

# ***Dicionário ActionScript***

---

## Marcas comerciais

Afterburner, AppletAce, Attain, Attain Enterprise Learning System, Attain Essentials, Attain Objects for Dreamweaver, Authorware, Authorware Attain, Authorware Interactive Studio, Authorware Star, Authorware Synergy, Backstage, Backstage Designer, Backstage Desktop Studio, Backstage Enterprise Studio, Backstage Internet Studio, Design in Motion, Director, Director Multimedia Studio, Doc Around the Clock, Dreamweaver, Dreamweaver Attain, Drumbeat, Drumbeat 2000, Extreme 3D, Fireworks, Flash, Fontographer, FreeHand, FreeHand Graphics Studio, Generator, Generator Developer's Studio, Generator Dynamic Graphics Server, Knowledge Objects, Knowledge Stream, Knowledge Track, Lingo, Live Effects, Macromedia, Macromedia M Logo & Design, Macromedia Flash, Macromedia Xres, Macromind, Macromind Action, MAGIC, Mediamaker, Object Authoring, Power Applets, Priority Access, Roundtrip HTML, Scriptlets, SoundEdit, ShockRave, Shockmachine, Shockwave, Shockwave Remote, Shockwave Internet Studio, Showcase, Tools to Power Your Ideas, Universal Media, Virtuoso, Web Design 101, Whirlwind e Xtra são marcas comerciais da Macromedia, Inc. e podem estar registradas nos EUA ou em outras jurisdições, inclusive internacionalmente. Outros nomes de produtos, logotipos, designs, títulos, palavras ou frases mencionados nesta publicação podem ser marcas comerciais, marcas de serviço ou nomes comerciais da Macromedia, Inc. ou de outras entidades e podem estar registrados em certas jurisdições, inclusive internacionais.

## Informações de terceiros

Tecnologia de compactação e descompactação de voz licenciada da Nellymoser, Inc. ([www.nellymoser.com](http://www.nellymoser.com)).



Tecnologia Sorenson™ Spark™ de compactação e descompactação de vídeo licenciada da Sorenson Media, Inc.

Este guia contém links para sites da Web de terceiros que não estão sob o controle da Macromedia. Neste caso, a Macromedia não é responsável pelo conteúdo de nenhum site vinculado. Se você acessar um dos sites da Web de terceiros mencionados neste guia, estará assumindo os riscos inerentes. A Macromedia oferece esses links apenas como uma conveniência, e a inclusão de um link não significa que a Macromedia apóia ou aceita qualquer responsabilidade pelo conteúdo apresentado nos sites de terceiros.

## Isenção de responsabilidade da Apple

A APPLE COMPUTER, INC. NÃO CONCEDE GARANTIA ALGUMA, NEM EXPLÍCITA NEM IMPLÍCITA, RELATIVA AO PACOTE DE SOFTWARES PARA COMPUTADORES EM ANEXO, A SUA COMERCIALIZAÇÃO OU SUA ADEQUAÇÃO A QUALQUER FINALIDADE ESPECÍFICA. A EXCLUSÃO DE GARANTIAS IMPLÍCITAS NÃO É PERMITIDA EM CERTOS ESTADOS. A EXCLUSÃO ACIMA PODE NÃO SE APLICAR NO SEU CASO. ESTA GARANTIA LHE CONCEDE DIREITOS LEGAIS ESPECÍFICOS. VOCÊ PODE TER OUTROS DIREITOS QUE VARIAM CONFORME O ESTADO.

Copyright © 2000 Macromedia, Inc. Todos os direitos reservados. Este manual não pode ser copiado, fotocopiado, reproduzido, traduzido ou convertido em nenhum formato eletrônico ou que possa ser lido por máquina, por inteiro ou em parte, sem o consentimento prévio por escrito da Macromedia, Inc.

## Agradecimentos

Direção: Erick Vera

Produção: Wayne Wieseler

Redação: Jody Bleyle, JuLee Burdekin, Mary Burger, Dale Crawford, Marcelle Taylor

Design instrucional: Stephanie Gowin, Barbara Nelson

Edição: Rosana Francescato, Lisa Stanziano, Anne Szabla

Design e produção de multimídia: Aaron Begley, Benjamin Salles, Noah Zilberberg

Design e produção de impressão: Chris Basmajian, Caroline Branch

Localização: Michael Dominguez, Cristina Guembe, Yoshika Hedberg, Tim Hussey, Masayo "Noppe" Noda, Simone Pux, Yoko Shindo, Yuko Yagi

Primeira edição: março de 2002

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

# ÍNDICE

Exemplo de entrada para a maioria dos elementos do ActionScript . . . . .	21
Exemplo de entrada para objetos e componentes . . . . .	22
Conteúdo do dicionário . . . . .	23
— (decremento) . . . . .	38
++ (incremento) . . . . .	39
! (NOT lógico) . . . . .	40
!= (diferença) . . . . .	41
!== (diferença estrita) . . . . .	41
% (módulo) . . . . .	42
%= (Atribuição de módulo) . . . . .	43
& (AND bit a bit) . . . . .	43
&& (AND de curto-circuito) . . . . .	44
&= (atribuição AND bit a bit) . . . . .	44
() (parênteses) . . . . .	45
– (subtração) . . . . .	46
* (multiplicação) . . . . .	47
*= (atribuição de multiplicação) . . . . .	47
, (vírgula) . . . . .	48
. (ponto) . . . . .	49
?: (condicional) . . . . .	50
/ (divisão) . . . . .	50
// (delimitador de comentário) . . . . .	51
/* (delimitador de comentário) . . . . .	51
/= (atribuição de divisão) . . . . .	52
[] (acesso de matriz) . . . . .	53
^(XOR bit a bit) . . . . .	55
^= (atribuição XOR bit a bit) . . . . .	55
{ } (inicializador de objeto) . . . . .	56
(OR bit a bit) . . . . .	57
(OR lógico) . . . . .	58
= (atribuição OR bit a bit) . . . . .	59
~ (NOT bit a bit) . . . . .	59
+ (adição) . . . . .	60
+= (atribuição de adição) . . . . .	61
< (menor que) . . . . .	62
<< (deslocamento para a esquerda bit a bit) . . . . .	63
<<= (deslocamento para a esquerda bit a bit e atribuição) . . . . .	64
<= (menor ou igual a) . . . . .	64

<> (diferença) . . . . .	66
= (atribuição) . . . . .	66
-= (atribuição de subtração) . . . . .	67
== (igualdade) . . . . .	68
=== (igualdade estrita) . . . . .	69
> (maior que) . . . . .	70
>= (maior ou igual a) . . . . .	70
>> (deslocamento para a direita bit a bit) . . . . .	71
>>= (deslocamento para a direita bit a bit e atribuição) . . . . .	72
>>> (deslocamento para a direita não assinado bit a bit) . . . . .	73
>>>= (deslocamento para a direita não assinado bit a bit e atribuição) . . . . .	73
Accessibility (objeto) . . . . .	74
Accessibility.isActive . . . . .	74
add . . . . .	75
and . . . . .	75
arguments (objeto) . . . . .	76
arguments.callee . . . . .	76
arguments.caller . . . . .	77
arguments.length . . . . .	77
Array (objeto) . . . . .	77
Array.concat . . . . .	79
Array.join . . . . .	80
Array.length . . . . .	81
Array.pop . . . . .	81
Array.push . . . . .	82
Array.reverse . . . . .	82
Array.shift . . . . .	83
Array.slice . . . . .	83
Array.sort . . . . .	84
Array.sortOn . . . . .	85
Array.splice . . . . .	86
Array.toString . . . . .	86
Array.unshift . . . . .	87
asfunction . . . . .	87
Boolean (função) . . . . .	88
Boolean (objeto) . . . . .	89
Boolean.toString . . . . .	90
Boolean.valueOf . . . . .	90
break . . . . .	90
Button (objeto) . . . . .	91
Resumo de eventos do objeto Button . . . . .	92
Button._alpha . . . . .	93
Button.enabled . . . . .	93
Button._focusrect . . . . .	93
Button.getDepth . . . . .	93
Button._height . . . . .	94
Button._highquality . . . . .	94
Button._name . . . . .	94
Button.onDragOut . . . . .	95
Button.onDragOver . . . . .	95

Button.onKeyDown	96
Button.onKeyUp	96
Button.onKillFocus	97
Button.onPress	97
Button.onRelease	98
Button.onReleaseOutside	98
Button.onRollOut	99
Button.onRollOver	99
Button.onSetFocus	100
Button._parent	100
Button._quality	100
Button._rotation	101
Button._soundbuftime	101
Button.tabEnabled	102
Button.tabIndex	102
Button._target	103
Button.trackAsMenu	103
Button._url	103
Button.useHandCursor	103
Button._visible	104
Button._width	104
Button._x	105
Button._xmouse	105
Button._xscale	105
Button._y	106
Button._ymouse	106
Button._yscale	106
call	107
chamar função	107
case	107
chr	108
clearInterval	108
Color (objeto)	109
Color.getRGB	110
Color.getTransform	110
Color.setRGB	111
Color.setTransform	111
continue	113
CustomActions (objeto)	113
CustomActions.get	114
CustomActions.install	114
CustomActions.list	115
CustomActions.uninstall	115
Date (objeto)	115
Date.getDate	118
Date.getDay	119
Date.getFullYear	119
Date.getHours	120
Date.getMilliseconds	120
Date.getMinutes	120

Date.getMonth . . . . .	121
Date.getSeconds . . . . .	121
Date.getTime . . . . .	121
Date.getTimezoneOffset . . . . .	122
Date.getUTCDate . . . . .	122
Date.getUTCDay . . . . .	123
Date.getUTCFullYear . . . . .	123
Date.getUTCHours . . . . .	123
Date.getUTCMilliseconds . . . . .	124
Date.getUTCMinutes . . . . .	124
Date.getUTCMonth . . . . .	124
Date.getUTCSeconds . . . . .	125
Date.getYear . . . . .	125
Date.setDate . . . . .	125
Date.setFullYear . . . . .	126
Date.setHours . . . . .	126
Date.setMilliseconds . . . . .	127
Date.setMinutes . . . . .	127
Date.setMonth . . . . .	127
Date.setSeconds . . . . .	128
Date.setTime . . . . .	128
Date.setUTCDate . . . . .	128
Date.setUTCFullYear . . . . .	129
Date.setUTCHours . . . . .	129
Date.setUTCMilliseconds . . . . .	130
Date.setUTCMinutes . . . . .	130
Date.setUTCMonth . . . . .	130
Date.setUTCSeconds . . . . .	131
Date.setYear . . . . .	131
Date.toString . . . . .	132
Date.UTC . . . . .	132
default . . . . .	133
delete . . . . .	134
do while . . . . .	135
duplicateMovieClip . . . . .	135
else . . . . .	136
else if . . . . .	137
#endinitclip . . . . .	138
eq (igual — específico de sequência de caracteres) . . . . .	138
escape . . . . .	139
eval . . . . .	139
evaluate . . . . .	140
false . . . . .	140
FCheckBox (componente) . . . . .	141
FCheckBox.setEnabled . . . . .	142
FCheckBox.getLabel . . . . .	142
FCheckBox.getValue . . . . .	143
FCheckBox.registerSkinElement . . . . .	143
FCheckBox.setChangeHandler . . . . .	144
FCheckBox.setEnabled . . . . .	145

FCheckBox.setLabel . . . . .	146
FCheckBox.setLabelPlacement . . . . .	146
FCheckBox.setSize . . . . .	147
FCheckBox.setStyleProperty . . . . .	147
FCheckBox.setValue . . . . .	148
FComboBox (component) . . . . .	148
FComboBox.addItem . . . . .	150
FComboBox.addItemAt . . . . .	151
FComboBox.setEnabled . . . . .	152
FComboBox.getItemAt . . . . .	152
FComboBox.getLength . . . . .	153
FComboBox.getRowCount . . . . .	153
FComboBox.getScrollPosition . . . . .	154
FComboBox.getSelectedIndex . . . . .	154
FComboBox.getSelectedItem . . . . .	155
FComboBox.getValue . . . . .	155
FComboBox.registerSkinElement . . . . .	156
FComboBox.removeAll . . . . .	157
FComboBox.removeItemAt . . . . .	157
FComboBox.replaceItemAt . . . . .	158
FComboBox.setChangeHandler . . . . .	158
FComboBox.setDataProvider . . . . .	159
FComboBox.setEditable . . . . .	161
FComboBox.setEnabled . . . . .	161
FComboBox.setItemSymbol . . . . .	162
FComboBox.setRowCount . . . . .	162
FComboBox.setSelectedIndex . . . . .	163
FComboBox.setSize . . . . .	163
FComboBox.setStyleProperty . . . . .	164
FComboBox.setValue . . . . .	164
FComboBox.sortItemsBy . . . . .	165
FListBox (component) . . . . .	166
FListBox.addItem . . . . .	167
FListBox.addItemAt . . . . .	168
FListBox.setEnabled . . . . .	169
FListBox.getItemAt . . . . .	169
FListBox.getLength . . . . .	170
FListBox.getRowCount . . . . .	170
FListBox.getScrollPosition . . . . .	171
FListBox.getSelectedIndex . . . . .	171
FListBox.getSelectedIndices . . . . .	172
FListBox.getSelectedItem . . . . .	172
FListBox.getSelectedItems . . . . .	173
FListBox.selectMultiple . . . . .	174
FListBox.getValue . . . . .	174
FListBox.registerSkinElement . . . . .	175
FListBox.removeAll . . . . .	176
FListBox.removeItemAt . . . . .	176
FListBox.replaceItemAt . . . . .	177
FListBox.setAutoHideScrollBar . . . . .	177

