

# macromedia® **FLASH™5**

## Guia de Referência do ActionSc

**ST**  
SuperTráfego  
[www.supertrafego.com](http://www.supertrafego.com)

**HOTEL DA WEB**  
★★★★



**COMPRE JÁ!**  
1000 apostilas + provas  
+ resumos + teses

**Buscar**  
na web

**super**  
**Ofertas**



  
macromedia

## Marcas Comerciais

Macromedia, o logotipo da Macromedia, o logotipo Made With Macromedia, Authorware, Backstage, Director, Extreme 3D e Fontographer são marcas registradas e 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 são marcas comerciais da Macromedia, Inc. Outros nomes de produtos, logotipos, designs, título, palavras ou frases mencionados nesta publicação podem ser marcas comerciais, marcas de serviço ou nomes comerciais da Macromedia, Inc. ou outras entidades e podem estar registradas em determinadas jurisdições.

## Isenção de Responsabilidade da Apple

A APPLE COMPUTER, INC. NÃO FORNECE GARANTIAS, EXPRESSAS OU IMPLÍCITAS, RELACIONADAS AO PACO DE SOFTWARE INCLUÍDO, A SUA COMERCIALIZAÇÃO OU ADEQUAÇÃO A UMA FINALIDADE ESPECÍFICA. A EXCLUSÃO DE GARANTIAS IMPLÍCITAS NÃO É PERMITIDA EM ALGUMAS JURISDIÇÕES. A EXCLUSÃO ACIMA PODE NÃO SE APLICAR A VOCÊ. ESSA GARANTIA CONCEDE-LHE DIREITOS LEGAIS ESPECÍFICOS. VOCÊ PODERÁ TER OUTROS DIREITOS QUE PODEM VARIAR DE UMA JURISDIÇÃO PARA OUTRA.

Copyright© 2000 Macromedia, Inc. Todos os direitos reservados. Este manual não pode ser copiado, fotocopiado, reproduzido, traduzido ou convertido em qualquer forma eletrônica ou legível por máquina, por inteiro ou em parte, sem a aprovação prévia, por escrito, da Macromedia, Inc.

Número da peça ZFL50M200PO

## Agradecimentos

Gerenciamento de projeto: Erick Vera

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

Edição: Peter Fenczik, Rosana Francescato, Ann Szabla

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

Design da Ajuda e da Documentação: Chris Basmajian e Noah Zilberberg

Produção: Chris Basmajian e Rebecca Godbois

Gerenciamento do Projeto de Localização: Yuko Yagi

Produção da Localização: Masayo "Noppe" Noda e Bowne Global Solutions

Agradecimentos especiais: Jeremy Clark, Brian Dister e toda a equipe do Flash Development, Michael Dominguez, Margaret Dumas, Sherri Harte, Yoshika Hedberg, Tim Hussey, Kipling Inscore, Alyn Kelley, Luciana de Oliveira Balsemão, Pete Santangeli, Denise Seymour e toda a equipe do Flash QA, Cyn Taylor e Eric Wittman

Primeira edição: Setembro de 2000

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



# SUMÁRIO



## INTRODUÇÃO

Introdução .....	17
Novidades do ActionScript do Flash 5 .....	17
Diferenças entre o ActionScript e o JavaScript .....	18
Edição de texto. ....	19
Sintaxe de ponto .....	19
Tipos de dados .....	19
Variáveis locais .....	19
Funções definidas pelo usuário .....	19
Objetos predefinidos .....	20
Ações de clipe. ....	20
Novas ações .....	20
Clipes Inteligentes .....	20
Depurador .....	20
Suporte a XML .....	21
Otimização. ....	21
Usando a Ajuda do Flash para ações .....	21

## CAPÍTULO 1

Noções básicas sobre o ActionScript. ....	23
Sobre como escrever scripts em ActionScript .....	24
Sobre como planejar e depurar scripts .....	25
Sobre como criar scripts orientados a objetos .....	26
Sobre o objeto MovieClip .....	27
Como é o fluxo dos scripts .....	28
Controlando o momento em que o ActionScript é executado. ...	31
Terminologia do ActionScript .....	32
Analisando um script simples .....	35
Usando o painel Ações. ....	37
Modo Normal .....	37
Modo Especialista .....	40
Alternando entre modos de edição .....	41
Usando um editor externo .....	42



Escolhendo opções do painel Ações .....	42
Realçando e verificando a sintaxe .....	44
Sobre o realce de erros .....	44
Atribuindo ações a objetos .....	45
Atribuindo ações a quadros .....	47

## CAPÍTULO 2

### Escrevendo scripts com o ActionScript .....

Usando a sintaxe do ActionScript .....	50
Sobre tipos de dados .....	54
Sobre variáveis .....	57
Usando operadores para manipular valores em expressões .....	62
Usando ações .....	69
Controlando o fluxo em scripts .....	71
Usando funções predefinidas .....	74
Criando funções personalizadas .....	76
Usando objetos predefinidos .....	79
Usando objetos personalizados .....	83
Abrindo arquivos do Flash 4 .....	86
Usando o Flash 5 para criar o conteúdo do Flash 4 .....	88

## CAPÍTULO 3

### Criando interatividade com o ActionScript .....

Criando um cursor personalizado .....	92
Obtendo a posição do mouse .....	94
Capturando pressionamentos de teclas .....	95
Criando um campo de texto de rolagem .....	97
Definindo valores de cores .....	100
Criando controles de som .....	102
Detectando colisões .....	105

## CAPÍTULO 4

### Trabalhando com clipes de filme .....

Sobre várias Linhas de Tempo .....	110
Sobre a relação hierárquica de Linhas de Tempo .....	112
Enviando mensagens entre Linhas de Tempo .....	114
Sobre caminhos de destino absolutos e relativos .....	117
Especificando caminhos de destino .....	121
Usando ações e métodos para controlar Linhas de Tempo .....	124





Sobre métodos versus ações . . . . .	125
Usando vários métodos ou ações para especificar uma	
Linha de Tempo como destino . . . . .	126
Atribuindo uma ação ou um método . . . . .	127
Carregando e descarregando filmes adicionais . . . . .	128
Alterando a posição e a aparência de um clipe de filme. . . . .	128
Arrastando clipes de filme . . . . .	129
Duplicando e removendo clipes de filme . . . . .	130
Anexando clipes de filme . . . . .	130
Criando clipes inteligentes . . . . .	131
Definindo parâmetros do clipe . . . . .	132

## CAPÍTULO 5

### Integrando o Flash com aplicativos da Web . . . . 139

Enviando e carregando variáveis para/de um arquivo remoto . . . . .	140
Usando loadVariables, getURL e loadMovie . . . . .	144
Sobre XML . . . . .	145
Usando o objeto XML . . . . .	145
Usando o objeto XMLSocket . . . . .	149
Criando formulários . . . . .	151
Criando um formulário de pesquisa . . . . .	152
Usando variáveis em formulários . . . . .	153
Verificando os dados inseridos . . . . .	153
Enviando mensagens para/do Flash Player. . . . .	155
Usando fscommand . . . . .	155
Sobre os métodos do Flash Player. . . . .	158

## CAPÍTULO 6

### Solucionando problemas do ActionScript . . . . . 159

Diretrizes para criação de páginas e solução de problemas . . . . .	160
Usando o Depurador. . . . .	162
Ativando a depuração em um filme . . . . .	163
Sobre a barra de status . . . . .	164
Sobre a lista de exibição . . . . .	164
Exibindo e modificando variáveis. . . . .	164
Usando a lista Observação . . . . .	166
Exibindo propriedades do filme e alterando	
propriedades editáveis . . . . .	167
Usando a janela Saída . . . . .	168



Usando Listar Objetos .....	169
Usando Listar Variáveis .....	170
Usando trace .....	171

## CAPÍTULO 7

Dicionário ActionScript .....	173
-------------------------------	-----

Exemplo de entrada para a maioria dos elementos do ActionScript ..	174
Exemplo de entrada para objetos .....	175
Conteúdo do dicionário .....	176
— (decremento) .....	189
++ (incremento) .....	189
! (NOT lógico) .....	191
!= (diferença) .....	191
% (módulo) .....	192
%= (atribuição de módulo) .....	193
& (AND bit a bit) .....	193
&& (AND de curto-circuito) .....	194
&= (atribuição AND bit a bit) .....	194
() (parênteses) .....	195
- (subtração) .....	196
* (multiplicação) .....	197
*= (atribuição de multiplicação) .....	198
, (vírgula) .....	198
. (operador ponto) .....	199
?: (condicional) .....	200
/ (divisão) .....	200
// (delimitador de comentário) .....	201
/* (delimitador de comentário) .....	202
/= (atribuição de divisão) .....	202
[] (operador de acesso de matriz) .....	203
^(XOR bit a bit) .....	204
^= (atribuição XOR bit a bit) .....	204
{ } (inicializador de objeto) .....	205
(OR bit a bit) .....	206
(OR) .....	207
= (atribuição OR bit a bit) .....	208
~ (NOT bit a bit) .....	208
+ (adição) .....	209
+= (atribuição de adição) .....	210
< (menor que) .....	210



<< (deslocamento para a esquerda bit a bit) . . . . .	211
<<= (deslocamento para a esquerda bit a bit e atribuição) . . . . .	212
<= (menor ou igual a) . . . . .	213
<> (diferença) . . . . .	213
= (atribuição) . . . . .	214
-= (atribuição de negação) . . . . .	215
== (igualdade) . . . . .	216
> (maior que) . . . . .	216
>= (maior ou igual a) . . . . .	217
>> (deslocamento para a direita bit a bit) . . . . .	218
>>= (deslocamento para a direita bit a bit e atribuição) . . . . .	219
>>> (deslocamento para a direita não assinado bit a bit) . . . . .	220
>>>= (deslocamento para a direita não assinado bit a bit e atribuição) . . . . .	221
add . . . . .	222
_alpha . . . . .	222
and . . . . .	223
Array (objeto) . . . . .	224
Array.concat . . . . .	226
Array.join . . . . .	226
Array.length . . . . .	227
Array.pop . . . . .	228
Array.push . . . . .	228
Array.reverse . . . . .	229
Array.shift . . . . .	229
Array.slice . . . . .	230
Array.sort . . . . .	230
Array.splice . . . . .	232
Array.toString . . . . .	232
Array.unshift . . . . .	233
Boolean (função) . . . . .	233
Boolean (objeto) . . . . .	234
Boolean.toString . . . . .	235
Boolean.valueOf . . . . .	235
break . . . . .	235
call . . . . .	236
chr . . . . .	236
Color (objeto) . . . . .	237
Color.getRGB . . . . .	238
Color.getTransform . . . . .	238
Color.setRGB . . . . .	239



Color.setTransform . . . . .	239
continue . . . . .	241
_currentframe . . . . .	242
Date (objeto) . . . . .	242
Date.getDate . . . . .	246
Date.getDay . . . . .	246
Date.getFullYear . . . . .	246
Date.getHours . . . . .	247
Date.getMilliseconds . . . . .	247
Date.getMinutes . . . . .	248
Date.getMonth . . . . .	248
Date.getSeconds . . . . .	248
Date.getTime . . . . .	249
Date.getTimezoneOffset . . . . .	249
Date.getUTCDate . . . . .	250
Date.getUTCDay . . . . .	250
Date.getUTCFullYear . . . . .	250
Date.getUTCHours . . . . .	251
Date.getUTCMilliseconds . . . . .	251
Date.getUTCMinutes . . . . .	251
Date.getUTCMonth . . . . .	252
Date.getUTCSeconds . . . . .	252
Date.getYear . . . . .	252
Date.setDate . . . . .	253
Date.setFullYear . . . . .	253
Date.setHours . . . . .	254
Date.setMilliseconds . . . . .	254
Date.setMinutes . . . . .	254
Date.setMonth . . . . .	255
Date.setSeconds . . . . .	255
Date.setTime . . . . .	255
Date.setUTCDate . . . . .	256
Date.setUTCFullYear . . . . .	256
Date.setUTCHours . . . . .	257
Date.setUTCMilliseconds . . . . .	257
Date.setUTCMinutes . . . . .	257
Date.setUTCMonth . . . . .	258
Date.setUTCSeconds . . . . .	258
Date.setYear . . . . .	259
Date.toString . . . . .	259



Date.UTC	260
delete	261
do... while	262
_droptarget	262
duplicateMovieClip	262
else	263
eq (igual — específico de sequência de caracteres)	263
escape	263
eval	266
evaluate	267
_focusrect	267
for	268
for..in	269
_framesloaded	271
fscommand	271
function	272
ge (maior ou igual a — específico de sequências de caracteres)	273
getProperty	274
getTimer	274
getURL	275
getVersion	276
gotoAndPlay	277
gotoAndStop	277
gt (maior que — específico de sequências de caracteres)	278
_height	278
_highquality	279
if	279
ifFrameLoaded	280
#include	280
Infinity	281
int	281
isFinite	281
isNaN	282
Key (objeto)	283
Key.BACKSPACE	285
Key.CAPSLOCK	285
Key.CONTROL	285
Key.DELETEKEY	286
Key.DOWN	286
Key.END	286



Key.ENTER . . . . .	287
Key.ESCAPE . . . . .	287
Key.getAscii . . . . .	287
Key.getCode . . . . .	288
Key.HOME . . . . .	288
Key.INSERT . . . . .	288
Key.isDown . . . . .	289
Key.isToggled . . . . .	289
Key.LEFT . . . . .	289
Key.PGDN . . . . .	290
Key.PGUP . . . . .	290
Key.RIGHT . . . . .	290
Key.SHIFT . . . . .	291
Key.SPACE . . . . .	291
Key.TAB . . . . .	291
Key.UP . . . . .	292
le (menor que ou igual a — específico da sequência de caracteres) . . . . .	292
length . . . . .	292
_level . . . . .	293
loadMovie . . . . .	294
loadVariables . . . . .	295
lt (menor que — sequência de caracteres específica) . . . . .	297
Math (objeto) . . . . .	297
Math.abs . . . . .	299
Math.acos . . . . .	299
Math.asin . . . . .	300
Math.atan . . . . .	300
Math.atan2 . . . . .	301
Math.ceil . . . . .	301
Math.cos . . . . .	301
Math.E . . . . .	302
Math.exp . . . . .	302
Math.floor . . . . .	303
Math.log . . . . .	303
Math.LOG2E . . . . .	304
Math.LOG10E . . . . .	304
Math.LN2 . . . . .	305
Math.LN10 . . . . .	305
Math.max . . . . .	306
Math.min . . . . .	306



Math.PI	307
Math.pow	307
Math.random	307
Math.round	308
Math.sin	308
Math.sqrt	309
Math.SQRT1_2	309
Math.SQRT2	309
Math.tan	310
maxscroll	310
mbchr	311
mblength	311
mbord	311
mbsubstring	312
Mouse (objeto)	312
Mouse.hide	313
Mouse.show	313
MovieClip (objeto)	314
MovieClip.attachMovie	315
MovieClip.duplicateMovieClip	316
MovieClip.getBounds	316
MovieClip.getBytesLoaded	317
MovieClip.getBytesTotal	317
MovieClip.getURL	318
MovieClip.globalToLocal	318
MovieClip.gotoAndPlay	319
MovieClip.gotoAndStop	319
MovieClip.hitTest	320
MovieClip.loadMovie	321
MovieClip.loadVariables	322
MovieClip.localToGlobal	323
MovieClip.nextFrame	324
MovieClip.play	324
MovieClip.prevFrame	324
MovieClip.removeMovieClip	325
MovieClip.startDrag	325
MovieClip.stop	326
MovieClip.stopDrag	326
MovieClip.swapDepths	326
MovieClip.unloadMovie	327



_name . . . . .	327
NaN . . . . .	328
ne (diferente — específico de sequência de caracteres) . . . . .	328
new . . . . .	328
newline . . . . .	328
nextFrame . . . . .	330
nextScene . . . . .	330
not . . . . .	331
null . . . . .	331
Number (função) . . . . .	332
Number (objeto) . . . . .	332
Number.MAX_VALUE . . . . .	334
Number.MIN_VALUE . . . . .	335
Number.NaN . . . . .	335
Number.NEGATIVE_INFINITY . . . . .	335
Number.POSITIVE_INFINITY . . . . .	336
Number.toString . . . . .	336
Number.valueOf . . . . .	337
Object (objeto) . . . . .	337
Object.toString . . . . .	338
Object.valueOf . . . . .	338
onClipEvent . . . . .	339
on(mouseEvent) . . . . .	341
or . . . . .	342
ord . . . . .	343
_parent . . . . .	343
parseFloat . . . . .	344
parseInt . . . . .	344
play . . . . .	345
prevFrame . . . . .	346
prevScene . . . . .	347
print . . . . .	347
printAsBitmap . . . . .	348
_quality . . . . .	350
random . . . . .	350
removeMovieClip . . . . .	351
return . . . . .	351
_root . . . . .	352
_rotation . . . . .	353
scroll . . . . .	354





Selection (objeto) . . . . .	354
Selection.getBeginIndex . . . . .	355
Selection.getCaretIndex. . . . .	356
Selection.getEndIndex. . . . .	356
Selection.setFocus. . . . .	356
Selection.setFocus . . . . .	357
Selection.setSelection . . . . .	357
set . . . . .	358
setProperty . . . . .	359
Sound (objeto) . . . . .	359
Sound.attachSound. . . . .	361
Sound.getPan . . . . .	361
Sound.getTransform . . . . .	362
Sound.getVolume . . . . .	362
Sound.setPan. . . . .	362
Sound.setTransform . . . . .	363
Sound.setVolume . . . . .	366
Sound.start . . . . .	367
Sound.stop . . . . .	367
_soundbuftime . . . . .	368
startDrag. . . . .	368
stop. . . . .	369
stopAllSounds. . . . .	369
stopDrag. . . . .	370
String (função) . . . . .	370
" " (delimitador de seqüência de caracteres). . . . .	371
String (objeto). . . . .	372
String.charAt. . . . .	374
String.charCodeAt . . . . .	374
String.concat. . . . .	375
String.fromCharCode . . . . .	375
String.indexOf . . . . .	376
String.lastIndexOf. . . . .	376
String.length . . . . .	377
String.slice. . . . .	377
String.split . . . . .	378
String.substr . . . . .	378
String.substring. . . . .	379
String.toLowerCase. . . . .	379
String.toUpperCase. . . . .	380



substring . . . . .	380
_target . . . . .	381
targetPath . . . . .	381
tellTarget . . . . .	382
this . . . . .	382
toggleHighQuality . . . . .	384
_totalframes . . . . .	384
trace . . . . .	385
typeof . . . . .	386
unescape . . . . .	386
unloadMovie . . . . .	387
updateAfterEvent . . . . .	387
_url . . . . .	388
var . . . . .	388
_visible . . . . .	389
void . . . . .	389
while . . . . .	389
_width . . . . .	391
with . . . . .	391
_x . . . . .	394
XML (objeto) . . . . .	395
XML.appendChild . . . . .	397
XML.attributes . . . . .	398
XML.childNodes . . . . .	398
XML.cloneNode . . . . .	399
XML.createElement . . . . .	399
XML.createTextNode . . . . .	399
XML.docTypeDecl . . . . .	400
XML.firstChild . . . . .	401
XML.hasChildNodes . . . . .	401
XML.insertBefore . . . . .	402
XML.lastChild . . . . .	402
XML.load . . . . .	403
XML.loaded . . . . .	403
XML.nextSibling . . . . .	404
XML.nodeName . . . . .	404
XML.nodeType . . . . .	405
XML.nodeValue . . . . .	405
XML.onLoad . . . . .	406
XML.parentNode . . . . .	407



XML.parseXML . . . . .	407
XML.previousSibling . . . . .	407
XML.removeNode . . . . .	408
XML.send . . . . .	408
XML.sendAndLoad . . . . .	408
XML.status . . . . .	409
XML.toString . . . . .	410
XML.xmlDecl . . . . .	411
XMLSocket (objeto) . . . . .	412
XMLSocket.close . . . . .	414
XMLSocket.connect . . . . .	414
XMLSocket.onClose . . . . .	416
XMLSocket.onConnect . . . . .	416
XMLSocket.onXML . . . . .	418
XMLSocket.send . . . . .	419
_xmouse . . . . .	420
_xscale . . . . .	420
_y . . . . .	421
_ymouse . . . . .	421
_yscale . . . . .	422

## APÊNDICE A

Associatividade e precedência de operadores . . . . .	423
Lista de operadores . . . . .	423

## APÊNDICE B

Teclas do Teclado e Valores de Códigos de Teclas . . . . .	427
Teclas do teclado numérico . . . . .	430
Teclas de função . . . . .	430

## APÊNDICE C

Mensagens de erro . . . . .	433
-----------------------------	-----



**COMPRE JÁ!**  
1000 apostilas + provas  
+ resumos + teses



# INTRODUÇÃO

## Introdução

.....



ActionScript é a linguagem de criação de scripts do Flash. O ActionScript pode ser usado para controlar objetos em filmes do Flash a fim de criar elementos para a navegação e interatividade, possibilitando a criação de filmes e aplicativos da Web com grande interatividade.

## Novidades do ActionScript do Flash 5

O ActionScript do Flash 5 oferece novos recursos incríveis para criação de sites da Web interativos com jogos sofisticados, formulários, pesquisas e interatividade em tempo real, como sistemas de bate-papo.

O ActionScript do Flash 5 tem diversos recursos e convenções de sintaxe novos que o tornam semelhante à linguagem de programação JavaScript básica. Este manual explicará os conceitos de programação básicos, como funções, variáveis, comandos, operadores, condicionais e loops. O capítulo 7 deste manual, “Dicionário do ActionScript”, contém uma entrada detalhada para cada elemento do ActionScript.

Este manual não tenta ensinar programação em geral. Há vários recursos disponíveis que fornecem mais informações sobre conceitos de programação gerais e sobre a linguagem JavaScript.

A ECMA (European Computers Manufacturers Association) escreveu um documento chamado ECMA-262, derivado do JavaScript, para servir como padrão internacional para a linguagem JavaScript. O ActionScript é baseado na especificação ECMA-262, disponível em <http://www.ecma.ch>.

A Netscape DevEdge Online tem uma central de desenvolvedores JavaScript (<http://developer.netscape.com/tech/javascript/index.html>) que contém documentação e artigos úteis para a compreensão do ActionScript. O recurso mais valioso é o Core JavaScript Guide, localizado em <http://developer.netscape.com/docs/manuals/js/core/jsguide/index.htm>.

## Diferenças entre o ActionScript e o JavaScript

Não é necessário saber JavaScript para usar e aprender ActionScript. Entretanto, se você souber JavaScript, o ActionScript lhe parecerá familiar. Estas são algumas das diferenças entre o ActionScript e o JavaScript:

- O ActionScript não oferece suporte a objetos específicos de navegadores, como Document, Window e Anchor.
- O ActionScript não oferece suporte completo a todos os objetos JavaScript predefinidos.
- O ActionScript oferece suporte a construções de sintaxe não permitidas no JavaScript (por exemplo, as ações `tellTarget` e `ifFrameLoaded` e a sintaxe de barra).
- O ActionScript não suporta algumas construções de sintaxe do JavaScript, como os rótulos `switch`, `continue`, `try`, `catch`, `throw` e `statement`.
- O ActionScript não suporta o construtor `Function` JavaScript.
- No ActionScript, a ação `eval` só pode executar referências de variáveis.
- No JavaScript, `toString` de `undefined` é `undefined`. No Flash 5, para que haja compatibilidade com o Flash 4, `toString` de `undefined` é " ".
- No JavaScript, a avaliação de `undefined` em um contexto numérico resulta em `NaN`. No Flash 5, para que haja compatibilidade com o Flash 4, a avaliação de `undefined` resulta em 0.
- O ActionScript não suporta o Unicode; ele suporta conjuntos de caracteres ISO-8859-1 e Shift-JIS.

## Edição de texto

Você pode inserir scripts diretamente no painel Ações no Modo Especialista. Além disso, você pode escolher os elementos em um menu pop-up ou em uma lista Caixa de Ferramentas, exatamente como no Flash 4.

## Sintaxe de ponto

Você pode usar a sintaxe de ponto para obter e definir propriedades e métodos de um objeto, incluindo instâncias de clipes de filmes e variáveis. A sintaxe de ponto pode ser usada em lugar da sintaxe de barra usada no Flash 4. A sintaxe de barra não é mais a preferida, mas o Flash Player ainda oferece suporte a ela.

## Tipos de dados

O ActionScript do Flash 5 oferece suporte aos seguintes tipos de dados: sequência de caracteres, número, booleano, objeto e clipe de filme. Vários tipos de dados permitem que você use diferentes tipos de informação no ActionScript. Por exemplo, você pode criar matrizes e matrizes associativas.

## Variáveis locais

Você pode declarar variáveis locais que expirem no fim da lista de ações ou da chamada de função. Isso permite gerenciar a memória e reutilizar nomes de variáveis. As variáveis do Flash 4 eram todas permanentes; até mesmo as temporárias, como contadores de loops permaneciam no filme até que este terminasse.

## Funções definidas pelo usuário

Você pode definir funções com parâmetros que retornem valores. Isso permite reutilizar blocos de código em seus scripts. No Flash 4, você podia reutilizar o código por meio da ação `call`, mas não podia passar parâmetros ou retornar valores.

## Objetos predefinidos

Você pode usar objetos predefinidos para acessar e manipular determinados tipos de informação. A seguir, há um exemplo de alguns dos objetos predefinidos:

- O objeto `Math` representa um complemento total de constantes e funções matemáticas internas, como `E` (constante de Euler), `cos` (Cosseno) e `atan` (Arco tangente).
- O objeto `Date` permite obter informações sobre a data e a hora em qualquer sistema que esteja executando o Flash Player.
- O objeto `Sound` permite adicionar sons a um filme e controlá-los enquanto o filme é reproduzido. Por exemplo, você pode ajustar o volume (`setVolume`) ou equilíbrio (`setPan`).
- O objeto `Mouse` permite ocultar o cursor padrão para que você possa usar um cursor personalizado.
- O objeto `MovieClip` permite controlar clipes de filme sem usar uma ação específica, como `tellTarget`. Você pode chamar um método como `play`, `loadMovie` ou `duplicateMovieClip` a partir de um nome de instância por meio da sintaxe de ponto (por exemplo, `myMovieClip.play()`).

## Ações de clipe

Você pode usar a ação `onClipEvent` para atribuir ações diretamente a instâncias de clipes de filmes no Palco. A ação `onClipEvent` tem eventos como `load`, `enterFrame`, `mouseMove` e `data` que permitem criar novos tipos de interatividade avançada.

## Novas ações

Você pode usar novas ações, como `do...while` e `for`, para criar loops complexos. Outras ações novas estão implementadas como métodos do objeto `MovieClip`; por exemplo, `getBounds`, `attachMovie`, `hitTest`, `swapDepths` e `globalToLocal`.

## Clipes Inteligentes

Os Clipes Inteligentes têm scripts internos que você, ou outro desenvolvedor, pode alterar sem usar o painel Ações. Você pode passar valores para um Clipe Inteligente por meio de parâmetros de clipe que podem ser definidos na Biblioteca.

## Depurador

O Depurador permite visualizar e alterar valores de variáveis e de propriedades em um filme que é executado no modo de teste, no Flash Player independente ou em um navegador. Isso permite encontrar problemas facilmente no ActionScript.



## Suporte a XML

O objeto XML predefinido permite converter o ActionScript em documentos XML e passá-los para aplicativos do servidor. Você também pode usar o objeto XML para carregar documentos XML em um filme do Flash e interpretá-los. O objeto XMLSocket predefinido permite criar uma conexão contínua com o servidor para passar dados XML para aplicativos em tempo real.

## Otimização

O Flash 5 executa algumas otimizações simples no código ActionScript para aumentar o desempenho e manter o tamanho dos arquivos. Como resultado dessas otimizações, o Flash 5 freqüentemente produzirá código binário ActionScript menor do que o Flash 4.

## Usando a Ajuda do Flash para ações

O Flash 5 contém ajuda relacionada ao contexto para cada ação disponível no painel Ações. Enquanto cria scripts, você pode obter informações sobre as ações que está usando.

**Para obter ajuda sobre ações:**

- 1 No painel Ações, selecione uma ação na lista Caixa de Ferramentas.
- 2 Clique no botão Ajuda, na parte superior do painel.  
O tópico relacionado à ação é exibido no navegador.



**COMPRE JÁ!**  
1000 apostilas + provas  
+ resumos + teses





# CAPÍTULO 1

## Noções básicas sobre o ActionScript

.....

O ActionScript, a linguagem de script do Flash, adiciona interatividade a um filme. Você pode configurar o filme para que eventos do usuário, como cliques em botões e uso de teclas, ativem scripts que informem ao filme a ação a ser executada. Por exemplo, você pode escrever um script para informar ao Flash que carregue diferentes filmes no Flash Player, dependendo do botão de navegação escolhido.

Imagine o ActionScript como uma ferramenta que permite criar um filme que se comporte exatamente como você deseja. Não é necessário compreender cada uso possível da ferramenta para começar a escrever; se você tiver um objetivo claro, pode começar a criar scripts com ações simples. Você pode incorporar novos elementos da linguagem à medida que os aprende para realizar tarefas mais complexas.

Este capítulo apresenta o ActionScript como uma linguagem de script orientada a objetos e fornece uma visão geral de seus termos. Ele também analisa um exemplo de script, para que você possa começar a se concentrar numa situação mais ampla.

Além disso, este capítulo apresenta o painel Ações, onde é possível criar scripts selecionando elementos do ActionScript ou inserindo texto na janela Script.

## Sobre como escrever scripts em ActionScript

Você pode começar a escrever scripts simples sem saber muito sobre o ActionScript. Tudo de que você precisa é um objetivo; depois, basta escolher as ações corretas. A melhor maneira de aprender como o ActionScript pode ser simples é criando um script. As etapas a seguir anexam um script a um botão que altera a visibilidade de um clipe de filme.

**Para alterar a visibilidade de um clipe de filme:**

- 1 Escolha Janela > Bibliotecas Comuns > Botões e, em seguida, escolha Janela > Bibliotecas Comuns > Clipes de Filme. Coloque um botão e um clipe de filme no Palco.
- 2 Selecione a instância do clipe de filme no Palco e escolha Janela > Painéis > Propriedades da Instância.
- 3 No campo Nome, insira **testMC**.
- 4 Selecione o botão no Palco e escolha Janela > Ações para abrir o painel Ações.
- 5 No painel Ações do Objeto, clique na categoria Ações para abri-la.
- 6 Clique duas vezes na ação `setProperty` para adicioná-la à lista Ações.
- 7 No menu pop-up Propriedade, escolha `_visible` (Visibilidade).
- 8 Para o parâmetro Target, insira **testMC**.
- 9 Para o parâmetro Value, insira **0**.

O código deve ser este:

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

- 10 Escolha Controlar > Testar Filme e clique no botão para fazer o clipe de filme desaparecer.

O ActionScript é uma linguagem de script orientada a objetos. Isso significa que as ações controlam objetos quando ocorre um determinado evento. Neste script, o evento é a liberação do mouse, o objeto é a instância do clipe de filme MC e a ação é `setProperty`. Quando o usuário clica no botão da tela, um evento `release` ativa um script que define a propriedade `_visible` para o objeto MC como `false` e faz com que ele se torne invisível.

O painel Ações pode ser usado para auxiliá-lo na criação de scripts simples. Para usar todos os recursos do ActionScript, é importante compreender como a linguagem funciona: os conceitos, os elementos e as regras que a linguagem usa para organizar informações e criar filmes interativos.

Esta seção explica o fluxo de trabalho do ActionScript, os conceitos fundamentais de scripts orientados a objetos, os objetos do Flash e o fluxo de scripts. Ela também descreve onde os scripts residem em um filme do Flash.

## Sobre como planejar e depurar scripts

Quando você escreve scripts para um filme inteiro, a quantidade e variedade de scripts pode ser grande. Decidir sobre as ações que serão usadas, como estruturar scripts de maneira eficiente e o local no qual os scripts devem ser colocados requer planejamento e teste cuidadosos, especialmente à medida que o filme se torna mais complexo.

Antes de começar a escrever scripts, formule seu objetivo e defina o que deseja alcançar. Isso é tão importante (e normalmente tão demorado) quanto desenvolver storyboards para seu trabalho. Comece escrevendo o que deve acontecer no filme, como neste exemplo:

- Desejo criar todo o meu site com o Flash.
- Os visitantes do site deverão fornecer seus nomes, que serão reutilizados em mensagens no site.
- O site terá uma barra de navegação flutuante com botões que ligam a cada seção do site.
- Quando um botão for clicado, a nova seção desaparecerá em direção ao centro do Palco.
- Uma cena terá um formulário de contato com o nome do usuário já preenchido.

Quando você souber o que deseja, poderá criar os objetos necessários e escrever os scripts para controlar esses objetos.

Fazer com que os scripts funcionem da forma como você quer é demorado, muitas vezes é preciso mais de um ciclo de definição, teste e depuração. A melhor abordagem é começar de forma simples e testar o trabalho freqüentemente. Quando fizer uma parte de um script funcionar, escolha Salvar como para salvar uma versão do arquivo (por exemplo, myMovie01.fla) e comece a escrever a próxima parte. Dessa forma, você identificará bugs com eficiência e garantirá que o ActionScript esteja sólido enquanto você escreve scripts mais complexos.

## Sobre como criar scripts orientados a objetos

Nos scripts orientados a objetos, você organiza informações em grupos chamados *classes*. Podem ser criadas várias instâncias de uma classe, chamadas *objects*, para uso nos scripts. Você pode usar as classes predefinidas do ActionScript e criar suas próprias classes.

Ao criar uma classe, você define todas as *propriedades* (características) e todos os *métodos* (comportamentos) de cada objeto que ela criar, exatamente como os objetos reais são definidos. Por exemplo, uma pessoa tem propriedades (como sexo, altura e cor do cabelo) e métodos (como falar, andar e jogar). Neste exemplo, “pessoa” é uma classe e cada indivíduo é um objeto, ou uma *instância* dessa classe.

No ActionScript, os objetos podem conter dados ou podem ser representados graficamente no Palco como clipes de filme. Todos os clipes de filme são instâncias da classe MovieClip predefinida. Cada instância de clipe de filme contém todas as propriedades (por exemplo, `_height`, `_rotation`, `_totalframes`) e todos os métodos (por exemplo, `gotoAndPlay`, `loadMovie`, `startDrag`) da classe MovieClip.

Para definir uma classe, você cria uma função especial chamada *função construtora*; as classes predefinidas têm funções construtoras que já estão definidas. Por exemplo, se desejar informações sobre um ciclista em seu filme, você pode criar uma função construtora, `Biker`, com as propriedades `time` e `distance` e o método `rate`, que informa a velocidade do ciclista:

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



Em seguida, você poderá criar cópias, isto é, instâncias da classe. O código a seguir cria instâncias do objeto `Biker` chamadas `emma` e `hamish`.

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

As instâncias também podem se comunicar umas com as outras. Para o objeto `Biker`, você deve criar um método chamado `shove` que permite que um ciclista empurre outro ciclista. (A instância `emma` pode chamar esse método `shove` se `hamish` se aproximar muito). Para passar informações para um método, você usa parâmetros (argumentos): por exemplo, o método `shove` pode ter os parâmetros *who* e *howFar*. Neste exemplo `emma` empurra `hamish` 10 pixels:

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

No script orientado a objetos, as classes podem receber propriedades e métodos umas das outras de acordo com uma ordem específica; isso é chamado de *herança*. A herança pode ser usada para estender ou redefinir as propriedades e os métodos de uma classe. Uma classe que herda propriedades e métodos de outra classe é chamada de *subclasse*. Uma classe que passa propriedades e métodos para outra classe é chamada de *superclasse*. Uma classe pode ser uma subclasse e uma superclasse ao mesmo tempo.

## Sobre o objeto `MovieClip`

As classes predefinidas do ActionScript são chamadas de *objetos*. Cada objeto permite que você acesse determinados tipos de informação. Por exemplo, o objeto `Date` tem métodos (por exemplo, `getFullYear`, `getMonth`) que permitem ler informações do relógio do sistema. O objeto `Sound` tem métodos (por exemplo, `setVolume`, `setPan`) que permitem controlar o som de um filme. O objeto `MovieClip` tem métodos que permitem controlar instâncias de clipes de filme (por exemplo, `play`, `stop` e `getURL`) além de obter e definir informações sobre suas propriedades (por exemplo, `_alpha`, `_framesloaded`, `_visible`).

Os clipes de filme são os objetos mais importantes de um filme do Flash, pois têm linhas de tempo que são executadas independentemente umas das outras. Por exemplo, se a linha de tempo principal tiver somente um quadro e um clipe de filme tiver dez quadros, cada quadro do clipe de filme ainda será reproduzido. Isso permite que as instâncias ajam como objetos autônomos que podem se comunicar uns com os outros.

Cada instância de clipe de filme tem um nome exclusivo, para que possa ser usada como alvo de uma ação. Por exemplo, pode haver várias instâncias no Palco (por exemplo, `leftClip` e `rightClip`) e somente uma deve ser reproduzida de cada vez. Para atribuir uma ação que faça uma determinada instância ser reproduzida, é preciso usar seu nome. No exemplo a seguir, o nome do clipe de filme é `leftClip`:

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



Os nomes de instâncias também permitem que você duplique, remova e arraste clipes de filme enquanto um filme é reproduzido. O exemplo a seguir duplica a instância `cartItem` para preencher uma cesta de compras com o número de itens comprados.

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

Os clipes de filme têm propriedades cujos valores podem ser definidos e recuperados dinamicamente com o ActionScript. A alteração e leitura dessas propriedades pode alterar a aparência e identidade de um clipe de filme além de ser a chave para criar interatividade. Por exemplo, o script a seguir usa a ação `setProperty` para definir a transparência (configuração `alpha`) da instância `navigationBar` como 10.

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

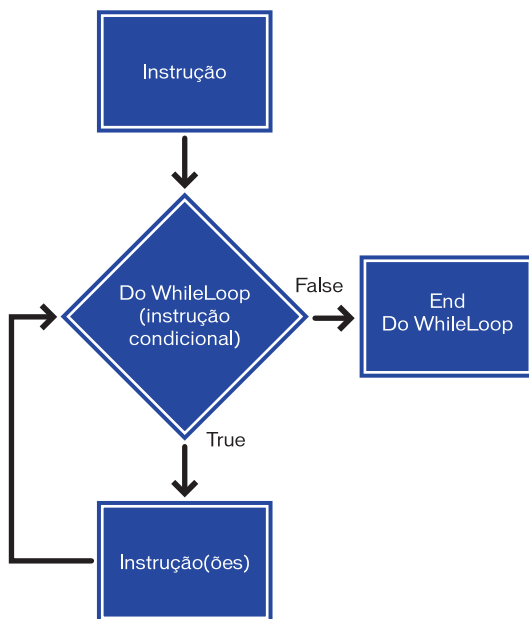
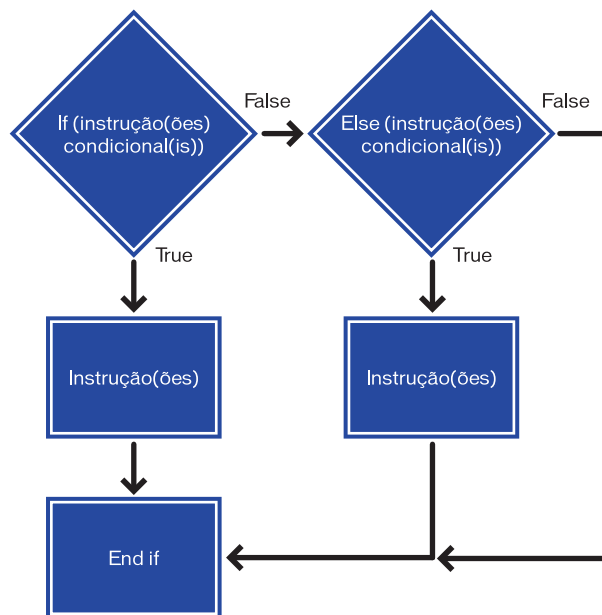
Para obter mais informações sobre outros tipos de objetos, consulte “Usando objetos predefinidos”, na página 79.

## Como é o fluxo dos scripts

O ActionScript segue um fluxo lógico. O Flash executa comandos ActionScript começando pelo primeiro comando e continuando em ordem até alcançar o comando final ou um comando que informe ao ActionScript que vá para outro local.

Algumas ações que enviam o ActionScript para outro local diferente do próximo comando são os comandos `if`, os loops `do...while` e a ação `return`.





Um comando `if` é chamado de condicional ou de “ramificação lógica”, pois controla o fluxo de um script com base na avaliação de uma determinada condição. Por exemplo, o código a seguir verifica se o valor da variável `number` é menor ou igual a 10. Se a verificação retornar `true` (por exemplo, o valor de `number` for 5), a variável `alert` será definida e exibirá seu valor em um campo de texto de entrada, como mostrado a seguir:

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

Também é possível adicionar comandos `else` para criar um comando condicional mais complexo. No exemplo a seguir, se a condição retornar `true` (por exemplo, o valor de `number` for 3), o comando entre o primeiro conjunto de chaves será executado e a variável `alert` será definida na segunda linha. Se a condição retornar `false` (por exemplo, o valor de `number` for 30), o primeiro bloco de código será ignorado e o comando entre as chaves após o comando `else` será executado, como mostrado a seguir:

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

Para obter mais informações, consulte “Usando comandos `if`”, na página 71.

Os loops repetem uma ação por um determinado número de vezes ou até que uma condição seja atendida. No exemplo a seguir, um clipe de filme é duplicado cinco vezes:

```
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);
```

Para obter informações detalhadas, consulte “Repetindo uma ação”, na página 72.

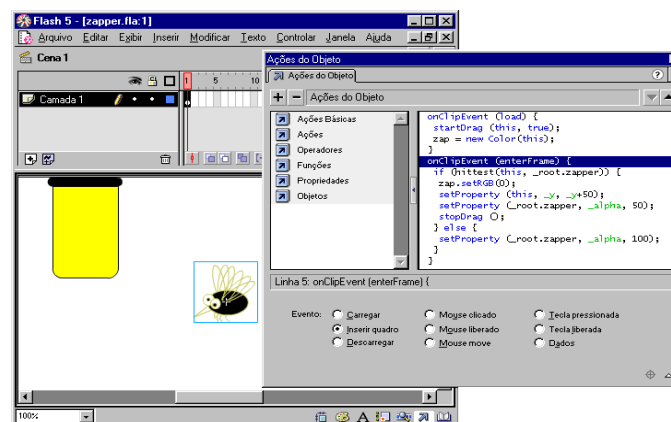
## Controlando o momento em que o ActionScript é executado

Quando você escreve um script, usa o painel Ações. Ele permite anexar o script a um quadro na linha de tempo principal ou na linha de tempo de qualquer clipe de filme, ou a um botão ou clipe de filme no Palco.

O Flash executa ações em momentos diferentes, dependendo de a que estão anexadas:

- As ações anexadas a um quadro são executadas quando a reprodução entra nesse quadro.
- As ações anexadas a um botão são incluídas em uma ação manipuladora on.
- As ações anexadas a um clipe de filme são incluídas em uma ação manipuladora onClipEvent.

As ações onClipEvent e on são chamadas de manipuladoras porque “manipulam”, ou gerenciam, um evento. (Um evento é uma ocorrência como um movimento do mouse, o pressionamento de uma tecla ou o carregamento de um clipe de filme). As ações de clipes de filme e de botão são executadas quando o evento especificado pelo manipulador ocorre. É possível anexar mais de um manipulador a um objeto, se desejar que as ações sejam executadas quando eventos diferentes ocorrerem. Para obter mais informações, consulte o capítulo 3, “Criando interatividade com o ActionScript”.



*Diversos manipuladores onClipEvent anexados a um clipe de filme no Palco.*

## Terminologia do ActionScript

Como qualquer outra linguagem de scripts, o ActionScript usa terminologia específica que segue determinadas regras de sintaxe. A lista a seguir fornece uma introdução a itens do ActionScript importantes, em ordem alfabética. Esses termos e a sintaxe que os regula são discutidos com mais detalhes no Capítulo 2, “Escrevendo scripts com o ActionScript”.

As **ações** são comandos que informam a um filme que faça algo enquanto é reproduzido. Por exemplo, `gotoAndStop` envia a reprodução para um quadro ou rótulo específico. Neste manual, os termos *ação* e *comando* são equivalentes.

Os **argumentos**, também chamados de parâmetros, são espaços reservados que permitem passar valores para funções. Por exemplo, a função a seguir, chamada de `welcome`, usa os dois valores que recebe nos argumentos `firstName` e `hobby`:

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

As **classes** são tipos de dados que podem ser criados para definir um novo tipo de objeto. Para definir uma classe de objeto, você cria uma função construtora.

As **constantes** são elementos que não se alteram. Por exemplo, a constante `TAB` sempre tem o mesmo significado. As constantes são úteis para comparar valores.

As **construtoras** são funções usadas para definir as propriedades e os métodos de uma classe. Por exemplo, o código a seguir cria uma nova classe `Circle` por meio da criação de uma função construtora chamada `Circle`:

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

Os **tipos de dados** são um conjunto de valores e operações que podem ser executadas neles. Os tipos de dados ActionScript são `string`, `number`, valores `true` e `false` (booleanos), `object` e `movie clip`. Para obter mais detalhes sobre esses elementos de linguagem, consulte “Sobre tipos de dados”, na página 54.

Os **eventos** são ações que ocorrem enquanto um filme é reproduzido. Por exemplo, são gerados diferentes eventos quando um clipe de filme é carregado, quando a reprodução entra em um quadro, quando o usuário clica em um botão ou clipe de filme ou quando digita no teclado.

As **expressões** são qualquer parte de um comando que produza um valor. Por exemplo, `2 + 2` é uma expressão.

As **funções** são blocos de código reutilizáveis para os quais podem ser passados argumentos (parâmetros) e que podem retornar um valor. Por exemplo, o nome de uma propriedade e o nome de uma instância de um clipe de filme são passados para a função `getProperty` e ela retorna o valor da propriedade. A função `getVersion` retorna a versão do Flash Player que reproduz o filme no momento.

Os **manipuladores** são ações especiais que “manipulam” ou gerenciam um evento, como `mouseDown` ou `load`. Por exemplo, `on` (`onMouseEvent`) e `onClipEvent` são manipuladores do ActionScript.

Os **identificadores** são nomes usados para indicar uma variável, uma propriedade, um objeto, uma função ou um método. O primeiro caractere deve ser uma letra, sublinhado (`_`) ou sinal de cifrão (`$`). Cada caractere posterior deve ser uma letra, número, sublinhado (`_`) ou sinal de cifrão (`$`). Por exemplo, `firstName` é o nome de uma variável.

As **instâncias** são objetos que pertencem a certas classes. Cada instância de uma classe contém todas as propriedades e métodos dessa classe. Todos os clipes de filme são instâncias com propriedades (por exemplo, `_alpha` e `_visible`) e métodos (por exemplo, `gotoAndPlay` e `getURL`) da classe `MovieClip`.

Os **nomes de instâncias** são nomes exclusivos que permitem fazer referências a instâncias de clipes de filme em scripts. Por exemplo, um símbolo principal na Biblioteca pode ser chamado de `contador` e as duas instâncias desse símbolo no filme podem ter nomes de instâncias `scorePlayer1` e `scorePlayer2`. O código a seguir define uma variável chamada `score` dentro de cada instância de clipe de filme por meio de nomes de instâncias:

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

As **palavras-chave** são palavras reservadas que têm significado especial. Por exemplo, `var` é uma palavra-chave usada para declarar variáveis locais.

Os **métodos** são funções atribuídas a um objeto. Depois que uma função é atribuída, pode ser chamada como método desse objeto. Por exemplo, no código a seguir, `clear` torna-se um método do objeto `controller`:

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



Os **objetos** são coleções de propriedades; cada objeto tem seu próprio nome e valor. Os objetos permitem que você acesse determinados tipos de informação. Por exemplo, o objeto predefinido `Date` fornece informações sobre o relógio do sistema.

Os **operadores** são termos que calculam um novo valor de um ou mais valores. Por exemplo, o operador de adição (+) adiciona dois ou mais valores para produzir um novo valor.

Os **caminhos de destino** são endereços hierárquicos de nomes de instâncias de clipes de filme, variáveis e objetos em um filme. Uma instância de clipe de filme pode ser nomeada no painel Instância. A linha de tempo principal sempre tem o nome `_root`. Você pode usar um caminho de destino para direcionar uma ação de um clipe de filme e obter ou definir o valor de uma variável. Por exemplo, o comando a seguir é o caminho de destino da variável `volume` dentro do clipe de filme `stereoControl`:

```
_root.stereoControl.volume
```

As **propriedades** são atributos que definem um objeto. Por exemplo, `_visible` é uma propriedade de todos os clipes de filme que define se eles estão visíveis ou ocultos.

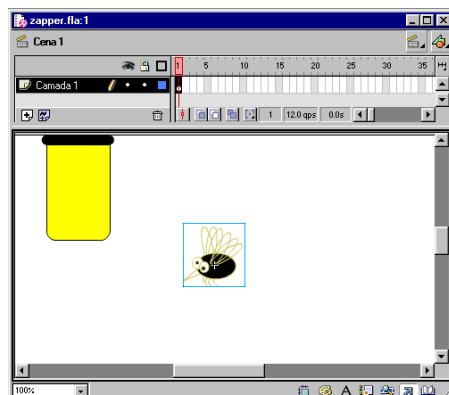
As **variáveis** são identificadores que mantêm valores de qualquer tipo de dado. As variáveis podem ser criadas, alteradas e atualizadas. Os valores que armazenam podem ser recuperados para uso em scripts. No exemplo a seguir, os identificadores no lado esquerdo do sinal de igual são variáveis.

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



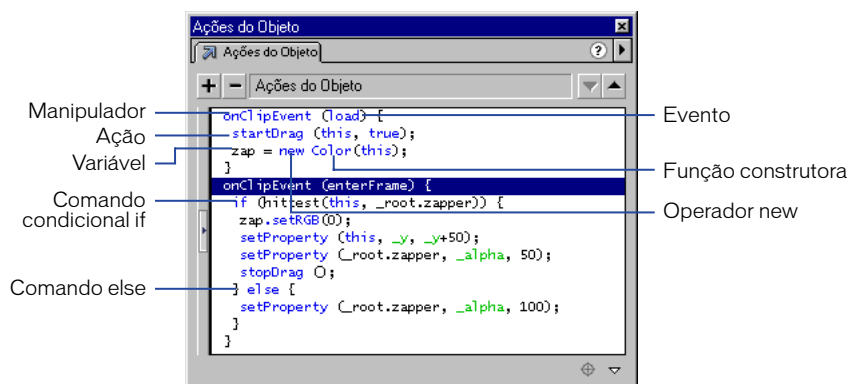
## Analisando um script simples

Nesta amostra de filme, quando um usuário arrasta o inseto para a armadilha, ele fica preto e cai, e a armadilha pisca. O filme tem um quadro de duração e contém dois objetos, a instância do clipe de filme do inseto e a instância do clipe de filme da armadilha. Cada clipe de filme também contém um quadro.



*Instâncias de clipe de filme bug e zapper no Palco, no quadro 1.*

Só há um script no filme; ele está anexado à instância bug, como no painel Ações do Objeto, mostrado a seguir:



*Painel Ações do Objeto com o script anexado à instância bug.*



É necessário que todos os objetos sejam clipes de filme, para que você possa dar nomes de instâncias a eles no painel Instância e manipulá-los com o ActionScript. O nome da instância do inseto é `bug` e o nome da instância da armadilha é `zapper`. No script, a referência ao inseto é feita como `this`, pois o script está anexado ao inseto e a palavra reservada `this` refere-se ao objeto que a chama.

Há dois manipuladores `onClipEvent` com dois eventos diferentes: `load` e `enterFrame`. As ações no comando `onClipEvent(load)` são executadas somente uma vez, quando o filme é carregado. As ações no comando `onClipEvent(enterFrame)` são executadas toda vez que a reprodução entra em um quadro. Mesmo em um filme com apenas um quadro, a reprodução ainda entra nesse quadro repetidamente e o script é executado, também repetidamente. As ações a seguir ocorrem dentro de cada manipulador `onClipEvent`:

**`onClipEvent(load)`** Uma ação `startDrag` torna o clipe de filme do inseto flutuante. Uma instância do objeto `Color` é criada com o operador `new` e com a função construtora de `Color`, `Color`, e atribuída à variável `zap`:

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

**`onClipEvent(enterFrame)`** Um comando condicional `if` avalia uma ação `hitTest` para verificar se a instância do inseto (`this`) está tocando a instância da armadilha (`_root.zapper`). Há dois resultados possíveis para a avaliação, `true` ou `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 a ação `hitTest` retornar `true`, o objeto `zap` criado pelo evento `load` é usado para definir a cor do inseto como preta. A propriedade `y` do inseto (`_y`) é definida como ela mesma mais 50, para que o inseto caia. A transparência da armadilha (`_alpha`) é definida como 50, para que fique esmaecida. A ação `stopDrag` impede que o inseto seja arrastado.

Se a ação `hitTest` retornar `false`, a ação após o comando `else` é executada e o valor `_alpha` da armadilha é definido como 100. Isso faz a armadilha piscar, pois seu valor `_alpha` vai de um estado inicial (100) a um estado em que é ativada (50) e volta ao estado inicial. A ação `hitTest` retorna `false` e os comandos `else` são executados depois que o inseto é atingido e cai.

Para reproduzir o filme, consulte a *Ajuda do Flash*.



## Usando o painel Ações

O painel Ações permite criar e editar ações para um objeto ou quadro com dois modos de edição diferentes. Você pode selecionar ações predefinidas na lista Caixa de Ferramentas, arrastar e soltar ações e usar botões para excluir ou reorganizar ações. No Modo Normal, você pode escrever ações usando campos de parâmetros (argumentos) que solicitam os argumentos corretos. No Modo Especialista, você pode escrever e editar ações diretamente em uma caixa de texto, de forma semelhante a escrever um script com um editor de texto.

### Para exibir o painel Ações:

Escolha Janela > Ações.

A seleção de uma instância de um botão ou de um clipe de filme ativa o painel Ações. Seu título é alterado para Ações do Objeto, se for selecionado um botão ou clipe de filme, e para Ações de Quadro, se for selecionado um quadro.

### Para selecionar um modo de edição:

- 1 Com o painel Ações exibido, clique na seta no canto superior direito do painel para exibir o menu pop-up.
- 2 No menu pop-up, escolha Modo Normal ou Modo Especialista.

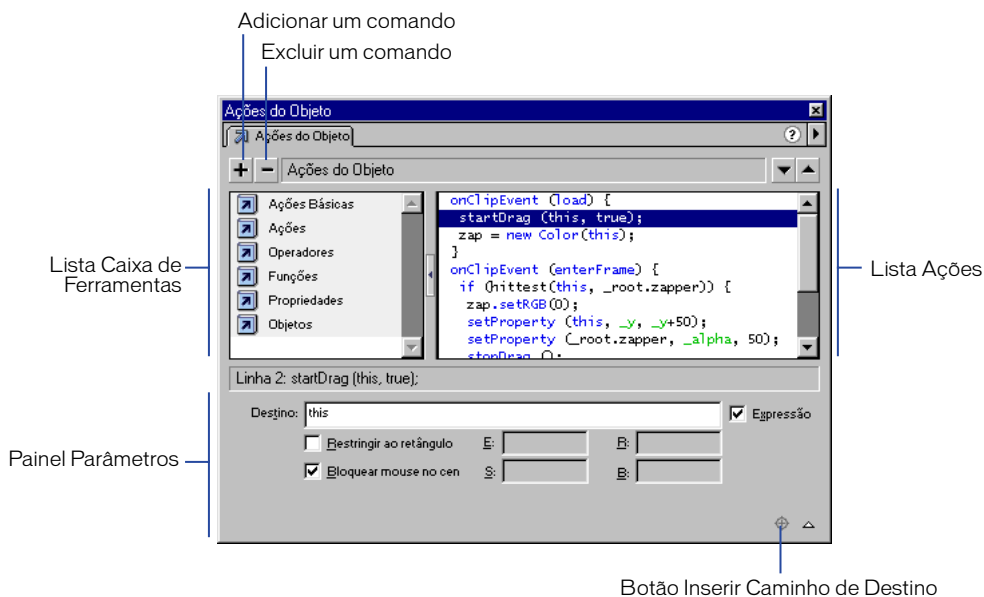
Cada script mantém seu próprio modo. Por exemplo, você pode criar o script de uma instância de um botão no Modo Normal e outra no Modo Especialista. A alternância do botão selecionado alterna o estado do modo do painel.

## Modo Normal

No Modo Normal, as ações são criadas selecionando-as em uma lista no lado esquerdo do painel, chamada de Caixa de Ferramentas. A lista Caixa de Ferramentas contém as categorias Ações Básicas, Ações, Operadores, Funções, Propriedades e Objetos. A categoria Ações Básicas contém as ações mais simples do Flash e só está disponível no Modo Normal. As ações selecionadas são listadas no lado direito do painel, na lista Ações. É possível adicionar ou alterar a ordem dos comandos das ações; também é possível inserir parâmetros (argumentos) para ações em campos de parâmetros na parte inferior do painel.



No Modo Normal, você pode usar os controles do painel Ações para excluir ou alterar a ordem dos comandos na lista Ações. Esses controles são especialmente úteis para gerenciar ações de quadros ou de botões que tenham vários comandos.



*Painel Ações no Modo Normal.*

**Para selecionar uma ação:**

- 1 Clique em uma categoria de Ações na caixa de ferramentas para exibir as ações dessa categoria.
- 2 Clique duas vezes em uma ação ou arraste-a para a janela Script.

**Para usar os campos de parâmetros:**

- 1 Clique no botão Parâmetros, no canto inferior direito do painel Ações, para exibir os campos.
- 2 Selecione uma ação e insira novos valores nos campos de parâmetros para alterar os parâmetros de ações existentes.



**Para inserir o caminho de destino de um clipe de filme:**

- 1 Clique no botão Caminho de Destino, no canto inferior direito do painel Ações, para exibir a caixa de diálogo Inserir Caminho de Destino.
- 2 Selecione um clipe de filme na lista de exibição.

**Para mover um comando para cima e para baixo na lista:**

- 1 Selecione um comando na lista Ações.
- 2 Clique nos botões Seta para Cima ou Seta para Baixo.

**Para excluir uma ação:**

- 1 Selecione um comando na lista Ações.
- 2 Clique no botão Excluir (-).

**Para alterar os parâmetros de ações existentes:**

- 1 Selecione um comando na lista Ações.
- 2 Insira novos valores nos campos de parâmetros.

**Para redimensionar a lista Caixa de Ferramentas ou Ações, escolha uma destas opções:**

- Arraste a barra divisora vertical que aparece entre as listas Caixa de Ferramentas e Ações.
- Clique duas vezes na barra divisora para recolher a lista Caixa de Ferramentas; clique duas vezes na barra novamente para reexibi-la.
- Clique no botão Seta para Esquerda ou Seta para a Direita, na barra divisora, para expandir ou recolher a lista.

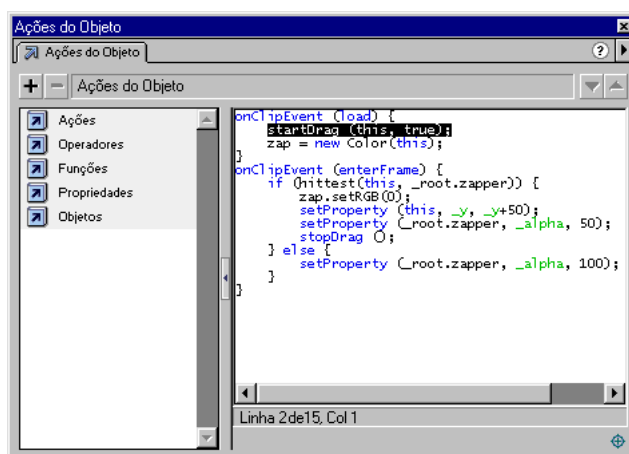
Mesmo quando a lista Caixa de Ferramentas está oculta, você pode acessar seus itens usando o botão com sinal de adição (+) no canto superior esquerdo do painel Ações.

## Modo Especialista

No Modo Especialista é possível criar ações inserindo ActionScript na caixa de texto à direita do painel ou selecionando ações na lista Caixa de Ferramentas à esquerda. Você pode editar ações, inserir parâmetros para ações ou excluir ações diretamente na caixa de texto, da mesma forma como faz para criar scripts em um editor de texto.

O Modo Especialista permite que usuários avançados do ActionScript editem seus scripts com um editor de texto, da mesma forma como fariam no JavaScript ou VBScript. O Modo Especialista difere do Modo Normal nos seguintes pontos:

- A seleção de um item no menu pop-up Adicionar ou na lista Caixa de Ferramentas faz com que o item seja inserido na área de edição de texto.
- Nenhum campo de parâmetros é exibido.
- No painel de botões, somente o botão com sinal de adição (+) funciona.
- Os botões Seta para Cima e Seta para Baixo permanecem inativos.



*Painel Ações no Modo Especialista.*

## Alternando entre modos de edição

A alteração de modos de edição ao escrever um script pode alterar a formatação do script. Por isso, é melhor usar um modo de edição por script.

Quando você alterna do Modo Normal para o Modo Especialista, o recuo e a formatação são mantidos. Apesar de ser possível converter scripts do Modo Normal com erros para o Modo Especialista, só é possível exportar os scripts depois que os erros forem corrigidos.

A alternância do Modo Especialista para o Modo Normal é um pouco mais complexa.

- Quando você alterna para o Modo Normal, o Flash reformata o script e elimina qualquer espaço em branco e recuo que você tenha adicionado.
- Se você alternar para o Modo Normal e voltar para o Modo Especialista, o Flash reformatará o script de acordo com a aparência no Modo Normal.
- Não é possível exportar ou converter para o Modo Normal os scripts do Modo Especialista que contenham erros; se você tentar convertê-los, receberá uma mensagem de erro.

### Para alternar os modos de edição:

Escolha Modo Normal ou Modo Especialista no menu pop-up no canto superior direito do painel Ações. Uma marca de seleção indica o modo selecionado.

### Para definir uma preferência de modo de edição:

- 1 Escolha Editar > Preferências.
- 2 Selecione a guia Geral.
- 3 Na seção Painel de Ações, selecione Modo Normal ou Modo Especialista no menu pop-up.

## Usando um editor externo

Apesar do Modo Especialista do painel Ações oferecer mais controle durante a edição de ActionScript, também é possível editar um script fora do Flash. Em seguida, você pode usar a ação `include` para adicionar os scripts escritos no editor externo a um script no Flash.

Por exemplo, o comando a seguir importa um arquivo de scripts:

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

O texto do arquivo de script substitui a ação `include`. O arquivo de texto deve estar presente quando o filme for exportado.

**Para adicionar scripts escritos em um editor externo a um script no Flash:**

- 1 Arraste `include` da lista Caixa de Ferramentas para a janela Script.
- 2 Insira o caminho para o arquivo externo na caixa Caminho.

O caminho deve ser relativo ao arquivo FLA. Por exemplo, se `myMovie.fl` e `externalfile.as` estiverem na mesma pasta, o caminho será `externalfile.as`. Se `externalfile.as` estiver em uma subpasta chamada `scripts`, o caminho será `scripts/externalfile.as`.

## Escolhendo opções do painel Ações

O painel Ações permite trabalhar com scripts de várias maneiras. Você pode alterar o tamanho da fonte na janela Script. Você pode importar um arquivo de texto que contenha ActionScript para o painel Ações e exportar ações com um arquivo de texto, procurar e substituir texto em um script e usar o realce de sintaxe para facilitar a leitura dos scripts e a detecção de erros. O painel Ações exibe realces de aviso para erros de sintaxe e incompatibilidades de versão do Flash Player. Ele também realça elementos do ActionScript *reprovados* ou que não sejam adequados.

Essas opções do painel Ações estão disponíveis no Modo Normal e no Modo Especialista, exceto quando indicado.

**Para alterar o tamanho da fonte na janela Script:**

- 1 No menu pop-up no canto superior direito do painel Ações, escolha Tamanho da Fonte.
- 2 Selecione Pequena, Normal ou Grande.



**Para importar um arquivo de texto que contenha ActionScript:**

- 1 No menu pop-up no canto superior direito do painel Ações, escolha Importar de Arquivo.
- 2 Selecione um arquivo de texto que contenha ActionScript e clique em Abrir.

**Observação:** Os scripts com erros de sintaxe só podem ser importados no Modo Especialista. No Modo Normal, você receberá uma mensagem de erro.

**Para exportar ações como um arquivo de texto.**

- 1 No menu pop-up no canto superior direito do painel Ações, escolha Exportar como Arquivo.
- 2 Escolha um local em que o arquivo deva ser salvo e clique em Salvar.

**Para imprimir ações:**

- 1 No menu pop-up no canto superior direito do painel Ações, escolha Imprimir.  
A caixa de diálogo Imprimir é exibida.
- 2 Escolha Opções e clique em Imprimir.

**Observação:** O arquivo impresso não incluirá informações sobre o arquivo Flash que o originou. É recomendado incluir essas informações em uma ação `comment` no script.

**Para procurar um texto em um script, escolha uma opção no menu pop-up do painel Ações.**

- Para ir para uma linha específica de um script, escolha Ir para Linha.
- Para localizar texto, escolha Localizar.
- Para localizar texto novamente, escolha Localizar Novamente.
- Para localizar e substituir texto, escolha Substituir.

No Modo Especialista, Substituir varre todo o corpo de texto em um script. No Modo Normal, Substituir procura e substitui texto somente no campo de parâmetro de cada ação. Por exemplo, não é possível substituir todas as ações `gotoAndPlay` por `gotoAndStop` no Modo Normal.

**Observação:** Use o comando Localizar ou Substituir para pesquisar a lista Ações atual. Para pesquisar o texto em todos os scripts de um filme, use o Movie Explorer. Para obter mais informações, consulte *Usando o Flash*.

## Realçando e verificando a sintaxe

O realce da sintaxe identifica certos elementos do ActionScript com cores específicas. Isso ajuda a impedir erros de sintaxe, como uso incorreto de maiúsculas e minúsculas em palavras-chave. Por exemplo, se a palavra-chave `typeof` tiver sido escrita como `type0f`, não ficará azul e você poderá reconhecer o erro. Quando o realce de sintaxe está ativado, o texto é realçado desta maneira:

- Palavras-chave e identificadores predefinidos (por exemplo, `gotoAndStop`, `play` e `stop`) têm a cor azul.
- Propriedades têm a cor verde.
- Comentários têm a cor magenta.
- Sequências de caracteres entre aspas são cinza.

### Para ativar ou desativar o realce da sintaxe:

No menu pop-up no canto superior direito do painel Ações, escolha Sintaxe de Cor. Uma marca de seleção indica que essa opção está ativada. Todos os scripts no seu filme serão realçados.

Recomenda-se verificar erros da sintaxe de um script antes de exportar um filme. Os erros são reportados na janela Saída. Você pode exportar um filme que contenha scripts errados. Entretanto, será avisado de que os scripts com erros não foram exportados.

### Para verificar os erros da sintaxe do script atual:

No menu pop-up no canto superior direito do painel Ações, escolha Verificar Sintaxe.

## Sobre o realce de erros

Todos os erros de sintaxe são realçados com um fundo vermelho na janela Script no Modo Normal. Isso facilita a detecção de problemas. Se você mover o ponteiro do mouse sobre uma ação com sintaxe incorreta, uma dica de ferramenta exibirá a mensagem de erro associada a essa ação. Quando você seleciona a ação, a mensagem de erro também é exibida no título do painel da área de parâmetros.

No Modo Normal, todas as incompatibilidades de exportação de ActionScript são realçadas com um fundo amarelo na janela Script. Por exemplo, se a versão de exportação do Flash Player estiver definida como Flash 4, o ActionScript que é suportado somente no Flash 5 Player será realçado em amarelo. A versão de exportação é determinada na caixa de diálogo Configurações de Publicação.

Todas as ações reprovadas são realçadas com um fundo verde na caixa de ferramentas. Elas são realçadas somente quando a versão de exportação do Flash estiver definida como Flash 5.



**Para definir a versão de exportação do Flash Player:**

- 1 Escolha Arquivo > Configurações de Publicação.
- 2 Clique na guia Flash.
- 3 Escolha uma versão de exportação no menu pop-up Versão.

**Observação:** Não é possível desativar o realce de erros da sintaxe.

**Para mostrar o realce da sintaxe obsoleta:**

No menu pop-up do painel Ações, escolha Mostrar Sintaxe Obsoleta.

Para obter uma lista completa de todas as mensagens de erro, consulte o apêndice C, “Mensagens de erro”.

## Atribuindo ações a objetos

Você pode atribuir uma ação a um botão ou clipe de filme para que uma ação seja executada quando o usuário clicar em um botão ou rolar o ponteiro do mouse sobre ele, ou quando o clipe de filme for carregado ou atingir um determinado quadro. A ação é atribuída a uma instância do botão ou do clipe de filme; outras instâncias do símbolo não são afetadas. (Para atribuir uma ação a um quadro, consulte “Atribuindo ações a quadros”, na página 47).

Ao atribuir uma ação a um botão, você deve aninhá-la dentro de um manipulador `on(mouse event)` e especificar os eventos do mouse ou do teclado que a ativam. Quando você atribui uma ação a um botão no Modo Normal, o manipulador `on(mouse event)` é inserido automaticamente.

Ao atribuir uma ação a um clipe de filme, você deve aninhá-la dentro de um manipulador `onClipEvent` e especificar os eventos do clipe que a ativam. Quando você atribui uma ação a um clipe de filme no Modo Normal, o manipulador `on(mouse event)` é inserido automaticamente.

Os comandos a seguir descrevem como atribuir ações a objetos pelo painel Ações no Modo Normal.

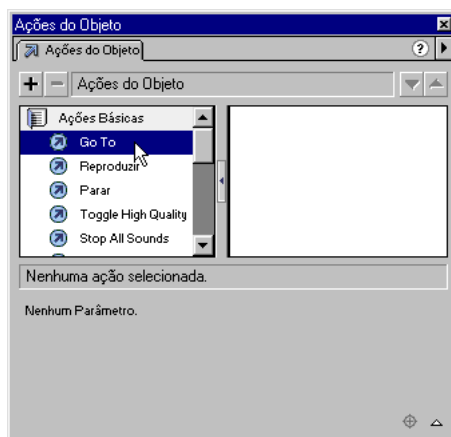
Após atribuir uma ação, use o comando Controlar > Testar Filme para testar seu funcionamento. A maioria das ações não funcionará no modo de edição.

**Para atribuir uma ação a um botão ou clipe de filme:**

- 1 Selecione uma instância de botão ou de clipe de filme e escolha Janela > Ações.  
Se a seleção não for um botão, uma instância de clipe de filme ou um quadro, ou se a seleção incluir vários objetos, o painel Ações fica esmaecido.
- 2 No menu pop-up no canto superior direito do painel Ações do Objeto, escolha Modo Normal.

**3** Para atribuir uma ação, escolha uma destas opções:

- Clique na pasta Ações, na lista Caixa de Ferramentas no lado esquerdo do painel Ações. Clique duas vezes em uma ação para adicioná-la à lista Ações no lado direito do painel.
- Arraste uma ação da lista Caixa de Ferramentas para a lista Ações.
- Clique no botão com sinal de adição (+) e escolha uma ação no menu pop-up.
- Use o atalho do teclado listado ao lado de cada ação no menu pop-up.



*Selecionando um objeto na caixa de ferramentas no Modo Normal.*

**4** Nos campos de parâmetros, na parte inferior do painel, selecione os parâmetros da ação conforme necessário.

Os parâmetros variam de acordo com a ação escolhida. Para obter informações detalhadas sobre os parâmetros necessários para cada ação, consulte o capítulo 7, “Dicionário do ActionScript”. Para inserir um caminho de destino para um clipe de filme em um campo de parâmetro, clique no botão Parâmetro de Destino, no canto inferior direito do painel Ações. Para obter mais informações, consulte o Capítulo 4, “Trabalhar com clipes de filme”.

**5** Repita as etapas 3 e 4 para atribuir ações adicionais, conforme necessário.

**Para testar a ação de um objeto:**

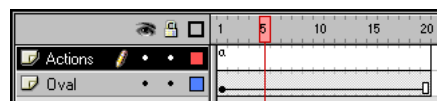
Escolha Controlar > Testar Filme.

## Atribuindo ações a quadros

Para que um filme faça algo quando atingir um quadro-chave, você atribui uma ação de quadro ao quadro-chave. Por exemplo, para criar um loop na linha de tempo entre os quadros 20 e 10, você deve adicionar a seguinte ação de quadro ao quadro 20:

```
gotoAndPlay (10);
```

Recomenda-se colocar ações de quadro em uma camada separada. Os quadros com ações exibem um pequeno *a* na linha de tempo.



Um “a” em um quadro-chave indica uma ação de quadro.

Após atribuir uma ação, escolha Controlar > Testar Filme para testar seu funcionamento. A maioria das ações não funcionará no modo de edição.

Os comandos a seguir descrevem como atribuir ações de quadros pelo painel Ações no Modo Normal. (Para obter mais informações sobre como atribuir uma ação a um botão ou clipe de filme, consulte “Atribuindo uma ação ou um método”, na página 127).

**Para atribuir uma ação a um quadro-chave:**

- 1** Selecione um quadro-chave na linha de tempo e escolha Janela > Ações.  
Se um quadro selecionado não for um quadro-chave, a ação será atribuída ao quadro-chave anterior. Se a seleção não for um quadro ou se incluir vários quadros-chave, o painel Ações ficará esmaecido.
- 2** No menu pop-up no canto superior direito do painel Ações de Quadro, escolha Modo Normal.



**3** Para atribuir uma ação, escolha uma destas opções:

- Clique na pasta Ações, na lista Caixa de Ferramentas no lado esquerdo do painel Ações. Clique duas vezes em uma ação para adicioná-la à lista Ações no lado direito do painel.
- Arraste uma ação da lista Caixa de Ferramentas para a lista Ações.
- Clique no botão com sinal de adição (+) e escolha uma ação no menu pop-up.
- Use o atalho do teclado listado ao lado de cada ação no menu pop-up.
- Nos campos de parâmetros, na parte inferior do painel, selecione os parâmetros da ação conforme necessário.

**4** Para atribuir ações adicionais, selecione outro quadro-chave e repita a etapa 3.

**Para testar uma ação de quadro:**

Escolha Controlar > Testar Filme.



## CAPÍTULO 2

### Escrevendo scripts com o ActionScript

.....

Ao criar scripts no ActionScript, você poderá escolher o nível de detalhes desejado. Para realizar ações simples, você pode usar o painel Ações no Modo Normal e criar scripts escolhendo opções em menus e listas. No entanto, se desejar usar o ActionScript para escrever scripts mais elaborados, você precisará entender como o ActionScript funciona como linguagem.

Assim como outras linguagens de script, o ActionScript consiste em componentes, como funções e objetos predefinidos, e permite que você crie seus próprios objetos e funções. O ActionScript segue suas próprias regras de sintaxe, reserva palavras-chave, fornece operadores e permite que você use variáveis para armazenar e recuperar informações.

O estilo e a sintaxe do ActionScript são muito semelhantes aos do JavaScript. O Flash 5 executa conversões no ActionScript escrito em qualquer versão anterior do Flash.

## Usando a sintaxe do ActionScript

O ActionScript possui regras de gramática e pontuação que determinam quais caracteres e palavras são usados para dar significado e em que ordem eles podem ser escritos. Por exemplo, no inglês, um ponto encerra uma frase. No ActionScript, um ponto-e-vírgula encerra um comando.

As regras gerais que se aplicam a todo o ActionScript são apresentadas a seguir. A maioria dos termos do ActionScript também têm seus próprios requisitos individuais; para obter as regras de um termo específico, consulte a entrada correspondente no Capítulo 7, “Dicionário do ActionScript”.

### Sintaxe de ponto

No ActionScript, um ponto (.) é usado para indicar as propriedades ou os métodos relacionados a um objeto ou clipe de filme. O ponto também é usado para identificar o caminho de destino para um clipe de filme ou uma variável. Uma expressão de sintaxe de ponto começa com o nome do objeto ou clipe de filme seguido por um ponto e termina com a propriedade, o método ou a variável que você deseja especificar.

Por exemplo, a propriedade `_x` do clipe de filme indica a posição do eixo *x* do clipe de filme no Palco. A expressão `ballMC._x` se refere à propriedade `_x` da instância `ballMC` do clipe de filme.

Outro exemplo: `submit` é uma variável definida no clipe de filme `form` que é aninhado dentro do clipe de filme `shoppingCart`. A expressão `shoppingCart.form.submit = true` define a variável `submit` da instância `form` como `true`.

A expressão do método de um objeto ou clipe de filme segue o mesmo padrão. Por exemplo, o método `play` da instância `ballMC` move a reprodução na Linha de Tempo de `ballMC`, conforme o seguinte comando:

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

A sintaxe de ponto também usa dois aliases especiais, `_root` e `_parent`. O alias `_root` se refere à Linha de Tempo principal. Você pode usar o alias `_root` para criar um caminho de destino absoluto. Por exemplo, o comando a seguir chama a função `buildGameBoard` no clipe de filme `functions` na Linha de Tempo principal:

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

Você pode usar o alias `_parent` para se referir a um clipe de filme no qual o clipe de filme atual esteja aninhado. Você também pode usar `_parent` para criar um caminho de destino relativo. Por exemplo, se o clipe de filme `dog` estiver aninhado dentro do clipe de filme `animal`, o seguinte comando na instância `dog` informará que `animal` deve parar:

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

Consulte o Capítulo 4, “Trabalhando com clipes de filme”.

