pre>do { trace (i) } foo (i < 100)</pre>
La lista dell'inizializzatore deve essere terminata con <terminatore>	A una lista di inizializzazione di un oggetto o di una matrice manca la parentesi <code>]</code> o <code>}</code> di chiusura.

Messaggio di errore	Descrizione
Identificatore non presente	È stato rilevato un token imprevisto al posto di un identificatore. Nell'esempio seguente 3 non è un identificatore valido. <code>var 3 = 4;</code>
Il costrutto '<costrutto>' JavaScript non è supportato	È stato rilevato un costrutto JavaScript non supportato da ActionScript. Questo messaggio viene visualizzato se si usa uno dei seguenti costrutti JavaScript: <code>void</code> , <code>switch</code> , <code>try</code> , <code>catch</code> o <code>throw</code> .
Il lato sinistro dell'operatore di assegnazione deve essere una variabile o una proprietà	È stato usato un operatore di assegnazione, ma il lato sinistro dell'assegnazione non è una variabile o una proprietà valida.
Il blocco di istruzioni deve terminare con '}'	È stato dichiarato un gruppo di istruzioni all'interno di parentesi graffe, ma manca la parentesi di chiusura.
Evento associato al mouse non presente	È stato dichiarato un gestore <code>On(MouseEvent)</code> oppure <code>onClipEvent</code> , ma non è stato specificato alcun evento o è stato rilevato un token imprevisto nella posizione in cui dovrebbe essere l'evento.
L'evento associato al mouse/clip filmato specificato non è valido	Lo script contiene un evento associato al mouse o al clip filmato non valido. Per la lista degli eventi associati al mouse o al clip filmato validi, consultare le voci <code>On(MouseEvent)</code> e <code>OnClipEvent</code> nel capitolo "Dizionario di ActionScript".
Identificatore del codice tasto non presente	È necessario specificare un codice tasto. Consultare l'appendice B per la lista dei codici tasto.
Codice tasto non valido	Il codice tasto specificato non esiste.
Individuati dati non validi in coda	Lo script o l'espressione sono stati analizzati sintatticamente in modo corretto, ma contengono caratteri aggiuntivi in coda che non è possibile analizzare sintatticamente.
Dichiarazione della funzione non consentita in questo punto	Una dichiarazione di funzione con nome è stata usata come espressione. Questo tipo di dichiarazione deve essere un'istruzione. Valida: <code>function sqr (x) { return x * x; }</code> Non valida: <code>var v = function sqr (x) { return x * x; }</code>
Nome della funzione non presente	Il nome specificato per questa funzione non è un nome di funzione valido.

Messaggio di errore	Descrizione
Nome del parametro non presente	In una dichiarazione di funzione non è presente un nome di parametro (argomento), ma è stato rilevato un token imprevisto.
Rilevato 'else' senza corrispondente 'if'	È stata rilevata un'istruzione <code>else</code> non preceduta da <code>if</code> . È possibile usare <code>else</code> solo insieme a un'istruzione <code>if</code> .
Il nome della scena deve essere una stringa racchiusa tra virgolette	L'argomento della scena di un'azione <code>gotoAndPlay</code> , <code>gotoAndStop</code> o <code>iffFrameLoaded</code> è del tipo errato. L'argomento della scena deve essere una costante di tipo stringa.
Errore interno	Si è verificato un errore interno nel compilatore di ActionScript. Inviare il file FLA che ha generato questo errore a Macromedia, con istruzioni dettagliate su come riprodurre il messaggio.
Cifre esadecimali dopo 0x non presenti	È stata rilevata la sequenza <code>0x</code> , ma questa non è seguita da cifre esadecimali valide.
Errore di apertura del file di inclusione: <nome file>	Si è verificato un errore durante l'apertura di un file incluso con la direttiva <code>include</code> . L'errore può essere dovuto alla mancanza del file o a un errore del disco.
Direttiva <code>#include</code> non strutturalmente corretta	Una direttiva <code>include</code> non è stata scritta correttamente. Una direttiva <code>include</code> deve usare la sintassi seguente: <code>#include "somefile.as"</code>
Il commento su più righe non è stato terminato	In un commento su più righe che inizia con <code>/*</code> manca il tag <code>*/</code> di chiusura.
Il valore letterale della stringa non è stato terminato correttamente	Il valore letterale di una stringa che inizia con virgolette di apertura, singole o doppie, non termina con le virgolette di chiusura.
Numero di parametri non corretto; <nome funzione> ne richiede esattamente <numero parametri>	È stata chiamata una funzione, ma è stato rilevato un numero di parametri non corretto.
GetProperty richiede un nome di proprietà	È stata chiamata la funzione <code>getProperty</code> , ma il secondo argomento non corrisponde al nome di una proprietà del clip filmato.
Il parametro <parametro> non può essere dichiarato più volte	Un nome di parametro compare più volte nella lista di parametri di una dichiarazione di funzione. Tutti i nomi di parametro devono essere univoci.