FListBox.setChangeHandler	178
FListBox.setDataProvider	179
FListBox.setEnabled	180
FListBox.setItemSymbol	181
FListBox.setRowCount	181
FListBox.setScrollPosition	182
FListBox.setSelectedIndex	183
FListBox.setSelectedIndices	183
FListBox.setSelectMultiple	184
FListBox.setSize	184
FListBox.setStyleProperty	185
FListBox.setWidth	185
FListBox.sortItemsBy	186
_focusrect	187
for	187
for..in	188
FPushButton (component)	189
FPushButton.setEnabled	190
FPushButton.getLabel	191
FPushButton.registerSkinElement	191
FPushButton.setClickHandler	192
FPushButton.setEnabled	193
FPushButton.setLabel	194
FPushButton.setSize	194
FPushButton.setStyleProperty	195
FRadioButton (component)	195
FRadioButton.getData	196
FRadioButton.setEnabled	197
FRadioButton.getLabel	197
FRadioButton.getState	198
FRadioButton.getValue	198
FRadioButton.registerSkinElement	199
FRadioButton.setChangeHandler	200
FRadioButton.setData	201
FRadioButton.setEnabled	202
FRadioButton.setGroupName	202
FRadioButton.setLabel	203
FRadioButton.setLabelPlacement	204
FRadioButton.setSize	204
FRadioButton.setState	205
FRadioButton.setStyleProperty	206
FRadioButton.setValue	206
FScrollBar (component)	207
FScrollBar.setEnabled	208
FScrollBar.getScrollPosition	209
FScrollBar.registerSkinElement	209
FScrollBar.setChangeHandler	210
FScrollBar.setEnabled	212
FScrollBar.setHorizontal	212
FScrollBar.setLargeScroll	213



FScrollBar.setScrollContent . . . . .	213
FScrollBar.setScrollPosition . . . . .	214
FScrollBar.setScrollProperties . . . . .	215
FScrollBar.setScrollTarget . . . . .	215
FScrollBar.setSize . . . . .	216
FScrollBar.setSmallScroll . . . . .	216
FScrollBar.setStyleProperty . . . . .	217
FScrollPane (component) . . . . .	218
FScrollPane.getPaneHeight . . . . .	219
FScrollPane.getPaneWidth . . . . .	219
FScrollPane.getScrollContent . . . . .	220
FScrollPane.getScrollPosition . . . . .	220
FScrollPane.loadScrollContent . . . . .	221
FScrollPane.refreshPane . . . . .	222
FScrollPane.registerSkinElement . . . . .	222
FScrollPane.setDragContent . . . . .	223
FScrollPane.setHScroll . . . . .	224
FScrollPane.setScrollContent . . . . .	224
FScrollPane.setScrollPosition . . . . .	225
FScrollPane.setSize . . . . .	225
FScrollPane.setStyleProperty . . . . .	226
FScrollPane.setVScroll . . . . .	227
FStyleFormat (object) . . . . .	227
Resumo das propriedades do objeto FStyleFormat . . . . .	228
FStyleFormat.addListener . . . . .	230
FStyleFormat.applyChanges . . . . .	230
FStyleFormat.arrow . . . . .	231
FStyleFormat.background . . . . .	232
FStyleFormat.backgroundDisabled . . . . .	232
FStyleFormat.check . . . . .	233
FStyleFormat.darkshadow . . . . .	233
FStyleFormat.face . . . . .	234
FStyleFormat.foregroundDisabled . . . . .	234
FStyleFormat.highlight . . . . .	235
FStyleFormat.highlight3D . . . . .	235
FStyleFormat.radioDot . . . . .	236
FStyleFormat.removeListener . . . . .	236
FStyleFormat.scrollTrack . . . . .	237
FStyleFormat.selection . . . . .	237
FStyleFormat.selectionDisabled . . . . .	238
FStyleFormat.selectionUnfocused . . . . .	238
FStyleFormat.shadow . . . . .	239
FStyleFormat.textAlign . . . . .	239
FStyleFormat.textBold . . . . .	240
FStyleFormat.textColor . . . . .	240
FStyleFormat.textDisabled . . . . .	241
FStyleFormat.textFont . . . . .	241
FStyleFormat.textIndent . . . . .	241
FStyleFormat.textItalic . . . . .	242
FStyleFormat.textLeftMargin . . . . .	242

FStyleFormat.textRightMargin . . . . .	243
FStyleFormat.textSelected . . . . .	243
FStyleFormat.textSize . . . . .	244
FStyleFormat.textUnderline . . . . .	244
Function (objeto) . . . . .	245
Function.apply . . . . .	245
Function.call . . . . .	246
Function.prototype . . . . .	247
fscommand . . . . .	247
function . . . . .	249
ge (maior ou igual a — específico de seqüências de caracteres). . . . .	251
getProperty . . . . .	251
getTimer . . . . .	252
getURL . . . . .	252
getVersion . . . . .	253
_global . . . . .	253
globalStyleFormat . . . . .	254
gotoAndPlay . . . . .	255
gotoAndStop . . . . .	255
gt (maior que — específico de seqüências de caracteres) . . . . .	256
_highquality . . . . .	256
if . . . . .	256
ifFrameLoaded . . . . .	258
#include . . . . .	258
#initclip . . . . .	259
instanceof . . . . .	260
int . . . . .	261
isFinite . . . . .	261
isNaN . . . . .	262
Key (objeto) . . . . .	262
Key.addListener . . . . .	264
Key.BACKSPACE . . . . .	264
Key.CAPSLOCK . . . . .	264
Key.CONTROL . . . . .	265
Key.DELETEKEY . . . . .	265
Key.DOWN . . . . .	265
Key.END . . . . .	265
Key.ENTER . . . . .	265
Key.ESCAPE . . . . .	266
Key.getAscii . . . . .	266
Key.getCode . . . . .	266
Key.HOME . . . . .	267
Key.INSERT . . . . .	267
Key.isDown . . . . .	267
Key.isToggled . . . . .	267
Key.LEFT . . . . .	268
Key.onKeyDown . . . . .	268
Key.onKeyUp . . . . .	268
Key.PGDN . . . . .	269
Key.PGUP . . . . .	269

Key.removeListener . . . . .	269
Key.RIGHT . . . . .	270
Key.SHIFT . . . . .	270
Key.SPACE . . . . .	270
Key.TAB . . . . .	270
Key.UP . . . . .	270
le (menor que ou igual a — específico da sequência de caracteres) . . . . .	271
length . . . . .	271
_level. . . . .	272
loadMovie . . . . .	272
loadMovieNum . . . . .	274
loadVariables . . . . .	275
loadVariablesNum . . . . .	276
LoadVars (objeto) . . . . .	277
LoadVars.contentType . . . . .	279
LoadVars.getBytesLoaded . . . . .	279
LoadVars.getBytesTotal . . . . .	279
LoadVars.load . . . . .	280
LoadVars.loaded . . . . .	280
LoadVars.onLoad . . . . .	281
LoadVars.send . . . . .	281
LoadVars.sendAndLoad . . . . .	282
LoadVars.toString . . . . .	282
lt (menor que — sequência de caracteres específica) . . . . .	283
Math (objeto) . . . . .	283
Math.abs . . . . .	285
Math.acos . . . . .	285
Math.asin . . . . .	285
Math.atan . . . . .	286
Math.atan2 . . . . .	286
Math.ceil . . . . .	286
Math.cos . . . . .	287
Math.E . . . . .	287
Math.exp . . . . .	288
Math.floor . . . . .	288
Math.log . . . . .	288
Math.LOG2E . . . . .	289
Math.LOG10E . . . . .	289
Math.LN2 . . . . .	290
Math.LN10 . . . . .	290
Math.max . . . . .	290
Math.min . . . . .	291
Math.PI . . . . .	291
Math.pow . . . . .	292
Math.random . . . . .	292
Math.round . . . . .	293
Math.sin . . . . .	293
Math.sqrt . . . . .	293
Math.SQRT1_2 . . . . .	294
Math.SQRT2 . . . . .	294

Math.tan . . . . .	295
maxscroll . . . . .	295
mbchr . . . . .	295
mblength . . . . .	296
mbord . . . . .	296
mbsubstring . . . . .	297
método . . . . .	297
Mouse (objeto) . . . . .	298
Mouse.addListener . . . . .	298
Mouse.hide . . . . .	299
Mouse.onMouseDown . . . . .	299
Mouse.onMouseMove . . . . .	300
Mouse.onMouseUp . . . . .	300
Mouse.removeListener . . . . .	301
Mouse.show . . . . .	301
MovieClip (objeto) . . . . .	301
MovieClip._alpha . . . . .	305
MovieClip.attachMovie . . . . .	305
MovieClip.beginFill . . . . .	306
MovieClip.beginGradientFill . . . . .	307
MovieClip.clear . . . . .	311
MovieClip.createEmptyMovieClip . . . . .	311
MovieClip.createTextField . . . . .	312
MovieClip._currentframe . . . . .	313
MovieClip.curveTo . . . . .	314
MovieClip._droptarget . . . . .	315
MovieClip.duplicateMovieClip . . . . .	315
MovieClip.enabled . . . . .	316
MovieClip.endFill . . . . .	316
MovieClip.focusEnabled . . . . .	317
MovieClip._focusrect . . . . .	317
MovieClip._framesloaded . . . . .	317
MovieClip.getBounds . . . . .	318
MovieClip.getBytesLoaded . . . . .	319
MovieClip.getBytesTotal . . . . .	319
MovieClip.getDepth . . . . .	319
MovieClip.getURL . . . . .	320
MovieClip.globalToLocal . . . . .	320
MovieClip.gotoAndPlay . . . . .	321
MovieClip.gotoAndStop . . . . .	321
MovieClip._height . . . . .	322
MovieClip._highquality . . . . .	322
MovieClip.hitArea . . . . .	323
MovieClip.hitTest . . . . .	323
MovieClip.lineStyle . . . . .	324
MovieClip.lineTo . . . . .	325
MovieClip.loadMovie . . . . .	326
MovieClip.loadVariables . . . . .	327
MovieClip.localToGlobal . . . . .	327
MovieClip.moveTo . . . . .	328

MovieClip._name	329
MovieClip.nextFrame	329
MovieClip.onData	329
MovieClip.onDragOut	330
MovieClip.onDragOver	330
MovieClip.onEnterFrame	331
MovieClip.onKeyDown	331
MovieClip.onKeyUp	332
MovieClip.onKillFocus	332
MovieClip.onLoad	333
MovieClip.onMouseDown	333
MovieClip.onMouseMove	334
MovieClip.onMouseUp	334
MovieClip.onPress	335
MovieClip.onRelease	335
MovieClip.onReleaseOutside	336
MovieClip.onRollOut	336
MovieClip.onRollOver	337
MovieClip.onSetFocus	337
MovieClip.onUnload	338
MovieClip._parent	338
MovieClip.play	339
MovieClip.prevFrame	339
MovieClip.removeMovieClip	339
MovieClip._rotation	340
MovieClip.setMask	340
MovieClip._soundbuftime	341
MovieClip.startDrag	341
MovieClip.stop	342
MovieClip.stopDrag	342
MovieClip.swapDepths	342
MovieClip.tabChildren	343
MovieClip.tabEnabled	344
MovieClip.tabIndex	344
MovieClip._target	345
MovieClip._totalframes	345
MovieClip.trackAsMenu	345
MovieClip.unloadMovie	345
MovieClip._url	346
MovieClip.useHandCursor	346
MovieClip._visible	347
MovieClip._width	347
MovieClip._x	347
MovieClip._xmouse	348
MovieClip._xscale	348
MovieClip._y	348
MovieClip._ymouse	349
MovieClip._yscale	349
NaN	349
ne (diferente — específico de sequência de caracteres)	350

new . . . . .	350
newline . . . . .	351
nextFrame . . . . .	351
nextScene . . . . .	352
not . . . . .	352
null . . . . .	353
Number (função). . . . .	353
Number (objeto). . . . .	354
Number.MAX_VALUE. . . . .	356
Number.MIN_VALUE . . . . .	356
Number.NaN . . . . .	356
Number.NEGATIVE_INFINITY. . . . .	356
Number.POSITIVE_INFINITY. . . . .	357
Number.toString . . . . .	357
Number.valueOf . . . . .	357
Object (objeto) . . . . .	358
Object.addProperty . . . . .	359
Object.__proto__ . . . . .	360
Object.registerClass . . . . .	361
Object.toString . . . . .	363
Object.unwatch . . . . .	363
Object.valueOf . . . . .	364
Object.watch . . . . .	364
onClipEvent . . . . .	366
on . . . . .	367
or . . . . .	368
ord . . . . .	368
_parent . . . . .	369
parseFloat . . . . .	369
parseInt . . . . .	370
play . . . . .	371
prevFrame . . . . .	372
prevScene . . . . .	372
print . . . . .	373
printAsBitmap. . . . .	374
printAsBitmapNum. . . . .	375
printNum . . . . .	376
_quality . . . . .	377
random . . . . .	378
removeMovieClip . . . . .	378
return . . . . .	379
_root . . . . .	379
scroll . . . . .	380
Selection (objeto) . . . . .	380
Selection.addListener. . . . .	381
Selection.getBeginIndex . . . . .	382
Selection.getCaretIndex. . . . .	382
Selection.getEndIndex. . . . .	382
Selection.getFocus . . . . .	383
Selection.onSetFocus . . . . .	383