## Sintaxe de barra

A sintaxe de barra foi usada no Flash 3 e 4 para indicar o caminho de destino de um clipe de filme ou de uma variável. Essa sintaxe ainda é suportada pelo Flash 5 Player, mas seu uso não é recomendado. Na sintaxe de barra, as barras são usadas no lugar dos pontos para indicar o caminho para um clipe de filme ou uma variável. Para indicar uma variável, coloque dois-pontos antes dela, da seguinte forma:

```
myMovieClip/childMovieClip:myVariable
```

Você pode escrever o mesmo caminho de destino na sintaxe de ponto, da seguinte forma:

```
myMovieClip.childMovieClip.myVariable
```

A sintaxe de barra era mais utilizada com a ação `tellTarget`, cujo uso também não é mais recomendado.

**Observação:** A ação `with` agora é mais usada do que a `tellTarget`, pois é mais compatível com a sintaxe de ponto. Para obter mais informações, consulte as entradas individuais correspondentes no Capítulo 7, “Dicionário do ActionScript”.

## Chaves

Os comandos do ActionScript são agrupados em blocos com chaves (`{ }`), conforme o seguinte script:

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

Consulte “Usando ações”, na página 69.

## Pontos-e-vírgulas

Um comando do ActionScript termina em ponto-e-vírgula, mas o Flash compilará o script com êxito mesmo que você omita esse ponto-e-vírgula. Por exemplo, os seguintes comandos terminam em pontos-e-vírgulas:

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

Os mesmos comandos poderiam ser escritos sem os pontos-e-vírgulas de finalização:

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

## Parênteses

Quando você definir uma função, coloque os argumentos entre parênteses:

```
function myFunction (nome, idade, leitor){  
    ...  
}
```

Quando você chamar uma função, inclua os argumentos passados para a função entre parênteses, da seguinte forma:

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

Você também pode usar parênteses para substituir a ordem de precedência do ActionScript ou para facilitar a leitura dos comandos do ActionScript. Consulte “Precedência de operadores”, na página 63.

Use parênteses também para avaliar uma expressão à esquerda de um ponto em uma sintaxe de ponto. Por exemplo, no comando a seguir, os parênteses fazem com que `new Color(this)` avalie e crie um objeto de nova cor:

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

Se você não tiver usado parênteses, precisará adicionar um comando ao código para avaliá-lo:

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

## Letras maiúsculas e minúsculas

No ActionScript, somente as palavras-chave diferenciam maiúsculas de minúsculas; no resto do ActionScript, você pode usar letras maiúsculas e minúsculas como desejar. Por exemplo, os seguintes comandos são equivalentes:

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

No entanto, é bom ser consistente na escolha das convenções de maiúsculas e minúsculas. Por exemplo, as convenções usadas neste manual facilitam a identificação de nomes de funções e variáveis na leitura de um código do ActionScript.

Se você não usar as maiúsculas e minúsculas da maneira correta nas palavras-chave, o script apresentará erros. Quando a Sintaxe de Cor é ligada no painel Ações, as palavras-chave escritas da maneira correta ficam azuis. Para obter mais informações, consulte “Palavras-chave”, na página 53 e “Realçando e verificando a sintaxe”, na página 44.



## Comentários

No painel Ações, use o comando `comment` para adicionar observações a uma ação de botão ou quadro quando desejar controlar o que pretende que a ação faça. Se você trabalhar em um ambiente colaborativo ou se estiver fornecendo exemplos, os comentários também serão úteis na transmissão de informações para outros desenvolvedores.

Quando você escolhe a ação `comment`, os caracteres `//` são inseridos no script. Será muito mais fácil entender um script, por mais simples que ele seja, se você fizer observações ao criá-lo:

```
on(release) {
    // create new Date object
    myDate = new Date();
    currentMonth = myDate.getMonth();
    // converte o número do mês no nome do mês
    monthName = calcMonth(currentMonth);
    year = myDate.getFullYear();
    currentDate = myDate.getDat ();
}
```

Os comentários são exibidos na cor rosa na janela Script. Eles podem ter qualquer tamanho, pois o tamanho do arquivo exportado não será afetado, e não precisam seguir as regras de palavras-chave ou sintaxe do `JavaScript`.

## Palavras-chave

O `JavaScript` reserva palavras para uso específico na linguagem, portanto você não pode usá-las como nomes de rótulos, variáveis ou funções. A seguinte tabela lista todas as palavras-chave do `JavaScript`:

<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>

Para obter mais informações sobre uma palavra-chave específica, consulte a entrada correspondente no Capítulo 7, “Dicionário do `JavaScript`”.

## Constantes

Uma constante é uma propriedade cujo valor nunca é alterado. As constantes estão listadas na Caixa de Ferramentas Ações e no Capítulo 7, “Dicionário do ActionScript”, com todas as letras maiúsculas.

Por exemplo, as constantes BACKSPACE, ENTER, QUOTE, RETURN, SPACE e TAB são propriedades do objeto Key e se referem às teclas do teclado. Para testar se o usuário está pressionando a tecla Enter, use o seguinte comando:

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

## Sobre tipos de dados

Os tipos de dados descrevem que tipos de informações um elemento do ActionScript ou variável pode conter. Há dois tipos de dados: primitivo e de referência. Os tipos de dados primitivos — string, number e Booleano — têm um valor constante e, portanto, possuem o valor real do elemento que representam. Os tipos de dados de referência — movie clip e object — possuem valores que podem ser alterados e, portanto, contêm referências ao valor real do elemento. Em determinadas situações, as variáveis que contêm tipos de dados primitivos comportam-se de maneira diferente das que contêm tipos de dados de referência. Consulte “Usando variáveis em um script”, na página 60.

Cada tipo de dados possui suas próprias regras e está listado aqui. São incluídas referências de tipos de dados que são discutidos mais detalhadamente.

### String

Uma sequência de caracteres é uma sequência de letras, números e sinais de pontuação. Insira sequências de caracteres em comandos do ActionScript colocando-as entre aspas simples ou aspas duplas. As sequências de caracteres são tratadas como caracteres e não como variáveis. Por exemplo, no seguinte comando, "L7" é uma sequência de caracteres:

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

Você pode usar o operador de adição (+) para *concatenar*, ou unir, duas sequências de caracteres. O ActionScript trata os espaços no começo ou final de uma sequência de caracteres como uma parte literal dessa sequência. A seguinte expressão inclui um espaço depois da vírgula:

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

Embora o ActionScript não faça distinção entre maiúsculas e minúsculas em suas referências a variáveis, nomes de instância e rótulos de quadro, as seqüências de caracteres literais fazem essa distinção. Por exemplo, os dois comandos a seguir colocam textos diferentes nas variáveis de campo de texto especificadas, pois "Hello" e "HELLO" são seqüências de caracteres literais.

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

Para incluir aspas em uma seqüência de caracteres, coloque uma barra invertida (\) antes dessas aspas. Isso é chamado de caractere de "escape". Há outros caracteres que só podem ser representados no ActionScript por seqüências de escape especiais. A seguinte tabela fornece todos os caracteres de escape do ActionScript:

Seqüências de escape	Caractere
\b	Caractere Backspace (ASCII 8)
\f	Caractere de alimentação de formulário (ASCII 12)
\n	Caractere de alimentação de linha (ASCII 10)
\r	Caractere de retorno de carro (ASCII 13)
\t	Caractere Tab (ASCII 9)
\"	Aspas duplas
\'	Aspas simples
\\	Barra invertida
\000 - \377	Um byte especificado em octal
\x00 - \xFF	Um byte especificado em hexadecimal
\u0000 - \uFFFF	Um caractere Unicode de 16 bits especificado em hexadecimal

## Number

O tipo de dados number são números de dupla precisão e ponto flutuante. Você pode manipular números usando os operadores aritméticos de adição (+), subtração (-), multiplicação (\*), divisão (/), módulo (%), incremento (++) e decremento (--). Você também pode usar métodos do objeto Math predefinido para manipular números. O seguinte exemplo usa o método sqrt (raiz quadrada) para retornar a raiz quadrada do número 100:

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

Consulte "Operadores numéricos", na página 64.

## Booleano

Um valor booleano é um valor que é `true` ou `false`. O ActionScript também converte os valores `true` e `false` em 1 e 0 quando apropriado. Os valores booleanos são freqüentemente usados com operadores lógicos em comandos do ActionScript que fazem comparações para controlar o fluxo de um script. Por exemplo, no seguinte script, o filme será reproduzido se a variável `password` for `true`:

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

Consulte “Usando comandos if”, na página 71 e “Operadores lógicos”, na página 65.

## Object

Um object é uma coleção de propriedades. Cada propriedade possui um nome e um valor. O valor de uma propriedade pode ser qualquer tipo de dados do Flash, até mesmo um tipo de dados object. Isso permite “aninhar” os objetos, ou seja, organizá-los uns dentro dos outros. Para especificar os objetos e suas propriedades, use o operador ponto (`.`). Por exemplo, no seguinte código, `hoursWorked` é uma propriedade de `weeklyStats`, que por sua vez é uma propriedade de `employee`:

```
employee.weeklyStats.hoursWorked
```

Você pode usar objetos predefinidos do ActionScript para acessar e manipular tipos específicos de informação. Por exemplo, o objeto `Math` possui métodos que executam operações matemáticas com números que você passa para eles. Este exemplo usa o método `sqrt`:

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

O objeto `MovieClip` do ActionScript possui métodos que permitem que você controle instâncias de símbolos de clipes de filme no Palco. Este exemplo usa os métodos `play` e `nextFrame`:

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

Você também pode criar seus próprios objetos para organizar as informações no filme. Para adicionar interatividade a um filme com o ActionScript, serão necessárias muitas informações diferentes: por exemplo, você precisará saber o nome de um usuário, a velocidade de uma bola, os itens em uma cesta de compras, o número de quadros carregados, o CEP do usuário e qual foi a última tecla pressionada. A criação de objetos personalizados permite que você organize essas informações em grupos, simplifique e reutilize os scripts. Para obter mais informações, consulte “Usando objetos personalizados”, na página 83.

## Movie clip

Clipes de filme são símbolos que podem reproduzir animações em um filme do Flash. Um clipe de filme é o único tipo de dados que se refere a elementos gráficos. O tipo de dados movie clip permite controlar símbolos de clipe de filme com métodos do objeto MovieClip. Chame os métodos usando o operador ponto (.), da seguinte forma:

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

## Sobre variáveis

Uma variável é um repositório que possui informações. O repositório é sempre o mesmo, mas o conteúdo pode mudar. Se alterar o valor de uma variável quando o filme estiver sendo reproduzido, você poderá registrar e salvar informações sobre as atividades do usuário, gravar valores que mudam à medida que o filme é reproduzido ou avaliar se uma condição é verdadeira ou falsa.

É bom atribuir sempre um valor conhecido a uma variável na primeira vez que você defini-la. Isso se chama inicializar uma variável e é realizado mais freqüentemente no primeiro quadro do filme. A inicialização de variáveis facilita o controle e a comparação do valor da variável à medida que o filme é reproduzido.

As variáveis podem conter qualquer tipo de dados: number, string, Booleano, object ou movie clip. Os tipos de dados que uma variável contém afetam a forma como o valor da variável é alterado quando atribuído em um script.

Os tipos mais comuns de informações que você pode armazenar em uma variável incluem uma URL, um nome de usuário, o resultado de uma operação matemática, o número de vezes que um evento ocorreu ou se um botão foi clicado. Cada instância de filme e clipe de filme possui seu próprio conjunto de variáveis, e cada variável possui seu próprio valor independentemente das variáveis em outros filmes ou clipes de filme.

## Nomeando uma variável

O nome de uma variável deve seguir estas regras:

- Ele deve ser um identificador.
- Ele não pode ser uma palavra-chave ou um literal booleano (true ou false).
- Ele deve ser exclusivo em seu escopo. (Consulte “Atribuindo um escopo a uma variável”, na página 59.)

## Atribuindo um tipo a uma variável

No Flash, você não precisa definir explicitamente que uma variável contém um número, uma seqüência de caracteres ou outros tipos de dados. O Flash determina o tipo de dados de uma variável quando o seguinte é atribuído à variável:

```
x = 3;
```

Na expressão `x = 3` o Flash avalia o elemento à direita do operador e determina se ele é do tipo `number`. Uma atribuição posterior pode alterar o tipo de `x`; por exemplo, `x = "hello"` altera o tipo de `x` para uma `string`. Uma variável à qual não tenha sido atribuído um valor tem o tipo `undefined`.

O `ActionScript` converte tipos de dados automaticamente sempre que uma expressão exige. Por exemplo, quando você passa um valor para a ação `trace`, `trace` converte automaticamente o valor em uma seqüência de caracteres e a envia para a janela Saída. Em expressões com operadores, o `ActionScript` converte tipos de dados conforme necessário; por exemplo, quando usado com uma seqüência de caracteres, o operador `+` espera que o outro operando seja uma seqüência de caracteres:

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

O `ActionScript` converte o número `7` na seqüência de caracteres `"7"` e o adiciona ao final da primeira seqüência de caracteres, o que resulta na seguinte seqüência de caracteres:

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

Quando você depura scripts, é útil determinar os tipos de dados de uma expressão ou variável para entender seu comportamento. É possível fazer isso com o operador `typeof`. Por exemplo:

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

Para converter uma seqüência de caracteres em um valor numérico, use a função `Number`. Para converter um valor numérico em uma seqüência de caracteres, use a função `String`. Consulte as entradas individuais correspondentes no Capítulo 7, “Dicionário `ActionScript`”, na página 173.

## Atribuindo um escopo a uma variável

O “escopo” de uma variável refere-se à área na qual a variável é conhecida e na qual pode ser referenciada. As variáveis no ActionScript podem ser globais ou locais. Uma variável global é compartilhada entre todas as Linhas de Tempo; uma variável local só está disponível dentro do seu próprio bloco de código (entre chaves).

Você pode usar o comando `var` para declarar uma variável local dentro de um script. Por exemplo, as variáveis `i` e `j` são usadas com frequência para contagem de loops. No seguinte exemplo, `i` é usado como uma variável local; ela só existe dentro da função `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;
        }
    }
}
```

Variáveis locais também ajudam a impedir colisões de nomes, que podem causar erros no filme. Por exemplo, se usar `name` como uma variável local, você poderá usá-la para armazenar o nome de um usuário em um contexto e o nome de uma instância de clipe de filme em outro; como essas variáveis são executadas em escopos separados, não há colisões.

É bom usar variáveis locais no corpo de uma função para que a função possa atuar como um código independente. Uma variável local só pode ser alterada no seu próprio bloco de código. Se uma expressão em uma função usar uma variável global, algum fator fora da função poderá alterar seu valor, o que irá alterar a função.



## Declaração de variável

Para declarar variáveis globais, use a ação `setVariables` ou o operador de atribuição (`=`). Os dois métodos obtêm os mesmos resultados.

Para declarar variáveis locais, use o comando `var` dentro do corpo de uma função. Os escopos de variáveis locais estão no bloco e expiram no final desse bloco. As variáveis locais não declaradas em um bloco expiram no final do script.

**Observação:** A ação `call` também cria um novo escopo de variável local para o script chamado. Quando o script chamado é encerrado, o escopo dessa variável local desaparece. No entanto, isso não é recomendado, pois a ação `call` foi substituída pela ação `with` que é mais compatível com a sintaxe de ponto.

Para testar o valor de uma variável, use a ação `trace` a fim de enviar o valor para a janela Saída. Por exemplo, `trace(hoursWorked)` envia o valor da variável `hoursWorked` para a janela Saída no modo de testar filme. Você também pode verificar e definir os valores da variável no Depurador no modo de testar filme. Para obter mais informações, consulte o Capítulo 6, “Solucionando problemas do ActionScript”.

## Usando variáveis em um script

Você deve declarar uma variável em um script antes de usá-la em uma expressão. Se você usar uma variável não-declarada, como no seguinte exemplo, o valor da variável será `undefined` e o script irá gerar um erro:

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

O comando que declara a variável `myWebSite` deverá vir primeiro para que a variável na ação `getURL` possa ser substituída por um valor.

Você pode alterar o valor de uma variável muitas vezes em um script. O tipo de dados que a variável contém afeta como e quando a variável será alterada. Tipos de dados primitivos, como seqüências de caracteres e números, são passados por valor. Isso significa que o conteúdo real da variável é passado para a variável.

No seguinte exemplo, `x` é definido como 15 e esse valor é copiado em `y`. Quando `x` é alterado para 30, o valor de `y` permanece 15 porque `y` não verifica o valor de `x`; ele contém o valor de `x` que foi passado.

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





Outro exemplo: a variável `in` contém um valor primitivo 9; portanto, o valor real é passado para a função `sqrt` e o valor retornado é 3:

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

```
var in = 9;  
var out = sqrt(in);
```

O valor da variável `in` não é alterado.

O tipo de dados de objeto pode conter informações tão numerosas e complexas que uma variável com esse tipo não possuirá o valor real, mas uma referência a esse valor. Essa referência é como um alias que aponta para o conteúdo da variável. Quando a variável precisa saber seu valor, a referência pede o conteúdo e responde sem transferir o valor para a variável.

O seguinte exemplo mostra como passar por referência:

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

O código acima cria um objeto Array chamado `myArray` que tem dois elementos. A variável `newArray` é criada e passa uma referência para `myArray`. Quando o segundo elemento de `myArray` é alterado, todas as variáveis com uma referência a ele são afetadas. A ação `trace` enviará ["tom", "jack"] para a janela Saída.

No próximo exemplo, `myArray` contém um objeto Array; portanto, ela é passada para a função `zeroArray` por referência. A função `zeroArray` altera o conteúdo da matriz em `myArray`.

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

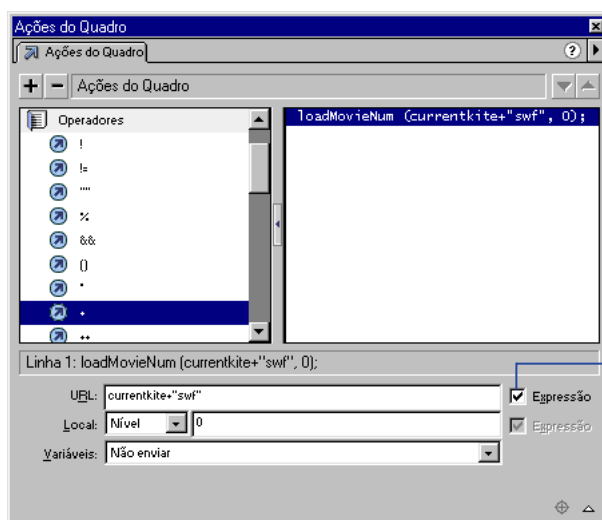
```
var myArray = new Array();  
myArray[0] = 1;  
myArray[1] = 2;  
myArray[2] = 3;  
  
var out = zeroArray(myArray)
```

A função `zeroArray` aceita um objeto Array como um argumento e define todos os elementos dessa matriz como 0. Ela pode modificar a matriz, pois a matriz é passada por referência.

As referências a todos os objetos que não sejam clipes de filme são chamadas *referências fixas*, pois os objetos com referências não podem ser excluídos. A referência a um clipe de filme é um tipo especial de referência chamado *referência flexível*. As referências flexíveis não forçam a existência do objeto para o qual a referência foi feita. Se um clipe de filme for destruído com uma ação como `removeMovieClip`, as referências a ele não funcionarão mais.

## Usando operadores para manipular valores em expressões

Uma expressão é qualquer comando que o Flash possa avaliar e que retorne um valor. Você pode criar uma expressão combinando operadores e valores, ou chamando uma função. Ao escrever uma expressão no painel Ações no Modo Normal, certifique-se de que a caixa Expressão esteja marcada no painel Parâmetros, caso contrário o campo conterá o valor literal de uma sequência de caracteres.



Caixa Expressão

Os operadores são caracteres que especificam como combinar, comparar ou modificar os valores de uma expressão. Os elementos aos quais o operador é aplicado são chamados *operandos*. Por exemplo, no comando a seguir, o operador + adiciona o valor de um literal numérico ao valor da variável `foo`; `foo + 3` são os operandos:

```
foo + 3
```

Esta seção descreve regras gerais sobre tipos comuns de operadores. Para obter informações detalhadas sobre cada operador mencionado aqui, bem como sobre operadores especiais que não se encaixam nessas categorias, consulte o Capítulo 7, “Dicionário do ActionScript”.

## Precedência de operadores

Quando dois ou mais operadores são usados no mesmo comando, alguns operadores têm precedência sobre outros. O ActionScript segue uma hierarquia precisa para determinar que operadores devem ser executados primeiro. Por exemplo, a multiplicação é sempre executada antes da adição; no entanto, itens entre parênteses têm precedência sobre a multiplicação. Portanto, como não há parênteses, o ActionScript executa a multiplicação primeiro no seguinte exemplo:

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

O resultado é 14.

Porém, quando a operação de adição está entre parênteses, o ActionScript executa a adição primeiro:

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

O resultado é 18.

Para obter uma tabela de todos os operadores e suas precedências, consulte o Apêndice B, “Associatividade e precedência de operadores”.

## Associatividade de operadores

Quando dois ou mais operadores compartilham a mesma precedência, sua associatividade determina a ordem na qual serão executados. A associatividade pode ser da esquerda para a direita ou da direita para a esquerda. Por exemplo, o operador de multiplicação possui uma associatividade da esquerda para a direita; portanto, os dois comandos a seguir são equivalentes:

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

Para obter uma tabela de todos os operadores e suas associatividades, consulte o Apêndice B, “Associatividade e precedência de operadores”.



## Operadores numéricos

Os operadores numéricos adicionam, subtraem, multiplicam, dividem e executam outras operações aritméticas. Parênteses e o sinal de menos são operadores aritméticos. A tabela a seguir lista os operadores numéricos do ActionScript:

Operador	Operação executada
+	Adição
*	Multiplicação
/	Divisão
%	Módulo
-	Subtração
++	Incremento
--	Decremento

## Operadores de comparação

Os operadores de comparação comparam os valores de expressões e retornam um valor booleano (true ou false). Esses operadores são mais usados em loops e em comandos condicionais. No seguinte exemplo, se a variável score for 100, um determinado filme será carregado; caso contrário, um filme diferente será carregado:

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

A tabela a seguir lista os operadores de comparação do ActionScript:

Operador	Operação executada
<	Menor que
>	Maior que
<=	Menor ou igual a
>=	Maior ou igual a

## Operadores de sequência de caracteres

O operador `+` tem um efeito especial quando opera em seqüências de caracteres: ele concatena os dois operandos da seqüência de caracteres. Por exemplo, o comando a seguir adiciona:

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

O resultado é "Congratulations, Donna!". Se apenas um dos operandos do operador `+` for uma seqüência de caracteres, o Flash converterá o outro operando em uma seqüência de caracteres.

Os operadores de comparação, `>`, `>=`, `<` e `<=` também têm um efeito especial quando operam em seqüências de caracteres. Esses operadores comparam duas seqüências de caracteres para determinar o que vem primeiro em ordem alfabética. Os operadores de comparação só irão comparar seqüências de caracteres se os dois operandos forem seqüências de caracteres. Se apenas um dos operandos for uma seqüência de caracteres, o ActionScript converterá ambos os operandos em números e executará uma comparação numérica.

**Observação:** A atribuição de tipos de dados do ActionScript no Flash 5 permite que os mesmos operadores sejam usados em diferentes tipos de dados. Não é mais necessário usar os operadores de seqüência de caracteres do Flash 4 (por exemplo, `eq`, `ge` e `lt`); a não ser que você esteja exportando como um filme do Flash 4.

## Operadores lógicos

Os operadores lógicos comparam valores booleanos (`true` e `false`) e retornam um terceiro valor booleano. Por exemplo, se os dois operandos forem avaliados como `true`, o operador AND lógico (`&&`) retornará `true`. Se os dois operandos forem avaliados como `true`, o operador OR lógico (`||`) retornará `false`. Os operadores lógicos são freqüentemente usados em conjunto com os operadores de comparação para determinar a condição de uma ação `if`. Por exemplo, no seguinte script, se ambas as expressões forem verdadeiras, a ação `if` executará:

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

A tabela a seguir lista os operadores lógicos do ActionScript:

Operador	Operação executada
<code>&amp;&amp;</code>	AND lógico
<code>  </code>	OR lógico
<code>!</code>	NOT lógico

## Operadores bit a bit

Os operadores bit a bit manipulam internamente números de ponto flutuante para transformá-los em inteiros de 32 bits, com os quais é mais fácil trabalhar. A operação bit a bit exata executada depende do operador, mas todas as operações bit a bit avaliam cada dígito de um número de ponto flutuante separadamente para calcular um novo valor.

A tabela a seguir lista os operadores bit a bit do ActionScript:

Operador	Operação executada
&	And bit a bit
	Or bit a bit
^	Xor bit a bit
-	Not bit a bit
<<	Desloca para a esquerda
>>	Desloca para a direita
>>>	Desloca para a direita sem preenchimento

## Operadores de igualdade e de atribuição

Você pode usar o operador de igualdade (==) para determinar se os valores ou as identidades de dois operandos são iguais. Essa comparação retorna um valor booleano (true ou false). Se os operandos forem seqüências de caracteres, números ou valores booleanos, eles serão comparados por valor. Se os operandos forem objetos ou matrizes, serão comparados por referência.

Você pode usar o operador de atribuição (=) para atribuir um valor a uma variável, da seguinte forma:

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

Você também pode usar o operador de atribuição para atribuir diversas variáveis na mesma expressão. No seguinte comando, o valor de b é atribuído às variáveis c e d:

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



Você também pode usar operadores de atribuição composta para combinar operações. Operadores compostos são executados em ambos os operandos e atribuem o novo valor ao primeiro operando. Por exemplo, os dois comandos a seguir são equivalentes:

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

A tabela a seguir lista os operadores de igualdade e de atribuição do ActionScript:

Operador	Operação executada
==	Igualdade
!=	Diferença
=	Atribuição
+=	Adição e atribuição
-=	Subtração e atribuição
*=	Multiplicação e atribuição
%=	Módulo e atribuição
/=	Divisão e atribuição
<=	Deslocamento para a esquerda bit a bit e atribuição
>=	Deslocamento para a direita bit a bit e atribuição
>>=	Deslocamento para a direita sem preenchimento e atribuição
^=	Xor bit a bit e atribuição
=	Or bit a bit e atribuição
&=	And bit a bit e atribuição

## Operadores ponto e de acesso a matriz

Você pode usar o operador ponto (.) e o operador de acesso a matriz ([ ]) para acessar qualquer propriedade de objeto predefinida ou personalizada do ActionScript, incluindo as propriedades de um clipe de filme.

O operador ponto usa o nome de um objeto à sua esquerda e o nome de uma propriedade ou variável à sua direita. O nome da propriedade ou variável não pode ser uma sequência de caracteres ou uma variável avaliada como uma sequência de caracteres; ele deve ser um identificador. Os seguintes exemplos usam o operador ponto:

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

O operador ponto e o operador de acesso a matriz executam a mesma função, mas o operador ponto usa um identificador como sua propriedade e o operador de acesso a matriz avalia seu conteúdo como um nome e, em seguida, acessa o valor da propriedade nomeada. Por exemplo, as duas linhas de código a seguir acessam a mesma variável `velocity` no clipe de filme `rocket`:

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

Você pode usar o operador de acesso a matriz para definir e recuperar variáveis e nomes de instância dinamicamente. Por exemplo, no seguinte código, a expressão dentro do operador [ ] é avaliada e o resultado da avaliação é usado como o nome da variável a ser recuperada do `name` do clipe de filme:

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

Caso você já conheça a sintaxe de barra do ActionScript do Flash 4, talvez você tenha feito o mesmo usando a função `eval` da seguinte forma:

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

O operador de acesso a matriz também pode ser usado à esquerda de um comando de atribuição. Isso permite que você defina nomes de objeto, variável e instância dinamicamente, conforme o seguinte exemplo:

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

Isso equivale à seguinte sintaxe de barra do ActionScript do Flash 4:

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

O operador de acesso a matriz também pode ser aninhado em si mesmo para simular matrizes multidimensionais.

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

Isso equivale à seguinte sintaxe de barra:

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

**Observação:** Se desejar escrever o ActionScript compatível com o Flash 4 Player, você poderá usar a ação `eval` com o operador `add`.



## Usando ações

Ações são comandos do ActionScript. Várias ações atribuídas ao mesmo quadro ou objeto criam um script. As ações podem atuar independentemente umas das outras, conforme os seguintes comandos:

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

Você também pode aninhar ações usando uma ação dentro de outra; isso permite que uma ação afete a outra. No seguinte exemplo, a ação `if` informa à ação `gotoAndPlay` quando executar:

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

As ações podem mover a reprodução na Linha de Tempo (`gotoAndPlay`), controlar o fluxo de um script criando loops (`do while`) ou lógicas condicionais (`if`), ou criar novas funções e variáveis (`function`, `setVariable`). A seguinte tabela lista todas as ações do ActionScript:

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

Para obter exemplos de uso e sintaxe de cada ação, consulte as entradas individuais no Capítulo 7, “Dicionário do ActionScript”.

**Observação:** Neste manual, o termo *ação* do ActionScript é sinônimo do termo *comando* do JavaScript.

## Escrevendo um caminho de destino

Para usar uma ação para controlar um clipe de filme ou um filme carregado, você deve especificar seu nome e endereço, o que é chamado *caminho de destino*. As seguintes ações usam um ou mais caminhos de destino como argumentos:

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

Por exemplo, a ação loadMovie usa os argumentos *URL*, *Location* e *Variables*. *URL* é o local, na Web, onde se encontra o filme que você deseja carregar. *Location* é o caminho de destino no qual o filme será carregado.

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

**Observação:** O argumento *Variables* não é necessário nesse exemplo.

O seguinte comando carrega a URL `http://www.mySite.com/myMovie.swf` na instância `bar` na Linha de Tempo principal, `_root`; `_root.bar` é o caminho de destino;

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

No ActionScript, um clipe de filme é identificado por seu nome de instância. Por exemplo, no seguinte comando, a propriedade `_alpha` do clipe de filme chamado `star` é definida com visibilidade de 50%:

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

**Para dar a um clipe de filme um nome de instância:**

- 1 Selecione o clipe de filme no Palco.
- 2 Escolha Janela > Painéis > Instância.
- 3 Insira um nome de instância no campo Nome.

**Para identificar um filme carregado:**

Use `_levelX` onde `X` é o número do nível especificado na ação `loadMovie` que carregou o filme.

Por exemplo, um filme carregado para o nível 5 possui o nome de instância `_level5`. No seguinte exemplo, um filme é carregado para o nível 5 e sua visibilidade é definida como falsa:

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

**Para inserir o caminho de destino de um filme:**

Clique no botão Caminho de Destino no painel Ações e selecione um clipe de filme na lista exibida.

Para obter mais informações sobre como escrever caminhos de destino, consulte o Capítulo 4, “Trabalhando com clipes de filme”.

## Controlando o fluxo em scripts

O ActionScript usa as ações `if`, `for`, `while`, `do...while` e `for...in` para executar uma ação, dependendo da existência de uma condição.

### Usando comandos if

Os comandos que verificam se uma condição é verdadeira ou falsa começam com o termo `if`. Se a condição existir, o ActionScript executará o comando seguinte. Se a condição não existir, o ActionScript passará para o próximo comando fora do bloco de código.

Para otimizar o desempenho do código, verifique as condições mais prováveis primeiro.

Os seguintes comandos testam várias condições. O termo `else if` especifica testes alternativos que poderão ser executados se as condições anteriores forem falsas.

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

## Repetindo uma ação

O ActionScript pode repetir uma ação por um número especificado de vezes ou enquanto uma condição específica existir. Use as ações `while`, `do...while`, `for`, e `for...in` para criar loops.

### Para repetir uma ação enquanto uma condição existir:

Use o comando `while`.

Um loop `while` avaliará uma expressão e executará o código em seu corpo se a expressão for `true`. Depois que cada comando do corpo for executado, a expressão será avaliada novamente. No seguinte exemplo, o loop é executado quatro vezes:

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

Você pode usar o comando `do...while` para criar um loop do mesmo tipo que o loop `while`. Em um loop `do...while` a expressão é avaliada na parte inferior do bloco de código para que o loop seja sempre executado pelo menos uma vez, da seguinte forma:

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

### Para repetir uma ação usando uma contagem interna:

Use o comando `for`.

A maioria dos loops usa uma contagem para controlar quantas vezes o loop é executado. Você pode declarar uma variável e escrever um comando que aumente ou diminua a variável sempre que o loop for executado. Na ação `for`, a contagem e o comando que incrementa a contagem fazem parte da ação, da seguinte forma:

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

Para criar um loop nos filhos de um clipe de filme ou objeto:

Use o comando `for...in`.

Os filhos incluem outros clipes de filme, funções, objetos e variáveis. O seguinte exemplo usa `trace` para imprimir seus resultados na janela Saída:

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

Esse exemplo produz os seguintes resultados na janela Saída:

```
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
```

O script pode iterar em um determinado tipo de filho — por exemplo, apenas em filhos de clipes de filme. Você pode fazer isso usando `for...in` juntamente com o operador `typeof`.

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

**Observação:** O comando `for...in` itera em propriedades de objetos na cadeia protótipo do objeto iterado. Se o protótipo de um objeto filho for `parent`, `for...in` também irá iterar nas propriedades de `parent`. Consulte “Criando herança”, na página 84.

Para obter mais informações sobre cada ação, consulte as entradas individuais no Capítulo 7, “Dicionário do ActionScript”.

## Usando funções predefinidas

Uma função é um bloco de códigos do ActionScript que pode ser reutilizado em qualquer lugar de um filme. Se você passar valores específicos chamados argumentos para uma função, a função operará sobre esses valores. Uma função também pode retornar valores. O Flash possui funções predefinidas que permitem que você acesse certas informações e execute certas tarefas, como a detecção de uma colisão (`hitTest`), obtendo o valor da última tecla pressionada (`keyCode`) e o número da versão do Flash Player que hospeda o filme (`getVersion`).

### Chamando uma função

Você pode chamar uma função em qualquer Linha de Tempo, incluindo um filme carregado. Cada função tem suas próprias características e algumas exigem que você passe determinados valores. Se você passar mais argumentos do que o exigido pela função, os valores extras serão ignorados. Se você não passar um argumento exigido, o tipo de dados `undefined` será atribuído aos argumentos vazios, o que poderá causar erros quando você exportar um script. Para chamar uma função, a mesma deve estar em um quadro que já tenha sido reproduzido.

As funções predefinidas do Flash estão listadas na tabela a seguir:

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

**Observação:** As funções de sequência de caracteres estão obsoletas e não foram listadas na tabela acima.

#### Para chamar uma função no Modo Especialista:

Use o nome da função. Passe os argumentos necessários entre parênteses.

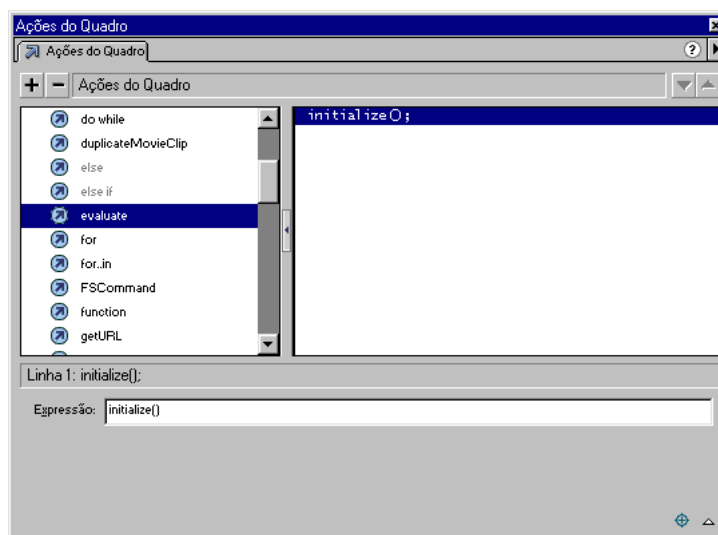
O seguinte exemplo chama a função `initialize` que não exige argumentos:

```
initialize();
```



**Para chamar uma função no Modo Normal:**

Use a ação `evaluate`. Insira o nome da função e os argumentos necessários no campo `Expressão`.



*Use a ação `evaluate` para chamar uma função no Modo Normal.*

Para chamar uma função em outra Linha de Tempo use um caminho de destino. Por exemplo, para chamar uma função `calculateTax` que tenha sido declarada na instância `functionsMovieClip`, use o seguinte caminho:

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

**Observação:** Passe os argumentos entre parênteses.

Para obter mais informações sobre cada função, incluindo funções de seqüências de caracteres obsoletas, consulte as entradas individuais no Capítulo 7, “Dicionário do ActionScript”.

## Criando funções personalizadas

Você pode definir funções para executar uma série de comandos sobre valores passados. Suas funções também podem retornar valores. Depois que uma função for definida, ela poderá ser chamada a partir de qualquer Linha de Tempo, incluindo a Linha de Tempo de um filme carregado.

Uma função pode ser considerada uma “caixa preta”: quando uma função é chamada, ela recebe dados (argumentos). Ela executa algumas operações e, em seguida, gera resultados (um valor de retorno). Uma função bem escrita possui comentários sobre os dados de entrada, os resultados e o objetivo. Dessa forma, o usuário da função não precisa entender seu funcionamento exato.

### Definindo uma função

Funções, assim como variáveis, são anexadas ao clipe de filme que as define. Quando uma função é redefinida, a nova definição substitui a antiga.

Para definir uma função, use a ação `function` seguida do nome da função, dos argumentos a serem passados para a função e dos comandos do ActionScript que indicam o que a função faz.

A seguinte função é chamada `Circle` com o argumento `radius`:

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

**Observação:** A palavra-chave `this` usada em um corpo da função é uma referência ao clipe de filme ao qual a função pertence.

Você também pode definir uma função criando um *literal de função*. Um literal de função é uma função sem nome declarada em uma expressão em vez de em um comando. Você pode usar um literal de função para definir uma função, retornar seu valor e atribuí-la a uma variável em uma expressão da seguinte forma:

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



## Passando argumentos para uma função

Os argumentos são os elementos nos quais uma função executa seu código. (Neste manual, os termos *argumento* e *parâmetro* são intercambiáveis.) Por exemplo, a seguinte função possui os argumentos `initials` e `finalScore`:

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

Quando a função é chamada, os argumentos necessários devem ser passados a ela. A função substitui os valores passados para os argumentos na definição da função. Neste exemplo, `scorecard` é o nome da instância de um clipe de filme; `display` e `score` são campos de texto editáveis na instância. A seguinte chamada de função atribui o valor "JEB" à variável `display` e o valor 45000 à variável `score`:

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

O argumento `initials` na função `fillOutScorecard` é semelhante a uma variável local; ele existe enquanto a função é chamada e desaparece quando a função é encerrada. Se você omitir argumentos durante uma chamada de função, os argumentos omitidos serão passados como `undefined`. Se você fornecer argumentos extra em uma chamada de função que não tenham sido solicitados pela declaração da função, eles serão ignorados.

## Usando variáveis locais em uma função

Variáveis locais são ferramentas valiosas para organizar códigos e facilitar sua compreensão. Quando uma função usa variáveis locais, ela pode ocultar suas variáveis de todos os outros scripts no filme; os escopos das variáveis locais encontram-se no corpo da função e são destruídos quando a função é encerrada. Os argumentos passados a uma função também são tratados como variáveis locais.

**Observação:** Se você modificar variáveis globais em uma função, use comentários de script para documentar essas modificações.

## Retornando valores de uma função

Você pode usar a ação `return` para retornar valores de funções. A ação `return` interrompe a função e a substitui pelo valor da ação `return`. Se o Flash não encontrar uma ação `return` antes do término de uma função, uma sequência de caracteres vazia será retornada. Por exemplo, a seguinte função retorna o quadrado do argumento `x`:

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

Algumas funções executam uma série de tarefas sem retornar um valor. Por exemplo, a seguinte função inicializa uma série de variáveis globais:

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

## Chamando uma função

Para invocar uma função usando o painel Ações no Modo Normal, use a ação `evaluate`. Passe os argumentos necessários entre parênteses. Você pode chamar uma função em qualquer Linha de Tempo, incluindo um filme carregado. Por exemplo, o comando a seguir invoca a função `sqr` no clipe de filme `MathLib` na Linha de Tempo principal, passa para ela o argumento `3` e armazena o resultado na variável `temp`:

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

No Flash 4, para simular a chamada de uma função, você pode escrever um script em um quadro após o término do filme e invocá-lo passando o nome do rótulo do quadro para a ação `call`. Por exemplo, se um script que inicializou variáveis estivesse em um quadro rotulado como `initialize`, você o chamaria da seguinte forma:

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

Esse tipo de script não é uma função verdadeira, pois não pode aceitar argumentos e não pode retornar um valor. Embora a ação `call` ainda funcione no Flash 5, seu uso não é recomendado.

## Usando objetos predefinidos

Você pode usar os objetos predefinidos do Flash para acessar certos tipos de informação. A maioria dos objetos predefinidos possui *métodos* (funções atribuídas a um objeto) que você pode chamar para retornar um valor ou executar uma ação. Por exemplo, o objeto Date retorna as informações do relógio do sistema e o objeto Sound permite que você controle elementos de som no filme.

Alguns objetos predefinidos possuem propriedades cujos valores você pode ler. Por exemplo, o objeto Key possui valores constantes que representam teclas no teclado. Cada objeto possui suas próprias características e recursos que podem ser usados no filme.

Os seguintes objetos são objetos predefinidos do Flash:

- Array
- Booleano
- Color
- Date
- Key
- Math
- MovieClip
- Number
- Object
- Selection
- Sound
- String
- XML
- XMLSocket

Instâncias de clipes de filme são representadas como objetos no ActionScript. Você pode chamar métodos de clipes de filme predefinidos da mesma forma que chamaria os métodos de qualquer outro objeto do ActionScript.

Para obter informações detalhadas sobre cada objeto, consulte as entradas correspondentes no Capítulo 7, “Dicionário do ActionScript”.

## Criando um objeto

Existem duas maneiras de criar um objeto: o operador `new` e o operador de inicialização do objeto (`{}`). Você pode usar o operador `new` para criar um objeto a partir de uma classe de objeto predefinida ou a partir de uma classe de objeto personalizada. Você pode usar o operador de inicialização do objeto (`{}`) para criar um objeto do tipo genérico `Object`.

Para usar o operador `new` a fim de criar um objeto, você precisa usá-lo com uma função construtora. (Uma função construtora é simplesmente uma função cujo único objetivo é criar um determinado tipo de objeto.) Os objetos predefinidos do `ActionScript` são essencialmente funções construtoras pré-escritas. O novo objeto *exemplifica* ou cria uma cópia do objeto e atribui todas as propriedades e métodos desse objeto. Isso é semelhante a arrastar um clipe de filme da Biblioteca para o Palco em um filme. Por exemplo, os seguintes comandos exemplificam um objeto `Date`:

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

Você pode acessar os métodos de alguns objetos predefinidos sem exemplificá-los. Por exemplo, o seguinte comando chama o método `random` do objeto `Math`:

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

Cada objeto que exige uma função construtora possui um elemento correspondente na caixa de ferramentas do painel `Ações`; por exemplo, `new Color`, `new Date`, `new String`, e assim por diante.

### Para criar um objeto como o operador `new` no Modo Normal:

- 1 Escolha `setVariable`
- 2 Insira um nome de variável no campo `Nome`.
- 3 Insira `new Object`, `new Color`, e assim por diante, no campo `Valor`. Insira os argumentos solicitados pela função construtora entre parênteses.
- 4 Marque a caixa `Expressão` do campo `Valor`.  
Se você não marcar a caixa `Expressão`, o valor inteiro será um literal de sequência de caracteres.

No seguinte código, o objeto `c` é criado a partir da construtora `Color`:

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

**Observação:** Um nome de objeto é uma variável à qual o tipo de dados do objeto é atribuído.

#### Para acessar um método no Modo Normal:

- 1 Selecione a ação `evaluate`.
- 2 Insira o nome do objeto no campo Expressão.
- 3 Insira uma propriedade do objeto no campo Expressão.

#### Para usar o operador de inicialização do objeto (`{}`) no Modo Normal:

- 1 Selecione a ação `setVariable`.
- 2 Insira o nome no campo Variável; esse é o nome do novo objeto.
- 3 Insira os pares de valor e nome da propriedade separados por ponto-e-vírgula dentro do operador de inicialização do objeto (`{}`).

Por exemplo, neste comando os nomes de propriedade são `radius` e `area` e seus valores são 5 e o valor de uma expressão:

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

Os parênteses fazem com que a expressão seja avaliada. O valor retornado é o valor da variável `area`.

Você também pode aninhar inicializadores de objeto e matriz, conforme este comando:

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

Para obter informações detalhadas sobre cada objeto, consulte as entradas correspondentes no Capítulo 7, “Dicionário do ActionScript”.

## Acessando propriedades de objetos

Use o operador ponto (`.`) para acessar o valor de propriedades em um objeto. O nome do objeto fica à esquerda do ponto e o nome da propriedade, à direita. Por exemplo, no seguinte comando, `myObject` é o objeto e `name` é a propriedade:

```
myObject.name
```

Para atribuir um valor a uma propriedade no Modo Normal, use a ação `setVariable`:

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

Para alterar o valor de uma propriedade, atribua um novo valor conforme mostrado aqui:

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

Você também pode usar o operador de acesso da matriz (`[]`) para acessar as propriedades de um objeto. Consulte “Operadores ponto e de acesso a matriz”, na página 68.

## Chamando métodos de objetos

Você pode chamar o método de um objeto usando o operador ponto seguido do método. O seguinte exemplo chama o método `setVolume` do objeto `Sound`:

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

Para chamar o método de um objeto predefinido no Modo Normal, use a ação `evaluate`.

## Usando o objeto `MovieClip`

Você pode usar os métodos do objeto `MovieClip` predefinido para controlar instâncias de símbolos de clipes de filme no Palco. O seguinte exemplo informa que a instância `dateCounter` deve ser reproduzida:

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

Para obter informações detalhadas sobre o objeto `MovieClip`, consulte a entrada correspondente no Capítulo 7, “Dicionário do `ActionScript`”.

## Usando o objeto `Array`

O objeto `Array` é um objeto predefinido do `ActionScript` comumente usado que armazena seus dados em propriedades numeradas e não em propriedades nomeadas. O nome do elemento de uma matriz é chamado de *índice*. Isso é útil para o armazenamento e a recuperação de certos tipos de informação, como listas de estudantes ou uma sequência de movimentos em um jogo.

Você pode atribuir elementos do objeto `Array` da mesma forma que atribuiria a propriedade de qualquer objeto:

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

Para acessar o segundo elemento da matriz, use a expressão `move[2]`.

O objeto `Array` tem uma propriedade `length` predefinida que é o valor do número de elementos na matriz. Quando um elemento do objeto `Array` é atribuído, e o índice do elemento é um inteiro positivo de modo que `index >= length`, `length` é atualizado automaticamente para `index + 1`.

## Usando objetos personalizados

Você pode criar objetos personalizados para organizar as informações nos scripts a fim de facilitar o armazenamento e o acesso definindo os métodos e as propriedades do objeto. Depois de criar uma “classe” ou um objeto principal, você poderá usar ou “criar” cópias (isto é, instâncias) desse objeto em um filme. Isso permite que você reutilize códigos e conserve o tamanho do arquivo.

Um objeto é um tipo de dados complexo que contém ou não propriedades. Cada propriedade, como uma variável, possui um nome e um valor. As propriedades são anexadas ao objeto e contêm valores que podem ser alterados e recuperados. Esses valores podem ser de qualquer tipo de dados: sequência de caracteres, número, booleano, objeto, clipe de filme ou `undefined`. As seguintes propriedades são de vários tipos de dados:

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

A propriedade de um objeto também pode ser um objeto. Na linha 4 do exemplo anterior, `account` é uma propriedade do objeto `customer` e `currentRecord` é uma propriedade do objeto `account`. O tipo de dados da propriedade `currentRecord` é um número.

## Criando um objeto

Você pode usar o operador `new` para criar um objeto a partir de uma função construtora. Uma função construtora recebe sempre o mesmo nome do tipo de objeto que está criando. Por exemplo, uma construtora que cria um objeto `account` será chamada `Account`. O seguinte comando cria um novo objeto a partir da função chamada `MyConstructorFunction`:

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

Quando `MyConstructorFunction` é chamada, o Flash passa o argumento oculto `this`, que é uma referência ao objeto que `MyConstructorFunction` está criando. Quando você define uma construtora, `this` permite que você se refira aos objetos que a construtora irá criar. Por exemplo, a seguinte função construtora cria um círculo:

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

Funções construtoras são comumente usadas para preencher métodos de um objeto.

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

Para usar um objeto em um script, você deve atribuí-lo a uma variável. Para criar um novo objeto circle com o radius 5, use o operador new para criar o objeto e atribuí-lo à variável local myCircle:

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

**Observação:** Os objetos possuem o mesmo escopo que a variável à qual são atribuídos. Consulte “Atribuindo um escopo a uma variável”, na página 59.

## Criando herança

Todas as funções têm uma propriedade prototype que é criada automaticamente quando a função é definida. Quando você usa uma função construtora para criar um novo objeto, todas as propriedades e métodos da propriedade prototype da construtora tornam-se propriedades e métodos da propriedade \_\_proto\_\_ do novo objeto. A propriedade prototype indica os valores de propriedade padrão para objetos criados com essa função. A transferência de valores através das propriedades \_\_proto\_\_ e prototype se chama herança.

A herança prossegue de acordo com uma hierarquia limitada. Quando você chama o método ou a propriedade de um objeto, o ActionScript verifica se esse elemento existe no objeto. Se ele não existir, o ActionScript procurará a informação (object.\_\_proto\_\_) na propriedade \_\_proto\_\_ do objeto. Se a propriedade chamada não for uma propriedade do objeto \_\_proto\_\_, o ActionScript examinará object.\_\_proto\_\_.\_\_proto\_\_.

É comum anexar métodos a um objeto atribuindo esses métodos à propriedade prototype do objeto. As seguintes etapas descrevem como definir um método de exemplo:

- 1 Defina a função construtora Circle da seguinte forma:

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



- 2 Defina o método `area` do objeto `Circle`. O método `area` calculará a área do círculo. Você pode usar um literal de função para definir o método `area` e a propriedade de área do objeto de protótipo do círculo da seguinte forma:

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

- 3 Crie uma instância do objeto `Circle` da seguinte forma:

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

- 4 Chame o método `area` do novo objeto `myCircle` da seguinte forma:

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

O `ActionScript` procura o método `area` no objeto `myCircle`. Como o objeto não possui um método `area`, o método `area` é procurado no objeto de protótipo `Circle.prototype`. O `ActionScript` o localiza e o chama.

Você também pode anexar um método a um objeto anexando o método a cada instância individual do objeto, conforme este exemplo:

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

Essa técnica não é recomendada. O uso do objeto `prototype` é mais eficiente, pois apenas uma definição de `area` é necessária e essa definição é copiada automaticamente em todas as instâncias criadas pela função `Circle`.

A propriedade `prototype` é suportada pelo `Flash Player` versão 5 e posterior. Para obter mais informações, consulte o Capítulo 7, “Dicionário do `ActionScript`”.

## Abrindo arquivos do Flash 4

O ActionScript mudou consideravelmente com essa versão do Flash 5. Agora, ele é uma linguagem orientada a objetos com vários tipos de dados e sintaxe de ponto. O ActionScript do Flash 4 só tinha um tipo de dados verdadeiro: string. Ele usava diferentes tipos de operadores em expressões para indicar se o valor devia ser tratado como uma sequência de caracteres ou como um número. No Flash 5, você pode usar um conjunto de operadores em todos os tipos de dados.

Quando você usa o Flash 5 para abrir um arquivo que foi criado no Flash 4, o Flash converte automaticamente as expressões do ActionScript para torná-las compatíveis com a nova sintaxe do Flash 5. Você verá as seguintes conversões de operadores e tipos de dados no código do ActionScript:

- O operador = no Flash 4 era usado para igualdade numérica. No Flash 5, == é o operador de igualdade e = é o operador de atribuição. Quaisquer = operadores nos arquivos do Flash 4 são convertidos automaticamente em ==.
- O Flash executa as conversões de tipo automaticamente para garantir que os operadores se comportarão como o esperado. Devido à introdução de vários tipos de dados, os seguintes operadores têm novos significados:

+, ==, !=, <>, <, >, >=, <=

- No ActionScript do Flash 4, esses operadores eram sempre operadores numéricos. No Flash 5, eles se comportam de maneira diferente dependendo dos tipos de dados dos operandos. Para evitar diferenças semânticas em arquivos importados, a função Number é inserida em volta de todos os operandos desses operadores. (Números constantes continuam sendo claramente números, portanto, eles não são incluídos em Number).
- No Flash 4, a sequência de escape \n gerava um caractere de retorno de carro (ASCII 13). No Flash 5, para cumprir com o padrão ECMA-262, \n gera um caractere de alimentação de linha (ASCII 10). Uma sequência \n nos arquivos FLA do Flash 4 é convertida automaticamente em \r.



- O operador & no Flash 4 era usado para adição de seqüências de caracteres. No Flash 5, & é o operador AND bit a bit. O operador de adição de seqüências de caracteres agora é chamado de add. Os operadores & nos arquivos do Flash 4 são convertidos automaticamente em operadores add.
- Muitas funções no Flash 4 não exigiam parênteses, por exemplo, Get Timer, Set Variable, Stop e Play. Para criar uma sintaxe consistente, a função getTimer do Flash 5 e todas as ações agora exigem parênteses. Esses parênteses são adicionados automaticamente durante a conversão.
- Quando a função getProperty é executada em um clipe de filme que não existe, ela retorna o valor undefined, não 0, no Flash 5. E undefined == 0 é false no ActionScript do Flash 5. O Flash corrige esse problema ao converter os arquivos do Flash 4 introduzindo funções Number em comparações de igualdade. No seguinte exemplo, Number força undefined a ser convertido em 0 para que a comparação tenha êxito:

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

**Observação:** Se você tiver usado quaisquer palavras-chave do Flash 5 como nomes de variáveis no ActionScript do Flash 4, a sintaxe retornará um erro no Flash 5. Para corrigir isso, renomeie as variáveis em todas as suas ocorrências. Consulte "Palavras-chave", na página 53.

## Usando o Flash 5 para criar o conteúdo do Flash 4

Se você estiver usando o Flash 5 para criar conteúdo para o Flash 4 Player (exportando como Flash 4), você não conseguirá tirar proveito de todos os recursos novos do ActionScript do Flash 5. No entanto, muitos recursos novos do ActionScript ainda estão disponíveis. O ActionScript do Flash 4 possui apenas um tipo de dados básico e primitivo que é usado para a manipulação numérica e de sequência de caracteres. Ao criar um filme para o Flash 4 Player, você precisa usar os operadores de sequência de caracteres obsoletos localizados na categoria Operadores de Sequência de Caracteres na caixa de ferramentas.

Você pode usar os seguintes recursos do Flash 5 ao exportar para o formato de arquivo SWF do Flash 4:

- O operador de acesso de objeto e matriz (`[]`).
- O operador ponto (`.`).
- Operadores lógicos, operadores de atribuição e operadores de pré-incremento e pós-incremento/decremento.
- O operador de módulo (`%`), todos os métodos e propriedades do objeto Math.

Esses operadores e essas funções não são suportados nativamente pelo Flash 4 Player. O Flash 5 deve exportá-los como aproximações de séries. Isso significa que os resultados são aproximados. Além disso, devido à inclusão de aproximações de séries no arquivo SWF, essas funções ocupam mais espaço nos arquivos SWF do Flash 4 do que nos arquivos SWF do Flash 5.

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



Os seguintes recursos do Flash 5 não podem ser usados em filmes exportados para o formato de arquivo SWF do Flash 4:

- Funções personalizadas
- Suporte a XML
- Variáveis locais
- Objetos predefinidos (exceto Math)
- Ações de clipes de filme
- Vários tipos de dados
- eval com sintaxe de ponto (por exemplo, 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





## CAPÍTULO 3

### Criando interatividade com o ActionScript

.....

Um filme interativo envolve o seu público. Usando o teclado, o mouse ou os dois, o seu público pode ir para diferentes partes dos filmes, mover objetos, inserir informações, clicar em botões e executar várias outras operações interativas.

Crie filmes interativos configurando scripts que são executados quando eventos específicos ocorrem. Os eventos que disparam um script ocorrem quando a reprodução atinge um quadro, quando um clipe de filme é carregado ou descarregado, ou quando o usuário clica em um botão ou pressiona teclas do teclado. Use o ActionScript para criar scripts que informem ao Flash qual ação executar quando o evento ocorre.

As ações básicas a seguir são maneiras comuns de controlar a navegação e a interação com o usuário em um filme:

- Reproduzindo e parando filmes
- Ajustando a qualidade de exibição de um filme
- Parando todos os sons
- Indo para um quadro ou uma cena
- Indo para uma URL diferente
- Verificando se um quadro está carregado
- Carregando e descarregando filmes adicionais

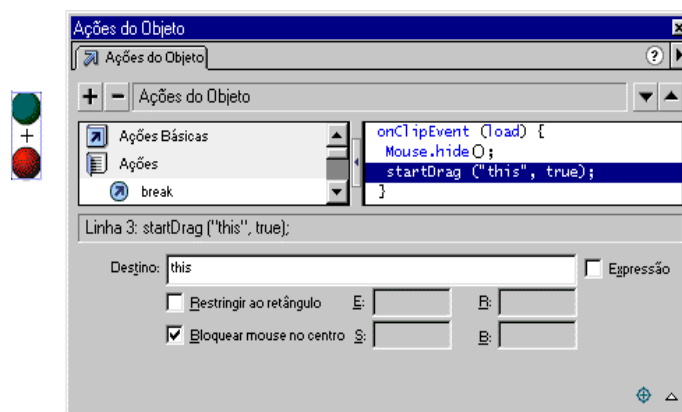
Para obter informações detalhadas sobre essas ações, consulte *Usando o Flash*.

Para criar uma interatividade mais complexa, é necessário compreender as seguintes técnicas:

- Criando um cursor personalizado
- Obtendo a posição do mouse
- Capturando pressionamentos de teclas
- Criando um campo de texto de rolagem
- Definindo valores de cores
- Criando controles de som
- Detectando colisões

## Criando um cursor personalizado

Para ocultar o cursor padrão (isto é, a representação do ponteiro do mouse na tela), use o método `hide` do objeto `Mouse` predefinido. Para usar um clipe de filme como o cursor personalizado, use a ação `startDrag`.



*Ações anexadas a um clipe de filme para criar um cursor personalizado.*



**Para criar um cursor personalizado:**

- 1 Crie um clipe de filme para ser usado como um cursor personalizado.
- 2 Selecione a instância do clipe de filme no Palco.
- 3 Escolha Janela > Ações para abrir o painel Ações do Objeto.
- 4 Na lista Caixa de Ferramentas, selecione Objetos, selecione Mouse e arraste hide para a janela Script.

O código deve ser este:

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

- 5 Na lista Caixa de Ferramentas, selecione Ações; depois, arraste startDrag para a janela Script.

- 6 Selecione a caixa Bloquear Mouse no Centro.

O código deve ser este:

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

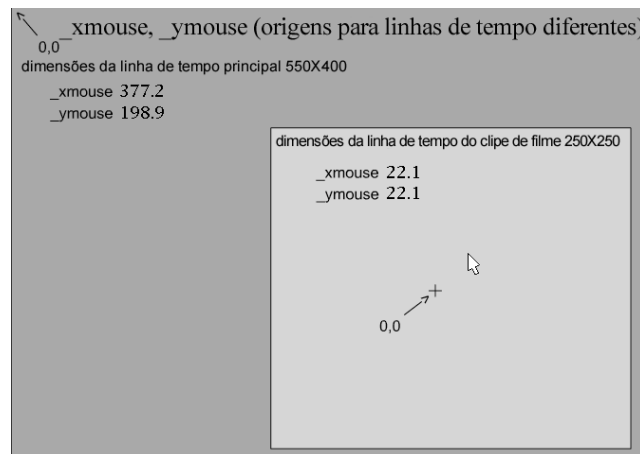
- 7 Escolha Controlar > Testar Filme para usar o cursor personalizado.

Os botões ainda funcionarão quando você usar um cursor personalizado. Recomenda-se colocar o cursor personalizado na camada superior da Linha de Tempo para que ele seja movido na frente dos botões e de outros objetos à medida que você mover o mouse no filme.

Para obter mais informações sobre os métodos do objeto Mouse, consulte suas entradas no Capítulo 7, “Dicionário do ActionScript”.

## Obtendo a posição do mouse

Você pode usar as propriedades `_xmouse` e `_ymouse` para localizar o ponteiro do mouse (cursor) em um filme. Cada Linha de Tempo possui uma propriedade `_xmouse` e `_ymouse` que retorna a localização do mouse no seu sistema de coordenadas.



*As propriedades `_xmouse` e `_ymouse` na Linha de Tempo principal e na Linha de Tempo de um clipe de filme.*

O comando a seguir poderia ser colocado em qualquer Linha de Tempo no filme `_level0` para retornar a posição `_xmouse` na Linha de Tempo principal:

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

Para determinar a posição do mouse em um clipe de filme, você pode usar o nome da instância do clipe de filme. Por exemplo, o comando a seguir poderia ser colocado em qualquer Linha de Tempo no filme `_level0` para retornar a posição `_ymouse` na instância `myMovieClip`:

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

Você também pode determinar a posição do mouse em um clipe de filme usando as propriedades `_xmouse` e `_ymouse` em uma ação do clipe, conforme mostrado a seguir:

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

As variáveis `x_pos` e `y_pos` são usadas como recipientes para armazenar os valores das posições do mouse. Você poderia usar essas variáveis em qualquer script em seu filme. No exemplo a seguir, os valores de `x_pos` e `y_pos` são atualizados toda vez que o usuário move o mouse.

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

Para obter mais informações sobre as propriedades `_xmouse` e `_ymouse`, consulte suas entradas no Capítulo 7, “Dicionário do ActionScript.”

## Capturando pressionamentos de teclas

Você pode usar os métodos do objeto `Key` predefinido para detectar a última tecla pressionada pelo usuário. O objeto `Key` não requer uma função de construção; para usar os seus métodos, basta chamar o próprio objeto, como no exemplo a seguir:

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

Você pode obter códigos de teclas virtuais ou valores ASCII de pressionamentos de teclas:

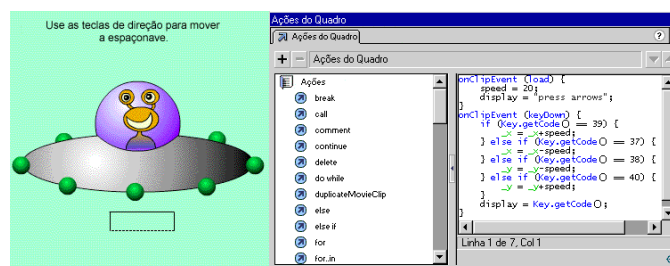
- Para obter o código de tecla virtual da última tecla pressionada, use o método `getCode`.
- Para obter o valor ASCII da última tecla pressionada, use o método `getAscii`.

Um código de tecla virtual é atribuído a cada tecla física de um teclado. Por exemplo, a tecla de direção para a esquerda possui o código de tecla virtual 37. Usando um código de tecla virtual, você pode garantir que os controles de seu filme sejam idênticos em todos os teclados, independentemente do idioma ou da plataforma.

Valores ASCII (American Standard Code for Information Interchange) são atribuídos aos 127 primeiros caracteres em todos os conjuntos de caracteres. Os valores ASCII fornecem informações sobre um caractere da tela. Por exemplo, a letra “A” e a letra “a” possuem valores ASCII diferentes.



Um lugar comum para o uso de `Key.getCode` é em um manipulador `onClipEvent`. Passando `keyDown` como o parâmetro, o manipulador instrui o ActionScript a verificar o valor da última tecla pressionada somente quando uma tecla é realmente pressionada. Este exemplo usa `Key.getCode` em um comando `if` para criar controles de navegação para a nave espacial.



**Para criar controles do teclado para um filme:**

- 1 Decida quais teclas usar e determine seus códigos de teclas virtuais usando uma destas abordagens:
  - Consulte a lista de códigos de teclas no Apêndice B, “Teclas do teclado e valores de códigos de teclas”.
  - Use um objeto `Key` constante. (Na lista Caixa de Ferramentas, selecione Objetos e, em seguida, selecione `Key`. As constantes são listadas com todas as letras em maiúsculas.)
  - Atribua a seguinte ação do clipe, escolha Controlar > Testar Filme e pressione a tecla desejada:

```
onClipEvent(keyDown) {
    trace(Key.getCode());
}
```

- 2 Selecione um clipe de filme no Palco.
- 3 Escolha Janela > Ações.
- 4 Clique duas vezes na ação `onClipEvent` na categoria Ações da caixa de ferramentas.
- 5 Escolha o evento Tecla pressionada no painel de parâmetros.
- 6 Clique duas vezes na ação `if` na categoria Ações da caixa de ferramentas.
- 7 Clique no parâmetro Condition, selecione Objetos e, em seguida, selecione `Key` e `getCode`.
- 8 Clique duas vezes no operador de igualdade (`==`) na categoria Operadores da caixa de ferramentas.

- 9 Insira o código de tecla virtual à direita do operador de igualdade.

O código deve ser este:

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

- 10 Selecione uma ação para ser executada se a tecla correta for pressionada.

Por exemplo, a ação a seguir faz com que a Linha de Tempo principal vá para o próximo quadro quando a Barra de espaços (32) é pressionada:

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

Para obter mais informações sobre os métodos do objeto Key, consulte suas entradas no Capítulo 7, “Dicionário do ActionScript”.

## Criando um campo de texto de rolagem

Você pode usar as propriedades scroll e maxscroll para criar um campo de texto de rolagem.

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

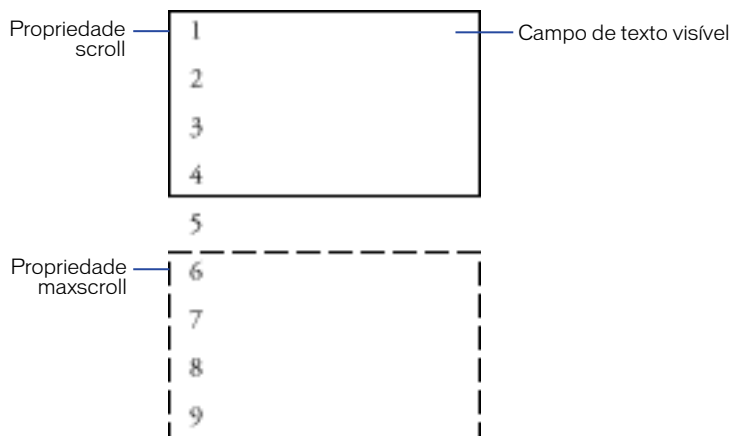


No painel Opções de Texto, você pode atribuir uma variável a qualquer campo de texto definido como Texto de Entrada ou Texto Dinâmico. O campo de texto funciona como uma janela que exibe o valor dessa variável.

Cada variável associada a um campo de texto possui uma propriedade scroll e maxscroll. Você pode usar essas propriedades para rolar o texto em um campo de texto. A propriedade scroll retorna o número da primeira linha visível em um campo de texto; você pode defini-la e recuperá-la. A propriedade maxscroll retorna a primeira linha visível em um campo de texto quando a linha inferior do texto está visível; você pode ler, porém não definir, essa propriedade.



Por exemplo, suponha que você tenha um campo de texto de quatro linhas. Se ele contiver a variável `speech`, que preencheria nove linhas do campo de texto, somente parte dessa variável poderá ser exibida de uma vez (identificada pela caixa sólida):

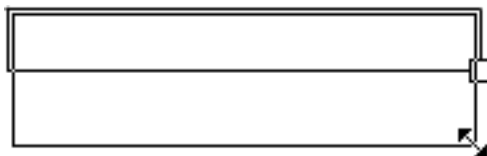


Você pode acessar essas propriedades usando a sintaxe de ponto, conforme mostrado a seguir:

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

#### Para criar um campo de texto de rolagem:

- 1 Arraste um campo de texto no Palco.
- 2 Escolha Janela > Painéis > Opções de Texto.
- 3 Escolha Texto de Entrada no menu pop-up.
- 4 Insira o nome de variável `text` no campo Variável.
- 5 Arraste o canto inferior direito do campo de texto para redimensionar o campo.

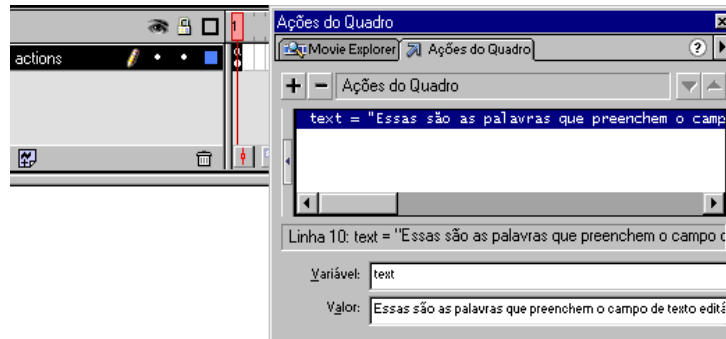




6 Escolha Janela > Ações.

7 Selecione o quadro 1 na Linha de Tempo principal e atribua uma ação `set variable` que defina o valor de `text`.

Nenhum texto será exibido no campo até que a variável seja definida. Portanto, embora você possa atribuir essa ação a qualquer quadro, botão ou clipe de filme, recomenda-se atribuí-la ao quadro 1 na Linha de Tempo principal, conforme mostrado aqui:



8 Escolha Janela > Bibliotecas Comuns > Botões e arraste um botão para o Palco.

9 Pressione Alt (Windows) ou Option (Macintosh) e arraste o botão para criar uma cópia.

10 Selecione o botão superior e escolha Janela > Ações.

11 Arraste a ação `set variables` da caixa de ferramentas para a janela Script no painel Ações.

12 Insira `text.scroll` na caixa Variável.

13 Insira `text.scroll -1` na caixa Valor e marque a caixa de seleção Expressão.

14 Selecione o botão Seta para baixo e atribua a seguinte ação `set variables`:

```
text.scroll = text.scroll+1;
```

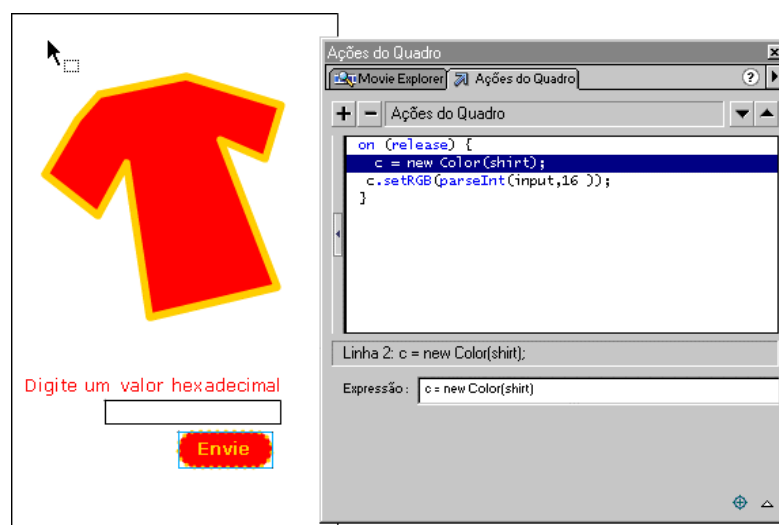
15 Escolha Controlar > Testar Filme para testar o campo de texto de rolagem.

Para obter mais informações sobre as propriedades `scroll` e `maxscroll`, consulte suas entradas no Capítulo 7, "Dicionário do ActionScript".



## Definindo valores de cores

Você pode usar os métodos do objeto `Color` predefinido para ajustar a cor de um clipe de filme. O método `setRGB` atribui valores RGB (vermelho, verde, azul) hexadecimais ao objeto, e o método `setTransform` define a porcentagem e os valores de desvio para os componentes de transparência (alpha), vermelho, verde e azul de uma cor. O exemplo a seguir usa `setRGB` para alterar a cor de um objeto com base na entrada do usuário.



*A ação do botão cria um objeto `Color` e altera a cor da camiseta com base na entrada do usuário.*

Para usar o objeto `Color`, você precisa criar uma instância do objeto e aplicá-la a um clipe de filme.

**Para definir o valor de cor de um clipe de filme:**

- 1 Selecione um clipe de filme no Palco e escolha Janela > Painéis > Instância.
- 2 Insira o nome da instância **colorTarget** na caixa Nome.
- 3 Arraste um campo de texto no Palco.
- 4 Escolha Janela > Painéis > Opções de Texto e atribua a ele o nome de variável **input**.
- 5 Arraste um botão para o Palco e selecione-o.



- 6 Escolha Janela > Ações.
- 7 Arraste a ação `set variable` da caixa de ferramentas para a janela Script.
- 8 Na caixa Variável, insira `c`.
- 9 Na caixa de ferramentas, selecione Objetos, Cor e arraste novo `Color` para a caixa Valor.
- 10 Marque a caixa de seleção Expressão.
- 11 Clique no botão Target Path e selecione `colorTarget`. Clique em OK.

O código na janela Script deve ser este:

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

- 12 Arraste a ação `evaluate` da caixa de ferramentas para a janela Script.
- 13 Insira `c` na caixa Expressão.
- 14 Na categoria Objetos da lista Caixa de Ferramentas, selecione Cor; depois, arraste `setRGB` para a caixa Expressão.
- 15 Selecione Functions e arraste `parseInt` para a caixa Expressão.

O código deve ser este:

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

- 16 Para o argumento de sequência de caracteres `parseInt`, insira `input`.

A sequência de caracteres a ser analisada é o valor inserido no campo de texto editável.

- 17 Para o argumento `radix` `parseInt`, insira `16`.

O `radix` é a base do sistema numérico a ser analisado. Neste caso, `16` é a base do sistema hexadecimal que o objeto `Color` usa. O código deve ser este:

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

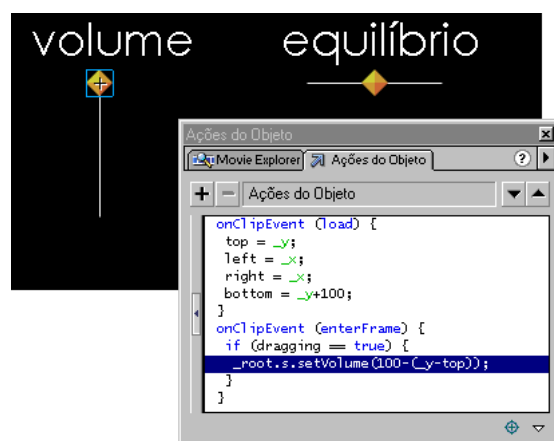
- 18 Escolha Controlar > Testar Filme para alterar a cor do clipe de filme.

Para obter mais informações sobre os métodos do objeto `Color`, consulte suas entradas no Capítulo 7, “Dicionário do ActionScript”.

## Criando controles de som

Para controlar os sons em um filme, use o objeto Sound predefinido. Para usar os métodos do objeto Sound, primeiro é necessário criar um novo objeto Sound. Depois, você poderá usar o método `attachSound` para inserir um som da biblioteca em um filme enquanto o filme está sendo executado.

O método `setVolume` do objeto Sound controla o volume e o método `setPan` ajusta o balanço da esquerda e da direita de um som. O exemplo a seguir usa `setVolume` e `setPan` para criar controles de volume e de balanço que o usuário pode ajustar.



*Quando o usuário arrasta o controle deslizante de volume, o método `setVolume` é chamado.*

**Para anexar um som a uma Linha de Tempo:**

- 1 Escolha Arquivo > Importar para importar um som.
- 2 Selecione o som na biblioteca e escolha Vinculação no menu Opções.
- 3 Selecione Exportar este Símbolo e atribua a ele o identificador **mySound**.
- 4 Selecione o quadro 1 na Linha de Tempo principal e escolha Janela > Ações.
- 5 Arraste a ação `set variable` da caixa de ferramentas para a janela Script.
- 6 Insira `s` na caixa Valor.
- 7 Na lista Caixa de Ferramentas, selecione Objetos, selecione Som e arraste `new Sound` para a caixa Valor.

O código deve ser este:

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

- 8 Clique duas vezes na ação `evaluate` na caixa de ferramentas.
- 9 Insira `s` na caixa Expressão.
- 10 Na categoria Objetos da lista Caixa de Ferramentas, selecione Som e arraste `attachSound` para a caixa Expressão.
- 11 Insira **“mySound”** no argumento de identificação de `attachSound`.
- 12 Clique duas vezes na ação `evaluate` na caixa de ferramentas.
- 13 Insira `s` na caixa Expressão.
- 14 Na categoria Objetos, selecione Som e arraste `start` para a caixa Expressão.

O código deve ser este:

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

- 15 Escolha Controlar > Testar Filme para ouvir o som.

**Para criar um controle de volume deslizante:**

- 1 Arraste um botão para o Palco.
- 2 Selecione o botão e escolha Inserir > Converter em Símbolo. Escolha o comportamento do clipe de filme.

Isso criará um clipe de filme com o botão em seu primeiro quadro.

- 3 Selecione o clipe de filme e escolha Editar > Editar Símbolo.
- 4 Selecione o botão e escolha Janela > Ações.
- 5 Insira as seguintes ações:

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

Os parâmetros `left`, `top`, `right` e `bottom` de `startDrag` são variáveis definidas em uma ação do clipe.

- 6 Escolha Editar > Editar Filme para retornar à Linha de Tempo principal.
- 7 Selecione o clipe de filme no Palco.

**8** Insira as seguintes ações:

```
onClipEvent (load) {
    top=_y;
    left=_x;
    right=_x;
    bottom=_y+100;
}

onClipEvent(enterFrame) {
    if (dragging==true){
        _root.s.setVolume(100-(_y-top));
    }
}
```

**9** Escolha Controlar > Testar Filme para usar o controle deslizante de volume.

**Para criar um controle deslizante de balanço:**

- 1** Arraste um botão para o Palco.
- 2** Selecione o botão e escolha Inserir > Converter em Símbolo. Escolha a propriedade do clipe de filme.
- 3** Selecione o clipe de filme e escolha Editar > Editar Símbolo.
- 4** Selecione o botão e escolha Janela > Ações.
- 5** Insira as seguintes ações:

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

on (release, releaseOutside) {
    stopDrag ();
    dragging = false;
}
```

Os parâmetros `left`, `top`, `right` e `bottom` de `startDrag` são variáveis definidas em uma ação do clipe.