Messaggio di errore	Descrizione
La variabile <variabile> non può essere dichiarata più volte	Un nome di variabile compare più volte in un'istruzione <code>var</code> . Tutti i nomi di variabile in un'istruzione <code>var</code> devono essere univoci.
I gestori 'on' non possono essere annidati all'interno di altri gestori 'on'	Un gestore <code>on</code> è stato dichiarato all'interno di un altro gestore <code>on</code> . Tutti i gestori <code>on</code> devono apparire in cima alla lista di azioni.
L'istruzione deve apparire all'interno del gestore <code>on</code>	Nelle azioni per un'istanza di un pulsante è stata dichiarata un'istruzione senza un blocco <code>on</code> circostante. Tutte le azioni per un'istanza di pulsante devono apparire all'interno di un blocco <code>on</code> .
L'istruzione deve apparire all'interno del gestore <code>onClipEvent</code>	Nelle azioni per un'istanza di un clip filmato è stata dichiarata un'istruzione senza un blocco <code>onClipEvent</code> circostante. Tutte le azioni per un'istanza di clip filmato devono apparire all'interno di un blocco <code>onClipEvent</code> .
Gli eventi associati al mouse sono consentiti solo per le istanze di pulsanti	Un gestore di un evento associato a un pulsante è stato dichiarato in una lista di azioni fotogramma o in una lista di azioni di un'istanza di clip filmato. Gli eventi associati a un pulsante sono consentiti solo nelle liste di azioni di istanze di pulsanti.
Gli eventi associati al clip sono consentiti solo per le istanze di clip filmato	Un gestore di un evento associato a un clip è stato dichiarato in una lista di azioni fotogramma o in una lista di azioni di un'istanza di un pulsante. Gli eventi associati al clip sono consentiti solo nelle liste di azioni di istanze di clip filmato.

INDICE ANALITICO

A

accesso

- metodi 80
- proprietà di un oggetto 67

ActionScript

- confronto con JavaScript 17
- Flash 4 86
- Flash 4 paragonato a Flash 5 18
- funzioni di Flash 4 supportate 87
- modifica con un editor di testo 39
- nuove funzioni 17
- ottimizzazione 21
- script 24
- sintassi 49
- supporto per JavaScript 18
- terminologia 32