Selection.removeListener . . . . .	384
Selection.setFocus . . . . .	384
Selection.setSelection . . . . .	385
set variable . . . . .	385
setInterval . . . . .	386
setProperty . . . . .	388
Sound (objeto) . . . . .	388
Sound.attachSound . . . . .	390
Sound.duration . . . . .	390
Sound.getBytesLoaded . . . . .	391
Sound.getBytesTotal . . . . .	391
Sound.getPan . . . . .	391
Sound.getTransform . . . . .	392
Sound.getVolume . . . . .	392
Sound.loadSound . . . . .	393
Sound.onLoad . . . . .	393
Sound.onSoundComplete . . . . .	394
Sound.position . . . . .	394
Sound.setPan . . . . .	395
Sound.setTransform . . . . .	395
Sound.setVolume . . . . .	397
Sound.start . . . . .	398
Sound.stop . . . . .	398
_soundbuftime . . . . .	399
Stage (objeto) . . . . .	399
Stage.addListener . . . . .	400
Stage.align . . . . .	400
Stage.height . . . . .	401
Stage.onResize . . . . .	401
Stage.removeListener . . . . .	402
Stage.scaleMode . . . . .	402
Stage.width . . . . .	402
startDrag . . . . .	403
stop . . . . .	403
stopAllSounds . . . . .	404
stopDrag . . . . .	404
String (função) . . . . .	405
" " (delimitador de sequência de caracteres) . . . . .	406
String (objeto) . . . . .	406
String.charAt . . . . .	408
String.charCodeAt . . . . .	408
String.concat . . . . .	409
String.fromCharCode . . . . .	409
String.indexOf . . . . .	410
String.lastIndexOf . . . . .	410
String.length . . . . .	411
String.slice . . . . .	411
String.split . . . . .	412
String.substr . . . . .	413
String.substring . . . . .	413

String.toLowerCase . . . . .	414
String.toUpperCase . . . . .	414
substring . . . . .	414
super . . . . .	415
switch . . . . .	416
System (objeto) . . . . .	417
System.capabilities (objeto) . . . . .	417
System.capabilities.hasAudioEncoder . . . . .	418
System.capabilities.hasAccessibility . . . . .	419
System.capabilities.hasAudio . . . . .	419
System.capabilities.hasMP3 . . . . .	419
System.capabilities.language . . . . .	419
System.capabilities.manufacturer . . . . .	420
System.capabilities.os . . . . .	421
System.capabilities.pixelAspectRatio . . . . .	421
System.capabilities.screenColor . . . . .	421
System.capabilities.screenDPI . . . . .	421
System.capabilities.screenResolution.x . . . . .	422
System.capabilities.screenResolution.y . . . . .	422
System.capabilities.version . . . . .	422
System.capabilities.hasVideoEncoder . . . . .	422
targetPath . . . . .	423
tellTarget . . . . .	423
TextField (objeto) . . . . .	424
TextField._alpha . . . . .	427
TextField.addListener . . . . .	427
TextField.autoSize . . . . .	428
TextField.background . . . . .	428
TextField.backgroundColor . . . . .	429
TextField.border . . . . .	429
TextField.borderColor . . . . .	429
TextField.bottomScroll . . . . .	430
TextField.embedFonts . . . . .	430
TextField._focusrect . . . . .	430
TextField.getDepth . . . . .	430
TextField.getFontList . . . . .	431
TextField.getNewTextFormat . . . . .	431
TextField.getTextFormat . . . . .	432
TextField._height . . . . .	432
TextField._highquality . . . . .	433
TextField.hscroll . . . . .	433
TextField.html . . . . .	434
TextField.htmlText . . . . .	434
TextField.length . . . . .	434
TextField.maxChars . . . . .	435
TextField.maxhscroll . . . . .	435
TextField.maxscroll . . . . .	435
TextField.multiline . . . . .	435
TextField._name . . . . .	436
TextField.onChanged . . . . .	436



TextField.onKillFocus . . . . .	436
TextField.onScroller. . . . .	437
TextField.onSetFocus. . . . .	437
TextField._parent . . . . .	437
TextField.password . . . . .	438
TextField._quality . . . . .	438
TextField.removeListener. . . . .	439
TextField.removeTextField. . . . .	439
TextField.replaceSel. . . . .	439
TextField.restrict . . . . .	440
TextField._rotation . . . . .	441
TextField.scroll . . . . .	441
TextField.selectable . . . . .	441
TextField.setNewTextFormat. . . . .	442
TextField.setTextFormat . . . . .	442
TextField._soundbuftime. . . . .	443
TextField.tabEnabled. . . . .	444
TextField.tabIndex. . . . .	444
TextField._target . . . . .	445
TextField.text . . . . .	445
TextField.textColor . . . . .	445
TextField.textHeight . . . . .	445
TextField.textWidth . . . . .	446
TextField.type . . . . .	446
TextField._url . . . . .	446
TextField.variable . . . . .	446
TextField._visible. . . . .	447
TextField._width . . . . .	447
TextField.wordWrap . . . . .	447
TextField._x. . . . .	448
TextField._xmouse. . . . .	448
TextField._xscale . . . . .	448
TextField._y. . . . .	449
TextField._ymouse . . . . .	449
TextField._yscale . . . . .	449
TextFormat (objeto) . . . . .	450
TextFormat.align . . . . .	452
TextFormat.blockIndent . . . . .	452
TextFormat.bold . . . . .	452
TextFormat.bullet . . . . .	452
TextFormat.color . . . . .	453
TextFormat.font . . . . .	453
TextFormat.getTextExtent . . . . .	453
TextFormat.indent. . . . .	454
TextFormat.italic . . . . .	454
TextFormat.leading . . . . .	454
TextFormat.leftMargin . . . . .	454
TextFormat.rightMargin . . . . .	455
TextFormat.size . . . . .	455
TextFormat.tabStops . . . . .	455

TextFormat.target	455
TextFormat.underline	456
TextFormat.url	456
this	456
toggleHighQuality	457
trace	458
true	459
typeof	459
undefined	460
unescape	461
unloadMovie	461
unloadMovieNum	462
updateAfterEvent	462
var	463
void	463
while	464
with	465
XML (objeto)	467
XML.appendChild	469
XML.attributes	470
XML.childNodes	470
XML.cloneNode	471
XML.contentType	471
XML.createElement	472
XML.createTextNode	472
XML.docTypeDecl	473
XML.firstChild	473
XML.getBytesLoaded	474
XML.getBytesTotal	474
XML.hasChildNodes	474
XML.ignoreWhite	475
XML.insertBefore	475
XML.lastChild	476
XML.load	476
XML.loaded	477
XML.nextSibling	477
XML.nodeName	477
XML.nodeType	478
XML.nodeValue	478
XML.onData	478
XML.onLoad	479
XML.parentNode	480
XML.parseXML	480
XML.previousSibling	481
XML.removeNode	481
XML.send	481
XML.sendAndLoad	482
XML.status	482
XML.toString	483
XML.xmlDecl	483

XMLSocket (objeto) . . . . .	484
XMLSocket.close. . . . .	486
XMLSocket.connect . . . . .	486
XMLSocket.onClose . . . . .	487
XMLSocket.onConnect. . . . .	488
XMLSocket.onData . . . . .	489
XMLSocket.onXML . . . . .	489
XMLSocket.send. . . . .	490



# Dicionário ActionScript

Este dicionário descreve a sintaxe e o uso de elementos do ActionScript no Macromedia Flash MX. Para usar os exemplos em um script, copie o código de exemplo 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, componentes e métodos. Para obter uma visão geral de todas as entradas do dicionário, consulte “Conteúdo do dicionário”, na página 23; as tabelas desta seção representam um bom começo para verificar os operadores simbólicos e métodos cuja classe de objeto ou componente seja desconhecida.

O ActionScript segue o padrão ECMA-262 (a especificação escrita pela Associação Européia de Fabricantes de Computadores), salvo indicação em contrário. Alguns elementos do ActionScript do Flash 5 (e anteriores) ficaram obsoletos e foram substituídos por novos elementos do ActionScript que correspondem ao padrão ECMA. Recomenda-se usar os novos elementos do Flash MX, embora o Flash Player 5 ainda ofereça suporte para alguns elementos obsoletos.

Há dois tipos de entradas neste dicionário:

- Entradas individuais para operadores, palavras-chave, funções, variáveis, propriedades, métodos e comandos;
- Entradas de objeto e de componente oferecem informações gerais sobre os objetos internos e componentes do Flash

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 ou componentes.

### Título da entrada

Todas as entradas são listadas em ordem alfabética. A ordem ignora maiúsculas e minúsculas, sublinhados no início e assim por diante.

### Disponibilidade

Esta seção informa quais as versões do Flash Player que oferecem suporte ao elemento. Isso não é o mesmo que a versão do Flash usada para criar o conteúdo. Por exemplo, se a ferramenta de criação Flash MX for usada para criar um conteúdo para o Flash Player 5, use apenas os elementos do ActionScript disponíveis para o Flash Player 5.

### **Uso**

Esta seção fornece a sintaxe correta para usar o elemento do ActionScript em seu código. A parte necessária da sintaxe está em *fonte de código* e o código fornecido pelo usuário está em *fonte de código em itálico*. Os colchetes ([]) indicam parâmetros opcionais.

### **Parâmetros**

Esta seção descreve qualquer parâmetro listado na sintaxe.

### **Retorna**

Se houver valores, esta seção identifica qual será retornado pelo elemento.

### **Descrição**

Esta seção identifica o tipo de elemento (por exemplo, um operador, um método, uma função etc.) e, em seguida, descreve como usar o elemento.

### **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 e componentes**

O exemplo de entrada do dicionário a seguir explica as convenções usadas para objetos e componentes ActionScript internos. Os objetos e componentes são listados em ordem alfabética com todos os outros elementos do dicionário. Os componentes do Flash são listados como FCheckBox, FComboBox e assim por diante.

### **Título da entrada**

O título da entrada fornece o nome do objeto ou do componente, que é seguido de um parágrafo com informações descritivas gerais.

### **Tabelas de resumo de método e propriedade**

Cada entrada de objeto e de componente contém uma tabela com todos os métodos associados. Se o objeto ou componente tiver propriedades (normalmente constantes), esses elementos serão resumidos em uma tabela complementar. Todos os métodos e propriedades listados nessas tabelas também têm suas próprias entradas do dicionário, que seguem a entrada do objeto e do componente.

### **Construtor**

Se um objeto ou componente necessitar do uso de um construtor para acessar métodos e propriedades, o construtor será descrito em cada entrada do objeto ou do componente. Essa descrição tem todos os elementos padrões (sintaxe, descrição etc.) das outras entradas do dicionário.

### **Listagens de métodos e propriedades**

Os métodos e as propriedades de um objeto ou componente são listados em ordem alfabética depois da entrada do objeto ou do componente.

## Conteúdo do dicionário

Todas as entradas do dicionário são listadas em ordem alfabética. Contudo, alguns operadores são símbolos e são apresentados na ordem ASCII. Além disso, os métodos associados a um objeto ou componente são listados junto com o nome do objeto ou componente—por exemplo, o método `abs` do objeto `Math` é listado como `Math.abs` e o método `getValue` do componente `FRadioButton` é listado como `FRadioButton.getValue`.

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 precedência e associatividade de operadores, consulte o apêndice A, “Associatividade e precedência de operadores”, no manual “Usando o Flash”.