- 6** Escolha Editar > Editar Filme para retornar à Linha de Tempo principal.
- 7** Selecione o clipe de filme no Palco.

**8** Insira as seguintes ações:

```
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** Escolha Controlar > Testar Filme para usar o controle deslizante de balanço.

Para obter mais informações sobre os métodos do objeto Sound, consulte suas entradas no Capítulo 7, “Dicionário do ActionScript”.

## Detectando colisões

Você pode usar o método `hitTest` do objeto `MovieClip` para detectar colisões em um filme. O método `hitTest` verifica se um objeto colidiu com um clipe de filme e retorna um valor booleano (`true` ou `false`). Você pode usar os parâmetros do método `hitTest` para especificar as coordenadas *x* e *y* de uma área de acertos no Palco ou usar o caminho de destino de outro clipe de filme como uma área de acertos.

Cada clipe em um filme é uma instância do objeto `MovieClip`. Isso permite que você chame métodos do objeto a partir de qualquer instância, conforme mostrado a seguir:

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

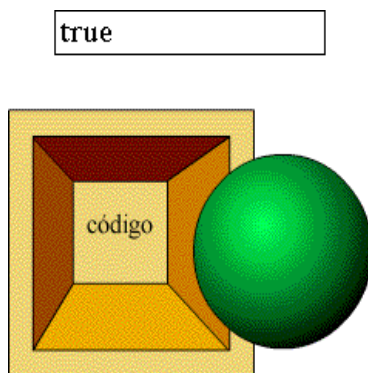


Você pode usar o método `hitTest` para testar a colisão de um clipe de filme e um ponto único.



*“True” aparece no campo de texto sempre que o cursor está sobre a área colorida.*

Você também pode usar o método `hitTest` para testar uma colisão entre dois cliques de filme.



*“True” aparece no campo de texto sempre que um clipe de filme toca no outro.*

**Para detectar a colisão entre um clipe de filme e um ponto no Palco:**

- 1** Selecione um clipe de filme no Palco.
- 2** Escolha Janela > Ações para abrir o painel Ações do Objeto.
- 3** Clique duas vezes em `trace` na categoria Ações na caixa de ferramentas.

- 4 Marque a caixa de seleção Expressão e insira o seguinte nessa caixa:

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

Este exemplo usa as propriedades `_xmouse` e `_ymouse` como as coordenadas *x* e *y* da área de acertos e envia os resultados para a janela Saída no modo de teste de filme. Você também pode definir um campo de texto no Palco para exibir os resultados ou usar os resultados em um comando `if`.

- 5 Escolha Controlar > Testar Filme e mova o mouse sobre o clipe de filme para testar a colisão.

**Para detectar a colisão em dois clipes de filme:**

- 1 Arraste dois clipes de filme para o Palco e atribua a eles os nomes de instância `mcHitArea` e `mcDrag`.
- 2 Crie um campo de texto no Palco e insira `status` na caixa Text Options Variable.
- 3 Selecione `mcHitArea` e escolha Janela > Ações.
- 4 Clique duas vezes em `evaluate` na caixa de texto.
- 5 Insira o código a seguir na caixa Expressão selecionando itens na caixa de ferramentas:  

```
_root.status=this.hitTest(_root.mcDrag);
```
- 6 Selecione a ação `onClipEvent` na janela Script e escolha `enterFrame` como o evento.
- 7 Selecione `mcDrag` e escolha Janela > Ações.
- 8 Clique duas vezes em `startDrag` na caixa de ferramentas.
- 9 Marque a caixa de seleção Bloquear Mouse no Centro.
- 10 Selecione a ação `onClipEvent` na janela Script e escolha o evento Mouse clicado.
- 11 Clique duas vezes em `stopDrag` na caixa de ferramentas.
- 12 Selecione a ação `onClipEvent` na janela Script e escolha o evento Mouse liberado.
- 13 Escolha Controlar > Testar Filme e arraste o clipe de filme para testar a detecção da colisão.

Para obter mais informações sobre o método `hitTest`, consulte sua entrada no Capítulo 7, “Dicionário do ActionScript”.







## CAPÍTULO 4

### Trabalhando com clipes de filme

.....

Um clipe de filme é um minifilme do Flash: ele possui suas próprias propriedades e Linha de Tempo. Um símbolo de clipe de filme da Biblioteca pode ser usado várias vezes em um filme do Flash; cada uso é chamado de *instância* do clipe de filme. Os clipes de filme podem estar aninhados uns nos outros. Para diferenciar uma instância da outra, você pode atribuir um nome a cada uma delas.

Qualquer objeto pode ser colocado na Linha de Tempo de um clipe de filme, incluindo outros clipes de filme. Os filmes que são carregados no Flash Player com `LoadMovie` também são minifilmes do Flash. Cada clipe de filme, cada filme carregado e a Linha de Tempo principal em um filme do Flash são objetos com propriedades e métodos que podem ser manipulados pelo ActionScript para criar uma animação não-linear e complexa e uma interatividade poderosa.

Você controla os clipes de filme usando ações e métodos do objeto `MovieClip`. As ações e os métodos podem ser anexados a quadros ou botões em um clipe de filme (ações de quadro e de botão), ou a uma instância específica de clipe de filme (ações do clipe). As ações de um clipe de filme podem controlar qualquer Linha de Tempo em um filme. Para controlar uma Linha de Tempo, você deve endereçá-la usando um caminho de destino. O caminho de destino indica a localização da Linha de Tempo no filme.

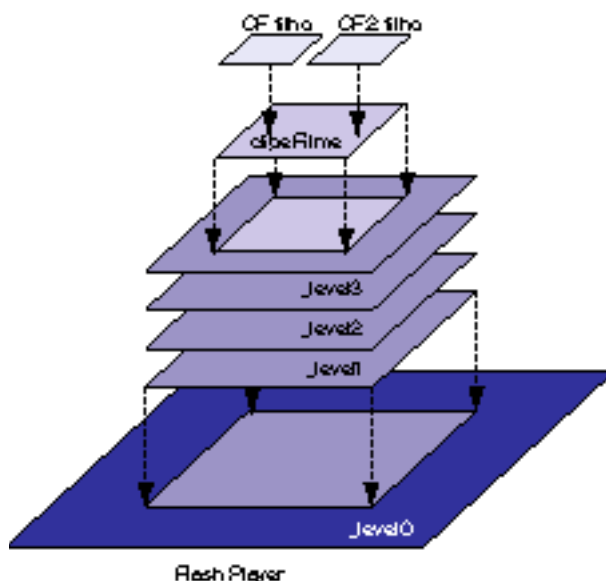
Também é possível transformar um clipe de filme em um Clipe Inteligente; um clipe de filme com o ActionScript que pode ser reprogramado sem o uso do painel de Ações. Os clipes inteligentes facilitam a transferência de objetos com lógica ActionScript complexa de um programador para um designer.



## Sobre várias Linhas de Tempo

Todo filme do Flash possui uma Linha de Tempo principal localizada no nível 0 do Flash Player. Você pode usar a ação `loadMovie` para carregar outros filmes do Flash (arquivos SWF) no Flash Player em qualquer nível acima do 0 (por exemplo, nível 1, nível 2, nível 15). Cada filme carregado em um nível do Flash Player possui uma Linha de Tempo.

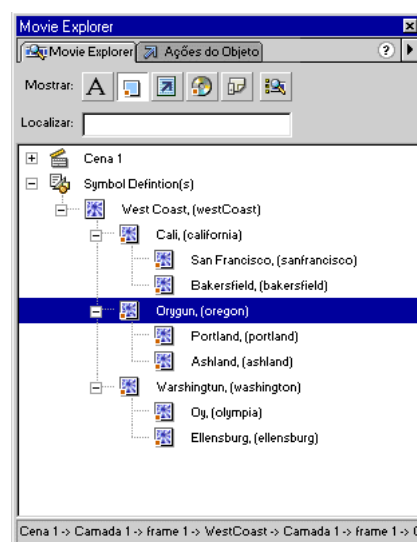
Os filmes do Flash de qualquer nível podem ter instâncias de clipes de filme em suas Linhas de Tempo. Cada instância de clipe de filme também possui uma Linha de Tempo e pode conter outros clipes de filme que também possuam Linhas de Tempo. As Linhas de Tempo dos clipes de filme e os níveis do Flash Player são organizados hierarquicamente para que você possa organizar e controlar facilmente os objetos de seu filme.



*A hierarquia de níveis e clipes de filme no Flash Player.*



No Flash, essa hierarquia de níveis e cliques de filme é denominada *lista de exibição*. Você poderá visualizar a lista de exibição no Movie Explorer quando estiver trabalhando no Flash. Você pode exibir essa lista no Depurador ao reproduzir o filme no modo de testar filme, no Flash Player independente ou em um navegador da Web.



*O Movie Explorer mostra a hierarquia de Linhas de Tempo denominada lista de exibição.*

As Linhas de Tempo de um filme do Flash são objetos e todas possuem as características (propriedades) e as habilidades (métodos) do objeto MovieClip predefinido. As Linhas de Tempo apresentam relações específicas entre si, dependendo de sua localização na lista de exibição. As Linhas de Tempo aninhadas em outras são afetadas pelas alterações efetuadas na Linha de Tempo na qual residem. Por exemplo, se `portland` for filho de `oregon` e você alterar a propriedade `_xscale` de `oregon`, `portland` também será dimensionado.

As Linhas de Tempo também podem enviar mensagens umas para as outras. Por exemplo, uma ação no último quadro de um clipe de filme pode informar que outro clipe de filme deve ser reproduzido.

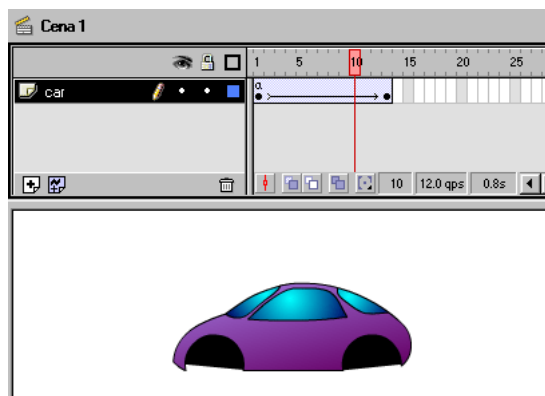
## Sobre a relação hierárquica de Linhas de Tempo

Quando você coloca uma instância de clipe de filme na Linha de Tempo de outro clipe de filme, um símbolo de clipe de filme contém a instância do outro clipe de filme — o primeiro clipe de filme é o *filho* e o segundo é o seu *pai*. A Linha de Tempo principal de um filme do Flash é o pai de todos os cliques de filme do seu nível.

As relações pai-filho de cliques de filme são hierárquicas. Para compreender essa hierarquia, considere a hierarquia de um computador: a unidade de disco rígido contém um diretório (ou uma pasta) raiz e subdiretórios. O diretório raiz é semelhante à Linha de Tempo principal de um filme do Flash: ele é o pai de todos os outros elementos. Os subdiretórios são semelhantes a cliques de filme. Você pode usar subdiretórios para organizar um conteúdo relacionado.

Da mesma forma, você pode usar a hierarquia de cliques de filme do Flash para organizar objetos visuais relacionados, geralmente de maneiras semelhantes ao comportamento dos objetos do mundo real. Qualquer alteração efetuada em um clipe de filme pai também será efetuada em seus filhos.

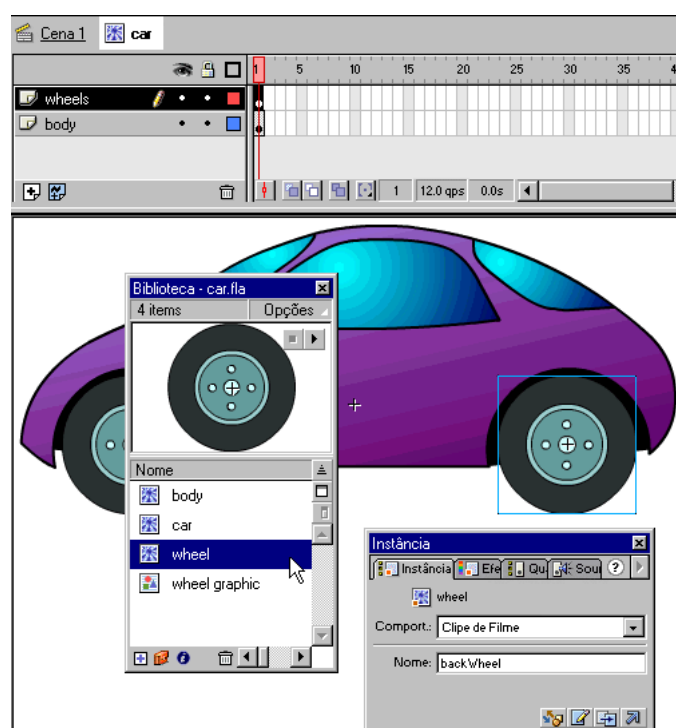
Por exemplo, você pode criar um filme do Flash que contenha um carro que se movimenta no Palco. Você pode usar um símbolo de clipe de filme para representar o carro e configurar uma interpolação de movimento para movê-lo no Palco.



*Uma interpolação de movimento move o clipe de filme do carro na Linha de Tempo principal.*



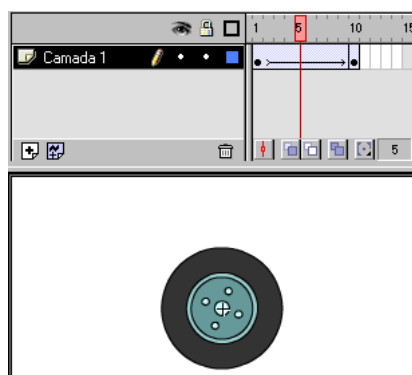
O carro é exibido lateralmente, com duas rodas visíveis. Quando o carro estiver em movimento, você desejará adicionar rodas que girem. Você criará então um clipe de filme para uma roda de carro e criará duas instâncias desse clipe, chamadas `frontWheel` e `backWheel`. Em seguida, você colocará as rodas na Linha de Tempo do clipe de filme do carro — não na Linha de Tempo principal. Como filhos de `car`, `frontWheel` e `backWheel` são afetados por todas as alterações efetuadas em `car`. Isso significa que eles se moverão com o carro à medida que ele se movimentar no Palco.



*As instâncias `frontWheel` e `backWheel` são colocadas na Linha de Tempo do clipe de filme `car`.*



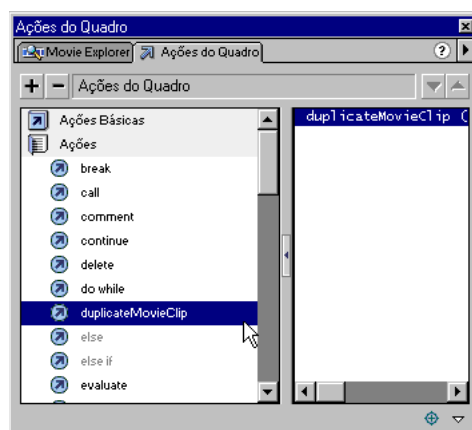
Para que as rodas girem, você pode configurar uma interpolação de movimento para girar o símbolo de roda de modo que as duas instâncias girem. Mesmo depois que você alterar `frontWheel` e `backWheel`, elas continuarão a ser afetadas pela interpolação em seu clipe de filme pai, `car`; as rodas girarão, mas também se moverão com o clipe de filme pai `car` no Palco.



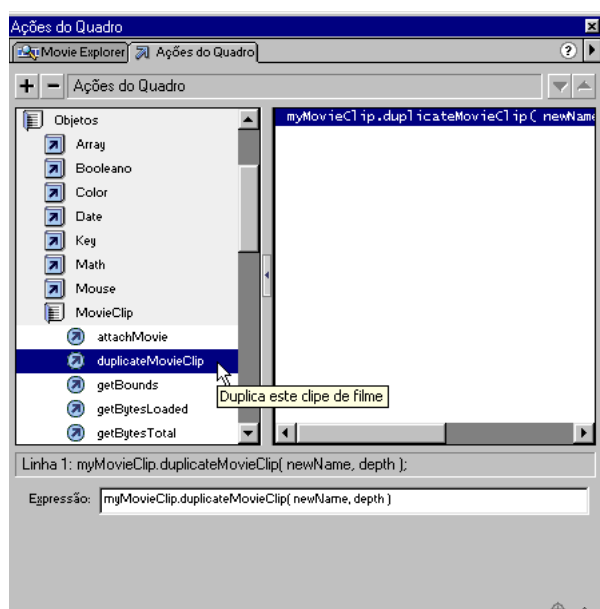
## Enviando mensagens entre Linhas de Tempo

Você pode enviar mensagens de uma Linha de Tempo para outra. Uma Linha de Tempo contém a ação, chamada *controlador*, e a outra recebe a ação, chamada *destino*. Você pode atribuir uma ação a um quadro ou botão em uma Linha de Tempo ou, se a Linha de Tempo for um clipe de filme, ao próprio clipe de filme.

Para especificar Linhas de Tempo como destino, você pode usar as ações da categoria Ações ou usar métodos do objeto `MovieClip` na categoria Objetos do painel Ações. Por exemplo, você pode usar a ação `duplicateMovieClip` como destino e fazer cópias de instâncias de um clipe de filme enquanto o filme é reproduzido.



*Você pode usar as ações da categoria Ações para especificar uma Linha de Tempo como destino.*



*Você pode usar os métodos do objeto MovieClip para especificar uma Linha de Tempo como destino.*

Para executar várias ações no mesmo destino, você pode usar a ação `with`. De maneira semelhante ao comando `with` do JavaScript, a ação `with` do ActionScript permite o endereçamento da Linha de Tempo de destino uma vez e, depois, a execução de várias ações nesse clipe; não é necessário endereçar a Linha de Tempo de destino em cada ação.

Você também pode usar a ação `tellTarget` para executar várias ações no mesmo destino.

Para realizar a comunicação entre Linhas de Tempo, siga este procedimento:

- Insira um nome de instância para o clipe de filme de destino.

Para nomear uma instância do clipe de filme, use o Painel de Instância (Janela > Painéis > Instância). As Linhas de Tempo carregadas em níveis usam o número de seu nível como um nome de instância, por exemplo, `_level6`.

- Insira o caminho de destino para o nome da instância no Painel de Ações.

Você pode inserir o caminho de destino manualmente ou usar a caixa de diálogo Inserir Caminho de Destino para especificar um clipe de filme como destino. (Consulte “Especificando caminhos de destino”, na página 121).

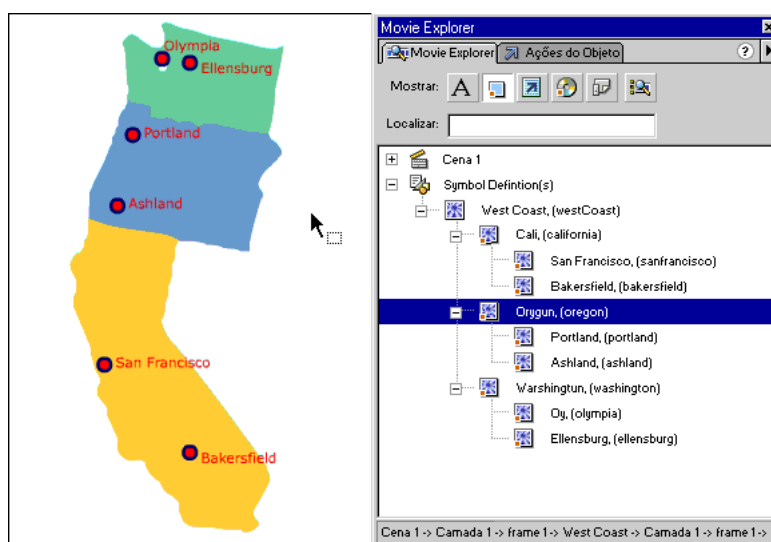
**Observação:** Durante a reprodução, a Linha de Tempo de um clipe de filme deve estar no Palco para que seja especificada como destino.





## Sobre caminhos de destino absolutos e relativos

Um caminho de destino é o endereço da Linha de Tempo que você deseja especificar como destino. A lista de exibição de Linhas de Tempo do Flash é semelhante à hierarquia de arquivos e pastas de um servidor da Web.



*O Movie Explorer mostra a lista de exibição de clipes de filme no modo de criação.*

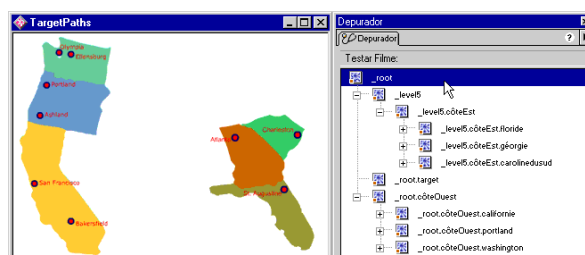
Como em um servidor da Web, cada Linha de Tempo do Flash pode ser endereçada de duas maneiras: com um caminho absoluto ou um caminho relativo. O caminho absoluto de uma instância é sempre o mesmo, independentemente da Linha de Tempo que chama a ação; por exemplo, o caminho absoluto para a instância `california` é sempre `_level0.westCoast.california`. O caminho relativo varia de acordo com o local de onde é chamado; por exemplo, o caminho relativo para `california` a partir de `sanfrancisco` é `_parent`, mas a partir de `portland`, é `_parent._parent.california`.

**Observação:** Para obter mais informações sobre o Movie Explorer, consulte *Usando o Flash*.

O caminho absoluto se inicia com o nome do nível no qual o filme é carregado e continua através da lista de exibição até chegar à instância de destino.

O primeiro filme a ser aberto no Flash Player é carregado no nível 0. É necessário atribuir a cada filme carregado adicional um número de nível. O nome do destino referente a um nível é `_levelX`, onde `X` é o número do nível em que o filme é carregado. Por exemplo, o primeiro filme aberto no Flash Player é chamado `_level0`, um filme carregado no nível 3 é chamado `_level3`.

No exemplo a seguir, dois filmes foram carregados no exibidor, `TargetPaths.swf` no nível 0 e `EastCoast.swf` no nível 5. Os níveis são indicados no Depurador, sendo que o nível 0 é indicado como `_root`.



*O Depurador mostra os caminhos absolutos de todas as Linhas de Tempo na lista de exibição, no modo de testar filme.*

Uma instância tem sempre o mesmo caminho absoluto, quer seja chamada a partir de uma ação em uma instância do mesmo nível ou de uma ação em outro nível. Por exemplo, a instância `bakersfield` do nível 0 tem sempre o seguinte caminho absoluto na sintaxe de ponto:

```
_level0.california.bakersfield
```

Na sintaxe de barra, os pontos são substituídos por barras no caminho absoluto, conforme mostrado a seguir:

```
_level0/california/bakersfield
```

Para a comunicação entre filmes em níveis diferentes, é necessário usar o nome do nível no caminho de destino. Por exemplo, a instância `portland` endereçaria a instância `atlanta` da seguinte maneira:

```
_level5.georgia.atlanta
```

Na sintaxe de ponto, você pode usar o alias `_root` para se referir à Linha de Tempo principal do nível atual. Para a Linha de Tempo principal, ou `_level0`, o alias `_root` significa `_level0` quando especificado como destino por um clipe que também está no `_level0`. Para um filme carregado no `_level5`, `_root` é igual a `_level5` quando especificado como destino por um clipe de filme também do nível 1. Por exemplo, uma ação chamada a partir da instância `southcarolina` poderia usar o seguinte caminho absoluto para especificar como destino a instância `florida`:

```
_root.eastCoast.florida
```

Na sintaxe de barra, você pode usar `/` para se referir à Linha de Tempo principal do nível atual, conforme mostrado a seguir:

```
/eastCoast/florida
```

Na sintaxe de ponto no modo absoluto ou relativo, você pode usar as mesmas regras de caminho de destino para identificar uma variável na Linha de Tempo ou uma propriedade de objeto. Por exemplo, o comando a seguir define o nome da variável no formulário da instância como o valor "Gilbert":

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

Na sintaxe de barra no modo absoluto ou relativo, você pode identificar uma variável em uma Linha de Tempo colocando dois-pontos (:) antes do nome da variável, conforme mostrado a seguir:

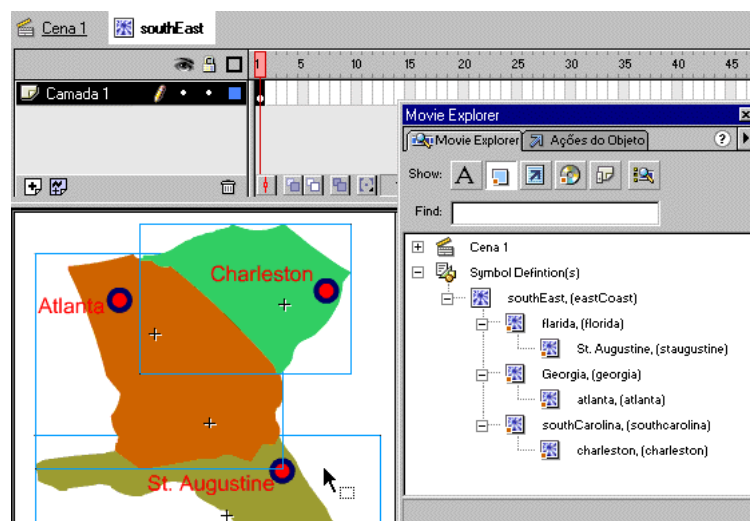
```
/form:name = "Gilbert";
```

O **caminho relativo** depende da relação entre a Linha de Tempo do controlador e a Linha de Tempo de destino. Você pode usar um caminho relativo para reutilizar ações, pois a mesma ação poderá especificar Linhas de Tempo diferentes como destino, dependendo de onde ela for colocada. Os caminhos relativos só podem endereçar destinos em seu próprio nível do Flash Player; eles não podem endereçar filmes carregados em outros níveis. Por exemplo, você não pode usar um caminho relativo em uma ação do `_level0` que especifica como destino uma Linha de Tempo no `_level5`.

Na sintaxe de ponto, você pode usar a palavra-chave `this` em um caminho de destino relativo para se referir à Linha de Tempo atual. Você pode usar o alias `_parent` em um caminho de destino relativo para indicar a Linha de Tempo pai da Linha de Tempo atual. O alias `_parent` pode ser usado repetidamente para subir um nível na hierarquia de clipes de filme, no mesmo nível do Flash Player. Por exemplo, `_parent._parent` controla um clipe de filme dois níveis acima na hierarquia.



No exemplo a seguir, cada cidade (charleston, atlanta e staugustine) é um filho de uma instância de estado e cada estado (southcarolina, georgia e florida) é um filho da instância eastCoast.



*O Movie Explorer mostra as relações pai-filho dos cliques de filme.*

Uma ação na Linha de Tempo da instância charleston pode usar o seguinte caminho de destino para especificar a instância southcarolina como destino:

`_parent`

Para especificar a instância eastCoast como destino a partir de uma ação em charleston, você pode usar o seguinte caminho relativo:

`_parent._parent`

Na sintaxe de barra, você pode usar dois pontos (..) para ir para um nível acima na hierarquia. Para especificar eastCoast como destino a partir de uma ação em charleston, você pode usar o seguinte caminho:

`../..`

Para especificar a instância atlanta como destino a partir de uma ação na Linha de Tempo de charleston, você pode usar o seguinte caminho relativo na sintaxe de ponto:

`_parent._parent.georgia.atlanta`

Os caminhos relativos são úteis para a reutilização de scripts. Por exemplo, você pode anexar um script a um clipe de filme que aumente esse clipe de filme em um nível a 150%, da seguinte maneira:

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

Em seguida, você pode reutilizar esse script colocando-o na Linha de Tempo de qualquer clipe de filme.

Para obter mais informações sobre endereçamento e sintaxe de ponto, consulte “Escrevendo scripts com o ActionScript”, na página 49.

Para obter mais informações sobre sintaxe de ponto e de barra, consulte “Usando a sintaxe do ActionScript”, na página 50.

## Especificando caminhos de destino

Para controlar um clipe de filme ou um filme carregado, você deve usar um caminho para especificar um destino. É necessário atribuir um nome de instância a um clipe de filme para especificá-lo como destino. Você pode especificar um destino de diversas maneiras:

- Insira um caminho de destino usando a caixa de diálogo e o botão Inserir Caminho de Destino no painel Ações.
- Insira manualmente o caminho de destino do clipe de filme no script.
- Crie uma expressão usando uma referência a um clipe de filme ou usando as funções predefinidas `targetPath` e `eval`.

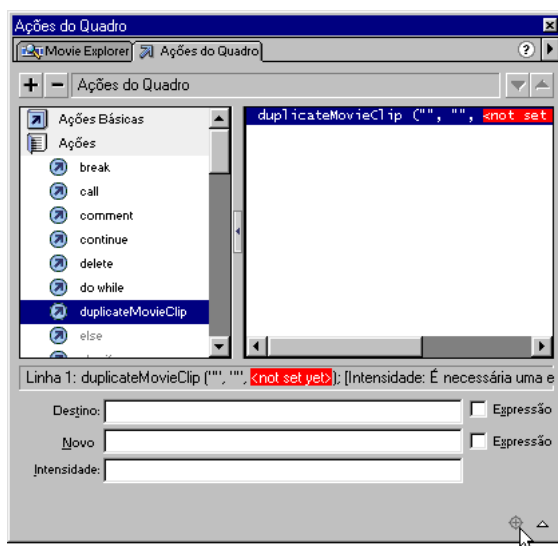
**Para inserir um caminho de destino usando a caixa de diálogo Inserir Caminho de Destino:**

- 1 Selecione a instância do clipe de filme, do quadro ou do botão à qual você deseja atribuir a ação.

Essa será a Linha de Tempo do controlador.

- 2 Escolha Janela > Ações para exibir o painel Ações
- 3 Na lista Caixa de Ferramentas, escolha uma ação na categoria Ações ou um método na categoria MovieClip dentro da categoria Objetos.
- 4 Clique no campo Destino ou no local do script para inserir o caminho de destino.

- 5 Clique no botão Inserir Caminho de Destino, no canto inferior direito do painel Ações, para exibir a caixa de diálogo Inserir Caminho de Destino.



- 6 No caixa de diálogo Inserir Caminho de Destino, escolha uma sintaxe: de ponto (o padrão) ou de barra.



- 7 Escolha Absoluto ou Relativo para o modo do caminho de destino.  
Consulte “Sobre caminhos de destino absolutos e relativos”, na página 117.
- 8 Especifique o seu destino seguindo um destes procedimentos:
- Selecione um clipe de filme na lista de exibição Inserir Caminho de Destino.
  - Insira um destino manualmente no campo Destino usando um caminho absoluto ou relativo e a sintaxe de ponto.
- 9 Clique em OK.



#### Para inserir um caminho de destino manualmente:

Siga as etapas 1-4 acima e insira um caminho de destino absoluto ou relativo no painel Ações.

#### Para usar uma expressão como o caminho de destino:

1 Siga as etapas 1-4 acima.

2 Siga um destes procedimentos:

- Insira manualmente uma referência como um caminho de destino. Uma referência é avaliada para determinar o caminho de destino. Você pode usar uma referência como um parâmetro para a ação `with`. No exemplo a seguir, a variável `index` é avaliada e multiplicada por 2. O valor resultante é usado como o nome do clipe de filme na instância `Block` que é instruída a ser reproduzida:

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

- Na categoria Funções da lista Caixa de Ferramentas, escolha a ação `targetPath`.

A função `targetPath` converte uma referência a um clipe de filme em uma sequência de caracteres que pode ser usada por ações, como `tellTarget`.

No exemplo a seguir, a função `targetPath` converte a referência `Board.Block[index*2+1]` em uma sequência de caracteres:

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

O exemplo anterior equivale à seguinte sintaxe de barra:

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



- Na categoria Funções da lista Caixa de Ferramentas, escolha a função `eval`.

A função `eval` converte uma sequência de caracteres em uma referência a um clipe de filme que pode ser usada como um caminho de destino por ações, como `with`.

O script a seguir avalia a variável `i`, adiciona-a à sequência de caracteres `"cat"` e atribui o valor resultante à variável `x`. A variável `x` é agora uma referência a uma instância do clipe de filme e pode chamar os métodos do objeto `MovieClip`, conforme mostrado a seguir:

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

Você também pode usar a função `eval` para chamar os métodos diretamente, conforme mostrado a seguir:

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

## Usando ações e métodos para controlar Linhas de Tempo

Você pode usar certas ações e métodos do objeto `MovieClip` para *especificar como destino* ou executar tarefas em um clipe de filme ou nível carregado. Por exemplo, a ação `setProperty` define uma propriedade (como `_width`) de uma Linha de Tempo como um valor (como 100). Alguns métodos do `MovieClip` duplicam a função de todas as ações que especificam Linhas de Tempo como destino. Também há métodos adicionais, como `hitTest` e `swapDepths`. Quer você use uma ação ou um método, a Linha de Tempo de destino deverá estar carregada no Flash Player quando a ação ou o método for chamado.

As ações a seguir podem especificar clipes de filme como destino: `loadMovie`, `unloadMovie`, `setProperty`, `startDrag`, `duplicateMovieClip` e `removeMovieClip`. Para usar essas ações, você deve inserir um caminho de destino no parâmetro `Target` da ação para indicar o destinatário da ação. Algumas dessas ações podem especificar clipes de filme ou níveis como destino e outras, somente clipes de filme.

Os seguintes métodos do objeto `MovieClip` podem controlar clipes de filme ou níveis carregados e não possuem ações equivalentes: `attachMovie`, `getBounds`, `getBytesLoaded`, `getBytesTotal`, `globalToLocal`, `localToGlobal`, `hitTest` e `swapDepths`.

Quando uma ação e um método oferecem funções semelhantes, você pode controlar os clipes de filme usando qualquer um dos dois. A opção escolhida dependerá de sua preferência e da familiaridade com a criação de scripts no ActionScript.

Para obter mais informações sobre os métodos do objeto `MovieClip` e sobre cada ação, consulte o Capítulo 7, "Dicionário ActionScript", na página 173.



## Sobre métodos versus ações

Para usar um método, chame-o usando o caminho de destino para o nome da instância, seguido de um ponto e, depois, do nome do método e dos argumentos, como nos comandos a seguir:

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

No primeiro comando, o método `play` faz com que a instância `myMovieClip` seja reproduzida. No segundo comando, o método `gotoAndPlay` envia a reprodução em `childClip` (que é filho da instância `parentClip`) para o quadro 3 e é reproduzido.

As ações que controlam uma Linha de Tempo possuem um parâmetro `Target` que especifica o caminho de destino. Por exemplo, no script a seguir a ação `startDrag` especifica a instância `customCursor` como destino e faz com que ela possa ser arrastada:

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

Ao usar um método, chame-o no final do caminho de destino. Por exemplo, o comando a seguir executa a mesma função de `startDrag`:

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

Os comandos criados com os métodos do objeto `MovieClip` costumam ser mais curtos, pois não precisam da ação `tellTarget`. O uso da ação `tellTarget` não é recomendado, pois essa ação não é compatível com o padrão ECMA-262.

Por exemplo, para fazer com que o clipe de filme `myMovieClip` comece a ser reproduzido usando os métodos do objeto `MovieClip`, use o seguinte código:

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

O código a seguir produz os mesmos resultados usando a ação `tellTarget`:

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

## Usando vários métodos ou ações para especificar uma Linha de Tempo como destino

Você pode usar a ação `with` para endereçar um clipe de filme de destino uma vez e executar uma série de ações nesse clipe. A ação `with` funciona com todos os objetos do `ActionScript`, (por exemplo `Array`, `Color` e `Sound`), e não somente com clipes de filme. A ação `tellTarget` é semelhante à `with`. No entanto, a ação `tellTarget` não é recomendada, pois ela não funciona com todos os objetos do `ActionScript` e não é compatível com ECMA-262.

A ação `with` requer um objeto como parâmetro. O objeto especificado é adicionado ao final do caminho de destino atual. Todas as ações aninhadas em uma ação `with` são executadas no novo caminho de destino, ou *escopo*. Por exemplo, no script a seguir na Linha de Tempo principal, a ação `with` recebe o objeto `donut.hole` para alterar as propriedades de `hole`:

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

É como se os comandos contidos na ação `with` fossem chamados a partir da Linha de Tempo da instância `hole`.

No exemplo a seguir, observe a economia obtida com o uso da ação `with` e dos métodos do objeto `MovieClip` para emitir vários comandos:

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

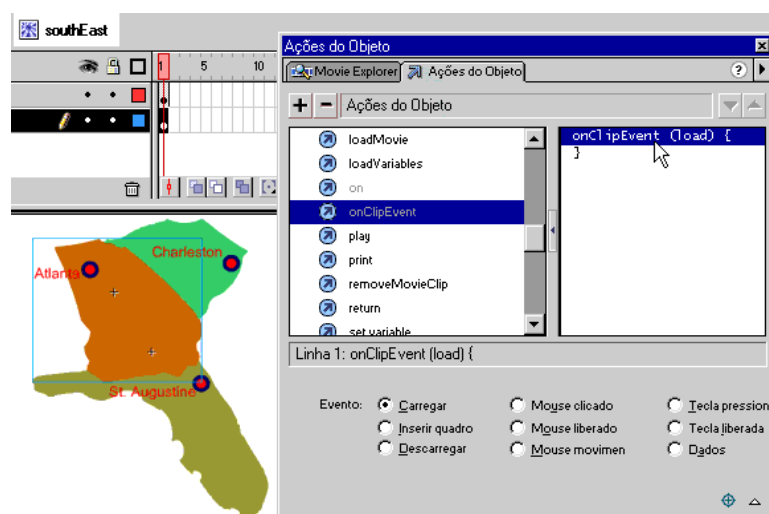
Para obter mais informações sobre a ação `tellTarget`, consulte *Usando o Flash*.



## Atribuindo uma ação ou um método

É possível atribuir ações e métodos a um botão ou a um quadro em uma Linha de Tempo ou a uma instância de um clipe de filme.

Para atribuir uma ação ou um método a uma instância de um clipe de filme, você deve usar um manipulador `onClipEvent`. Todas as ações anexadas à instância são aninhadas em um manipulador `onClipEvent` e executadas após ele ser disparado. A ação `onClipEvent` é disparada por eventos da Linha de Tempo (como o carregamento de um filme) ou do usuário (como um clique do mouse ou um pressionamento de tecla). Por exemplo, `onClipEvent(mouseMove)` dispara uma ação toda vez que o usuário move o mouse.



*A ação `onClipEvent` é atribuída a uma instância no Palco. Os eventos `onClipEvent` são listados no painel de parâmetros, no Painel de Ações.*

## Carregando e descarregando filmes adicionais

Você pode usar o método ou a ação `loadMovie` para reproduzir filmes adicionais sem fechar o Flash Player ou alternar entre filmes sem carregar outra página HTML (Hypertext Markup Language, linguagem de marcação de hipertexto). Também pode usar `loadMovie` para enviar variáveis para um script CGI (Common Gateway Interface, interface comum de gateway), que gera um arquivo SWF como sua saída CGI. Ao carregar um filme, você pode especificar um nível ou clipe de filme de destino no qual o filme será carregado.

O método e a ação `unloadMovie` removem um filme carregado anteriormente por `loadMovie`. Descarregar explicitamente filmes com `unloadMovie` assegura uma transição suave entre filmes e poderá aliviar a memória requerida pelo Flash player. Use a ação `loadMovie` para:

- Reproduzir uma sequência de faixas de propaganda que sejam arquivos SWF, colocando uma ação `loadMovie` no final de cada filme SWF para carregar o próximo filme
- Desenvolver uma interface ramificada que permita que o usuário escolha entre diferentes arquivos SWF
- Criar uma interface de navegação com controles de navegação no nível 0 que carreguem outros níveis. O carregamento de níveis produz transições mais suaves do que o carregamento de novas páginas HTML em um navegador.

## Alterando a posição e a aparência de um clipe de filme

Para alterar as propriedades de um clipe de filme à medida que ele é reproduzido, você pode usar a ação `setProperty` ou criar um comando que atribua um valor a uma propriedade. Se você carregar um filme em um destino, o filme carregado herdar as propriedades do clipe de filme de destino. Depois que o filme for carregado, você poderá alterar essas propriedades.

Algumas propriedades, chamadas *somente leitura*, possuem valores que podem ser lidos, porém não definidos. Você pode criar comandos para definir qualquer propriedade que não seja somente leitura. O comando a seguir define a propriedade `_alpha` da instância do clipe de filme `wheel` que é filho da instância `car`:

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

Além disso, você pode criar comandos que obtenham o valor de uma propriedade do clipe de filme. Por exemplo, o comando a seguir obtém o valor da propriedade `_xmouse` na Linha de Tempo principal e define a propriedade `_x` da instância `customCursor` como esse valor:

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

Você também pode usar a função `getProperty` para recuperar as propriedades de um clipe de filme.

As propriedades `_x`, `_y`, `_rotation`, `_xscale`, `_yscale`, `_height`, `_width`, `_alpha` e `_visible` são afetadas pelas transformações do pai do clipe de filme e transformam o clipe de filme e todos os seus filhos. As propriedades `_focusrect`, `_highquality`, `_quality` e `_soundbuftime` são globais; elas pertencem somente à Linha de Tempo do nível 0. Todas as outras propriedades pertencem aos cliques de filme ou níveis carregados. A tabela abaixo lista todas as propriedades do clipe de filme:

Propriedades			
<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>	

## Arrastando cliques de filme

Você pode usar o método ou a ação `startDrag` para fazer com que um clipe de filme possa ser arrastado durante a reprodução de um filme. Por exemplo, você pode criar um clipe de filme arrastável para jogos, funções do tipo arrastar e soltar, interfaces personalizáveis, barras de rolagem e controles deslizantes.

Um clipe de filme poderá ser arrastado até ser parado explicitamente por `stopDrag` ou até que outro clipe de filme seja especificado como destino com `startDrag`. É possível arrastar somente um clipe de filme de cada vez.

Para criar um comportamento arrastar e soltar mais complexo, você pode avaliar a propriedade `_droptarget` do clipe de filme que está sendo arrastado. Por exemplo, você pode examinar a propriedade `_droptarget` para verificar se o filme foi arrastado para um clipe de filme específico (como, por exemplo, um clipe de filme “trash can”) e, em seguida, disparar outra ação. Consulte “Usando comandos `if`”, na página 71 e “Usando operadores para manipular valores em expressões”, na página 62.

## Duplicando e removendo clipes de filme

Você pode criar ou remover instâncias de um clipe de filme à medida que o filme é reproduzido usando `duplicateMovieClip` ou `removeMovieClip`, respectivamente. O método e a ação `duplicateMovieClip` criam dinamicamente uma nova instância do clipe de filme, atribuindo um novo nome de instância e uma intensidade a ele. Um clipe de filme duplicado sempre inicia no quadro 1, mesmo que o clipe de filme original esteja em outro quadro quando duplicado, e está sempre acima de todos os clipes de filme predefinidos colocados na Linha de Tempo. As variáveis não são copiadas no clipe de filme duplicado.

Para excluir um clipe de filme criado com `duplicateMovieClip`, use `removeMovieClip`. Os clipes de filme duplicados também serão removidos se o clipe de filme pai for excluído.

## Anexando clipes de filme

Você pode recuperar uma cópia de um clipe de filme de uma biblioteca e reproduzi-la como parte de seu filme usando o método `attachMovie`. Esse método carrega outro clipe de filme no seu clipe e o reproduz à medida que o filme é executado.

Para usar o método `attachMovie`, é necessário atribuir um nome exclusivo ao clipe de filme que está sendo anexado na caixa de diálogo Propriedades de Vinculação de Símbolo.

### Para nomear um clipe de filme para compartilhamento:

- 1 Na Biblioteca de filmes, selecione o clipe de filme que você deseja anexar.
- 2 Na janela Biblioteca, escolha Vinculação no menu Opções.
- 3 Em Vinculação, escolha Exportar este Símbolo.
- 4 Na caixa de diálogo Propriedades de Vinculação de Símbolo, em Identificador, insira um nome para o clipe de filme. O nome deve ser diferente do nome do símbolo contido na biblioteca.
- 5 Clique em OK.

### Para anexar um clipe de filme a outro:

- 1 No painel Ações, especifique o destino ao qual você deseja anexar um clipe de filme.
- 2 Na lista Caixa de Ferramentas, selecione o objeto `MovieClip` e, em seguida, selecione o método `attachMovie`.



### 3 Defina os seguintes argumentos:

- Para `idName`, especifique o nome do Identificador inserido na caixa de diálogo Propriedades de Vinculação de Símbolo.
- Para `newName`, insira um nome de instância para o clipe anexado de modo que você possa especificá-lo como destino.
- Para `depth`, insira o nível no qual o filme duplicado será anexado ao clipe de filme. Cada filme anexado possui sua própria ordem de empilhamento, sendo que o nível 0 é o nível do filme de origem. Os clipes de filme anexados estão sempre sobre o clipe de filme original.

Por exemplo:

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

## Criando clipes inteligentes

Um Clipe Inteligente é um clipe de filme com parâmetros definidos que podem ser alterados. Esses parâmetros são então passados para ações no Clipe Inteligente que alteram o comportamento do clipe.

Para criar um Clipe Inteligente, atribua parâmetros a um símbolo de clipe de filme na Biblioteca. Você pode criar comandos do ActionScript no Clipe Inteligente que operem nos parâmetros do clipe, da mesma maneira que usa argumentos em uma definição de função. Você pode selecionar uma instância do Clipe Inteligente no Palco e alterar os valores dos parâmetros no painel Parâmetros do Clipe. Durante a reprodução, os valores definidos no painel são enviados para o Clipe Inteligente antes que quaisquer ações do filme sejam executadas.

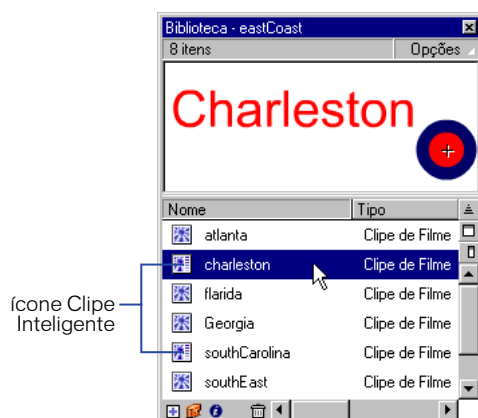
Os Clipes Inteligentes são úteis para passar elementos complicados do Flash de um programador para um designer. O programador pode criar ações no Clipe Inteligente com variáveis que controlam o clipe e o filme. O designer poderá então alterar os valores dessas variáveis no painel Parâmetros do Clipe sem precisar abrir o painel Ações.

Você pode usar Clipes Inteligentes para criar elementos de interface — como botões de opção, menus pop-up, dicas de ferramentas, pesquisas, jogos e avatares. Qualquer clipe de filme que você desejar reutilizar de maneira diferente sem alterar os scripts seria um bom Clipe Inteligente.

Além disso, você pode criar uma interface personalizada no Flash para o painel Parâmetros do Clipe a fim de ajudar os designers que estão personalizando o clipe.

## Definindo parâmetros do clipe

Os parâmetros do clipe são dados que são passados para um clipe de filme quando ele é carregado em um filme. Você pode definir os parâmetros do clipe ao criar páginas para o seu filme. Também pode usar esses parâmetros em ações para alterar a aparência e o comportamento do Clipe Inteligente durante a reprodução do filme. Um ícone especial na janela Biblioteca indica um clipe de filme com parâmetros de clipe definidos.



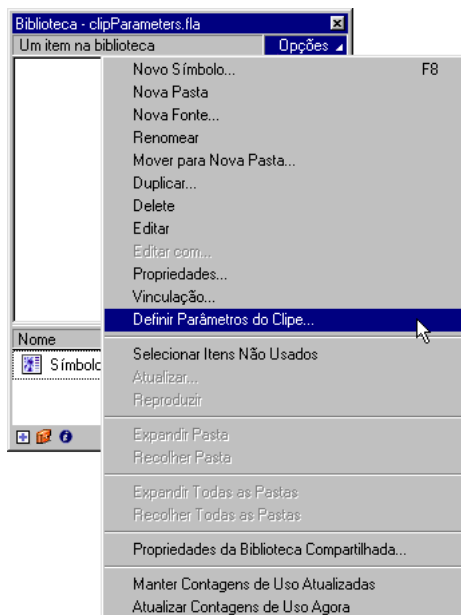
**Para definir parâmetros de clipe para um clipe de filme:**

- 1 Selecione um símbolo de clipe de filme em sua biblioteca de filmes e execute um destes procedimentos para exibir a caixa de diálogo Parâmetros do Clipe:
  - Clique com o botão direito do mouse (Windows) ou clique a tecla Control (Macintosh) e escolha Definir Parâmetros do Clipe no menu de contexto.





- Escolha Definir Parâmetros do Clipe no menu Opções no canto superior direito da janela Biblioteca.



## 2 Use os controles da caixa de diálogo Parâmetros do Clipe da seguinte maneira:

- Clique no botão com sinal de adição (+) para adicionar um novo par de nomes/valores ou parâmetros adicionais a um par de nomes/valores selecionado.
- Clique no botão com o sinal de subtração (-) para excluir um par de nomes/valores.
- Use os botões de direção para alterar a ordem dos parâmetros na lista.
- Selecione um campo clicando duas vezes nele e insira um valor.



- 3 Em Nome, insira um identificador exclusivo para o parâmetro.
- 4 Em Tipo, escolha o tipo dos dados que o parâmetro conterá no menu pop-up:
  - Selecione Default para usar uma sequência de caracteres ou um valor numérico
  - Selecione Array para obter uma lista dinâmica de itens que possa ser aumentada ou diminuída.
  - Selecione Object para declarar vários elementos relacionados a nomes e valores, como um objeto point com elementos  $x$  e  $y$ .
  - Selecione List para limitar a seleção a várias opções, como verdadeiro ou falso ou Vermelho, Verde ou Azul.
- 5 Em Valor, selecione o valor padrão que o parâmetro conterá no menu pop-up.
- 6 Se desejar usar uma interface personalizada para o painel Parâmetros do Clipe, siga um destes procedimentos:
  - Insira um caminho relativo para o arquivo SWF de interface personalizada no campo Link para interface personalizada.
  - Clique na pasta Link para interface personalizada e navegue para o arquivo SWF de interface personalizada.Consulte “Criando uma interface personalizada”, na página 136.
- 7 Em Descrição, insira observações que serão exibidas no painel Parâmetros do Clipe e que descrevam a função de cada parâmetro.

Você pode incluir na Descrição qualquer informação que o usuário do Clipe Inteligente deva saber. Por exemplo, uma explicação dos métodos definidos.
- 8 Escolha Bloquear na Instância para impedir que os usuários renomeiem os parâmetros no painel Parâmetros do Clipe.

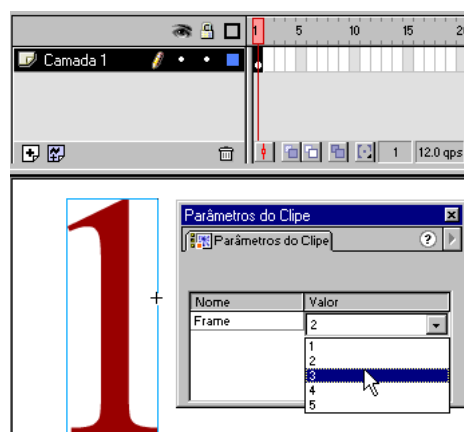
Recomenda-se deixar os nomes dos parâmetros bloqueados.
- 9 Clique em OK.

## Definindo parâmetros do clipe

Você pode criar ações no Clipe Inteligente que usem os parâmetros definidos para alterar o comportamento de um Clipe Inteligente. Em um exemplo simples, se você definir um parâmetro do clipe com o nome `Frame`, poderá escrever o seguinte script no Clipe Inteligente que usa esse parâmetro:

```
onClipEvent(load){
    gotoAndStop(Frame);
}
```

Em seguida, você poderá selecionar o Clipe Inteligente no Palco e definir o valor do parâmetro `Frame` no painel Parâmetros do Clipe para alterar o quadro que será reproduzido.



**Para definir os parâmetros de um Clipe Inteligente:**

- 1 Selecione uma instância do Clipe Inteligente no Palco.  
Os Clipes Inteligentes são clipes, portanto somente o primeiro quadro será exibido no modo de criação.
- 2 Escolha Janela > Painéis > Parâmetros do Clipe para exibir o painel Parâmetros do Clipe.



3 No painel Parâmetros do Clipe, siga um destes procedimentos:

- Clique duas vezes no campo Valor para selecioná-lo e insira um valor para cada parâmetro.

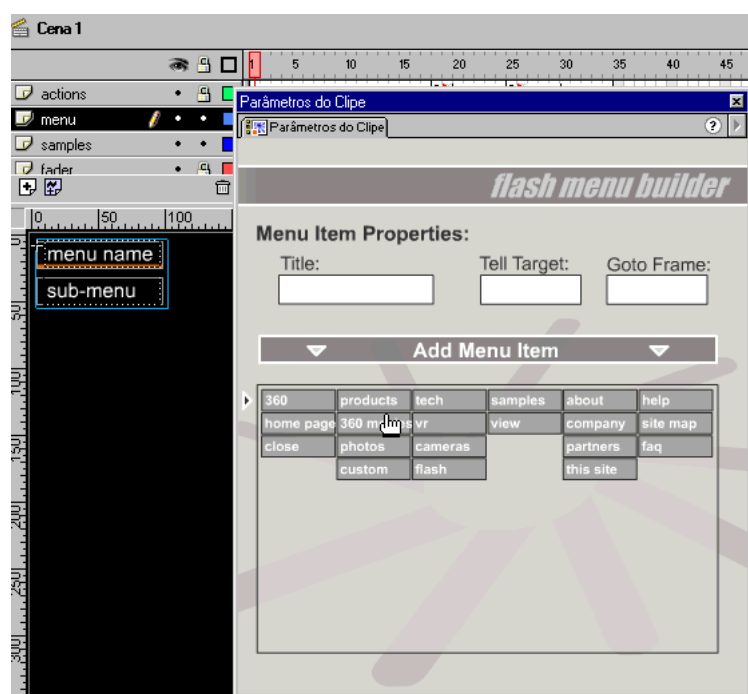
Se o parâmetro tiver sido definido como uma Lista, um menu pop-up será exibido.

- Se uma interface personalizada tiver sido definida, use os elementos de interface fornecidos.

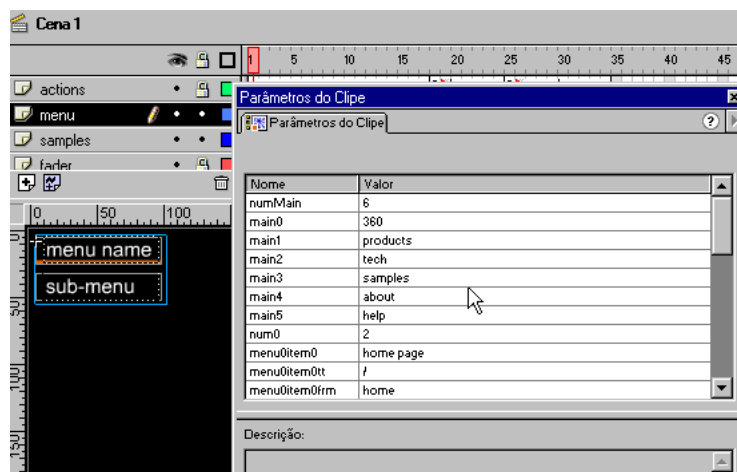
4 Escolha Controlar > Testar Filme para ver a alteração do comportamento do Clipe Inteligente.

## Criando uma interface personalizada

Uma interface personalizada é um filme do Flash que permite que você insira os valores a serem passados para o Clipe Inteligente. A interface personalizada substitui a interface do painel Parâmetros do Clipe.



Painel Parâmetros do Clipe com um filme de interface personalizada.



*O mesmo Clipe Inteligente sem uma interface personalizada no painel Parâmetros do Clipe.*

Todos os valores inseridos através de uma interface personalizada são passados do painel Parâmetros do Clipe para o Clipe Inteligente através de um clipe de filme intermediário ou de intercâmbio na interface personalizada. O clipe de filme de intercâmbio deve ter o nome de instância `xch`. Se uma interface personalizada for selecionada na caixa de diálogo Definir Parâmetros do Clipe, a instância do Clipe Inteligente passará os parâmetros definidos para o clipe de filme `xch` e todos os novos valores inseridos na interface personalizada serão copiados para `xch` e passados novamente para o Clipe Inteligente.

Você deve colocar o clipe `xch` na Linha de Tempo principal do filme de interface e `xch` deve ser sempre carregado. O clipe de filme `xch` deve conter somente os valores a serem passados para o Clipe Inteligente. Ele não deve conter gráficos, outros clipes de filme ou comandos do ActionScript; `xch` é simplesmente um recipiente através do qual valores são passados. Você pode transferir os objetos de nível superior, como Arrays e Objects, através do clipe `xch`. No entanto, você não deve passar Arrays ou Objects aninhados.

#### Para criar uma interface personalizada para um Clipe Inteligente:

- 1 Escolha Arquivo > Novo para criar um novo filme do Flash.
- 2 Escolha Inserir > Novo Símbolo para criar o clipe de filme de intercâmbio.
- 3 Crie uma nova camada chamada “Clipe de Intercâmbio”.
- 4 Com a camada “Clipe de Intercâmbio” selecionada, arraste o clipe de filme de intercâmbio da janela Biblioteca para o Palco no quadro 1.

- 5 Selecione o clipe de filme de intercâmbio no Palco, escolha Janela > Painéis > Instância e insira o nome xch.
- 6 Crie os elementos de interface com os quais o autor interagirá para definir os parâmetros do clipe. Por exemplo, um menu pop-up, botões de opção ou itens de menu do tipo arrastar e soltar.
- 7 Use a ação `set variable` para copiar valores de objetos e de variáveis para a instância xch.

Por exemplo, se um botão for usado como um elemento de interface, ele poderá ter uma ação que defina o valor da variável `vertical` e passe-o para xch, conforme mostrado a seguir:

```
on (release){  
    _root.xch.vertical = true;  
}
```

- 8 Exportar o filme como um arquivo SWF.

Para usar o SWF de interface personalizada com um Clipe Inteligente, é necessário vinculá-los na caixa de diálogo Definir Parâmetros do Clipe na Biblioteca que contém o Clipe Inteligente. Recomenda-se salvar o arquivo SWF na mesma pasta que o FLA que contém o Clipe Inteligente. Se você reutilizar o Clipe Inteligente em outro arquivo ou passá-lo para outro desenvolvedor, o Clipe Inteligente e o SWF de interface personalizada deverão permanecer nos mesmos locais relativos.



## CAPÍTULO 5

### Integrando o Flash com aplicativos da Web

.....

Os filmes do Flash podem enviar e carregar informações para/de arquivos remotos. Para enviar e carregar variáveis, use a ação `loadVariables` ou `getURL`. Para carregar um filme do Flash Player de um local remoto, use a ação `loadMovie`. Para enviar e carregar dados XML, use o objeto XML ou XMLSocket. Você pode estruturar os dados XML usando os métodos do objeto XML predefinido.

Também pode criar formulários do Flash consistindo em elementos de interface comuns, como campos de texto e menus pop-up, para obter os dados que serão enviados a um aplicativo do servidor.

Para estender o Flash para que ele possa enviar e receber mensagens do ambiente de host do filme — por exemplo, o Flash Player ou uma função JavaScript em um navegador da Web — você pode usar `fscommand` e os métodos do Flash Player.

## Enviando e carregando variáveis para/de um arquivo remoto

Um filme do Flash é uma janela para a captura e a exibição de informações, que se assemelha a uma página HTML (linguagem de marcação de hipertexto). Diferentemente das páginas HTML, os filmes do Flash podem permanecer carregados no navegador e ser atualizados continuamente com novas informações sem precisar de atualização. Você pode usar as ações e os métodos de objetos do Flash para enviar e receber informações de arquivos XML, arquivos de texto e scripts do servidor.

Os scripts do servidor podem solicitar informações específicas de um banco de dados e retransmiti-las entre o banco de dados e um filme do Flash. Os scripts do servidor podem ser escritos em diferentes linguagens: alguns dos mais comuns são Perl, ASP (Microsoft Active Server Pages, páginas ASP) e PHP.

O armazenamento das informações em um banco de dados e sua recuperação permite que você crie um conteúdo dinâmico e personalizado para o seu filme. Por exemplo, você poderia criar um quadro de mensagens, perfis pessoais para os usuários ou um carrinho de compras que lembre o que o usuário comprou para que possa determinar suas preferências.

Você pode usar várias ações e métodos de objetos do ActionScript para transferir informações para/de um filme. Cada ação e método usa um protocolo para transferir as informações, e também requer que elas sejam formatadas de determinada maneira.

As seguintes ações usam o protocolo HTTP ou HTTPS para enviar informações no formato codificado da URL: `getURL`, `loadVariables`, `loadMovie`.

Os métodos a seguir usam o protocolo HTTP ou HTTPS para enviar informações como XML: `XML.send`, `XML.load`, `XML.sendAndLoad`.

Os seguintes métodos criam e usam uma conexão de soquete TCP/IP para enviar informações como XML: `XMLSocket.connect`, `XMLSocket.send`.



## Sobre segurança

Ao reproduzir um filme do Flash em um navegador da Web, você pode carregar dados no filme somente a partir de um arquivo que esteja em um servidor no mesmo subdomínio. Isso impede que os filmes do Flash façam o download de informações a partir dos servidores de outras pessoas.

Para determinar o subdomínio de um URL que consiste em um ou dois componentes, use o domínio inteiro:

Domínio	Subdomínio
http://macromedia	macromedia
http://macromedia.com	macromedia.com

Para determinar o subdomínio de um URL que consiste em mais de dois componentes, remova o último nível:

Domínio	Subdomínio
http://x.y.macromedia.com	y.macromedia.com
http://www.macromedia.com	macromedia.com



O gráfico a seguir mostra como o Flash Player determina se deve ou não permitir uma solicitação HTTP:

**PALCO 1**

Esta solicitação é para: loadVariables, xml.load, xml.sendAndLoad ou xmlsocket.connect?

**PALCO 2**

Esta solicitação é para uma URL relativa?

**PALCO 3**

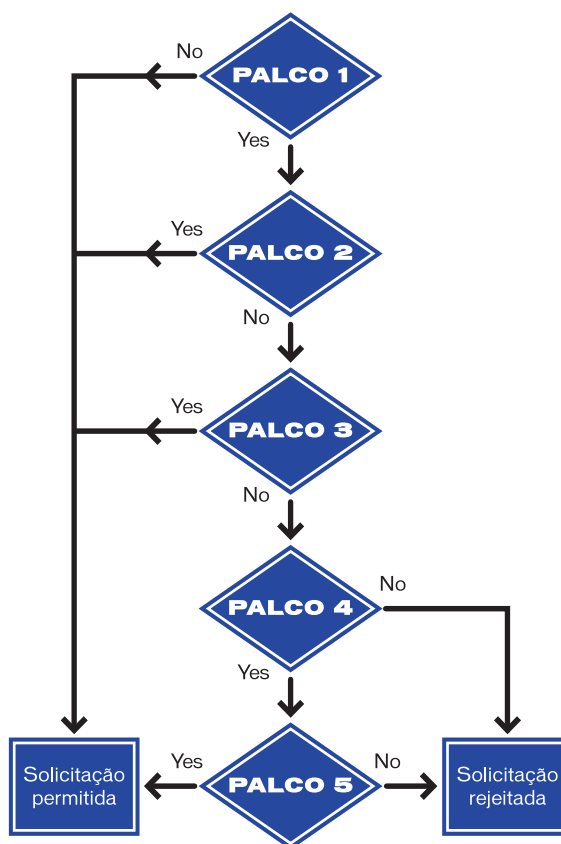
O filme que está fazendo a solicitação foi carregado de um disco local? (Sua URS começa com file: ou res:)

**PALCO 4**

A URL que está sendo solicitada começa com http://, https:// ou ftp://?

**PALCO 5**

O nome de domínio do filme solicitado corresponde ao nome de domínio da URL solicitada?



Ao usar o objeto XMLSocket para criar uma conexão de soquete com um servidor, você deve usar uma porta de número 1024 ou superior. (As portas com números inferiores são normalmente usadas para Telnet, FTP, a World Wide Web ou Finger).

O Flash usa os recursos de segurança HTTP e HTTPS e de navegadores padrão. Basicamente, ele oferece a mesma segurança disponível no HTML padrão. Você deve seguir as mesmas regras usadas ao criar sites da Web seguros em HTML. Por exemplo, para dar suporte a senhas protegidas no Flash, é necessário estabelecer a autenticação de senha com uma solicitação a um servidor Web.

Para criar uma senha, use um campo de texto para solicitar uma senha do usuário. Submeta-a a um servidor em uma ação `loadVariables` ou em um método `XML.sendAndLoad` usando uma URL HTTPS com o método POST. O servidor Web poderá então verificar se a senha é válida. Dessa maneira, a senha nunca estará disponível no arquivo SWF.

## Verificando os dados carregados

Cada ação e método que carrega dados em um filme (exceto `XMLSocket.send`) é *assíncrona*; os resultados da ação são retornados em um momento indeterminado.

Antes de usar os dados carregados em um filme, você deve verificar se eles foram carregados. Por exemplo, você não pode carregar variáveis e manipular os valores dessas variáveis no mesmo script. No script a seguir, você só poderá usar a variável `lastFrameVisited` após certificar-se de que a variável tenha sido carregada a partir do arquivo `myData.txt`:

```
loadVariables("myData.txt", 0);  
gotoAndPlay(lastFrameVisited);
```

Cada ação e método possui uma técnica específica que pode ser usada para verificar os dados que foram carregados. Se você usar a ação `loadVariables` ou `loadMovie`, poderá carregar informações no destino de um clipe de filme e usar o evento `data` da ação `onClipEvent` para executar um script. Se você usar a ação `loadVariables` para carregar os dados, a ação `onClipEvent(data)` será executada quando a última variável for carregada. Se você usar a ação `loadMovie` para carregar os dados, a ação `onClipEvent(data)` será executada toda vez que um fragmento do filme for enviado ao Flash Player.

Por exemplo, a ação do botão a seguir carrega as variáveis do arquivo `myData.txt` no clipe de filme `loadTargetMC`:

```
on(release){  
    loadVariables("myData.txt", _root.loadTargetMC);  
}
```

Uma ação atribuída à instância `loadTargetMC` usa a variável `lastFrameVisited` que é carregada a partir do arquivo `myData.txt`. A ação a seguir só será executada depois que todas as variáveis, incluindo `lastFrameVisited`, forem carregadas:

```
onClipEvent(data) {  
    gotoAndPlay(lastFrameVisited);  
}
```

Se você usar os métodos `XML.load` e `XMLSocket.connect`, poderá definir um manipulador que processará os dados quando eles chegarem. Um manipulador é uma propriedade do objeto XML ou XMLSocket à qual você atribui uma função definida. Os manipuladores são chamados automaticamente quando as informações são recebidas. Para o objeto XML, use `XML.onLoad`. Para o objeto XMLSocket, use `XMLSocket.onConnect`.

Para obter mais informações, consulte “Usando o objeto XML”, na página 145 e “Usando o objeto XMLSocket”, na página 149.

## Usando loadVariables, getURL e loadMovie

As ações `loadVariables`, `getURL` e `loadMovie` comunicam-se com os scripts do servidor usando o protocolo HTTP. Cada ação envia todas as variáveis da Linha de Tempo à qual ela está anexada e manipula sua resposta da seguinte maneira:

- `getURL` retorna todas as informações para uma janela do navegador, e não para o Flash Player.
- `loadVariables` carrega as variáveis em uma Linha de Tempo especificada no Flash Player.
- `loadMovie` carrega um filme em um nível especificado no Flash Player.

Ao usar a ação `loadVariables`, `getURL` ou `loadMovie`, você pode especificar vários argumentos:

- *URL* é o arquivo onde residem as variáveis remotas.
- *Local* é o nível ou o destino no filme que recebe as variáveis.

Para obter mais informações sobre níveis e destinos, consulte “Sobre várias Linhas de Tempo”, na página 110.

**Observação:** A ação `getURL` não requer esse argumento.

- *Variáveis* define o método HTTP, GET ou POST, através do qual as variáveis serão enviadas.

Por exemplo, se desejasse controlar as pontuações mais altas de um jogo, você poderia armazená-las em um servidor e usar uma ação `loadVariables` para carregá-las no filme toda vez que alguém jogasse esse jogo. A ação seria semelhante a esta:

```
loadVariables("http://www.mySite.com/scripts/high_score.php",  
_root.scoreClip, GET)
```

Essa ação carrega as variáveis do script PHP chamado `high_score.php` na instância do clipe de filme `scoreClip` usando o método HTTP GET.

Todas as variáveis carregadas com a ação `loadVariables` devem estar no formato de aplicativo MIME/x-www-urlformencoded padrão (um formato padrão usado pelos scripts CGI). O arquivo especificado no argumento URL da ação `loadVariables` deve criar os pares de valores e de variáveis nesse formato para que o Flash possa lê-los.

O arquivo pode especificar qualquer número de variáveis; os pares de valores e de variáveis devem estar separados com um E comercial (&) e as palavras dentro de um valor devem estar separadas com um sinal de mais (+). Por exemplo, esta frase define diversas variáveis:

```
highScore1=54000&playerName1=rockin+good&highScore2=53455&playerName2=bonehelmet&highScore3=42885&playerName3=soda+pop
```

Para obter mais informações sobre `loadVariables`, `getURL` e `loadMovie`, consulte suas entradas no Capítulo 7, “Dicionário do ActionScript”.

## Sobre XML

O XML (*Extensible Markup Language*) está se tornando o padrão para o intercâmbio de dados estruturados em aplicativos de Internet. Você pode integrar dados no Flash com servidores que usam a tecnologia XML para criar aplicativos sofisticados, como um sistema de bate-papo ou de corretagem.

No XML, assim como no HTML, você pode usar marcas para *marcar* ou especificar o corpo do texto. No HTML, você pode usar marcas predefinidas para indicar como o texto deve aparecer em um navegador da Web (por exemplo, a marca `<b>` indica que esse texto deve estar em negrito). No XML, você define as marcas que identificam o tipo de um dado (por exemplo, `<password>VerySecret</password>`). O XML separa a estrutura das informações da maneira como ela é exibida. Isso permite que o mesmo documento XML seja usado e reutilizado em ambientes diferentes.

Toda marca XML é chamada de *nó* ou elemento. Cada nó possui um tipo (elemento XML 1 ou nó de texto 3), e os elementos também podem ter atributos. Um nó aninhado em outro nó é denominado *filho* ou *childNodes*. Essa estrutura de árvore hierárquica de nós é denominada DOM XML — de modo semelhante ao DOM JavaScript, que é a estrutura de elementos de um navegador da Web.

No exemplo a seguir, `<PORTFOLIO>` é o nó pai; ele não possui nenhum atributo e contém o *childNodes* `<HOLDING>` que possui os atributos `SYMBOL`, `QTY`, `PRICE` e `VALUE`:

```
<PORTFOLIO>
  <HOLDING SYMBOL="RICH"
    QTY="75"
    PRICE="245.50"
    VALUE="18412.50" />
</PORTFOLIO>
```

## Usando o objeto XML

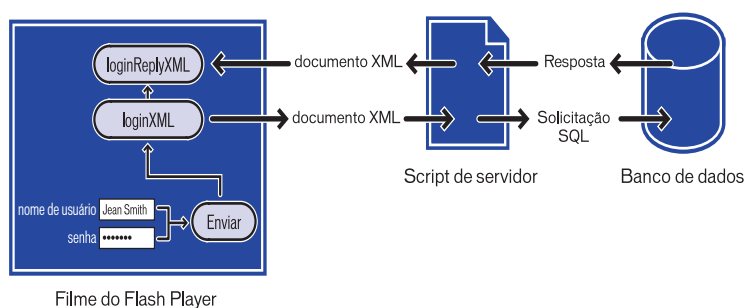
Você pode usar os métodos do objeto ActionScript XML (por exemplo, `appendChild`, `removeNode` e `insertBefore`) para estruturar os dados XML no Flash para enviá-los a um servidor e para manipular e interpretar os dados XML descarregados.

Você pode usar os seguintes métodos do objeto XML para enviar e carregar dados XML em um servidor através do método `POST HTTP`:

- `load` faz o download do XML a partir de um URL e coloca-o em um objeto ActionScript XML.
- `send` envia um objeto XML para um URL. Todas as informações retornadas são enviadas para outra janela do navegador.
- `sendAndLoad` envia um objeto XML para um URL. Todas as informações retornadas são colocadas em um objeto ActionScript XML.

Por exemplo, você poderia criar um sistema de corretagem para a venda de títulos que armazenasse todas as suas informações (nomes de usuários, senhas, identificações de sessões, títulos em carteira e informações sobre transações) em um banco de dados.

O script do servidor que transfere as informações entre o Flash e o banco de dados lê e grava os dados no formato XML. Você pode usar o ActionScript para converter as informações obtidas no filme do Flash (por exemplo, um nome de usuário e uma senha) em um objeto XML e enviar os dados para script do servidor como um documento XML. Você também pode usar o ActionScript para carregar o documento XML retornado pelo servidor em um objeto XML para ser usado no filme



*O fluxo e a conversão de dados entre um filme do Flash Player, um documento de script do servidor e um banco de dados.*

A validação da senha para o sistema de corretagem requer dois scripts: uma função definida no quadro um e um script que cria e envia os objetos XML anexados ao botão SUBMIT no formulário.

Quando os usuários inserem suas informações nos campos de texto do filme do Flash com as variáveis `username` e `password`, as variáveis devem ser convertidas em XML antes de serem enviadas ao servidor. A primeira seção do script carrega as variáveis em um objeto XML recentemente criado chamado `loginXML`. Quando o usuário pressiona o botão SUBMIT, o objeto `loginXML` é convertido em uma seqüência de caracteres do XML e enviado para o servidor.

O script a seguir está anexado ao botão SUBMIT. Para compreender o script, leia as linhas de comentário de cada script conforme indicado pelos caracteres `//`:

```
on (release) {
    // A. Cria um documento XML com um elemento LOGIN
    loginXML = new XML();
    loginElement = loginXML.createElement("LOGIN");
    loginElement.attributes.username = username;
    loginElement.attributes.password = password;
    loginXML.appendChild(loginElement);

    // B. Cria um objeto XML para armazenar a resposta do servidor
    loginReplyXML = new XML();
    loginReplyXML.onLoad = onLoginReply;

    // C. Envia o elemento LOGIN para o servidor,
    //     coloca a resposta em loginReplyXML
    loginXML.sendAndLoad("https://www.imexstocks.com/main.cgi",
        loginReplyXML);
}
```

A primeira seção do script gera o seguinte XML quando o usuário pressiona o botão SUBMIT:

```
<LOGIN USERNAME="JeanSmith" PASSWORD="VerySecret" />
```

O servidor recebe o XML, gera uma resposta XML e envia-a novamente ao filme do Flash. Se a senha for aceita, o servidor responderá da seguinte maneira:

```
<LOGINREPLY STATUS="OK" SESSION="rnr6f7vkj2oe14m7jkkycilb" />
```

Esse XML inclui um atributo `SESSION` que contém uma identificação de sessão exclusiva, gerada aleatoriamente, que será usada em todas as comunicações entre o cliente e o servidor durante todo o resto da sessão. Se a senha for rejeitada, o servidor responderá com a seguinte mensagem:

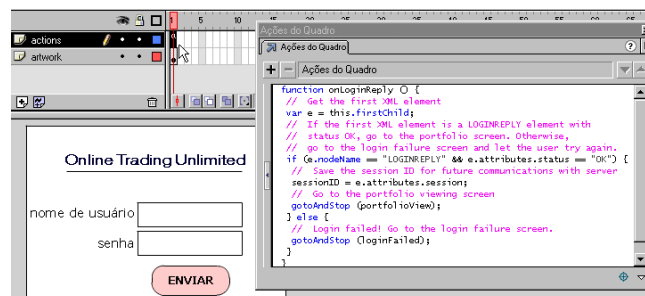
```
<LOGINREPLY STATUS="FAILED" />
```

O nó XML `LOGINREPLY` deve ser carregado em um objeto XML em branco no filme do Flash. O comando a seguir cria o objeto XML `loginReplyXML` para receber o nó XML:

```
// B. Cria um objeto XML para armazenar a resposta do servidor
loginReplyXML = new XML();
loginReplyXML.onLoad = onLoginReply;
```

O segundo comando atribui a função `onLoginReply` ao manipulador `loginReplyXML.onLoad`.

O elemento XML LOGINREPLY chega de modo assíncrono, de maneira semelhante aos dados de uma ação loadVariables e é carregado no objeto loginReplyXML. Quando os dados chegam, o método onLoad do objeto loginReplyXML é chamado. Você deve definir a função onLoginReply e atribuí-la ao manipulador loginReplyXML.onLoad para que processe o elemento LOGINREPLY. A função onLoginReply é atribuída ao quadro que contém o botão SUBMIT.



*A função onLoginReply é definida no primeiro quadro do filme.*

A função onLoginReply é definida no primeiro quadro do filme. Para compreender o script, leia as linhas de comentário de cada script conforme indicado pelos caracteres //:

```
function onLoginReply() {
    // Obtém o primeiro elemento XML
    var e = this.firstChild;
    // Se o primeiro elemento XML for um elemento LOGINREPLY com
    // status OK, vai para a tela do portfólio. Caso contrário,
    // vai para a tela de falha de login e permite que o usuário /
    // tente novamente.
    if (e.nodeName == "LOGINREPLY" && e.attributes.status == "OK") {
        // Salva a identificação de sessão para comunicação futura com o
        // servidor
        sessionId = e.attributes.session;
        // Vai para a tela de exibição do portfólio
        gotoAndStop("portfolioView");
    } else {
        // Falha no login! Vai para a tela de falha de login.
        gotoAndStop("loginFailed");
    }
}
```





A primeira linha dessa função, `var e = this.firstChild`, usa a palavra-chave `this` para se referir ao objeto XML `loginReplyXML` que acabou de ser carregado com o XML do servidor. Você pode usar `this` porque `onLoginReply` foi chamada como `loginReplyXML.onLoad`; portanto, embora `onLoginReply` seja aparentemente uma função simples, ela se comporta realmente como um método de `loginReplyXML`.

Para enviar o nome do usuário e a senha como XML para o servidor e para carregar uma resposta XML novamente no filme do Flash, você pode usar o método `sendAndLoad`, conforme mostrado a seguir:

```
// C. Envia o elemento LOGIN para o servidor,  
//      coloca a resposta em loginReplyXML  
loginXML.sendAndLoad("https://www.imexstocks.com/main.cgi",  
loginReplyXML);
```

Para obter mais informações sobre os métodos XML, consulte suas entradas no Capítulo 7, “Dicionário do ActionScript”.

**Observação:** Este design é apenas um exemplo, e não há qualquer garantia quanto ao nível de segurança que ele oferece. Se você estiver implementando um sistema protegido por senha seguro, certifique-se de que tenha um bom conhecimento de segurança de rede.

## Usando o objeto XMLSocket

O ActionScript fornece um objeto `XMLSocket` predefinido que permite o estabelecimento de uma conexão contínua com um servidor. Uma conexão de soquete permite que o servidor envie informações ao cliente assim que elas estiverem disponíveis. Sem uma conexão contínua, o servidor deve aguardar uma solicitação HTTP. Essa conexão aberta remove as questões de latência e é normalmente usada para aplicativos em tempo real, como aplicativos de bate-papo. Os dados são enviados através da conexão de soquete como uma sequência de caracteres e devem estar no formato XML. Você pode usar o objeto XML para estruturar os dados.

Para criar uma conexão de soquete, é necessário criar um aplicativo do servidor para aguardar a solicitação da conexão de soquete e enviar uma resposta ao filme do Flash. Esse tipo de aplicativo do servidor pode ser criado em uma linguagem de programação, como Java.

Você pode usar os métodos `connect` e `send` do objeto ActionScript `XMLSocket` para transferir XML para/de um servidor através de uma conexão de soquete. O método `connect` estabelece uma conexão de soquete com uma porta do servidor Web. O método `send` envia um objeto XML para o servidor especificado na conexão de soquete.



Quando você chama o método `connect` do objeto `XMLSocket`, o Flash Player abre uma conexão TCP/IP com o servidor e mantém a conexão aberta até que ocorra uma das seguintes situações:

- O método `close` do objeto `XMLSocket` é chamado.
- Não existe mais nenhuma referência ao objeto `XMLSocket`.
- O Flash Player é encerrado.
- A conexão é interrompida (por exemplo, o modem é desconectado).

O exemplo a seguir cria uma conexão de soquete XML e envia os dados do objeto XML `myXML`. Para compreender o script, leia as linhas de comentário de cada script conforme indicado pelos caracteres `//`:

```
//cria um novo objeto XMLSocket
sock = new XMLSocket();
//chama o seu método de conexão para estabelecer uma conexão com
//a porta 1024 do servidor no URL sock.connect
//("http://www.myserver.com", 1024);
//define uma função para atribuir o objeto de soquete que trata da
//resposta do servidor. Se a conexão tiver êxito, envia o //objeto
//myXML. Se ela falhar, fornece uma mensagem de erro em um campo
//de texto.
function onSockConnect(success){
    if (success){
        sock.send(myXML);
    } else {
        msg="There has been an error connecting to"+serverName;
    }
}
//atribui a função onSockConnect à propriedade onConnect
sock.onConnect = onSockConnect;
```

Para obter mais informações, consulte a entrada referente a `XMLSocket` no Capítulo 7, “Dicionário do ActionScript”.

## Criando formulários

Os formulários do Flash fornecem um tipo avançado de interatividade — uma combinação de botões, filmes e campos de texto que permitem o envio de informações para outro aplicativo em um servidor local ou remoto. Todos os elementos de formulário comuns (como botões de opção, listas drop-down e caixas de seleção) podem ser criados como filmes ou botões com a aparência do design global de seu site da Web. O elemento de formulário mais comum é um campo de entrada de texto.

Os tipos comuns de formulários que usam esses elementos de interface compreendem interfaces de bate-papo, formulários de pedidos e interfaces de pesquisa. Por exemplo, um formulário do Flash pode obter informações sobre endereço e enviá-las a outro aplicativo que compila as informações em uma mensagem de correio eletrônico ou em um arquivo de banco de dados. Até mesmo um campo de texto único é considerado um formulário e pode ser usado para obter dados do usuário e exibir os resultados.

Os formulários requerem dois componentes principais: os elementos da interface do Flash que compõem o formulário e um aplicativo do servidor ou um script do cliente para processar as informações inseridas pelo usuário. As etapas a seguir descrevem o procedimento geral para a criação de um formulário no Flash.

### Para criar um formulário:

- 1 Coloque os elementos de interface no filme usando o layout desejado.  
Você pode usar os elementos de interface da biblioteca comum Botões – Avançado ou criar os seus próprios elementos.
- 2 No painel Opções de Texto, defina os campos de texto como Entrada e atribua um nome de variável exclusivo a cada um.  
  
Para obter mais informações sobre como criar campos de texto editáveis, consulte *Usando o Flash*.
- 3 Atribua uma ação que envie, carregue ou envie e carregue os dados.

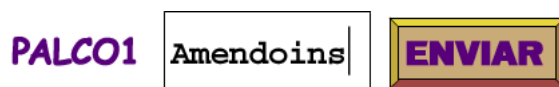
## Criando um formulário de pesquisa

Um exemplo de formulário simples é um campo de pesquisa com um botão Submit. Como introdução à criação de formulários, o exemplo a seguir fornece instruções para a criação de uma interface de pesquisa usando a ação `getURL`. Inserindo as informações necessárias, os usuários podem enviar uma palavra-chave para um mecanismo de pesquisa em um servidor Web remoto.

Para criar um formulário de pesquisa simples:

- 1 Crie um botão para submeter os dados inseridos.
- 2 Crie um rótulo, um campo de texto em branco e uma instância do botão no Palco.

Sua tela deve ter uma aparência semelhante a esta:



- 3 Selecione o campo de texto e escolha Janela > Painéis > Opções de Texto.
- 4 No painel Opções de Texto, defina as seguintes opções:
  - Escolha Texto de Entrada no menu pop-up.
  - Selecione Borda/seg. plano
  - Especifique um nome de variável.

**Observação:** É possível que cada mecanismo de pesquisa necessite de um nome de variável específico. Vá para o site da Web do mecanismo de pesquisa para obter detalhes.

- 5 No Palco, selecione o botão e escolha Janela > Ações.

O painel Ações do Objeto é exibido.

**Observação:** Uma marca de seleção ao lado de Ações no menu Janela indica que o painel está aberto.

- 6 Arraste a ação `getURL` da caixa de ferramentas para a janela Script.

7 No painel Parâmetros, defina as seguintes opções:

- Para URL, insira a URL do mecanismo de pesquisa.
- Para Janela, selecione `_blank`. Isso abrirá uma nova janela que exibe os resultados da pesquisa.
- Para Variáveis, selecione Send Using GET.

8 Para testar o formulário, escolha Arquivo > Visualizar Publicação > HTML.

## Usando variáveis em formulários

Você pode usar variáveis em um formulário para armazenar a entrada do usuário. Para definir as variáveis, use campos de texto editáveis ou atribua ações aos botões nos elementos da interface. Por exemplo, cada item de um menu pop-up é um botão com uma ação que define uma variável para indicar o item selecionado. Você pode atribuir um nome de variável a um campo de entrada de texto. O campo de texto funciona como uma janela que exibe o valor dessa variável.

Quando você envia informações para/de um script do servidor, as variáveis do filme do Flash devem corresponder às variáveis do script. Por exemplo, se o script esperar uma variável chamada `senha`, o campo de texto em que os usuários inserem a senha deverá ter o nome de variável `senha`.

Alguns scripts necessitam de variáveis ocultas, que são variáveis que o usuário nunca vê. Para criar uma variável oculta no Flash, você pode definir uma variável em um quadro do clipe de filme que contenha os outros elementos do formulário. As variáveis ocultas são enviadas para o script do servidor junto com todas as outras variáveis definidas na Linha de Tempo que contém a ação que submete o formulário.

## Verificando os dados inseridos

Para um formulário que envia variáveis para um aplicativo em um servidor Web, você desejará verificar se os usuários estão inserindo as informações adequadas. Por exemplo, você não deseja que os usuários insiram texto em um campo de telefone. Use várias ações `set variable` junto com `for` e `if` para avaliar os dados inseridos.

A ação de exemplo a seguir verifica se os dados inseridos são um número e se o número está no formato `###-###-####`. Se os dados forem válidos, a mensagem “Este é um número de telefone válido!” será exibida. Se os dados não forem válidos, a mensagem “Este é um número de telefone inválido!” será exibida.



Para usar esse script em um filme, crie dois campos de texto no Palco e escolha Entrada no painel Opções de Texto para cada campo. Atribua a variável `phoneNumber` a um campo de texto e a variável `message` ao outro. Anexe a seguinte ação a um botão no Palco, ao lado dos campos de texto:

```
on (release) {
    valid = validPhoneNumber(phoneNumber);
    if (valid) {
        message = "Este é um número de telefone inválido!";
    } else {
        message = "Este é um número de telefone inválido!";
    }
    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;
    }
}
```

Para enviar os dados, crie um botão que tenha uma ação semelhante à seguinte. (Substitua os argumentos de `getURL` pelos argumentos adequados ao seu filme.)

```
on (release) {
    if (valid) {
        getURL("http://www.websserver.com", "_self", "GET");
    }
}
```

Para obter mais informações sobre essas instruções do ActionScript, consulte `set`, `for` e `if` no Capítulo 7, “Dicionário ActionScript”, na página 173.

## Enviando mensagens para/do Flash Player

Para enviar mensagens de um filme do Flash para o seu ambiente de host (por exemplo, um navegador da Web, um filme do Director ou o Flash Player independente), você pode usar a ação `fscommand`. Isso permite que você estenda o seu filme usando os recursos do host. Por exemplo, você poderia enviar uma ação `fscommand` para uma função JavaScript em uma página HTML que abre uma nova janela do navegador com propriedades específicas.

Para controlar um filme do Flash Player a partir de linguagens de script do navegador da Web, como JavaScript, VBScript e Microsoft JScript, você pode usar os métodos do Flash Player — funções que enviam mensagens de um ambiente de host para o filme do Flash. Por exemplo, você poderia ter um link em uma página HTML que envie o filme do Flash para um quadro específico.

### Usando `fscommand`

Use a ação `fscommand` para enviar uma mensagem para qualquer programa que esteja hospedando o Flash Player. A ação `fscommand` contém dois parâmetros: *comando* e *argumentos*. Para enviar uma mensagem para a versão independente do Flash Player, você deve usar comandos e argumentos predefinidos.

Por exemplo, a ação a seguir define o exibidor independente para escalar o filme para o tamanho de tela cheia quando o usuário solta o botão:

```
on(release){  
    fscommand("fullscreen", "true");  
}
```



A tabela a seguir mostra os valores que podem ser especificados para os parâmetros *comando* e *argumentos* da ação *fscommand* para controlar um filme reproduzido no exibidor independente (incluindo projetores):

<i>comando</i>	<i>argumentos</i>	<b>Objetivo</b>
quit	Nenhum	Fecha o projetor.
fullscreen	true ou false	A especificação de true define o Flash Player no modo de tela cheia. A especificação de false retorna o exibidor para a exibição normal de menu.
allowscale	true ou false	A especificação de false define o exibidor para que o filme seja sempre desenhado em seu tamanho original e nunca escalado. A especificação de true força o filme a ser escalado para 100% do exibidor.
showmenu	true ou false	A especificação de true ativa o conjunto completo de itens do menu de contexto. A especificação de false torna esmaecidos todos os itens do menu de contexto, exceto Sobre o Flash Player.
exec	Caminho para o aplicativo	Executa um aplicativo no projetor.

Para usar *fscommand* para enviar uma mensagem para uma linguagem de script como JavaScript em um navegador da Web, você pode passar dois argumentos quaisquer nos parâmetros *comando* e *argumentos*. Esses argumentos podem ser seqüências de caracteres ou expressões e serão usados em uma função JavaScript que “captura” ou manipula a ação *fscommand*.

Uma ação *fscommand* chama a função JavaScript *movienam\_DoFSCCommand* na página HTML que incorpora o filme do Flash, onde *movienam* é o nome do Flash Player atribuído pelo atributo NAME da marca EMBED ou pelo atributo ID da marca OBJECT. Se o nome atribuído ao Flash Player for *myMovie*, a função JavaScript chamada será *myMovie\_DoFSCCommand*.





Para usar a ação `fscommand` para abrir uma caixa de mensagem em um filme do Flash na página HTML através de JavaScript:

- 1 Na página HTML que incorpora o filme do Flash, adicione o seguinte código JavaScript:

```
function theMovie_DoFSCommand(command, args) {
    if (command == "messagebox") {
        alert(args);
    }
}
```

Se você publicar o seu filme usando o Flash com o modelo `FSCommand` nas Configurações de Publicação HTML, esse código será inserido automaticamente. Os atributos `NAME` e `ID` do filme serão o nome do arquivo. Por exemplo, para o arquivo `myMovie.fl`, os atributos seriam definidos como `myMovie`. Para obter mais informações sobre publicação, consulte *Usando o Flash*.

- 2 No filme do Flash, adicione a ação `fscommand` a um botão:

```
fscommand("messagebox", "This is a message box invoked from  
within Flash.")
```

Você também pode usar expressões para a ação `fscommand` e argumentos, como no exemplo a seguir:

```
fscommand("messagebox", "Hello, " & name & ", welcome to our  
Web site!")
```

- 3 Escolha Arquivo > Visualizar Publicação > HTML para testar o filme.

A ação `fscommand` pode enviar mensagens ao Macromedia Director que são interpretadas pelo Lingo como seqüências de caracteres, eventos ou código Lingo executável. Se a mensagem for uma seqüência de caracteres ou um evento, você deverá criar o código Lingo para recebê-la a partir da ação `fscommand` e executar uma ação no Director. Para obter mais informações, consulte o Director Support Center em <http://www.macromedia.com/support/director>.

No Visual Basic, Visual C++ e em outros programas que podem hospedar controles ActiveX, `fscommand` envia um evento VB com duas seqüências de caracteres que podem ser manipuladas na linguagem de programação do ambiente. Para obter mais informações, use as palavras-chave método do Flash para pesquisar o Centro de Suporte Flash em <http://www.macromedia.com/support/flash>.

## Sobre os métodos do Flash Player

Você pode usar os métodos do Flash Player para controlar um filme no Flash Player a partir das linguagens de script do navegador da Web, como JavaScript e VBScript. Assim como outros métodos, você pode usar os métodos do Flash Player para enviar chamadas aos filmes do Flash Player a partir de um ambiente de script diferente do ActionScript. Cada método possui um nome, e a maioria dos métodos contém argumentos. Os argumentos especificam um valor sobre o qual o método opera. O cálculo executado por alguns métodos retorna um valor que pode ser usado pelo ambiente de script.

Duas tecnologias diferentes permitem a comunicação entre o navegador e o Flash Player: LiveConnect (Netscape Navigator 3.0 ou posterior no Windows 95/98/2000/NT ou Power Macintosh) e ActiveX (Microsoft Internet Explorer 3.0 e posterior no Windows 95/98/2000/NT). Embora as técnicas de script sejam semelhantes para todos os navegadores e linguagens, há propriedades e eventos adicionais disponíveis para uso com os controles ActiveX.

Para obter mais informações, incluindo uma lista completa dos métodos de script do Flash Player, use as palavras-chave `método do Flash` para pesquisar o Centro de Suporte Flash em <http://www.macromedia.com/support/flash>.



## CAPÍTULO 6

### Solucionando problemas do ActionScript

.....

O nível de sofisticação de algumas ações, especialmente quando combinadas umas com as outras, pode tornar os filmes do Flash complexos. Como em qualquer linguagem de programação, você pode escrever um ActionScript incorreto, o que causa erros nos scripts. O uso de técnicas de criação corretas facilita a solução de problemas com o filme quando algo inesperado ocorrer.

O Flash tem diversas ferramentas que ajudam a testar filmes em modo de teste ou em um navegador. O Depurador mostra uma lista hierárquica de clipes de filme carregados no momento no Flash Player. Ele também permite exibir e modificar valores enquanto o filme é reproduzido. No modo de teste de filme, a janela Saída exibe mensagens de erro e listas de variáveis e objetos. Também é possível usar a ação `trace` nos scripts para enviar observações de programação e valores de expressões para a janela Saída.

## Diretrizes para criação de páginas e solução de problemas

Se você seguir os métodos de criação corretos ao criar scripts, os filmes terão menos erros de programação. Você pode usar as seguintes diretrizes para ajudar a impedir problemas e a corrigi-los rapidamente quando ocorrerem.

### Usando métodos de criação corretos

Recomenda-se salvar várias versões do filme durante o trabalho. Para salvar uma versão com um nome diferente a cada meia hora, escolha Arquivo > Salvar como. Você pode usar o histórico de revisão para identificar o momento em que um problema começou localizando o arquivo mais recente sem esse problema. Usando essa abordagem, sempre haverá uma versão funcionando, mesmo que um arquivo seja corrompido.

Outro método de criação importante é testar assim que possível, frequentemente e em todas as plataformas de destino para localizar problemas tão logo eles ocorram. Sempre que fizer uma alteração significativa ou antes de salvar uma versão, escolha Controlar > Testar Filme para executar o filme em modo de teste. Em modo de teste, o filme é executado em uma versão do player independente.

Se o filme for exibido para o público-alvo na Web, será importante testá-lo também em um navegador. Em determinadas situações (por exemplo, ao desenvolver um site de intranet), você poderá conhecer o navegador e a plataforma do público-alvo. Entretanto, se estiver desenvolvendo um site da Web, teste o filme em todos os navegadores e em todas as plataformas potenciais.

Recomenda-se seguir estes métodos de criação de páginas:

- Use a ação `trace` para enviar comentários para a janela Saída. (Consulte “Usando trace”, na página 171.)
- Use a ação `comment` para incluir observações instrutivas que são exibidas somente no painel Ações. (Consulte “Comentários”, na página 53.)
- Use convenções de nomenclatura consistentes para identificar elementos em um script. Por exemplo, deve-se evitar espaços em nomes. Inicie nomes de variáveis e funções com letra minúscula e use maiúsculas para cada palavra nova (`myVariableName`, `myFunctionName`). Inicie nome de funções construtoras com letra maiúscula (`MyConstructorFunction`). É muito importante escolher um estilo que faça sentido para você e usá-lo de modo consistente.



- Use nomes de variáveis que reflitam o tipo de informação que a variável contém. Por exemplo, uma variável que contenha informações sobre o último botão pressionado poderia ser chamada de `lastButtonPressed`. Um nome como `foo` dificultaria o reconhecimento da variável.
- Use campos de texto editáveis em camadas guia para controlar valores variáveis, em vez de usar o Depurador.
- Use o Movie Explorer em modo de edição de filme para ver a lista de exibição e todas as ações em um filme. Consulte Ajuda do Flash.
- Use a ação `for...in` para criar loops das propriedades de clipes de filme, incluindo clipes de filme filhos. A ação `for...in` pode ser usada com a ação `trace` para enviar uma lista de propriedades para a janela Saída. Consulte “Repetindo uma ação”, na página 72.

## Usando uma lista de verificação de solução de problemas

Como em todos os ambientes de scripts, há determinados erros que os programadores normalmente cometem. A lista a seguir oferece um bom ponto de partida para solucionar problemas com seu filme:

- Certifique-se de que esteja no modo de teste de filme.  
Somente os botões e as ações simples (por exemplo, `gotoAndPlay` e `stop`) funcionarão no modo de criação. Escolha Controlar > Ativar Ações de Quadro Simples ou Controlar > Ativar Botões Simples para ativar essas ações.
- Certifique-se de não haver ações de quadro em várias camadas que estejam em conflito umas com as outras.
- Se estiver trabalhando em Modo Normal no painel Ações, certifique-se de que seu comando esteja definido como expressão.  
Se estiver passando uma expressão em uma ação e não tiver marcado a caixa Expressão, o valor será passado como sequência de caracteres. Consulte “Usando operadores para manipular valores em expressões”, na página 62.
- Certifique-se de que elementos ActionScripts diferentes não tenham o mesmo nome.  
É melhor dar um nome exclusivo a cada variável, função, objeto e propriedade. Entretanto, as variáveis locais são uma exceção. Elas só precisam ser exclusivas dentro de seu escopo e são reutilizadas freqüentemente como contadores. Consulte “Atribuindo um escopo a uma variável”, na página 59.

Para obter mais dicas sobre solução de problemas em um filme do Flash, consulte o Flash Support Center em <http://www.macromedia.com/support/flash> (site em inglês).



## Usando o Depurador

O Depurador permite encontrar erros em um filme enquanto ele estiver sendo executado no Flash Player. Você pode ver a lista de exibição de clipes de filme e filmes carregados e alterar os valores de variáveis e de propriedades, determinando os valores corretos. Depois, pode voltar para os scripts e editá-los para que produzam os resultados corretos. Para usar o Depurador, é necessário executar o Flash Debug Player, que é uma versão especial do Flash Player.

O Flash Debug Player é instalado automaticamente com o aplicativo de criação Flash 5. Ele permite fazer o download da lista de exibição, dos pares de nome e valor de variáveis e dos pares de nome e valor de propriedades para o Depurador no Flash.

### Para exibir o Depurador:

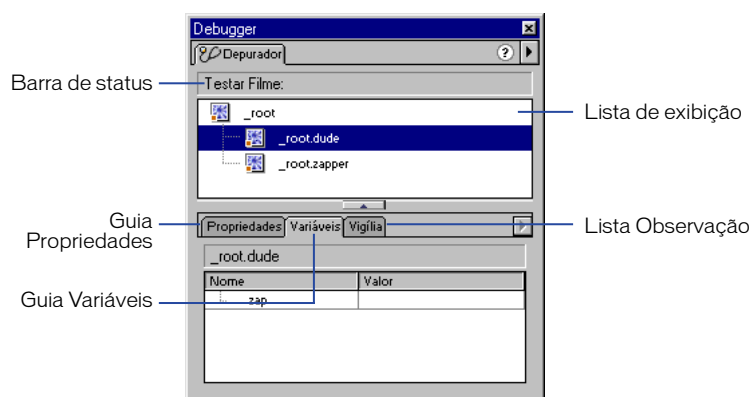
Escolha Janela > Depurador.

O Depurador será aberto em estado inativo. Não são mostradas informações na lista de exibição até que seja emitido um comando do Flash Player.

### Para ativar o Depurador no modo de teste de filme:

Escolha Controlar > Depurar Filme.

O Depurador será aberto em estado ativo.





## Ativando a depuração em um filme

Ao exportar um filme do Flash Player, você pode ativar a depuração no filme e criar uma senha de depuração. Se você não ativar a depuração, o Depurador não será ativado.

Como em JavaScript ou HTML, qualquer variável ActionScript da parte cliente pode ser visualizada pelo usuário. Para armazenar variáveis de maneira segura, você deve enviá-las a um aplicativo no servidor, em vez de armazená-las no filme.

Entretanto, como desenvolvedor do Flash, você pode ter outros segredos comerciais, como estruturas de clipes de filmes, que não deseja revelar. Para garantir que somente usuários confiáveis possam assistir seus filmes no Flash Debug Player, você pode publicá-los com uma senha do Depurador.

### Para ativar a depuração e criar uma senha:

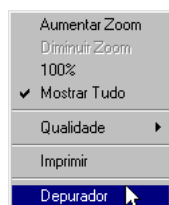
- 1 Escolha Arquivo > Configurações de Publicação.
- 2 Clique na guia Flash.
- 3 Selecione Depuração Permitida.
- 4 Para definir uma senha, insira-a na caixa Senha.

Sem essa senha, não será possível fazer download de informações para o Depurador. Se deixar o campo de senha em branco, não será exigida uma senha.

### Para ativar o Depurador em um navegador:

- 1 Clique com o botão direito (Windows) ou clique mantendo a tecla Control pressionada (Macintosh) para abrir o menu de contexto do Flash Debug Player.
- 2 Escolha Depurador.

**Observação:** O Depurador pode ser usado para monitorar somente um filme de cada vez. Para usar o Depurador, o Flash deve estar aberto.



*menu de contexto do Flash Debug Player.*

## Sobre a barra de status

Uma vez ativado, a barra de status do Depurador exibe a URL ou caminho de arquivo local do filme. O Flash Player é implementado de formas diferentes, dependendo do ambiente de reprodução. A barra de status do Depurador exibe o tipo de Flash Player que executa o filme.

- Modo de teste de filme
- Player independente
- Plug-in Netscape

O plug-in Netscape é usado com o Netscape Navigator no Windows e no Macintosh e com o Microsoft Internet Explorer no Macintosh.

- Controle ActiveX

O controle ActiveX é usado com o Internet Explorer no Windows.

## Sobre a lista de exibição

Quando o Depurador está ativo, mostra uma lista de exibição dinâmica do clipe de filme. Você pode expandir ou recolher a ramificação para exibir todos os cliques de filme que estão no Palco no momento. Quando os cliques de filme são adicionados ou removidos do filme, a lista de exibição reflete as alterações imediatamente. Ela pode ser redimensionada movendo-se o divisor horizontal ou arrastando-se o canto inferior direito.

## Exibindo e modificando variáveis

No Depurador, a guia Variáveis exibe os nomes e valores de qualquer variável no filme. Se você alterar o valor de uma variável na guia Variáveis, poderá ver a alteração refletida no filme enquanto ele é executado. Por exemplo, para testar a detecção de colisão em um jogo, você poderia inserir o valor de variável que posicione uma bola no local correto próximo a uma parede.

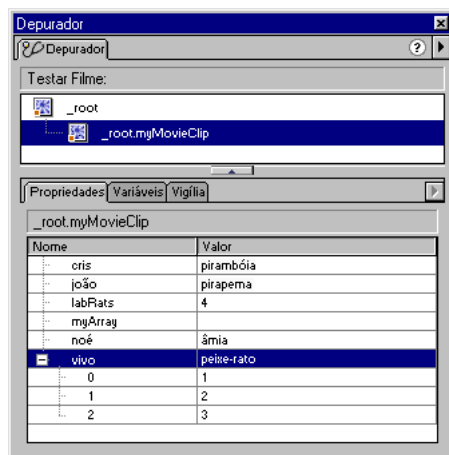
**Para exibir uma variável:**

- 1 Selecione o clipe de filme que contém a variável da lista de exibição.
- 2 Clique na guia Variáveis.





A lista de exibição é atualizada automaticamente enquanto o filme é reproduzido. Se um clipe de filme for removido do filme em um determinado quadro, também será removido da lista de exibição no Depurador; dessa forma, os valores e nomes de variáveis são removidos.



#### Para modificar o valor de uma variável:

Selecione o valor e insira um novo.

O valor deve ser constante (por exemplo, "Hello", 3523 ou "http://www.macromedia.com"), não uma expressão (por exemplo,  $x + 2$  ou `eval("name: " + i)`). O valor pode ser uma sequência de caracteres (qualquer valor entre aspas – “”), um número ou um Booleano (true ou false).

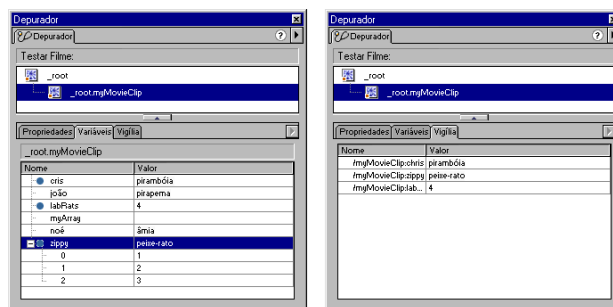
As variáveis Object e Array são exibidas na guia Variáveis. Clique no botão com sinal de adição (+) para ver suas propriedades e valores. Entretanto, não é possível inserir valores de Object e Array (por exemplo, {name: "este é um objeto"} ou [1, 2, 3]) nos campos de valores.

**Observação:** Para enviar o valor de uma expressão em modo de teste de filme, use a ação trace. Consulte “Usando trace”, na página 171.

## Usando a lista Observação

Para monitorar um conjunto de variáveis críticas de forma organizada, você pode marcá-las para serem exibidas na lista de observação. Essa lista exibe o caminho absoluto para a variável e o valor. Você também pode inserir o valor de uma nova variável na lista de observação.

Somente variáveis podem ser adicionadas à lista de observação, não as propriedades e funções.



*Variáveis marcadas para a lista Observação e variáveis na lista Observação.*

**Para adicionar variáveis à lista de observação, escolha uma destas opções:**

- Na guia Variáveis, clique com o botão direito (Windows) ou clique mantendo a tecla Control pressionada (Macintosh) em uma variável selecionada e escolha Observar no menu de contexto. É exibido um ponto azul próximo à variável.
- Na guia Observação, clique com o botão direito (Windows) ou clique mantendo a tecla Control pressionada (Macintosh) e escolha Adicionar no menu de contexto. Insira o nome da variável e o valor nos campos.

**Para remover as variáveis da lista de observação:**

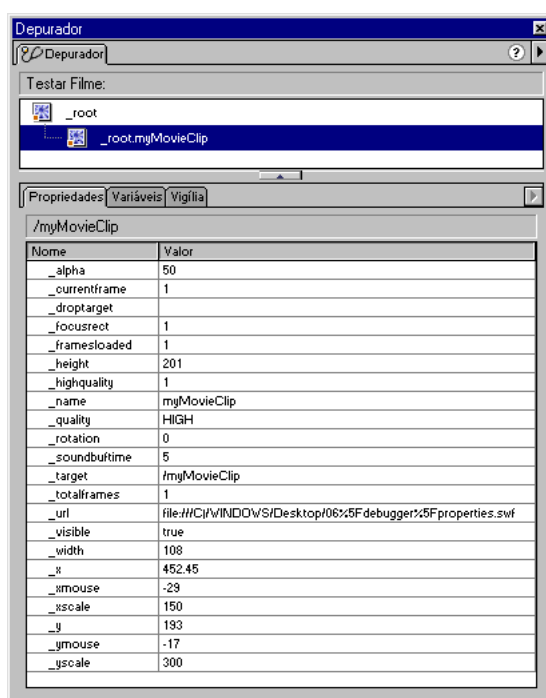
Na guia Observação, clique com o botão direito (Windows) ou clique mantendo a tecla Control pressionada (Macintosh) e escolha Remover no menu de contexto.

## Exibindo propriedades do filme e alterando propriedades editáveis

A guia Propriedades do Depurador exibe todos os valores de propriedades de qualquer clipe de filme no Palco. Você pode alterar o valor de uma propriedade e ver a alteração refletida no filme enquanto ele é executado. Algumas propriedades de clipe de filme são somente leitura e não podem ser alteradas.

Para exibir as propriedades de um clipe de filme:

- 1 Selecione um clipe de filme na lista de exibição.
- 2 Clique na guia Propriedades.



#### Para modificar o valor de uma propriedade:

Selecione o valor e insira um novo.

O valor deve ser uma constante (por exemplo, 50 ou "clearwater") em vez de uma expressão (por exemplo,  $x + 50$ ). O valor pode ser uma sequência de caracteres (qualquer valor entre aspas – " "), um número ou um Booleano (true ou false). Não é possível inserir valores de objetos e matrizes (por exemplo, {id: "rogue"} ou [1, 2, 3]) no Depurador.

Para obter mais informações, consulte "String", na página 54 e "Usando operadores para manipular valores em expressões", na página 62.

**Observação:** Para enviar o valor de uma expressão em modo de teste de filme, use a ação trace. Consulte "Usando trace", na página 171.

## Usando a janela Saída

Em modo de teste de filme, a janela Saída exibe informações para ajudá-lo a solucionar problemas com o filme. Algumas informações, como erros de sintaxe, são exibidas automaticamente. Você pode exibir outras informações usando os comandos Listar Objetos e Listar Variáveis. Consulte "Usando Listar Objetos", na página 169 e "Usando Listar Variáveis", na página 170.

Se você usar a ação trace em seus scripts, poderá enviar informações específicas para a janela Saída enquanto o filme é executado. Isso poderia incluir observações sobre o status do filme ou o valor de uma expressão. Consulte "Usando trace", na página 171.

#### Para exibir a janela Saída:

- 1 Se o filme não estiver sendo executado em modo de teste, escolha Controlar > Testar Filme.
- 2 Escolha Janela > Saída.  
A janela Saída é exibida.

**Observação:** Se houver erros de sintaxe em um script, a janela Saída será exibida automaticamente.

3 Para trabalhar com o conteúdo da janela Saída, use o menu Opções:

- Para copiar o conteúdo da janela Saída para a Área de transferência, escolha Opções > Copiar.
- Para limpar o conteúdo da janela, escolha Opções > Limpar.
- Para salvar o conteúdo da janela em um arquivo de texto, escolha Opções > Salvar.
- Para imprimir o conteúdo da janela, escolha Opções > Imprimir.

## Usando Listar Objetos

No modo de teste de filme, o comando Listar Objetos exibe o nível, o quadro, o tipo de objeto (forma, filme, clipe ou botão) e o caminho de destino de uma instância de clipe de filme em uma lista hierárquica. Isso é especialmente útil para localizar o caminho de destino e o nome da instância corretos. Ao contrário do Depurador, a lista não é atualizada automaticamente enquanto o filme é exibido; o comando Listar Objetos deve ser escolhido toda vez que você desejar enviar as informações para a janela Saída.

**Para exibir uma lista de objetos em um filme:**

- 1 Se o filme não estiver sendo executado em modo de teste, escolha Controlar > Testar Filme.
- 2 Escolha Depurar > Listar Objetos.

Uma lista de todos os objetos no Palco é exibida na janela Saída, como neste exemplo:

```
Layer #0: Frame=3
Clipe de Filme: Frame=1 Target=_root.MC
  Forma:
    Clipe de Filme: Frame=1 Target=_root.instance3
      Forma:
        Botão:
          Clipe de Filme: Frame=1 Target=_root.instance3.instance2
            Forma:
```

**Observação:** O comando Listar Objetos não lista todos os objetos de dados do ActionScript. Neste contexto, um objeto é considerado como uma forma ou símbolo no Palco.

## Usando Listar Variáveis

Em modo de teste de filme, o comando Listar Variáveis exibe uma lista de todas as variáveis no filme no momento. Isso é especialmente útil para localizar o caminho de destino e o nome da variável corretos. Ao contrário do Depurador, a lista não é atualizada automaticamente enquanto o filme é exibido; o comando Listar Variáveis deve ser escolhido toda vez que você deseja enviar as informações para a janela Saída.

**Para exibir uma lista de variáveis em um filme:**

- 1 Se o filme não estiver sendo executado em modo de teste, escolha Controlar > Testar Filme.
- 2 Escolha Depurar > Listar Variáveis.

Uma lista de todas as variáveis no filme é exibida na janela Saída, como neste exemplo:

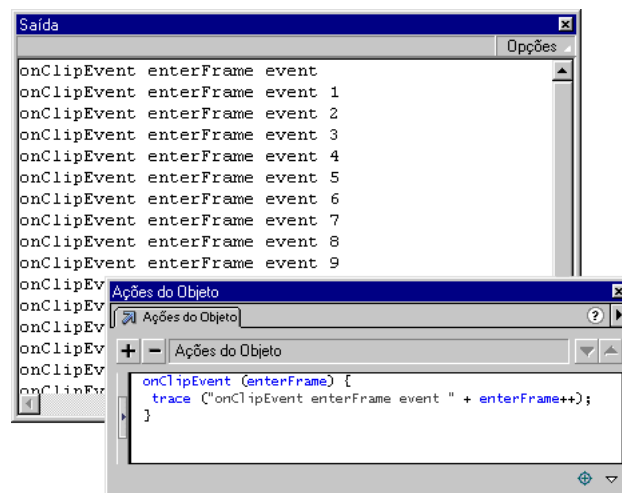
```
Level #0:  
  Variable _root.country = "Sweden"  
  Variable _root.city = "San Francisco"  
Movie Clip: Target=""  
Variable _root.instance1.firstName = "Rick"
```

## Usando trace

Ao usar a ação `trace` em um script, você pode enviar informações para a janela Saída. Por exemplo, ao testar um filme ou uma cena, você pode enviar observações de programação específicas para a janela ou fazer com que os resultados sejam exibidos quando for pressionado um botão ou quando um quadro for reproduzido. A ação `trace` é semelhante ao comando JavaScript `alert`.

Quando você usa a ação `trace` em um script, você pode usar expressões como argumentos. O valor de uma expressão é exibido na janela Saída no modo de teste de filme, como a seguir:

```
onClipEvent(enterFrame) {
    trace("onClipEvent enterFrame" + enterFrame++)
}
```



*A ação `trace` retorna valores que são exibidos na janela Saída.*







## CAPÍTULO 7

### Dicionário ActionScript

.....

Esta parte do *Manual de Referência do ActionScript* descreve a sintaxe e o uso dos elementos do ActionScript no Flash 5 e em versões posteriores. As entradas neste manual são as mesmas da Ajuda do Dicionário ActionScript. Para usar os exemplos em um script, copie o texto de exemplo da Ajuda do dicionário ActionScript e cole-o no painel Ações no Modo Especialista.

O dicionário lista todos os elementos do ActionScript — operadores, palavras-chave, comandos, ações, propriedades, funções, objetos e métodos. Para obter uma visão geral de todas as entradas do dicionário, consulte Conteúdo do dicionário, na página 176; as tabelas desta seção são um bom ponto de partida para verificar operadores simbólicos e métodos cuja classe de objeto você não conheça.

O ActionScript segue o padrão ECMA-262 (a especificação escrita pela Associação Européia de Fabricantes de Computadores), salvo indicação em contrário.

Há dois tipos de entradas neste dicionário:

- Entradas individuais para operadores, palavras-chave, funções, variáveis, propriedades, métodos e comandos;
- Entradas de objetos, que fornecem detalhes em geral sobre objetos predefinidos.

Use as informações nas entradas de exemplo para interpretar a estrutura e as convenções usadas nesses dois tipos de entradas.

## Exemplo de entrada para a maioria dos elementos do ActionScript

O exemplo de entrada do dicionário a seguir explica as convenções usadas para todos os elementos do ActionScript que não sejam objetos.

### Título da entrada

Todas as entradas são listadas em ordem alfabética. A ordem ignora maiúsculas e minúsculas, sublinhados no início e assim por diante.

### Sintaxe

A seção “Sintaxe” fornece a sintaxe correta para uso do elemento do ActionScript em seu código. A parte de código da sintaxe está em *fonte de código* e os argumentos que devem ser fornecidos por você estão em *fonte de código em itálico*. Chaves indicam argumentos de código opcionais.

### Argumentos

Esta seção descreve os argumentos listados na sintaxe.

### Descrição

Esta seção identifica o elemento (por exemplo, como um operador, um método, uma função ou outro elemento) e descreve como ele é usado.

### Exibidor

Esta seção informa quais versões do Exibidor oferecem suporte ao elemento. Isso não é o mesmo que a versão do Flash usada para criar o conteúdo. Por exemplo, se você estiver criando conteúdo para o Flash 4 Player com a ferramenta de criação Flash 5, não poderá usar elementos do ActionScript que só estejam disponíveis para o Flash 5 Player.

Com a introdução do ActionScript do Flash 5, alguns elementos do ActionScript do Flash 4 (e anteriores) ficaram obsoletos. Apesar do Flash 5 Player ainda oferecer suporte aos elementos obsoletos, recomenda-se que você use os novos elementos do Flash 5.

Além disso, a funcionalidade de operadores foi muito ampliada no Flash 5. Não só foram introduzidos vários novos operadores matemáticos, como alguns dos operadores antigos são agora capazes de manipular novos tipos de dados. Para manter a consistência de tipos de dados, os arquivos do Flash 4 são modificados automaticamente quando importados para o ambiente de criação do Flash 5, mas essas modificações não afetarão a funcionalidade do script original. Para obter mais informações, consulte as entradas de + (adição), < (menor que), > (maior que), <= (menor ou igual a), >= (maior ou igual a), != (diferença) e = (igualdade).

### Exemplo

Esta seção fornece um exemplo de código que demonstra como usar o elemento.

### Consulte também

Esta seção lista entradas do dicionário ActionScript relacionadas.

## Exemplo de entrada para objetos

O seguinte exemplo de entrada do dicionário explica as convenções usadas para objetos ActionScript predefinidos. Os objetos são listados em ordem alfabética com todos os outros elementos no dicionário.

### Título da entrada

O título da entrada fornece o nome do objeto. O nome do objeto é seguido por um parágrafo que contém informações gerais sobre o objeto.

### Tabelas de resumo de método e propriedade

Cada objeto contém uma tabela que lista todos os métodos associados ao objeto. Se o objeto tiver propriedades (normalmente constantes), esses elementos são resumidos na tabela adicional. Todos os métodos e propriedades listados nessas tabelas também têm suas próprias entradas do dicionário, que seguem a entrada do objeto.

### Construtor

Se o objeto necessitar do uso de um construtor para acessar seus métodos e propriedades, o construtor é descrito no fim da entrada do objeto. Essa descrição tem todos os elementos padrão (descrição de sintaxe, etc.) das outras entradas do dicionário.

### Listagens de métodos e propriedades

Os métodos e as propriedades de um objeto são listadas em ordem alfabética após a entrada do objeto.

## Conteúdo do dicionário

Todas as entradas do dicionário são listadas em ordem alfabética. Entretanto, alguns operadores são símbolos e são apresentados na ordem ASCII. Além disso, os métodos associados a um objeto são listados junto com o nome do objeto — por exemplo, o método `abs` do objeto `Math` é listado como `Math.abs`.

As duas tabelas a seguir o ajudarão a localizar esses elementos. A primeira lista os operadores simbólicos na ordem em que ocorrem no dicionário. A segunda lista todos os outros elementos do `ActionScript`.

**Observação:** Para operadores de precedência e de associatividade, consulte o apêndice A.

### Operadores simbólicos

--	(decremento)
++	incremento
!	(NOT lógico)
!=	(diferença)
%	(módulo)
%=	(atribuição de módulo)
&	(AND bit a bit)
&&	(AND de curto-circuito)
&=	(atribuição AND bit a bit)
()	(parênteses)
-	(subtração)
*	(multiplicação)
*=	(atribuição de multiplicação)
,	(vírgula)
.	(ponto)
? :	(condicional)
/	(divisão)
//	(delimitador de comentário)
/*	(delimitador de comentário)
/=	(atribuição de divisão)



## Operadores simbólicos

[]	(acesso de matriz)
^	(XOR bit a bit)
^=	(atribuição XOR bit a bit)
{}	(inicializador de objeto)
	(OR bit a bit)
	(OR lógico)
=	(atribuição OR bit a bit)
-	(NOT bit a bit)
+	(adição)
+=	(atribuição de adição)
<	(menor que)
<<	(deslocamento para a esquerda bit a bit)
<<=	(deslocamento para a esquerda bit a bit e atribuição)
<=	(menor ou igual a)
<>	(diferença)
=	(atribuição)
-=	(atribuição de negação)
==	(igualdade)
>	(maior que)
>=	(maior ou igual a)
>>	(deslocamento para a direita bit a bit)
>>=	(deslocamento para a direita bit a bit e atribuição)
>>>	(deslocamento para a direita não assinado bit a bit)
>>>=	(deslocamento para a direita não assinado bit a bit e atribuição)



A tabela a seguir lista todos os elementos do ActionScript que não são operadores simbólicos.

Elemento do ActionScript	Consulte a entrada
abs	Math.abs, na página 299
acos	Math.acos, na página 299
add	add, na página 222
and	and, na página 223
_alpha	_alpha, na página 222
appendChild	XML.appendChild, na página 397
Array	Array (objeto), na página 224
asin	Math.asin, na página 300
atan	Math.atan, na página 300
atan2	Math.atan2, na página 301
attachMovie	MovieClip.attachMovie, na página 315
attachSound	Sound.attachSound, na página 361
attributes	XML.attributes, na página 398
BACKSPACE	Key.BACKSPACE, na página 285
Booleano	Boolean (função), na página 233, Boolean (objeto), na página 234
break	break, na página 235
call	call, na página 236
CAPSLOCK	Key.CAPSLOCK, na página 285
ceil	Math.ceil, na página 301
charAt	String.charAt, na página 374
charCodeAt	String.charCodeAt, na página 374
childNodes	XML.childNodes, na página 398
chr	chr, na página 236
cloneNode	XML.cloneNode, na página 399
close	XMLSocket.close, na página 414
Cor	Color (objeto), na página 237



Elemento do ActionScript	Consulte a entrada
concat	Array.concat, na página 226, String.concat, na página 375
connect	XMLSocket.connect, na página 414
constructor	Array, Boolean, Color, Date, Number, Object, Sound, String, XML, XMLSocket
continue	continue, na página 241
CONTROL	Key.CONTROL, na página 285
cos	Math.cos, na página 301
createElement	XML.createElement, na página 399
createTextNode	XML.createTextNode, na página 399
_currentframe	_currentframe, na página 242
Date	Date (objeto), na página 242
delete	delete, na página 261
DELETEKEY	Key.DELETEKEY, na página 286
docTypeDecl	XML.docTypeDecl, na página 400
do...while	do... while, na página 262
DOWN	Key.DOWN, na página 286
_droptarget	_droptarget, na página 263
duplicateMovieClip	duplicateMovieClip, na página 264, MovieClip.duplicateMovieClip, na página 316
E	Math.E, na página 302
else	else, na página 265
END	Key.END, na página 286
ENTER	Key.ENTER, na página 287
eq	eq (igual – específico de sequência de caracteres), na página 265
escape (função)	escape, na página 265
ESCAPE (constante)	Key.ESCAPE, na página 287
eval	eval, na página 266
evaluate	evaluate, na página 267



Elemento do ActionScript	Consulte a entrada
exp	Math.exp, na página 302
firstChild	XML.firstChild, na página 401
floor	Math.floor, na página 303
_focusrect	_focusrect, na página 267
for	for, na página 268
for..in	for..in, na página 269
_framesloaded	_framesloaded, na página 271
fromCharCode	String.fromCharCode, na página 375
fscommand	fscommand, na página 271
function	function, na página 272
ge	ge (maior ou igual a – específico de seqüências de caracteres), na página 273
getAscii	Key.getAscii, na página 287
getBeginIndex	Selection.getBeginIndex, na página 355
getBounds	MovieClip.getBounds, na página 316
getBytesLoaded	MovieClip.getBytesLoaded, na página 317
getBytesTotal	MovieClip.getBytesTotal, na página 317
getCaretIndex	Selection.getCaretIndex, na página 356
getCode	Key.getCode, na página 288
getDate	Date.getDate, na página 246
getDay	Date.getDay, na página 246
getEndIndex	Selection.getEndIndex, na página 356
getFocus	Selection.getFocus, na página 356
getFullYear	Date.getFullYear, na página 246
getHours	Date.getHours, na página 247
getMilliseconds	Date.getMilliseconds, na página 247
getMinutes	Date.getMinutes, na página 248
getMonth	Date.getMonth, na página 248
getPan	Sound.getPan, na página 361





Elemento do ActionScript	Consulte a entrada
getProperty	getProperty, na página 274
getRGB	Color.setRGB, na página 239
getSeconds	Date.getSeconds, na página 248
getTime	Date.getTime, na página 249
getTimer	getTimer, na página 274
getTimezoneOffset	Date.getTimezoneOffset, na página 249
getTransform	Color.getTransform, na página 238, Sound.getTransform, na página 362
getURL	getURL, na página 275, MovieClip.getURL, na página 318
getUTCDate	Date.getUTCDate, na página 250
getUTCDay	Date.getUTCDay, na página 250
getUTCFullYear	Date.getUTCFullYear, na página 250
getUTCHours	Date.getUTCHours, na página 251
getUTCMilliseconds	Date.getUTCMilliseconds, na página 251
getUTCMinutes	Date.getUTCMinutes, na página 251
getUTCMonth	Date.getUTCMonth, na página 252
getUTCSeconds	Date.getUTCSeconds, na página 252
getVersion	getVersion, na página 276
getVolume	Sound.getVolume, na página 362
getYear	Date.getYear, na página 252
globalToLocal	MovieClip.globalToLocal, na página 318
gotoAndPlay	gotoAndPlay, na página 277, MovieClip.gotoAndPlay, na página 319
gotoAndStop	gotoAndStop, na página 277, MovieClip.gotoAndStop, na página 319
gt	gt (maior que – específico de seqüências de caracteres), na página 278
hasChildNodes	XML.hasChildNodes, na página 401
_height	_height, na página 278
hide	Mouse.hide, na página 313



Elemento do ActionScript	Consulte a entrada
_highquality	_highquality, na página 279
hitTest	MovieClip.hitTest, na página 320
HOME	Key.HOME, na página 288
if	if, na página 279
ifFrameLoaded	ifFrameLoaded, na página 280
#include	#include, na página 280
indexOf	String.indexOf, na página 376
Infinity	Infinity, na página 281
INSERT	Key.INSERT, na página 288
insertBefore	XML.insertBefore, na página 402
int	int, na página 281
isDown	Key.isDown, na página 289
isFinite	isFinite, na página 281
isNaN	isNaN, na página 282
isToggled	Key.isToggled, na página 289
join	Array.join, na página 226
Key	Key (objeto), na página 283
lastChild	XML.lastChild, na página 402
lastIndexOf	String.indexOf, na página 376
le	le (menor que ou igual a – específico da sequência de caracteres), na página 292
LEFT	Key.LEFT, na página 289
length	length, na página 292, Array.length, na página 227, String.length, na página 377
LN2	Math.LN2, na página 305
LN10	Math.LN10, na página 305
load	XML.load, na página 403
loaded	XML.loaded, na página 403
loadMovie	loadMovie, na página 294, MovieClip.loadMovie, na página 321

Elemento do ActionScript	Consulte a entrada
loadVariables	loadVariables, na página 295, MovieClip.loadVariables, na página 322
localToGlobal	MovieClip.localToGlobal, na página 323
log	Math.log, na página 303
LOG2E	Math.LOG2E, na página 304
LOG10E	Math.LOG10E, na página 304
lt	le (menor que ou igual a – específico da sequência de caracteres), na página 292
Math	Math (objeto), na página 297
max	Math.max, na página 306
maxscroll	maxscroll, na página 310
MAX_VALUE	Number.MAX_VALUE, na página 334
mbchr	mbchr, na página 311
mblength	mblength, na página 311
mbord	mbord, na página 311
mbsubstring	mbsubstring, na página 312
min	Math.min, na página 306
MIN_VALUE	Number.MIN_VALUE, na página 335
Mouse	Mouse (objeto), na página 312
MovieClip	MovieClip (objeto), na página 314
_name	_name, na página 327
NaN	NaN, na página 328, Number.NaN, na página 335
ne	ne (diferente – específico de sequência de caracteres), na página 328
NEGATIVE_INFINITY	Number.NEGATIVE_INFINITY, na página 335
new ( <b>operador</b> )	new, na página 328
newline	newline, na página 329
nextFrame	nextFrame, na página 330, MovieClip.nextFrame, na página 324
nextScene	nextScene, na página 330



Elemento do ActionScript	Consulte a entrada
nextSibling	XML.nextSibling, na página 404
nodeName	XML.nodeName, na página 404
nodeType	XML.nodeType, na página 405
nodeValue	XML.nodeValue, na página 405
not	not, na página 331
null	null, na página 331
Number	Number (função), na página 332, Number (objeto), na página 332
Object	Object (objeto), na página 337
On	on(mouseEvent), na página 341
onClipEvent	onClipEvent, na página 339
onClose	XMLSocket.onClose, na página 416
onConnect	XMLSocket.onConnect, na página 416
OnLoad	XML.onLoad, na página 406
onXML	XMLSocket.onXML, na página 418
or (OR lógico)	or, na página 342
ord	ord, na página 343
_parent	_parent, na página 343
parentNode	XML.parentNode, na página 407
parseFloat	parseFloat, na página 344
parseInt	parseInt, na página 344
parseXML	XML.parseXML, na página 407
PGDN	Key.PGDN, na página 290
PGUP	Key.PGUP, na página 290
PI	Math.PI, na página 307
play	play, na página 345, MovieClip.play, na página 324
pop	Array.pop, na página 228
POSITIVE_INFINITY	Number.POSITIVE_INFINITY, na página 336
pow	Math.pow, na página 307



Elemento do ActionScript	Consulte a entrada
prevFrame	prevFrame, na página 346, MovieClip.prevFrame, na página 324
previousSibling	XML.previousSibling, na página 407
prevScene	prevScene, na página 347
print	print, na página 347
printAsBitmap	printAsBitmap, na página 348
push	Array.push, na página 228
_quality	_quality, na página 350
random	random, na página 350, Math.random, na página 307
removeMovieClip	removeMovieClip, na página 351, MovieClip.removeMovieClip, na página 325
removeNode	XML.removeNode, na página 408
return	return, na página 351
reverse	Array.reverse, na página 229
RIGHT	Key.RIGHT, na página 290
_root	_root, na página 352
_rotation	_rotation, na página 353
round	Math.round, na página 308
scroll	scroll, na página 354
Selection	Selection (objeto), na página 354
send	XML.send, na página 408, XMLSocket.send, na página 419
sendAndLoad	XML.sendAndLoad, na página 408
set	set, na página 358
setDate	Date.setDate, na página 253
setFocus	Selection.setFocus, na página 357
setFullYear	Date.setFullYear, na página 253
setHours	Date.setHours, na página 254
setMilliseconds	Date.setMilliseconds, na página 254

Elemento do ActionScript	Consulte a entrada
setMinutes	Date.setMinutes, na página 254
setMonth	Date.setMonth, na página 255
setPan	Sound.setPan, na página 362
setProperty	setProperty, na página 359
setRGB	Color.setRGB, na página 239
setSeconds	Date.setSeconds, na página 255
setSelection	Selection.setSelection, na página 357
setTime	Date.setTime, na página 255
setTransform	Color.setTransform, na página 239, Sound.setTransform, na página 363
setUTCDate	Date.setUTCDate, na página 256
setUTCFullYear	Date.setUTCFullYear, na página 256
setUTCHours	Date.setUTCHours, na página 257
setUTCMilliseconds	Date.setUTCMilliseconds, na página 257
setUTCMinutes	Date.setUTCMinutes, na página 257
setUTCMonth	Date.setUTCMonth, na página 258
setUTCSeconds	Date.setUTCSeconds, na página 258
setVolume	Sound.setVolume, na página 366
setYear	Date.setYear, na página 259
shift (método)	Array.shift, na página 229
SHIFT (constante)	Key.SHIFT, na página 291
show	Mouse.show, na página 313
sin	Math.sin, na página 308
slice	Array.slice, na página 230, String.slice, na página 377
sort	Array.sort, na página 230
Sound	Sound (objeto), na página 359
_soundbuftime	_soundbuftime, na página 368
SPACE	Key.SPACE, na página 291



Elemento do ActionScript	Consulte a entrada
splice	Array.splice, na página 232
split	String.split, na página 378
sqrt	Math.sqrt, na página 309
SQRT1_2	Math.SQRT1_2, na página 309
SQRT2	Math.SQRT2, na página 309
start	Sound.start, na página 367
startDrag	startDrag, na página 368, MovieClip.startDrag, na página 325
status	XML.status, na página 409
stop	stop, na página 369, MovieClip.stop, na página 326, Sound.stop, na página 367
stopAllSounds	stopAllSounds, na página 369
stopDrag	stopDrag, na página 370, MovieClip.stopDrag, na página 326
String	String (função), na página 370, String (objeto), na página 372, " " (delimitador de sequência de caracteres), na página 371
substr	String.substr, na página 378
substring	substring, na página 380, String.substring, na página 379
swapDepths	MovieClip.swapDepths, na página 326
TAB	Key.TAB, na página 291
tan	Math.tan, na página 310
_target	_target, na página 381
targetPath	targetPath, na página 381
tellTarget	tellTarget, na página 382
this	this, na página 383
toggleHighQuality	toggleHighQuality, na página 384
toLowerCase	String.toLowerCase, na página 379
toString	Array.toString, na página 232, Boolean.toString, na página 235, Date.toString, na página 259, Number.toString, na página 336, Object.toString, na página 338, XML.toString, na página 410



Elemento do ActionScript	Consulte a entrada
_totalframes	_totalframes, na página 384
toUpperCase	String.toUpperCase, na página 380
trace	trace, na página 385
typeof	typeof, na página 386
unescape	unescape, na página 386
unloadMovie	unloadMovie, na página 387, MovieClip.unloadMovie, na página 327
unshift	Array.shift, na página 229
UP	Key.UP, na página 292
updateAfterEvent	updateAfterEvent, na página 387
_url	_url, na página 388
UTC	Date.UTC, na página 260
valueOf	Boolean.valueOf, na página 235, Number.valueOf, na página 337, Object.valueOf, na página 338
var	var, na página 388
_visible	_visible, na página 389
void	void, na página 389
while	while, na página 389
_width	_width, na página 391
with	with, na página 391
_x	_x, na página 394
XML	XML (objeto), na página 395
xmlDecl	XML.xmlDecl, na página 411
XMLSocket	XMLSocket (objeto), na página 412
_xmouse	_xmouse, na página 420
_xscale	_xscale, na página 420
_y	_y, na página 421
_ymouse	_ymouse, na página 421
_yscale	_yscale, na página 422



## -- (decremento)

### Sintaxe

--*expressão*  
*expressão*--

### Argumentos

*expressão* Uma variável, número, elemento em uma matriz ou propriedade de um objeto.

### Descrição

Operador; um operador unitário pré-decremento ou pós-decremento que subtrai 1 da *expressão*. A forma pré-decremento do operador (--*expressão*) subtrai 1 da *expressão* e retorna o resultado. A forma pós-decremento do operador (*expressão*--) subtrai 1 da *expressão* e retorna o valor inicial da *expressão* (o resultado anterior à subtração).

### Exibidor

Flash 4 ou posterior.

### Exemplo

A forma pré-decremento do operador decrementa x para 2 ( $x - 1 = 2$ ) e retorna o resultado como y:

```
x = 3;  
y = --x
```

A forma pós-decremento do operador decrementa x para 2 ( $x - 1 = 2$ ) e retorna o valor original ( $x = 3$ ) como o resultado y:

```
Se x = 3;  
y = x--
```

## ++ (incremento)

### Sintaxe

++*expressão*  
*expressão*++

### Argumentos

*expressão* Uma variável, número, elemento em uma matriz ou propriedade de um objeto.

### Descrição

Operador; um operador unitário pré-incremento ou pós-incremento que adiciona 1 à expressão. A forma pré-incremento do operador ( $++$ expressão) adiciona 1 à expressão e retorna o resultado. A forma pós-incremento do operador (expressão $++$ ) adiciona 1 à expressão e retorna o valor inicial da expressão (o resultado anterior à adição).

A forma pré-incremento do operador incrementa  $x$  para 2 ( $x + 1 = 2$ ) e retorna o resultado como  $y$ :

```
x = 1;
y = ++x
```

A forma pós-incremento do operador incrementa  $x$  para 2 ( $x + 1 = 2$ ) e retorna o valor original ( $x = 1$ ) como o resultado  $y$ :

```
x = 1;
y = x++
```

### Exibidor

Flash 4 ou posterior.

### Exemplo

O exemplo a seguir usa  $++$  como operador pré-incremento com um comando `while`.

```
i = 0
while(i++ < 5){
// esta seção será executada cinco vezes
}
```

O exemplo a seguir usa  $++$  como operador pré-incremento:

```
var a = [];
var i = 0;
while (i < 10) {
  a.push(++i);
}
trace(a.join());
```

Este script imprime o seguinte:

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

O exemplo a seguir usa  $++$  como operador pós-incremento:

```
var a = [];
var i = 0;
while (i < 10) {
  a.push(i++);
}
trace(a.join());
```

Este script imprime o seguinte:

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

## ! (NOT lógico)

### Sintaxe

`! expressão`

### Argumentos

*expressão* Uma variável ou expressão avaliada.

### Descrição

Operador (lógico); inverte o valor booleano de uma variável ou expressão. Se *expressão* for uma variável com um valor absoluto ou convertido `true`, `!variável` o valor de `! expressão` é `false`. Se a expressão `x && y` for avaliada como `false`, a expressão `!(x && y)` será avaliada como `true`. Esse operador é idêntico ao operador `not` usado no Flash 4.

### Exibidor

Flash 4 ou posterior.

### Exemplo

No exemplo a seguir, a variável `happy` é definida como `false`, a condição `if` avalia a condição `!happy`, e, se a condição for `true`, `trace` envia uma sequência de caracteres para a janela Saída.

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

O exemplo a seguir ilustra o resultado do operador `!`:

`! true` retorna `false`

`! false` retorna `true`

## != (diferença)

### Sintaxe

`expressão1 != expressão2`

### Argumentos

*expressão1*, *expressão2* Números, seqüências de caracteres, booleanos, variáveis, objetos, matrizes ou funções.

### Descrição

Operador (diferença); testa o oposto exato do operador `==`. Se *expressão1* for igual a *expressão2*, o resultado será `false`. Como com o operador `==`, a definição de *igual* depende dos tipos de dados comparados.

- Números, seqüências de caracteres e valores booleanos são comparados por valor.
- Variáveis, objetos, matrizes e funções são comparadas por referência.

#### Exibidor

Flash 5 ou posterior.

#### Exemplo

O exemplo a seguir ilustra o resultado do operador !=.

```
5 != 8 retorna true
```

```
5 != 5 retorna false
```

O exemplo a seguir ilustra o uso do operador != em um comando if:

```
a = "David";  
b = "Fool"  
if (a != b)  
trace("David is not a fool");
```

#### Consulte também

== (igualdade), na página 216

## % (módulo)

#### Sintaxe

*expressão1 % expressão2*

#### Argumentos

*expressão1*, *expressão2* Números, inteiros, números de ponto flutuante ou seqüências de caracteres que são convertidas em um valor numérico.

#### Descrição

Operador (aritmético); calcula o resto da *expressão1* dividida por *expressão2*. Se um dos argumentos (ou ambos) da *expressão* não for numérico, o operador módulo tentará convertê-lo(s) em números.

#### Exibidor

Flash 4 ou posterior. Em arquivos do Flash 4, o operador % é expandido no arquivo SWF como  $x - \text{int}(x/y) * y$  e pode não ser tão rápido ou preciso quanto a implementação do Flash 5 Player.

#### Exemplo

A seguir há um exemplo numérico do uso do operador %:

```
12 % 5 retorna 2
```

```
4,3 % 2,1 retorna 0,1
```

## %= (atribuição de módulo)

### Sintaxe

*expressão1* %= *expressão2*

### Argumentos

*expressão1*, *expressão2* Inteiros e variáveis.

### Descrição

Operador (atribuição); atribui a *expressão1* o valor de *expressão1* % *expressão2*.

### Exibidor

Flash 4 ou posterior.

### Exemplo

O exemplo a seguir ilustra o uso do operador %= com variáveis e números:

$x \text{ \%} = y$  é o mesmo que  $x = x \text{ \%} y$

Se  $x = 14$  e  $y = 5$  então

$x \text{ \%} = 5$  retorna 4

### Consulte também

% (módulo), na página 192

## & (AND bit a bit)

### Sintaxe

*expressão1* & *expressão2*

### Argumentos

*expressão1*, *expressão2* Qualquer número.

### Descrição

Operador (bit a bit); converte *expressão1* e *expressão2* em inteiros não assinados de 32 bits e executa uma operação AND booleana em cada bit dos argumentos inteiros. O resultado é um novo inteiro não assinado de 32 bits.

### Exibidor

Flash 5 ou posterior. No Flash 4, o operador & era usado para concatenar seqüências de caracteres. No Flash 5, o operador & é um AND bit a bit e os operadores add e + concatenam seqüências de caracteres. Os arquivos do Flash 4 que usam o operador & são atualizados automaticamente para usarem add quando trazidos para o ambiente de criação Flash 5.

## && (AND de curto-circuito)

### Sintaxe

*expressão1* && *expressão2*

### Argumentos

*expressão1*, *expressão2* Números, seqüências de caracteres ou funções.

### Descrição

Operador (lógico); executa uma operação booleana nos valores de uma ou de ambas as expressões. Faz com que o interpretador Flash avalie *expressão1* (a expressão da esquerda) e retorne `false` se a avaliação for `false`. Se *expressão1* for avaliada como `true`, *expressão2* (da direita) é avaliada. Se *expressão2* for avaliada como `true`, o resultado final será `true`; caso contrário, será `false`.

### Exibidor

Flash 4 ou posterior.

### Exemplo

Este exemplo atribui os valores das expressões avaliadas às variáveis `winner` e `loser` para realizar um teste:

```
winner = (chocolateEggs >=10) && (jellyBeans >=25);  
loser = (chocolateEggs <=1) && (jellyBeans <= 5);  
if (winner) {  
    alert = "Você venceu!";  
    if (loser) {  
        alert = "ISSO é uma caçada infeliz!";  
    }  
} else {  
    alert = "Somos todos vencedores!";  
}
```

## &= (atribuição AND bit a bit)

### Sintaxe

*expressão1* &= *expressão2*

### Argumentos

*expressão1*, *expressão2* Inteiros e variáveis.

### Descrição

Operador (atribuição bit a bit); atribui a *expressão1* o valor de *expressão1* & *expressão2*.

#### Exibidor

Flash 5 ou posterior.

#### Exemplo

O exemplo a seguir ilustra o uso do operador `&=` com variáveis e números:

`x &= y` é o mesmo que `x = x & y`

Se `x = 15` e `y = 9` então

`x &= 9` retorna 9

#### Consulte também

`&` (AND bit a bit), na página 193

## () (parênteses)

#### Sintaxe

*(expressão1, expressão2);*

*função(functionCall1, ..., functionCallN);*

#### Argumentos

*expressão1, expressão2* Números, seqüências de caracteres ou texto.

*função* A função a ser executada no conteúdo entre parênteses.

*functionCall1...functionCallN* Uma série de funções a serem executadas antes que o resultado seja passado para a função fora dos parênteses.

#### Descrição

Operador (geral); executa uma operação de agrupamento em um ou mais argumentos, ou envolve um ou mais argumentos e passa o resultado como parâmetro para uma função fora dos parênteses.

Uso 1: Executa uma operação de agrupamento em uma ou mais expressões para controlar a ordem de execução dos operadores na expressão. Esse operador substitui a ordem de precedência automática e faz com que as expressões entre parênteses sejam avaliadas primeiro. Quando os parênteses estão aninhados, o Flash avalia o conteúdo dos parênteses mais internos antes do conteúdo dos mais externos.

Uso 2: Envolve um ou mais argumentos e os passa como parâmetros para a função fora dos parênteses.

#### Exibidor

Flash 4 ou posterior.

### Exemplo

(Uso 1) – Os comandos a seguir ilustram o uso de parênteses para controlar a ordem de execução de expressões (o resultado é exibido abaixo de cada comando)

```
(2 + 3) * (4 + 5)
45
2 + (3 * (4 + 5))
29
2 + (3 * 4) + 5
19
```

(Uso 2) – O exemplo a seguir ilustra o uso dos parênteses com uma função:

```
getDate();
invoice(item, amount);
```

### Consulte também

with, na página 391

## - (subtração)

### Sintaxe

(Negação) *–expressão*

(Subtração) *expressão1 - expressão2*

### Argumentos

*expressão1*, *expressão2* Qualquer número.

### Descrição

Operador (aritmético); usado para negação ou subtração. Quando usado para negação, reverte o sinal da *expressão* numérica. Quando usado para subtração, executa uma subtração aritmética em duas expressões numéricas, subtraindo *expressão2* de *expressão1*. Quando ambas as expressões são inteiros, a diferença é um inteiro. Quando uma ou ambas as expressões são números de ponto flutuante, a diferença é um número de ponto flutuante.

### Exibidor

Flash 4 ou posterior.



#### Exemplo

(Negação) Este comando reverte o sinal da expressão  $2 + 3$ :

$-(2 + 3)$

O resultado é -5.

(Subtração) Este comando subtrai o inteiro 2 do inteiro 5:

$5 - 2$

O resultado é 3, que é um inteiro.

(Subtração): Este comando subtrai o número de ponto flutuante 1,5 do número de ponto flutuante 3,25:

$\text{put } 3,25 - 1,5$

O resultado é 1,75, que é um número de ponto flutuante.

## \* (multiplicação)

#### Sintaxe

*expressão1 \* expressão2*

#### Argumentos

*expressão1*, *expressão2* Números inteiros ou de ponto flutuante.

#### Descrição

Operador (aritmético); multiplica duas expressões numéricas. Se ambas as expressões forem inteiros, o produto será um inteiro. Se uma ou ambas as expressões forem números de ponto flutuante, o produto será um número de ponto flutuante.

#### Exibidor

Flash 4 ou posterior.

#### Exemplo

Este comando multiplica os inteiros 2 e 3:

$2 * 3$

O resultado é 6, que é um inteiro.

#### Exemplo

Este comando multiplica os números de ponto flutuante 2,0 e 3,1416:

$2.0 * 3.1416$

O resultado é 6,2832, que é um número de ponto flutuante.



## **\*= (atribuição de multiplicação)**

### **Sintaxe**

*expressão1* \*= *expressão2*

### **Argumentos**

*expressão1*, *expressão2* Inteiros, números de ponto flutuante ou seqüências de caracteres.

### **Descrição**

Operador (atribuição); atribui à *expressão1* o valor de *expressão1* \* *expressão2*.

### **Exibidor**

Flash 4 ou posterior.

### **Exemplo**

O exemplo a seguir ilustra o uso do operador \*= com variáveis e números:

$x *= y$  é o mesmo que  $x = x * y$

Se  $x = 5$  e  $y = 10$  então

$x *= 10$  retorna 50

### **Consulte também**

\* (multiplicação), na página 197

## **, (vírgula)**

### **Sintaxe**

*expressão1*, *expressão2*

### **Argumentos**

*expressão* Qualquer número, variável, seqüência de caracteres, elemento de matriz ou outros dados.

### **Descrição**

Operador; instrui o Flash a avaliar *expressão1*, depois *expressão2* e retornar o valor de *expressão2*. Esse operador é principalmente usado com o comando de loop `for`.

### **Exibidor**

Flash 4 ou posterior.

### Exemplo

O exemplo de código a seguir usa um operador vírgula:

```
var a=1, b=2, c=3;
```

Isso é o equivalente a escrever o seguinte:

```
var a=1;  
var b=2;  
var c=3;
```

## . (operador ponto)

### Sintaxe

*objeto.propriedade\_ou\_metodo*

*nome\_da\_instancia.variavel*

*nome\_da\_instancia.instancia\_filha.variavel*

### Argumentos

*objeto* Uma instância de um objeto. Alguns objetos requerem que as instâncias sejam criadas com o construtor do objeto. O objeto pode ser qualquer objeto ActionScript predefinido ou um objeto personalizado. Esse argumento está sempre à esquerda do operador ponto (.).

*propriedade\_ou\_metodo* O nome de uma propriedade ou de um método associado ao objeto. Todos os métodos e propriedades válidos dos objetos predefinidos estão listados nas tabelas de resumo Método e Propriedade de cada objeto. Esse argumento está sempre à direita do operador ponto (.).

*nome\_da\_instancia* O nome da instância de um clipe de filme.

*instancia\_filha* Uma instância de clipe de filme que seja filha do clipe de filme principal.

*variavel* Uma variável em um clipe de filme.

### Descrição

Operador; usado para navegar por hierarquias de clipes de filmes a fim de acessar clipes de filmes, variáveis ou propriedades filhos aninhados. O operador ponto é usado também para testar ou definir as propriedades de um objeto, executar um método de um objeto ou criar uma estrutura de dados.

### Exibidor

Flash 4 ou posterior.

### Consulte também

[ ] (operador de acesso de matriz), na página 203

### Exemplo

Este comando identifica o valor atual da variável `hairColor` pelo clipe de filme `person`:

```
person.hairColor
```

Isso equivale à seguinte sintaxe do Flash 4:

```
/person:hairColor
```

### Exemplo

O código a seguir ilustra como o operador ponto pode ser usado para criar uma estrutura de uma matriz:

```
account.name = "Gary Smith";  
account.address = "123 Main St ";  
account.city = "Any Town";  
account.state = "CA";  
account.zip = "12345";
```

## ?: (condicional)

### Sintaxe

*expressão1 ? expressão2 : expressão3*

### Argumentos

*expressão1* Uma expressão que é avaliada para um valor booleano, normalmente uma expressão de comparação.

*expressão2, expressão3* Valores de qualquer tipo.

### Descrição

Operador (condicional); instrui o Flash a avaliar *expressão1*, e retornar o valor de *expressão2* se *expressão1* for true; caso contrário, retorna o valor de *expressão3*.

### Exibidor

Flash 4 ou posterior.

## / (divisão)

### Sintaxe

*expressão1 / expressão2*

### Argumentos

*expressão* Qualquer número.

### Descrição

Operador (aritmético); divide *expressão1* por *expressão2*. Os argumentos da expressão e resultados da operação de divisão são tratados e expressados como números de ponto flutuante de precisão dupla.

**Exibidor**

Flash 4 ou posterior.

**Exemplo**

Este comando divide o número de ponto flutuante 22,0 por 7,0 e exibe o resultado na janela Saída:

```
trace(22,0 / 7,0);
```

O resultado é 3,1429, que é um número de ponto flutuante.

## // (delimitador de comentário)

**Sintaxe**

```
// comentário
```

**Argumentos**

*comentário* Texto que não faz parte do código e deve ser ignorado pelo interpretador.

**Descrição**

Comentário; indica o início de um comentário de script. Qualquer texto que apareça entre o delimitador de comentário // e o caractere de fim de linha é interpretado como comentário e ignorado pelo interpretador ActionScript.

**Exibidor**

Flash 1 ou posterior.

**Exemplo**

Este script usa barras delimitadoras de comentário para identificar a primeira, terceira, quinta e sétima linhas como comentários:

```
// define a posição X do clipe de filme ball  
ball = getProperty(ball._x);  
// define a posição Y do clipe de filme ball  
ball = getProperty(ball._y);  
// define a posição X do clipe de filme kitty  
kitty = getProperty(kitty._x);  
// define a posição Y do clipe de filme kitty  
kitty_y = getProperty(kitty._y);
```

**Consulte também**

/\* (delimitador de comentário), na página 202

## /\* (delimitador de comentário)

### Sintaxe

```
/* comentário */  
/*  
* comentário  
* comentário  
*/
```

### Argumentos

*comentário* Qualquer texto.

### Descrição

Comentário; indica uma ou mais linhas de comentários de script. Qualquer texto que apareça entre marca de começo do comentário `/*` e a marca de fechamento de comentário `*/` é interpretado como comentário e ignorado pelo interpretador ActionScript. Use a primeira sintaxe para identificar comentários com uma única linha e use a segunda sintaxe para identificar comentários com várias linhas sucessivas. Se não for usada a marca de fechamento `*/` quando essa forma de delimitador de comentário for usada, o compilador ActionScript retornará uma mensagem de erro.

### Exibidor

Flash 5 ou posterior.

### Consulte também

`//` (delimitador de comentário), na página 201

## /= (atribuição de divisão)

### Sintaxe

```
expressão1 /= expressão2
```

### Argumentos

*expressão1*, *expressão2* Inteiros, números de ponto flutuante ou seqüências de caracteres.

### Descrição

Operador (atribuição bit a bit) atribui a *expressão1* o valor de *expressão1* / *expressão2*.

#### Exibidor

Flash 4 ou posterior.

#### Exemplo

O exemplo a seguir ilustra o uso do operador /= com variáveis e números:

```
x /= y é o mesmo que x = x / y  
x = 10;  
y = 2;  
x /= y;  
// x agora contém o valor 5
```

## [] (operador de acesso de matriz)

#### Sintaxe

```
minhaMatriz[ "a0", "a1", ... "aN" ];  
object[ valor1, valor2, ... valorN ];
```

#### Argumentos

*minhaMatriz* O nome de uma matriz.

*a0, a1, ... aN* Elementos em uma matriz.

*valor1, 2, ... N* Nomes de propriedades.

#### Descrição

Operador; cria um novo objeto inicializando as propriedades especificadas nos argumentos ou inicializa uma nova matriz com os elementos (*a0*) especificados nos argumentos.

O objeto criado tem o objeto `Object` genérico como seu protótipo. Usar este operador é o mesmo que chamar `new Object` e preencher as propriedades com o operador de atribuição. O uso desse operador é uma alternativa ao uso do operador `new`, o que permite a criação rápida e conveniente de objetos.

#### Exibidor

Flash 4 ou posterior.

### Exemplo

Os exemplos de código a seguir são duas maneiras de criar um novo objeto Array vazio.