aggiunta di note 53

apertura

- file di Flash 4 86

apertura di finestra con messaggio 153

applicazioni Web

- connessione continua 146
- integrazione di Flash 137

argomenti 32

- in parentesi 52
- passaggio a funzioni 77

assegnazione di un nome alle variabili 57

assegnazione di un tipo alle variabili 58

associatività

- operatori 63

associazione di clip filmato 128

associazione di suoni 100

attachMovie, metodo 122

attachMovieClip, metodo 128

- argomenti 128

attachSound, metodo 100

Attiva azioni fotogramma semplici 157

Attiva pulsanti semplici 157

audio

- associazione 100

controllo bilanciamento 102

creazione dei controlli del volume 100

azione di spostamento di un clip filmato 127

azione wrapper 20

azioni 32

assegnazione a fotogrammi 47

assegnazione a oggetti 45

assegnazione per il controllo dei filmati 125

attivazione di quelle semplici 157

azioni fotogramma 47

con percorsi target 70

confronto con i metodi 123

eliminazione 39

esportazione 42

guida sensibile al contesto 21

indirizzamento di clip filmato 122

interattività 89

lista 69

modifica dei parametri 39

nuove funzioni 18

parametri dei pulsanti 48

principali 89

prova 45

riordinamento 39

ripetizione 72

selezione 38

stampa 42

tracce 166

azioni asincrone 140

azioni cicliche 72

azioni di esportazione 42

azioni di prova 45

azioni di riordinamento 39

azioni di stampa 42

azioni fotogramma

assegnazione 47

assegnazione a fotogrammi chiave 47

in conflitto in diversi livelli 157

posizionamento 47

B

barra di stato., Debugger 160
bilanciamento(audio), controllo 102

C

campi di ricerca 148
campi di testo 137
 scorrevoli 95
campi di testo di input
 nei moduli 147
campi di testo scorrevoli 95
caratteri speciali 55
caratteristiche 25
caricamento dati
 sicurezza 138
CGI, script
 formato standard 141
chiamata 57
 metodi di un oggetto 82
chiamata di metodi 57
childNodes 142
classi 25
 definizione 32
clip filmato
 assegnazione del nome a un'istanza 70
 associazione 128
 cambiamento della visibilità 24
 ciclo tra elementi secondari 72
 condivisione 128
 controllo 119
 definizione dei parametri clip 130
 di scambio 135
 duplicazione 27, 128
 eliminazione 128
 informazioni su 107
 inserimento del percorso target 39
 lista di oggetti 164
 modifica di proprietà nel Debugger 162
 nomi istanza 27
 rappresentazione grafica 25
 relazione gerarchica 110
 rilevamento della presenza di collisioni 104
 struttura gerarchica 109
 tipi di dati 57
 trascinamento 127
 visualizzazione Debugger 160
 visualizzazione di proprietà 163

clip intelligenti
 creazione 129
 impostazione dei parametri clip 133
codici di tasto
 determinazione 93
collegamento di clip filmato 128
collisioni
 rilevamento 104
 tra clip filmato 106
 tra clip filmato e un punto sullo stage 105
Color, oggetto 98
comando Elenca variabili 165
comando Prova filmato 45, 156
combinazioni di operazioni 66
commenti
 esempio 53
 sintassi 53
 sintassi colorata 43, 53
comportamenti 25
comunicazione
 tra linee temporali 112
comunicazione con Flash Player 151
condizioni
 verifica 71
conflitti di nomi 59
connessioni tramite socket 146
 script di esempio 147
contatori
 ripetizione azione con 72
controlli ActiveX 154
 stato di visualizzazione 160
controlli della tastiera 94
controllo audio 100
controllo dei clip filmato
 metodi 122
controllo dei filmati
 requisiti 119
convenzioni di denominazione 156
Core JavaScript Guide 18
costanti
 definizione 32
 sintassi 54
creazione
 clip intelligenti 129
creazione di oggetti 80
creazione di password 140
creazione di script 49
creazione di script con ActionScript 24

D

- dati caricati
 - verifica 140
- Debugger
 - attivazione 158
 - attivazione in un browser Web 159
 - barra di stato 160
 - Flash Debug Player 158
 - lista di controllo 162
 - password 158
 - proprietà del filmato 162
 - uso 158
 - variabili 160
 - visualizzazione di clip filmato 160
- determinazione della posizione del mouse 92
- dichiarazione di variabili 59
- distinzione tra maiuscole e minuscole
 - parole chiave 52
 - stringhe 54
- DOM XML 142
- droptarget, proprietà 127
- duplicateMovieClip, azione 113
- uplicazione di clip filmato 128

E

- ECMA (Associazione europea dei produttori di computer) 17
- editor esterni 41
- elementi di interfaccia
 - clip intelligenti 129
 - personalizzati 129
- Elenca oggetti, comando 164
- elenco di verifica, script 157
- eliminazione
 - clip filmato 128
 - filmati caricati 125
- eliminazione, azioni 39
- ereditarietà
 - creazione 84
- errori
 - conflitto di nomi 59
 - controllo della sintassi 43
 - messaggi 44
- errori di sintassi
 - controllo 43
 - evidenziazione 44
 - identificazione 43

- esecuzione degli operatori
 - ordine di associatività 63
 - ordine di precedenza 63
- esecuzione di un'applicazione da un proiettore 152
- Esplora filmato 157
 - visualizzazione 115
- esportazione in Flash 4 87
- espressioni
 - assegnazione di più variabili 66
 - confronto di valori 64
 - definizione 32
 - informazioni su 62
- eventi
 - definizione 32
- evidenziazione della sintassi 43
 - attivazione e disattivazione 73
 - obsoleta 44
- Extensible Markup Language 142