Operadores simbólicos	
--	-- (decremento)
++	++ (incremento)
!	! (NOT lógico)
!=	!= (diferença)
!==	!== (diferença estrita)
%	% (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)
[]	[] (acesso de matriz)
^	^ (XOR bit a bit)
^=	^= (atribuição XOR bit a bit)
{}	{ } (inicializador de objeto)
	(OR bit a bit)

---

**Operadores simbólicos**

---

	(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 subtração)
==	== (igualdade)
===	=== (igualdade estrita)
>	> (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 <code>ActionScript</code>	Consulte a entrada
<code>abs</code>	<code>Math.abs</code>
<code>acos</code>	<code>Math.acos</code>
<code>add</code>	<code>add</code>
<code>addItem</code>	<code>FComboBox.addItem</code> , <code>FListBox.addItem</code>
<code>addItemAt</code>	<code>FComboBox.addItemAt</code> , <code>FListBox.addItem</code>
<code>addListener</code>	<code>FStyleFormat.addListener</code> <code>FStyleFormat.addListener</code> , <code>Key.addListener</code> , <code>Mouse.addListener</code> , <code>Selection.addListener</code> , <code>Stage.addListener</code> <code>Stage.addListener</code> , <code>TextField.addListener</code>
<code>addProperty</code>	<code>Object.addProperty</code>
<code>and</code>	<code>and</code>
<code>align</code>	<code>Stage.align</code> , <code>TextFormat.align</code>
<code>_alpha</code>	<code>MovieClip._alpha</code> , <code>Button._alpha</code> , <code>TextField._alpha</code>
<code>appendChild</code>	<code>XML.appendChild</code>
<code>apply</code>	<code>Function.apply</code>
<code>applyChanges</code>	<code>FStyleFormat.applyChanges</code> <code>FStyleFormat.applyChanges</code>



<b>Elemento do ActionScript</b>	<b>Consulte a entrada</b>
Argumentos	arguments (objeto)
Array	Array (objeto)
arrow	FStyleFormat.arrow
asfunction	asfunction
asin	Math.asin
atan	Math.atan
atan2	Math.atan2
attachMovie	MovieClip.attachMovie
attachSound	Sound.attachSound
attributes	XML.attributes
autosize	TextField.autoSize
background	FStyleFormat.background, TextField.background
backgroundColor	TextField.backgroundColor
backgroundDisabled	FStyleFormat.backgroundDisabled
BACKSPACE	Key.BACKSPACE
beginFill	MovieClip.beginFill
beginGradientFill	MovieClip.beginGradientFill
blockIndent	TextFormat.blockIndent
bold	TextFormat.bold
Booleano	Boolean (função), Boolean (objeto)
border	TextField.border
borderColor	TextField.borderColor
bottomScroll	TextField.bottomScroll
break	break
bullet	TextFormat.bullet
Button	Button (objeto)
call	call, Function.call
call function	chamar função
callee	arguments.callee
caller	arguments.caller
capabilities	System.capabilities (objeto)
CAPSLock	Key.CAPSLock
case	case
ceil	Math.ceil
charAt	String.charAt
charCodeAt	String.charCodeAt
check	FStyleFormat.check
childNodes	XML.childNodes
chr	chr
clear	MovieClip.clear
clearInterval	clearInterval

Elemento do ActionScript	Consulte a entrada
cloneNode	XML.cloneNode
close	XMLSocket.close
Cor	Color (objeto), TextFormat.color
concat	Array.concat, String.concat
connect	XMLSocket.connect
constructor	Array (objeto), Boolean (objeto), Color (objeto), Date (objeto), Number (função), Object (objeto), Sound (objeto), String (objeto), XML (objeto), XMLSocket (objeto)
contentType	LoadVars.contentType, XML.contentType
ccntinue	continue
CONTROL	Key.CONTROL
cos	Math.cos
createElement	XML.createElement
createEmptyMovieClip	MovieClip.createEmptyMovieClip
createTextField	MovieClip.createTextField
createTextNode	XML.createTextNode
_currentframe	MovieClip._currentframe
curveTo	MovieClip.curveTo
Date	Date (objeto)
darkshadow	FStyleFormat.darkshadow
default	defaultdefault
delete	delete
DELETEKEY	Key.DELETEKEY
docTypeDecl	XML.docTypeDecl
do while	do while
DOWN	Key.DOWN
_droptarget	MovieClip._droptarget
duplicateMovieClip	duplicateMovieClip, MovieClip.duplicateMovieClip
duration	Sound.duration
E	Math.E
#endinitclip	#endinitclip
else	else
else if	else if
embedFonts	TextField.embedFonts
enabled	Button.enabled, MovieClip.enabled
END	Key.END
endFill	MovieClip.endFill
ENTER	Key.ENTER
eq	eq (igual – específico de seqüência de caracteres)
escape (função)	escape
ESCAPE (constante)	Key.ESCAPE
eval	eval

<b>Elemento do ActionScript</b>	<b>Consulte a entrada</b>
evaluate	evaluate
exp	Math.exp
face	FStyleFormat.face
false	false
FCheckBox	FCheckBox (componente)
FComboBox	FListBox (component)
firstChild	XML.firstChild
FListBox	FListBox (component)
floor	Math.floor
focusEnabled	MovieClip.focusEnabled
_focusrect	_focusrect, Button._focusrect, TextField._focusrect, MovieClip._focusrect
fonte	TextFormat.font
for	for
for..in	for..in
foregroundDisabled	FStyleFormat.foregroundDisabled
FPushButton	FPushButton (component)
FRadioButton	FPushButton (component)
_framesloaded	MovieClip._framesloaded
fromCharCode	String.fromCharCode
fscommand	fscommand
FScrollBar	FScrollBar (component)
FScrollPane	FScrollPane (component)
FStyleFormat	FStyleFormat (object)
function	function, Function (objeto)
ge	ge (maior ou igual a – específico de seqüências de caracteres)
get	CustomActions.get
getAscii	Key.getAscii
getBeginIndex	Selection.getBeginIndex
getBounds	MovieClip.getBounds
getBytesLoaded	LoadVars.getBytesLoaded, MovieClip.getBytesLoaded, Sound.getBytesLoaded, XML.getBytesLoaded
getBytesTotal	LoadVars.getBytesTotal, MovieClip.getBytesTotal, Sound.getBytesTotal, XML.getBytesTotal
getCaretIndex	Selection.getCaretIndex
getCode	Key.getCode
getData	FRadioButton.getData
getDate	Date.getDate
getDay	Date.getDay
getDepth	Button.getDepth, MovieClip.getDepth, TextField.getDepth
getEnabled	FCheckBox.getEnabled, FComboBox.getEnabled, FListBox.getEnabled, FPushButton.getEnabled, FRadioButton.getEnabled, FScrollBar.getEnabled

<b>Elemento do ActionScript</b>	<b>Consulte a entrada</b>
getEndIndex	Selection.getEndIndex
getFocus	Selection.getFocus
getFontList	TextField.getFontList
getFullYear	Date.getFullYear
getHours	Date.getHours
getItemAt	FComboBox.getItemAt, FListBox.addItemAt
getLabel	FCheckBox.getLabel, FPushButton.getLabel, FRadioButton.getLabel
getLength	FComboBox.getLength, FListBox.getLength
getMilliseconds	Date.getMilliseconds
getMinutes	Date.getMinutes
getMonth	Date.getMonth
getNewTextFormat	TextField.getNewTextFormat
getPan	Sound.getPan
getPaneHeight	FScrollPane.getPaneHeight
getPaneWidth	FScrollPane.getPaneWidth
getProperty	getProperty
getRowCount	FComboBox.getRowCount, FListBox.getRowCount
getRGB	Color.getRGB
getScrollContent	FScrollPane.getScrollContent
getScrollPosition	FComboBox.getScrollPosition, FListBox.getScrollPosition, FScrollBar.getScrollPosition, FScrollPane.getScrollPosition
getSeconds	Date.getSeconds
getSelectedIndex	FComboBox.getSelectedIndex, FListBox.getSelectedIndex
getSelectedIndices	FListBox.getSelectedIndices
getSelectedItem	FComboBox.getSelectedItem, FListBox.getSelectedItem
getSelectedItems	FListBox.getSelectedItem
getSelectMultiple	FListBox.getSelectMultiple
getState	FRadioButton.getState
getTextExtent	TextFormat.getTextExtent
getTextFormat	TextField.getTextFormat
getTime	Date.getTime
getTimer	getTimer
getTimezoneOffset	Date.getTimezoneOffset
getTransform	Color.getTransform, Sound.getTransform
getURL	getURL, MovieClip.getURL
getUTCDate	Date.getUTCDate
getUTCDay	Date.getUTCDay
getUTCFullYear	Date.getUTCFullYear
getUTCHours	Date.getUTCHours
getUTCMilliseconds	Date.getUTCMilliseconds
getUTCMinutes	Date.getUTCMinutes

Elemento do ActionScript	Consulte a entrada
getUTCMonth	Date.getUTCMonth
getUTCSeconds	Date.getUTCSeconds
getValue	FCheckBox.getValue, FComboBox.getValue, FListBox.getValue, FRadioButton.getValue
getVersion	getVersion
getVolume	Sound.getVolume
getYear	Date.getYear
_global	_global
globalStyleFormat	globalStyleFormat
globalToLocal	MovieClip.globalToLocal
goto	gotoAndPlay, gotoAndStop
gotoAndPlay	gotoAndPlay, MovieClip.gotoAndPlay
gotoAndStop	gotoAndStop, MovieClip.gotoAndStop
gt	gt (maior que – específico de seqüências de caracteres)
hasAccessibility	System.capabilities.hasAccessibility
hasAudio	System.capabilities.hasAudio
hasAudioEncoder	System.capabilities.hasAudioEncoder
hasMP3	System.capabilities.hasMP3
hasVideoEncoder	System.capabilities.hasVideoEncoder
hasChildNodes	XML.hasChildNodes
height	Stage.height
_height	MovieClip._height, TextField._height, Button._height
hide	Mouse.hide
highlight	FStyleFormat.highlight
highlight3D	FStyleFormat.highlight3D
_highquality	_highquality, Button._highquality, MovieClip._highquality, TextField._highquality
hitArea	MovieClip.hitArea
hitTest	MovieClip.hitTest
HOME	Key.HOME
hscroll	TextField.hscroll
html	TextField.html
htmlText	TextField.htmlText
if	if
ifFrameLoaded	ifFrameLoaded
ignoreWhite	XML.ignoreWhite
#include	#include
indent	TextFormat.indent
indexOf	String.indexOf
#initclip	#initclip
INSERT	Key.INSERT
insertBefore	XML.insertBefore

<b>Elemento do ActionScript</b>	<b>Consulte a entrada</b>
install	CustomActions.install
instanceof	instanceof
int	int
isActive	Accessibility.isActive
isDown	Key.isDown
isFinite	isFinite
isNaN	isNaN
isToggled	Key.isToggled
italic	TextFormat.italic
join	Array.join
Key	Key (objeto)
language	System.capabilities.language
lastChild	XML.lastChild
lastIndexOf	String.lastIndexOf
le	le (menor que ou igual a — específico da sequência de caracteres)
leading	TextFormat.leading
LEFT	Key.LEFT
leftMargin	TextFormat.leftMargin
length	arguments.length, Array.length, String.length, Sound.loadSound, TextField.length
level	_level
lineStyle	MovieClip.lineStyle
lineTo	MovieClip.lineTo
list	CustomActions.uninstall
LN2	Math.LN2
LN10	Math.LN10
load	XML.load, LoadVars.load
loaded	XML.loaded, LoadVars.loaded
loadMovie	loadMovie, MovieClip.loadMovie
loadMovieNum	loadMovieNum
loadScrollContent	FScrollPane.loadScrollContent
loadSound	Sound.loadSound
loadVariables	loadVariables, MovieClip.loadVariables
loadVariablesNum	loadVariablesNum
LoadVars	LoadVars (objeto)
localToGlobal	MovieClip.localToGlobal
log	Math.log
LOG2E	Math.LOG2E
LOG10E	Math.LOG10E
lt	lt (menor que — sequência de caracteres específica)
manufacturer	System.capabilities.manufacturer

Elemento do ActionScript	Consulte a entrada
Math	Math (objeto)
max	Math.max
maxChars	TextField.maxChars
maxhscroll	TextField.maxhscroll
maxscroll	maxscroll, TextField.maxscroll
MAX_VALUE	Number.MAX_VALUE
mbchr	mbchr
mblength	mblength
mbord	mbord
mbsubstring	mbsubstring
method	método
min	Math.min
MIN_VALUE	Number.MIN_VALUE
Mouse	Mouse (objeto)
moveTo	MovieClip.moveTo
MovieClip	MovieClip (objeto)
multiline	TextField.multiline
_name	MovieClip._name, TextField._name, Button._name
NaN	NaN, Number.NaN
ne	ne (diferente – específico de sequência de caracteres)
NEGATIVE_INFINITY	Number.NEGATIVE_INFINITY
new (operador)	new
newline	newline
nextFrame	nextFrame, MovieClip.nextFrame
nextScene	nextScene
nextSibling	XML.nextSibling
nodeName	XML.nodeName
nodeType	XML.nodeType
nodeValue	XML.nodeValue
not	not
null	null
Number	Number (função), Number (objeto)
Object	Object (objeto)
on	on
onClipEvent	onClipEvent
onClose	XMLSocket.onClose
onChanged	TextField.onChanged
onConnect	XMLSocket.onConnect
onData	XML.onData, XMLSocket.onData
onDragOut	Button.onDragOut, MovieClip.onDragOut
onDragOver	Button.onDragOver, MovieClip.onDragOver