```
myArray = [];  
myArray = new Array();
```

A seguir, há o exemplo de uma matriz simples.

```
myArray = ["vermelho", "laranja", "amarelo", "verde", "azul",  
"púrpura"]  
myArray[0]="vermelho"  
myArray[1]="amarelo"  
myArray[2]="verde"  
myArray[3]="azul"  
myArray[4]="púrpura"
```

## ^(XOR bit a bit)

### Sintaxe

*expressão1* ^ *expressão2*

### Argumentos

*expressão1*, *expressão2* Qualquer número.

### Descrição

Operador (bit a bit); converte *expressão1* e *expressão2* em inteiros não assinados de 32 bits e retorna um 1 em cada posição de bit onde os bits correspondentes na *expressão1* ou *expressão2*, mas não em ambas, sejam 1.

### Exibidor

Flash 5 ou posterior.

### Exemplo

15 ^ 9 retorna 6  
(1111 ^ 1001 = 0110)

## ^= (atribuição XOR bit a bit)

### Sintaxe

*expressão1* ^= *expressão2*

### Argumentos

*expressão1*, *expressão2* Inteiros e variáveis.

### Descrição

Operador (atribuição composta); atribui a *expressão1* o valor de *expressão1* ^ *expressão2*.

### Exibidor

Flash 5 ou posterior.



### Exemplo

A seguir, há o exemplo de uma operação ^=.

```
// 15 decimal = 1111 binário  
x = 15;  
// 9 decimal = 1001 binário  
x ^= y;  
retorna  
x ^ y (0110 binário)
```

O exemplo a seguir ilustra o uso do operador ^= com variáveis e números:

```
x ^= y é o mesmo que x = x ^ y  
Se x = 15 e y = 9 então  
15 ^= 9 retorna 6
```

### Consulte também

^(XOR bit a bit), na página 204

## { } (inicializador de objeto)

### Sintaxe

```
objeto {nome1: valor1,  
        nome1: valor2,  
        ...  
        nomeN: valorN };
```

### Argumentos

*objeto* O objeto a ser criado.

*nome1, 2, ... N* O nome da propriedade.

*valor1, 2, ... N* O valor correspondente para cada propriedade *nome*.

### Descrição

Operador; cria um novo objeto e o inicializa com os pares de propriedades *nome* e *valor* especificados. O objeto criado tem o objeto `Object` genérico como seu protótipo. Usar este operador é o mesmo que chamar `new Object` e preencher os pares de propriedades com o operador de atribuição. O uso desse operador é uma alternativa ao uso do operador `new`, o que permite a criação rápida e conveniente de objetos.

### Exibidor

Flash 5 ou posterior.

### Exemplo

O código a seguir mostra como um objeto vazio pode ser criado com o operador inicializador de objeto e com `new Object`.

```
object = {};  
object = new Object();
```

O exemplo a seguir cria um objeto `account` inicializando as propriedades `name`, `address`, `city`, `state`, `zip` e `balance`.

```
account = { name: "John Smith",  
            address: "123 Main Street",  
            city: "Blossomville",  
            state: "California",  
            zip: "12345",  
            balance: "1000" };
```

O exemplo a seguir mostra como inicializadores de matriz e de objeto podem ser aninhados um no outro.

```
person = { name: "Peter Piper",  
           children: [ "Jack", "Jill", "Moe", ] };
```

O exemplo a seguir é outra maneira de usar as informações no exemplo anterior, com o mesmo resultado.

```
person = new Person();  
person.name = 'John Smith';  
person.children = new Array();  
person.children[0] = 'Jack';  
person.children[1] = 'Jill';  
person.children[2] = 'Moe';
```

### Consulte também

`[]` (operador de acesso de matriz), na página 203

`new`, na página 328

`Object` (objeto), na página 337

## | (OR bit a bit)

### Sintaxe

*expressão1* | *expressão2*

### Argumentos

*expressão1*, *expressão2* Qualquer número.

### Descrição

Operador (bit a bit); converte *expressão1* e *expressão2* em inteiros não assinados de 32 bits e retorna um 1 em cada posição de bit onde os bits correspondentes na *expressão1* ou *expressão2* sejam 1.

#### Exibidor

Flash 5 ou posterior.

#### Exemplo

A seguir, há o exemplo de uma operação OR bit a bit. Observe que 15 é 1111 binário.

```
// 15 decimal = 1111 binário  
x = 15;  
// 9 decimal = 1001 binário  
y = 9;  
// x | y = binário  
z = x | y;  
z = 15
```

A seguir, há outra maneira de expressar o exemplo anterior.

```
15 | 9 retorna 15  
(1111 | 1001 = 1111)
```

## || (OR)

#### Sintaxe

*expressão1* || *expressão2*

#### Argumentos

*expressão1*, *expressão2* Um valor ou expressão booleana que converte em um valor booleano.

#### Descrição

Operador (lógico); avalia *expressão1* e *expressão2*. O resultado é `true` se uma ou ambas as expressões forem avaliadas como `true`; o resultado será `false` somente se ambas as expressões forem avaliadas como `false`.

Com expressões não-booleanas, o operador lógico OR faz com que o Flash avalie a expressão da esquerda; se ela puder ser convertida em `true`, o resultado será `true`. Caso contrário, ele avaliará a expressão da direita e o resultado será o valor dessa expressão.

#### Exibidor

Flash 4 ou posterior.

#### Exemplo

O exemplo a seguir usa o operador `||` em um comando `if`:

```
want = true;  
need = true;  
love = false;  
if (want || need || love){  
    trace("dois em três não é ruim");  
}
```

## |= (atribuição OR bit a bit)

### Sintaxe

*expressão1* |= *expressão2*

### Argumentos

*expressão1*, *expressão2* Inteiros e variáveis.

### Descrição

Operador (atribuição); atribui a *expressão1* o valor de *expressão1* | *expressão2*.

### Exibidor

Flash 5 ou posterior.

### Exemplo

O exemplo a seguir ilustra o uso do operador |= com variáveis e números:

x |= y é o mesmo que x = x | y  
Se x = 15 e y = 9 então  
x |= 9 retorna 15

### Consulte também

| (OR bit a bit), na página 206

## ~ (NOT bit a bit)

### Sintaxe

~ *expressão*

### Argumentos

*expressão* Qualquer número.

### Descrição

Operador (bit a bit); converte a *expressão* em um inteiro não assinado de 32 bits, depois inverte os bits. Ou, resumindo, altera o sinal de um número e subtrai 1.

Uma operação NOT bit a bit altera o sinal de um número e subtrai 1.

### Exibidor

Flash 5 ou posterior.

### Exemplo

O exemplo a seguir é uma explicação numérica de uma operação NOT bit a bit executada em uma variável:

~a, retorna -1 se a = 0 e retorna -2 se a = 1, assim:  
~0=-1 e ~1=-2

## + (adição)

### Sintaxe

*expressão1* + *expressão2*

### Argumentos

*expressão1*, *expressão2* Inteiros, números, números de ponto flutuante ou seqüências de caracteres.

### Descrição

Operador; adiciona expressões numéricas ou concatena seqüências de caracteres. Se uma expressão for uma seqüência de caracteres, todas as outras expressões são convertidas em seqüências de caracteres e concatenadas.

Se ambas as expressões forem inteiros, a soma será um inteiro; se uma ou ambas as expressões forem números de ponto flutuante, a soma será um número de ponto flutuante.

### Exibidor

Flash 4, Flash 5 ou posterior. No Flash 5, + é um operador numérico ou concatenador de seqüências de caracteres, dependendo do tipo de dados do argumento. No Flash 4, + é somente um operador numérico. Os arquivos do Flash 4 trazidos para o ambiente de criação Flash 5 passam por um processo de conversão para manter a integridade dos tipos de dados. O primeiro exemplo a seguir ilustra o processo de conversão.

### Exemplo

O exemplo a seguir ilustra a conversão de um arquivo do Flash 4 que contém uma comparação do tipo numérica.

Arquivo do Flash 4:

`x + y`

Arquivo do Flash 5 convertido:

`Number(x) + Number(y)`

Este comando adiciona os inteiros 2 e 3 e exibe o inteiro resultante, 5, na janela Saída:

`trace (2 + 3);`

Este comando adiciona os números de ponto flutuante 2,5 e 3,25 e exibe o resultado, 5,7500, que é um número de ponto flutuante, na janela Saída:

`trace (2,5 + 3,25);`

Este comando concatena duas seqüências de caracteres e exibe o resultado, "hoje é meu aniversário", na janela Saída:

`"hoje é meu" + "aniversário"`

### Consulte também

`add`, na página 222

## += (atribuição de adição)

### Sintaxe

*expressão1* += *expressão2*

### Argumentos

*expressão1*, *expressão2* Inteiros, números de ponto flutuante ou seqüências de caracteres.

### Descrição

Operador (atribuição composta); atribui a *expressão1* o valor de *expressão1* + *expressão2*. Este operador também executa concatenação de seqüências de caracteres.

### Exibidor

Flash 4 ou posterior.

### Exemplo

O exemplo a seguir ilustra um uso numérico do operador +=:

$x += y$  é o mesmo que  $x = x + y$

Se  $x = 5$  e  $y = 10$  então

$x += 10$  retorna 15

Este exemplo ilustra o uso do operador += com uma expressão de seqüência de caracteres:

$x = \text{"Meu nome é"}$

$x += \text{"Maria"}$

O resultado do código acima é:

"Meu nome é Maria"

### Consulte também

+ (adição), na página 209

## < (menor que)

### Sintaxe

*expressão1* < *expressão2*

### Argumentos

*expressão1*, *expressão2* Números ou seqüências de caracteres.

### Descrição

Operador (comparação); compara duas expressões e determina se *expressão1* é menor que *expressão2* (true) ou se *expressão1* é maior ou igual a *expressão2* (false). Expressões de seqüências de caracteres são avaliadas e comparadas com base no número de caracteres na seqüência.



### Exibidor

Flash 4, Flash 5 ou posterior. No Flash 5, < é um operador de comparação capaz de manipular diversos tipos de dados. No Flash 4, < é um operador numérico. Os arquivos do Flash 4 trazidos para o ambiente de criação Flash 5 passam por um processo de conversão para manter a integridade dos tipos de dados. O primeiro exemplo a seguir ilustra o processo de conversão.

### Exemplo

O exemplo a seguir ilustra a conversão de um arquivo do Flash 4 que contém uma comparação do tipo numérica.

Arquivo do Flash 4:

`x < y`

Arquivo do Flash 5 convertido:

`Number(x) < Number(y)`

Os exemplos a seguir ilustram retornos `true` e `false` para números e seqüências de caracteres:

`3 < 10` or `"Al" < "Jack"` retorna `true`  
`10 < 3` or `"Jack" < "Al"` retorna `false`

## << (deslocamento para a esquerda bit a bit)

### Sintaxe

`expressão1 << expressão2`

### Argumentos

*expressão1* Um número, seqüência de caracteres ou expressão a ser deslocado para a esquerda.

*expressão2* Um número, seqüência de caracteres ou expressão que converte para um inteiro de 0 a 31.

### Descrição

Operador (bit a bit); converte *expressão1* e *expressão2* em inteiros de 32 bits e desloca todos os bits em *expressão1* para a esquerda de acordo com o número de casas especificado pelo inteiro que resulta da conversão de *expressão2*. As posições de bits esvaziadas como resultado dessa operação são preenchidas com 0. O deslocar um valor 1 posição para a esquerda é o equivalente a multiplicá-lo por 2.

### Exibidor

Flash 5 ou posterior.

#### Exemplo

O exemplo a seguir desloca o inteiro 1 dez bits para a esquerda.

$x = 1 \ll 10$

O resultado dessa operação é  $x = 1024$ . Isso é porque 1 decimal é igual a 1 binário, 1 binário deslocado 10 para a esquerda é 10000000000 binário e 10000000000 binário é 1024 decimal.

O exemplo a seguir desloca o inteiro 7 oito bits para a esquerda.

$x = 7 \ll 8$

O resultado dessa operação é  $x = 1792$ . Isso é porque 7 decimal é igual a 111 binário, 111 binário deslocado 8 para a esquerda é 11100000000 binário e 11100000000 binário é 1792 decimal.

#### Consulte também

$\gg=$  (deslocamento para a direita bit a bit e atribuição), na página 219

## $\ll=$ (deslocamento para a esquerda bit a bit e atribuição)

#### Sintaxe

*expressão1*  $\ll=$  *expressão2*

#### Argumentos

*expressão1* Um número, seqüência de caracteres ou expressão a ser deslocado para a esquerda.

*expressão2* Um número, seqüência de caracteres ou expressão que converte em um inteiro de 0 a 31.

#### Descrição

Operador (atribuição composta); este operador executa uma operação de deslocamento para esquerda bit a bit e armazena o conteúdo como um resultado em *expressão1*.

#### Exibidor

Flash 5 ou posterior.

#### Exemplo

As duas expressões a seguir são equivalentes.

$A \ll= B$

$A = (A \ll B)$

#### Consulte também

$\ll$  (deslocamento para a esquerda bit a bit), na página 211

$\gg=$  (deslocamento para a direita bit a bit e atribuição), na página 219



## <= (menor ou igual a)

### Sintaxe

*expressão1* <= *expressão2*

### Argumentos

*expressão1*, *expressão2* Números ou seqüências de caracteres.

### Descrição

Operador (comparação); compara duas expressões e determina se *expressão1* é menor ou igual a *expressão2* (true) ou se *expressão1* é maior que *expressão2* (false).

### Exibidor

Flash 4, Flash 5 ou posterior. No Flash 5, <= é um operador de comparação capaz de manipular diversos tipos de dados. No Flash 4, <= é um operador numérico. Os arquivos do Flash 4 trazidos para o ambiente de criação Flash 5 passam por um processo de conversão para manter a integridade dos tipos de dados. O primeiro exemplo a seguir ilustra o processo de conversão.

### Exemplo

O exemplo a seguir ilustra a conversão de um arquivo do Flash 4 que contém uma comparação do tipo numérica.

Arquivo do Flash 4:

`x <= y`

Arquivo do Flash 5 convertido:

`Number(x) <= Number(y)`

Os exemplos a seguir ilustram resultados true e false para números e seqüências de caracteres:

`5 <= 10` or `"Al" <= "Jack"` retorna true

`10 <= 5` or `"Jack" <= "Al"` retorna false

## <> (diferença)

### Sintaxe

*expressão1* <> *expressão2*

### Argumentos

*expressão1*, *expressão2* Números, seqüências de caracteres, booleanos, variáveis, objetos, matrizes ou funções.

#### Descrição

Operador (diferença); testa o oposto exato do operador `==`. Se *expressão1* for igual a *expressão2*, o resultado será `false`. Como com o operador `==`, a definição de *igual* depende dos tipos de dados comparados.

- Números, seqüências de caracteres e valores booleanos são comparados por valor.
- Variáveis, objetos, matrizes e funções são comparadas por referência.

Este operador está obsoleto no Flash 5 e os usuários são encorajados a usar o novo operador `!=`.

#### Exibidor

Flash 2 ou posterior.

#### Consulte também

`!=` (diferença), na página 191

## = (atribuição)

#### Sintaxe

*expressão1* = *expressão2*

#### Argumentos

*expressão1* Uma variável, elemento de uma matriz ou propriedade de um objeto.

*expressão2* Um valor de qualquer tipo.

#### Descrição

Operador (atribuição); atribui o tipo de *expressão2* (o argumento da direita) à variável, ao elemento da matriz ou à propriedade em *expressão1*.

#### Exibidor

Flash 4, Flash 5 ou posterior. No Flash 5, `=` é um operador de atribuição e o operador `==` é usado para avaliar igualdade. No Flash 4, `=` é um operador de igualdade numérico. Os arquivos do Flash 4 trazidos para o ambiente de criação do Flash 5 passam por um processo de conversão para manter a integridade dos tipos de dados. O primeiro exemplo a seguir ilustra o processo de conversão.

### Exemplo

O exemplo a seguir ilustra a conversão de um arquivo do Flash 4 que contém uma comparação do tipo numérica.

Arquivo do Flash 4:

```
x = y
```

Arquivo do Flash 5 convertido:

```
Number(x) == Number(y)
```

O exemplo a seguir usa o operador de atribuição para atribuir o tipo de dado numérico à variável x.

```
x = 5
```

O exemplo a seguir usa o operador de atribuição para atribuir o tipo de dado de sequência de caracteres à variável x.

```
x = "hello"
```

## -= (atribuição de negação)

### Sintaxe

*expressão1 -= expressão2*

### Argumentos

*expressão1*, *expressão2* Inteiros, números de ponto flutuante ou seqüências de caracteres.

### Descrição

Operador (atribuição composta); atribui a *expressão1* o valor de *expressão1* - *expressão2*.

### Exibidor

Flash 4 ou posterior.

### Exemplo

O exemplo a seguir ilustra o uso do operador -= com variáveis e números:

x -= y é o mesmo que x = x - y

Se x = 5 e y = 10 então

x -= 10 retorna -5

## == (igualdade)

### Sintaxe

*expressão1* == *expressão2*

### Argumentos

*expressão1*, *expressão2* Números, seqüências de caracteres, booleanos, variáveis, objetos, matrizes ou funções.

### Descrição

Operador (igualdade); testa a igualdade de duas expressões. O resultado será true se as expressões forem iguais.

A definição de *igual* depende do tipo de dado do argumento:

- Números, seqüências de caracteres e valores booleanos são comparados por valor e considerados iguais se tiverem o mesmo valor. Por exemplo, duas seqüências de caracteres são iguais se tiverem o mesmo número de caracteres.
- Variáveis, objetos, matrizes e funções são comparadas por referência. Duas variáveis são iguais se fizerem referência ao mesmo objeto, matriz ou função. Duas matrizes separadas nunca são consideradas iguais, mesmo que tenham o mesmo número de elementos.

### Exibidor

Flash 5 ou posterior.

### Exemplo

O exemplo a seguir usa o operador == com um comando if:

```
a = "David" , b = "David";  
if (a == b)  
trace("David é David");
```

## > (maior que)

### Sintaxe

*expressão1* > *expressão2*

### Argumentos

*expressão1*, *expressão2* Inteiros, números de ponto flutuante ou seqüências de caracteres.

### Descrição

Operador (comparação); compara duas expressões e determina se *expressão1* é maior que *expressão2* (true) ou se *expressão1* é menor ou igual a *expressão2* (false).

### Exibidor

Flash 4, Flash 5 ou posterior. No Flash 5, > é um operador de comparação capaz de manipular diversos tipos de dados. No Flash 4, > é um operador numérico. Os arquivos do Flash 4 trazidos para o ambiente de criação do Flash 5 passam por um processo de conversão para manter a integridade dos tipos de dados. O exemplo a seguir ilustra o processo de conversão.

### Exemplo

O exemplo a seguir ilustra a conversão de um arquivo do Flash 4 que contém uma comparação do tipo numérica.

Arquivo do Flash 4:

`x > y`

Arquivo do Flash 5 convertido:

`Number(x) > Number(y)`

## >= (maior ou igual a)

### Sintaxe

`expressão1 >= expressão2`

### Argumentos

`expressão1`, `expressão2` Sequências de caracteres, inteiros ou números de ponto flutuante.

### Descrição

Operador (comparação); compara duas expressões e determina se `expressão1` é maior ou igual a `expressão2` (true) ou se `expressão1` é menor que `expressão2` (false).

### Exibidor

Flash 4, Flash 5 ou posterior. No Flash 5, >= é um operador de comparação capaz de manipular diversos tipos de dados. No Flash 4, >= é um operador numérico. Os arquivos do Flash 4 trazidos para o ambiente de criação Flash 5 passam por um processo de conversão para manter a integridade dos tipos de dados. O exemplo a seguir ilustra o processo de conversão.

### Exemplo

O exemplo a seguir ilustra a conversão de um arquivo do Flash 4 que contém uma comparação do tipo numérica.

Arquivo do Flash 4:

`x >= y`

Arquivo do Flash 5 convertido:

`Number(x) >= Number(y)`



## » (deslocamento para a direita bit a bit)

### Sintaxe

*expressão1* >> *expressão2*

### Argumentos

*expressão1* Um número, sequência de caracteres ou expressão a ser deslocado para a direita.

*expressão2* Um número, sequência de caracteres ou expressão que converte em um inteiro de 0 a 31.

### Descrição

Operador (bit a bit); converte *expressão1* e *expressão2* em inteiros de 32 bits e desloca todos os bits em *expressão1* para a direita de acordo com o número de casas especificado pelo inteiro que resulta da conversão de *expressão2*. Bits deslocados para fora para a direita são descartados. Para manter o sinal da *expressão* original, os bits na esquerda são preenchidos com 0, se o bit mais significativo (o bit mais à esquerda) de *expressão1* for 0, e preenchidos com 1, se o bit mais significativo for 1. O deslocamento de um valor 1 posição para a direita equivale a dividir por 2 e descartar o resto.

### Exibidor

Flash 5 ou posterior.

### Exemplo

O exemplo a seguir converte 65535 em um inteiro de 32 bits e o desloca oito bits para a direita.

x = 65535 >> 8

O resultado da operação acima é:

x = 255

Isso é porque 65535 decimal é igual a 1111111111111111 binário (*dezesseis 1*), 1111111111111111 binário deslocado 8 bits para a direita é 11111111 binário e 11111111 binário é 255 decimal. O bit mais significativo é 0, pois os inteiros são de 32 bits, portanto o bit de preenchimento é 0.



O exemplo a seguir converte -1 em um inteiro de 32 bits e o desloca um bit para a direita.

```
x = -1 >> 1
```

O resultado da operação acima é:

```
x = -1
```

Isso é porque -1 decimal é igual a 11111111111111111111111111111111 binário (trinta e dois 1), o deslocamento um bit para a direita faz com que o bit menos significativo (bit mais à direita) seja descartado e o bit mais significativo seja preenchido com 1. O resultado é 11111111111111111111111111111111 (*trinta e dois 1*) binário, que representa o inteiro de 32 bits -1.

#### Consulte também

>>= (deslocamento para a direita bit a bit e atribuição), na página 219

## >>= (deslocamento para a direita bit a bit e atribuição)

#### Sintaxe

*expressão1* =>> *expressão2*

#### Argumentos

*expressão1* Um número, sequência de caracteres ou expressão a ser deslocado para a esquerda.

*expressão2* Um número, sequência de caracteres ou expressão que converte para um inteiro de 0 a 31.

#### Descrição

Operador (atribuição composta); este operador executa uma operação de deslocamento para direita bit a bit e armazena o conteúdo como um resultado em *expressão1*.

#### Exibidor

Flash 5 ou posterior.

### Exemplo

As duas expressões a seguir são equivalentes.

```
A >>= B  
A = (A >> B)
```

O código comentado a seguir usa o operador bit a bit >>=. Ele também é um exemplo do uso de todos os operadores bit a bit.

```
function convertToBinary(number)  
{  
  var result = "";  
  for (var i=0; i<32; i++) {  
    // Extrai o bit menos significativo pelo uso de AND bit a bit  
    var lsb = number & 1;  
    // Adiciona esse bit a nossa seqüência de caracteres de resultado  
    result = (lsb ? "1" : "0") + result;  
    // Desloca o número um bit para a direita para ver próximo bit  
    number >>= 1;  
    return result;  
  }  
  convertToBinary(479)  
  //Retorna a seqüência de caracteres  
  00000000000000000000000011101111  
  //A seqüência de caracteres acima é a representação binária  
  do número decimal 479.
```

### Consulte também

<< (deslocamento para a esquerda bit a bit), na página 211

## >>> (deslocamento para a direita não assinado bit a bit)

### Sintaxe

*expressão1* >>> *expressão2*

### Argumentos

*expressão1* Um número, seqüência de caracteres ou expressão a ser deslocado para a direita.

*expressão2* Um número, seqüência de caracteres ou expressão que converte em um inteiro de 0 a 31.

### Descrição

Operador (bit a bit); o mesmo que o operador de deslocamento para a direita bit a bit (>>), exceto que ele não mantém o sinal da *expressão* original, pois os bits na esquerda sempre são preenchidos com 0.



#### Exibidor

Flash 5 ou posterior.

#### Exemplo

O exemplo a seguir converte -1 em um inteiro de 32 bits e o desloca um bit para a direita.

```
x = -1 >>> 1
```

O resultado da operação acima é:

```
x = 2147483647
```

Isso é porque -1 decimal é 11111111111111111111111111111111 binário (*trinta e dois 1*) e, quando é deslocado um bit (não assinado) para a direita, o bit menos significativo (mais à direita) é descartado e o bit mais significativo (mais à esquerda) é preenchido com um 0. O resultado é:

01111111111111111111111111111111 binário,

que representa o inteiro de 32 bits 2147483647.

#### Consulte também

>>= (deslocamento para a direita bit a bit e atribuição), na página 219

## >>>= (deslocamento para a direita não assinado bit a bit e atribuição)

#### Sintaxe

*expressão1 >>> = expressão2*

#### Argumentos

*expressão1* Um número, sequência de caracteres ou expressão a ser deslocado para a esquerda.

*expressão2* Um número, sequência de caracteres ou expressão que converte em um inteiro de 0 a 31.

#### Descrição

Operador (atribuição composta); executa uma operação de deslocamento para direita bit a bit não assinada e armazena o conteúdo como um resultado em *expressão1*.

#### Exibidor

Flash 5 ou posterior.

#### Exemplo

As duas expressões a seguir são equivalentes.

A >>>= B

A = (A >>> B)

#### Consulte também

>>> (deslocamento para a direita não assinado bit a bit), na página 220

>>= (deslocamento para a direita bit a bit e atribuição), na página 219

## add

#### Sintaxe

*seq\_caract1* add *seq\_caract2*

#### Argumentos

*seq\_ênciã de caracteres1,2* Qualquer seq\_ênciã de caracteres.

#### Descrição

Operador; concatena duas ou mais seq\_ênciãs de caracteres. O operador `add` substitui o operador `&` do Flash 4; os arquivos do Flash 4 que usam o operador `&` são convertidos automaticamente para usar o operador `add` para concatenação de seq\_ênciãs de caracteres quando trazidos para o ambiente de criação Flash 5. Entretanto, o operador `add` é reprovado no Flash 5 e recomenda-se o uso do operador `+` ao criar conteúdo para o Flash 5 Player. Use o operador `add` para concatenar seq\_ênciãs de caracteres se estiver criando conteúdo para o Flash 4 ou versões anteriores do Exibidor.

#### Exibidor

Flash 4 ou posterior.

#### Consulte também

`+` (adição), na página 209

## \_alpha

#### Sintaxe

*nome\_da\_instância*.\_alpha

*nome\_da\_instância*.\_alpha = *valor*;

#### Argumentos

*nome\_da\_instância* O nome da instância de um clipe de filme.

*valor* Um número de 0 a 100 que especifique a transparência alpha.

### Descrição

Propriedade; define ou recupera a transparência alpha (*valor*) do clipe de filme. Os valores válidos vão de 0 (totalmente transparente) a 100 (totalmente opaco). Os objetos em um clipe de filme com `_alpha` definida como 0 são ativos, apesar de serem invisíveis. Por exemplo, um botão em um clipe de filme com propriedade `_alpha` definida como 0 ainda pode ser clicado.

### Exibidor

Flash 4 ou posterior.

### Exemplo

Os comandos a seguir definem como 30% a propriedade `_alpha` de um clipe de filme chamado `star` quando o botão é clicado.

```
on(release) {  
    setProperty(star._alpha = 30);  
}
```

ou

```
on(release) {  
    star._alpha = 30;  
}
```

## and

### Sintaxe

*condição1* and *condição2*

### Argumentos

*condição1*, *condição2* Condições ou expressões que avaliam como `true` ou `false`.

### Descrição

Operador; executa uma operação lógica AND no Flash 4 Player. Se ambas as expressões forem avaliadas como `true`, toda a expressão é `true`.

### Exibidor

Flash 4 ou posterior. Este operador foi reprovado no Flash 5 e os usuários são encorajados a usar o novo operador `&&`.

### Consulte também

`&&` (AND de curto-circuito), na página 194

## Array (objeto)

O objeto Array permite acessar e manipular matrizes. Uma matriz é um objeto cujas propriedades são identificadas por números que representam suas posições na matriz. Às vezes, esse número é chamado de índice. Todas as matrizes são de base zero, o que significa que o primeiro elemento na matriz é [0], o segundo é [1], etc. No exemplo a seguir, myArray contém os meses do ano, identificados por números.

```
myArray[0] = "Janeiro"
myArray[1] = "Fevereiro"
myArray[2] = "Março"
myArray[3] = "Abril"
```

Para criar um objeto Array, use o construtor `new Array`. Para acessar os elementos de uma matriz, use o operador de acesso de matriz `[ ]`.

### Resumo de métodos do objeto Array

Método	Descrição
concat	Concatena os argumentos e os retorna como uma nova matriz.
join	Reúne todos os elementos de uma matriz em uma sequência de caracteres.
pop	Remove o último elemento de uma matriz e retorna seu valor.
push	Adiciona um ou mais elementos ao fim de uma matriz e retorna o novo tamanho da matriz.
reverse	Inverte a direção de uma matriz.
shift	Remove o primeiro elemento de uma matriz e retorna seu valor.
slice	Extrai uma seção de uma matriz e a retorna como uma nova matriz.
sort	Classifica uma matriz no local.
splice	Adiciona e/ou remove elementos de uma matriz.
toString	Retorna um valor de sequência de caracteres que representa os elementos no objeto Array.
unshift	Adiciona um ou mais elementos ao início de uma matriz e retorna o novo tamanho da matriz.



## Resumo de propriedades do objeto Array

Propriedade	Descrição
length	Retorna o tamanho da matriz.

## Construtor do objeto Array

### Sintaxe

```
new Array();
new Array(tamanho);
new Array(elemento0, elemento1, elemento2,...elementoN);
```

### Argumentos

*tamanho* Um inteiro que especifica o número de elementos na matriz. No caso de elementos não contíguos, o tamanho especifica o número do índice do último elemento na matriz mais 1. Para obter mais informações, consulte a propriedade `Array.length`.

*elemento0...elementoN* Uma lista de dois ou mais valores arbitrários. Os valores podem ser números, nomes ou outros elementos especificados em uma matriz. O primeiro elemento em uma matriz sempre tem o índice, ou posição, 0.

### Descrição

Construtor; permite acessar e manipular elementos em uma matriz. Matrizes têm base zero e os elementos são indexados por seus números ordinais.

Se você não especificar um argumento, será criada uma matriz com tamanho zero.

### Exibidor

Flash 5 ou posterior.

### Exemplo

O exemplo a seguir cria um novo objeto Array com tamanho inicial 0.

```
myArray = new Array();
```

O exemplo a seguir cria o novo objeto Array A-Team, com tamanho inicial 4.

```
A-Team = new Array("Jody", "Mary", "Marcelle", "Judy");
```

Os elementos iniciais da matriz A-Team são estes:

```
myArray[0] = "Jody"
myArray[1] = "Mary"
myArray[2] = "Marcelle"
myArray[3] = "Judy"
```

### Consulte também

`Array.length`, na página 227

## Array.concat

### Sintaxe

`myArray.concat(valor0, valor1, ... valorN);`

### Argumentos

`valor0, ... valorN` Números, elementos ou seqüências de caracteres a serem concatenados em uma nova matriz.

### Descrição

Método; concatena os elementos especificados nos argumentos, se houver, cria e retorna uma nova matriz. Se os argumentos especificarem uma matriz, os elementos dessa matriz serão concatenados, em vez da própria matriz.

### Exibidor

Flash 5 ou posterior.

### Exemplo

O código a seguir concatena duas matrizes:

```
alpha = new Array("a","b","c");  
numeric = new Array(1,2,3);  
alphaNumeric=alpha.concat(numeric); // cria a matriz  
["a","b","c",1,2,3]
```

O código a seguir concatena três matrizes:

```
num1=[1,3,5];  
num2=[2,4,6];  
num3=[7,8,9];  
nums=num1.concat(num2,num3) // cria a matriz [1,3,5,2,4,6,7,8,9]
```

## Array.join

### Sintaxe

`myArray.join();`

`myArray.join(separador);`

### Argumentos

`separador` Um caractere ou uma seqüência de caracteres que separa elementos da matriz na seqüência de caracteres retornada. Se você omitir esse argumento, será usada uma vírgula como separador padrão.

### Descrição

Método; converte os elementos de uma matriz em seqüências de caracteres, concatena esses elementos, insere o separador especificado entre eles e retorna a seqüência de caracteres resultante.

#### Exibidor

Flash 5 ou posterior.

#### Exemplo

O exemplo a seguir cria uma matriz com três elementos. Depois, reúne a matriz três vezes: com o separador padrão, depois com uma vírgula e um espaço e, então, com um sinal de adição.

```
a = new Array("Terra", "Lua", "Sol")
// atribui "Terra, Lua, Sol" a myVar1
myVar1=a.join();
// atribui "Terra, Lua, Sol" a myVar2
myVar2=a.join(", ");
// atribui "Terra + Lua + Sol" a myVar3
myVar3=a.join(" + ");
```

## Array.length

#### Sintaxe

*myArray.length*;

#### Argumentos

Nenhum.

#### Descrição

Propriedade; contém o tamanho da matriz. Essa propriedade é atualizada automaticamente quando são adicionados novos elementos à matriz. Durante a atribuição `myArray[índice] = valor`; se `índice` for um número e `índice+1` for maior do que a propriedade `length`, a propriedade `length` será atualizada para `índice + 1`.

#### Exibidor

Flash 5 ou posterior.

#### Exemplo

O código a seguir explica como a propriedade `length` é atualizada.

```
// tamanho inicial é 0
myArray = new Array();
//myArray.length é atualizada para 1
myArray[0] = 'a'
//myArray.length é atualizada para 2
myArray[1] = 'b'
//myArray.length é atualizada para 10
myArray[9] = 'c'
```

## Array.pop

### Sintaxe

```
myArray.pop();
```

### Argumentos

Nenhum.

### Descrição

Método; remove o último elemento de uma matriz e retorna o valor desse elemento.

### Exibidor

Flash 5 ou posterior.

### Exemplo

O código a seguir cria a matriz `myPets`, com quatro elementos, depois remove seu último elemento.

```
myPets = ["gato", "cachorro", "pássaro", "peixe"];  
popped = myPets.pop();
```

## Array.push

### Sintaxe

```
myArray.push(valor,...);
```

### Argumentos

*valor* Um ou mais valores a serem anexados à matriz.

### Descrição

Método; adiciona um ou mais elementos ao fim de uma matriz e retorna o novo tamanho da matriz.

### Exibidor

Flash 5 ou posterior.

### Exemplo

O código a seguir cria a matriz `myPets`, com dois elementos, depois adiciona dois elementos a ela. Depois da execução do código, `pushed` contém 4.

```
myPets = ["gato", "cachorro"];  
pushed = myPets.push("pássaro", "peixe")
```



## Array.reverse

### Sintaxe

`myArray.reverse();`

### Argumentos

Nenhum.

### Descrição

Método; reverte a matriz no local.

### Exibidor

Flash 5 ou posterior.

### Exemplo

O exemplo a seguir mostra um uso do método `Array.reverse`.

```
var numbers = [1, 2, 3, 4, 5, 6];  
trace(numbers.join())  
  numbers.reverse()  
  trace(numbers.join())
```

Saída:

```
1,2,3,4,5,6  
6,5,4,3,2,1
```

## Array.shift

### Sintaxe

`myArray.shift();`

### Argumentos

Nenhum.

### Descrição

Método; remove o primeiro elemento de uma matriz e retorna esse elemento.

### Exibidor

Flash 5 ou posterior.

### Exemplo

O código a seguir cria a matriz `myPets` e remove o primeiro elemento dela:

```
myPets = ["gato", "cachorro", "pássaro", "peixe"];  
shifted = myPets.shift();
```

O valor de retorno é `gato`.

### Consulte também

`Array.pop`, na página 228

`Array.unshift`, na página 233

## Array.slice

### Sintaxe

```
myArray.slice(início, fim);
```

### Argumentos

*início* Um número que especifica o índice do ponto inicial do segmento.

Se *início* for um número negativo, o ponto inicial começará no fim da matriz, onde -1 é o último elemento.

*fim* Um número que especifica o índice do ponto final do segmento. Se esse argumento for omitido, o segmento incluirá todos os elementos do ponto inicial ao fim da matriz. Se *fim* for um número negativo, o ponto final será especificado a partir do fim da matriz, onde -1 é o último elemento.

### Descrição

Método; extrai um segmento ou subsequência de caracteres da matriz e o retorna como uma nova matriz, sem modificar a matriz original. A matriz retornada inclui o elemento *inicial* e todos os elementos até, mas não incluindo, o elemento *final*.

### Exibidor

Flash 5 ou posterior.

## Array.sort

### Sintaxe

```
myArray.sort();  
myArray.sort(func_ordem);
```

### Argumentos

*func\_ordem* Uma função de comparação opcional usada para determinar a ordem de classificação. Dados os argumentos A e B, a função de ordenação deve executar uma classificação desta maneira:

- -1 se A aparecer antes de B na sequência classificada
- 0 se A = B
- 1 se A aparecer depois de B na sequência classificada

### Descrição

Método; classifica a matriz no local, sem fazer uma cópia. Se você omitir o argumento *orderfunc*, o Flash classificará os elementos no local com o operador de comparação <.

### Exibidor

Flash 5 ou posterior.

### Exemplo

O exemplo a seguir usa `Array.sort` sem especificar o argumento `orderfunc`.

```
var fruits = ["oranges", "apples", "strawberries",
             "pineapples", "cherries"];
trace(fruits.join())
fruits.sort()
trace(fruits.join())
```

Saída:

```
oranges,apples,strawberries,pineapples,cherries
apples,cherries,oranges,pineapples,strawberries
```

O exemplo a seguir usa `array.sort` com uma função de ordenação especificada.

```
var passwords = [
    "gary:foo",
    "mike:bar",
    "john:snafu",
    "steve:yuck",
    "daniel:1234"
];
function order (a, b) {
    // Entradas a serem classificadas estão na forma
    // nome:senha
    // Classifica usando somente a parte do nome da
    // entrada como chave.
    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());
}
```

Saída:

```
daniel:1234
gary:foo
john:snafu
mike:bar
steve:yuck
```

## Array.splice

### Sintaxe

`myArray.splice(início, contagExcluir, valor0, valor1...valorN);`

### Argumentos

*início* O índice do elemento na matriz onde a inserção e/ou exclusão começa.

*deleteCount* O número de elementos a serem excluídos. Esse número inclui o elemento especificado no argumento *início*. Se não houver valores especificados para *deleteCount*, o método exclui todos os valores a partir do elemento *início* até o último elemento na matriz.

*valor* Zero ou mais valores a serem inseridos na matriz no ponto de inserção especificado no argumento *início*. Esse argumento é opcional.

### Descrição

Método; adiciona e/ou remove elementos de uma matriz. Este método modifica a própria matriz sem fazer uma cópia.

### Exibidor

Flash 5 ou posterior.

## Array.toString

### Sintaxe

`myArray.toString();`

### Argumentos

Nenhum.

### Descrição

Método; retorna um valor de seqüência de caracteres que representa os elementos no objeto Array especificado. Todos os elementos da matriz, iniciando pelo índice 0 e terminando no índice `myArray.length-1`, é convertido em uma seqüência de caracteres concatenados e separados por vírgulas.

### Exibidor

Flash 5 ou posterior.

### Exemplo

O exemplo a seguir cria myArray e a converte em uma sequência de caracteres.

```
myArray = new Array();  
myArray[0] = 1;  
myArray[1] = 2;  
myArray[2] = 3;  
myArray[3] = 4;  
myArray[4] = 5;  
  
trace(myArray.toString())
```

Saída:

1,2,3,4,5

## Array.unshift

### Sintaxe

*myArray.unshift(valor1, valor2, ... valorN);*

### Argumentos

*valor1, ... valorN* Um ou mais números, elementos ou variáveis a serem inseridos no início da matriz.

### Descrição

Método; adiciona um ou mais elementos ao início de uma matriz e retorna o novo tamanho da matriz.

### Exibidor

Flash 5 ou posterior.

## Boolean (função)

### Sintaxe

*Boolean(expressão);*

### Argumentos

*expressão* A variável, o número ou a sequência de caracteres a ser convertida em um booleano.

### Descrição

Função; converte o argumento especificado em um booleano e retorna o valor de Boolean.

### Exibidor

Flash 5 ou posterior.

## Boolean (objeto)

O objeto Boolean é um objeto envoltório simples que funciona da mesma maneira que o objeto Boolean JavaScript padrão. Use o objeto Boolean para recuperar o tipo de dados primitivo ou a representação da sequência de caracteres do objeto Boolean.

### Resumo de métodos do objeto Boolean

Método	Descrição
<code>toString</code>	Retorna a representação da sequência de caracteres ( <code>true</code> ) ou ( <code>false</code> ) do objeto Boolean.
<code>valueOf</code>	Retorna o tipo de valor primitivo do objeto Boolean especificado.

### Construtor do objeto Boolean

#### Sintaxe

```
new Boolean();
```

```
new Boolean(x);
```

#### Argumentos

*x* Número, sequência de caracteres, booleano, objeto, clipe de filme ou outra expressão. Esse argumento é opcional.

#### Descrição

Construtor; cria uma instância do objeto Boolean. Se você omitir o argumento *x*, o objeto Boolean é inicializado com um valor `false`. Se você especificar *x*, o método avaliará o argumento e retornará o resultado como valor Boolean, de acordo com as seguintes regras de cálculo.

- Se *x* for um número, a função retornará `true` se *x* for diferente de 0 ou `false` se *x* for qualquer outro número.
- Se *x* for um booleano, a função retornará *x*.
- Se *x* for um objeto ou clipe de filme, a função retornará `true` se *x* for diferente de `null`; caso contrário, a função retornará `false`.
- Se *x* for uma sequência de caracteres, a função retornará `true` se `Number(x)` for diferente de 0; caso contrário, a função retornará `false`.

**Observação:** Para manter a compatibilidade com o Flash 4, a manipulação de seqüências de caracteres pelo objeto Boolean não usa o padrão ECMA-262.

#### Exibidor

Flash 5 ou posterior.

## Boolean.toString

### Sintaxe

`Boolean.toString();`

### Argumentos

Nenhum.

### Descrição

Método; retorna a representação da sequência de caracteres, `true` ou `false`, do objeto Boolean.

### Exibidor

Flash 5 ou posterior.

## Boolean.valueOf

### Sintaxe

`Boolean.valueOf();`

### Argumentos

Nenhum.

### Descrição

Método; retorna o tipo de valor primitivo do objeto Boolean especificado e converte o objeto envoltório Boolean em seu tipo de valor primitivo.

### Exibidor

Flash 5 ou posterior.

## break

### Sintaxe

`break;`

### Argumentos

Nenhum.

### Descrição

Ação; é exibida em um loop (`for`, `for...in`, `do...while` ou `while`). A ação `break` instrui o Flash a ignorar o resto do corpo do loop, parar a ação de loop e executar o comando após o comando loop. Use a ação `break` para interromper uma série de loops aninhados.

#### Exibidor

Flash 4 ou posterior.

#### Exemplo

O exemplo a seguir usa a ação `break` para sair de um loop infinito.

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

## call

#### Sintaxe

`call(quadro);`

#### Argumentos

*quadro* O nome ou número do quadro a ser chamado para o contexto do script.

#### Descrição

Ação; alterna o contexto do script atual para o script anexado no quadro que está sendo chamado. Não haverá variáveis locais, uma vez que a execução do script é concluída.

#### Exibidor

Flash 4 ou posterior. Essa ação está obsoleta no Flash 5 e recomenda-se que você use a ação `function`.

#### Consulte também

`function`, na página 272

## chr

#### Sintaxe

`chr(número);`

#### Argumentos

*número* O número do código ASCII a ser convertido em caractere.

#### Descrição

Função de sequência de caracteres; converte código ASCII em caracteres.





### Exibidor

Flash 4 ou posterior. Esta função foi reprovada no Flash 5; recomenda-se o uso do método `String.fromCharCode`.

### Exemplo

O exemplo a seguir converte o número 65 na letra "A".

```
chr(65) = "A"
```

### Consulte também

`String.fromCharCode`, na página 375

## Color (objeto)

O objeto `Color` permite definir e recuperar o valor de cor RGB e a transformação de cor de cliques de filmes. Somente o Flash 5 e as versões mais recentes do Flash Player oferecem suporte ao objeto `Color`.

É necessário usar o construtor `new Color()` para criar uma instância do objeto `Color` antes de chamar os métodos do objeto `Color`.

### Resumo de métodos do objeto Color

Método	Descrição
<code>getRGB</code>	Retorna o valor RGB definido pela última chamada <code>setRGB</code> .
<code>getTransform</code>	Retorna a informação de transformação definida pela última chamada <code>setTransform</code> .
<code>setRGB</code>	Define a representação hexadecimal do valor RGB de um objeto <code>Color</code> .
<code>setTransform</code>	Define a transformação de cor de um objeto <code>Color</code> .

### Construtor do objeto Color

#### Sintaxe

```
new Color(destino);
```

#### Argumentos

*destino* O nome do clipe de filme ao qual é aplicada a nova cor.

#### Descrição

Construtor; cria um objeto `Color` para o clipe de filme especificado pelo argumento *destino*.

**Exibidor**

Flash 5 ou posterior.

**Exemplo**

O exemplo a seguir cria um objeto new Color chamado myColor para o filme myMovie.

```
myColor = new Color(myMovie);
```

## Color.getRGB

**Sintaxe**

```
myColor.getRGB();
```

**Argumentos**

Nenhum.

**Descrição**

Método; retorna os valores numéricos definidos pela última chamada setRGB.

**Exibidor**

Flash 5 ou posterior.

**Exemplo**

O código a seguir recupera o valor RGB como sequência de caracteres hexadecimal:

```
value = (getRGB()).toString(16);
```

**Consulte também**

Color.setRGB, na página 239

## Color.getTransform

**Sintaxe**

```
myColor.getTransform();
```

**Argumentos**

Nenhum.

**Descrição**

Método; retorna os valores de transformação definidos pela última chamada setTransform.

**Exibidor**

Flash 5 ou posterior.

**Consulte também**

Color.setTransform, na página 239

## Color.setRGB

### Sintaxe

`myColor.setRGB(0xRRGGBB);`

### Argumentos

`0xRRGGBB` Cor hexadecimal ou RGB a ser definida. *RR*, *GG* e *BB* consistem cada um em dois dígitos hexadecimais que especifiquem o deslocamento de cada componente de cor.

### Descrição

Método; especifica uma cor RGB para o objeto Color. Quando este método é chamado, todas as definições anteriores são substituídas pelo método `setTransform`.

### Exibidor

Flash 5 ou posterior.

### Exemplo

O exemplo a seguir define o valor de cor RGB do clipe de filme `myMovie`.

```
myColor = new Color(myMovie);  
myColor.setRGB(0x993366);
```

### Consulte também

`Color.setTransform`, na página 239

## Color.setTransform

### Sintaxe

`myColor.setTransform(objetoTransformCor);`

### Argumentos

`objetoTransformCor` Um objeto criado com o construtor do objeto `Object` genérico, especificando valores de transformação de cor para parâmetros. O objeto de transformação de cor deve ter os parâmetros *ra*, *rb*, *ga*, *gb*, *ba*, *bb*, *aa*, *ab*, que são explicados abaixo.

### Descrição

Método; define informações de transformação de cor de um objeto Color.

O argumento *objetoTransformCor* é um objeto que você cria com o objeto genérico Object, com parâmetros que especifiquem os valores de porcentagem e de deslocamento dos componentes vermelho, verde, azul e alpha (transparência) de uma cor, inseridos no formato *0xRRGGBBAA*.

Os parâmetros de um objeto de transformação de uma cor são definidos desta maneira:

- *ra* é a porcentagem do componente vermelho (-100 a 100).
- *rb* é o deslocamento do componente vermelho (-255 a 255).
- *ga* é a porcentagem do componente verde (-100 a 100).
- *gb* é o deslocamento do componente verde (-255 a 255).
- *ba* é a porcentagem do componente azul (-100 a 100).
- *bb* é o deslocamento do componente azul (-255 a 255).
- *aa* é a porcentagem de alpha (-100 a 100).
- *ab* é o deslocamento de alpha (-255 a 255).

Você cria um objeto de transformação de cor desta maneira:

```
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;
```

Você também poderia usar esta sintaxe:

```
myColorTransform = { ra: '50', rb: '244', ga: '40', gb: '112',  
ba: '12', bb: '90', aa: '40', ab: '70' }
```

### Exibidor

Flash 5 ou posterior.



### Exemplo

O exemplo a seguir mostra o processo de criação de um objeto new Color para um filme de destino, criando um objeto de transformação de cor com os parâmetros definidos acima por meio do construtor Object e passando o objeto de transformação de cor para um objeto Color por meio do método setTransform.

```
//Cria um objeto de cor chamado myColor para o filme de destino
myMovie
myColor = new Color(myMovie);
//Cria um objeto de transformação de cor chamado myColorTransform
usando o objeto genérico Object
myColorTransform = new Object;
// Define os valores de myColorTransform
myColorTransform = { ra: '50', rb: '244', ga: '40', gb: '112',
ba: '12', bb: '90', aa: '40', ab: '70'}

//Associa o objeto de transformação de cor ao objeto Color criado
para myMovie
myColor.setTransform(myColorTransform);
```

## continue

### Sintaxe

continue;

### Argumentos

Nenhum.

### Descrição

Ação; é exibida em diversos tipos de comandos de loop.

Em um loop while, continue faz o Flash ignorar o resto do corpo do loop e saltar para o início do loop, onde a condição é testada.

Em um loop do...while, continue faz o Flash ignorar o resto do corpo do loop e saltar para o fim do loop, onde a condição é testada.

Em um loop for, continue faz o Flash ignorar o resto do corpo do loop e saltar para a avaliação da pós-expressão for do loop.

Em um loop for...in, continue faz o Flash ignorar o resto do corpo do loop e voltar ao início do loop, onde o próximo valor na enumeração é processado.

### Exibidor

Flash 4 ou posterior.

### Consulte também

do... while, na página 262

for, na página 268

for...in, na página 269

while, na página 389

## **\_currentframe**

### **Sintaxe**

*nome\_da\_instancia.\_currentframe*

### **Argumentos**

*nome\_da\_instancia* O nome da instância de um clipe de filme.

### **Descrição**

Propriedade (somente leitura); retorna o número do quadro em que a reprodução está na Linha de Tempo no momento.

### **Exibidor**

Flash 4 ou posterior.

### **Exemplo**

O exemplo a seguir usa `_currentframe` para direcionar um filme para cinco quadros adiante do quadro que contém a ação:

```
gotoAndStop(_currentframe + 5);
```

## **Date (objeto)**

O objeto `Date` permite recuperar os valores de data e hora relativos à hora universal (Hora de Greenwich, agora chamada de Hora Coordenada Universal) ou relativos ao sistema operacional em que o Flash Player está sendo executado. Para chamar os métodos do objeto `Date`, você deve primeiro criar uma instância do objeto `Date` com o construtor.

O objeto `Date` requer o Flash 5 Player.

Os métodos do objeto `Date` não são estáticos, mas se aplicam somente à instância individual do objeto `Date` especificado quando o método é chamado.



## Resumo de métodos do objeto Date

Método	Descrição
getDate	Retorna o dia do mês do objeto Date especificado, de acordo com a hora local.
getDay	Retorna o dia do mês do objeto Date especificado, de acordo com a hora local.
getFullYear	Retorna o ano com quatro dígitos do objeto Date especificado, de acordo com a hora local.
getHours	Retorna a hora do objeto Date especificado, de acordo com a hora local.
getMilliseconds	Retorna os milissegundos do objeto Date especificado, de acordo com a hora local.
getMinutes	Retorna os minutos do objeto Date especificado, de acordo com a hora local.
getMonth	Retorna o mês do objeto Date especificado, de acordo com a hora local.
getSeconds	Retorna os segundos do objeto Date especificado, de acordo com a hora local.
getTime	Retorna o número de milissegundos desde a meia-noite de 1º de janeiro de 1970, hora universal, do objeto Date especificado.
getTimezoneOffset	Retorna a diferença, em minutos, entre o a hora local do computador e a hora universal.
getUTCDate	Retorna o dia (data) do mês do objeto Date especificado, de acordo com a hora universal.
getUTCDay	Retorna o dia da semana do objeto Date especificado, de acordo com a hora universal.
getUTCFullYear	Retorna o ano com quatro dígitos do objeto Date especificado, de acordo com a hora universal.
getUTCHours	Retorna a hora do objeto Date especificado, de acordo com a hora universal.
getUTCMilliseconds	Retorna os milissegundos do objeto Date especificado, de acordo com a hora universal.
getUTCMinutes	Retorna o minuto do objeto Date especificado, de acordo com a hora universal.
getUTCMonth	Retorna o mês do objeto Date especificado, de acordo com a hora universal.
getUTCSeconds	Retorna os segundos do objeto Date especificado, de acordo com a hora universal.



Método	Descrição
getYear	Retorna o ano do objeto Date especificado, de acordo com a hora local.
setDate	Retorna o dia do mês de um objeto Date especificado, de acordo com a hora local.
setFullYear	Define o ano completo de um objeto Date, de acordo com a hora local.
setHours	Define as horas de um objeto Date, de acordo com a hora local.
setMilliseconds	Define os milissegundos de um objeto Date, de acordo com a hora local.
setMinutes	Define os minutos de um objeto Date, de acordo com a hora local.
setMonth	Define o mês de um objeto Date, de acordo com a hora local.
setSeconds	Define os segundos de um objeto Date, de acordo com a hora local.
setTime	Define a data, em milissegundos, do objeto Date especificado.
setUTCDate	Define a data do objeto Date especificado, de acordo com a hora universal.
setUTCFullYear	Define o ano do objeto Date especificado, de acordo com a hora universal.
setUTCHours	Define a hora do objeto Date especificado, de acordo com a hora universal.
setUTCMilliseconds	Define os milissegundos do objeto Date especificado, de acordo com a hora universal.
setUTCMinutes	Define o minuto do objeto Date especificado, de acordo com a hora universal.
setUTCMonth	Define o mês representado pelo objeto Date especificado, de acordo com a hora universal.
setUTCSeconds	Define os segundos do objeto Date especificado, de acordo com a hora universal.
setYear	Define o ano do objeto Date especificado, de acordo com a hora local.
toString	Retorna um valor de sequência de caracteres representando a data e a hora armazenadas no objeto Date especificado.
Date.UTC	Retorna o número de milissegundos entre a meia-noite de 1º de janeiro de 1970, hora universal, e a hora especificada.



## Construtor do objeto Date

### Sintaxe

```
new Date();
```

```
new Date(ano [, mês [, data [, hora [, minutos [, segundos [,  
milissegundos ]]]]] ] );
```

### Argumentos

*ano* Um valor entre 0 e 99 indica 1900 a 1999; caso contrário, é necessário especificar todos os 4 dígitos do ano.

*mês* Um inteiro entre 0 (janeiro) e 11 (dezembro). Esse argumento é opcional.

*data* Um inteiro entre 1 e 31. Esse argumento é opcional.

*hora* Um inteiro entre 0 (meia-noite) e 23 (11 p.m.).

*minutos* Um inteiro entre 0 e 59. Esse argumento é opcional.

*segundos* Um inteiro entre 0 e 59. Esse argumento é opcional.

*milissegundos* Um inteiro entre 0 e 999. Esse argumento é opcional.

### Descrição

Objeto; constrói um objeto new Date que mantém a data e hora atuais.

### Exibidor

Flash 5 ou posterior.

### Exemplo

O exemplo a seguir recupera a data e hora atuais.

```
now = new Date();
```

O exemplo a seguir cria um objeto new Date para o aniversário de Gary, 7 de agosto de 1974.

```
gary_birthday = new Date (74, 7, 7);
```

O exemplo a seguir cria um objeto new Date, concatena os valores retornados dos métodos `getMonth`, `getDate` e `getFullYear`, do objeto Date e os exibe no campo de texto especificado pela variável `dateTextField`.

```
myDate = new Date();  
dateTextField = (mydate.getMonth() + "/" + myDate.getDate() + "/" +  
+ mydate.getFullYear());
```

## Date.getDate

### Sintaxe

`myDate.getDate();`

### Argumentos

Nenhum.

### Descrição

Método; retorna o dia do mês (um inteiro de 1 a 31) do objeto Date especificado, de acordo com a hora local.

### Exibidor

Flash 5 ou posterior.

## Date.getDay

### Sintaxe

`myDate.getDay();`

### Argumentos

Nenhum.

### Descrição

Método; retorna o dia da semana (0 para domingo, 1 para segunda-feira 1, etc.) do objeto Date especificado, de acordo com a hora local. A hora local é determinada pelo sistema operacional em que o Flash Player está sendo executado.

### Exibidor

Flash 5 ou posterior.

## Date.getFullYear

### Sintaxe

`myDate.getFullYear();`

### Argumentos

Nenhum.

### Descrição

Método; retorna o ano completo (um número de quatro dígitos, por exemplo, 2000) do objeto Date especificado, de acordo com a hora local. A hora local é determinada pelo sistema operacional em que o Flash Player está sendo executado.

**Exibidor**

Flash 5 ou posterior.

**Exemplo**

O exemplo a seguir usa o construtor para criar um objeto new Date e enviar o valor retornado pelo método getFullYear para a janela Saída.

```
myDate = new Date();  
trace(myDate.getFullYear());
```

## Date.getHours

**Sintaxe**

```
myDate.getHours();
```

**Argumentos**

Nenhum.

**Descrição**

Método; retorna a hora (um inteiro de 0 a 23) do objeto Date especificado, de acordo com a hora local. A hora local é determinada pelo sistema operacional em que o Flash Player está sendo executado.

**Exibidor**

Flash 5 ou posterior.

## Date.getMilliseconds

**Sintaxe**

```
myDate.getMilliseconds();
```

**Argumentos**

Nenhum.

**Descrição**

Método; retorna os milissegundos (um inteiro de 0 a 999) do objeto Date especificado, de acordo com a hora local. A hora local é determinada pelo sistema operacional em que o Flash Player está sendo executado.

**Exibidor**

Flash 5 ou posterior.

## Date.getMinutes

### Sintaxe

`myDate.getMinutes();`

### Argumentos

Nenhum.

### Descrição

Método; retorna os minutos (um inteiro de 0 a 59) do objeto Date especificado, de acordo com a hora local. A hora local é determinada pelo sistema operacional em que o Flash Player está sendo executado.

### Exibidor

Flash 5 ou posterior.

## Date.getMonth

### Sintaxe

`myDate.getMonth();`

### Argumentos

Nenhum.

### Descrição

Método; retorna o mês (0 para janeiro, 1 para fevereiro, etc.) do objeto Date especificado, de acordo com a hora local. A hora local é determinada pelo sistema operacional em que o Flash Player está sendo executado.

### Exibidor

Flash 5 ou posterior.

## Date.getSeconds

### Sintaxe

`myDate.getSeconds();`

### Argumentos

Nenhum.

### Descrição

Método; retorna os segundos (um inteiro de 0 a 59) do objeto Date especificado, de acordo com a hora local. A hora local é determinada pelo sistema operacional em que o Flash Player está sendo executado.

### Exibidor

Flash 5 ou posterior.

## Date.getTime

### Sintaxe

`myDate.getTime();`

### Argumentos

Nenhum.

### Descrição

Método; retorna o número de milissegundos (um inteiro de 0 a 999) desde a meia-noite de 1º de janeiro de 1970, hora universal, do objeto Date especificado. Use este método para representar um instante específico no tempo ao comparar duas ou mais horas definidas em fusos horários diferentes.

### Exibidor

Flash 5 ou posterior.

## Date.getTimezoneOffset

### Sintaxe

`mydate.getTimezoneOffset();`

### Argumentos

Nenhum.

### Descrição

Método; retorna a diferença, em minutos, entre o a hora local do computador e a hora universal.

### Exibidor

Flash 5 ou posterior.

### Exemplo

O exemplo a seguir retorna a diferença entre o horário de verão de São Francisco e a hora universal. O horário de verão é fatorado no resultado retornado somente se a data definida no objeto Date estiver no horário de verão.

```
new Date().getTimezoneOffset();
```

O resultado é este:

420 (7 horas \* 60 minutos/hora = 420 minutos)

## Date.getUTCDate

### Sintaxe

*myDate*.getUTCDate();

### Argumentos

Nenhum.

### Descrição

Método; retorna o dia (data) do mês do objeto Date especificado, de acordo com a hora universal.

### Exibidor

Flash 5 ou posterior.

## Date.getUTCDay

### Sintaxe

*myDate*.getUTCDay();

### Argumentos

Nenhum.

### Descrição

Método; retorna o dia da semana do objeto Date especificado, de acordo com a hora universal.

## Date.getUTCFullYear

### Sintaxe

*myDate*.getUTCFullYear();

### Argumentos

Nenhum.

### Descrição

Método; retorna o ano com quatro dígitos do objeto Date especificado, de acordo com a hora universal.

### Exibidor

Flash 5 ou posterior.

## Date.getUTCHours

### Sintaxe

`myDate.getUTCHours();`

### Argumentos

Nenhum.

### Descrição

Método; retorna a hora do objeto Date especificado, de acordo com a hora universal.

### Exibidor

Flash 5 ou posterior.

## Date.getUTCMilliseconds

### Sintaxe

`myDate.getUTCMilliseconds();`

### Argumentos

Nenhum.

### Descrição

Método; retorna os milissegundos do objeto Date especificado, de acordo com a hora universal.

### Exibidor

Flash 5 ou posterior.

## Date.getUTCMinutes

### Sintaxe

`myDate.getUTCMinutes();`

### Argumentos

Nenhum.

### Descrição

Método; retorna os minutos do objeto Date especificado, de acordo com a hora universal.

### Exibidor

Flash 5 ou posterior.



## Date.getUTCMonth

### Sintaxe

`myDate.getUTCMonth();`

### Argumentos

Nenhum.

### Descrição

Método; retorna o mês do objeto Date especificado, de acordo com a hora universal.

### Exibidor

Flash 5 ou posterior.

## Date.getUTCSeconds

### Sintaxe

`myDate.getUTCSeconds();`

### Argumentos

Nenhum.

### Descrição

Método; retorna os segundos do objeto Date especificado, de acordo com a hora universal.

### Exibidor

Flash 5 ou posterior.

## Date.getYear

### Sintaxe

`myDate.getYear();`

### Argumentos

Nenhum.

### Descrição

Método; retorna o ano do objeto Date especificado, de acordo com a hora local. A hora local é determinada pelo sistema operacional em que o Flash Player está sendo executado. O ano é o ano completo menos 1900. Por exemplo, o ano 2000 é representado como 100.

### Exibidor

Flash 5 ou posterior.



## Date.setDate

### Sintaxe

`myDate.setDate(data);`

### Argumentos

*data* Um inteiro de 1 a 31.

### Descrição

Método; define o dia do mês do objeto Date especificado, de acordo com a hora local. A hora local é determinada pelo sistema operacional em que o Flash Player está sendo executado.

### Exibidor

Flash 5 ou posterior.

## Date.setFullYear

### Sintaxe

`myDate.setFullYear(ano [, mês [, data]] );`

### Argumentos

*ano* Um número de quatro dígitos que especifique um ano. Números de dois dígitos não representam anos; por exemplo, 99 não é o ano 1999, mas o ano 99.

*mês* Um inteiro entre 0 (janeiro) e 11 (dezembro). Esse argumento é opcional.

*data* Um número entre 1 e 31. Esse argumento é opcional.

### Descrição

Método; define o ano do objeto Date especificado, de acordo com a hora local. Se os argumentos *mês* e *data* forem especificados, também serão definidos para a hora local. A hora local é determinada pelo sistema operacional em que o Flash Player está sendo executado.

Os resultados de `getUTCDay` e `getDay` podem ser alterado como resultado da chamada deste método.

### Exibidor

Flash 5 ou posterior.

## Date.setHours

### Sintaxe

`myDate.setHours(hora);`

### Argumentos

*hora* Um inteiro entre 0 (meia-noite) e 23 (11 p.m.).

### Descrição

Método; define a hora do objeto Date especificado, de acordo com a hora local. A hora local é determinada pelo sistema operacional em que o Flash Player está sendo executado.

### Exibidor

Flash 5 ou posterior.

## Date.setMilliseconds

### Sintaxe

`myDate.setMilliseconds(milissegundos);`

### Argumentos

*milissegundos* Um inteiro de 0 a 999.

### Descrição

Método; define os milissegundos do objeto Date especificado, de acordo com a hora local. A hora local é determinada pelo sistema operacional em que o Flash Player está sendo executado.

### Exibidor

Flash 5 ou posterior.

## Date.setMinutes

### Sintaxe

`myDate.setMinutes(minuto);`

### Argumentos

*minuto* Um inteiro de 0 a 59.

### Descrição

Método; define os minutos de um objeto Date especificado, de acordo com a hora local. A hora local é determinada pelo sistema operacional em que o Flash Player está sendo executado.

### Exibidor

Flash 5 ou posterior.

## Date.setMonth

### Sintaxe

`myDate.setMonth(mês [, data ])`;

### Argumentos

*mês* Um inteiro entre 0 (janeiro) e 11 (dezembro).

*data* Um inteiro entre 1 e 31. Esse argumento é opcional.

### Descrição

Método; define o mês do objeto Date especificado, de acordo com a hora local. A hora local é determinada pelo sistema operacional em que o Flash Player está sendo executado.

### Exibidor

Flash 5 ou posterior.

## Date.setSeconds

### Sintaxe

`myDate.setSeconds(segundos)`;

### Argumentos

*segundos* Um inteiro de 0 a 59.

### Descrição

Método; define os segundos do objeto Date especificado, de acordo com a hora local. A hora local é determinada pelo sistema operacional em que o Flash Player está sendo executado.

### Exibidor

Flash 5 ou posterior.

## Date.setTime

### Sintaxe

`myDate.setTime(milissegundos)`;

### Argumentos

*milissegundos* Um inteiro de 0 a 999.

### Descrição

Método; define a data, em milissegundos, do objeto Date especificado.

### Exibidor

Flash 5 ou posterior.

## Date.setUTCDate

### Sintaxe

`myDate.setUTCDate(data);`

### Argumentos

*data* Um inteiro de 1 a 31.

### Descrição

Método; define a data do objeto Date especificado, de acordo com a hora universal. Quando este método é chamado, os outros campos do Date especificado não são modificados, mas, se o dia da semana for alterado como resultado da chamada desse método, os métodos `getUTCDay` e `getDay` podem reportar um novo valor.

### Exibidor

Flash 5 ou posterior.

## Date.setUTCFullYear

### Sintaxe

`myDate.setUTCFullYear(ano [, mês [, data]])`

### Argumentos

*ano* O ano especificado com quatro dígitos completos; por exemplo, 2000.

*mês* Um inteiro entre 0 (janeiro) e 11 (dezembro). Esse argumento é opcional.

*data* Um inteiro entre 1 e 31. Esse argumento é opcional.

### Descrição

Método; define o ano do objeto Date (*mydate*) especificado, de acordo com a hora universal.

Opcionalmente, este método também pode definir o mês e a data representados pelo objeto Date especificado. Nenhum outro campo do objeto Date é modificado. A chamada de `setUTCFullYear` pode fazer com que `getUTCDay` e `getDay` reportem um novo valor se o dia da semana for alterado como resultado dessa operação.

### Exibidor

Flash 5 ou posterior.

## Date.setUTCHours

### Sintaxe

```
myDate.setUTCHours(hora [, minutos [, segundos [,  
milissegundos]]]);
```

### Argumentos

*hora* Um inteiro entre 0 (meia-noite) e 23 (11 p.m.).

*minutos* Um inteiro entre 0 e 59. Esse argumento é opcional.

*segundos* Um inteiro entre 0 e 59. Esse argumento é opcional.

*milissegundos* Um inteiro entre 0 e 999. Esse argumento é opcional.

### Descrição

Método; define a hora do objeto Date especificado, de acordo com a hora universal.

### Exibidor

Flash 5 ou posterior.

## Date.setUTCMilliseconds

### Sintaxe

```
myDate.setUTCMilliseconds(milissegundos);
```

### Argumentos

*milissegundos* Um inteiro de 0 a 999.

### Descrição

Método; define os milissegundos do objeto Date especificado, de acordo com a hora universal.

### Exibidor

Flash 5 ou posterior.

## Date.setUTCMinutes

### Sintaxe

```
myDate.setUTCMinutes(minutos [, segundos [, milissegundos]]);
```

### Argumentos

*minuto* Um inteiro de 0 a 59.

*segundos* Um inteiro entre 0 e 59. Esse argumento é opcional.

*milissegundos* Um inteiro entre 0 e 999. Esse argumento é opcional.

#### Descrição

Método; define os minutos do objeto Date especificado, de acordo com a hora universal.

#### Exibidor

Flash 5 ou posterior.

## Date.setUTCMonth

#### Sintaxe

```
myDate.setUTCMonth(mês [, data]);
```

#### Argumentos

*mês* Um inteiro entre 0 (janeiro) e 11 (dezembro).

*data* Um inteiro entre 1 e 31. Esse argumento é opcional.

#### Descrição

Método; define o mês e, opcionalmente, o dia (*data*) do objeto Date especificado, de acordo com a hora universal. Quando este método é chamado, os outros campos do objeto Date especificado não são modificados, mas, se o dia da semana for alterado como resultado da especificação do argumento *data* quando for `setUTCMonth` for chamado, os métodos `getUTCDay` e `getDay` podem reportar um novo valor.

#### Exibidor

Flash 5 ou posterior.

## Date.setUTCSeconds

#### Sintaxe

```
myDate.setUTCSeconds(segundos [, milissegundos]);
```

#### Argumentos

*segundos* Um inteiro de 0 a 59.

*milissegundos* Um inteiro entre 0 e 999. Esse argumento é opcional.

#### Descrição

Método; define os segundos do objeto Date especificado, de acordo com a hora universal.

#### Player

Flash 5 ou posterior.

## Date.setYear

### Sintaxe

`myDate.setYear(ano);`

### Argumentos

*ano* Um número de quatro dígitos; por exemplo, 2000.

### Descrição

Método; define o ano do objeto Date especificado, de acordo com a hora local. A hora local é determinada pelo sistema operacional em que o Flash Player está sendo executado.

### Exibidor

Flash 5 ou posterior.

## Date.toString

### Sintaxe

`myDate.toString();`

### Argumentos

Nenhum.

### Descrição

Método; retorna, em um formato legível, um valor de sequência de caracteres para o objeto Date especificado.

### Exibidor

Flash 5 ou posterior.

### Exemplo

O exemplo a seguir retorna as informações no objeto Date `dataNascimento` como uma sequência de caracteres.

```
var dataNascimento = new Date(74, 7, 7, 18, 15);  
trace (dataNascimento.toString());
```

Saída (para Hora padrão do Pacífico):

Qua Ago 7 18:15:00 GMT-0700 1974

## Date.UTC

### Sintaxe

`Date.UTC(ano, mês [, data [, hora [, minutos [, segundos  
[, milissegundos ]]]]]);`

### Argumentos

*ano* Um número de quatro dígitos; por exemplo, 2000.

*mês* Um inteiro entre 0 (janeiro) e 11 (dezembro).

*data* Um inteiro entre 1 e 31. Esse argumento é opcional.

*hora* Um inteiro entre 0 (meia-noite) e 23 (11 p.m.).

*minutos* Um inteiro entre 0 e 59. Esse argumento é opcional.

*segundos* Um inteiro entre 0 e 59. Esse argumento é opcional.

*milissegundos* Um inteiro entre 0 e 999. Esse argumento é opcional.

### Descrição

Método; retorna o número de milissegundos entre a meia-noite de 1º de janeiro de 1970, hora universal, e a hora especificada. Este é um método estático chamado pelo construtor do objeto Date, não por um objeto Date específico. Esse método permite criar um objeto Date que assuma uma hora universal, enquanto o construtor de Date assume a hora local.

### Exibidor

Flash 5 ou posterior.

### Exemplo

O exemplo a seguir cria um objeto `new Date gary_birthday`, definido na hora universal. Esta é a variação de hora universal do exemplo usado para o método construtor `new Date()`.

```
gary_birthday = new Date(Date.UTC(1974, 7, 8));
```



## delete

### Sintaxe

`delete (referência);`

### Argumentos

*referência* O nome da variável ou do objeto a ser eliminado.

### Descrição

Operador; elimina o objeto ou a variável especificada como *referência* e retorna `true` se o objeto for excluído com êxito; caso contrário, retorna `false`. Esse operador é útil para liberar memória usada por. Entretanto, `delete` é um operador usado normalmente como comando:

```
delete x;
```

O operador `delete` pode falhar e retornar `false` se *referência* não existir ou não puder ser excluída. Objetos e propriedades predefinidos e variáveis declaradas com `var` não podem ser excluídas.

### Exibidor

Flash 5 ou posterior.

### Exemplo

O exemplo a seguir cria um objeto, usa e o exclui assim que não for mais necessário.

```
account = new Object();  
    account.name = 'Jon';  
    account.balance = 10000;  
    ...  
    delete account;
```

O exemplo a seguir exclui uma propriedade de um objeto.

```
// cria um novo objeto "account"  
account = new Object();  
// atribui nome de propriedade a account  
    account.name = 'Jon';  
// exclui a propriedade  
delete account.name;
```



O exemplo a seguir é outro exemplo de exclusão da propriedade de um objeto.

```
// cria um objeto Array com tamanho 0
array = new Array();
// Array.length é agora 1
array[0] = "abc";
// adiciona outro elemento à matriz, Array.length é agora 2
array[1] = "def";
// adiciona outro elemento à matriz, Array.length é agora 3
array[2] = "ghi";
// array[2] é excluída, mas Array.length não é alterada,
delete array[2];
```

O exemplo a seguir ilustra o comportamento de delete em referências de objetos.

```
// cria um novo objeto e atribui a variável ref1 para fazer
referência ao objeto
ref1 = new Object();
ref1.name = "Jody";
// copia a variável de referência para uma nova variável e
exclui ref1
ref2 = ref1;
delete ref1;
```

Se ref1 não tivesse sido copiada para ref2, o objeto teria sido excluído durante a exclusão de ref1, pois não haveria referências a ele. Se ref2 fosse excluída, não haveria mais referências ao objeto e ele seria eliminado, liberando a memória que ele estava usando.

**Consulte também**  
var, na página 388

## do... while

### Sintaxe

```
do {
    comando;
} while (condição);
```

### Argumentos

*condição* A condição a ser avaliada.

*comando* O comando a ser executado, desde que *condição* seja avaliada como true.

### Descrição

Ação; executa os comandos e avalia a condição em um loop, desde que a condição seja verdadeira.

**Exibidor**

Flash 4 ou posterior.

**Consulte também**

break, na página 235

continue, na página 241

## **\_droptarget**

**Sintaxe**

*nomeInstanciaArrastavel*.\_droptarget

**Argumentos**

*nome da instância arrastável* O nome da instância de um clipe de filme que sofreu uma ação `startDrag`.

**Descrição**

Propriedade (somente leitura); retorna o caminho absoluto, em notação de sintaxe de barra, da instância do clipe de filme em que *nome da instância arrastável* foi solta. A propriedade `_droptarget` sempre retorna um caminho que começa por `/`. Para comparar a propriedade `_droptarget` de uma instância a uma referência, use `eval` para converter o valor retornado de sintaxe de barra para uma referência.

**Exibidor**

Flash 4 ou posterior.

**Exemplo**

O exemplo a seguir avalia a propriedade `_droptarget` da instância do clipe de filme `garbage` e usa `eval` para convertê-la de sintaxe de barra em uma referência de sintaxe de ponto. A referência `garbage` é, então, comparada com a referência à instância do clipe de filme `trash`. Se as duas referências forem equivalentes, a visibilidade de `garbage` será definida como `false`. Se não forem equivalentes, a instância de `garbage` será redefinida para sua posição original.