F

- file remoti
 - comunicazione con 137
- filmati
 - caricamento addizionali 125
 - controllo in Flash Player 154
 - lista di variabili 165
 - passaggio di informazioni tra filmati 138
 - prova in un browser 156
 - ridimensionamento in base a Flash Player 152
 - scaricamento 125
 - sicurezza 138
 - sostituzione con il filmato caricato 125
 - visualizzazione nelle dimensioni originali 152
- filmati caricati
 - controllo 119
 - identificazione 70
- filmato di esempio 34
- finestra con messaggio, visualizzazione 153
- Finestra dello script
 - modifica del carattere 42
- finestre di dialogo nei moduli 148
- Flash 4, file
 - apertura 86
- Flash 5
 - creazione di contenuto Flash 4 87
- Flash Debug Player 158
- Flash Player 152

- comunicazione con 151
- disattivazione del menu di scelta rapida 152
- metodi 137, 154
- ridimensionamento filmati 152
- tipo di visualizzazione 160
- versione di esportazione 44
- visualizzazione a schermo interno 152
- visualizzazione dei menu normale 152
- visualizzazione del menu di scelta rapida 152
- formato MIME application/x-www-urlformencoded 141
- fotogrammi
 - assegnazione di azioni a 47
- fotogrammi chiave
 - assegnazione di azioni a fotogrammi 47
- fscommand, azione 137
 - comandi e argomenti 152
 - comunicazione con Director 153
 - uso 152
- funzioni
 - chiamata 78
 - definizione 33, 76
 - di costruzione 26
 - esempio 32
 - passaggio di argomenti a 77
 - personalizzati 76
 - predefinite 74
 - regole 74
 - valori restituiti 77
 - variabili locali in 77
- funzioni assegnate 33
- funzioni di costruzione
 - esempio 26, 32
- funzioni personalizzate 76
- funzioni predefinite 74
 - lista 74

G

- gerarchia
 - clip filmato 109
 - clip filmato principale-secondario 110
 - ereditarietà 84
- gestori
 - definizione 33
 - verifica dei dati XML 140
- getBounds, metodo 122
- getBytesLoaded, metodo 122

- getBytesTotal, metodo 122
- getCode, metodo 94
- getURL, azione 138
 - comunicazione con script lato server 141
 - modulo di ricerca 148
- globalToLocal, metodo 122
- Guida di Flash
 - azioni 21

H

- hitTest, azione
 - esempio 36
- hitTest, metodo 104
 - controllo dei filmati 122
- HTTP, protocollo 138
 - comunicazione con script lato server 141
- HTTP, richieste
 - permesso 138
- HTTPS, protocollo 138

I

- identificatori
 - con valori 34
 - definizione 33
- identificazione
 - azione duplicateMovieClip 113
- if, istruzioni 29, 71
- importazione in ActionScript 42
- in lettere maiuscole 52
- indirizzi gerarchici 34
- informazioni
 - passaggio tra filmati 138
- inserimento di percorsi target 119
- Inserisci percorso target, pulsante 119
- interattività
 - complessa 90
 - creazione 89
 - moduli 147
- interfaccia personalizzata 129
 - clip filmato xch 135
 - creazione 134
- invio di informazioni
 - a file remoti 137
 - formato con codifica URL 138
 - formato XML 138
 - via connessione tramite socket TCP/IP 138
- istanze

- copia 26
 - definizione 33
- istruzioni
 - impostazione come espressione 157
 - raggruppamento 51
 - riordinamento 39
 - salti logici 29
 - terminazione 51
- istruzioni condizionali 29
- istruzioni di raggruppamento 51
- istruzioni terminazione 51

J

- JavaScript
 - confronto con ActionScript 17
 - Developer Central 18
 - invio di messaggi a 152
 - istruzione alert 166
 - istruzione with 114
 - linguaggio supportati 18
 - modifica 39
 - standard internazionale 17

K

- Key, oggetto 93

L

- linee temporali
 - alias principale 117
 - comunicazione tra 112
 - controllo 122
 - indirizzamento di più azioni 124
 - multiple 108
- Lista a sinistra
 - ridimensionamento 39
- Lista delle azioni
 - ridimensionamento 39
- lista di controllo
 - Debugger 162
- lista operatori 407
- LiveConnect 154
- livelli 70
 - caricamento 125
 - caricamento di filmati 108
 - denominazione nel percorso target 116
 - gerarchia 109

- percorso assoluto 116
- loadMovie, azione 138
 - comunicazione con script lato server 141
 - livelli 108
 - verifica dei dati caricati 140
- loadVariables, azione 138
 - comunicazione con script lato server 141
 - verifica dei dati caricati 140
- localToGlobal, metodo 122