<b>Elemento do ActionScript</b>	<b>Consulte a entrada</b>
onEnterFrame	MovieClip.onEnterFrame
onKeyDown	Button.onKeyDown, Key.onKeyDown, MovieClip.onKeyDown
onKeyUp	Button.onKeyUp, Key.onKeyUp, MovieClip.onKeyUp
onKillFocus	Button.onKillFocus, MovieClip.onKillFocus, TextField.onKillFocus
onLoad	LoadVars.onLoad, MovieClip.onLoad, Sound.onLoad, XML.onLoad
onMouseDown	Mouse.onMouseDown, MovieClip.onMouseDown
onMouseMove	Mouse.onMouseMove, MovieClip.onMouseMove
onMouseUp	Mouse.onMouseUp, MovieClip.onMouseUp
onPress	Button.onPress, MovieClip.onPress
onRelease	Button.onRelease, MovieClip.onRelease
onReleaseOutside	Button.onReleaseOutside, MovieClip.onReleaseOutside
onResize	Stage.onResize
onRollOut	Button.onRollOut
onRollOver	Button.onRollOver
onScroller	TextField.onScroller
onSetFocus	Button.onSetFocus, MovieClip.onSetFocus, Selection.onSetFocus, TextField.onSetFocus
onSort	Array.pop
onSoundComplete	Sound.onSoundComplete
onUnload	MovieClip.onUnload
onXML	XMLSocket.onXML
or (logical OR)	or
ord	ord
os	System.capabilities.os
_parent	_parent, Button._parent, MovieClip._parent, TextField._parent
parentNode	XML.parentNode
parseFloat	parseFloat
parseInt	parseInt
parseXML	XML.parseXML
password	TextField.password
PGDN	Key.PGDN
PGUP	Key.PGUP
PI	Math.PI
pixelAspectRatio	System.capabilities.pixelAspectRatio
play	play, MovieClip.play
pop	Array.pop
position	Sound.position
POSITIVE_INFINITY	Number.POSITIVE_INFINITY
pow	Math.pow
prevFrame	prevFrame, MovieClip.prevFrame



Elemento do ActionScript	Consulte a entrada
previousSibling	XML.previousSibling
prevScene	prevScene
print	print
printAsBitmap	printAsBitmap
printAsBitmapNum	printAsBitmapNum
printNum	printNum
__proto__	Object.__proto__
push	Array.push
_quality	_quality, TextField._quality, Button._quality
radioDot	FStyleFormat.radioDot
random	random, Math.random
refreshPane	FScrollPane.refreshPane
registerClass	Object.registerClass
registerSkinElement	FCheckBox.registerSkinElement, FComboBox.registerSkinElement, FListBox.registerSkinElement, FPushButton.registerSkinElement, FRadioButton.registerSkinElement, FScrollPane.registerSkinElement, FScrollPane.registerSkinElement
removeAll	FComboBox.removeAll, FListBox.removeAll
removeItemAt	FComboBox.removeItemAt, FListBox.removeItemAt
removeListener	FStyleFormat.removeListener, Key.removeListener, Mouse.removeListener, Selection.removeListener, Stage.removeListener, TextField.removeListener
removeMovieClip	removeMovieClip, MovieClip.removeMovieClip
removeNode	XML.removeNode
removeTextField	TextField.removeTextField
replaceItemAt	FComboBox.replaceItemAt, FListBox.replaceItemAt
replaceSel	TextField.replaceSel
resolutionX	Capabilities.screenResolutionX
resolutionY	Capabilities.screenResolutionY
restrict	TextField.restrict
return	return
reverse	Array.reverse
RIGHT	Key.RIGHT
rightMargin	TextFormat.rightMargin
_root	_root
_rotation	MovieClip._rotation, Button._rotation, TextField._rotation
round	Math.round
scaleMode	Stage.scaleMode
screenColor	System.capabilities.screenColor
screenDPI	System.capabilities.screenDPI
screenResolution.x	System.capabilities.screenResolution.x
screenResolution.y	System.capabilities.screenResolution.y

Elemento do ActionScript	Consulte a entrada
scroll	scroll, TextField.scroll
scrollTrack	FStyleFormat.scrollTrack
selectable	TextField.selectable
selection	FStyleFormat.selection
Selection	Selection (objeto)
selectionDisabled	FStyleFormat.selectionDisabled
selectionUnfocused	FStyleFormat.selectionUnfocused
send	LoadVars.send, XML.send, XMLSocket.send
sendAndLoad	LoadVars.sendAndLoad, XML.sendAndLoad
set variable	set variable
setAutoHideScrollBar	FListBox.setAutoHideScrollBar
setChangeHandler	FCheckBox.setChangeHandler, FComboBox.setChangeHandler, FListBox.setChangeHandler, FRadioButton.setChangeHandler, FScrollBar.setChangeHandler
setClickHandler	FPushButton.setClickHandler
setData	FRadioButton.setData
setDataProvider	FComboBox.setDataProvider, FListBox.setDataProvider
setDate	Date.setDate
setDragContent	FScrollPane.setDragContent
setEditable	FComboBox.setEditable
setEnabled	FCheckBox.setEnabled, FComboBox.setEnabled, FListBox.setEnabled, FPushButton.setEnabled, FRadioButton.setEnabled, FScrollBar.setEnabled
setFocus	Selection.setFocus
setFullYear	Date.setFullYear
setGroupName	FRadioButton.setGroupName
setHorizontal	FScrollBar.setHorizontal
setHours	Date.setHours
setHScroll	FScrollPane.setHScroll
setInterval	setInterval
setItemSymbol	FComboBox.setItemSymbol, FListBox.setItemSymbol
setLabel	FCheckBox.setLabel, FPushButton.setLabel, FRadioButton.setLabel
setLabelPlacement	FCheckBox.setLabelPlacement, FRadioButton.setLabelPlacement
setLargeScroll	FScrollBar.setLargeScroll
setMask	MovieClip.setMask
setMilliseconds	Date.setMilliseconds
setMinutes	Date.setMinutes
setMonth	Date.setMonth
setNewTextFormat	TextField.setNewTextFormat
setPan	Sound.setPan
setProperty	setProperty
setRGB	Color.setRGB

Elemento do ActionScript	Consulte a entrada
setRowCount	FComboBox.setRowCount, FListBox.setRowCount
setScrollContent	FScrollBar.setSize, FScrollPane.setScrollContent
setScrollPosition	FListBox.setScrollPosition, FScrollBar.setScrollPosition, FScrollPane.setScrollPosition
setScrollProperties	FScrollBar.setScrollProperties
setScrollTarget	FScrollBar.setScrollTarget
setSeconds	Date.setSeconds
setSelectedIndex	FComboBox.setSelectedIndex, FListBox.setSelectedIndex
setSelectedIndices	FListBox.setSelectedIndices
setSelection	Selection.setSelection
setSelectMultiple	FListBox.setSelectMultiple
setSize	FCheckBox.setSize, FComboBox.setSize, FListBox.setSize, FPushButton.setSize, FScrollBar.setSize, FScrollPane.setSize
setSmallScroll	FScrollBar.setSmallScroll
setState	FRadioButton.setState
setStyleProperty	FCheckBox.setStyleProperty, FComboBox.setStyleProperty, FListBox.setStyleProperty, FPushButton.setStyleProperty, FRadioButton.setStyleProperty, FScrollBar.setStyleProperty, FScrollPane.setStyleProperty
setTextFormat	TextField.setTextFormat
setTime	Date.setTime
setTransform	Color.setTransform, Sound.setTransform
setUTCDate	Date.setUTCDate
setUTCFullYear	Date.setUTCFullYear
setUTCHours	Date.setUTCHours
setUTCMilliseconds	Date.setUTCMilliseconds
setUTCMinutes	Date.setUTCMinutes
setUTCMonth	Date.setUTCMonth
setUTCSeconds	Date.setUTCSeconds
setValue	FCheckBox.setValue, FComboBox.setValue, FRadioButton.setValue
setVolume	Sound.setVolume
setVScroll	FScrollPane.setVScroll
setWidth	FListBox.setWidth
setYear	Date.setYear
shadow	FStyleFormat.shadow
shift (method)	Array.shift
SHIFT (constant)	Key.SHIFT
show	Mouse.show
sin	Math.sin
size	TextFormat.size
slice	Array.slice, String.slice
sort	Array.sort
sortItemsBy	FComboBox.sortItemsBy, FListBox.sortItemsBy

<b>Elemento do ActionScript</b>	<b>Consulte a entrada</b>
Sound	Sound (objeto)
_soundbuftime	_soundbuftime, TextField._soundbuftime, MovieClip._soundbuftime, Button._soundbuftime
SPACE	Key.SPACE
splice	Array.splice
split	String.split
sqrt	Math.sqrt
SQRT1_2	Math.SQRT1_2
SQRT2	Math.SQRT2
start	Sound.start
startDrag	startDrag, MovieClip.startDrag
status	XML.status
stop	stop, MovieClip.stop, Sound.stop
stopAllSounds	stopAllSounds
stopDrag	stopDrag, MovieClip.stopDrag
String	String (função), String (objeto)
substr	String.substring
substring	substring, String.substring
super	super
swapDepths	MovieClip.swapDepths
switch	switch
System	System (objeto)
TAB	Key.TAB
tabChildren	MovieClip.tabChildren
tabEnabled	Button.tabEnabled, TextField.tabEnabled, MovieClip.tabEnabled
tabIndex	Button.tabIndex, MovieClip.tabIndex, TextField.tabIndex
tabStops	TextFormat.tabStops
tan	Math.tan
target	TextFormat.target
_target	Button._target, MovieClip._target, TextField._target
targetPath	targetPath
tellTarget	tellTarget
text	TextField.text
textAlign	FStyleFormat.textAlign
textBold	FStyleFormat.textBold
textColor	FStyleFormat.textColor, TextField.textColor
textDisabled	FStyleFormat.textDisabled
TextField	TextField (objeto)
textFont	FStyleFormat.textFont
TextFormat	TextFormat (objeto)
textHeight	TextField.textHeight

Elemento do ActionScript	Consulte a entrada
textIndent	FStyleFormat.textIndent
textItalic	FStyleFormat.textItalic
textLeftMargin	FStyleFormat.textLeftMargin
textRightMargin	FStyleFormat.textRightMargin
textSelected	FStyleFormat.textSelected
textSize	FStyleFormat.textSize
textUnderline	FStyleFormat.textUnderline
textWidth	TextField.textWidth
this	this
toggleHighQuality	toggleHighQuality
toLowerCase	String.toLowerCase
toString	Array.toString, Boolean.toString, Date.toString, Number.toString, Object.toString, XML.toString
_totalframes	MovieClip._totalframes
toUpperCase	String.toUpperCase
trace	trace
trackAsMenu	Button.trackAsMenu, MovieClip.trackAsMenu
true	true
type	TextField.type
typeof	typeof
undefined	undefined
underline	TextFormat.underline
unescape	unescape
uninstall	CustomActions.uninstall
unloadMovie	unloadMovie, MovieClip.unloadMovie
unloadMovieNum	unloadMovieNum
unshift	Array.unshift
unwatch	Object.unwatch
UP	Key.UP
updateAfterEvent	updateAfterEvent
url	TextFormat.url
_url	MovieClip._url, TextField._url, Button._url
useHandCursor	Button.useHandCursor, MovieClip.useHandCursor
UTC	Date.UTC
valueOf	Boolean.valueOf, Number.valueOf, Object.valueOf
var	var
variable	TextField.variable
version	System.capabilities.version
_visible	MovieClip._visible, Button._visible, TextField._visible
void	void
watch	Object.watch
while	while

Elemento do ActionScript	Consulte a entrada
width	Stage.width
_width	MovieClip._width, TextField._width, Button._width
with	with
wordwrap	TextField.wordWrap
_x	Button._x, MovieClip._x, TextField._x
XML	XML (objeto)
xmlDecl	XML.xmlDecl
XMLSocket	XMLSocket (objeto)
_xmouse	Button._xmouse, MovieClip._xmouse, TextField._xmouse
_xscale	Button._xscale, MovieClip._xscale, TextField._xscale
_y	Button._y, MovieClip._y, TextField._y
_ymouse	Button._ymouse, MovieClip._ymouse, TextField._ymouse
_yscale	Button._yscale, MovieClip._yscale, TextField._yscale

## — (decremento)

### Disponibilidade

Flash Player 4.

### Uso

--*expressão*

*expressão*--

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Operador (aritmético); um operador unitário pré-decremento e 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).

### Exemplo

A forma pré-decremento do operador decrementa x para 2 ( $x - 1 = 2$ ) e retorna o resultado como y:

```
x = 3;
y = --x;
//y é igual a 2
```

A forma pós-decremento do operador decrementa x para 2 ( $x - 1 = 2$ ) e retorna o valor original de x como o resultado y:

```
x = 3;
y = x--;
//y é igual a 3
```

## ++ (incremento)

### Disponibilidade

Flash Player 4.