```
if (eval(garbage._droptarget) == _root.trash) {  
    garbage._visible = false;  
} else {  
    garbage._x = x_pos;  
    garbage._y = y_pos;  
}
```

As variáveis `x_pos` e `y_pos` são definidas no quadro 1 do filme com o seguinte script:

```
x_pos = garbage._x;  
y_pos = garbage._y;
```

**Consulte também**

`startDrag`, na página 368

## duplicateMovieClip

### Sintaxe

`duplicateMovieClip(destino, novo nome, profundidade);`

### Argumentos

*destino* O caminho de destino do filme a ser duplicado.

*novonome* Um identificador exclusivo para o clipe de filme duplicado.

*profundidade* O nível de profundidade do clipe de filme. O nível de profundidade é a ordem de empilhamento que determina como os cliques de filme e outros objetos são exibidos quando se sobrepõem. O primeiro clipe de filme que você criar, ou instância que arrastar para o Palco, terá uma profundidade atribuída de nível 0. É necessário atribuir a cada clipe de filme sucessivo ou duplicado um nível de profundidade diferente para impedir que ele substitua filmes em níveis ocupados ou o clipe de filme original.

### Descrição

Ação; cria uma instância de um clipe de filme enquanto o filme é reproduzido. Cliques de filme duplicados sempre começam no quadro 1, não importa o quadro em que o clipe de filme original estava. As variáveis no clipe de filme pai não são copiadas para o clipe de filme duplicado. Se o clipe de filme pai for excluído, o clipe de filme duplicado também será. Use a ação ou método `removeMovieClip` para excluir uma instância de clipe de filme criada com `duplicateMovieClip`.

### Exibidor

Flash 4 ou posterior.

### Exemplo

Este comando duplica a instância do clipe de filme `flower` dez vezes. A variável `i` é usada para criar um novo nome de instância e uma profundidade.

```
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;
    }
}
```

### Consulte também

`removeMovieClip`, na página 351

`MovieClip.removeMovieClip`, na página 325

## else

### Sintaxe

`else {comando(s)};`

### Argumentos

*comando(s)* Uma série de comandos alternativos a serem executados se a condição especificada no comando `if` for `false`.

### Descrição

Ação; especifica as ações, condições, argumentos ou outros condicionais a serem executados se o comando inicial `if` retornar `false`.

### Exibidor

Flash 4 ou posterior.

### Consulte também

`if`, na página 279

## eq (igual – específico de seqüência de caracteres)

### Sintaxe

*expressão1* eq *expressão2*

### Argumentos

*expressão1*, *expressão2* Números, seqüências de caracteres ou variáveis.

### Descrição

Operador de comparação; compara a igualdade de duas expressões e retorna `true` se *expressão1* for igual a *expressão2*; caso contrário, retorna `false`.

### Exibidor

Flash 1 ou posterior. Esta função foi reprovada no Flash 5; recomenda-se o uso do novo operador `==` (igualdade).

### Consulte também

`==` (igualdade), na página 216

## escape

### Sintaxe

`escape(expressão);`

### Argumentos

*expressão* A expressão a ser convertida em seqüência de caracteres e codificada em um formato de código URL.

### Descrição

Função; converte o argumento em uma sequência de caracteres e a codifica em formato de código URL, onde todos os caracteres alfanuméricos usam sequências de caracteres hexadecimais % para escape.

### Exibidor

Flash 5 ou posterior.

### Exemplo

```
escape("Hello{[World]}");
```

O resultado do código acima é:

```
Hello%7B%5BWorld%5D%7D
```

### Consulte também

unescape, na página 386

## eval

### Sintaxe

```
eval(expressão);
```

### Argumentos

*expressão* Uma sequência de caracteres que contém o nome de uma variável, propriedade, objeto ou clipe de filme a ser recuperado.

### Descrição

Função; acessa variáveis, propriedades, objetos ou clipe de filme, por nome. Se *expressão* for uma variável ou propriedade, será retornado o valor da variável ou propriedade. Se *expressão* for um objeto ou clipe de filme, será retornada uma referência ao objeto ou clipe de filme. Se não for possível encontrar o elemento nomeado em *expressão*, será retornado não-definido.

No Flash 4, a função `eval` era usada para simular uma matriz; no Flash 5, recomenda-se usar o objeto `Array` para criar matrizes.

**Observação:** A ação `eval` do ActionScript não é a mesma que a função `eval` do JavaScript e não pode ser usada para avaliar comandos.

### Exibidor

Flash 5 ou posterior com funcionalidade completa. Você pode usar `eval` ao exportar para o Flash 4 Player, mas deve usar notação de barra e só pode acessar variáveis, não propriedades ou objetos.

### Exemplo

O exemplo a seguir usa `eval` para determinar o valor da variável `x` e a define como o valor de `y`.

```
x = 3;  
y = eval("x");
```

O exemplo a seguir usa `eval` para fazer referência ao objeto de clipe de filme associado à instância `Ball` do clipe de filme no Palco.

```
eval("_root.Ball");
```

**Consulte também**

Array (objeto), na página 224

## evaluate

**Sintaxe**

*comando*;

**Argumentos**

Nenhum.

**Descrição**

Ação; cria uma nova linha vazia e insere um ; para inserção de comandos de script exclusivos por meio do campo Expressão no painel Ações. O comando `evaluate` também permite que os usuários que criem scripts no Modo Normal do painel Ações do Flash 5 chamem funções.

**Exibidor**

Flash 5 ou posterior.

## \_focusrect

**Sintaxe**

```
_focusrect = Booleano;
```

**Argumentos**

*Booleano* `true` ou `false`.

**Descrição**

Propriedade (global); especifica se é exibido um retângulo amarelo em volta do botão que tem foco no momento. O valor padrão `true` (não zero) exibe um retângulo em volta do botão ou campo de texto com foco no momento quando o usuário pressiona a tecla Tab para navegar. Especifique `false` para exibir somente o estado “sobre” do botão (se houver um definido) enquanto os usuários navegam.

**Exibidor**

Flash 4 ou posterior.

## for

### Sintaxe

```
for(início; condição; próximo); {  
  comando;  
}
```

### Argumentos

*início* Uma expressão a ser avaliada antes de começar a sequência de loop, normalmente uma expressão de atribuição. Também é permitido um comando `var` para esse argumento.

*condição* Uma condição que seja avaliada como `true` ou `false`. A condição é avaliada antes de cada iteração do loop; o loop termina quando a condição é avaliada como `false`.

*próximo* Uma expressão que é avaliada após cada iteração do loop; normalmente, uma expressão de atribuição com os operadores `++` (incremento) ou `--` (decremento).

*comando* Um comando a ser executado no corpo do loop.

### Descrição

Ação; um construtor de loop que avalia a expressão *início* (inicializar) uma vez e começa a sequência do loop pelo qual o *comando* é executado e a próxima expressão é avaliada enquanto a *condição* for avaliada como `true`.

Algumas propriedades não podem ser enumeradas pelas ações `for` ou `for..in`. Por exemplo, os métodos internos do objeto `Array` (`Array.sort` e `Array.reverse`) não são incluídos na enumeração de um objeto `Array` e as propriedades de clipe de filme, como `_x` e `_y`, não são enumeradas.

### Exibidor

Flash 5 ou posterior.



### Exemplo

O exemplo a seguir usa `for` para adicionar os elementos a uma matriz.

```
for(i=0; i<10; i++) {
    array [i] = (i + 5)*10;
}
```

Retorna a seguinte matriz:

[50, 60, 70, 80, 90, 100, 110, 120, 130, 140]

O exemplo a seguir mostra o uso de `for` para executar a mesma ação repetidamente. No código abaixo, o loop `for` adiciona os números de 1 a 100.

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

### Consulte também

`++` (incremento), na página 189

`--` (decremento), na página 189

`for..in`, na página 269

`var`, na página 388

## for..in

### Sintaxe

```
for(viterando_variável in objeto){
    comando;
```

### Argumentos

*iterando\_variável* O nome de uma variável que age como iterando, fazendo referência a cada propriedade de um objeto ou elemento em uma matriz.

*objeto* O nome de um objeto a ser iterado.

*comando* Um comando a ser executado para cada iteração.

### Descrição

Ação; realiza um loop pelas propriedades de um objeto ou de elementos de uma matriz e executa o *comando* para cada propriedade de um objeto.

Algumas propriedades não podem ser enumeradas pelas ações `for` ou `for..in`. Por exemplo, os métodos internos do objeto Array (`Array.sort` e `Array.reverse`) não são incluídos na enumeração de um objeto Array e as propriedades de clipe de filme, como `_x` e `_y`, não são enumeradas.

O construtor `for..in` itera em propriedades de objetos na cadeia protótipo do objeto iterado. Se o protótipo do filho for pai, a iteração pelas propriedades do filho com `for..in` também fará a iteração pelas propriedades do pai.

### Exibidor

Flash 5 ou posterior.

### Exemplo

O exemplo a seguir mostra o uso de `for..in` para iterar pelas propriedades de um objeto.

```
myObject = { name:'Tara', age:27, city:'San Francisco' };  
for (name in myObject) {  
    trace ("myObject." + name + " = " + myObject[name]);  
}
```

A saída deste exemplo é:

```
myObject.name = Tara  
myObject.age = 27  
myObject.city = San Francisco
```

O exemplo a seguir mostra o uso do operador `typeof` com `for..in` para iterar por um determinado tipo de filho.

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

O exemplo a seguir enumera os filhos de um clipe de filme e envia cada um para o quadro 2 de suas respectivas linhas de tempo. O clipe de filme `RadioButtonGroup` é um pai com vários filhos, `_RedRadioButton_`, `_GreenRadioButton_` e `_BlueRadioButton_`.

```
for (var name in RadioButtonGroup) {  
    RadioButtonGroup[name].gotoAndStop(2);  
}
```

## `_framesloaded`

### Sintaxe

`nome_da_instância._framesloaded`

### Argumentos

`nome_da_instância` O nome da instância de clipe de filme a ser avaliada.

### Descrição

Propriedade (somente leitura); o número de quadros que foram carregados de um filme em fluxo. Esta propriedade é útil para determinar se o conteúdo de um determinado quadro e todos os quadros antes dele foram carregados e estão disponíveis localmente no navegador de um usuário. Isso é útil para monitorar o processo de download de filmes grandes. Por exemplo, você pode exibir uma mensagem para os usuários indicando que o filme está carregando até que um determinado quadro do filme tenha sido carregado.

### Exibidor

Flash 4 ou posterior.

### Exemplo

O exemplo a seguir mostra o uso da propriedade `_framesloaded` para coordenar o início do filme com o número de quadros carregados.

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

## `fscommand`

### Sintaxe

`fscommand(comando, argumentos);`

### Argumentos

*comando* Uma sequência de caracteres passada para o aplicativo host para qualquer uso.

*argumentos* Uma sequência de caracteres passada para o aplicativo host para qualquer uso.

### Descrição

Ação; permite que o filme do Flash se comunique com o programa que hospeda o Flash Player. Em um navegador, `fscommand` chama a função JavaScript `nomedofilme_Dofsccommand` na página HTML que contém o filme do Flash, onde `nomedofilme` é o nome do Flash Player, conforme atribuído pelo atributo `NAME` da marca `EMBED` ou pela propriedade `ID` da marca `OBJECT`. Se o nome `theMovie` estiver atribuído ao Flash Player, a função JavaScript chamada será `theMovie_Dofsccommand`.

### Exibidor

Flash 3 ou posterior.

## function

### Sintaxe

```
function nome_da_função ([argumento0, argumento1,...argumentoN]){
    comando(s)
}
```

```
function ([argumento0, argumento1,...argumentoN]){
    comando(s)
}
```

### Argumentos

*nome\_da\_função* O nome da nova função.

*argumento* Nenhuma ou mais seqüências de caracteres, números ou objetos a serem passados para função.

*comando(s)* Nenhum ou mais comandos ActionScript que você definiu para o corpo da função.

### Descrição

Ação; um conjunto de comandos que você define para a realização de uma determinada tarefa. Você pode *declarar*, ou definir, uma função em um local e chamá-la de diferentes scripts em um filme. Quando você define uma função, também pode especificar argumentos para ela. Os argumentos são espaços reservados para valores em que a função operará. Você pode passar diferentes argumentos, também chamados de parâmetros, para uma função toda vez que chamá-la.

Use a ação `return no(s) comando(s)` de uma função para fazê-la retornar, ou gerar, um valor.



**Uso 1:** Declara uma função com o *nome\_da\_função*, os *argumentos* e o(s) *comando(s)* especificados. Quando uma função é chamada, a declaração da função é chamada. Não é permitido repassar uma referência; na mesma lista de ações, uma função pode ser declarada após ser chamada. Uma declaração de função substitui qualquer declaração anterior da mesma função. Esta sintaxe pode ser usada sempre que for permitido um comando.

**Uso 2:** Cria uma função anônima e a retorna. Esta sintaxe é usada em expressões e é particularmente útil para a instalação de métodos em objetos.

#### Exibidor

Flash 5 ou posterior.

#### Exemplo

(Uso 1) O exemplo a seguir define a função *sqr*, que aceita um argumento, e retorna o *quadrado* ( $x*x$ ) do argumento. Observe que, se a função for declarada e usada no mesmo script, a declaração de função pode aparecer após o uso da função.

```
y=sqr(3);
function sqr(x) {
return x*x;
}
```

(Uso 2) A função a seguir define um objeto *Circle*:

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

O comando a seguir define uma função anônima que calcula a área de um círculo e a anexa ao objeto *Circle* como um método:

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

## ge (maior ou igual a – específico de seqüências de caracteres)

#### Sintaxe

*expressão1* ge *expressão2*

#### Argumentos

*expressão1*, *expressão2* Números, seqüências de caracteres ou variáveis.

#### Descrição

Operador (comparação); compara *expressão1* com a *expressão2* e retorna *true* se *expressão1* for maior ou igual a *expressão2*; caso contrário, retorna *false*.

#### Exibidor

Flash 4 ou posterior. Este operador está obsoleto no Flash 5; recomenda-se o uso do novo operador `>=`.

#### Consulte também

`>=` (maior ou igual a), na página 217

## getProperty

#### Sintaxe

```
getProperty(nome_da_instância, propriedade);
```

#### Argumentos

*nome\_da\_instância* O nome da instância de um clipe de filme para o qual a propriedade está sendo recuperada.

*propriedade* Uma propriedade de um clipe de filme, como uma coordenada *x* ou *y*.

#### Descrição

Função; retorna o valor da *propriedade* especificada para a instância do clipe de filme.

#### Exibidor

Flash 4 ou posterior.

#### Exemplo

O exemplo a seguir recupera a coordenada do eixo horizontal (*\_x*) do clipe de filme *myMovie*.

```
getProperty(_root.myMovie_item._x);
```

## getTimer

#### Sintaxe

```
getTimer();
```

#### Argumentos

Nenhum.

#### Descrição

Função; retorna o número de milissegundos decorridos deste o início da reprodução do filme.

#### Exibidor

Flash 4 ou posterior.

## getURL

### Sintaxe

```
getURL(url [, janela [, variáveis]]);
```

### Argumentos

*url* A URL de onde o documento deve ser obtido. A URL deve estar no mesmo subdomínio que a URL onde o filme reside atualmente.

*janela* Um argumento opcional que especifica a janela ou quadro HTML em que o documento deve ser carregado. Insira o nome de uma janela específica ou escolha um destes nomes de destino reservados:

- `_self` especifica o quadro atual na janela atual.
- `_blank` especifica uma nova janela.
- `_parent` especifica o pai do quadro atual.
- `_top` especifica o quadro de nível superior na janela atual.

*variáveis* Um argumento opcional que especifica um método para o envio de variáveis. Se não houver variáveis, omita este argumento; caso contrário, especifique se as variáveis devem ser carregadas com um método GET ou POST. GET anexa as variáveis ao fim da URL e é usado para um pequeno número de variáveis. POST envia as variáveis em um cabeçalho HTTP separado e é usado para grandes seqüências de caracteres de variáveis.

### Descrição

Ação; carrega um documento de uma URL específica em uma janela ou passa variáveis para outro aplicativo em uma URL definida. Para testar esta ação, certifique-se de que o arquivo a ser carregado esteja no local especificado. Para usar uma URL absoluta (por exemplo, `http://www.meuservidor.com`), você precisa de uma conexão de rede.

### Exibidor

Flash 2 ou posterior. As opções GET e POST estão disponíveis somente para o Flash 4 e versões posteriores do Exibidor.

### Exemplo

Este exemplo carrega uma nova URL em uma janela em branco do navegador. A ação `getURL` direciona a variável `incomingAd` como o parâmetro `url` para que você possa alterar a URL carregada sem que seja necessário editar o filme do Flash. O valor da variável `incomingAd` é passado para o Flash no início do filme com uma ação `loadVariables`.

```
on(release) {  
    getURL(incomingAd, "_blank");  
}
```

### Consulte também

`loadVariables`, na página 295

`XML.send`, na página 408

`XML.sendAndLoad`, na página 408

`XMLSocket.send`, na página 419

## getVersion

### Sintaxe

```
getVersion();
```

### Argumentos

Nenhum.

### Descrição

Função; retorna uma sequência de caracteres contendo informações sobre a versão e plataforma do Flash Player.

Esta função não funciona no modo de teste de filme e somente retornará informações sobre as versões 5 ou posteriores do Flash Player.

### Exemplo

O exemplo a seguir mostra uma sequência de caracteres retornada pela função `getVersion`:

```
WIN 5,0,17,0
```

Isso indica que a plataforma é Windows e o número da versão do Flash Player é versão 5 principal, versão secundária 17(5.0r17).

### Exibidor

Flash 5 ou posterior.



## gotoAndPlay

### Sintaxe

`gotoAndPlay(cena, quadro);`

### Argumentos

*cena* O nome da cena para onde a reprodução é enviada.

*quadro* O número do quadro para onde a reprodução é enviada.

### Descrição

Ação; envia a reprodução para o quadro especificado em uma cena e reproduz a partir desse quadro. Se não for especificada uma cena, a reprodução segue para o quadro especificado na cena atual.

### Exibidor

Flash 2 ou posterior.

### Exemplo

Quando o usuário clica em um botão ao qual está atribuída a ação `gotoAndPlay`, a reprodução é enviada para o quadro 16 e começa a reproduzir.

```
on(release) {  
    gotoAndPlay(16);  
}
```

## gotoAndStop

### Sintaxe

`gotoAndStop(cena, quadro);`

### Argumentos

*cena* O nome da cena para onde a reprodução é enviada.

*quadro* O número do quadro para onde a reprodução é enviada.

### Descrição

Ação; envia a reprodução para o quadro especificado em uma cena e interrompe. Se não for especificada uma cena, a reprodução segue para o quadro especificado na cena atual.

### Exibidor

Flash 2 ou posterior.

### Exemplo

Quando o usuário clica em um botão ao qual está atribuída a ação `gotoAndStop`, a reprodução é enviada para o quadro 5 e o filme pára de ser reproduzido.

```
on(release) {  
    gotoAndStop(5);  
}
```

## gt (maior que – específico de seqüências de caracteres)

### Sintaxe

*expressão1* gt *expressão2*

### Argumentos

*expressão1*, *expressão2* Números, seqüências de caracteres ou variáveis.

### Descrição

Operador (comparação); compara *expressão1* a *expressão2* e retorna true se *expressão1* for maior que *expressão2*; caso contrário, retorna false.

### Exibidor

Flash 4 ou posterior. Este operador foi reprovado no Flash 5; recomenda-se o uso do novo operador >.

### Consulte também

> (maior que), na página 216

## \_height

### Sintaxe

*nome\_da\_instância*.\_height  
*nome\_da\_instância*.\_height = *valor*;

### Argumentos

*nome\_da\_instância* O nome da instância de um clipe de filme para o qual a propriedade \_height deve ser definida ou recuperada.

*valor* Um inteiro que especifique a altura do filme, em pixels.

### Descrição

Propriedade; define e recupera a altura do espaço ocupado pelo conteúdo de um filme. Nas versões anteriores do Flash, \_height e \_width eram propriedades somente leitura; no Flash 5, elas podem ser definidas.

### Exibidor

Flash 4 ou posterior.

### Exemplo

O exemplo de código a seguir define a altura e a largura de um clipe de filme quando o usuário clicar com o mouse.

```
onClipEvent(mouseDown) {  
    _width=200;  
    _height=200;  
}
```



## **\_highquality**

### **Sintaxe**

`_highquality = valor;`

### **Argumentos**

*valor* O nível de sem serrilhado aplicado ao filme. Especifique 2 (MELHOR) para aplicar alta qualidade com a suavização de bitmap sempre ativada. Especifique 1 (alta qualidade) para aplicar o recurso sem serrilhado; isso suavizará os bitmaps se o filme não contiver animação. Especifique 0 (baixa qualidade) para evitar o recurso sem serrilhado.

### **Descrição**

Propriedade (global); especifica o nível de sem serrilhado aplicado no filme atual.

### **Exibidor**

Flash 4 ou posterior.

### **Consulte também**

`_quality`, na página 350

`toggleHighQuality`, na página 384

## **if**

### **Sintaxe**

```
if(condição) {  
  
    comando(s);  
  
}
```

### **Argumentos**

*condição* Uma condição que seja avaliada como `true` ou `false`. Por exemplo, `if(name == "Erica")`, avalia a variável `name` para verificar se é "Erica".

*comando(s)* As instruções a serem executadas se ou quando a condição for avaliada como `true`.

### **Descrição**

Ação; avalia uma condição para determinar a próxima ação em um filme. Se a condição for `true`, o Flash executa os comandos seguintes. Use `if` para criar lógica ramificada em seus scripts.

### **Exibidor**

Flash 4 ou posterior.

### **Consulte também**

`else`, na página 265

`for`, na página 268

`for...in`, na página 269

## ifFrameLoaded

### Sintaxe

```
ifFrameLoaded(cena, quadro) {  
comando;}
```

```
ifFrameLoaded(quadro) {  
comando;}
```

### Argumentos

*cena* A cena que está sendo consultada.

*quadro* O número ou rótulo do quadro a ser carregado antes que o próximo comando seja executado.

### Descrição

Ação; verifica se o conteúdo de um quadro específico está disponível localmente. Use `ifFrameLoaded` para iniciar a reprodução de uma animação simples enquanto o resto do filme é descarregado para o computador local. A diferença entre usar `_framesloaded` e `ifFrameLoaded` é que `_framesloaded` permite que você adicione comandos `if` ou `else`, e a ação `ifFrameLoaded` permite que você especifique um número de quadros em um comando simples.

### Exibidor

Flash 3 ou posterior. A ação `ifFrameLoaded` está obsoleta no Flash 5; o uso da ação `_framesloaded` é encorajado.

### Consulte também

`_framesloaded`, na página 271

## #include

### Sintaxe

```
#include "nome_do_arquivo.as";
```

### Argumentos

*nome\_do\_arquivo.as* O nome do arquivo a ser incluído; `.as` é a extensão de arquivo recomendada.

### Descrição

Ação; inclui o conteúdo do arquivo especificado no argumento quando o filme é testado, publicado ou exportado. A ação `#include` é chamada quando você testa, publica ou exporta. A ação `#include` é verificada quando ocorre uma verificação de sintaxe.

### Exibidor

N/A

## Infinity

### Sintaxe

`Infinity`

### Argumentos

Nenhum.

### Descrição

Variável de alto nível; uma variável predefinida com o valor ECMA-262 para `infinity`.

### Exibidor

Flash 5 ou posterior.

## int

### Sintaxe

`int(valor);`

### Argumentos

*valor* Um número a ser arredondado para um inteiro.

### Descrição

Função; converte um número decimal no valor inteiro mais próximo.

### Exibidor

Flash 4 ou posterior. Esta função está obsoleta no Flash 5; o uso do método `Math.floor` é recomendado.

### Consulte também

`Math.floor`, na página 303

## isFinite

### Sintaxe

`isFinite(expressão);`

### Argumentos

*expressão* O booleano, a variável ou outra expressão a ser avaliada.

### Descrição

Função de alto nível; avalia o argumento e retorna `true` se for um número finito e `false` se for um número infinito ou infinito negativo. A presença do infinito ou infinito negativo indica uma condição de erro matemático como uma divisão por 0.

#### Exibidor

Flash 5 ou posterior.

#### Exemplo

A seguir são mostrados exemplos dos valores retornados por `isFinite`:

`isFinite(56)` retorna `true`

`isFinite(Number.POSITIVE_INFINITY)` retorna `false`

`isNaN(Number.POSITIVE_INFINITY)` retorna `false`

## isNaN

#### Sintaxe

`isNaN(expressão)`;

#### Argumentos

*expressão* O booleano, a variável ou outra expressão a ser avaliada.

#### Descrição

Função de alto nível; avalia o argumento e retorna `true` se o valor não for um número (NaN), que indica a presença de erros matemáticos.

#### Exibidor

Flash 5 ou posterior.

#### Exemplo

O exemplo a seguir ilustra o valor retornado por `isNaN`:

`isNaN("Tree")` retorna `true`

`isNaN(56)` retorna `false`

`isNaN(Number.POSITIVE_INFINITY)` retorna `false`

## Key (objeto)

O objeto Key é um objeto de alto nível que você pode acessar sem usar um construtor. Use os métodos do objeto Key para criar uma interface que pode ser controlada por um usuário com um teclado padrão. As propriedades do objeto Key são constantes que representam as teclas mais comumente usadas para controlar jogos. Consulte o Apêndice B, “Teclas do Teclado e Valores do Código de Tecla”, para obter uma lista completa dos principais valores de código das teclas.

### Exemplo

```
onClipEvent (enterFrame) {
    if(Key.isDown(Key.RIGHT)) {
        setProperty ("", _x, _x+10);
    }
}
ou
onClipEvent (enterFrame) {
    if(Key.isDown(39)) {
        setProperty("", _x, _x+10);
    }
}
```

### Resumo dos métodos para o objeto Key

Método	Descrição
getAscii;	Retorna o valor ASCII da última tecla pressionada.
getCode;	Retorna o código de tecla virtual da última tecla pressionada.
isDown;	Retorna true se a tecla especificada no argumento estiver pressionada.
isToggled;	Retorna true se a tecla Num Lock ou Caps Lock estiver ativada.



## Resumo das propriedades do objeto Key

Todas as propriedades do objeto Key são constantes.

Propriedade	Descrição
BACKSPACE	Constante associada ao valor do código de tecla da tecla Backspace (9).
CAPSLCK	Constante associada ao valor do código de tecla da tecla Caps Lock (20).
CONTROL	Constante associada ao valor do código de tecla da tecla Control (17).
DELETEKEY	Constante associada ao valor do código de tecla da tecla Delete (46).
DOWN	Constante associada ao valor do código de tecla da tecla Down Arrow (40).
END	Constante associada ao valor do código de tecla da tecla End (35).
ENTER	Constante associada ao valor do código de tecla da tecla Enter (13).
ESCAPE	Constante associada ao valor do código de tecla da tecla Escape (27).
HOME	Constante associada ao valor do código de tecla da tecla Home (36).
INSERT	Constante associada ao valor do código de tecla da tecla Insert (45).
LEFT	Constante associada ao valor do código de tecla da tecla Left Arrow (37).
PGDN	Constante associada ao valor do código de tecla da tecla Page Down (34).
PGUP	Constante associada ao valor do código de tecla da tecla Page Up (33).
RIGHT	Constante associada ao valor do código de tecla da tecla Right Arrow (39).
SHIFT	Constante associada ao valor do código de tecla da tecla Shift (16).
SPACE	Constante associada ao valor do código de tecla da Barra de espaços (32).
TAB	Constante associada ao valor do código de tecla da tecla Tab (9).
UP	Constante associada ao valor do código de tecla da tecla Up Arrow (38).



## Key.BACKSPACE

### Sintaxe

Key.BACKSPACE

### Argumentos

Nenhum.

### Descrição

Propriedade; constante associada ao valor do código de tecla da tecla Backspace (9).

### Exibidor

Flash 5 ou posterior.

## Key.CAPSLOCK

### Sintaxe

Key.CAPSLOCK

### Argumentos

Nenhum.

### Descrição

Propriedade; constante associada ao valor do código de tecla da tecla Caps Lock (20).

### Exibidor

Flash 5 ou posterior.

## Key.CONTROL

### Sintaxe

Key.CONTROL

### Argumentos

Nenhum.

### Descrição

Propriedade; constante associada ao valor do código de tecla da tecla Control (17).

### Exibidor

Flash 5 ou posterior.

## Key.DELETEKEY

### Sintaxe

Key.DELETEKEY

### Argumentos

Nenhum.

### Descrição

Propriedade; constante associada ao valor do código de tecla da tecla Delete (46).

### Exibidor

Flash 5 ou posterior.

## Key.DOWN

### Sintaxe

Key.DOWN

### Argumentos

Nenhum.

### Descrição

Propriedade; constante associada ao valor do código de tecla da tecla Down Arrow (40).

### Exibidor

Flash 5 ou posterior.

## Key.END

### Sintaxe

Key.END

### Argumentos

Nenhum.

### Descrição

Propriedade; constante associada com o valor do código de chave da tecla End (35).

### Exibidor

Flash 5 ou posterior.

## Key.ENTER

### Sintaxe

Key.ENTER

### Argumentos

Nenhum.

### Descrição

Propriedade; constante associada ao valor do código de tecla da tecla Enter (13).

### Exibidor

Flash 5 ou posterior.

## Key.ESCAPE

### Sintaxe

Key.ESCAPE

### Argumentos

Nenhum.

### Descrição

Propriedade; constante associada ao valor do código de tecla da tecla Escape (27).

### Exibidor

Flash 5 ou posterior.

## Key.getAscii

### Sintaxe

Key.getAscii();

### Argumentos

Nenhum.

### Descrição

Método; retorna o código ASCII da última tecla pressionada ou liberada.

### Exibidor

Flash 5 ou posterior.

## Key.getCode

### Sintaxe

Key.getCode();

### Argumentos

Nenhum.

### Descrição

Método; retorna o valor do código de tecla da última tecla pressionada. Use as informações no Apêndice B, “Teclas do Teclado e Valores de Código de Teclas”, para corresponder o valor de código de tecla retornado com a tecla virtual em um teclado padrão.

### Exibidor

Flash 5 ou posterior.

## Key.HOME

### Sintaxe

Key.HOME

### Argumentos

Nenhum.

### Descrição

Propriedade; constante associada ao valor do código de tecla da tecla Home (36).

### Exibidor

Flash 5 ou posterior.

## Key.INSERT

### Sintaxe

Key.INSERT

### Argumentos

Nenhum.

### Descrição

Propriedade; constante associada ao valor do código de tecla da tecla Insert (45).

### Exibidor

Flash 5 ou posterior.



## Key.isDown

### Sintaxe

`Key.isDown(código_de_tecla);`

### Argumentos

*código\_de\_tecla* O valor do código de tecla atribuído a uma tecla específica ou à propriedade do objeto Key associada a uma tecla específica. O Apêndice B, “Teclas do Teclado e Valores do Código de Tecla”, lista todos os códigos de tecla associados à teclas em um teclado padrão.

### Descrição

Método; retorna `true` se a tecla especificada em *código\_de\_tecla* é pressionada. No Macintosh, os valores do código de tecla das teclas Caps Lock e Num Lock keys são idênticos.

### Exibidor

Flash 5 ou posterior.

## Key.isToggled

### Sintaxe

`Key.isToggled(código_de_tecla)`

### Argumentos

*código\_de\_tecla* O código de tecla da tecla Caps Lock (20) ou Num Lock (144).

### Descrição

Método; retorna `true` se a tecla Caps Lock ou Num Lock estiver ativada (ou for alternada). No Macintosh, os valores de código de tecla para essas teclas são idênticos.

### Exibidor

Flash 5 ou posterior.

## Key.LEFT

### Sintaxe

`Key.LEFT`

### Argumentos

Nenhum.

### Descrição

Propriedade; constante associada ao valor do código de tecla para a tecla Left Arrow (37).

### Exibidor

Flash 5 ou posterior.

## Key.PGDN

### Sintaxe

Key.PGDN

### Argumentos

Nenhum.

### Descrição

Propriedade; constante associada ao valor do código de tecla para a tecla Page Down (34).

### Exibidor

Flash 5 ou posterior.

## Key.PGUP

### Sintaxe

Key.PGUP

### Argumentos

Nenhum.

### Descrição

Propriedade; constante associada ao valor do código de tecla da tecla Page Up (33).

### Exibidor

Flash 5 ou posterior.

## Key.RIGHT

### Sintaxe

Key.RIGHT

### Argumentos

Nenhum.

### Descrição

Propriedade; constante associada ao valor do código de tecla da tecla Right Arrow (39).

### Exibidor

Flash 5 ou posterior.

## Key.SHIFT

### Sintaxe

Key.SHIFT

### Argumentos

Nenhum.

### Descrição

Propriedade; constante associada ao valor do código de tecla da tecla Shift (16).

### Exibidor

Flash 5 ou posterior.

## Key.SPACE

### Sintaxe

Key.SPACE

### Argumentos

Nenhum.

### Descrição

Propriedade; constante associada ao valor do código de tecla da Barra de espaços (32).

### Exibidor

Flash 5 ou posterior.

## Key.TAB

### Sintaxe

Key.TAB

### Argumentos

Nenhum.

### Descrição

Propriedade; constante associada ao valor do código de tecla da tecla Tab (9).

### Exibidor

Flash 5 ou posterior.



## Key.UP

### Sintaxe

Key.UP

### Argumentos

Nenhum.

### Descrição

Propriedade; constante associada ao valor do código de tecla da tecla Up Arrow (38).

### Exibidor

Flash 5 ou posterior.

## le (menor que ou igual a – específico da seqüência de caracteres)

### Sintaxe

*expressão1* le *expressão2*

### Argumentos

*expressão1*, *expressão2* Números, seqüências de caracteres ou variáveis.

### Descrição

Operador (comparação); compara a *expressão1* com a *expressão2* e retorna true se *expressão1* for menor ou igual a *expressão2*; caso contrário, retorna false.

### Exibidor

Flash 4 ou posterior. Este operador está obsoleto no Flash 5; o uso do novo operador <= é recomendado.

### Consulte também

<= (menor ou igual a), na página 213

## length

### Sintaxe

length(*expressão*);

length(*variável*);

### Argumentos

*expressão* Qualquer seqüência de caracteres.

*variável* O nome de uma variável.





### Descrição

Função de sequência de caracteres; retorna o comprimento da sequência de caracteres ou do nome da variável especificada.

### Exibidor

Flash 4 ou posterior. Esta função, juntamente com todas as funções de sequência de caracteres, estão obsoletas no Flash 5. Recomenda-se que você use os métodos e a propriedade `length` do objeto String para executar as mesmas operações.

### Exemplo

O exemplo a seguir retorna o valor da sequência de caracteres Hello:

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

O resultado é 5.

### Consulte também

" " (delimitador de sequência de caracteres), na página 371  
`String.length`, na página 377

## \_level

### Sintaxe

```
_levelN;
```

### Argumentos

*N* Um inteiro não negativo que especifica o nível de intensidade. Por padrão, `_level` é definido como 0, o filme na base da hierarquia.

### Descrição

Propriedade; uma referência ao filme raiz Linha de Tempo de `levelN`.

Carregue os filmes usando a ação `loadMovie`, antes de destiná-los através da propriedade `_level`.

No Flash Player, os filmes são atribuídos a um número conforme a ordem na qual são carregados. O filme carregado primeiro é carregado no nível mais baixo, o nível 0. O filme no nível 0 define a taxa de quadros, cor de fundo e tamanho de quadro de todos os filmes carregados em seguida. Em seguida, os filmes são empilhados em camadas numeradas de forma crescente acima do filme no nível 0. O nível no qual um clipe de filme reside também é conhecido como o nível da intensidade ou intensidade.

### Exibidor

Flash 4 ou posterior.



### Exemplo

O exemplo a seguir encerra a Linha de Tempo do filme no nível 0.

```
_level0.stop();
```

O exemplo a seguir envia a Linha de Tempo do filme no nível 4 para o quadro 5.

O filme no nível 4 deve ter sido carregado previamente com uma ação `loadMovie`

```
_level4.gotoAndStop(5);
```

### Consulte também

`loadMovie`, na página 294

`MovieClip.swapDepths`, na página 326

## loadMovie

### Sintaxe

```
loadMovie(url [,local/destino, variáveis]);
```

### Argumentos

*url* Uma URL absoluta ou relativa do arquivo SWF a ser carregado.

Um caminho relativo deve ser relativo ao SWF. A URL deve estar no mesmo subdomínio da URL na qual o filme reside no momento. Para uso no Flash Player ou para testes no modo de teste de filme no ambiente de criação do Flash, todos os arquivos SWF devem ser armazenados na mesma pasta, e os nomes de arquivo não podem incluir especificações de pasta ou unidade de disco.

*destino* Um argumento opcional que especifica um clipe de filme de destino que será substituído pelo filme carregado. O filme carregado herda as propriedades de posição, rotação e dimensionamento do clipe de filme de destino. Especificar um *destino* é o mesmo que especificar o *local* (nível) de um filme de destino; não é necessário especificar os dois.

*local* Um argumento opcional que especifica o nível no qual o filme é carregado. O filme carregado herda as propriedades de posição, rotação e dimensionamento do clipe de filme de destino. Para carregar o novo filme além dos filmes existentes, especifique um nível que não esteja ocupado por outro filme. Para substituir um filme existente por um filme carregado, especifique um nível que esteja sendo ocupado por outro filme no momento. Para substituir o filme original e descarregar todos os níveis, carregue o novo filme no nível 0. O filme no nível 0 define a taxa de quadros, cor de fundo e tamanho do quadro de todos os outros filmes carregados.

*variáveis* Um argumento opcional que especifica um método para enviar variáveis associadas ao filme a ser carregado. O argumento deve ser a seqüência de caracteres “GET” ou “POST”. Se não houver variáveis, omita esse argumento; caso contrário, especifique se deseja carregar variáveis usando um método GET ou POST. GET anexa as variáveis ao fim da URL e é usado para pequenos números de variáveis. POST envia as variáveis em um cabeçalho HTTP em separado e é usado para longas seqüências de caracteres de variáveis.



### Descrição

Ação; reproduz filmes adicionais sem fechar o Flash Player. Normalmente, o Flash Player exibe um único filme do Flash Player (arquivo SWF) e, depois disso, é fechado. A ação `loadMovie` permite que você exiba vários filmes de uma vez ou alterne entre os filmes sem carregar outro documento HTML.

Você pode carregar filmes no nível no qual os arquivos SWF já estão carregados. Se você o fizer, o novo filme irá substituir o arquivo SWF existente. Se você carregar um novo filme no nível 0, cada nível é descarregado e o nível 0 é substituído pelo novo filme. Use a ação `loadVariables` para manter o filme ativo e atualize as variáveis com os novos valores.

Use a ação `unloadMovie` para remover os filmes carregados com a ação `loadMovie`.

### Exibidor

Flash 3 ou posterior.

### Exemplo

Este comando `loadMovie` é anexado a um botão de navegação chamado `Produtos`. Há um clipe de filme invisível no Palco com o nome de instância `dropZone`. A ação `loadMovie` usa este clipe de filme como o parâmetro de destino para carregar os produtos no arquivo SWF, na posição correta no Palco.

```
on(release) {  
    loadMovie("products.swf",_root.dropZone);  
}
```

### Consulte também

`unloadMovie`, na página 387

`_level`, na página 293

## loadVariables

### Sintaxe

```
loadVariables (url [,local [, variáveis]]);
```

### Argumentos

*url* Uma URL absoluta ou relativa na qual as variáveis estão localizadas. O host da URL deve estar no mesmo subdomínio que o filme quando acessado através de um navegador da web.

*local* Um nível ou destino a receber as variáveis. No Flash Player, os arquivos de filme são atribuídos a um número conforme a ordem na qual são carregados. O primeiro filme é carregado no nível mais baixo (nível 0). Dentro da ação `loadMovie`, especifique um número de nível para cada filme sucessivo. Esse argumento é opcional.

*variáveis* Um argumento opcional que especifica um método para enviar variáveis. Se não houver variáveis, omita esse argumento; caso contrário, especifique se deseja carregar variáveis usando um método GET ou POST. O GET anexa as variáveis no final da URL e é usado para pequenos números de variáveis. O POST envia as variáveis em um cabeçalho HTTP em separado e é usado para maiores seqüências de caracteres de variáveis.

#### Descrição

Ação; lê os dados de um arquivo externo, como um arquivo de texto ou texto gerado por um script CGI, Active Server Pages (ASP) ou Personal Home Page (PHP), e define os valores das variáveis em um filme ou clipe de filme. Essa ação também pode ser usada para atualizar as variáveis no filme ativo com novos valores.

O texto na URL especificada deve ter o formato MIME padrão *aplicativn/x-www-formato de url codificado* (um formato padrão usado por scripts CGI). O filme e as variáveis a serem carregadas devem residir no mesmo subdomínio. Qualquer número de variáveis pode ser especificado. Por exemplo, a frase abaixo define várias variáveis:

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

#### Exibidor

Flash 4 ou posterior.

#### Exemplo

Este exemplo carrega informações de um arquivo de texto em campos de texto na Linha de Tempo principal (nível 0). Os nomes das variáveis dos campos de texto devem corresponder aos nomes das variáveis no arquivo data.txt.

```
on(release) {  
    loadVariables("data.txt", 0);  
}
```

#### Consulte também

getURL, na página 275

MovieClip.loadMovie, na página 321

MovieClip.loadVariables, na página 322

## lt (menor que – sequência de caracteres específica)

### Sintaxe

*expressão1* lt *expressão2*

### Argumentos

*expressão1*, *expressão2* Números, seqüências de caracteres ou variáveis.

### Descrição

Operador (comparação); compara *expressão1* com a *expressão2* e retorna true se a *expressão1* for menor que *expressão2*; caso contrário, retorna false.

### Exibidor

Flash 4 ou posterior. Esse operador está obsoleto no Flash 5; o uso do novo operador < (menor que) é recomendado.

### Consulte também

< (menor que), na página 210

## Math (objeto)

O objeto Math é um objeto de alto nível que você pode acessar sem usar um construtor.

Use os métodos e propriedades desse objeto para acessar e manipular constantes e funções matemáticas. Todas as propriedades e métodos do objeto Math são estáticos e devem ser chamados com a sintaxe `Math.method(argumento)` ou `Math.constant`. No ActionScript, as constantes são definidas com a precisão máxima dos números de ponto flutuante IEEE 754 de precisão dupla.

O objeto Math é totalmente suportado no Flash 5 Player. No Flash 4 Player, os métodos do objeto Math funcionam, mas eles são emulados usando-se aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash 5 Player.

Vários métodos do objeto Math usam o radiano de um ângulo como um argumento. Você pode usar a equação abaixo para calcular os valores radianos ou simplesmente passar a equação (inserindo um valor para graus) para o argumento radiano.

Para calcular um valor radiano, use esta fórmula:

`radiano = Math.PI/180 * grau`

O exemplo a seguir mostra a passagem de uma equação como um argumento para calcular o seno de um ângulo de 45 graus:

`Math.SIN(Math.PI/180 * 45)` é o mesmo que `Math.SIN(.7854)`



## Resumo de métodos do objeto Math

Método	Descrição
abs	Calcula um valor absoluto.
acos	Calcula um arco cosseno.
asin	Calcula um arco seno.
atan	Calcula um arco tangente.
atan2	Calcula um ângulo do eixo x ao ponto.
ceil	Arredonda um número para o inteiro mais próximo.
cos	Calcula um cosseno.
exp	Calcula um valor exponencial.
floor	Arredonda um número para o inteiro mais próximo.
log	Calcula um logaritmo natural.
max	Retorna o maior de dois inteiros.
min	Retorna o menor de dois inteiros.
pow	Calcula $x$ elevado a potência de $y$ .
random	Retorna um número pseudo-aleatório entre 0.0 e 1.0.
round	Arredonda para o inteiro mais próximo.
sin	Calcula um seno.
sqrt	Calcula uma raiz quadrada.
tan	Calcula uma tangente.



## Resumo de propriedades do objeto Math

Todas as propriedades do objeto Math são constantes.

Propriedade	Descrição
E	Constante de Euler e a base de logaritmos naturais (aproximadamente 2,718).
LN2	O logaritmo natural de 2 (aproximadamente 0,693).
LOG2E	O logaritmo de base 2 de e (aproximadamente 1,442).
LN10	O logaritmo natural de 10 (aproximadamente 2,302).
LOG10E	O logaritmo de base 10 de e (aproximadamente 0,434).
PI	A razão entre a circunferência de um círculo e o seu diâmetro (aproximadamente 3,14159).
SQRT1_2	O inverso da raiz quadrada de 1/2 (aproximadamente 0,707).
SQRT2	A raiz quadrada de 2 (aproximadamente 1,414).

## Math.abs

### Sintaxe

`Math.abs(x);`

### Argumentos

*x* Qualquer número.

### Descrição

Método; calcula e retorna um valor absoluto do número especificado pelo argumento *x*.

### Exibidor

Flash 5 ou posterior. No Flash 4 Player, os métodos e propriedades do objeto Math são emulados usando-se aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash 5 Player.

## Math.acos

### Sintaxe

`Math.acos(x);`

### Argumentos

*x* Um número de -1.0 a 1.0.



#### Descrição

Método; calcula e retorna o arco cosseno de um número especificado no argumento  $x$ , em radianos.

#### Exibidor

Flash 5 ou posterior. No Flash 4 Player, os métodos e propriedades do objeto Math são emulados usando-se aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash 5 Player.

## Math.asin

#### Sintaxe

`Math.asin(x);`

#### Argumentos

$x$  Um número de -1.0 a 1.0.

#### Descrição

Método; calcula e retorna o arco seno de um número especificado no argumento  $x$ , em radianos.

#### Exibidor

Flash 5 ou posterior. No Flash 4 Player, os métodos e propriedades do objeto Math são emulados usando-se aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash 5 Player.

## Math.atan

#### Sintaxe

`Math.atan(x);`

#### Argumentos

$x$  Qualquer número.

#### Descrição

Método; calcula e retorna o arco tangente do número especificado no argumento  $x$ . O valor retornado está entre o pi negativo dividido por 2 e o pi positivo dividido por 2.

#### Exibidor

Flash 5 ou posterior. No Flash 4 Player, os métodos e propriedades do objeto Math são emulados usando-se aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash 5 Player.



## Math.atan2

### Sintaxe

`Math.atan2(y, x);`

### Argumentos

*x* Um número que especifica a coordenada *x* do ponto.

*y* Um número que especifica a coordenada *y* do ponto.

### Descrição

Método; calcula e retorna o arco tangente de  $y/x$  em radianos. O valor retornado representa o ângulo referente ao cateto oposto de um triângulo retângulo, onde *x* é o cateto adjacente e *y* é o cateto oposto.

### Exibidor

Flash 5 ou posterior. No Flash 4 Player, os métodos e propriedades do objeto Math são emulados usando-se aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash 5 Player.

## Math.ceil

### Sintaxe

`Math.ceil(x);`

### Argumentos

*x* Um número ou expressão.

### Descrição

Método; retorna o teto do número ou expressão especificada. O teto de um número é o número inteiro mais próximo que é maior que ou igual ao número.

### Exibidor

Flash 5 ou posterior. No Flash 4 Player, os métodos e propriedades do objeto Math são emulados usando-se aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash 5 Player.

## Math.cos

### Sintaxe

`Math.cos(x);`

### Argumentos

*x* Um ângulo medido em radianos.

### Descrição

Método; retorna o cosseno (um valor de -1.0 a 1.0) do ângulo especificado pelo argumento *x*. O ângulo *x* deve ser especificado em radianos. Use as informações descritas na introdução do objeto Math para calcular um radiano.

#### Exibidor

Flash 5 ou posterior. No Flash 4 Player, os métodos e propriedades do objeto Math são emulados usando-se aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash 5 Player.

## Math.E

#### Sintaxe

Math.E

#### Argumentos

Nenhum.

#### Descrição

Constante; uma constante matemática para a base de logaritmos naturais, expressos como  $e$ . O valor aproximado de  $e$  é 2,71828.

#### Exibidor

Flash 5 ou posterior. No Flash 4 Player, os métodos e propriedades do objeto Math são emulados usando-se aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash 5 Player.

## Math.exp

#### Sintaxe

Math.exp( $x$ );

#### Argumentos

$x$  O expoente; um número ou uma expressão.

#### Descrição

Método; retorna o valor da base do logaritmo natural ( $e$ ) a potência do expoente especificado no argumento  $x$ . A constante Math.E pode fornecer o valor de  $e$ .

#### Exibidor

Flash 5 ou posterior. No Flash 4 Player, os métodos e propriedades do objeto Math são emulados usando-se aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash 5 Player.

## Math.floor

### Sintaxe

`Math.floor(x);`

### Argumentos

*x* Um número ou uma expressão.

### Descrição

Método; retorna o piso do número ou expressão especificada no argumento *x*. O piso é o inteiro mais próximo que é menor que ou igual ao número ou expressão especificada.

### Exibidor

Flash 5 ou posterior. No Flash 4 Player, os métodos e propriedades do objeto Math são emulados usando-se aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash 5 Player.

### Exemplo

O exemplo a seguir retorna um valor 12.

```
Math.floor(12.5);
```

## Math.log

### Sintaxe

`Math.log(x);`

### Argumentos

*x* Um número ou expressão com um valor maior que 0.

### Descrição

Método; retorna o logaritmo natural do argumento *x*.

### Exibidor

Flash 5 ou posterior. No Flash 4 Player, os métodos e propriedades do objeto Math são emulados usando-se aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash 5 Player.

## Math.LOG2E

### Sintaxe

Math.LOG2E

### Argumentos

Nenhum.

### Descrição

Constante; uma constante matemática do logaritmo de base 2 da constante  $e$  (Math.E), expressa como  $\log_2 e$ , com um valor aproximado de 1,442695040888963387.

### Exibidor

Flash 5 ou posterior. No Flash 4 Player, os métodos e propriedades do objeto Math são emulados usando-se aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash 5 Player.

## Math.LOG10E

### Sintaxe

Math.LOG10E

### Argumentos

Nenhum.

### Descrição

Constante; uma constante matemática para o logaritmo de base 10 da constante  $e$  (Math.E), expressa como  $\log_{10} e$ , com um valor aproximado de 0,43429448190325181667.

### Exibidor

Flash 5 ou posterior. No Flash 4 Player, os métodos e propriedades do objeto Math são emulados usando-se aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash 5 Player.

## Math.LN2

### Sintaxe

Math.LN2

### Argumentos

Nenhum.

### Descrição

Constante; uma constante matemática do logaritmo natural de 2, expressa como  $\log_e 2$ , com um valor aproximado de 0,69314718055994528623.

### Exibidor

Flash 5 ou posterior. No Flash 4 Player, os métodos e propriedades do objeto Math são emulados usando-se aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash 5 Player.

## Math.LN10

### Sintaxe

Math.LN10

### Argumentos

Nenhum.

### Descrição

Constante; uma constante matemática do logaritmo natural de 10, expressa como  $\log_e 10$ , com um valor aproximado de 2,3025850929940459011.

### Exibidor

Flash 5 ou posterior. No Flash 4 Player, os métodos e propriedades do objeto Math são emulados usando-se aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash 5 Player.

## Math.max

### Sintaxe

`Math.max(x, y);`

### Argumentos

*x* Um número ou expressão.

*y* Um número ou expressão.

### Descrição

Método; avalia *x* e *y* e retorna o maior valor.

### Exibidor

Flash 5 ou posterior. No Flash 4 Player, os métodos e propriedades do objeto Math são emulados usando-se aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash 5 Player.

## Math.min

### Sintaxe

`Math.min(x, y);`

### Argumentos

*x* Um número ou expressão.

*y* Um número ou expressão.

### Descrição

Método; avalia *x* e *y* e retorna o menor valor.

### Exibidor

Flash 5 ou posterior. No Flash 4 Player, os métodos e propriedades do objeto Math são emulados usando-se aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash 5 Player.

## Math.PI

### Sintaxe

`Math.PI`

### Argumentos

Nenhum.

### Descrição

Constante; uma constante matemática da razão entre a circunferência de um círculo e o seu diâmetro expressa como pi, com um valor de 3,14159265358979.

### Exibidor

Flash 5 ou posterior. No Flash 4 Player, os métodos e propriedades do objeto `Math` são emulados usando-se aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash 5 Player.

## Math.pow

### Sintaxe

`Math.pow(x, y);`

### Argumentos

*x* Um número a ser elevado a uma potência.

*y* Um número que especifica a potência à qual o argumento *x* é elevado.

### Descrição

Método; calcula e retorna *x* a potência de *y*,  $x^y$ .

### Exibidor

Flash 5 ou posterior. No Flash 4 Player, os métodos e propriedades do objeto `Math` são emulados usando-se aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash 5 Player.

## Math.random

### Sintaxe

`Math.random();`

### Argumentos

Nenhum.

### Descrição

Método; retorna um número pseudo-aleatório entre 0.0 e 1.0.



#### Exibidor

Flash 5 ou posterior. No Flash 4 Player, os métodos e propriedades do objeto Math são emulados usando-se aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash 5 Player.

**Consulte também**  
random, na página 350

## Math.round

#### Sintaxe

`Math.round(x);`

#### Argumentos

*x* Qualquer número.

#### Descrição

Método; arredonda o valor do argumento *x* para cima ou para baixo para o inteiro mais próximo e retorna o valor.

#### Exibidor

Flash 5 ou posterior. No Flash 4 Player, os métodos e propriedades do objeto Math são emulados usando-se aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash 5 Player.

## Math.sin

#### Sintaxe

`Math.sin(x);`

#### Argumentos

*x* Um ângulo medido em radianos.

#### Descrição

Método; calcula e retorna o seno do ângulo especificado em radianos. Use as informações descritas na introdução do objeto Math para calcular um radiano.

#### Exibidor

Flash 5 ou posterior. No Flash 4 Player, os métodos e propriedades do objeto Math são emulados usando-se aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash 5 Player.

#### Consulte também

Math (objeto), na página 297



## Math.sqrt

### Sintaxe

`Math.sqrt(x);`

### Argumentos

*x* Qualquer número ou expressão maior que ou igual a 0.

### Descrição

Método; calcula e retorna a raiz quadrada do número especificado.

### Exibidor

Flash 5 ou posterior. No Flash 4 Player, os métodos e propriedades do objeto Math são emulados usando-se aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash 5 Player.

## Math.SQRT1\_2

### Sintaxe

`Math.SQRT1_2`

### Argumentos

Nenhum.

### Descrição

Constante; uma constante matemática do inverso da raiz quadrada de meio (1/2), com um valor aproximado de 0,707106781186.

### Exibidor

Flash 5 ou posterior. No Flash 4 Player, os métodos e propriedades do objeto Math são emulados usando-se aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash 5 Player.

## Math.SQRT2

### Sintaxe

`Math.SQRT2`

### Argumentos

Nenhum.

### Descrição

Constante; uma constante matemática representando a raiz quadrada de 2, expressa como, com um valor aproximado de 1,414213562373.

### Exibidor

Flash 5 ou posterior. No Flash 4 Player, os métodos e propriedades do objeto Math são emulados usando-se aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash 5 Player.

## Math.tan

### Sintaxe

`Math.tan(x);`

### Argumentos

*x* Um ângulo medido em radianos.

### Descrição

Método; calcula e retorna a tangente do ângulo especificado. Use as informações descritas na introdução do objeto Math para calcular um radiano.

### Exibidor

Flash 5 ou posterior. No Flash 4 Player, os métodos e propriedades do objeto Math são emulados usando-se aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash 5 Player.

## maxscroll

### Sintaxe

`nome_da_variável.maxscroll = x`

### Argumentos

*variable\_name* O nome de uma variável associada a um campo de texto.

*x* O número de linha que é o valor máximo permitido para a propriedade `scroll`, com base na altura do campo de texto. É um valor somente leitura definido pelo Flash.

### Descrição

Propriedade; uma propriedade somente leitura que funciona com a propriedade `scroll` para controlar a exibição de informações em um campo de texto. Esta propriedade pode ser recuperada, mas não modificada.

### Exibidor

Flash 4 ou posterior.

### Consulte também

`scroll`, na página 354

## mbchr

### Sintaxe

`mbchr(número);`

### Argumentos

*número* O número a ser convertido em um caractere de vários bytes.

### Descrição

Função String; converte um número de código ASCII em um caractere de vários bytes.

### Exibidor

Flash 4 ou posterior. Essa função está obsoleta no Flash 5; o uso do método `String.fromCharCode` é encorajado.

### Consulte também

`String.fromCharCode`, na página 375

## mblength

### Sintaxe

`mblength(seqüência de caracteres);`

### Argumentos

*seqüência de caracteres* Uma seqüência de caracteres.

### Descrição

Função String; retorna o tamanho da seqüência de caracteres de vários bytes.

### Exibidor

Flash 4 ou posterior. Essa função está obsoleta no Flash 5; o uso do objeto e métodos String é recomendado.

## mbord

### Sintaxe

`mbord(caractere);`

### Argumentos

*caractere* O caractere a ser convertido em um número de vários bytes.

### Descrição

Função String; converte o caractere especificado em um número de vários bytes.

#### Exibidor

Flash 4 ou posterior. Essa função está obsoleta Flash 5; o uso do método `String.charCodeAt` é recomendado.

#### Consulte também

`String.fromCharCode`, na página 375

## mbsubstring

#### Sintaxe

`mbsubstring(valor, indice, contagem);`

#### Argumentos

*valor* A sequência de caracteres de vários bytes da qual extrair uma nova sequência de caracteres de vários bytes.

*indice* O número do primeiro caractere a ser extraído.

*contagem* O número de caracteres a serem incluídos na sequência de caracteres extraída, sem incluir o caractere índice.

#### Descrição

Função `String`; extrai uma nova sequência de caracteres de vários bytes de uma sequência de caracteres de vários bytes.

#### Exibidor

Flash 4 ou posterior. Esta função está obsoleta no Flash 5; o uso do método `string.substr` é recomendado.

#### Consulte também

`String.substr`, na página 378

## Mouse (objeto)

Use os métodos do objeto `Mouse` para ocultar e mostrar o cursor no filme. Por padrão, o cursor fica visível, mas você pode ocultá-lo e implementar um cursor personalizado usando um clipe de filme.

### Resumo do método `Mouse`

Método	Descrição
<code>hide</code>	Oculto o cursor no filme.
<code>show</code>	Exibe o cursor no filme.

## Mouse.hide

### Sintaxe

`Mouse.hide();`

### Argumentos

Nenhum.

### Descrição

Método; oculta o cursor em um filme. Por padrão, o cursor fica visível.

### Exibidor

Flash 5 ou posterior.

### Exemplo

O código a seguir, anexado a um clipe de filme na Linha de Tempo principal, oculta o cursor padrão e define as posições *x* e *y* da instância do clipe de filme `customCursor` como as posições *x* e *y* do mouse na Linha de tempo principal:

```
onClipEvent(enterFrame) {  
    Mouse.hide();  
    customCursorMC_x = _root._xmouse;  
    customCursorMC_y = _root._ymouse;  
}
```

### Consulte também

`_xmouse`, na página 420

`_ymouse`, na página 421

`Mouse.show`, na página 313

## Mouse.show

### Sintaxe

`Mouse.show();`

### Argumentos

Nenhum.

### Descrição

Método; torna o cursor visível em um filme. Por padrão, o cursor fica visível.

### Exibidor

Flash 5 ou posterior.

### Consulte também

`_xmouse`, na página 420

`_ymouse`, na página 421

`Mouse.show`, na página 313

## MovieClip (objeto)

Os métodos do objeto MovieClip fornecem a mesma funcionalidade que as ações padrão dos clipes de filme de destino. Há métodos adicionais para fornecer a funcionalidade que não está disponível quando se usa as ações padrão listadas na categoria Ações do Painel de Ações. Você não precisa usar um método construtor para chamar os métodos do objeto MovieClip; em vez disso, faça referência às instâncias do clipe de filme por nome, usando a sintaxe a seguir:

```
anyMovieClip.play();
anyMovieClip.gotoAndPlay(3);
```

### Resumo de métodos do objeto MovieClip

Método	Descrição
attachMovie	Anexa um filme à biblioteca.
duplicateMovieClip	Duplica o clipe de filme especificado.
getBounds	Retorna as coordenadas x e y mínimas e máximas de um filme em um espaço de coordenadas especificado.
getBytesLoaded	Retorna o número de bytes carregados do clipe de filme especificado.
getBytesTotal	Retorna o tamanho do clipe de filme em bytes.
getUrl	Recupera um documento de uma URL.
globalToLocal	Converte o objeto do ponto das coordenadas do Palco nas coordenadas locais do clipe de filme especificado.
gotoAndPlay	Envia a reprodução para um quadro em específico no clipe de filme e reproduz o filme.
gotoAndStop	Envia a reprodução para um quadro em específico no clipe de filme e encerra o filme.
hitTest	Retorna true se há interseção entre a caixa delimitadora do clipe de filme especificado e a caixa delimitadora do clipe de filme de destino.
loadMovie	Carrega o filme no clipe de filme.
loadVariables	Carrega as variáveis de uma URL ou outro local no clipe de filme.
localToGlobal	Converte um objeto do ponto das coordenadas locais de um clipe de filme nas coordenadas do Palco global.
nextFrame	Envia a reprodução para o próximo clipe de filme.
play	Reproduz o clipe de filme especificado.



Método	Descrição
prevFrame	Envia a reprodução para o quadro anterior do clipe de filme.
removeMovieClip	Remove o clipe de filme da Linha de Tempo se ele foi criado com uma ação ou método duplicateMovieClip ou com o método attachMovie.
startDrag	Especifica um clipe de filme como arrastável e começa a arrastá-lo.
stop	Pára o filme que está sendo reproduzido no momento.
stopDrag	Pára o arraste de qualquer clipe de filme que esteja sendo arrastado.
swapDepths	Troca o nível de intensidade de um filme especificado pelo filme no nível de intensidade especificado.
unloadMovie	Remove um filme carregado com loadMovie.

## MovieClip.attachMovie

### Sintaxe

*anyMovieClip.attachMovie(idName, novo nome, intensidade);*

### Argumentos

*idName* O nome do filme na biblioteca a ser anexado. É o nome inserido no campo Identificador na caixa de diálogo Propriedades de Vinculação do Símbolo.

*novo nome* Um nome de uma instância exclusiva do clipe de filme que está sendo anexado.

*intensidade* Um inteiro que especifica o nível de intensidade no qual o filme é colocado.

### Descrição

Método; cria uma nova instância de um clipe de filme na biblioteca e a anexa ao filme especificado por *anyMovieClip*. Use a ação ou método `removeMovieClip` ou `unloadMovie` para remover um clipe de filme anexado com `attachMovie`.

### Exibidor

Flash 5 ou posterior.

### Consulte também

`removeMovieClip`, na página 351

`unloadMovie`, na página 387

`MovieClip.removeMovieClip`, na página 325

`MovieClip.unloadMovie`, na página 327

## MovieClip.cloneMovieClip

### Sintaxe

```
anyMovieClip.cloneMovieClip(novo nome, intensidade);
```

### Argumentos

*novo nome* Um identificador exclusivo do clipe de filme duplicado.

*intensidade* Um número que especifica o nível de intensidade no qual o filme especificado deve ser colocado.

### Descrição

Método; cria uma instância do clipe de filme especificado enquanto o filme está sendo executado. Os clipes de filme duplicados sempre começam a reprodução no quadro 1, independente do quadro atual do clipe de filme quando o método `cloneMovieClip` é chamado. As variáveis no clipe de filme pai não são copiadas para o clipe de filme duplicado. Se o clipe de filme pai for excluído, o clipe de filme duplicado também o será. Os clipes de filme adicionados com `cloneMovieClip` podem ser excluídos com a ação ou o método `removeMovieClip`.

### Exibidor

Flash 5 ou posterior.

### Consulte também

`removeMovieClip`, na página 351

`MovieClip.removeMovieClip`, na página 325

## MovieClip.getBounds

### Sintaxe

```
anyMovieClip.getBounds(targetCoordinateSpace);
```

### Argumentos

*targetCoordinateSpace* O caminho de destino da Linha de Tempo cujo espaço de coordenadas você deseja usar como ponto de referência.

### Descrição

Método; retorna os valores das coordenadas *x* e *y* mínimos e máximos do `MovieClip` para o espaço de coordenadas de destino especificado no argumento. O objeto de retorno conterá as propriedades {*xMin*, *xMax*, *yMin*, *yMax*}. Use os métodos `localToGlobal` e `globalToLocal` do objeto `MovieClip` para converter as coordenadas locais do clipe de filme em coordenadas do Palco ou as coordenadas do Palco em coordenadas locais, respectivamente.

### Exibidor

Flash 5 ou posterior.





### Exemplo

O exemplo a seguir usa `getBounds` para recuperar a caixa delimitadora da instância `myMovieClip` no espaço de coordenadas do filme principal.

```
myMovieClip.getBounds(._root);
```

### Consulte também

`MovieClip.globalToLocal`, na página 318

`MovieClip.localToGlobal`, na página 323

## MovieClip.getBytesLoaded

### Sintaxe

```
anyMovieClip.getBytesLoaded();
```

### Argumentos

Nenhum.

### Descrição

Método; retorna o número de bytes carregados do objeto do clipe de filme especificado. Como os clipes de filme internos são carregados automaticamente, o resultado de retorno deste método e de `MovieClip.getBytesTotal` será o mesmo se o objeto do clipe de filme especificado fizer referência a um clipe de filme interno. Esse método deve ser usado em filmes carregados. Você pode comparar o valor de `getBytesLoaded` com o valor de `getBytesTotal` para determinar que porcentagem de um filme externo foi carregada.

### Exibidor

Flash 5 ou posterior.

## MovieClip.getBytesTotal

### Sintaxe

```
anyMovieClip.getBytesTotal();
```

### Argumentos

Nenhum.

### Descrição

Método; retorna o tamanho, em bytes, do objeto do clipe de filme especificado. Para os clipes de filme externos, (o clipe raiz ou um clipe de filme que está sendo carregado em um destino ou um nível) o valor de retorno é o tamanho do arquivo SWF.

### Exibidor

Flash 5 ou posterior.

## MovieClip.getURL

### Sintaxe

```
anyMovieClip.getURL(URL [,janela, variáveis]);
```

### Argumentos

**URL** Uma URL a partir da qual obter o documento.

**janela** Um argumento opcional que especifica o nome, quadro ou expressão que especifica a janela ou quadro HTML no qual o documento foi carregado. Você também pode usar um dos seguintes nomes de destino reservados: `_self` especifica o quadro atual no janela atual, `_blank` especifica uma nova janela, `_parent` especifica o pai do quadro atual, `_top` especifica o quadro de alto nível na janela atual.

**variáveis** Um argumento opcional que especifica um método para enviar variáveis associado ao filme a ser carregado. Se não houver variáveis, omite esse argumento; caso contrário, especifique se deseja carregar as variáveis usando um método GET ou POST. GET anexa as variáveis ao final da URL e é usado para pequenos números de variáveis. POST envia as variáveis em um cabeçalho HTTP em separado e é usado para maiores seqüências de caracteres de variáveis.

### Descrição

Método; carrega um documento da URL especificada na janela especificada. O método `getURL` também pode ser usado para passar variáveis para outro aplicativo definido na URL usando o método GET ou POST.

### Exibidor

Flash 5 ou posterior.

## MovieClip.globalToLocal

### Sintaxe

```
anyMovieClip.globalToLocal(ponto);
```

### Argumentos

**ponto** O nome ou identificador de um objeto criado com o objeto genérico Object especificando as coordenadas *x* e *y* como propriedades.

### Descrição

Método; converte o objeto *point* das coordenadas do Palco (global) em coordenadas do clipe de filme (local).

### Exibidor

Flash 5 ou posterior.

### Exemplo

O exemplo a seguir converte as coordenadas  $x$  e  $y$  globais do objeto *point* em coordenadas locais do clipe de filme.

```
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();  
}
```

### Consulte também

`MovieClip.localToGlobal`, na página 323

`MovieClip.getBounds`, na página 316

## MovieClip.gotoAndPlay

### Sintaxe

```
anyMovieClip.gotoAndPlay(quadro);
```

### Argumentos

*quadro* O número do quadro para o qual a reprodução será enviada.

### Descrição

Método; inicia a reprodução do filme no quadro especificado.

### Exibidor

Flash 5 ou posterior.

## MovieClip.gotoAndStop

### Sintaxe

```
anyMovieClip.gotoAndStop(quadro);
```

### Argumentos

*quadro* O número do quadro para o qual a reprodução será enviada.

### Descrição

Método; pára a reprodução do filme no quadro especificado.

### Exibidor

Flash 5 ou posterior.

## MovieClip.hitTest

### Sintaxe

```
anyMovieClip.hitTest(x, y, shapeFlag);
```

```
anyMovieClip.hitTest(destino);
```

### Argumentos

*x* A coordenada *x* da área de acertos no Palco.

*y* A coordenada *y* da área de acertos no Palco.

As coordenadas *x* e *y* são definidas no espaço de coordenadas globais.

*destino* O caminho de destino do área de acertos que pode entrar em interseção ou sobrepor a instância especificada por *anyMovieClip*. Normalmente, *destino* representa um botão ou um campo de entrada de texto.

*shapeFlag* Um valor booleano que especifica se vai avaliar a forma completa da instância especificada (*true*), ou apenas a caixa delimitadora (*false*). Esse argumento só pode ser especificado se a área de acertos for identificada com os argumentos das coordenadas *x* e *y*.

### Descrição

Método; avalia a instância especificada por *anyMovieClip* para ver se ela se sobrepõe ou entra em interseção com a área de acertos identificada pelos argumentos das coordenadas *destino* ou *x* e *y*.

O uso 1 compara as coordenadas *x* e *y* com a forma ou com a caixa delimitadora da instância especificada, de acordo com a definição de *shapeFlag*. Se *shapeFlag* for definido como *true*, somente a área realmente ocupada pela instância no Palco é avaliada e se *x* e *y* se sobrepuserem em algum ponto, um valor *true* é retornado. Isso é útil para determinar se o clipe de filme está dentro de uma área de acertos, ou Hotspot, especificada.

O uso 2 avalia as caixas delimitadoras de *destino* e da instância especificada e retorna *true* se elas se sobrepuserem ou entrarem em interseção em algum ponto.

### Exibidor

Flash 5 ou posterior.

### Exemplo

O exemplo a seguir usa `hitTest` com as propriedades `x_mouse` e `y_mouse` para determinar se o mouse está sobre a caixa delimitadora de destino.

```
if (hitTest( _root._xmouse, _root._ymouse, false));
```

O exemplo a seguir usa `hitTest` para determinar se a `ball` do clipe de filme se sobrepõe ou entra em interseção com o `square` do clipe de filme.

```
if(_root.ball, hitTest(_root.square)){  
trace("ball intersects square");  
}
```

### Consulte também

`MovieClip.localToGlobal`, na página 323

`MovieClip.globalToLocal`, na página 318

`MovieClip.getBounds`, na página 316

## MovieClip.loadMovie

### Sintaxe

```
anyMovieClip.loadMovie(url [,variáveis]);
```

### Argumentos

*url* Uma URL absoluta ou relativa do arquivo SWF a ser carregado.

Um caminho relativo deve ser relativo ao SWF. A URL deve estar no mesmo subdomínio que a URL na qual o filme reside no momento. Para uso no Flash Player ou para teste no modo de teste de filme no ambiente de criação do Flash, todos os arquivos SWF devem ser armazenados na mesma pasta, e os nomes dos arquivos não podem incluir especificações de pasta ou unidade de disco.

*variáveis* Um argumento opcional que especifica um método para enviar variáveis associado ao filme a ser carregado. O argumento deve ser a sequência de caracteres "GET" ou "POST". Se não houver variáveis, omita esse argumento; caso contrário, especifique se deseja carregar as variáveis usando um método GET ou POST. GET anexa as variáveis ao final da URL e é usado para pequenos números de variáveis. POST envia as variáveis em um cabeçalho HTTP em separado e é usado para maiores sequências de caracteres de variáveis.

### Descrição

Método; reproduz filmes adicionais sem fechar o Flash Player. Normalmente, o Flash Player exibe um único filme Flash Player (arquivo SWF) e, depois, é fechado. O método `loadMovie` permite que você exiba vários filmes de uma vez ou alterne entre os filmes sem carregar outro documento HTML.

Use a ação `unloadMovie` para remover os filmes carregados com a ação `loadMovie`.

Use o método `loadVariables` para manter o filme ativo e atualizar as variáveis com os novos valores.

#### Exibidor

Flash 5 ou posterior.

#### Consulte também

`MovieClip.loadVariables`, na página 322

`MovieClip.unloadMovie`, na página 327

## MovieClip.loadVariables

### Sintaxe

```
anyMovieClip.loadVariables(url, variáveis);
```

### Argumentos

*url* A URL absoluta ou relativa do arquivo externo. O host da URL deve estar no mesmo subdomínio que o clipe de filme.

*variáveis* O método para recuperar as variáveis. GET anexa as variáveis ao final da URL, e é usado para pequenos números de variáveis. POST envia as variáveis em um cabeçalho HTTP em separado e é usado para maiores seqüências de caracteres de variáveis.

### Descrição

Método; lê dados de um arquivo externo e define os valores das variáveis em um filme ou clipe de filme. O arquivo externo pode ser um arquivo de texto gerado por um script CGI, Active Server Pages (ASP) ou PHP, e pode conter qualquer número de variáveis.

Esse método também pode ser usado para atualizar variáveis no filme ativo com novos valores.

Esse método requer que o texto na URL esteja no formato MIME padrão: *aplicativo/x-www-url em formato codificado* (formato de scripts CGI).

#### Exibidor

Flash 5 ou posterior.

#### Consulte também

`MovieClip.loadMovie`, na página 321

## MovieClip.localToGlobal

### Sintaxe

*anyMovieClip.localToGlobal(ponto);*

### Argumentos

*ponto* O nome ou identificador de um objeto criado com o objeto Object, que especifica as coordenadas *x* e *y* como propriedades.

### Descrição

Método; converte o objeto *point* das coordenadas do clipe de filme (local) em coordenadas no Palco (global).

### Exibidor

Flash 5 ou posterior.

### Exemplo

O exemplo a seguir converte as coordenadas *x* e *y* do objeto *point* das coordenadas do clipe de filme (local) em coordenadas do Palco (globais). As coordenadas *x* e *y* locais são especificadas usando *xmouse* e *ymouse* para recuperar as coordenadas *x* e *y* da posição do 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();  
}
```

### Consulte também

MovieClip.globalToLocal, na página 318



## MovieClip.nextFrame

### Sintaxe

*anyMovieClip*.nextFrame();

### Argumentos

Nenhum.

### Descrição

Método; envia a reprodução para o próximo quadro do clipe de filme.

### Exibidor

Flash 5 ou posterior.

## MovieClip.play

### Sintaxe

*anyMovieClip*.play();

### Argumentos

Nenhum.

### Descrição

Método; reproduz o clipe de filme.

### Exibidor

Flash 5 ou posterior.

## MovieClip.prevFrame

### Sintaxe

*anyMovieClip*.prevFrame();

### Argumentos

Nenhum.

### Descrição

Método; envia a reprodução do quadro anterior e o pára.

### Exibidor

Flash 5 ou posterior.



## MovieClip.removeMovieClip

### Sintaxe

*anyMovieClip.removeMovieClip();*

### Argumentos

Nenhum.

### Descrição

Método; remove uma instância de clipe de filme criado com a ação `duplicateMovieClip` ou os métodos `duplicateMovieClip` ou `attachMovie` do objeto `MovieClip`.

### Exibidor

Flash 5 ou posterior.

### Consulte também

`MovieClip.loadMovie`, na página 321

`MovieClip.attachMovie`, na página 315

## MovieClip.startDrag

### Sintaxe

*anyMovieClip.startDrag([bloqueio, esquerda, direita, superior, inferior]);*

### Argumentos

*bloqueio* Um valor booleano que especifica se o clipe de filme arrastável é bloqueado no centro da posição do mouse (*true*), ou bloqueado no ponto no qual o usuário clicou no clipe de filme na primeira vez (*false*). Esse argumento é opcional.

*esquerdo, superior, direito, inferior* Valores relativos às coordenadas do clipe de filme pai que especificam um retângulo de restrição para o clipe de filme. Esses argumentos são opcionais.

### Descrição

Método; permite que o usuário arraste o clipe de filme especificado. O filme permanece arrastável até que seja explicitamente encerrado chamando o método `stopDrag` ou até que outro clipe de filme se torne arrastável. Somente um clipe de filme é arrastável de cada vez.

### Exibidor

Flash 5 ou posterior.

### Consulte também

`MovieClip.stopDrag`, na página 326

`_droptarget`, na página 263

## MovieClip.stop

### Sintaxe

*anyMovieClip*.stop();

### Argumentos

Nenhum.

### Descrição

Método; pára o clipe de filme em execução no momento.

### Exibidor

Flash 5 ou posterior.