M

- Macromedia Director
 - comunicazione con 153
 - manipolazione dei numeri 55
 - matrici a più dimensioni 68
 - metodi 25, 57
 - accesso 80
 - assegnazione 125
 - chiamata 123
 - confronto con le azioni 123
 - definizione 33
 - indirizzamento di più linee temporali 124
 - oggetto 79
 - metodi di un oggetto
 - chiamata 82
 - metodo ASCII 93
 - modalità di modifica
 - preferenza 41
 - sostituzione 40
 - modalità di prova filmato 157
 - Modalità esperto 39
 - chiamata di funzione 74
 - Modalità normale 37
 - chiamata di funzione 75
 - modifica di script
 - esternamente 41
 - modalità 40
 - moduli
 - creazione 137, 147
 - elementi richiesti 148
 - ricerca 148
 - variabili 149
 - verifica dei dati 150
- Mostra sintassi obsoleta, comando 44
- MovieClip, oggetto
 - controllo dei filmati 122
 - informazioni su 27

uso 82

N

navigazione

controllo 89

Netscape DevEdge Online 18

nodì 142

nomefilmato_DoFSCCommand 152

nomi 33

nomi istanza

assegnazione 70

clip filmato 27

definizione 33

impostazione dinamica 67

numeri 55

conversione in interi a 32 bit 66

Nuovo operatore 80

O

oggetti 25

assegnazione di azioni 45

creazione 80

creazione di personalizzati 83

definizione 33

personalizzati 83

predefiniti 79

tipi di dati 56

oggetti istanziati 80

oggetti personalizzati 83

oggetti predefiniti

lista 79

onClipEvent(enterFrame)

esempio 36

onClipEvent(load)

esempio 35

operatore di accesso matrice 67

operatore di inizializzazione degli oggetti 80

operatore punto 67

operatori

accesso matrice 67

assegnazione 66

associatività 63

bit a bit 66

combinazione con valori 62

confronto 64

definizione 33

logici 65

numerici 64

punto 67

stringa. 65

uguaglianza 66

operatori bit a bit 66

operatori di assegnazione 66

composta 66

operatori di uguaglianza 66

operatori logici 65

operatori numerici 64

operatori stringa 65

ordine di esecuzione 28

controllo 30

Output, finestra

comando Elenca variabili 165

Elenca oggetti, comando 164

opzioni 164

uso 164

P

pannello Azioni 37

categorie 37

modalità di modifica 37

Modalità normale

Lista a sinistra 37

opzioni 41

visualizzazione 37

pannello Azioni oggetto 35

parametri

argomenti e 77

immissione 38

modifica 39

passaggio a funzioni 77

visualizzazione 46

parametri clip

assegnazione 129

definizione 130

impostazione 132

impostazione di un clip intelligente 133

Parametri clip, pannello

sostituzione con l'interfaccia personalizzata 134

parametri, campi 38

parole chiave

definizione 33

distinzione tra maiuscole e minuscole 52

lista 53

- sintassi colorata 43
- parole riservate 33
 - lista 53
 - this 35
- passaggio di variabili
 - per riferimento 61
 - per valore 60
- password
 - creazione 140
 - Debugger 158
- percorsi target 115
 - definizione 34
 - espressione 121
 - immissione 71
 - inserimento 39
 - nomi di livello 116
 - specifica 70, 119
- percorso target assoluto 115
- percorso target relativo 115
- Plug-in di Netscape 160
- porte
 - connessione XMLSocket 139
- posizione del mouse
 - determinazione 92
- preferenze
 - modalità di modifica 41
- _parent, alias 117
- progettazione di script 25
- proiettori
 - esecuzione di un'applicazione 152
- proprietà 25
 - collezioni 33
 - definizione 34
 - non modificabili 54
 - sintassi colorata 43
- Proprietà di concatenamento del simbolo, finestra di
 - dialogo 128
- proprietà di un oggetto
 - accesso 81
- Proprietà maxscroll 95
- Proprietà scroll 95
- prototype, proprietà 84
- prova
 - filmati 156
 - script 156
 - valori di variabili 60
- prova di azioni fotogramma 48