### Uso

`++expressão`

`expressão++`

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Operador (aritmético); um operador unário pré-incremento e pós-incremento que adiciona 1 à *expressão*. A *expressão* pode ser uma variável, um elemento em uma matriz ou uma propriedade de um objeto. 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
//y é igual a 2
```

A forma pós-incremento do operador incrementa *x* para 2 ( $x + 1 = 2$ ) e retorna o valor original de *x* como o resultado *y*:

```
x = 1;
y = x++;
//y é igual a 1
```

### Exemplo

O exemplo a seguir usa ++ como operador pós-incremento para fazer com que um loop `while` seja executado cinco vezes.

```
i = 0;
while(i++ < 5){
  trace("isto é execução " + i);
}
```

Este exemplo usa ++ como operador pré-incremento:

```
var a = [];
var i = 0;
while (i < 10) {
  a.push(++i);
}
trace(a.join());
```

Este script exibe o seguinte resultado na janela Saída:

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

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 exibe o seguinte resultado na janela Saída:

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

## ! (NOT lógico)

### Disponibilidade

Flash Player 4.

### Uso

*!expressão*

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Operador (lógico); inverte o valor booleano de uma variável ou expressão. Se *expressão* for uma variável com o valor absoluto ou convertido `true`, o valor de *!expressão* será `false`. Se a expressão `x && y` for avaliada como `false`, a expressão *!(x && y)* será avaliada como `true`.

As expressões a seguir ilustram o resultado do uso do operador !:

*! true* retorna `false`

*! false* retorna `true`

### 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`, a ação `trace` enviará uma sequência de caracteres para a janela Saída.

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



## != (diferença)

### Disponibilidade

Flash Player 5.

### Uso

*expressão1* != *expressão2*

### Parâmetros

Nenhum.

### Retorna

Nada.

### 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.

### Exemplo

O exemplo a seguir ilustra o resultado do operador !=:

5 != 8 retorna `true`

5 != 5 retorna `false`

Este exemplo ilustra o uso do operador != em um comando `if`.

```
a = "David";  
b = "Fool"  
if (a != b){  
    trace("David is not a fool");  
}
```

### Consulte também

!= (diferença estrita), == (igualdade), === (igualdade estrita)

## !== (diferença estrita)

### Disponibilidade

Flash Player 6.

### Uso

*expressão1* !== *expressão2*

### Descrição

Operador; testa o oposto exato do operador ===. O operador diferença estrita executa a mesma operação que o operador diferença, exceto a conversão dos tipos de dados. Se *expressão1* for igual a *expressão2* e os tipos de dados forem iguais, o resultado será `false`. Da mesma forma que 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 comparados por referência.

### Exemplo

O código a seguir exibe o valor retornado de operações que usam os operadores de igualdade, igualdade estrita e diferença estrita.

```
s1 = new String("5");
s2 = new String("5");
s3 = new String("Hello");
n  = new Number(5);
b  = new Boolean(true);
```

```
s1 == s2; // true
s1 == s3; // false
s1 == n;  // true
s1 == b;  // false
```

```
s1 === s2; // true
s1 === s3; // false
s1 === n;  // false
s1 === b;  // false
```

```
s1 !== s2; // false
s1 !== s3; // true
s1 !== n;  // true
s1 !== b;  // true
```

### Consulte também

!= (diferença), == (igualdade), === (igualdade estrita)

## % (módulo)

### Disponibilidade

Flash Player 4. Nos arquivos do Flash 4, o operador % é expandido no arquivo SWF como `x - int(x/y) * y` e pode não ser tão rápido ou preciso quanto nas versões posteriores do Flash Player.

### Uso

*expressão1 % expressão2*

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Operador (aritmético); calcula o resto da *expressão1* dividida por *expressão2*. Se um dos parâmetros *expressão* não for numérico, o operador módulo tentará convertê-lo(s) em números. A *expressão* pode ser um número ou uma sequência de caracteres convertida em um valor numérico.

### Exemplo

A seguir é apresentado um exemplo numérico que usa o operador módulo (%).

```
trace (12 % 5);
// retorna 2
trace (4,3 % 2,1);
// retorna aproximadamente 0,1
```

## %= (Atribuição de módulo)

### Disponibilidade

Flash Player 4.

### Uso

*expressão1* %= *expressão2*

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Operador (atribuição composta aritmética); atribui a *expressão1* o valor de *expressão1* % *expressão2*. Por exemplo, as duas expressões a seguir são idênticas:

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

### Exemplo

O exemplo a seguir atribui o valor 4 à variável *x*.

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

### Consulte também

% (módulo)

## & (AND bit a bit)

### Disponibilidade

Flash Player 5. 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 + são usados para concatenar 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.

### Uso

*expressão1* & *expressão2*

### Parâmetros

Nenhum.

### Retorna

Nada.

### 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 parâmetros inteiros. O resultado é um novo inteiro não assinado de 32 bits.

## && (AND de curto-circuito)

### Disponibilidade

Flash Player 4.

### Uso

*expressão1* && *expressão2*

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Operador (lógico); executa uma operação booleana nos valores de uma ou de ambas as expressões. Avalia *expressão1* (a expressão do lado esquerdo do operador) e retorna *false* se a expressão for avaliada como *false*. Se *expressão1* for avaliada como *true*, *expressão2* (a expressão do lado direito do operador) será avaliada. Se *expressão2* for avaliada como *true*, o resultado final será *true*; caso contrário, será *false*.

### Exemplo

Este exemplo usa o operador && para realizar um teste e determinar se um jogador venceu o jogo. As variáveis *turns* e *score* são atualizadas quando for a vez de um jogador ou quando esse marcar um ponto durante o jogo. O script “Você venceu o jogo!” será exibido na janela Saída quando a pontuação do jogador atingir 75, ou mais, em 3 voltas, ou menos.

```
turns=2;
score=77;
winner = (turns <= 3) && (score >= 75);
if (winner) {
    trace("Você venceu o jogo!");
} else {
    trace("Tente novamente!");
}
```

## &= (atribuição AND bit a bit)

### Disponibilidade

Flash Player 5.

### Uso

*expressão1* &= *expressão2*

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Operador (atribuição composta bit a bit); atribui a *expressão1* o valor da *expressão1* & *expressão2*. Por exemplo, as duas expressões a seguir são idênticas.

```
x &= y  
x = x & y
```

### Exemplo

O exemplo a seguir atribui o valor 9 a x.

```
x = 15;  
y = 9;  
trace(x &= y);  
// retorna 9
```

### Consulte também

& (AND bit a bit)

## () (parênteses)

### Disponibilidade

Flash Player 4.

### Uso

```
(expressão1, expressão2);  
função(parâmetro1, ..., parâmetroN);
```

### Parâmetros

*expressão1*, *expressão2* Números, seqüências de caracteres, variáveis ou texto.

*função* A função a ser executada no conteúdo entre parênteses.

*parâmetro1...parâmetroN* Uma série de parâmetros que devem ser executados antes de os resultados serem transferidos como parâmetros para a função fora dos parênteses.

### Retorna

Nada.

### Descrição

Operador; executa uma operação de agrupamento em um ou mais parâmetros, ou envolve um ou mais parâmetros e os passa como parâmetros para uma função fora dos parênteses.

Uso 1: Controla a ordem de execução dos operadores na expressão. Os parênteses substituem a ordem de precedência normal e fazem com que as expressões neles inseridas sejam avaliadas em primeiro lugar. Quando os parênteses estão aninhados, o conteúdo dos parênteses mais internos é avaliado antes do conteúdo dos mais externos.

Uso 2: Envolve um ou mais parâmetros e os passa como parâmetros para a função fora dos parênteses.

### Exemplo

Uso 1: Os comandos a seguir ilustram o uso de parênteses para controlar a ordem de execução das expressões. O valor de cada expressão é exibido abaixo de cada linha da seguinte maneira:

```
trace((2 + 3) * (4 + 5));  
// é exibido 45
```

```
trace(2 + (3 * (4 + 5)));  
// é exibido 29
```

```
trace(2 + (3 * 4) + 5);  
// é exibido 19
```

Uso 2: Os exemplos a seguir ilustram o uso de parênteses com funções.

```
getDate();  
  
invoice(item, amount);  
  
function traceParameter(param){  
    trace(param);  
}  
traceParameter(2*2);
```

### Consulte também

with

## - (subtração)

### Disponibilidade

Flash Player 4.

### Uso

(Negação)  $-expressão$

(Subtração)  $expressão1 - expressão2$

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Operador (aritmético); usado para negação ou subtração.

Uso 1: Quando usado para negação, reverte o sinal da *expressão* numérica.

Uso 2: Quando usado para subtração, executa uma subtração aritmética em duas expressões numéricas, subtraindo *expressão2* de *expressão1*. Quando ambas as expressões são inteiras, a diferença é um inteiro. Quando uma ou ambas as expressões são números de ponto flutuante, a diferença é um número de ponto flutuante.

### Exemplo

Uso 1: O comando a seguir reverte o sinal da expressão  $2 + 3$ .

```
-(2 + 3)
```

O resultado é -5.

Uso 2: O comando a seguir subtrai o inteiro 2 do inteiro 5.

$5 - 2$

O resultado é 3, que é um inteiro.

Uso 2: O comando a seguir subtrai o número de ponto flutuante 1,5 do número de ponto flutuante 3,25.

$3,25 - 1,5$

O resultado é 1,75, que é um número de ponto flutuante.

## **\* (multiplicação)**

### **Disponibilidade**

Flash Player 4.

### **Uso**

*expressão1 \* expressão2*

### **Parâmetros**

Nenhum.

### **Retorna**

Nada.

### **Descrição**

Operador (aritmético); multiplica duas expressões numéricas. Se ambas as expressões forem inteiras, o produto será um inteiro. Se uma ou ambas as expressões forem números de ponto flutuante, o produto será um número de ponto flutuante.

### **Exemplo**

O comando a seguir 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)**

### **Disponibilidade**

Flash Player 4.

### **Uso**

*expressão1 \*= expressão2*

### **Parâmetros**

Nenhum.

**Retorna**

Nada.

**Descrição**

Operador (atribuição composta aritmética); atribui a *expressão1* o valor da *expressão1* \* *expressão2*. Por exemplo, as duas expressões a seguir são idênticas:

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

**Exemplo**

O exemplo a seguir atribui o valor 50 à variável *x*.

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

**Exemplo**

A segunda e a terceira linhas do exemplo a seguir calculam as expressões do lado direito do sinal de igual e atribuem os resultados a *x* e *y*.

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

**Consulte também**

\* (multiplicação)

## , (vírgula)

**Disponibilidade**

Flash Player 4.

**Uso**

*expressão1*, *expressão2*

**Parâmetros**

Nenhum.

**Retorna**

Nada.

**Descrição**

Operador; avalia *expressão1*, depois *expressão2* e retorna o valor de *expressão2*. Esse operador é principalmente usado com o comando de loop *for*.

**Exemplo**

O exemplo de código a seguir usa um operador vírgula:

```
var a=1, b=2, c=3;
```

Isso é equivalente a escrever o seguinte código:

```
var a=1;
var b=2;
var c=3;
```



## . (ponto).

### Disponibilidade

Flash Player 4.

### Uso

*objeto.propriedade\_ou\_metodo*

*nome\_da\_instancia.variavel*

*nome\_da\_instancia.instancia\_filha.variavel*

### Parâmetros

*objeto* Uma instância de um objeto. O objeto pode ser qualquer objeto ActionScript interno ou um objeto personalizado. Esse parâmetro está sempre à esquerda do operador ponto (.).

*propriedade\_ou\_método* O nome de uma propriedade ou de um método associado ao objeto. Todos os métodos e as propriedades válidos dos objetos internos estão listados nas tabelas de resumo Método e Propriedade de cada objeto. Esse parâmetro sempre está à direita do operador ponto (.).

*instancename* O nome da instância de um clipe de filme.

*childinstance* Uma instância do clipe de filme que seja filha do clipe do filme principal, ou que esteja nele aninhada.

*variável* Uma variável na Linha do tempo do nome da instância do clipe de filme à esquerda do operador ponto (.).

### Retorna

Nada.

### Descrição

Operador; usado para navegar por hierarquias de clipes de filmes, para acessar clipes de filmes, variáveis ou propriedades (filha) 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.

### Exemplo

O comando a seguir identifica o valor atual da variável `hairColor` no 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 a estrutura de uma matriz:

```
account.name = "Gary Smith";  
account.address = "123 Main St";  
account.city = "Any Town";  
account.state = "CA";  
account.zip = "12345";
```

### Consulte também

[ ] (acesso de matriz)

## ?: (condicional)

### Disponibilidade

Flash Player 4.

### Uso

*expressão1* ? *expressão2* : *expressão3*

### Parâmetros

*expressão1* Uma expressão que é avaliada para um valor booleano, normalmente uma expressão de comparação como `x < 5`.

*expressão2*, *expressão3* Valores de qualquer tipo.

### Retorna

Nada.

### Descrição

Operador; instrui o Flash a avaliar *expressão1* e, se o valor de *expressão1* for `true`, ele retornará o valor de *expressão2*; caso contrário, retornará o valor de *expressão3*.

### Exemplo

O comando a seguir atribui o valor de variável `x` a variável `z`, pois *expressão1* foi avaliada como `true`:

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

## / (divisão)

### Disponibilidade

Flash Player 4.

### Uso

*expressão1* / *expressão2*

### Parâmetros

*expressão* Um número ou uma variável que avalia um número.

### Retorna

Nada.

### Descrição

Operador (aritmético); divide *expressão1* por *expressão2*. O resultado da operação de divisão é um número de dupla precisão e ponto flutuante.

### Exemplo

O comando a seguir 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)

### Disponibilidade

Flash 1.

### Uso

*// comentário*

### Parâmetros

*comentário* Qualquer caractere.

### Retorna

Nada.

### Descrição

Comentário; indica o início de um comentário de script. Qualquer caractere que apareça entre o delimitador de comentário *//* e o caractere de fim de linha será interpretado como comentário e ignorado pelo interpretador ActionScript.

### Exemplo

Este script usa delimitadores de comentário para identificar a primeira, terceira, quinta e sétima linhas como comentários.