## MovieClip.stopDrag

### Sintaxe

*anyMovieClip*.stopDrag();

### Argumentos

Nenhum.

### Descrição

Método; encerra uma ação de arraste implementada com o método `startDrag`. Um filme permanece arrastável até que um método `stopDrag` seja adicionado ou até que um outro filme se torne arrastável. Somente um clipe de filme é arrastável de cada vez.

### Exibidor

Flash 5 ou posterior.

### Consulte também

`_droptarget`, na página 263

`MovieClip.startDrag`, na página 325

## MovieClip.swapDepths

### Sintaxe

*anyMovieClip*.swapDepths(*intensidade*);

*anyMovieClip*.swapDepths(*destino*);

### Argumentos

*destino* A instância do clipe de filme cuja intensidade está sendo trocada pela instância especificada em *anyMovieClip*. As duas instâncias devem ter o mesmo clipe de filme pai.

*intensidade* Um número que especifica o nível de intensidade no qual o *anyMovieClip* deve ser colocado.



#### Descrição

Método; troca a ordem do empilhamento (nível de intensidade), ou z, da instância especificada pelo filme especificado pelo argumento *destino*, ou pelo filme que ocupa, no momento, o nível de *intensidade* especificado no argumento. Os dois filmes devem ter o mesmo clipe de filme pai. Trocar o nível de intensidade do clipe de filme tem o efeito de mover um filme para frente ou para trás de outro. Se um filme fica interpolado quando esse método é chamado, a interpolação é encerrada.

#### Exibidor

Flash 5 ou posterior.

#### Consulte também

`_level`, na página 293

## MovieClip.unloadMovie

#### Sintaxe

```
anyMovieClip.unloadMovie();
```

#### Argumentos

Nenhum.

#### Descrição

Método; remove um clipe de filme carregado com os métodos do MovieClip `loadMovie` ou `attachMovie`.

#### Exibidor

Flash 5 ou posterior.

#### Consulte também

`MovieClip.loadMovie`, na página 321

`MovieClip.attachMovie`, na página 315

## \_name

#### Sintaxe

```
nome da instância._name  
nome da instância._name = valor;
```

#### Argumentos

*nome da instância* Um nome da instância de um clipe de filme no qual a propriedade `_name` deve ser definida ou recuperada.

*valor* Uma sequência de caracteres que especifica um nome novo de instância.

#### Descrição

Propriedade; especifica o nome da instância do clipe de filme.

#### Exibidor

Flash 4 ou posterior.

## NaN

### Sintaxe

NaN

### Argumentos

Nenhum.

### Descrição

Variável; uma variável predefinida com o valor IEEE 754 para NaN (Não Número).

### Exibidor

Flash 5 ou posterior.

## ne (diferente – específico de sequência de caracteres)

### Sintaxe

*expressão1* ne *expressão2*

### Argumentos

*expressão1*, *expressão2* Números, seqüências de caracteres ou variáveis.

### Descrição

Operador (comparação); compara a *expressão1* com a *expressão2* e retorna true se a *expressão1* não for igual à *expressão2*; caso contrário, retorna false.

### Exibidor

Flash 4 ou posterior. Esse operador está obsoleto no Flash 5; o uso do novo operador != (diferente) é recomendado.

### Consulte também

!= (diferença), na página 191

## new

### Sintaxe

novo *construtor*();

### Argumentos

*construtor* Uma função seguida por argumentos opcionais em parênteses. Normalmente, a função é o nome do tipo de objeto (por exemplo, Array, Math, Number, Object) a ser criado.

### Descrição

Operador; cria um objeto novo, inicialmente anônimo, chama a função identificada pelo argumento *construtor*, passa os argumentos opcionais em parênteses e passa o objeto criado recentemente como um valor da palavra-chave *this*. A função construtora poderá, então, usar *this* para criar o novo objeto.

A propriedade *\_prototype\_* do objeto da função construtora é copiado para a propriedade *\_proto\_* do novo objeto. Como resultado, o novo objeto suporta todos os métodos e propriedades especificados no objeto *prototype* da função construtora.

### Exibidor

Flash 5 ou posterior.

### Exemplo

O exemplo a seguir cria os objetos *book1* e *book2* usando o novo operador.

```
function Book(nome, preço)
{
    this.name = nome;
    this.price = preço;
}
book1 = new Book("Confederacy of Dunces", 19.95);
book2 = new Book("The Floating Opera", 10.95);
```

### Consulte também

[] (operador de acesso de matriz), na página 203

{} (inicializador de objeto), na página 205

A seção do método *constructor* em uma entrada do objeto.

## newline

### Sintaxe

`newline`

### Argumentos

Nenhum.

### Descrição

Constante; insere um caractere de retorno de carro ( `{ }` ) inserindo uma linha em branco no código do *ActionScript*. Use `newline` para aumentar o espaço para informações recuperadas por uma função ou ação em seu código.

### Exibidor

Flash 4 ou posterior.

## nextFrame

### Sintaxe

`nextFrame();`

### Argumentos

Nenhum.

### Descrição

Ação; envia a reprodução para o próximo quadro e o encerra.

### Exibidor

Flash 2 ou posterior.

### Exemplo

Quando o usuário clica em um botão para o qual a ação `nextFrame` tenha sido atribuída, a reprodução é enviada para o próximo quadro.

```
on (release) {  
    nextFrame(5);  
}
```

## nextScene

### Sintaxe

`nextScene();`

### Argumentos

Nenhum.

### Descrição

Ação; envia a reprodução para o quadro 1 da próxima cena e a encerra.

### Exibidor

Flash 2 ou posterior.

### Exemplo

Essa ação é atribuída a um botão que, quando pressionado e liberado, envia a reprodução para o quadro 1 da próxima cena.

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

## not

### Sintaxe

`not expressão`

### Argumentos

*expressão* Qualquer variável ou outra expressão que converta em um valor booleano.

### Descrição

Operador; executa uma operação NOT lógica no Flash 4 Player.

### Exibidor

Flash 4 ou posterior. Esse operador está obsoleto no Flash 5; o uso do novo operador ! (NOT lógico) é recomendado.

### Consulte também

! (NOT lógico), na página 191

## null

### Sintaxe

`null`

### Argumentos

Nenhum.

### Descrição

Palavra-chave; um valor especial que pode ser atribuído a variáveis, ou retornado por uma função se nenhum dado tiver sido fornecido. Você pode usar `null` para representar os valores ausentes ou não ter um tipo de dados definido.

### Exibidor

Flash 5 ou posterior.

### Exemplo

Em um contexto numérico, `null` avalia até 0. Os testes de igualdade podem ser executados com `null`. Neste comando, um nó de árvore binário não tem filho à esquerda; por isso, o campo do filho à esquerda pode ser definido como `null`.

```
if (tree.left == null) {  
    tree.left = new TreeNode();  
}
```

## Number (função)

### Sintaxe

`Number(expressão);`

### Argumentos

*expressão* A sequência de caracteres, o booleano ou outra expressão a ser convertida em número.

### Descrição

Função; converte o argumento *x* em um número e retorna um valor como a seguir:

Se *x* for um número, o valor de retorno é *x*.

Se *x* for um booleano, o valor de retorno é 1 se *x* for `true` e 0 se *x* for `false`.

Se *x* for uma sequência de caracteres, a função tenta analisar *x* como um número decimal com um expoente inicial opcional, isto é, 1,57505e-3.

Se *x* for indefinido, o valor de retorno é 0.

Essa função é usada para converter os arquivos do Flash 4 que contêm operadores obsoletos que são importados no ambiente de criação do Flash 5. Consulte o operador `&` para obter mais informações.

### Exibidor

Flash 4 ou posterior.

### Consulte também

`Number (objeto)`, na página 332

## Number (objeto)

O objeto `Number` é um objeto wrapper simples do tipo de dados número; isso significa que você pode manipular valores numéricos primitivos usando os métodos e propriedades associados ao objeto `Number`. A funcionalidade fornecida por este objeto é idêntica à do objeto `Number` do JavaScript.

Use o construtor `Number` quando chamar os métodos do objeto `Number`, mas você não precisa usá-lo quando chamar as propriedades do objeto `Number`. Os exemplos a seguir especificam a sintaxe para chamar os métodos e propriedades do objeto `Number`:





Este é um exemplo para chamar o método `toString` do objeto `Number`:

```
myNumber = new Number(1234);  
myNumber.toString();
```

Retorna uma sequência de caracteres contendo uma representação binária do número 1234.

Este é um exemplo de como chamar a propriedade `MIN_VALUE` (também chamada de constante) do objeto `Number`:

```
smallest = Number.MIN_VALUE
```

## Resumo de métodos do objeto Number

Método	Descrição
<code>toString</code>	Retorna a representação da sequência de caracteres do objeto <code>Number</code> .
<code>valueOf</code>	Retorna o valor primitivo do objeto <code>Number</code>

## Resumo das propriedades do objeto Number

Propriedade	Descrição
<code>MAX_VALUE</code>	Constante que representa o maior número representável (IEEE 754 de dupla precisão). Esse número é aproximadamente 1,7976931348623158e+308.
<code>MIN_VALUE</code>	Constante que representa o menor número representável (IEEE 754 de dupla precisão). Esse número é aproximadamente 5e-324.
<code>NaN</code>	Constante que representa o valor um valor Não Número (NaN).
<code>NEGATIVE_INFINITY</code>	Constante que representa o valor do infinito negativo.
<code>POSITIVE_INFINITY</code>	Constante que representa o valor do infinito positivo. Esse valor é o mesmo da variável <code>Infinity</code> global.



## Construtor do objeto Number

### Sintaxe

```
myNumber = new Number(valor);
```

### Argumentos

*valor* O valor numérico do objeto Number que está sendo criado, ou um valor a ser convertido em um número.

### Descrição

Construtor; cria um novo objeto Number. Você deve usar o construtor Number quando estiver usando os métodos `toString` e `valueOf` do objeto Number. Não use um construtor quando estiver usando as propriedades do objeto Number. O construtor `new Number` é usado, basicamente, como um espaço reservado. Uma instância do objeto Number não é a mesma que a função Number que converte um argumento em um valor primitivo.

### Exibidor

Flash 5 ou posterior.

### Exemplo

O código a seguir cria objetos `new Number`.

```
n1 = new Number(3.4);
```

```
n2 = new Number(-10);
```

### Consulte também

Number (função), na página 332.

## Number.MAX\_VALUE

### Sintaxe

```
Number.MAX_VALUE
```

### Argumentos

Nenhum.

### Descrição

Propriedade; o maior número representável (IEEE 754 de dupla precisão). Esse número é aproximadamente 1,79E+308.

### Exibidor

Flash 5 ou posterior.

## Number.MIN\_VALUE

### Sintaxe

Number.MIN\_VALUE

### Argumentos

Nenhum.

### Descrição

Propriedade; o menor número representável (IEEE 754 de dupla precisão).  
Esse número é aproximadamente 5e-324.

### Exibidor

Flash 5 ou posterior.

## Number.NaN

### Sintaxe

Number.NaN

### Argumentos

Nenhum.

### Descrição

Propriedade; o valor IEEE-754 que representa Não Número (NaN).

### Exibidor

Flash 5 ou posterior.

## Number.NEGATIVE\_INFINITY

### Sintaxe

Number.NEGATIVE\_INFINITY

### Argumentos

Nenhum.

### Descrição

Propriedade; retorna o valor IEEE 754 que representa o infinito negativo.  
Esse valor é o mesmo da variável global *Infinity*.

O infinito negativo é um valor numérico especial que é retornado quando uma operação ou função matemática retorna um valor negativo maior do que pode ser representado.

### Exibidor

Flash 5 ou posterior.

## Number.POSITIVE\_INFINITY

### Sintaxe

`Number.POSITIVE_INFINITY`

### Argumentos

Nenhum.

### Descrição

Propriedade; retorna o valor IEEE 754 que representa o infinito positivo. Este valor é o mesmo que a variável global `Infinity`.

O infinito positivo é um valor numérico especial retornado quando uma operação ou função matemática retorna um valor maior do que pode ser representado.

### Exibidor

Flash 5 ou posterior.

## Number.toString

### Sintaxe

`myNumber.toString(raiz);`

### Argumentos

*raiz* Especifica a base numérica (de 2 a 36) a ser usada para a conversão número em sequência de caracteres. Se você não especificar o argumento *raiz*, o valor padrão é 10.

### Descrição

Método; retorna a representação da sequência de caracteres do objeto `Number` especificado (*myNumber*).

### Exibidor

Flash 5 ou posterior.

### Exemplo

O exemplo a seguir usa o método `Number.toString`, especificando 2 para o argumento *raiz*:

```
myNumber = new Number (1000);  
(1000).toString(2);
```

Retorna uma sequência de caracteres que contém o valor da representação binária do número 1000.

## Number.valueOf

### Sintaxe

```
myNumber.valueOf();
```

### Argumentos

Nenhum.

### Descrição

Método; retorna o valor primitivo do objeto Number especificado e converte o objeto Number wrapper em um tipo de valor primitivo.

### Exibidor

Flash 5 ou posterior.

## Object (objeto)

O objeto Object genérico está na raiz da hierarquia de classes do ActionScript. A sua funcionalidade representa um pequeno subconjunto do que é fornecido pelo objeto Object do JavaScript.

O objeto genérico Object requer o Flash 5 Player.

### Resumo de métodos do objeto Object

Método	Descrição
toString	Converte o objeto especificado em uma sequência de caracteres e o retorna.
valueOf	Retorna o valor primitivo do objeto Object.

### Construtor do objeto Object

#### Sintaxe

```
new Object();
```

```
new Object(valor);
```

#### Argumentos

*valor* Um número, booleano ou sequência de caracteres a ser convertido em um objeto. Esse argumento é opcional. Se você não especificar *valor*, o construtor cria um novo objeto com propriedades não definidas.

**Descrição**

Construtor; cria um novo objeto Object.

**Exibidor**

Flash 5 ou posterior.

**Consulte também**

Sound.setTransform, na página 363

Color.setTransform, na página 239

## Object.toString

**Sintaxe**

```
myObject.toString();
```

**Argumentos**

Nenhum

**Descrição**

Método; converte o objeto especificado em uma sequência de caracteres e o retorna.

**Exibidor**

Flash 5 ou posterior.

## Object.valueOf

**Sintaxe**

```
myObject.valueOf();
```

**Argumentos**

Nenhum.

**Descrição**

Método; retorna o valor primitivo do objeto especificado. Se o objeto não tiver um valor primitivo, o objeto é retornado.

**Exibidor**

Flash 5 ou posterior.

## onClipEvent

### Sintaxe

```
onClipEvent(movieEvent){  
...  
}
```

### Argumentos

O *movieEvent* é um evento disparador que executa ações que são atribuídas a uma instância de clipe de filme. Qualquer um dos valores a seguir pode ser especificado pelo argumento *movieEvent*.

- **load** A ação é iniciada assim que o clipe de filme é criado e aparece na Linha de Tempo.
- **unload** A ação é iniciada no primeiro quadro depois do clipe de filme ser removido da Linha de Tempo. As ações associadas ao evento do clipe de filme **Unload** são processadas antes que as ações sejam anexadas ao quadro atingido.
- **enterFrame** A ação é iniciada à medida que cada quadro é reproduzido, da mesma forma que as ações anexadas a um clipe de filme. As ações associadas ao evento do clipe de filme **OnEnterFrame** são processadas depois das ações que tenham sido anexadas aos quadros afetados.
- **mouseMove** A ação é iniciada toda vez que o mouse é movido. Use as propriedades **\_xmouse** e **\_ymouse** para determinar a posição atual do mouse.
- **mouseDown** A ação é iniciada quando o botão do mouse é pressionado.
- **mouseUp** A ação é iniciada quando o botão esquerdo do mouse é liberado.
- **keyDown** A ação é iniciada quando uma tecla é pressionada. Use o método **Key.getCode** para recuperar as informações sobre a última tecla pressionada.
- **keyUp** A ação é iniciada quando uma tecla é liberada. Use o método **Key.getCode** para recuperar informações sobre a última tecla pressionada.
- **data** A ação é iniciada quando os dados são recebidos em uma ação **loadVariables** ou **loadMovie**. Quando especificado com uma ação **loadVariables**, o evento **data** ocorre somente uma vez, quando a última variável é carregada. Quando especificado com uma ação **loadMovie**, o evento **data** ocorre repetidamente, à medida que cada seção de dados é recuperada.

### Descrição

Manipulador; dispara ações definidas por uma instância específica de um clipe de filme.

### Exibidor

Flash 5 ou posterior.

### Exemplo

O comando a seguir inclui o script de um arquivo externo quando a instância do clipe de filme é carregada e aparece pela primeira vez na Linha de Tempo.

```
onClipEvent(load) {  
    #include "myScript.as"  
}
```

O exemplo a seguir usa `onClipEvent` com o evento de filme `keyDown`.

Normalmente, o evento de filme `keyDown` é usado juntamente com um ou mais métodos e propriedades associados ao objeto `Key`. No script abaixo, `key.getCode` é usado para localizar a tecla pressionada pelo usuário, o valor retornado associado às propriedades do objeto `Key` `RIGHT` ou `LEFT`, e o filme é direcionado adequadamente.

```
onClipEvent(load) {  
    if (Key.getCode() == Key.RIGHT) {  
        _parent.nextFrame();  
    }  
    else if (Key.getCode() == Key.LEFT){  
        _parent.prevFrame();  
    }  
}
```

O exemplo a seguir usa `onClipEvent` com o evento de filme `mouseMove`.

As propriedades `xmouse` e `ymouse` controlam a posição do mouse.

```
onClipEvent(mouseMove) {  
    stageX=_root.xmouse;  
    stageY=_root.ymouse;  
}
```

### Consulte também

`on(mouseEvent)`, na página 341

`Key` (objeto), na página 283

`_xmouse`, na página 420

`_ymouse`, na página 421



## on(mouseEvent)

### Sintaxe

```
on(mouseEvent) {  
  comando;  
}
```

### Argumentos

*comando* Os comandos a serem executados quando o evento mouseEvent ocorre

Uma ação *mouseEvent* pode ter um dos seguintes argumentos:

- **press** O botão do mouse é pressionado enquanto o ponteiro está sobre o botão.
- **release** O botão do mouse é liberado enquanto o ponteiro está sobre o botão.
- **releaseOutside** O botão do mouse é liberado enquanto o ponteiro está fora do botão.
- **rollOver** O ponteiro do mouse rola sobre o botão.
- **rollOut** O ponteiro rola fora da área do botão.
- **dragOver** Enquanto o ponteiro está sobre o botão, o botão do mouse é pressionado, rolado para fora do botão e, em seguida, rolado novamente sobre o botão.
- **dragOut** Enquanto o ponteiro está sobre o botão, o botão do mouse é pressionado e rolado para fora da área do botão.
- **keyPress** (“*tecla*”) A *tecla* especificada é pressionada. A parte *tecla* do argumento é especificada usando qualquer um dos códigos de tecla listados no Apêndice B, “Teclas do Teclado e Valores de Código de Tecla” ou qualquer uma das constantes listadas em Resumo das propriedades do objeto Key, na página 284.

### Descrição

Manipulador; especifica o evento do mouse ou o pressionamento de tecla que dispara uma ação.

### Exibidor

Flash 2 ou posterior.



### Exemplo

No script a seguir, a ação `startDrag` é executada quando o mouse é pressionado e o script condicional é executado quando o mouse é liberado e o objeto é ignorado

```
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();
}
```

### Consulte também

Key (objeto), na página 283

onClipEvent, na página 339

## or

### Sintaxe

*condição1* ou *condição2*

### Argumentos

*condição1,2* Uma expressão que pode receber valor `true` ou `false`.

### Descrição

Operador; avalia *condição1* e *condição2* e se alguma das expressões for `true`, toda a expressão será `true`.

### Exibidor

Flash 4 ou posterior. Este operador é obsoleto no Flash 5, e os usuários são encorajados a usar o novo operador `||`.

### Consulte também

`||` (OR), na página 207

## ord

### Sintaxe

`ord(caractere);`

### Argumentos

*caractere* O caractere a ser convertido em um número de código ASCII.

### Descrição

Função de sequência de caracteres; converte caracteres em números de código ASCII.

### Exibidor

Flash 4 ou posterior. Esta função está obsoleta no Flash 5 e recomenda-se o uso dos métodos e propriedades do objeto `String`.

### Consulte também

`String` (objeto), na página 372

## \_parent

### Sintaxe

`_parent.propriedade = x`  
`_parent._parent.propriedade = x`

### Argumentos

*propriedade* A propriedade que está sendo especificada para o clipe de filme pai e atual.

*x* O valor definido para a propriedade. Este é um argumento opcional e pode não precisar ser definido dependendo da propriedade.

### Descrição

Propriedade; especifica ou retorna uma referência ao clipe de filme que contém o clipe de filme atual. O clipe de filme atual é o clipe de filme que contém o script em execução no momento. Use `_parent` para especificar um caminho relativo.

### Exibidor

Flash 4 ou posterior.

### Exemplo

No exemplo a seguir, o clipe de filme `desk` é um filho do clipe de filme `classroom`. Quando o script abaixo é executado dentro do clipe de filme `desk`, a reprodução irá pular para o quadro 10 da Linha de Tempo do clipe de filme `classroom`.

```
_parent.gotoAndStop(10);
```

### Consulte também

`_root`, na página 352

`targetPath`, na página 381

## parseFloat

### Sintaxe

`parseFloat(seqüência de caracteres);`

### Argumentos

*seqüência de caracteres* A seqüência de caracteres a ser analisada e convertida em um número de ponto flutuante.

### Descrição

Função; converte uma seqüência de caracteres em um número de ponto flutuante. A função analisa e retorna os números da seqüência de caracteres até que o analisador alcance um caractere que não seja parte do número inicial. Se a seqüência de caracteres não começar com um número que possa ser analisado, `parseFloat` retorna NaN ou 0. O espaço em branco que precede os inteiros válidos é ignorado, pois são caracteres precedentes não numéricos.

### Exibidor

Flash 5 ou posterior.

### Exemplo

A seguir estão exemplos de uso do `parseFloat` para avaliar os vários tipos de números:

`parseFloat( "-2 " )` retorna -2

`parseFloat( "2.5 " )` retorna 2.5

`parseFloat( "3.5e6 " )` retorna 3.5e6, ou 3500000

`parseFloat( "foobar " )` retorna NaN

## parseInt

### Sintaxe

`parseInt(expressão, raiz);`

### Argumentos

*expressão* A seqüência de caracteres, o número de ponto flutuante ou outra expressão a ser analisada e convertida em um inteiro.

*raiz* Um inteiro que representa a raiz (base) do número a ser analisado. Os valores válidos vão de 2 a 36. Este argumento é opcional.

### Descrição

Função; converte uma seqüência de caracteres em um inteiro. Se a seqüência de caracteres especificada nos argumentos não puder ser convertida em um número, a função retorna NaN ou 0. Os inteiros que começam com 0 ou especificam uma raiz de 8 são interpretados como números octais. Os inteiros que começam com 0x são interpretados como números hexadecimais. O espaço em branco que precede os inteiros válidos é ignorado, pois são caracteres precedentes não numéricos.

**Exibidor**

Flash 5 ou posterior.

**Exemplo**

A seguir são mostrados exemplos do uso de `parseInt` para avaliar os vários tipos de números.

`parseInt("3.5")` retorna 3.5

`parseInt("bar")` retorna NaN

`parseInt("4foo")` retorna 4

**Exemplo**

Conversão hexadecimal:

`parseInt("0x3F8")` retorna 1016

`parseInt("3E8", 16)` retorna 1000

Conversão binária:

`parseInt("1010", 2)` retorna 10 (a representação decimal do binário 1010)

Análise de número octal (nesse caso, o número octal é identificado pela raiz, 8):

`parseInt("777", 8)` retorna 511 (a representação decimal do octal 777)

## play

**Sintaxe**

`play();`

**Argumentos**

Nenhum.

**Descrição**

Ação; move a reprodução para frente na Linha de Tempo.

**Exibidor**

Flash 2 ou posterior.

### Exemplo

O código a seguir usa um comando `if` para verificar o valor de um nome que o usuário insere. Se o usuário inserir `Steve`, a ação `play` é chamada e a reprodução move para frente na Linha de Tempo. Se o usuário inserir qualquer coisa diferente de `Steve`, o filme não é reproduzido e um campo de texto com o nome de variável `alert` é exibido.

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

## prevFrame

### Sintaxe

`prevFrame()`;

### Argumentos

Nenhum.

### Descrição

Ação; envia a reprodução para o quadro anterior e o encerra.

### Exibidor

Flash 2 ou posterior.

### Exemplo

Quando o usuário clicar em um botão para o qual uma ação `prevFrame` é atribuída, a reprodução é enviada ao quadro anterior.

```
on(release) {  
    prevFrame(5);  
}
```

### Consulte também

`MovieClip.prevFrame`, na página 324

## prevScene

### Sintaxe

```
prevScene();
```

### Argumentos

Nenhum.

### Descrição

Ação; envia a reprodução para o quadro 1 da cena anterior e a encerra.

### Exibidor

Flash 2 ou posterior.

### Consulte também

nextScene, na página 330

## print

### Sintaxe

```
print (destino, "bmovie");
```

```
print (destino, "bmax");
```

```
print (destino, "bframe");
```

### Argumentos

*destino* O nome da instância do clipe de filme a ser impresso. Por padrão, todos os quadros do filme são impressos. Se desejar imprimir somente quadros em específico no filme, indique os quadros para impressão anexando um rótulo do quadro #P aos quadros do ambiente de criação.

*bmovie* Indica a caixa delimitadora de um quadro específico em um filme como a área de impressão de todos os quadros imprimíveis no filme. Anexe um rótulo #b (no ambiente de criação) para indicar o quadro cuja caixa delimitadora você deseja usar como a área de impressão.

*bmax* Indica uma composição de todas as caixas delimitadoras, de todos os quadros imprimíveis, como a área de impressão. Especifique o argumento *bmax* quando os quadros imprimíveis em seu filme variarem em tamanho.

*bframe* Indica que a caixa delimitadora de cada quadro imprimível deve ser usada como a área de impressão do quadro. Isso altera a área de impressão de cada quadro e dimensiona os objetos para caberem na área de impressão. Use *bframe* se você tiver objetos de tamanhos diferentes em cada quadro e desejar que cada objeto ocupe toda a página impressa.

### Descrição

Ação; imprime o clipe de filme de *destino* de acordo com o modificador da impressora especificado no argumento. Se você deseja imprimir somente quadros específicos no filme de destino, anexe um rótulo de quadro #P aos quadros que deseja imprimir. Apesar da ação `print` ter como resultado uma qualidade melhor do que a ação `printAsBitmap`, ela não pode ser usada para imprimir filmes que usam transparências alpha ou efeitos de cor especiais.

Se você não especificar um argumento de área de impressão, a área de impressão é determinada pelo tamanho do Palco do filme carregado, por padrão. O filme não herda o tamanho do Palco do filme principal. Você pode controlar a área de impressão especificando os argumentos *bmovie*, *bmax* ou *bframe*.

Todos os elementos imprimíveis em um filme devem ser totalmente carregados antes da impressão começar.

O recurso de impressão do Flash Player suporta as impressoras PostScript e não PostScript. As impressoras não PostScript convertem vetores em bitmaps.

### Exibidor

Flash 5 ou posterior.

### Exemplo

O exemplo a seguir irá imprimir todos os quadros imprimíveis em *myMovie* com a área de impressão definida pela caixa delimitadora do quadro com o rótulo do quadro #b anexado.

```
print("myMovie", "bmovie");
```

O exemplo a seguir irá imprimir todos os quadros imprimíveis em *myMovie* com uma área de impressão definida pela caixa delimitadora de cada quadro.

```
print("myMovie", "bframe");
```

### Consulte também

`printAsBitmap`, na página 348

## printAsBitmap

### Sintaxe

```
printAsBitmap(destino, "bmovie");
```

```
printAsBitmap(destino, "bmax");
```

```
printAsBitmap(destino, "bframe");
```

### Argumentos

*destino* O nome da instância do clipe de filme a ser impresso. Por padrão, todos os quadros do filme são impressos. Se deseja imprimir somente quadros específicos no filme, indique-os anexando um rótulo de quadro #P no ambiente de criação.





*bmovie* Indica a caixa delimitadora de um quadro específico em um filme como a área de impressão de todos os quadros imprimíveis no filme. Anexe um rótulo #b (no ambiente de criação) para indicar o quadro cuja caixa delimitadora deseja usar como área de impressão.

*bmax* Indica uma composição de todas as caixas delimitadores, de todos os quadros imprimíveis, como a área de impressão. Especifique o argumento *bmax* quando os quadros imprimíveis em seu filme variarem em tamanho.

*bframe* Indica uma caixa delimitadora de cada quadro imprimível a ser usada como área de impressão para o quadro. Isso altera a área de impressão de cada quadro e dimensiona os objetos para caberem na área de impressão. Use *bframe* se você tiver objetos de tamanhos diferentes em cada quadro e desejar que cada objeto ocupe toda a página impressa.

#### Descrição

Ação; imprime o clipe de filme de *destino* como um bitmap. Use `printAsBitmap` para imprimir filmes que contenham quadros com objetos que usam transparência ou efeitos de cor. A ação `printAsBitmap` imprime na resolução mais alta disponível da impressora para manter a maior definição e qualidade possível. Para calcular o tamanho do arquivo imprimível de um quadro indicado para ser impresso como um bitmap, multiplique a largura do pixel pela altura do pixel pela resolução da impressora.

Se o seu filme não contiver transparências alpha ou efeitos de cor, recomenda-se o uso da ação `print` para obter qualidade melhor.

Por padrão, a área de impressão é determinada pelo tamanho do Palco do filme carregado. O filme não herda o tamanho do Palco do filme principal. Você pode controlar a área de impressão especificando os argumentos *bmovie*, *bmax* ou *bframe*.

Todos os elementos imprimíveis em um filme devem ser totalmente carregados antes que a impressão possa começar.

O recurso de impressão do Flash Player suporta as impressoras PostScript e não PostScript. As impressoras não PostScript convertem vetores em bitmaps.

#### Exibidor

Flash 5 ou posterior.

#### Consulte também

`print`, na página 347

## **\_quality**

### **Sintaxe**

```
_quality  
_quality = x;
```

### **Argumentos**

*x* Uma seqüência de caracteres que especifica um dos seguintes valores:

**LOW** Qualidade baixa. Os gráficos são apresentados sem serrilhado, os bitmaps não são suavizados.

**MEDIUM** Qualidade média. Os gráficos são apresentados sem serrilhado usando uma grade 2x2, mas os bitmaps não são suavizados. Adequado para filmes que não contêm texto.

**HIGH** Qualidade alta. Os gráficos são apresentados sem serrilhado usando uma grade 4x4 e os bitmaps são suavizados se o filme for estático. Essa é a configuração de qualidade padrão usada pelo Flash.

**BEST** Qualidade muito alta. Os gráficos são apresentados sem serrilhado usando uma grade 4x4 e os bitmaps sempre são suavizados.

### **Descrição**

Propriedade (global); define ou recupera a qualidade usada para um filme.

### **Exibidor**

Flash 5 ou posterior.

### **Exemplo**

O exemplo a seguir define a qualidade de `oldQuality` como **HIGH**.

```
oldQuality = _quality  
_quality = "HIGH";
```

### **Consulte também**

`_highquality`, na página 279

## **random**

### **Sintaxe**

```
random();
```

### **Argumentos**

*valor* O maior inteiro para o qual `random` retornará um valor.

### **Descrição**

Função; retorna um inteiro aleatório entre 0 e o inteiro especificado no argumento *valor*.

#### Exibidor

Flash 4. Esta função está obsoleta no Flash 5; o uso do método `Math.random` é recomendado.

#### Exemplo

No uso a seguir de `random` é retornado um valor de 0, 1, 2, 3 ou 4.

```
random(5);
```

#### Consulte também

`Math.random`, na página 307

## removeMovieClip

#### Sintaxe

```
removeMovieClip(destino);
```

#### Argumentos

*destino* O caminho de destino de uma instância de clipe de filme criada com `duplicateMovieClip`, ou o nome da instância de um clipe de filme criada com os métodos `attachMovie` ou `duplicateMovie` do objeto `MovieClip`.

#### Descrição

Ação; exclui uma instância de clipe de filme criada com os métodos `attachMovie` ou `duplicateMovieClip` do objeto `MovieClip`, ou com a ação `duplicateMovieClip`.

#### Exibidor

Flash 4 ou posterior.

#### Consulte também

`duplicateMovieClip`, na página 264

`MovieClip.duplicateMovieClip`, na página 316

`MovieClip.attachMovie`, na página 315

`MovieClip.removeMovieClip`, na página 325

## return

#### Sintaxe

```
return[expressão];
```

```
return
```

#### Argumentos

*expressão* Um tipo, sequência de caracteres, número, matriz ou objeto a ser avaliado e retornado como um valor da função. Esse argumento é opcional.

### Descrição

Ação; especifica o valor retornado pela função. Quando a ação `return` é executada, a expressão é avaliada e retornada como um valor da função. A ação `return` faz com que a função pare de ser executada. Se o comando `return` for usado sozinho ou se o Flash não encontrar uma ação `return` durante uma ação de repetição, ele retorna `null`.

### Exibidor

Flash 5 ou posterior.

### Exemplo

O seguinte exemplo mostra como usar o `return`:

```
function sum(a, b, c){  
    return a + b + c;  
}
```

### Consulte também

`function`, na página 272

## \_root

### Sintaxe

```
_root;  
_root.movieClip;  
_root.ação;
```

### Argumentos

*movieClip* O nome da instância de um clipe de filme.

*ação* O valor definido para a propriedade. Esse é um argumento opcional e não precisa ser definido dependendo da propriedade.

### Descrição

Propriedade; especifica ou retorna uma referência à Linha de Tempo do filme raiz. Se um filme tem vários níveis, a Linha de Tempo do filme raiz está no nível contido no script sendo executado no momento. Por exemplo, se um script no nível 1 avaliar `_root`, o nível 1 é retornado.

Especificar `_root` é o mesmo que usar a notação de barra “/” para especificar um caminho absoluto dentro do nível atual.

### Exibidor

Flash 4 ou posterior.

### Exemplo

O exemplo a seguir pára a Linha de Tempo do nível que contém o script sendo executado no momento.

```
_root1.stop();
```

O exemplo a seguir envia a Linha de Tempo no nível atual para o quadro 3.

```
_root.gotoAndStop(3);
```

### Consulte também

`_parent`, na página 343

`targetPath`, na página 381

## **`_rotation`**

### Sintaxe

*nome da instância*.`_rotation`

*nome da instância*.`_rotation` = *inteiro*

### Argumentos

*inteiro* O número de graus a girar o clipe de filme.

*nome da instância* O clipe de filme a ser girado.

### Descrição

Propriedade; especifica a rotação do clipe de filme em graus.

### Exibidor

Flash 4 ou posterior.

## scroll

### Sintaxe

`nome_da_variável.scroll = x`

### Argumentos

`nome_da_variável` O nome da variável associada ao campo de texto.

`x` O número da linha da linha visível mais acima no campo de texto. Você pode especificar esse valor ou usar o valor padrão de 1. O Flash Player atualiza esse valor à medida que o usuário navega pelo campo de texto.

### Descrição

Propriedade; controla a exibição de informações em um campo de texto associado a uma variável. A propriedade `scroll` define onde o campo de texto começa exibindo o conteúdo; depois de defini-lo, o Flash Player o atualiza à medida que o usuário rola pelo campo de texto. A propriedade `scroll` é útil para direcionar os usuários para um parágrafo em específico em um trecho longo, ou para criar campos de texto de rolagem. Essa propriedade pode ser recuperada e modificada.

### Exibidor

Flash 4 ou posterior.

### Consulte também

`maxscroll`, na página 310

## Selection (objeto)

O objeto Selection permite que você defina e controle o campo de texto editável focalizado no momento. Esse campo é o campo no qual o cursor do usuário está localizado no momento. Os índices de intervalo de seleção são com base zero (onde a primeira posição é 0, a segunda posição é 1 e assim por diante).

Não há nenhum método construtor para o objeto Selection e só pode haver um campo focalizado no momento de cada vez.



## Resumo de métodos do objeto Selection

Método	Descrição
getBeginIndex	Retorna o índice no começo do intervalo da seleção. Retorna -1 se não houver índice ou campo selecionado no momento.
getCaretIndex	Retorna a posição atual do cursor no intervalo de seleção focalizado. Retorna -1 se não houver posição de cursor ou intervalo de seleção focalizado no momento.
getEndIndex	Retorna o índice no final do intervalo de seleção. Retorna -1 se não houver índice ou campo selecionado no momento.
getFocus	Retorna o nome da variável do campo de texto editável no momento. Retorna null se não houver campo de texto editável no momento.
setFocus	Enfoca o campo de texto editável associado à variável especificada no argumento.
setSelection	Define os índices de início e de fim do intervalo de seleção.

## Selection.getBeginIndex

### Sintaxe

Selection.getBeginIndex();

### Argumentos

Nenhum.

### Descrição

Método; retorna o índice do início do intervalo da seleção. Se não houver índice ou se nenhum campo no momento focalizado, o método retorna -1. Os índices do intervalo de seleção são baseados em zero (onde a primeira posição é 0, a segunda posição é 1 e assim por diante).

### Exibidor

Flash 5 ou posterior.

## Selection.getCaretIndex

### Sintaxe

`Selection.getCaretIndex();`

### Argumentos

Nenhum.

### Descrição

Método; retorna o índice da posição do cursor intermitente. Se nenhum cursor intermitente for exibido, o método retorna -1. Os índices do intervalo de seleção são baseados em zero (onde a primeira posição é 0, a segunda posição é 1 e assim por diante).

### Exibidor

Flash 5 ou posterior.

## Selection.getEndIndex

### Sintaxe

`Selection.getEndIndex();`

### Argumentos

Nenhum.

### Descrição

Método; retorna o índice final do intervalo de seleção focalizado no momento. Se não houver índice, ou se não houver um intervalo de seleção selecionado, o método retorna -1. Os índices do intervalo de seleção são baseados em zero (onde a primeira posição é 0, a segunda posição é 1 e assim por diante).

### Exibidor

Flash 5 ou posterior.

## Selection.getFocus

### Sintaxe

`Selection.getFocus();`

### Argumentos

Nenhum.

### Descrição

Método; retorna o nome da variável do campo de texto editável focalizado no momento. Se nenhum campo de texto estiver focalizado no momento, o método retorna null.



#### Exibidor

Flash 5 ou posterior.

#### Exemplo

O código a seguir retorna o nome da variável.

```
_root.anyMovieClip.myTextField.
```

## Selection.setFocus

#### Sintaxe

```
Selection.setFocus(variável);
```

#### Argumentos

*variável* Uma sequência de caracteres que especifica o nome de uma variável associada ao campo de texto usando uma notação com pontos ou barra.

#### Descrição

Método; focaliza o campo de texto editável associado à *variável* específica.

#### Exibidor

Flash 5 ou posterior.

## Selection.setSelection

#### Sintaxe

```
Selection.setSelection(início, fim);
```

#### Argumentos

*início* O índice inicial do intervalo de seleção.

*fim* O índice final do intervalo de seleção.

#### Descrição

Método; define o intervalo de seleção do campo de texto focalizado no momento. O novo intervalo de seleção iniciará no índice especificado no argumento *início* e terminará no índice especificado no argumento *fim*. Os índices do intervalo de seleção são baseados em zero (onde a primeira posição é 0, a segunda posição é 1 e assim por diante). Esse método não tem efeito se não houver campo de texto focalizado no momento.

#### Exibidor

Flash 5 ou posterior.

## set

### Sintaxe

```
variável = expressão;  
set(variável, expressão);
```

### Argumentos

*variável* O nome do recipiente que retém o valor do argumento *expressão*.

*expressão* O valor (ou uma frase que pode ser avaliada em um valor) atribuído à variável.

### Descrição

Ação; atribui um valor a uma variável. Uma variável é um recipiente que contém informações. O recipiente é sempre o mesmo, mas o conteúdo pode mudar. Ao alterar o valor de uma variável quando o filme estiver sendo reproduzido, você poderá registrar e salvar informações sobre as atividades do usuário, gravar valores que mudam à medida que o filme é reproduzido ou avaliar se uma condição é true ou false.

As variáveis podem conter números ou sequência de caracteres. Cada instância de filme e clipe de filme possui seu próprio conjunto de variáveis, e cada variável possui seu próprio valor independentemente das variáveis em outros filmes ou clipes de filme.

O ActionScript é uma linguagem sem texto. Isso significa que as variáveis não precisam ser definidas explicitamente como contendo um número ou uma sequência de caracteres. O Flash interpreta o tipo de dados como um inteiro ou uma sequência de caracteres, de forma adequada.

Use o comando `set` juntamente com a ação `call` para passar ou retornar valores.

### Exibidor

Flash 4 ou posterior.

### Exemplo

Este exemplo define uma variável chamada `orig_x_pos` que armazena a posição do eixo *x* original do clipe de filme `ship` para redefinir o envio em sua localização inicial posteriormente no filme.

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

Isso é equivalente a escrever o seguinte:

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

### Consulte também

`var`, na página 388  
`call`, na página 236

## setProperty

### Sintaxe

`setProperty(destino, propriedade, expressão);`

### Argumentos

*destino* O caminho para o nome da instância do clipe de filme cuja propriedade está sendo definida.

*propriedade* A propriedade a ser definida.

*expressão* O valor ao qual a propriedade é definida.

### Descrição

Ação; altera a propriedade de um clipe de filme à medida que o filme é reproduzido.

### Exibidor

Flash 4 ou posterior.

### Exemplo

Esse comando define a propriedade `_alpha` de um clipe de filme chamado `star` como 30 por cento quando o botão é clicado.

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

### Consulte também

`getProperty`, na página 274

## Sound (objeto)

O objeto `Sound` permite que você defina e controle sons em uma instância de um clipe de filme em particular, ou para a Linha de Tempo global, se você não especificar um *destino* ao criar um novo objeto `Sound`. Você deve usar o construtor `new Sound` para criar uma instância do objeto `Sound` antes de chamar os métodos do objeto `Sound`.

O objeto `Sound` é suportado somente no Flash Player 5.



## Resumo de métodos do objeto Sound

Método	Descrição
attachSound	Anexa o som especificado no argumento.
getPan	Retorna o valor da chamada setPan anterior.
getTransform	Retorna o valor da chamada setTransform anterior.
getVolume	Retorna o valor da chamada setVolume anterior.
setPan	Define a distribuição esquerda/direita do som.
setTransform	Define a transformação de um som.
setVolume	Define o nível de volume de um som.
start	Inicia reproduzindo um som desde o início ou, opcionalmente, de um ponto de desvio definido no argumento.
stop	Pára o som especificado ou todos os sons em reprodução no momento.

## Construtor do objeto Sound

### Sintaxe

```
new Sound();  
new Sound(destino);
```

### Argumentos

*destino* A instância do clipe de filme à qual o objeto Sound se aplica. Esse argumento é opcional.

### Descrição

Método; cria um novo objeto Sound para um clipe de filme especificado. Se você não especificar um *destino*, o objeto Sound controla todos os sons na Linha de Tempo global.

### Exibidor

Flash 5 ou posterior.

### Exemplo

```
GlobalSound = new Sound();  
MovieSound = new Sound(mymovie);
```



## Sound.attachSound

### Sintaxe

```
mySound.attachSound("idName");
```

### Argumentos

*idName* O nome da nova instância do som. Esse nome é o mesmo inserido para o identificador na caixa de diálogo Propriedades de Vinculação de Símbolo. Esse argumento deve estar entre " " (aspas).

### Descrição

Método; anexa o som especificado no argumento *idName* ao objeto Sound especificado. O som deve estar na biblioteca do filme atual e ser especificado para exportação na caixa de diálogo Propriedades de Vinculação de Símbolo. Você deve chamar `Sound.start` para iniciar a reprodução do som.

### Exibidor

Flash 5 ou posterior.

### Consulte também

`Sound.start`, na página 367

## Sound.getPan

### Sintaxe

```
mySound.getPan();
```

### Argumentos

Nenhum.

### Descrição

Método; retorna o nível de pan definido na última chamada `setPan` como um inteiro de -100 a 100. A configuração de pan controla a distribuição esquerda-direita dos sons futuros e atuais em um filme.

Esse método é cumulativo com os métodos `setVolume` ou `setTransform`.

### Exibidor

Flash 5 ou posterior.

### Consulte também

`Sound.setPan`, na página 362

`Sound.setTransform`, na página 363

## Sound.getTransform

### Sintaxe

`mySound.getTransform();`

### Argumentos

Nenhum.

### Descrição

Método; retorna as informações de transformação do som do objeto Sound especificado na última chamada `setTransform`.

### Exibidor

Flash 5 ou posterior.

### Consulte também

`Sound.setTransform`, na página 363

## Sound.getVolume

### Sintaxe

`mySound.getVolume();`

### Argumentos

Nenhum.

### Descrição

Método; retorna o nível do volume de som como um inteiro de 0 a 100, no qual 0 é sem volume e 100 é o volume total. A configuração padrão é 100.

### Exibidor

Flash 5 ou posterior.

### Consulte também

`Sound.setVolume`, na página 366

## Sound.setPan

### Sintaxe

`mySound.setPan(pan);`

### Argumentos

*pan* Um inteiro que especifica a distribuição esquerda-direita de um som. O intervalo de valores válidos -100 a 100, no qual -100 usa somente o canal da esquerda, 100 usa somente o canal direito e 0 distribui o som uniformemente entre os dois canais.



### Descrição

Método; determina como o som é reproduzido nos canais esquerdo e direito (alto-falantes). Para os sons mono, *pan* afeta o alto-falante (esquerdo ou direito) pelo qual o som passa.

Esse método é cumulativo com os métodos `setVolume` e `setTransform` e chamar esse método exclui e atualiza as configurações anteriores de `setPan` e `setTransform`.

### Exibidor

Flash 5 ou posterior.

### Exemplo

O exemplo a seguir usa `setVolume` e `setPan` para controlar um objeto `Sound` com o destino especificado "u2."

```
onClipEvent(mouseDown) {  
    // cria um objeto som e  
    s = new Sound(this);  
    // anexa um som na biblioteca  
    s.attachSound("u2");  
    //define o volume em 50%  
    s.setVolume(50);  
    //desliga o som no canal direito  
    s.setPan(-100);  
    //inicia 30 segundos no som e o reproduz 5 vezes  
    s.start(30, 5);  
}
```

### Consulte também

`Sound.setTransform`, na página 363

`Sound.setVolume`, na página 366

## Sound.setTransform

### Sintaxe

```
mySound.setTransform(soundTransformObject);
```

### Argumentos

*soundTransformObject* Um objeto criado com o construtor de um objeto `Object` genérico.



### Descrição

Método; define as informações de transformação do som de um objeto Sound. Esse método é cumulativo com os métodos `setVolume` e `setPan` e chamar esse método exclui e atualiza as configurações de `setPan` ou `setVolume` anteriores. Essa chamada é para usuários experientes que desejam adicionar efeitos interessantes aos sons.

Os sons ocupam quantidade considerável de espaço em disco e memória. Como o som estéreo usa duas vezes mais dados do que os sons mono, geralmente é melhor usar sons mono de 22 KHz de 6 bits. Você pode usar o método `setTransform` para reproduzir sons mono como estéreo, sons estéreo como mono e para adicionar efeitos de som interessantes.

O argumento `sound transformObject` é um objeto que você cria usando o método construtor do Object genérico com parâmetros que especificam como o som é distribuído para os canais esquerdo e direito (alto-falantes).

Os parâmetros do `sound transformObject` são os seguintes:

ll Uma porcentagem que especifica a quantidade de som do canal esquerdo a ser reproduzida no alto-falante esquerdo (-100 a 100).

lr Uma porcentagem que especifica a quantidade de som do canal direito a ser reproduzida no alto-falante esquerdo (-100 a 100).

rr Uma porcentagem que especifica a quantidade de som do canal direito a ser reproduzida no alto-falante direito (-100 a 100).

rl Uma porcentagem que especifica a quantidade de som do canal esquerdo a ser reproduzida no alto-falante direito (-100 a 100).

O resultado da rede de parâmetros é representado pela seguinte fórmula:

$$\text{leftOutput} = \text{left input} * ll + \text{right input} * lr$$

$$\text{rightOutput} = \text{right input} * rr + \text{left input} * rl$$

Os valores para os sons do canal esquerdo e direito são determinados pelo tipo (estéreo ou mono) do som do filme.

Os sons estéreo dividem a entrada de som uniformemente entre os alto-falantes esquerdo e direito e, por padrão, têm as seguintes configurações de transformação:

$$ll = 100$$

$$lr = 0$$

$$rr = 100$$

$$rl = 0$$

Os sons mono reproduzem toda a entrada de som no alto-falante esquerdo e, por padrão, têm as seguintes configurações de transformação:

$$ll = 100$$

$$lr = 100$$

$$rr = 0$$

$$rl = 0$$



### Exibidor

Flash 5 ou posterior.

### Exemplo

O exemplo a seguir cria um sound transformobject que é reproduzido nos canais esquerdo e direito do canal esquerdo.

```
mySoundTransformObject = new Object  
mySoundTransformObject.l1 = 100  
mySoundTransformObject.lr = 100  
mySoundTransformObject.rr = 0  
mySoundTransformObject.rl = 0
```

O código acima cria um sound transform object. Para poder aplicá-lo a um objeto sound, você precisa passar o objeto para o objeto Sound usando setTransform da seguinte forma:

```
mySound.setTransform(mySoundTransformObject);
```

A seguir temos exemplos de configurações que podem ser definidas usando setTransform, mas não podem ser definidas usando setVolume ou setPan, mesmo se combinados.

Esse código reproduz os canais esquerdo e direito através do canal esquerdo:

```
mySound.setTransform(soundTransformObjectLeft);
```

No código acima, *soundTransformObjectLeft* tem os seguintes parâmetros:

```
l1 = 100  
lr = 100  
rr = 0  
rl = 0
```

### Exemplo

Este código reproduz um som estéreo como mono:

```
setTransform(soundTransformObjectMono);
```

No código acima, *soundTransformObjectMono* tem os seguintes parâmetros:

```
l1 = 50  
lr = 50  
rr = 50  
rl = 50
```



### Exemplo

Este código reproduz o canal esquerdo na metade de sua capacidade e adiciona o restante do canal esquerdo ao canal direito:

```
setTransform(soundTransformObjectHalf);
```

No código acima, *soundTransformObjectHalf* tem os seguintes parâmetros:

```
ll = 50  
lr = 0  
rr = 100  
rl = 50
```

### Consulte também

Construtor do objeto Object, na página 337

## Sound.setVolume

### Sintaxe

```
mySound.setVolume(volume);
```

### Argumentos

*volume* Um número de 0 a 100 que representa um nível de volume. 100 é o volume total e 0 é nenhum volume. A configuração padrão é 100.

### Descrição

Método; define o volume do objeto sound.

Esse método é cumulativo com os métodos *setPan* e *setTransform*.

### Exibidor

Flash 5 ou posterior.

### Exemplo

O exemplo a seguir define o volume em 50% e transfere o som, com o passar do tempo, do alto-falante esquerdo para o direito.

```
onClipEvent (load) {  
    i = -100;  
    s = new Sound(this);  
    s.setVolume(50);  
}  
onClipEvent (enterFrame) {  
    S.setPan(i++);  
}
```

### Consulte também

*Sound.setPan*, na página 362

*Sound.setTransform*, na página 363

## Sound.start

### Sintaxe

```
mySound.start();
```

```
mySound.start([secondOffset, loop]);
```

*secondOffset* Um argumento opcional que permite que você inicie a reprodução do som em um ponto específico. Por exemplo, se você tem um som de 30 segundos e o som inicia a reprodução no meio, especifique 15 para o argumento *secondOffset*. O som não é atrasado 15 segundos; em vez disso, ele inicia a sua reprodução na marca de 15 segundos.

*loop* Um argumento opcional que permite que você especifique o número de vezes que o som deve ser repetido.

### Descrição

Método; inicia a reprodução do último som anexado, do início se nenhum argumento for especificado, ou em um ponto especificado pelo argumento *secondOffset*.

### Exibidor

Flash 5 ou posterior.

### Consulte também

Sound.setPan, na página 362

Sound.stop, na página 367

## Sound.stop

### Sintaxe

```
mySound.stop();
```

```
mySound.stop(["idName"]);
```

### Argumentos

*idName* Um argumento opcional que especifica que um som em específico pára de ser reproduzido. O argumento *idName* deve estar entre aspas (" ").

### Descrição

Método; se nenhum argumento for especificado, pára todos os sons em exibição no momento; caso contrário, apenas o som especificado no argumento *idName*.

### Exibidor

Flash 5 ou posterior.

### Consulte também

Sound.start, na página 367

## \_soundbuftime

### Sintaxe

`_soundbuftime = inteiro;`

### Argumentos

*inteiro* O número de segundos decorridos antes que o filme comece a ser reproduzido.

### Descrição

Propriedade (global); estabelece o número de segundos de som de fluxo para o pré-buffer. O valor padrão é 5 segundos.

### Exibidor

Flash 4 ou posterior.

## startDrag

### Sintaxe

`startDrag(destino);`

`startDrag(destino,[lock]);`

`startDrag(destino [,bloqueio [,esquerdo , superior , direito, inferior]]);`

### Argumentos

*destino* O caminho de destino do clipe de filme a ser arrastado.

*bloqueio* Um valor booleano que especifica se o clipe de filme a ser arrastado está bloqueado o no centro da posição do mouse (*true*) ou no ponto onde o usuário clicou pela primeira vez no clipe de filme (*false*). Esse argumento é opcional.

*esquerdo, superior, direito, inferior* Valores relativos às coordenadas do pai do clipe de filme que especificam um retângulo de restrição para o clipe de filme. Esses argumentos são opcionais.

### Descrição

Ação; torna o clipe de filme de *destino* arrastável enquanto o filme está sendo exibido. Somente um clipe de filme pode ser arrastado de cada vez. Quando uma operação *startDrag* é executada, o clipe de filme permanece arrastável até que seja explicitamente encerrado por uma ação *stopDrag* ou até que uma ação *startDrag* para outro clipe de filme seja chamada.

### Exemplo

Para criar um clipe de filme que os usuários possam posicionar em qualquer parte e anexar as ações `startDrag` e `stopDrag` a um botão dentro do clipe de filme, conforme mostrado a seguir:

```
on(press) {  
    startDrag("",true);  
}  
on(release) {  
    stopDrag();  
}
```

### Consulte também

`stopDrag`, na página 370

`_droptarget`, na página 263

## stop

### Sintaxe

`stop;`

### Argumentos

Nenhum.

### Descrição

Ação; encerra o filme em exibição. A utilidade mais comum dessa ação é controlar clipes de filme com botões.

### Exibidor

Flash 3 ou posterior.

## stopAllSounds

### Sintaxe

`stopAllSounds();`

### Argumentos

Nenhum.

### Descrição

Ação; encerra a reprodução de todos os sons de um filme sem interromper a exibição do filme. O som voltará a ser reproduzido quando a exibição do filme avançar os quadros em que se encontra.

### Exibidor

Flash 3 ou posterior.

### Exemplo

O código a seguir pode ser aplicado a um botão que, quando clicado, encerra todos os sons do filme.

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

### Consulte também

Sound (objeto), na página 359

## stopDrag

### Sintaxe

```
stopDrag();
```

### Argumentos

Nenhum.

### Descrição

Ação; encerra a operação de arraste em andamento.

### Exibidor

Flash 4 ou posterior.

### Exemplo

Este comando encerra a operação de arraste na instância `mc` quando o usuário libera o botão do mouse.

```
on(press) {  
    startDrag("mc");  
}  
on(release) {  
    stopdrag();  
}
```

### Consulte também

`startDrag`, na página 368  
`_droptarget`, na página 263

## String (função)

### Sintaxe

```
String(expressão);
```

### Argumentos

*expression* O número, booleano, variável ou objeto a ser convertido em uma seqüência de caracteres.

### Descrição

Função; retorna uma sequência de caracteres que representa o parâmetro especificado da seguinte forma:

Se *x* for booleano, a sequência de caracteres retornada será `true` ou `false`.

Se *x* for um número, a sequência de caracteres retornada será uma representação decimal do número.

Se *x* for uma sequência de caracteres, a sequência retornada será *x*.

Se *x* for um objeto, o valor retornado será uma sequência de caracteres que representa o objeto gerado pela chamada da propriedade da sequência de caracteres ao objeto ou pela chamada de `object.toString`, se tal propriedade não existir.

Se *x* for um clipe de filme, o valor de retorno será o caminho de destino do clipe de filme em notação de barra (/).

Se *x* for indefinido, o valor de retorno será uma sequência de caracteres vazia.

### Exibidor

Flash 3 ou posterior.

### Consulte também

`Object.toString`, na página 338

`Number.toString`, na página 336

`String (objeto)`, na página 372

`" "` (delimitador de sequência de caracteres), na página 371

## " " (delimitador de sequência de caracteres)

### Sintaxe

`"texto"`

### Argumentos

*texto* Qualquer texto.

### Descrição

Delimitador de sequência de caracteres; quando usado antes de depois de uma sequência de caracteres, as aspas indicam que a sequência de caracteres é um valor literal — não uma variável, um valor numérico ou outro elemento ActionScript.

#### Exibidor

Flash 4 ou posterior.

#### Exemplo

Este comando utiliza aspas para indicar que a sequência de caracteres “Prince Edward Island” é um valor literal, não o valor de uma variável:

```
province = "Prince Edward Island"
```

#### Consulte também

String (objeto), na página 372

String (função), na página 370

## String (objeto)

O objeto String é um wrapper para o tipo de dados primitivo de sequência de caracteres, que permite usar os métodos e as propriedades do objeto String para manipular tipos de valores primitivos de sequências de caracteres. Você pode converter o valor de qualquer objeto em uma sequência de caracteres usando a função `String()`.

Todos os métodos do objeto String, exceto `concat`, `fromCharCode`, `slice` e `substr`, são genéricos. Isso significa que os próprios métodos podem chamar `this.toString` antes de permitir suas operações, e podem ser usados com outros objetos que não sejam objetos String.

Você pode chamar qualquer um dos métodos do objeto String usando o método construtor `new String` ou usando o valor literal de uma sequência de caracteres. Se você especificar o valor literal de uma sequência de caracteres, o interpretador ActionScript automaticamente o converterá em um objeto String temporário, chamará o método e depois descartará o objeto String temporário. Você também pode utilizar a propriedade `String.length` com o valor literal de uma sequência de caracteres.

É importante não confundir o valor literal de uma sequência de caracteres com uma instância do objeto String. No exemplo a seguir, a primeira linha de código cria o valor literal da sequência de caracteres `s1` e a segunda linha de código cria uma instância do objeto String `s2`.

```
s1 = "foo"  
s2 = new String("foo")
```

Recomenda-se que você use literais de sequência de caracteres se não precisar especificamente usar um objeto String, pois esses objetos por ter comportamento intuitivo.





## Resumo de métodos do objeto String

Método	Descrição
charAt	Retorna um número que corresponde à colocação do caractere na sequência de caracteres.
charCodeAt	Retorna o valor do caractere de um índice determinado como um inteiro de 16 bits entre 0 e 65535.
concat	Combina o texto de duas sequências de caracteres e retorna uma nova sequência de caracteres
fromCharCode	Retorna uma sequência de caracteres feita de caracteres especificados em argumentos.
indexOf	Pesquisa a sequência de caracteres e retorna o índice do valor especificado nos argumentos. Se o valor ocorrer mais de uma vez, o índice da primeira ocorrência é retornado. Se o valor não for encontrado, -1 é retornado.
lastIndexOf	Retorna a última ocorrência da subsequência de caracteres na sequência de caracteres que aparece antes da posição de início especificado no argumento ou -1 se não encontrado.
slice	Extrai uma seção de uma sequência de caracteres e retorna uma nova sequência de caracteres.
split	Divide um objeto string em um vetor de sequência de caracteres separando a sequência de caracteres em subsequências.
substr	Retorna um número especificado de caracteres em uma sequência de caracteres, começando no local especificado no argumento.
substring	Retorna os caracteres entre dois índices, especificado nos argumentos, na sequência de caracteres.
toLowerCase	Converte a sequência de caracteres em minúsculas e retorna o resultado.
toUpperCase	Converte a sequência de caracteres em maiúsculas e retorna o resultado.

## Resumo de propriedades do objeto String

Propriedade	Descrição
length	Retorna o tamanho da sequência de caracteres



## Construtor do objeto String

### Sintaxe

`new String(valor);`

### Argumentos

*valor* O valor inicial do objeto new String.

### Descrição

Construtor; cria um objeto new String.

### Exibidor

Flash 5 ou posterior.

### Consulte também

String (função), na página 370

" " (delimitador de sequência de caracteres), na página 371

## String.charAt

### Sintaxe

`myString.charAt(índice);`

### Argumentos

*índice* O número do caractere na sequência de caracteres a ser retornado.

### Descrição

Método; retorna o caractere especificado pelo argumento *índice*. O índice do primeiro caractere em uma sequência de caracteres é 0. Se *índice* não for um número de 0 a `string.length - 1` é retornada uma sequência de caracteres vazia.

### Exibidor

Flash 5 ou posterior.

## String.charCodeAtAt

### Sintaxe

`myString.charCodeAtAt(índice);`

### Argumentos

*índice* O número do caractere do qual o valor é recuperado.

#### Descrição

Método; retorna o valor do caractere especificado por *índice*. O valor retornado é um inteiro de 16 bits de 0 a 65535.

Este método é semelhante a `string.charAt` exceto pelo fato de que o valor retornado é do caractere em um local específico, e não da subsequência de caracteres que o contém.

#### Exibidor

Flash 5 ou posterior.

## String.concat

#### Sintaxe

```
myString.concat(valor1,...valorN);
```

#### Argumentos

*valor1,...valorN* Zero ou mais valores a serem concatenados.

#### Descrição

Método; combina os valores especificados e retorna uma nova sequência de caracteres. Se necessário, cada argumento *valor* é convertido em uma sequência de caracteres e anexado, em ordem, ao final da sequência de caracteres.

#### Exibidor

Flash 5 ou posterior.

## String.fromCharCode

#### Sintaxe

```
myString.fromCharCode(c1,c2,...cN);
```

#### Argumentos

*c1,c2,...cN* Os caracteres a serem transformados em sequência de caracteres.

#### Descrição

Método; retorna uma sequência de caracteres composta de caracteres especificados em argumentos.

#### Exibidor

Flash 5 ou posterior.



## String.indexOf

### Sintaxe

`myString.indexOf(valor);`

`myString.índice de (valor, início);`

### Argumentos

*valor* Um inteiro ou sequência de caracteres a ser pesquisado em *myString*.

*início* Um inteiro que especifica o ponto de início da subsequência de caracteres. Esse argumento é opcional.

### Descrição

Método; pesquisa a sequência de caracteres e retorna a posição da primeira ocorrência do *valor* especificado. Se o valor não for encontrado, o método retorna -1.

### Exibidor

Flash 5 ou posterior.

## String.lastIndexOf

### Sintaxe

`myString.lastIndexOf(subseqüência de caracteres);`

`myString.lastIndexOf(subseqüência de caracteres, início);`

### Argumentos

*subseqüência de caracteres* Um inteiro ou sequência de caracteres que especifica a sequência a ser pesquisada.

*início* Um inteiro que especifica o ponto de início dentro da subsequência de caracteres. Esse argumento é opcional.

### Descrição

Método; pesquisa a sequência de caracteres e retorna o índice da última ocorrência da subsequência de caracteres encontrada na sequência de caracteres de chamada. Se a *subseqüência de caracteres* não for encontrada, o método retorna -1.

### Exibidor

Flash 5 ou posterior.

## String.length

### Sintaxe

`string.length`

### Argumentos

Nenhum.

### Descrição

Propriedade; retorna o número de caracteres no objeto String especificado. O índice do último caractere de qualquer sequência de caracteres *x*, é *x.length-1*.

### Exibidor

Flash 5 ou posterior.

## String.slice

### Sintaxe

`myString.slice(início, fim);`

### Argumentos

*início* Um número que especifica o índice do ponto inicial do pedaço. Se *início* for um número negativo, o ponto inicial é determinado a partir do final da sequência de caracteres, onde -1 é o último caractere.

*término* Um número que especifica o ponto final do pedaço. Se *término* não for especificado, o pedaço inclui todos os caracteres do início ao fim da sequência de caracteres. Se *fim* for um número negativo, o ponto final é determinado a partir do final da sequência de caracteres, onde -1 é o último caractere.

### Descrição

Método; extrai uma parte, ou subsequência de caracteres do objeto String especificado; depois, retorna-o como uma nova sequência, sem modificar o objeto String original. A sequência de caracteres inclui o caractere de *início* e todos os caracteres até (mas não incluindo) o caractere de *fim*.

### Exibidor

Flash 5 ou posterior.

## String.split

### Sintaxe

`myString.split(delimitador);`

### Argumentos

*delimitador* O caractere usado para delimitar a sequência de caracteres.

### Descrição

Método; divide um objeto String quebrando a sequência de caracteres no qual o argumento *delimitador* especificado ocorrer e retorna as subsequências de caracteres em um vetor. Se nenhum delimitador for especificado, o vetor retornado contém somente um elemento — a própria sequência de caracteres. Se o delimitador for uma sequência de caracteres vazia, cada caractere no objeto String se torna um elemento do vetor.

### Exibidor

Flash 5 ou posterior.

## String.substr

### Sintaxe

`myString.substr(início, tamanho);`

### Argumentos

*início* Um inteiro que indica a posição do primeiro caractere na subsequência de caracteres a ser criada. Se *início* for um número negativo, a posição inicial é determinada a partir do final da sequência de caracteres, onde -1 é o último caractere.

*tamanho* O número de caracteres na subsequência de caracteres que está sendo criada. Se *tamanho* não for especificado, a subsequência de caracteres inclui todos os caracteres do início ao fim da sequência de caracteres.

### Descrição

Método; retorna os caracteres em uma sequência de caracteres do índice especificado no argumento *início* até o número de caracteres especificado no argumento *tamanho*.

### Exibidor

Flash 5 ou posterior.

## String.substring

### Sintaxe

`myString.substring(de, para);`

### Argumentos

*de* Um inteiro que indica a posição do primeiro caractere na subsequência de caracteres que está sendo criada. Os valores válidos para *de* vão de 0 a `string.length - 1`.

*para* Um inteiro que é 1+ o índice do último caractere na subsequência de caracteres que está sendo criada. Os valores válidos para *para* vão de 1 até `string.length`. Se o argumento *para* não for especificado, o final da subsequência de caracteres é o final da sequência de caracteres. Se *de* for igual a *para*, o método retorna uma sequência de caracteres vazia. Se *de* é maior que *para*, os argumentos são trocados automaticamente antes que a função seja executada.

### Descrição

Método; retorna uma sequência de caracteres que consiste de caracteres entre os pontos especificados nos argumentos *de* e *para*.

### Exibidor

Flash 5 ou posterior.

## String.toLowerCase

### Sintaxe

`myString.toLowerCase();`

### Argumentos

Nenhum.

### Descrição

Método; retorna uma cópia do objeto String, com todos os caracteres em maiúsculas convertidos em minúsculas.

### Exibidor

Flash 5 ou posterior.



## String.toUpperCase

### Sintaxe

`myString.toUpperCase();`

### Argumentos

Nenhum.

### Descrição

Método; retorna uma cópia do objeto String, com todos os caracteres em maiúsculas convertidos em minúsculas.

### Exibidor

Flash 5 ou posterior.

## substring

### Sintaxe

`substring(seqüência de caracteres, índice, contagem);`

### Argumentos

*seqüência de caracteres* A seqüência de caracteres da qual extrair a nova seqüência de caracteres.

*índice* O número do primeiro caractere a ser extraído.

*contagem* O número de caracteres a ser incluído na seqüência de caracteres extraída, sem incluir o caractere índice.

### Descrição

Função String; extrai parte de uma seqüência de caracteres.

### Exibidor

Flash 4 ou posterior. Essa função é obsoleta no Flash 5.

### Consulte também

String.substring, na página 379



## **\_target**

### **Sintaxe**

*nome da instância*.\_target

### **Argumentos**

*nome da instância* O nome da instância do clipe de filme.

### **Descrição**

Propriedade (somente leitura); retorna o caminho de destino da instância do clipe de filme especificada no argumento *nome da instância*.

### **Exibidor**

Flash 4 ou posterior.

## **targetPath**

### **Sintaxe**

`targetPath(movieClipObject);`

### **Argumentos**

*movieClipObject* Referência (por exemplo, `_root` ou `_parent`) ao clipe de filme cujo caminho de destino está sendo recuperado.

### **Descrição**

Função; retorna uma sequência de caracteres que contém o caminho de destino *movieClipObject*. O caminho de destino é retornado em notação com pontos. Para recuperar o caminho de destino em notação de barras, use a propriedade `_target`.

### **Exibidor**

Flash 5 ou posterior.

### **Exemplo**

Os exemplos a seguir são equivalentes. O primeiro exemplo usa a notação com pontos e o segundo exemplo usa a notação com barras.

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

É equivalente a:

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

### **Consulte também**

`eval`, na página 266

## tellTarget

### Sintaxe

```
tellTarget(destino) {  
    comando;  
}
```

### Argumentos

*destino* Uma sequência de caracteres de destino que especifica a Linha de Tempo a ser controlada.

*comando* Instruções aplicadas a Linha de Tempo de destino.

### Descrição

Ação; aplica as instruções especificadas no argumento *comandos* à Linha de Tempo especificada no argumento *destino*. A ação `tellTarget` é útil para controles de navegação. Atribua `tellTarget` a botões que encerram ou começam clipes de filme em qualquer local do Palco. Você também pode fazer clipes de filme irem para um quadro em particular no clipe. Por exemplo, atribua `tellTarget` a botões que encerram ou começam clipes de filme no Palco ou solicitem que os clipes de filme pulem para um quadro em particular.

A ação `tellTarget` é muito semelhante à ação `with`, exceto pelo fato de que `with` considera um clipe de filme ou outro objeto como um *destino* e `tellTarget` requer um caminho de destino para um clipe de filme e não pode controlar objetos.

### Exibidor

Flash 3 ou posterior. Essa ação está obsoleta no Flash 5; o uso da ação `with` é recomendado.

### Exemplo

Esse comando `tellTarget` controla na instância do clipe de filme `ball` na Linha de Tempo principal. O quadro 1 do clipe de filme está vazio e tem uma ação `stop` de forma que não é visível no Palco. Quando o botão com a ação a seguir é clicado, `tellTarget` informa à reprodução no clipe do filme `ball` para ir para o quadro 2 e reproduzir a animação que começa nele.

```
on(release) {  
    tellTarget("ball") {  
        gotoAndPlay(15);  
    }  
}
```

### Consulte também

`with`, na página 391

## this

### Sintaxe

this

### Argumentos

Nenhum.

### Descrição

Palavra-chave; faz referência a uma instância de objeto ou de clipe de filme.

A palavra-chave `this` tem o mesmo objetivo e função no ActionScript como no JavaScript, com um pouco mais de funcionalidade. No ActionScript, quando um script é executado, `this` faz referência à instância do clipe de filme que contém o script. Quando usada com uma chamada de método, `this` contém uma referência ao objeto que contém o método executado.

### Exibidor

Flash 5 ou posterior.

### Exemplo

No exemplo a seguir, a palavra-chave `this` faz referência ao objeto `Circle`.

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

No comando a seguir atribuído a um quadro, a palavra-chave `this` faz referência ao clipe de filme atual.

```
//define a propriedade alpha do clipe de filme atual como 20.  
star._alpha = 20;
```

No comando a seguir em um manipulador `onClipEvent`, a palavra-chave `this` faz referência ao clipe de filme atual.

```
//quando o clipe de filme é carregado, uma operação startDrag é  
iniciada para o clipe de filme atual.
```

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

### Consulte também

`new`, na página 328

## toggleHighQuality

### Sintaxe

`toggleHighQuality();`

### Argumentos

Nenhum.

### Descrição

Ação; ativa ou desativa o modo sem serrilhado no Flash Player. O modo sem serrilhado suaviza as bordas dos objetos e reduz a reprodução do filme. A ação `toggleHighQuality` afeta todos os filmes no Flash Player.

### Exibidor

Flash 2 ou posterior.

### Exemplo

O código a seguir pode ser aplicado a um botão que, quando clicado, ativa ou desativa o modo sem serrilhado.

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

### Consulte também

`_quality`, na página 350

`_highquality`, na página 279

## \_totalframes

### Sintaxe

*nome do quadro*.`_totalframes`

### Argumentos

*nome do quadro* O nome do clipe do filme a ser avaliado.

### Descrição

Propriedade (somente leitura); avalia um clipe de filme especificado no argumento *nome da instância* e retorna o número total de quadros no filme.

### Exibidor

Flash 4 ou posterior.

## trace

### Sintaxe

`trace(expressão);`

### Argumentos

*expressão* Um comando a ser avaliado. Quando você testa o filme, os resultados do argumento *expressão* são exibidos na janela Saída.

### Descrição

Ação; avalia a *expressão* e exibe os resultados na janela Saída no modo de teste de filme.

Use `trace` para registrar as observações de programação ou para exibir mensagens na janela Saída enquanto testa um filme. Use o parâmetro *expressão* para verificar se uma condição existe, ou para exibir valores na janela Saída. A ação `trace` é semelhante função `alert` no JavaScript.

### Exibidor

Flash 4 ou posterior.

### Exemplo

Este exemplo é de um jogo no qual uma instância de clipe de filme arrastável chamada `rabbi` deve ser liberada em um destino específico. Um comando adicional avalia a propriedade `_droptarget` e executa diferentes ações dependendo do local onde `rabbi` é liberado. A ação `trace` é usada ao final do script para avaliar a localização do clipe de filme `rabbi`, e exibe os resultados na janela Saída. Se `rabbi` não se comportar como esperado (por exemplo, se ele se encaixar no destino errado), os valores enviados para a janela Saída pela ação `trace` ajudarão a determinar o problema no 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

### Sintaxe

`typeof(expressão);`

### Argumentos

*expressão* Uma seqüência de caracteres, clipe de filme, objeto ou função.

### Descrição

Operador; um operador unário colocado antes de um único argumento. Faz com que o Flash avalie a *expressão*; o resultado é uma seqüência de caracteres que especifica se a expressão é uma seqüência de caracteres, clipe de filme, objeto ou função.

### Exibidor

Flash 5 ou posterior.

## unescape

### Sintaxe

`unescape(x);`

### Argumentos

*x* Uma seqüência de caracteres com seqüências hexadecimais de escape.

### Descrição

Função de alto nível; avalia o argumento *x* como uma seqüência de caracteres, decodifica a seqüência de caracteres de um formato de codificação URL (convertendo todas as seqüências hexadecimais em caracteres ASCII) e retorna uma seqüência de caracteres.

### Exibidor

Flash 5 ou posterior.

### Exemplo

O exemplo a seguir ilustra o processo de conversão escape em unescape.

```
escape("Hello{[World]}");
```

O resultado de escape é o seguinte:

```
("Hello%7B%5BWorld%5D%7D");
```

Use unescape para retornar ao formato original:

```
unescape("Hello%7B%5BWorld%5D%7D")
```

O resultado é o seguinte:

```
Hello{[World]}
```

## unloadMovie

### Sintaxe

```
unloadMovie(local);
```

### Argumentos

*local* O nível de intensidade do clipe de filme do qual descarregar o filme.

### Descrição

Ação; remove um filme do Flash Player que foi carregado previamente usando a ação `loadMovie`.

### Exibidor

Flash 3 ou posterior.

### Exemplo

O exemplo a seguir descarrega o filme principal, deixando o Palco em branco.

```
unloadMovie(_root);
```

O exemplo a seguir descarrega o filme no nível 15, quando o usuário clica no mouse.

```
on(press) {  
    unloadMovie(_level15);  
}
```

### Consulte também

`loadMovie`, na página 294

## updateAfterEvent

### Sintaxe

```
updateAfterEvent(movie clip event);
```

### Argumentos

*movie clip event* Você pode especificar um dos seguintes valores como um evento de clipe de filme:

- `mouseMove` A ação é iniciada toda vez que o mouse é movido. Use as propriedades `_xmouse` e `_ymouse` para determinar a posição do mouse atual.
- `mouseDown` A ação é iniciada se o botão esquerdo do mouse for pressionado.
- `mouseUp` A ação é iniciada se o botão esquerdo do mouse for liberado.
- `keyDown` A ação é iniciada quando uma tecla é pressionada. Use o método `Key.getCode` para recuperar informações sobre a última tecla pressionada.
- `keyUp` A ação é iniciada quando uma tecla é liberada. Use o método `key.getCode` para recuperar as informações sobre a última tecla pressionada.

### Descrição

Ação; atualiza a exibição (independente dos quadros por segundo definidos para o filme) depois que o evento do clipe especificado nos argumentos tiver sido concluído. Essa ação não é listada no painel de Ações do Flash. Usar `updateAfterEvent` com ações de arraste que especificam as propriedades `_x` e `_y` durante o movimento do mouse permite que os objetos sejam arrastados suavemente sem o efeito da tela piscando.

### Exibidor

Flash 5 ou posterior.

### Consulte também

`onClipEvent`, na página 339

## \_url

### Sintaxe

*nome da instância*.`_url`

### Argumentos

*nome da instância* O clipe de filme de destino.