- pulsante Invia 148
- puntatori personalizzati
 - creazione 90

R

- raccolta di dati 137
- recupero di informazioni da file remoti 137
- relazioni principale-secondario 110
- removeMovieClip, azione 128
- RGB, metodo 98
- riferimenti
 - permanenti 61
- riferimenti fissi 61
- riferimenti removibili 61
- riferimento a variabili 68
- rilevamento della presenza di collisioni 104
- rilevamento tasti premuti 93
- ripetizione ciclica
 - oggetti secondari 72
- ripetizione di azioni 72
- risoluzione dei problemi
 - azione with trace 166
 - elenco di verifica 157
 - indicazioni 156
 - lista di oggetti 164
 - lista di variabili 165
 - panoramica 155
 - uso della finestra Output 164

S

- salto logico 29
- salvataggio di script 156
- Scheda Proprietà 163
- Scheda Variabili 160
- script
 - controllo dell'esecuzione 30
 - controllo flusso 71
 - creazione 49
 - debug 158
 - dichiarazione di variabili 60
 - esempio 34
 - flusso 28
 - importazione 42
 - indicazioni 156
 - inserimento di commenti 156
 - ordine di esecuzione 28

- progettazione 25
- ricerca 42
- risoluzione dei problemi 155
- script lato server
 - formato XML 143
 - linguaggi 137
- scripting orientato agli oggetti 25
- segnaposto 32
- sequenze di caratteri 54
- sequenze di escape 55
- set di caratteri ISO-8859-1 18
- set di caratteri Shift-JIS 18
- set variable, azione
 - verifica dei dati 150
- setPan, metodo 100
- setTransform, metodo 98
- setVolume, metodo 100
- sicurezza 138
 - HTML standard 139
- simboli animati 57
- sintassi
 - barra inclinata 51
 - distinzione tra maiuscole e minuscole 52
 - parentesi 52
 - parentesi graffe 51
 - punto 50
 - punto e virgola 51
 - regole 49
- Sintassi colorata, comando 43
- sintassi del punto 50
 - percorsi target 116
- sintassi della barra inclinata 51
 - percorsi target 117
- sintassi evidenziata 43
- siti remoti
 - connessione continua 146
- Sound, oggetto 100
- specifica ECMA-262 17
 - azione tellTarget 123
- stringhe 54
 - caratteri di escape 55
 - sintassi colorata 43
- stringhe di concatenazione 54
- swapDepths, metodo 122

T

- targetPath, funzione 121

- tasti premuti
 - rilevamento 93
- TCP/IP, connessione
 - con oggetto XMLSocket 147
 - invio di informazioni 138
- termini, definizione 32
- testo
 - ricerca di testo negli script 42
- testo di input 95
- testo dinamico 95
- this 35
 - alias della linea temporale corrente 117
- tipi di dati
 - Boolean 56
 - clip filmato 57
 - definizione 32
 - Number 55
 - Object 56
 - regole 54
- tipi di dati di base 54
 - Flash 4 87
- tipi di dati puntatore 54
- trascinamento di clip filmato
 - valutazione 127

U

- unloadMovie, azione 125
- URL, sottodomini 138

V

- valori
 - manipolazione in espressioni 62
- valori ASCII 93
- valori booleani 56
 - confronto 65
- valori dei colori
 - impostazione 98
- variabili
 - assegnazione di nomi significativi 156
 - assegnazione di più 66
 - assegnazione di un nome 57
 - assegnazione di un'area di validità 58
 - caricamento da file remoti 137
 - controllo dei valori tramite campi di testo 157
 - conversione in XML 144
 - definizione 34

- determinazione del tipo 58
- dichiarazione 59
- impostazione dinamica 67
- invio a file remoti 137
- modifica di valori nel Debugger 161
- modifica nel Debugger 160
- nascoste 149
- nei moduli 149
- passaggio con i clip intelligenti 129
- passaggio di valori 60
- percorso assoluto 162
- prova 60
- regole 57
- riferimento al valore 61
- rimozione dalla lista di controllo 162
- uso in uno script 60
- verifica 150
- variabili globali 58
- variabili locali 58
 - esempio 59
 - in funzioni 77
- VBScript 39
- verifica dei dati immessi 150
 - script di esempio 150
- visualizzazione
 - menu di scelta rapida di Flash Player 152
- volume
 - controlli 100
 - controllo scorrevole 101

W

- with, azione 114
 - indirizzamento di più linee temporali 124

X

- xch, nome di istanza 135
- XML 142
 - esempio di conversione di variabile 143
 - gerarchia 142
 - invio di informazioni con metodi XML 138
 - invio di informazioni tramite socket TCP/IP 138
 - script lato server 143
- XML, metodi dell'oggetto 143
- XMLSocket, oggetto
 - metodi 146
 - uso 146