```
// registra a posição X do clipe de filme ball
ballX = ball._x;
// registra a posição Y do clipe de filme ball
ballY = ball._y;
// registra a posição X do clipe de filme bat
batX = bat._x;
// registra a posição Y do clipe de filme bat
batY = bat._y;
```

### Consulte também

*/\** (delimitador de comentário)

## /\* (delimitador de comentário)

### Disponibilidade

Flash Player 5.

### Uso

```
/* comentário */
/*
comentário
comentário
*/
```

### Parâmetros

*comentário* Qualquer caractere.

### Retorna

Nada.

### Descrição

Comentário; indica uma ou mais linhas de comentários de script. Qualquer caractere que apareça entre a marca de começo de comentário `/*` e a marca de fechamento de comentário `*/` é interpretado como comentário e ignorado pelo interpretador ActionScript. Use o primeiro tipo de sintaxe para identificar comentários de uma linha. Use o segundo tipo de sintaxe para identificar comentários de várias linhas sucessivas. Se a marca de fechamento `*/` não for usada com essa forma de delimitador de comentário, será retornada uma mensagem de erro.

### Exemplo

Este script usa delimitadores de comentário no início do script.

```
/* grava as posições X e Y dos  
clipes de filme ball e bat  
*/
```

```
ballX = ball._x;  
ballY = ball._y;  
batX = bat._x;  
batY = bat._y;
```

### Consulte também

`//` (delimitador de comentário)

## **/= (atribuição de divisão)**

### Disponibilidade

Flash Player 4.

### Uso

*expressão1 /= expressão2*

### Parâmetros

*expressão1*, *expressão2* Um número ou uma variável que é avaliada para um número.

### Retorna

Nada.

### Descrição

Operador (atribuição composta aritmética); atribui a *expressão1* o valor da *expressão1* / *expressão2*. Por exemplo, os dois comandos a seguir são equivalentes:

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

### Exemplo

O código a seguir ilustra o uso do operador `/=` com variáveis e números.

```
x = 10;  
y = 2;  
x /= y;  
// x agora contém o valor 5
```

## [] (acesso de matriz)

### Disponibilidade

Flash Player 4.

### Uso

```
myArray = [ "a0", a1, ...aN];  
myMultiDimensionalArray = [[ "a0", ...aN], ...[ "a0", ...aN]]  
myArray[E] = valor  
myMultiDimensionalArray[E][E] = valor  
objeto["valor"];
```

### Parâmetros

*myArray* O nome de uma matriz.

*a0, a1, ...aN* Elementos em uma matriz.

*myMultiDimensionalArray* O nome de uma matriz multidimensional simulada.

*E* O número (ou índice) de um elemento em uma matriz.

*objeto* O nome de um objeto.

*valor* Uma sequência de caracteres ou expressão que dá nome a uma propriedade do objeto.

### Retorna

Nada.

### Descrição

Operador; inicializa uma nova matriz ou uma matriz multidimensional com os elementos especificados (*a0*, e assim por diante), ou acessa elementos em uma matriz. O operador de acesso de matriz permite definir e recuperar dinamicamente nomes de instância, de variável e de objeto. Além disso, permite o acesso às propriedades de objeto.

Uso 1: Uma matriz é um objeto cujas propriedades são denominadas *elementos*, que são identificados individualmente por um número chamado de *índice*. Na criação de uma matriz, os elementos ficam entre o operador de acesso de matriz (ou *colchetes*). Uma matriz pode conter elementos de vários tipos. Por exemplo, a matriz a seguir, denominada *funcionário*, possui três elementos; o primeiro é um número e os outros dois são sequências de caracteres (dentro de aspas).

```
funcionário = [15, "Bárbara", "Erick"];
```

Uso 2: Para simular matrizes multidimensionais, é possível aninhar os colchetes. O código a seguir cria uma matriz denominada *ticTacToe* com três elementos; cada um deles também é uma matriz com três elementos.

```
ticTacToe = [[1,2,3],[4,5,6],[7,8,9]];
```

```
// escolha Depurar > Listar Variáveis no modo Testar filme  
// para visualizar uma lista dos elementos de matriz
```

Uso 3: Coloque o índice de cada elemento entre colchetes para acessá-lo diretamente; é possível adicionar um novo elemento a uma matriz, alterar ou recuperar o valor de um elemento existente. O primeiro elemento de uma matriz é sempre 0:

```
myArray[0] = 15;  
myArray[1] = "Olá";  
myArray[2] = true;
```

Use colchetes para adicionar um quarto elemento, como no exemplo a seguir:

```
myArray[3] = "George";
```

Uso 4: Para acessar um elemento em uma matriz multidimensional, use colchetes. O primeiro conjunto de colchetes identifica o elemento na matriz original, e o segundo conjunto identifica o elemento na matriz aninhada. A linha de código a seguir envia o número 6 para a janela Saída.

```
ticTacToe = [[1,2,3],[4,5,6],[7,8,9]];  
trace(ticTacToe[1][2]);
```

```
// retorna 6
```

Uso 5: É possível usar o operador de acesso de matriz em vez da função `eval` para definir e recuperar dinamicamente valores de nomes de cliques de filme ou qualquer propriedade de um objeto:

```
name["mc" + i] = "canto_esquerdo";
```

### Exemplo

Uso 1: Os exemplos de código a seguir mostram duas maneiras diferentes de criar um novo objeto Array vazio; a primeira linha usa colchetes.

```
myArray = [];  
myArray = new Array();
```

Uso 1 e 2: O exemplo a seguir cria uma matriz denominada `funcionário` e usa a ação `trace` para enviar os elementos para a janela Saída. Na quarta linha, é alterado um elemento da matriz e a quinta linha envia a matriz recentemente modificada para a janela Saída:

```
funcionário=["Bárbara", "George", "Maria"];  
trace(funcionário);  
// Bárbara, George, Maria  
funcionário[2]="Sam";  
trace(funcionário);  
// Bárbara, George, Sam
```

Uso 3: No exemplo a seguir, a expressão dentro dos colchetes ("`pedaço`" + `i`) é avaliada e o resultado é usado como nome da variável a ser recuperada no clique de filme `mc`. Neste exemplo, a variável `i` deve estar na mesma Linha de tempo que o botão. Se a variável `i` for igual a 5, por exemplo, o valor da variável `piece5` no clique de filme `mc` será exibido na janela Saída:

```
on(release){  
    x = mc["pedaço"+i];  
    trace(x);  
}
```

Uso 3: No código a seguir, a expressão dentro dos colchetes é avaliada e o resultado da avaliação é usado como o nome da variável a ser recuperada do nome do clique de filme:

```
group["A" + i];
```

Caso conheça a sintaxe de barra do ActionScript do Flash 4, use a função `eval` para obter o mesmo resultado:

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

Uso 3: Também é possível usar o operador de acesso de matriz do lado esquerdo de um comando de atribuição para definir dinamicamente a instância, a variável e os nomes de objeto:

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

#### Consulte também

Array (objeto), Object (objeto), `eval`

## ^(XOR bit a bit)

### Disponibilidade

Flash Player 5.

### Uso

*expressão1* ^ *expressão2*

### Parâmetros

*expressão1*, *expressão2* Um número.

### Retorna

Nenhum.

### 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.

### Exemplo

O exemplo a seguir usa o operador XOR bit a bit nos decimais 15 e 9 e atribui o resultado à variável `x`.

```
// 15 decimal = 1111 binário
// 9 decimal = 1001 binário
x = 15 ^ 9
trace(x)
// 1111 ^ 1001 = 0110
// retorna 6 decimal( = 0110 binário)
```

## ^= (atribuição XOR bit a bit)

### Disponibilidade

Flash Player 5.

### Uso

*expressão1* ^= *expressão2*

### Parâmetros

*expressão1*, *expressão2* Inteiros e variáveis.

### Retorna

Nenhum.

### Descrição

Operador (atribuição composta bit a bit); atribui a *expressão1* o valor de *expressão1* ^ *expressão2*. Por exemplo, os dois comandos a seguir são equivalentes:

```
x ^= y
x = x ^ y
```

### Exemplo

A seguir, há o exemplo de uma operação ^=.

```
// 15 decimal = 1111 binário
x = 15;
// 9 decimal = 1001 binário
y = 9;
trace(x ^= y);
//retorna 6 decimal ( = 0110 binário)
```

### Consulte também

^(XOR bit a bit)

## { } (inicializador de objeto)

### Disponibilidade

Flash Player 5.

### Uso

```
objeto = {nome1: valor1, nome2: valor2,...nomeN: valorN};
```

### Parâmetros

*objeto* O objeto a ser criado.

*nome1,2,...N* Os nomes das propriedades.

*valor1,2,...N* Os valores correspondentes de cada propriedade *nome*.

### Retorna

Nenhum.

### Descrição

Operador; cria um novo objeto e o inicializa com os pares de propriedades *nome* e *valor* especificados. Usar este operador é o mesmo que usar a sintaxe `new Object` e preencher os pares de propriedades com o operador de atribuição. O protótipo do objeto recém-criado é genericamente denominado como objeto *Object*.

### Exemplo

A primeira linha do código a seguir cria um objeto vazio usando o operador de inicialização do objeto; a segunda cria um novo objeto usando uma função construtora.

```
object = {};  
object = new Object();
```



O exemplo a seguir cria um objeto `account` e inicializa as propriedades `nome`, `endereço`, `cidade`, `estado`, `cep` e `saldo` com os respectivos valores.

```
account = { nome: "Betty Skate",
  endereço: "123 Main Street",
  cidade: "Blossomville",
  estado: "Califórnia",
  CEP: "12345",
  saldo: "1000" };
```

O exemplo a seguir mostra como inicializadores de matriz e de objeto podem ser aninhados um no outro.

```
person = { nome: "Gina Vechio",
  children: [ "Ruby", "Chickie", "Puppa" ] };
```

O exemplo a seguir usa as informações do exemplo anterior e apresenta o mesmo resultado usando as funções construtoras.

```
person = new Person();
person.name = 'Gina Vechio';
person.children = new Array();
person.children[0] = 'Ruby';
person.children[1] = 'Chickie';
person.children[2] = 'Puppa';
```

#### Consulte também

`[]` (acesso de matriz), `new`, `Object` (objeto)

## | (OR bit a bit)

### Disponibilidade

Flash Player 5.

### Uso

*expressão1* | *expressão2*

### Parâmetros

*expressão1*, *expressão2* Um número.

### Retorna

Nenhum.

### 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.

### Exemplo

A seguir, há o exemplo de uma operação OR bit a bit.

```
// 15 decimal = 1111 binário
x = 15;
// 9 decimal = 1001 binário
y = 9;
trace(x | y);
// 1111 | 0011 = 1111
//retorna 15 decimal (= 1111 binário)
```

## || (OR lógico)

### Disponibilidade

Flash Player 4.

### Uso

*expressão1* || *expressão2*

### Parâmetros

*expressão1*, *expressão2* Um valor ou uma expressão booleana convertida em um valor booleano.

### Retorna

Nenhum.

### Descrição

Operador (lógico); avalia a *expressão1* e a *expressão2*. O resultado será (*true*) se uma ou ambas as expressões forem avaliadas como *true*; o resultado será (*false*) apenas se ambas as expressões forem avaliadas como *false*. É possível usar o operador OR lógico com qualquer número de operandos; se algum operando for avaliado como *true*, o resultado será *true*.

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.

### Exemplo

O exemplo a seguir usa o operador || em um comando *if*: A segunda expressão é avaliada como *true* para que o resultado final seja *true*:

```
x = 10
y = 250
start = false
if(x > 25 || y > 200 || start){
    trace('o teste de OR lógico passou');
}
```

### Exemplo

Este exemplo demonstra como uma expressão não-booleana pode apresentar um resultado inesperado. Se a expressão da esquerda for convertida em *true*, esse resultado será retornado sem converter a expressão da direita.