### Descrição

Propriedade (somente leitura); recupera a URL do arquivo SWF do qual o clipe de filme foi descarregado.

### Exibidor

Flash 4 ou posterior.

## var

### Sintaxe

```
var variableName1 [= valor1] [...,variableNameN [=valorN]];
```

### Argumentos

*variableName* O nome da variável a ser declarada.

*valor* O valor que está sendo atribuído à variável.

### Descrição

Ação; usado para declarar variáveis locais. Se você declarar variáveis locais em uma função, as variáveis são definidas para a função e expiram no final da chamada de função. Se as variáveis não são declaradas em um bloco, mas a lista de ações foi executada com uma ação `call`, as variáveis são locais e expiram no final da lista atual. Se as variáveis não forem declaradas em um bloco e a lista de ações atuais não foi executada com a ação `call`, as variáveis não são locais.

### Exibidor

Flash 5 ou posterior.



## \_visible

### Sintaxe

*nome da instância*.\_visible  
*nome da instância*.\_visible = *booleano*;

### Argumentos

*Booleano* Insere um valor *true* ou *false* para especificar se o clipe de filme é visível.

### Descrição

Propriedade; determina se o filme especificado pelo argumento *nome da instância* é visível. Os clipes de filme que não são visíveis (propriedade definida como *false*) são desativados. Por exemplo, um botão em um clipe de filme com a propriedade *\_visible* definida como *false* não pode ser clicado.

### Exibidor

Flash 4 ou posterior.

## void

### Sintaxe

void (*expressão*);

### Argumentos

*expressão* Uma expressão de qualquer valor.

### Descrição

Operador; um operador unário que descarta o valor da *expressão* e retorna um valor indefinido. O operador *void* sempre é usado para avaliar a URL para testar resultados sem exibir a expressão avaliada na janela do navegador. O operador *void* também é usado em comparações, usando o operador *==* para testar os valores indefinidos.

### Exibidor

Flash 5 ou posterior.

## while

### Sintaxe

```
while(condição) {  
  comando(s);  
}
```

### Argumentos

*condição* O comando que é reavaliado toda vez que a ação *while* é executada. Se o comando *for true*, a expressão *no(s) comando(s)* é executada.

*comando(s)* A expressão a ser executada se a condição *for true*.

### Descrição

Ação; executa um comando ou série de comandos repetidamente um loop enquanto o argumento condição for true. No final de cada ação while, o Flash reinicia o loop testando a condição novamente. Se a condição for false ou igual a 0, o Flash pula para o primeiro comando depois da ação while.

O loop é normalmente usado para executar um ação enquanto uma variável de contador for menor do que um valor especificado. No fim de cada loop, o contador é incrementado, até que o valor de limite seja alcançado, a condição não seja mais true, e o loop termine.

### Exibidor

Flash 4 ou posterior.

### Exemplo

Este exemplo duplica cinco clipes de filme no Palco, todos com uma posição x e y geradas aleatoriamente, propriedade xscale e yscale e \_alpha para conseguirem um efeito difuso. A variável foo é inicializada com o valor 0. O argumento condição é definido de forma que o loop while será executado cinco vezes, ou enquanto o valor da variável foo for menor do que 5. Dentro do loop while, um clipe de filme é duplicado e setProperty é usado para ajustar as várias propriedades do clipe de filme duplicado. O último comando do loop incrementa foo de forma que quando o valor alcança 5, o argumento condição avalia para false, e o loop não será executado.

```
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;  
    }  
}
```

### Consulte também

do... while, na página 262  
continue, na página 241

## **\_width**

### **Sintaxe**

*nome da instância*.\_width  
*nome da instância*.\_width = *valor*;

### **Argumentos**

*valor* A largura do filme em pixels.

*nome da instância* Um nome de instância de um clipe do filme no qual a propriedade `_width` deve ser definida ou recuperada.

### **Descrição**

Propriedade; define a largura do filme. Nas versões anteriores do Flash, `_height` e `_width` eram propriedades somente leitura; no Flash 5 elas podem ser definidas e recuperadas.

### **Exibidor**

Flash 4 como uma propriedade somente leitura. No Flash 5 ou posterior, essa propriedade pode ser definida e recuperada.

### **Exemplo**

O exemplo de código a seguir define a altura e a largura das propriedades de um clipe de filme quando o usuário clica no mouse.

```
onClipEvent(mouseDown) {  
    _width=200;  
    _height=200;  
}
```

### **Consulte também**

`_height`, na página 278

## **with**

x209E6 | IDS\_ACTIONHELP\_WITH, comando with

### **Sintaxe**

```
with (objeto) {  
    comando(s);  
}
```

### **Argumentos**

*objeto* Uma instância de um objeto ActionScript ou do clipe de filme.

*comando(s)* Uma ação ou grupo de ações entre aspas.



### Descrição

Ação; temporariamente altera o escopo (ou caminho de destino) usado para avaliar expressões e ações no(s) *comando(s)*. Depois que a ação *with* for executada, a cadeia do escopo é restaurada ao seu estado original.

O *object* se torna o contexto no qual as propriedades, variáveis e funções são lidas. Por exemplo, se *objeto* for *myArray*, e duas das propriedades especificadas forem *length* e *concat*, essas propriedades são lidas automaticamente como *myArray.length* e *myArray.concat*. Em outro exemplo, se *objeto* for *state.california*, as ações ou comandos dentro da ação *with* poderiam ser chamadas de dentro da instância *california*.

Para localizar o valor de um identificador no(s) *comando(s)*, o ActionScript inicia no começo da cadeia do escopo especificado pelo *objeto* e pesquisa o identificador no nível da cadeia do escopo, em uma ordem específica.

A cadeia do escopo usada pela ação *with* para resolver identificadores começa com o primeiro item na lista a seguir e continua até o último, como mostrado a seguir:

- *objeto* referenciado pela ação *with* mais interna
- *objeto* referenciado pela ação *with* mais externa
- Objeto Activation (Um objeto temporário que é criado automaticamente quando uma função chamada mantém as variáveis locais chamadas na função.)
- Clipe de filme que contém o script sendo executado no momento
- Objeto Global (objetos predefinidos como *Math*, *String*)

No Flash 5 a ação *with* substitui a ação *tellTarget* obsoleta. Você é encorajado a usar *with* em vez de *tellTarget*, pois é uma extensão do ActionScript padrão do padrão ECMA 262. A diferença principal entre a ação *with* e *tellTarget* é que *with* considera um clipe de filme ou outro objeto como seu argumento, enquanto *tellTarget* considera uma seqüência de caracteres de caminho de destino que identifica um clipe de filme, e não pode ser usado em objetos de destino.

Para definir uma variável dentro de uma ação *with*, a variável deve ter sido declarada fora da ação *with*, ou você deve inserir o caminho completo para a Linha de Tempo na qual deseja que a variável viva. Se você definir a ação *with* sem tê-la declarado, a ação *with* procurará o valor de acordo com a cadeia do escopo. Se a variável não existir ainda, o novo valor será definido na Linha de Tempo da qual a ação *with* foi chamada.

### Exemplo

O exemplo a seguir define as propriedades  $x$  e  $y$  da instância `someOtherMovieClip`, e instrui `someOtherMovieClip` a ir para o quadro 3 e encerrar:

```
with (someOtherMovieClip) {  
    _x = 50;  
    _y = 100;  
    gotoAndStop(3);  
}
```

O código a seguir mostra como escrever o código anterior sem usar uma ação `with`.

```
someOtherMovieClip._x = 50;  
someOtherMovieClip._y = 100;  
someOtherMovieClip.gotoAndStop(3);
```

Esse código também pode ser escrito usando a ação `tellTarget`.

```
tellTarget ("someOtherMovieClip") {  
    _x = 50;  
    _y = 100;  
    gotoAndStop(3);  
}
```

A ação `with` é útil para fornecer acesso a vários itens simultaneamente em uma cadeia de escopo. No exemplo a seguir, o objeto `Math` interno é posicionado no início da cadeia de escopo. Definir `Math` como o objeto padrão resolve os identificadores `cos`, `sin` e `PI` para `Math.cos`, `Math.sin` e `Math.PI`, respectivamente. Os identificadores `a`, `x`, `y` e `r` não são métodos ou propriedades do objeto `Math`, mas como existem no escopo de ativação do objeto da função `polar`, eles resolvem as variáveis locais correspondentes.

```
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)  
}
```



Você pode utilizar ações `with` aninhadas para ter acesso a informações em vários escopos. No exemplo a seguir, a instância `fresno` e a instância `salinas` são filhas da instância `california`. O código define os valores `_alpha` de `fresno` e `salinas` sem alterar o valor `_alpha` de `california`.

```
with (california){
  with (fresno){
    _alpha = 20;
  }
  with (salinas){
    _alpha = 40;
  }
}
```

#### Consulte também

`tellTarget`, na página 382

## **\_x**

#### Sintaxe

*nome da instância*.`_x`

*nome da instância*.`_x` = *inteiro*

#### Argumentos

*inteiro* A coordenada local *x* do filme.

*nome da instância* O nome da instância de um clipe de filme.

#### Descrição

Propriedade; define a coordenada *x* do filme relativa às coordenadas locais do clipe de filme pai. Se um clipe de filme estiver na Linha de Tempo principal, seu sistema de coordenadas se refere ao canto esquerdo superior do Palco como (0, 0). Se o clipe de filme estiver dentro de outro clipe de filme que possua modificações, será usado o sistema de coordenadas local do clipe de filme que o contém. Assim, para um clipe de filme girado 90° para a esquerda, o clipe filho herda um sistema de coordenadas que é girado 90° para a direita. As coordenadas do clipe de filme se referem à posição do ponto de registro.

#### Exibidor

Flash 3 ou posterior.

#### Consulte também

`_y`, na página 421

`_xscale`, na página 420



## XML (objeto)

Use os métodos e propriedades do objeto XML para carregar, analisar, enviar, montar e manipular árvores de documento XML.

Você deve usar o construtor `new XML()` para criar uma instância do objeto XML antes de chamar qualquer um de seus métodos.

O XML é suportado pelo Flash 5 e pelas versões posteriores do Flash Player.

### Resumo dos métodos do objeto XML

Método	Descrição
<code>appendChild</code>	Anexa um nó ao fim da lista filha do objeto especificado.
<code>cloneNode</code>	Clona o nó especificado e, opcionalmente, clona recursivamente todos os filhos.
<code>createElement</code>	Cria um novo elemento XML.
<code>createTextNode</code>	Cria um novo nó de texto XML.
<code>hasChildNodes</code>	Retorna <code>true</code> se o nó especificado tiver nós filhos; caso contrário, retorna <code>false</code> .
<code>insertBefore</code>	Insere um nó na frente de um nó existente na lista de filhos do nó especificado.
<code>load</code>	Carrega um documento (especificado pelo objeto XML) a partir de uma URL.
<code>onLoad</code>	Uma função de retorno de chamada para <code>load</code> e <code>sendAndLoad</code> .
<code>parseXML</code>	Analisa um documento XML na árvore de objeto XML especificada.
<code>removeNode</code>	Remove o nó especificado de seu pai.
<code>send</code>	Envia o objeto XML especificado para uma URL.
<code>sendAndLoad</code>	Envia o objeto XML especificado para uma URL e carrega a resposta do servidor em outro objeto XML.
<code>toString</code>	Converte o nó especificado e todos os seus filhos em texto XML.



## Resumo das propriedades do objeto XML

Propriedade	Descrição
docTypeDecl	Define e retorna informações sobre a declaração DOCTYPE de um documento XML.
firstChild	Faz referência ao primeiro filho na lista do nó especificado.
lastChild	Faz referência ao último filho na lista do nó especificado.
loaded	Verifica se o objeto XML especificado foi carregado.
nextSibling	Faz referência ao próximo irmão na lista de filhos do nó pai.
nodeName	Retorna o nome da marca de um elemento XML.
nodeType	Retorna o tipo do nó especificado (elemento XML ou nó de texto).
nodeValue	Retorna o texto do nó especificado se o nó for um nó de texto.
parentNode	Faz referência ao nó pai do nó especificado.
previousSibling	Faz referência ao irmão anterior na lista de filhos do nó pai.
status	Retorna um código de status numérico que indica o êxito ou a falha de uma operação de análise de um documento XML.
xmlDecl	Define e retorna informações sobre uma declaração de um documento XML.

## Resumo de coleções do objeto XML

Método	Descrição
attributes	Retorna um vetor associativo que contém todos os atributos do nó especificado.
childNodes	Retorna um vetor que contém referências aos nós filhos do nó especificado.

## Construtor do objeto XML

### Sintaxe

```
new XML();
```

```
new XML(origem);
```

### Argumentos

*origem* O documento XML analisado para criar o novo objeto XML.



#### Descrição

Construtor; cria um novo objeto XML. Você deve usar o método construtor para criar uma instância do objeto XML antes de chamar qualquer método do objeto XML.

A primeira sintaxe cria um objeto XML novo e vazio.

A segunda sintaxe cria um novo objeto XML analisando o documento XML especificado no argumento *origem*, e preenche o objeto XML recentemente criado com a árvore de documentos XML resultante.

**Observação:** Os métodos `createElement` e `createTextNode` são os métodos do 'construtor' para criar os elementos e nós de texto em uma árvore de documento XML.

#### Exibidor

Flash 5 ou posterior.

#### Exemplo

O exemplo a seguir cria um novo objeto XML vazio.

```
myXML = new XML();
```

#### Consulte também

`XML.createTextNode`, na página 399

`XML.createElement`, na página 399

## XML.appendChild

#### Sintaxe

```
myXML.appendChild(childNode);
```

#### Argumentos

*childNode* O nó filho a ser adicionado à lista de filhos do objeto XML especificado.

#### Descrição

Método; anexa o nó filho especificado à lista de filhos do objeto XML. O nó filho anexado é colocado na estrutura depois de removido de seu nó pai existente, se houver algum.

#### Exibidor

Flash 5 ou posterior.

#### Exemplo

O exemplo a seguir clona o último nó do `doc1` e o anexa ao `doc2`.

```
doc1 = new XML(src1);  
doc2 = new XML();  
node = doc1.lastChild.cloneNode(true);  
doc2.appendChild(node);
```

## XML.attributes

### Sintaxe

`myXML.attributes;`

### Argumentos

Nenhum.

### Descrição

Coleção (leitura gravação); retorna um vetor associativo que contém todos os atributos do objeto XML especificado.

### Exibidor

Flash 5 ou posterior.

### Exemplo

O exemplo a seguir grava os nomes dos atributos XML na janela Saída.

```
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);
```

A seguir está o que é escrito janela Saída:

Val  
Primeiro:

## XML.childNodes

### Sintaxe

`myXML.childNodes;`

### Argumentos

Nenhum.

### Descrição

Coleção (somente leitura); retorna um vetor dos filhos do objeto XML especificado. Cada elemento no vetor é uma referência a um objeto XML que representa um nó filho. Essa é uma propriedade somente leitura e não pode ser usada para manipular nós filhos. Use os métodos `appendChild`, `insertBefore`, e `removeNode` para manipular nós filhos.

Essa coleção não é definida para os nós de texto (`nodeType == 3`).

### Exibidor

Flash 5 ou posterior.

## XML.cloneNode

### Sintaxe

`myXML.cloneNode(profundidade);`

### Argumentos

*profundidade* Valor booleano que especifica se os filhos do objeto XML específico são clonados de forma recursiva.

### Descrição

Método; cria e retorna um novo nó XML do mesmo tipo, valor, nome e atributos do objeto XML especificado. Se *profundidade* for definido como `true`, todos os nós filhos são clonados de forma recursiva, resultando em uma cópia exata da árvore de documentos do objeto original.

### Exibidor

Flash 5 ou posterior.

## XML.createElement

### Sintaxe

`myXML.createElement(nome);`

### Argumentos

*nome* O nome da marca do elemento XML que está sendo criado.

### Descrição

Método; cria um novo elemento XML com o nome especificado no argumento. Inicialmente, o novo elemento não tem pai nem filhos. O método retorna uma referência ao objeto XML criado recentemente que representa o elemento. Esse método e `createTextNode` são os métodos construtores para criação de nós de um objeto XML.

### Exibidor

Flash 5 ou posterior.

## XML.createTextNode

### Sintaxe

`myXML.createTextNode(texto);`

### Argumentos

*texto* O texto usado para criar o novo nó de texto.

### Descrição

Método; cria um novo nó de texto XML com o texto especificado. Inicialmente, o novo nó não tem pai, e os nós texto não podem ter filhos. Esse método retorna uma referência ao objeto XML que representa o novo nó de texto. Esse método e o `createElement` são os métodos do construtor para criação de nós de um objeto XML.

### Exibidor

Flash 5 ou posterior.

## XML.docTypeDecl

### Sintaxe

```
myXML.XMLdocTypeDecl;
```

### Argumentos

Nenhum.

### Descrição

Propriedade; define e retorna informações sobre a declaração DOCTYPE do documento XML. Depois que o texto XML tiver sido analisado em um objeto XML, a propriedade `XML.docTypeDecl` do objeto XML é definida como o texto da declaração DOCTYPE do documento XML. Por exemplo, `<!DOCTYPE greeting SYSTEM "hello.dtd">`. Essa propriedade é definida usando uma representação de sequência de caracteres da declaração DOCTYPE, não de um objeto do nó XML.

O analisador XML do ActionScript não é um analisador de validação. A declaração DOCTYPE é lida pelo analisador e armazenada na propriedade `docTypeDecl`, mas nenhuma validação DTD é executada.

Se nenhuma declaração DOCTYPE for encontrada durante a operação de análise, `XML.docTypeDecl` é definido como indefinido. `XML.toString` mostra o conteúdo de `XML.docTypeDecl` imediatamente depois da declaração XML armazenada em `XML.xmlDecl`, e antes de qualquer outro texto no objeto XML. Se `XML.docTypeDecl` não for definido, nenhuma declaração DOCTYPE é mostrada.

### Exibidor

Flash 5 ou posterior.

### Exemplo

O exemplo a seguir usa `XML.docTypeDecl` para definir a declaração DOCTYPE de um objeto XML.

```
myXML.docTypeDecl = "<!DOCTYPE greeting SYSTEM \"hello.dtd\">";
```

### Consulte também

`XML.toString`, na página 410

`XML.xmlDecl`, na página 411

## XML.firstChild

### Sintaxe

`myXML.firstChild;`

### Argumentos

Nenhum.

### Descrição

Propriedade (somente leitura); avalia o objeto XML especificado e faz referência ao primeiro filho na lista de filhos do nó pai. Essa propriedade é `null` se o nó não tiver filhos. Essa propriedade é indefinida se o nó for um nó de texto. Essa é uma propriedade somente leitura e não pode ser usada para manipular nós filhos; use os métodos `appendChild`, `insertBefore` e `removeNode` para manipulá-los.

### Exibidor

Flash 5 ou posterior.

### Consulte também

`XML.appendChild`, na página 397

`XML.insertBefore`, na página 402

`XML.removeNode`, na página 408

## XML.hasChildNodes

### Sintaxe

`myXML.hasChildNodes();`

### Argumentos

Nenhum.

### Descrição

Método; avalia o objeto XML especificado e retorna `true` se houver nós filhos; caso contrário, retorna `false`.

### Exibidor

Flash 5 ou posterior.

### Exemplo

O exemplo a seguir usa as informações do objeto XML em uma função definida pelo usuário.

```
if (rootNode.hasChildNodes()) {  
    myfunc (rootNode.firstChild);  
}
```

## XML.insertBefore

### Sintaxe

`myXML.insertBefore(childNode, beforeNode);`

### Argumentos

*childNode* O nó a ser inserido.

*beforeNode* O nó antes do ponto de inserção do *childNode*.

### Descrição

Método; insere um novo nó filho na lista de filhos do objeto XML, antes de *beforeNode*.

### Exibidor

Flash 5 ou posterior.

## XML.lastChild

### Sintaxe

`myXML.lastChild;`

### Argumentos

Nenhum.

### Descrição

Propriedade (somente leitura); avalia o objeto XML e faz referência ao último nó filho na lista de filhos do nó pai. Esse método retorna `null` se o nó não tiver filhos. Essa é uma propriedade somente leitura e não pode ser usada para manipular nós filhos; use os métodos `appendChild`, `insertBefore` e `removeNode` para manipulá-los.

### Exibidor

Flash 5 ou posterior.

### Consulte também

`XML.appendChild`, na página 397

`XML.insertBefore`, na página 402

`XML.removeNode`, na página 408

## XML.load

### Sintaxe

```
myXML.load(url);
```

### Argumentos

*url* A URL onde o documento XML a ser carregado está localizada. A URL deve estar no mesmo subdomínio que a URL onde o filme reside no momento.

### Descrição

Método; carrega um documento XML da URL especificada e substitui o conteúdo do objeto XML especificado pelo objeto com os dados XML descarregados. O processo de carregamento é assíncrono; ele não termina imediatamente após o método `load` ser carregado. Quando `load` é executado, a propriedade do objeto XML `loaded` é definida como `false`. Quando os dados XML terminam de descarregar, a propriedade `loaded` é definida como `true` e o método `onLoad` é chamado. Os dados XML não são analisados até que sejam totalmente descarregados. Se o objeto XML continha anteriormente árvores XML, elas são descartadas.

Você pode especificar sua própria função de chamada no lugar do método `onLoad`.

### Exibidor

Flash 5 ou posterior.

### Exemplo

A seguir temos um simples exemplo usando `XML.load`.

```
doc = new XML();  
doc.load ("theFile.xml");
```

### Consulte também

`XML.onLoad`, na página 406

`XML.loaded`, na página 403

## XML.loaded

### Sintaxe

```
myXML.loaded;
```

### Argumentos

Nenhum.

### Descrição

Propriedade (somente leitura); determina se o processo de carregamento do documento iniciado pela chamada `XML.load` foi concluído. Se o processo for concluído com êxito, o método retorna `true`; caso contrário, ele retorna `false`.

#### Exibidor

Flash 5 ou posterior.

#### Exemplo

O exemplo a seguir usa o `XML.loaded` em um script simples.

```
if (doc.loaded) {  
    gotoAndPlay(4)  
}
```

## XML.nextSibling

#### Sintaxe

`myXML.nextSibling;`

#### Argumentos

Nenhum.

#### Descrição

Propriedade (somente leitura); avalia o objeto XML e faz referência ao próximo irmão na lista de filhos do nó pai. Esse método retorna `null` se o nó não tiver um nó irmão próximo. Essa é uma propriedade somente leitura e não pode ser usada para manipular nós filhos. Use os métodos `appendChild`, `insertBefore` e `removeNode` para manipulá-los.

#### Exibidor

Flash 5 ou posterior.

#### Consulte também

`XML.appendChild`, na página 397

`XML.insertBefore`, na página 402

`XML.removeNode`, na página 408

## XML.nodeName

#### Sintaxe

`myXML.nodeName;`

#### Argumentos

Nenhum.

#### Descrição

Propriedade; considera ou retorna o nome do objeto XML. Se o objeto XML for um elemento XML (`nodeType == 1`), `nodeName` é o nome da marca que representa o nó no arquivo XML. Por exemplo, `TITLE` é o `nodeName` de uma marca `TITLE` em HTML. Se o objeto XML for um nó texto (`nodeType == 3`), o `nodeName` é `null`.



**Exibidor**  
Flash 5 ou posterior.

**Consulte também**  
XML.nodeType, na página 405

## XML.nodeType

**Sintaxe**  
`myXML.nodeType;`

**Argumentos**  
Nenhum.

**Descrição**  
Propriedade (somente leitura); considera ou retorna um valor `nodeType`, onde 1 é um elemento XML e 3 é um nó texto.

**Exibidor**  
Flash 5 ou posterior.

**Consulte também**  
XML.nodeValue, na página 405

## XML.nodeValue

**Sintaxe**  
`myXML.nodeValue;`

**Argumentos**  
Nenhum.

**Descrição**  
Propriedade; retorna o valor do nó do objeto XML. Se o objeto XML for um nó texto, o `nodeType` é 3, e o `nodeValue` é o texto de nó. Se o objeto XML for um elemento XML, ele tem um `nodeValue null` e é somente leitura.

**Exibidor**  
Flash 5 ou posterior.

**Consulte também**  
XML.nodeType, na página 405



## XML.onLoad

### Sintaxe

`myXML.onLoad(êxito);`

### Argumentos

*êxito* Um valor booleano que indica se o objeto XML foi carregado com êxito com uma operação `XML.load` ou `XML.sendAndLoad`.

### Descrição

Método; chamado pelo Flash Player quando um documento XML é recebido do servidor. Se o documento XML for recebido com êxito, o argumento *êxito* é `true`. Se o documento não for recebido, ou se ocorreu algum erro ao receber a resposta do servidor, o argumento *êxito* é `false`. A implementação padrão deste método não está ativa. Para substituir a implementação padrão, atribua uma função que contém suas próprias ações.

### Exibidor

Flash 5 ou posterior.

### Exemplo

O exemplo a seguir cria um filme do Flash simples para um aplicativo de comércio eletrônico. Nós usamos o método `sendAndLoad` para transmitir um elemento XML que contém o nome e a senha do usuário, e instalar um manipulador `onLoad` para lidar com a resposta do servidor.

```
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")
    }
}
```

### Consulte também

function, na página 272

XML.load, na página 403

XML.sendAndLoad, na página 408

## XML.parentNode

### Sintaxe

`myXML.parentNode;`

### Argumentos

Nenhum.

### Descrição

Propriedade (somente leitura); faz referência ao nó pai do objeto XML especificado, ou retorna `null` se o nó não tiver pai. Essa é uma propriedade somente leitura e não pode ser usada para manipular nós filhos; use os métodos `appendChild`, `insertBefore`, e `removeNode` para manipular os filhos.

### Exibidor

Flash 5 ou posterior.

## XML.parseXML

### Sintaxe

`myXML.parseXML(origem);`

### Argumentos

*origem* O texto XML a ser analisado e transmitido para o objeto XML especificado.

### Descrição

Método; analisa do texto XML especificado no argumento *origem*, e preenche o objeto XML especificado com a árvore XML resultante. Quaisquer árvores existentes no objeto XML são descartadas.

### Exibidor

Flash 5 ou posterior.

## XML.previousSibling

### Sintaxe

`myXML.previousSibling;`

### Descrição

Propriedade (somente leitura); avalia o objeto XML e faz referência ao irmão anterior na lista de filhos do nó pai. Retorna `null` se o nó não tiver um nó irmão anterior. Essa é uma propriedade somente leitura e não pode ser usada para manipular nós filhos; use os métodos `appendChild`, `insertBefore` e `removeNode` para manipular nós filhos.

### Exibidor

Flash 5 ou posterior.

## XML.removeNode

### Sintaxe

```
myXML.removeNode();
```

### Argumentos

Nenhum.

### Descrição

Método; remove o objeto XML especificado de seu pai.

### Exibidor

Flash 5 ou posterior.

## XML.send

### Sintaxe

```
myXML.send(url);  
myXML.send(url, janela);
```

### Argumentos

*url* A URL de destino do objeto XML especificado.

*janela* A janela do navegador para exibir os dados retornados pelo servidor: *\_self* especifica o quadro atual na janela atual, *\_blank* especifica uma nova janela, *\_parent* especifica o pai do quadro atual e *\_top* especifica o quadro de alto nível na janela atual.

### Descrição

Método; codifica o objeto XML em um documento XML e o envia para a URL especificada usando o método POST.

### Exibidor

Flash 5 ou posterior.

## XML.sendAndLoad

### Sintaxe

```
myXML.sendAndLoad(url, targetXMLObject);
```

### Argumentos

*url* A URL de destino do objeto XML especificado. A URL deve estar no mesmo subdomínio que a URL de onde o filme foi descarregado.

*targetXMLObject* Um objeto XML criado com o método construtor XML que receberá as informações de retorno do servidor.

#### Descrição

Método; codifica o objeto XML especificado em um documento XML, o envia para a URL especificada usando o método POST, descarrega a resposta do servidor e carrega-a no *targetXMLObject* especificado nos argumentos. A resposta do servidor é carregada da mesma maneira usada pelo método load.

#### Exibidor

Flash 5 ou posterior.

#### Consulte também

XML.load, na página 403

## XML.status

#### Sintaxe

```
myXML.status;
```

#### Argumentos

Nenhum.

#### Descrição

Propriedade; define automaticamente e retorna um valor numérico indicando se o documento XML foi analisado com êxito no objeto XML. A seguir, temos uma lista de códigos de status numéricos e uma descrição de cada.

- 0 Sem erro; análise concluída com êxito.
- -2 Uma seção CDATA não foi terminada adequadamente.
- -3 A declaração XML não foi terminada adequadamente.
- -4 A declaração DOCTYPE não foi terminada adequadamente.
- -5 Um comentário não foi terminado adequadamente.
- -6 Um elemento XML foi mal formado.
- -7 Out of memory.
- -8 Um valor de atributo não foi terminado adequadamente.
- -9 Uma marca de início não correspondeu a uma marca de fim.
- -10 Foi encontrada uma marca de fim sem uma marca de início correspondente.

#### Exibidor

Flash 5 ou posterior.

## XML.toString

### Sintaxe

`myXML.toString();`

### Argumentos

Nenhum.

### Descrição

Método; avalia o objeto XML, constrói uma representação de texto da estrutura XML incluindo o nó, filhos e atributos, e retorna o resultado como uma sequência de caracteres.

Para objetos XML de alto nível (os criados com o construtor), `XML.toString` mostra a declaração XML do documento (armazenado em `XML.xmlDecl`), seguido pela declaração `DOCTYPE` do documento (armazenada em `XML.docTypeDecl`), seguido pela representação do texto de todos os nós XML no objeto. A declaração XML não é mostrada se `XML.xmlDecl` for indefinido. A declaração `DOCTYPE` não é mostrada se `XML.docTypeDecl` for indefinido.

### Exibidor

Flash 5 ou posterior.

### Exemplo

O código a seguir é um exemplo do método `XML.toString`.

```
node = new XML("<h1>test</h1>");  
trace(node.toString());  
sends  
<H1>test</H1>  
para a janela de saída
```

### Consulte também

`XML.xmlDecl`, na página 411

`XML.docTypeDecl`, na página 400

## XML.xmlDecl

### Sintaxe

`myXML.xmlDecl;`

### Argumentos

Nenhum.

### Descrição

Propriedade; define e retorna informações sobre uma declaração XML do documento. Depois que o documento XML é analisado em um objeto XML, essa propriedade é definida usando o texto da declaração XML do documento. Essa propriedade é definida usando uma representação de sequência de caracteres da declaração XML, não de um objeto do nó XML. Se nenhuma declaração XML foi encontrada durante a operação de análise, a propriedade é definida como indefinida. `XML.toString` mostra o conteúdo de `XML.xmlDecl` antes de qualquer outro texto no objeto XML. Se `XML.xmlDecl` contiver o tipo indefinido, nenhuma declaração XML é mostrada.

### Exibidor

Flash 5 ou posterior.

### Exemplo

O exemplo a seguir usa `XML.xmlDecl` para definir a declaração do documento XML de um objeto XML.

```
myXML.xmlDecl = "<?xml version=\"1.0\" ?>";
```

### Consulte também

`XML.toString`, na página 410

`XML.docTypeDecl`, na página 400

## XMLSocket (objeto)

O objeto XMLSocket implementa soquetes do cliente que permitem que o computador que está executando o Flash Player se comunique com um computador servidor identificado pelo endereço IP ou nome de domínio.

### Usando o objeto XMLSocket

Para usar o objeto XMLSocket, o computador servidor deve executar um daemon que compreenda o protocolo usado pelo objeto XMLSocket. O protocolo é o seguinte:

- Mensagens XML são enviadas através de uma conexão de soquetes de fluxo TCP/IP full-duplex.
- Cada mensagem XML é um documento XML completo, terminado por um byte zero.
- Um número ilimitado de mensagens XML pode ser enviado e recebido por uma conexão XMLSocket.

O objeto XMLSocket é útil para aplicativos cliente servidor que requerem uma latência baixa, como sistemas de bate-papo em tempo real. Uma solução de bate-papo baseada em HTTP pesquisa o servidor frequentemente e descarrega novas mensagens usando uma solicitação HTTP. Comparando, uma solução de bate-papo XMLSocket mantém uma conexão aberta com o servidor, o que permite que o servidor envie mensagens de chegada imediatamente sem uma solicitação do cliente.

Configurar um servidor para se comunicar com o objeto XMLSocket pode ser difícil. Se o seu aplicativo não requerer interatividade em tempo real, use a ação `loadVariables`, ou a conectividade do servidor XML baseado em HTTP do Flash (`XML.load`, `XML.sendAndLoad`, `XML.send`), em vez do objeto XMLSocket.

Para usar os métodos do objeto XMLSocket, use a construtora `new XMLSocket` para criar um novo objeto XMLSocket.





## XMLSocket e segurança

Como o objeto XMLSocket estabelece e mantém uma conexão aberta com o servidor, as restrições a seguir foram colocadas no objeto XMLSocket por motivos de segurança.

- O método `XMLSocket.connect` pode ser conectado somente com números de porta TCP maiores ou iguais a 1024. Uma consequência dessa restrição é que os daemons do servidor que se comunicam com o objeto XMLSocket também devem ser atribuídos a números de porta maiores ou iguais a 1024. Os números de porta abaixo de 1024 são usados, freqüentemente, pelos serviços de sistema como FTP, Telnet, e HTTP, barrando, dessa forma, o objeto XMLSocket dessas portas. A restrição do número de porta limita a possibilidade desses recursos serem acessados e abusados de forma não adequada.
- O método `XMLSocket.connect` pode se conectar somente a computadores no mesmo subdomínio onde o arquivo SWF (filme) reside. Essa restrição não se aplica aos filmes que estejam sendo executados fora de um disco local. (Essa restrição é idêntica às regras de segurança do `loadVariables`, `XML.sendAndLoad` e `XML.load`.)

## Resumo de métodos do objeto XMLSocket

Método	Descrição
<code>close</code>	Fecha uma conexão de soquete aberta.
<code>connect</code>	Estabelece uma conexão com o servidor especificado.
<code>onClose</code>	Uma função de chamada que é chamada quando uma conexão XMLSocket é fechada.
<code>onConnect</code>	Uma função de chamada que é chamada quando uma conexão XMLSocket é estabelecida.
<code>onXML</code>	Uma função de chamada que é chamada quando um objeto XML chega do servidor.
<code>send</code>	Envia um objeto XML para o servidor.

## Construtor do objeto XMLSocket

### Sintaxe

```
new XMLSocket();
```

### Argumentos

Nenhum.

### Descrição

Construtor; cria um novo objeto XMLSocket. O objeto XMLSocket não é conectado inicialmente com qualquer servidor. Você deve chamar o método `XMLSocket.connect` para conectar o objeto ao servidor.

### Exibidor

Flash 5 ou posterior.

### Exemplo

```
myXMLSocket = new XMLSocket();
```

### Consulte também

`XMLSocket.connect`, na página 414

## XMLSocket.close

### Sintaxe

```
myXMLSocket.close();
```

### Argumentos

Nenhum.

### Descrição

Método; fecha a conexão especificada pelo objeto XMLSocket.

### Exibidor

Flash 5 ou posterior.

### Consulte também

`XMLSocket.connect`, na página 414

## XMLSocket.connect

### Sintaxe

```
myXMLSocket.connect(host, porta);
```

### Argumentos

*host* Um nome de domínio DNS totalmente qualificado, ou um endereço IP na forma *aaa.bbb.ccc.ddd*. Você também pode especificar `null` para se conectar ao servidor host no qual o filme reside.

*porta* O número da porta TCP no host usado para estabelecer uma conexão. O número da porta deve ser 1024 ou superior.

### Descrição

Método; estabelece uma conexão com o host de Internet especificado usando a porta TCP especificada (deve ser 1024 ou superior), e retorna `true` ou `false` dependendo do êxito da conexão. Se você não sabe o número da porta de sua máquina host de Internet, entre em contato com o administrador da rede. Se o plug-in Flash Netscape ou controle ActiveX estiver sendo usado, o host especificado no argumento deve ter o mesmo subdomínio do host do qual o filme foi descarregado.

Se você especificar `null` para o argumento `host`, o host que foi contactado será o host onde o filme que chama o `XMLSocket.connect` reside. Por exemplo, se o filme foi descarregado de `http://www.seusite.com`, especificar `null` para o argumento `host` é o mesmo que inserir o endereço IP em `www.seusite.com`.

Se `XMLSocket.connect` retorna um valor `true`, o palco inicial do processo da conexão obteve êxito; mais tarde, o método `XMLSocket.onConnect` é chamado para determinar se a conexão final obteve êxito ou falhou. Se `XMLSocket.connect` retorna `false`, uma conexão não pode ser estabelecida.

### Exibidor

Flash 5 ou posterior.

### Exemplo

O exemplo a seguir usa `XMLSocket.connect` para se conectar com o host onde o filme reside, e usa `trace` para exibir o valor de retorno que indica o êxito ou falha da conexão.

```
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!")  
}
```

### Consulte também

`function`, na página 272

`XMLSocket.onConnect`, na página 416

## XMLSocket.onClose

### Sintaxe

`myXMLSocket.onClose();`

### Argumentos

Nenhum.

### Descrição

Método; uma função de chamada que é chamada somente quando uma conexão aberta é fechada pelo servidor. A implementação padrão desse método não executa nenhuma ação. Para substituir a implementação padrão, atribua uma função que contém suas próprias ações.

### Exibidor

Flash 5 ou posterior.

### Consulte também

`function`, na página 272

`XMLSocket.onConnect`, na página 416

## XMLSocket.onConnect

### Sintaxe

`myXMLSocket.onConnect(êxito);`

### Argumentos

*êxito* Um valor booleano que indica se uma conexão de soquete foi estabelecida com êxito (`true` ou `false`).

### Descrição

Método; uma função de retorno de chamada chamada pelo Flash Player quando uma solicitação de conexão iniciada pelo método `XMLSocket.connect` obtém êxito ou falha. Se a conexão obtém êxito, o argumento *êxito* é `true`; caso contrário, o argumento *êxito* é `false`.

A implementação padrão desse método não executa nenhuma ação. Para substituir a implementação padrão, atribua uma função que contém suas próprias ações.

### Exibidor

Flash 5 ou posterior.

### Exemplo

O exemplo a seguir ilustra o processo de especificação de uma função de substituição do método `onConnect` em uma aplicação de bate-papo simples.

A função controla para qual tela os usuários são levados, dependendo do êxito da conexão estabelecida. Se a conexão for estabelecida, os usuários são levados para a tela de bate-papo principal no quadro chamado `startChat`. Se a conexão não tiver êxito, os usuários vão para uma tela com as informações de solução de problemas no quadro rotulado `connectionFailed`.

```
function myOnConnect(success) {  
    if (success) {  
        gotoAndPlay("startChat")  
    } else {  
        gotoAndStop("connectionFailed")  
    }  
}
```

Depois de criar o objeto `XMLSocket` usando o método construtor, o script instala no método `onConnect` usando o operador de atribuição:

```
socket = new XMLSocket()  
socket.onConnect = myOnConnect
```

Finalmente, a conexão é iniciada. Se a conexão retornar `false`, o filme é enviado para o quadro chamado `connectionFailed`, e `onConnect` nunca é chamado. Se a conexão retornar `true`, o filme pula para um quadro chamado `waitForConnection`, que é a tela "Please wait". O filme permanece no quadro `waitForConnection` até que o manipulador `onConnect` seja chamado, o que acontece em algum momento no futuro dependendo da latência da rede.

```
if (!socket.connect(null, 2000)) {  
    gotoAndStop("connectionFailed")  
} else {  
    gotoAndStop("waitForConnection")  
}
```

### Consulte também

`XMLSocket.connect`, na página 414

função, na página 272

## XMLSocket.onXML

### Sintaxe

`myXMLSocket.onXML(objeto);`

### Argumento

*objeto* Uma instância do objeto XML que contém um documento XML recebido de um servidor.

### Descrição

Método; uma função de retorno de chamada chamada pelo Flash Player quando o objeto XML especificado que contém um documento XML chega através de uma conexão XMLSocket aberta. Uma conexão XMLSocket pode ser usada para transferir um número ilimitado de documentos XML entre o cliente e o servidor. Cada documento é terminado com um byte zero. Quando o Flash Player recebe o byte zero, ele analisa todo o XML recebido desde o byte zero anterior ou desde que a conexão foi estabelecida, se essa for a primeira mensagem recebida. Cada lote de XML analisado é tratado como um único documento XML e passado para o método `onXML`.

A implementação padrão desse método não executa ações. Para substituir a implementação padrão, atribua uma função que contém ações definidas por você.

### Exibidor

Flash 5 ou posterior.

### Exemplo

A função a seguir substitui a implementação padrão do método `onXML` em um aplicativo de bate-papo simples. A função `myOnXML` instrui o aplicativo de bate-papo para reconhecer um único elemento XML, `MESSAGE`, no seguinte formato:

```
<MESSAGE USER="John" TEXT="Hello, my name is John!" />.
```

O manipulador `onXML` deve primeiro ser instalado no objeto XMLSocket da seguinte forma:

```
socket.onXML = myOnXML;
```

A função `displayMessage` deve ser uma função definida pelo usuário que exibe a mensagem recebida para o usuário.

```
function myOnXML(doc) {  
    var e = doc.firstChild;  
    if (e != null && e.nodeName == "MESSAGE") {  
        displayMessage(e.attributes.user, e.attributes.text);  
    }  
}
```

### Consulte também

function, na página 272



## XMLSocket.send

### Sintaxe

`myXMLSocket.send(objeto);`

### Argumentos

*objeto* Um objeto XML ou outros dados a serem transmitidos para o servidor.

### Descrição

Método; converte o objeto ou dados XML especificados no argumento *objeto* em uma sequência de caracteres e transmite para o servidor, seguido por um byte zero. Se o *objeto* é um objeto XML, a sequência de caracteres é a representação textual XML do objeto XML. A operação `send` é assíncrona; ela é retornada imediatamente, mas os dados podem ser transmitidos posteriormente. O método `XMLSocket.send` não retorna um valor que indica se os dados foram transmitidos com êxito.

Se o objeto *myXMLSocket* não for conectado com o servidor (usando `XMLSocket.connect`), a operação `XMLSocket.send` irá falhar.

### Exibidor

Flash 5 ou posterior.

### Exemplo

O exemplo a seguir ilustra como você poderia especificar um nome e senha de usuário para enviar o objeto XML *myXML* para o servidor.

```
var myXML = new XML();  
var myLogin = myXML.createElement("login");  
myLogin.attributes.username = usernameTextField;  
myLogin.attributes.password = passwordTextField;  
myXML.appendChild(myLogin);  
myXMLSocket.send(myXML);
```

### Consulte também

`XMLSocket.connect`, na página 414



## **\_xmouse**

### **Sintaxe**

*nome da instância*.\_xmouse

### **Argumentos**

*nome da instância* O nome da instância do clipe de filme.

### **Descrição**

Propriedade (somente leitura); retorna a coordenada *x* da posição do mouse.

### **Exibidor**

Flash 5 ou posterior.

### **Consulte também**

Mouse (objeto), na página 312

\_ymouse, na página 421

## **\_xscale**

### **Syntax**

*nome da instância*.\_xscale

*nome da instância*.\_xscale = *porcentagem*;

### **Argumentos**

*porcentagem* Um valor em porcentagem que especifica a porcentagem para o dimensionamento horizontal do filme. O valor padrão é 100.

*nome da instância* O nome de uma instância de clipe de filme.

### **Descrição**

Propriedade; determina o dimensionamento horizontal (*porcentagem*) do clipe de filme como aplicado do ponto do registro do clipe de filme. O ponto de registro padrão é (0,0).

Dimensionar o sistema de coordenadas local afeta as configurações das propriedades *\_x* e *\_y*, que são definidas em pixels. Por exemplo, se o clipe de filme pai for dimensionado para 50%, definir a propriedade *\_x* move um objeto no clipe de filme pela metade do número de pixels, como se o filme tivesse sido dimensionado em 100%.

### **Exibidor**

Flash 4 ou posterior.

### **Consulte também**

\_xscale, na página 420



## **\_y**

### **Sintaxe**

*nome da instância*.**\_y**  
*nome da instância*.**\_y** = inteiro;

### **Argumentos**

*inteiro* A coordenada y local do clipe de filme.

*nome da instância* O nome da instância do clipe de filme.

### **Descrição**

Propriedade; define a coordenada y do filme relativa às coordenadas locais do clipe de filme pai. Se um clipe de filme estiver na Linha de Tempo principal, seu sistema de coordenadas refere-se ao canto superior esquerdo do Palco como (0, 0). Se o clipe de filme estiver dentro de outro clipe de filme que tem transformações, o clipe de filme está no sistema de coordenadas local do clipe de filme anexado. Contudo, para um clipe de filme girado 90° para a direita, os filhos do clipe de filme herdam um sistema de coordenadas que é girado 90° para a direita. As coordenadas do clipe de filme referem-se à posição do ponto do registro.

### **Exibidor**

Flash 3 ou posterior.

### **Consulte também**

**\_yscale**, na página 422

## **\_ymouse**

### **Sintaxe**

*nome da instância*.**\_ymouse**

### **Argumentos**

*nome da instância* O nome da instância de um clipe de filme.

### **Descrição**

Propriedade (somente leitura); indica a coordenada y da posição do mouse.

### **Exibidor**

Flash 5 ou posterior.

### **Consulte também**

**Mouse** (objeto), na página 312

**\_xmouse**, na página 420

## **\_yscale**

### **Sintaxe**

*nome da instância*.\_yscale  
*nome da instância*.\_yscale = *porcentagem*;

### **Argumentos**

*porcentagem* Um valor em porcentagem que especifica o dimensionamento vertical do filme. O valor padrão é 100.

*nome da instância* O nome da instância do clipe de filme.

### **Descrição**

Propriedade; define a escala vertical (*porcentagem*) do clipe de filme conforme aplicado do ponto de registro do clipe de filme. O ponto de registro padrão é (0,0).

Dimensionar o sistema de coordenadas local afeta as configurações da propriedade *\_x* e *\_y*, que são definidas em pixels. Por exemplo, se o clipe de filme pai é dimensionado em 50%, definir a propriedade *\_x* move um objeto no clipe de filme pela metade do número de pixels, como se o filme tivesse sido dimensionado em 100%.

### **Exibidor**

Flash 4 ou posterior.

### **Consulte também**

*\_x*, na página 394

*\_y*, na página 421





# APÊNDICE A

## Associatividade e precedência de operadores

### Lista de operadores

Esta tabela lista todos os operadores do Action Script e sua associatividade, da precedência mais alta para a mais baixa.

Operador	Descrição	Associatividade
<b>Precedência mais alta</b>		
+	Mais unário	Direita para a esquerda
-	Menos unário	Direita para a esquerda
~	Complemento de um bit a bit	Direita para a esquerda
!	NOT lógico	Direita para a esquerda
not	NOT lógico (estilo do Flash 4)	Direita para a esquerda
++	Pós-incremento	Esquerda para a direita
--	Pós-decremento	Esquerda para a direita
()	Chamada de função	Esquerda para a direita



Operador	Descrição	Associatividade
[ ]	Elemento de matriz	Esquerda para a direita
.	Membro da estrutura	Esquerda para a direita
++	Pré-incremento	Direita para a esquerda
--	Pré-decremento	Direita para a esquerda
new	Alocar objeto	Direita para a esquerda
delete	Desalocar objeto	Direita para a esquerda
typeof	Tipo de objeto	Direita para a esquerda
void	Retorna um valor indefinido	Direita para a esquerda
*	Multiplicar	Esquerda para a direita
/	Dividir	Esquerda para a direita
%	Módulo	Esquerda para a direita
+	Adicionar	Esquerda para a direita
add	Concatenação de seqüência de caracteres (antes &)	Esquerda para a direita
-	Subtrair	Esquerda para a direita
<<	Deslocamento para a esquerda bit a bit	Esquerda para a direita
>>	Deslocamento para a direita bit a bit	Esquerda para a direita
>>>	Deslocamento para a direita bit a bit (sem sinal)	Esquerda para a direita
<	Menor que	Esquerda para a direita



Operador	Descrição	Associatividade
<=	Menor ou igual a	Esquerda para a direita
>	Maior que	Esquerda para a direita
>=	Maior ou igual a	Esquerda para a direita
lt	Menor que (versão de seqüência de caracteres)	Esquerda para a direita
le	Menor ou igual a (versão de seqüência de caracteres)	Esquerda para a direita
gt	Maior que (versão de seqüência de caracteres)	Esquerda para a direita
ge	Maior ou igual a (versão de seqüência de caracteres)	Esquerda para a direita
==	Igual	Esquerda para a direita
!=	Diferente	Esquerda para a direita
eq	Igual (versão de seqüência de caracteres)	Esquerda para a direita
ne	Diferente (versão de seqüência de caracteres)	Esquerda para a direita
&	AND bit a bit	Esquerda para a direita
^	XOR bit a bit	Esquerda para a direita
	OR bit a bit	Esquerda para a direita
&&	AND lógico	Esquerda para a direita
and	AND lógico (Flash 4)	Esquerda para a direita
	OR lógico	Esquerda para a direita



Operador	Descrição	Associatividade
or	OR lógico (Flash 4)	Esquerda para a direita
?:	Condicional	Direita para a esquerda
=	Atribuição	Direita para a esquerda
"*=, /=, %=, +=, -=, &=,  =, ^=, <<=, >>=, >>="	Atribuição composta	Direita para a esquerda
,	Avaliação múltipla	Esquerda para a direita
<b>Precedência mais baixa</b>		



## APÊNDICE B

### Teclas do Teclado e Valores de Códigos de Teclas

.....

As tabelas a seguir listam todas as teclas de um teclado padrão e os valores de códigos de teclas correspondentes que são usados para identificar as teclas no ActionScript. Para obter mais informações, consulte a descrição do objeto Key no Capítulo 7, “Dicionário do ActionScript”.



## Letras de A a Z e números padrão de 0 a 9

Tecla alfabética ou numérica	Código da tecla
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





Tecla alfabética ou numérica	Código da tecla
S	83
T	84
U	85
V	86
W	87
X	88
Y	89
Z	90
0	48
1	49
2	50
3	51
4	52
5	53
6	54
7	55
8	56
9	57



## Teclas do teclado numérico

Tecla do teclado numérico	Código da tecla
Numpad 0	96
Numpad 1	97
Numpad 2	98
Numpad 3	99
Numpad 4	100
Numpad 5	101
Numpad 6	102
Numpad 7	103
Numpad 8	104
Numpad 9	105
Multiplicar	106
Adicionar	107
Enter	108
Subtrair	109
Decimal	110
Dividir	111

## Teclas de função

Tecla de função	Código da tecla
F1	112
F2	113
F3	114
F4	115
F5	116
F6	117
F7	118

Tecla de função	Código da tecla
F8	119
F9	120
F10	121
F11	122
F12	123

## Outras teclas

Tecla	Código da tecla
Backspace	8
Tab	9
Clear	12
Enter	13
Shift	16
Control	17
Alt	18
Caps Lock	20
Esc	27
Spacebar	32
Page Up	33
Page Down	34
End	35
Home	36
Seta para Esquerda	37
Seta para Cima	38
Seta para Direita	39
Seta para Baixo	40
Insert	45



Tecla	Código da tecla
Delete	46
Help	47
Num Lock	144
::	186
= +	187
- _	189
/ ?	191
` ~	192
[ {	219
\	220
] }	221
" '	222



## APÊNDICE C

### Mensagens de erro

.....

A tabela a seguir contém uma lista das mensagens de erro retornadas pelo compilador do Flash. Uma explicação de cada mensagem é fornecida para ajudá-lo a solucionar problemas com arquivos de filme.

Mensagem de erro	Descrição
Propriedade <propriedade> não existe	Uma propriedade inexistente foi encontrada. Por exemplo, <code>x = _green</code> é inválido, pois não existe nenhuma propriedade <code>_green</code> .
O operador <operador> deve ser seguido de um operando	Um operador sem um operando foi encontrado. Por exemplo, <code>x = 1 +</code> requer um operando após o operador <code>+</code> . Um operador é seguido de um operando inválido. Por exemplo, <code>trace(1+)</code> ; é sintaticamente incorreto.
Erro de sintaxe	Esta mensagem é emitida sempre que um erro de sintaxe não específico é encontrado.
Era esperado um nome de campo após o operador <code>'.'</code>	Você deve especificar um nome de campo válido ao usar a sintaxe <code>object.field</code> .
<Token> esperado	Um token inválido ou não esperado foi encontrado. Por exemplo, na sintaxe abaixo, o token <code>foo</code> não é válido. O token esperado é <code>while</code> . <pre>do {     trace (i) } foo (i &lt; 100)</pre>
A lista do inicializador deve ser terminada por <terminador>	O <code>]</code> ou <code>}</code> de fechamento está faltando na lista do inicializador de uma matriz ou objeto.

Mensagem de erro	Descrição
Identificador esperado	Um token não esperado foi encontrado no lugar de um identificador. No exemplo abaixo, 3 não é um identificador válido. <code>var 3 = 4;</code>
A construção do JavaScript 'construção' não é suportada	Uma construção do JavaScript que não é suportada pelo ActionScript foi encontrada. Esta mensagem será exibida se qualquer uma das seguintes construções do JavaScript for usada: void, switch, try, catch ou throw.
O lado esquerdo do operador de atribuição deve ser uma variável ou propriedade.	Um operador de atribuição foi usado, mas o lado esquerdo da atribuição não era uma propriedade ou uma variável legal.
O bloco de comando deve ser terminado por '}'	Um grupo de comandos foi declarado entre chaves, mas está faltando a chave de fechamento.
Evento esperado	Um manipulador On(MouseEvent) ou onClipEvent foi declarado, mas nenhum evento foi especificado, ou um token não esperado foi encontrado onde um evento deveria aparecer.
Evento inválido	O script contém um evento de mouse ou de clipe inválido. Para obter uma lista de eventos de mouse e de clipe válidos, consulte as entradas On(MouseEvent) e OnClipEvent no capítulo do dicionário do ActionScript.
Código de chave esperado	É necessário especificar um código de chave. Consulte o Apêndice B para obter uma lista de códigos de chave.
Código de chave inválido	O código de chave especificado não existe.
Encontrado rastro de lixo	O script ou a expressão foi analisado corretamente, mas continha caracteres de rastro adicionais que não puderam ser analisados.
Função ilegal	Uma declaração de função nomeada foi usada como uma expressão. As declarações de funções nomeadas devem ser comandos. Válida: <code>function sqr (x) { return x * x; }</code> Inválida: <code>var v = function sqr (x) { return x * x; }</code>
Nome de função esperado	O nome especificado para esta função não é um nome de função válido.



Mensagem de erro	Descrição
Nome de parâmetro esperado	Um nome de parâmetro (argumento) era esperado na declaração de uma função, mas um token não esperado foi encontrado.
'else' encontrado sem 'if' correspondente	Um comando <code>else</code> foi encontrado, mas nenhum <code>if</code> apareceu antes dele. Você só pode usar <code>else</code> junto com um comando <code>if</code> .
Erro do tipo de cena	O argumento de cena de uma ação <code>gotoAndPlay</code> , <code>gotoAndStop</code> ou <code>iframeLoaded</code> era do tipo errado. O argumento de cena deve ser uma constante de sequência de caracteres.
Erro interno	Ocorreu um erro interno no compilador do ActionScript. Envie o arquivo FLA que gerou esse erro para a Macromedia, com instruções detalhadas sobre como reproduzir a mensagem.
Dígitos hexadecimais esperados após 0x	A sequência 0x foi encontrada, mas não estava seguida de dígitos hexadecimais válidos.
Erro ao abrir o arquivo #include	Houve um erro ao abrir um arquivo incluído na diretiva <code>include</code> . É possível que o erro tenha ocorrido porque o arquivo estava ausente ou devido a um erro no disco.
Erro na diretiva #include	Uma diretiva <code>include</code> não foi escrita corretamente. A diretiva <code>include</code> deve usar a seguinte sintaxe: <code>#include "somefile.as"</code>
Comentário de várias linhas não terminado	Um comentário de várias linhas iniciado com <code>/*</code> não inclui a marca de fechamento <code>*/</code> .
Literal da sequência de caracteres não terminado adequadamente	Um literal de sequência de caracteres iniciado com uma aspa de abertura (simples ou dupla) não inclui a aspa de fechamento.
Função <função> exige <número> parâmetros	Uma função foi chamada, mas um número não esperado de parâmetros foi encontrado.
Esperado um nome de propriedade em GetProperty.	A função <code>getProperty</code> foi chamada, mas o segundo argumento não era o nome de uma propriedade do clipe de filme.
Não é possível declarar o parâmetro <parâmetro> várias vezes	Um nome de parâmetro apareceu várias vezes na lista de parâmetros da declaração de uma função. Todos os nomes de parâmetros devem ser exclusivos.



Mensagem de erro	Descrição
Não é possível declarar a variável «variável» várias vezes	Um nome de variável apareceu várias vezes em um comando <code>var</code> . Todos os nomes de variáveis em um comando <code>var</code> único devem ser exclusivos.
Não é possível aninhar manipuladores 'on' em outros manipuladores 'on'	Um manipulador <code>on</code> foi declarado dentro de outro manipulador <code>on</code> . Todos os manipuladores <code>on</code> devem aparecer no nível superior de uma lista de ações.
O comando deve estar dentro do manipulador <code>on</code>	Nas ações da instância de um botão, um comando foi declarado fora de um bloco <code>on</code> . Todas as ações da instância de um botão devem aparecer dentro de um bloco <code>on</code> .
O comando deve estar dentro do manipulador <code>onClipEvent</code>	Nas ações da instância de um clipe de filme, um comando foi declarado fora de um bloco <code>onClipEvent</code> . Todas as ações da instância de um clipe de filme devem aparecer dentro de um bloco <code>onClipEvent</code> .
Eventos de mouse são permitidos somente para instâncias de botões	Um manipulador de eventos de botão foi declarado em uma lista de ações de quadros ou em uma lista de ações de instâncias de clipes de filme. Os eventos de botão são permitidos somente nas listas de ações de instâncias de botões.
Eventos de clipe são permitidos somente para instâncias de clipes de filme	Um manipulador de eventos de clipe foi declarado em uma lista de ações de quadros ou em uma lista de ações de instâncias de botões. Os eventos de clipe são permitidos somente nas listas de ações de instâncias de clipes de filme.



# ÍNDICE REMISSIVO

## A

- abrindo
  - a caixa de mensagem 157
  - arquivos do Flash 4 86
- ação
  - Drag Movie Clip 129
  - duplicateMovieClip 115
  - envoltório 20
  - removeMovieClip 130
  - unloadMovie 128
- ação fscommand 139
  - comandos e argumentos 156
  - comunicando-se com o Director 157
  - usando 155
- ação getURL 140
  - comunicando-se com scripts do servidor 144
  - formulário de pesquisa 152
- ação hitTest
  - exemplo 36
- ação loadMovie 140
  - comunicando-se com scripts do servidor 144
  - níveis 110
  - verificando os dados carregados 143
- ação loadVariables 140
  - comunicando-se com scripts do servidor 144
  - verificando os dados carregados 143
- ação set variable
  - verificando os dados 153
- ação with 116
  - especificando várias Linhas de Tempo como destino 126
- acessando
  - métodos 81
  - propriedades de objetos 68
- ações 32
  - ações de quadro 47
  - ajuda relacionada ao contexto 21
  - alterando parâmetros 39
  - assíncronas 143
  - ativando simples 161
  - atribuindo a objetos 45
  - atribuindo a quadros 47
  - atribuindo para controlar filmes 127
  - básicas 91
  - com caminhos de destino 70
  - comparadas a métodos 125
  - especificando clipes de filme como destino 124
  - excluindo 39
  - exportando 43
  - imprimindo 33
  - interatividade 91
  - listadas 69
  - novos recursos 18
  - parâmetros de botão 48
  - reordenando 39
  - repetindo 72
  - selecionando 38
  - testando 45
  - trace 171
- ações de quadro
  - atribuindo 47
  - atribuindo a quadros-chave 47
  - colocação 47
  - em camadas conflitantes 161
- ActionScript
  - comparado com o JavaScript 17
  - editando com editor de texto 40
  - escrevendo scripts 24
  - Flash 4 86
  - Flash 4 comparado com o Flash 5 18
  - novos recursos 17
  - otimização 21
  - recursos do Flash 4 suportados 88
  - sintaxe 50
  - suporte ao JavaScript 18
  - terminologia 32
- adicionando
  - observações 53
- agrupando
  - comandos 51

- Ajuda do Flash
    - ações 21
  - anexando
    - clipes de filme 130
    - sons 102
  - aplicativos da Web
    - conexão contínua 149
    - integrando o Flash com 139
  - argumentos 32
    - entre parênteses 52
    - passando para funções 77
  - arquivos do Flash 4
    - abrindo 86
  - arquivos remotos
    - comunicando-se com 140
  - arrastando clipes de filme
    - avaliando 129
  - associatividade
    - operadores 63
  - ativar
    - Ações de Quadro Simples 161
    - Botões Simples 161
  - atribuindo
    - um tipo a variáveis 58
- B**
- balanço (som), controlando 104
  - barra de status, Depurador 164
  - botão
    - Inserir Caminho de Destino 121
    - Submit 152
- C**
- caixa de diálogo
    - Propriedades de Vinculação de Símbolo 130
  - caixa de mensagem, exibindo 157
  - caixas de diálogo em formulários 152
  - caminho de destino absoluto 117
  - caminho de destino relativo 117
  - caminhos de destino 117
    - definidos 34
    - especificando 70, 121
    - expressão 123
    - inserindo 39, 71
    - nomes de níveis 118
  - campos de entrada de texto
    - em formulários 151
  - campos de parâmetros 38
  - campos de pesquisa 152
  - campos de texto 139
    - rolagem 97
  - campos de texto de rolagem 97
  - capturando
    - pressionamentos de teclas 95
  - caracteres especiais 55
  - características 26
  - carregando dados
    - segurança 141
  - chamando 57
    - métodos 57
    - métodos de objetos 82
  - childNodes 145
  - classes 26
    - definidas 32
  - clipes de filme
    - alterando a visibilidade 24
    - alterando propriedades no Depurador 167
    - anexando 130
    - arrastando 129
    - atribuindo um nome de instância 70
    - compartilhando 130
    - controlando 121
    - definindo parâmetros do clipe 132
    - detectando colisões 105
    - duplicando 28, 130
    - exibição no Depurador 164
    - exibindo a hierarquia 111
    - exibindo propriedades 167
    - inserindo caminho de destino 39
    - intercâmbio 137
    - listando objetos 169
    - nomes de instância 27
    - relação hierárquica 112
    - removendo 130
    - representação gráfica 26
    - sobre 109
    - tipo de dados 57
  - Clipes Inteligentes
    - criando 131
    - definindo parâmetros do clipe 135
  - códigos de teclas
    - obtendo 95
  - colisões



- detectando 105
- entre clipes de filme 107
- entre um clipe de filme e um ponto no Palco 106
- colisões de nomes 59
- comando
  - Listar Objetos 169
  - Listar Variáveis 170
  - Mostrar Sintaxe Obsoleta 45
  - Sintaxe de Cor 44
  - Testar Filme 45, 160
- comandos
  - agrupando 51
  - definindo como expressões 161
  - ramificações lógicas 30
  - reordenando 39
  - terminando 51
- comandos condicionais 30
- comandos if 30, 71
- combinando
  - operações 67
- comentários
  - cor da sintaxe 53
  - exemplo 53
  - sintaxe 53
  - sintaxe de cor 44
- comportamentos 26
- comunicando-se
  - com o Flash Player 155
  - entre Linhas de Tempo 114
- concatenando
  - seqüências de caracteres 54
- condições
  - verificando 71
- conexão TCP/IP
  - com o objeto XMLHttpRequest 150
  - enviando informações 140
- conexões de soquete 149
  - script de exemplo 150
- conjunto de caracteres ISO-8859-1 18
- conjunto de caracteres Shift-JIS 18
- constantes
  - definidas 32
  - sintaxe 54
- contadores
  - repetindo ação com 72
- controlando
  - o som 102

- controlando clipes de filme
  - métodos 124
- controlando filmes
  - requisitos 121
- controles ActiveX 158
  - exibindo o status 164
- controles do teclado 96
- convenções de nomenclatura 160
- Core JavaScript Guide 18
- criando
  - Clipes Inteligentes 131
  - instâncias de objetos 80
  - objetos 80
  - senhas 143
- criando um loop
  - objetos filho 73
- cursores personalizados
  - criando 92

**D**

- dados carregados
  - verificando 143
- declarando
  - variáveis 60
- Depurador
  - ativando 163
  - ativando em navegador 163
  - barra de status 164
  - exibir clipes de filme 164
  - Flash Debug Player 162
  - lista Observação 166
  - propriedades do filme 167
  - senha 163
  - usando 162
  - variáveis 164
- detectando
  - colisões 105
- diferenciação de maiúsculas e minúsculas
  - palavras-chave 52
  - strings 54
- DOM XML 145
- duplicando
  - clipes de filme 130



## E

- ECMA (European Computers Manufacturers Association) 18
- editando scripts
  - externamente 42
  - modo 41
- editores externos 42
- elementos de interface
  - Clipes Inteligentes 131
  - personalizada 131
- endereços hierárquicos 34
- enviando informações
  - formato codificado URL 140
  - para arquivos remotos 140
  - via conexão de soquete TCP/IP 140
  - XMLformat 140
- erros
  - colisão de nomes 59
  - mensagens 44
  - verificando a sintaxe 44
- erros de sintaxe
  - identificando 44
  - realçando 44
  - verificando 44
- escrevendo
  - scripts 49
  - scripts em ActionScript 24
- espaços reservados 32
- especificação ECMA-262 18
  - ação tellTarget 125
- especificando como destino
  - ação duplicateMovieClip 115
- eventos
  - definidos 32
- excluindo
  - ações 39
- executando o aplicativo no projetor 156
- executando operadores
  - ordenar por associação 63
  - por precedência 63
- exemplo de filme 35
- exibindo
  - menu de contexto do Flash Player 156
- exportando
  - ações 43
- exportando para o Flash 4 88
- expressões

- atribuindo várias variáveis 66
- comparando valores 64
- definidas 32
- sobre 62

Extensible Markup Language 145

## F

filmes

- carregando adicionais 128
- controlando no Flash Player 158
- descarregando 128
- escalando para o Flash Player 156
- listando variáveis 170
- mantendo o tamanho original 156
- protegendo 141
- substituindo por um filme carregado 128
- testando no navegador 160
- transferindo informações entre 140

filmes carregados

- controlando 121
- identificando 71

Flash 5

- criando o conteúdo do Flash 4 88

Flash Debug Player 162

Flash Player

- comunicando-se com 155
- escalando filmes para 156
- exibição normal de menu 156
- exibindo a tela cheia 156
- exibindo o menu de contexto 156
- exibindo o tipo 164
- métodos 139, 158
- tornando o menu de contexto esmaecido 156
- versão de exportação 45

formato de aplicativo MIME/

- x-www-urlformencoded 144

formulários

- criando 139, 151
- elementos necessários 151
- pesquisa 152
- variáveis 153
- verificando os dados 153

função

- targetPath 123

funções

- chamando 78
- construtoras 26



- definidas 33
- definindo 76
- exemplo 32
- passando argumentos para 77
- personalizadas 76
- predefinidas 74
- regras 74
- retornando valores 78
- variáveis locais em 77
- funções atribuídas 33
- funções construtoras
  - exemplo 32
- funções de construção
  - exemplo 26
- funções personalizadas 76
- funções predefinidas 74
- listadas 74

## G

- guia
  - Propriedades 167
  - Variáveis 164

## H

- herança
  - criando 84
- hierarquia
  - clipe de filme 111
  - clipes de filme pai-filho 112
  - herança 84

## I

- identificadores
  - com valores 34
  - definidos 33
- importando
  - ActionScript 43
- imprimindo
  - ações 43
- informações
  - transferindo entre filmes 140
- inserindo
  - caminhos de destino 121
- instâncias
  - copiando 27
  - definidas 33

- interatividade
  - complexa 92
  - criando 91
  - formulários 151
- interface personalizada 131
  - clipe de filme xch 137
  - criando 136

## J

- janela Saída
  - comando Listar Objetos 169
  - comando Listar Variáveis 170
  - opções 168
  - usando 168
- Janela Script
  - alterando fonte 42
- JavaScript
  - central de desenvolvedores 18
  - comando alert 171
  - comando with 116
  - comparado com o ActionScript 17
  - editando 40
  - enviando mensagens para 156
  - linguagem suportada 18
  - padrão internacional 18

## L

- Linhas de Tempo
  - alias pai 119
  - comunicando-se entre 114
  - controlando 124
  - especificando como destino com várias ações 126
  - várias 110
- lista Ações
  - redimensionando 39
- lista Caixa de Ferramentas
  - redimensionando 39
- lista de valores de cores RRB 423
- lista de verificação, script 161
- lista Observação
  - Depurador 166
- LiveConnect 158

## M

- Macromedia Director
  - comunicando-se com 157

- manipuladores
  - definidos 33
  - verificando dados XML 143
- manipulando
  - números 55
- matrizes multidimensionais 68
- método
  - Ascii 95
  - attachMovie 124
  - attachMovieClip 130
  - attachSound 102
  - getBounds 124
  - getBytesLoaded 124
  - getBytesTotal 124
  - getCode 96
  - globalToLocal 124
  - hitTest 105
    - controlando filmes 124
  - localToGlobal 124
  - RGB 100
  - setPan 102
  - setTransform 100
  - setVolume 102
  - swapDepths 124
- método attachMovieClip
  - argumentos 130
- métodos 26, 57
  - acessando 81
  - atribuindo 127
  - chamando 125
  - comparados a ações 125
  - definidos 33
  - especificando várias Linhas de Tempo como destino 126
  - objeto 79
- métodos de objetos
  - chamando 82
- métodos do objeto XML 145
- modo
  - Normal 37
- modo de teste de filme 161
- modo Especialista 40
  - chamando uma função 74
- modo Normal
  - chamando uma função 75
- modos de edição
  - alternando 41

- preferência 41
- movendo clipes
  - criando um loop nos filhos 73
- Movie Explorer 161
  - exibição 117
- movienamename\_DoFSCCommand 156

## N

- navegação
  - controlando 91
- Netscape DevEdge Online 18
- níveis 71
  - caminho absoluto 118
  - carregando 128
  - carregando filmes em 110
  - hierarquia 111
  - nomeando no caminho de destino 118
- nome de instância xch 137
- nomeando
  - variáveis 57
- nomes 33
- nomes de instância
  - clipes de filme 27
- nomes de instâncias
  - atribuindo 70
  - definidos 33
  - definindo dinamicamente 68
- nós 145
- números 55
  - convertendo em inteiros de 32 bits 66

## O

- objeto
  - Color 100
  - Key 95
  - Sound 102
- objeto MovieClip
  - controlando filmes 124
  - sobre 27
- objeto XMLSocket
  - métodos 149
  - usando 149
  - verificando dados 143
- objetos 26
  - atribuindo ações 45
  - criando 80

- criando personalizados 83
- definidos 34
- personalizados 83
- predefinidos 79
- tipo de dados 56
- objetos MovieClip
  - usando 82
- objetos personalizados 83
- objetos predefinidos
  - listados 79
- obtendo
  - a posição do mouse 94
  - dados 139
  - informações de arquivos remotos 140
- onClipEvent(enterFrame) {
  - exemplo 36
- onClipEvent(load)
  - exemplo 36
- operador do inicializador do objeto 80
- operador new 80
- operadores
  - acesso a matriz 68
  - associatividade 63
  - atribuição 66
  - bit a bit 66
  - combinando com valores 62
  - comparação 64
  - definidos 34
  - igualdade 66
  - lógicos 65
  - numéricos 64
  - ponto 68
  - seqüência de caracteres 65
- operadores bit a bit 66
- operadores de acesso a matriz 68
- operadores de atribuição 66
  - compostos 67
- operadores de igualdade 66
- operadores de seqüência de caracteres 65
- operadores lógicos 65
- operadores numéricos 64
- operadores ponto 68
- ordem de execução 28
  - controlando 31

## P

- alias \_parent 119

- painel
  - Ações do Objeto 35
  - Parâmetros do Clipe
    - substituindo pela interface personalizada 136
- painel Ações 37
  - categorias 37
  - exibindo 37
  - modo de edição 37
  - modo Normal
    - lista Caixa de Ferramentas 37
  - opções 42
- palavras reservadas 33
  - listadas 53
  - this 36
- palavras-chave
  - definidas 33
  - diferenciação de maiúsculas e minúsculas 52
  - listadas 53
  - sintaxe de cor 44
- parâmetros
  - alterando 39
  - argumentos e 77
  - exibindo 46
  - inserindo 38
  - passando para funções 77
- parâmetros do clipe
  - atribuindo 131
  - definindo 132, 135
  - definindo Clipe Inteligente 135
- passando valores
  - por conteúdo 60
  - por referência 61
- planejando
  - scripts 25
- Plug-in Netscape 164
- portas
  - conexão XMLSocket 142
- posição do mouse
  - obtendo 94
- preferências
  - modo de edição 41
- pressionamentos de teclas
  - capturando 95
- projetores
  - executando o aplicativo 156
- propriedade
  - droptarget 129



- maxscroll 97
- prototype 84
- scroll 97
- propriedades 26
  - coleções 34
  - definidos 34
  - inalteradas 54
  - sintaxe de cor 44
- propriedades de objetos
  - acessando 81
- protocolo HTTP 140
  - comunicando-se com scripts do servidor 144
- protocolo HTTPS 140

## Q

- quadros
  - atribuindo ações a 47
- quadros-chave
  - atribuindo ações de quadro 47

## R

- ramificação lógica 30
- realçando
  - sintaxe 44
- realce de sintaxe 44
  - ativando e desativando 44
  - obsoletas 45
- referenciando
  - variáveis 59
- referências fixas 62
- referências flexíveis 62
- referências permanentes 62
- relações pai-filho 112
- removendo
  - clipes de filme 130
  - filmes carregados 128
- reordenando
  - ações 39
- repetindo
  - ações 72

## S

- salvando
  - scripts 160
- scripts
  - controlando a execução 31

- controlando o fluxo 71
- declarando variáveis 60
- depurando 162
- diretrizes 160
- escrevendo 49
- exemplo 35
- fluxo 28
- importando 43
- incluindo comentários 160
- ordem de execução 28
- planejando 25
- procurando 43
- solucionando problemas 159
- scripts CGI
  - formato padrão 144
- scripts côté serveur
  - format XML 146
- scripts do servidor
  - linguagens 140
- scripts orientados a objetos 26
- segurança 141
  - HTML padrão 142
- senhas
  - criando 143
  - Depurador 163
- seqüências de caracteres 54
  - caracteres de escape 55
  - sintaxe de cor 44
- seqüências de escape 55
- símbolos animados 57
- sintaxe
  - chaves 51
  - de barra 51
  - de ponto 50
  - diferenciação de maiúsculas e minúsculas 52
  - parênteses 52
  - ponto-e-vírgula 51
  - regras 50
- sintaxe de barra 51
  - caminhos de destino 119
- sintaxe de ponto 50
  - caminhos de destino 119
- sites remotos
  - conexão contínua 149
- solucionando problemas
  - com a ação trace 171
  - diretrizes 160



- lista de verificação 161
- listando objetos 169
- listando variáveis 170
- usando a janela Saída 168
- visão geral 159

sons

- anexando 102
- controle de balanço 104
- criando controles de volume 102

strings 54

subdomínios do URL 141

## T

terminando

- comandos 51

termos, definição 32

testando

- ações 45
- ações de quadro 48
- filmes 160
- scripts 160
- valores de variáveis 60

texto

- procurando em scripts 43

texto de entrada 97

texto dinâmico 97

this 36

- alias da Linha de Tempo atual 119

tipos de dados

- Booleano 56
- clipes de filme 57
- definidos 32
- number 55
- objetos 56
- regras 54

tipos de dados de referência 54

tipos de dados primitivos 54

- Flash 4 88

tornando o menu de contexto do

- Flash Player esmaecido 156

## U

uso de maiúsculas e minúsculas 52

## V

valores

- manipulando em expressões 62

valores ASCII 95

valores booleanos 56

- comparando 65

valores de cores

- definindo 100

variáveis

- alterando valores no Depurador 165
- atribuindo nomes significativos 161
- atribuindo um escopo 59
- atribuindo várias 66
- caminho absoluto 166
- carregando de arquivos remotos 140
- controlando valores com campos de texto 161
- convertendo em XML 146
- declarando 60
- definidas 34
- definindo dinamicamente 68
- determinando o tipo 58
- em formulários 153
- enviando para arquivos remotos 140
- modificando no Depurador 164
- nomeando 57
- ocultas 153
- passando com Clipes Inteligentes 131
- passando conteúdo 60
- referenciando um valor 61
- regras 57
- removendo da lista Observação 166
- testando 60
- usando em scripts 60
- verificando 153

variáveis globais 59

variáveis locais 59

- em funções 77
- exemplo 59

VBScript 40

verificando os dados inseridos 153

- script de exemplo 154

vinculando

- clipes de filme 130

volume

- controle deslizante 103
- controles 102

## X

XML 145

conversão de variáveis de exemplo 145  
dans les scripts côté serveur 146  
enviando informações com métodos XML 140  
enviando informações via soquete TCP/IP 140  
hierarquia 145