```
function fx1(){
    trace ("fx1 chamado");
    retorna true;
}
function fx2(){
    trace ("fx2 chamado");
    return true;
}
if (fx1() || fx2()){
    trace ("comando IF inserido");
}
//O que se segue é enviado para a janela Saída:
// fx1 chamado
// comando IF inserido
```

## |= (atribuição OR bit a bit)

### Disponibilidade

Flash Player 5.

### Uso

*expressão1* |= *expressão2*

### Parâmetros

*expressão1*, *expressão2* Um número ou uma variável.

### Retorna

Nenhum.

### Descrição

Operador (atribuição bit a bit); atribui a *expressão1* o valor de *expressão1* | *expressão2*. Por exemplo, os dois comandos a seguir são equivalentes:

```
x |= y;  
x = x | y;
```

### Exemplo

O exemplo a seguir usa o operador |=:

```
// 15 decimal = 1111 binário  
x = 15;  
// 9 decimal = 1001 binário  
y = 9;  
trace(x |= y);  
// 1111 |= 1001  
// retorna 15 decimal (= 1111 binário)
```

### Consulte também

| (OR bit a bit)

## ~ (NOT bit a bit)

### Disponibilidade

Flash Player 5.

### Uso

~ *expressão*

### Parâmetros

*expressão* Um número.

### Retorna

Nenhum.

### Descrição

Operador (bit a bit); converte a *expressão* em um inteiro não assinado de 32 bits, depois inverte os bits. Uma operação NOT bit a bit altera o sinal de um número e subtrai 1.

### Exemplo

O exemplo a seguir mostra uma operação NOT bit a bit executada em uma variável.

```
a = 0;
trace ("quando a = 0, ~a = "+~a);
// quando a = 0, ~a = -1
a = 1;
trace ("quando a = 1, ~a = "+~a);
// quando a = 0, ~a = -2
// portanto, ~0=-1 e ~1=-2
```

## + (adição)

### Disponibilidade

Flash Player 4; Flash Player 5. No Flash 5, + é um operador numérico ou um concatenador de seqüências de caracteres, dependendo do tipo de dado do parâmetro. 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 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)
```

### Uso

```
expressão1 + expressão2
```

### Parâmetros

*expressão1*, *expressão2* Números ou seqüências de caracteres.

### Retorna

Nenhum.

### Descrição

Operador; adiciona expressões numéricas ou concatena (combina) 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 inteiras, a soma será um inteiro; se uma ou ambas as expressões forem números de ponto flutuante, a soma será um número de ponto flutuante.

### Exemplo

O exemplo a seguir concatena duas seqüências de caracteres e exibe o resultado na janela Saída.

```
nome = "Cola";
instrumento = "Baterias";
trace (nome + " toca " + instrumento);
```

### Exemplo

As variáveis associadas a campos de texto dinâmico e de entrada têm a sequência de caracteres como tipo de dado. No exemplo a seguir, o depósito da variável é um campo de texto de entrada no Palco. Depois que um usuário inserir um valor, o script tenta adicionar o depósito a `oldBalance`. Contudo, como depósito é um tipo de dado de sequência de caracteres, o script concatena (combina os dados para formar uma sequência de caracteres) os valores da variável em vez de somá-los.

```
oldBalance = 1345,23;  
currentBalance = deposit + oldBalance;  
trace (currentBalance);
```

Por exemplo, se um usuário inserir 475 no campo de texto de depósito, a ação `trace` envia o valor 4751345,23 para a janela Saída.

Para corrigir isso, use a função `Number` para converter a sequência de caracteres em um número, como no exemplo a seguir:

```
currentBalance = Number(deposit) + oldBalance;
```

### Exemplo

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, 5,75, que é um número de ponto flutuante, na janela Saída:

```
trace (2,5 + 3,25);
```

### Consulte também

`add`

## **+= (atribuição de adição)**

### Disponibilidade

Flash Player 4.

### Uso

*expressão1* += *expressão2*

### Parâmetros

*expressão1*, *expressão2* Números ou seqüências de caracteres.

### Retorna

Nada.

### Descrição

Operador (atribuição composta aritmética); atribui a *expressão1* o valor de *expressão1* + *expressão2*. Por exemplo, os dois comandos a seguir têm o mesmo resultado:

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

Este operador também executa concatenação de seqüências de caracteres. Todas as regras do operador de adição (+) são aplicadas ao operador de atribuição de adição (+=).

### Exemplo

O exemplo a seguir mostra um uso numérico do operador +=.

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

Este exemplo usa o operador += com uma expressão de sequência de caracteres e envia "Meu nome é Gilberto" para a janela Saída.

```
x = "Meu nome é"  
x += "Gilberto"  
trace (x)
```

### Consulte também

+ (adição)

## < (menor que)

### Disponibilidade

Flash Player 4; Flash Player 5. No Flash 5, < (menor que) é um operador de comparação que pode gerenciar vários 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 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)
```

### Uso

*expressão1* < *expressão2*

### Parâmetros

*expressão1*, *expressão2* Números ou seqüências de caracteres.

### Descrição

Operador (comparação); compara duas expressões e determina se *expressão1* é menor que *expressão2*; em caso positivo, o operador retorna *true*. Se *expressão1* for maior ou igual a *expressão2*, o operador retorna *false*. As expressões de seqüência de caracteres são avaliadas em ordem alfabética; todas as letras maiúsculas vêm antes das minúsculas.

### Exemplo

Os exemplos a seguir ilustram retornos `true` e `false` para comparações numéricas e de seqüências de caracteres:

```
3 < 10;  
// true  
  
10 < 3;  
// false  
  
"Allen" < "Jack";  
// true  
  
"Jack" < "Allen";  
// false  
  
"11" < "3";  
//true  
  
"11" < 3;  
// comparação numérica  
// false  
  
"C" < "abc";  
// false  
  
"A" < "a";  
// true
```

## << (deslocamento para a esquerda bit a bit)

### Disponibilidade

Flash Player 5.

### Uso

*expressão1* << *expressão2*

### Parâmetros

*expressão1* Número ou expressão a ser deslocada para a esquerda.

*expressão2* Um número ou expressão que converte em um inteiro de 0 a 31.

### Retorna

Nada.

### 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 bit que estiverem vazias como resultado dessa operação são preenchidas com 0. O deslocamento de um valor em uma posição para a esquerda é o equivalente a multiplicá-lo por 2.

### Exemplo

No exemplo a seguir, o inteiro 1 é deslocado 10 bits para a esquerda.

```
x = 1 << 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.

No exemplo a seguir, o inteiro 7 é deslocado 8 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 bits para a esquerda é 11100000000 binário e 11100000000 binário é 1792 decimal.

#### Consulte também

$\gg$ = (deslocamento para a direita bit a bit e atribuição),  $\gg$  (deslocamento para a direita bit a bit),  $\ll$ = (deslocamento para a esquerda bit a bit e atribuição)

## $\ll$ = (deslocamento para a esquerda bit a bit e atribuição)

#### Disponibilidade

Flash Player 5.

#### Uso

*expressão1*  $\ll$ = *expressão2*

#### Parâmetros

*expressão1* Número ou expressão a ser deslocada para a esquerda.

*expressão2* Um número ou expressão que converte em um inteiro de 0 a 31.

#### Retorna

Nada.

#### Descrição

Operador (atribuição composta bit a bit); esse operador executa uma operação de deslocamento para esquerda bit a bit e armazena o conteúdo como um resultado na *expressão1*. As duas expressões a seguir são equivalentes.

$A \ll B$   
 $A = (A \ll B)$

#### Consulte também

$\ll$  (deslocamento para a esquerda bit a bit),  $\gg$ = (deslocamento para a direita bit a bit e atribuição),  $\gg$  (deslocamento para a direita bit a bit)

## $\leq$ (menor ou igual a)

#### Disponibilidade

Flash Player 4.

Arquivo do Flash 4:

$x \leq y$

Arquivo do Flash 5 convertido:

`Number(x) <= Number(y)`

#### Uso

*expressão1*  $\leq$  *expressão2*



## Parâmetros

*expressão1, expressão2* Números ou seqüências de caracteres.

## Retorna

Nada.

## Descrição

Operador (comparação); compara duas expressões e determina se *expressão1* é menor ou igual a *expressão2*; em caso positivo, o operador retorna *true*. Se *expressão1* for maior que *expressão2*, o operador retorna *false*. As expressões de seqüência de caracteres são avaliadas em ordem alfabética; todas as letras maiúsculas vêm antes das minúsculas.

No Flash 5, o operador menor ou igual a (*<=*) é um operador de comparação, com capacidade para gerenciar vários 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 a conversão de um arquivo do Flash 4 que contém uma comparação do tipo numérica.

## Exemplo

Os exemplos a seguir ilustram resultados *true* e *false* para comparações numéricas e de seqüências de caracteres:

```
5 <= 10;
// true

2 <= 2;
// true

10 <= 3;
// false

"Allen" <= "Jack";
// true

"Jack" <= "Allen";
// false

"11" <= "3";
//true

"11" <= 3;
// comparação numérica
// false

"C" <= "abc";
// false

"A" <= "a";
// true
```

## ⟷ (diferença)

### Disponibilidade

Flash 2.

### Uso

*expressão1* <> *expressão2*

### Parâmetros

*expressão1*, *expressão2* Número, sequência de dados, valor booleano, variável, objeto, matriz ou função.

### Retorna

Nada.

### Descrição

Operador (diferença); testa o oposto exato do operador ==. Se *expressão1* for igual a *expressão2*, o resultado será *false*. Assim como 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 !=.

### Consulte também

!= (diferença)

## = (atribuição)

### Disponibilidade

Flash Player 4.

Arquivo do Flash 4:

*x* = *y*

Arquivo do Flash 5 convertido:

Number(*x*) == Number(*y*)

### Uso

*expressão1* = *expressão2*

### Parâmetros

*expressão1* Variável, elemento de uma matriz ou propriedade de um objeto.

*expressão2* Valor de qualquer tipo.

### Retorna

Nada.

### Descrição

Operador; atribui o tipo de *expressão2* (o parâmetro da direita) à variável, ao elemento da matriz ou à propriedade em *expressão1*.

No Flash 5, = é um operador de atribuição e o operador == é usado para avaliar a igualdade. No Flash 4, = é um operador de igualdade 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.

#### Exemplo

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"
```

#### Consulte também

== (igualdade)

## -= (atribuição de subtração)

#### Disponibilidade

Flash Player 4.

#### Uso

```
expressão1 -= expressão2
```

#### Parâmetros

*expressão1*, *expressão2* Número ou expressão que avalie um número.

#### Retorna

Nada.

#### Descrição

Operador (atribuição composta aritmética); atribui a *expressão1* o valor de *expressão1* - *expressão2*. Por exemplo, os dois comandos a seguir são equivalentes:

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

As expressões de sequência de caracteres devem ser convertidas em números ou será retornado NaN.

#### Exemplo

O exemplo a seguir usa o operador -= para subtrair 10 de 5 e atribui o resultado à variável x.

```
x = 5;  
y = 10;  
x -= y  
trace(x);  
//retorna -5
```

#### Exemplo

O exemplo a seguir mostra como converter sequências de caracteres em números.

```
x = "5";  
y = "10";  
x -= y;  
trace(x);  
// retorna -5
```

## == (igualdade)

### Disponibilidade

Flash Player 5.

### Uso

*expressão1* == *expressão2*

### Parâmetros

*expressão1*, *expressão2* Número, sequência de caracteres, valor Booleano, variável, objeto, matriz ou função.

### Retorna

Nada.

### 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 parâmetro:

- Números e valores booleanos são comparados por valor e, se tiverem o mesmo valor, são considerados iguais.
- Expressões de sequência de caracteres são iguais se tiverem o mesmo número de caracteres e os caracteres forem idênticos.
- 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.

### Exemplo

O exemplo a seguir usa o operador `==` com um comando `if`:

```
a = "David" , b = "David";
if (a == b){
    trace("David é David");
}
```

### Exemplo

Estes exemplos mostram os resultados de operações que comparam tipos misturados.

```
x = "5"; y = "5";
trace(x == y);
// true
```

```
x = "5"; y = "66";
trace(x ==y);
// false
```

```
x = "chris"; y = "steve";
trace (x == y);
//false
```

### Consulte também

`!=` (diferença), `===` (igualdade estrita), `!==` (diferença estrita)

## === (igualdade estrita)

### Disponibilidade

Flash Player 6.

### Uso

*expressão1* === *expressão2*

### Descrição

Operador; testa a igualdade de duas expressões; o operador de igualdade estrita é executado da mesma forma que o operador de igualdade, exceto pela conversão dos tipos de dados. Se ambas as expressões forem idênticas, inclusive os tipos de dados, o resultado será `true`.

A definição de *igual* depende do tipo de dado do parâmetro:

- Números e valores booleanos são comparados por valor e, se tiverem o mesmo valor, são considerados iguais.
- Expressões de sequência de caracteres são iguais se tiverem o mesmo número de caracteres e os caracteres forem idênticos.
- 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.

### Exemplo

O código a seguir exibe o valor retornado de operações que usam os operadores de igualdade, igualdade estrita e diferença estrita.

```
s1 = new String("5");
s2 = new String("5");
s3 = new String("Hello");
n  = new Number(5);
b  = new Boolean(true);
```

```
s1 == s2; // true
s1 == s3; // false
s1 == n;  // true
s1 == b;  // false
```

```
s1 === s2; // true
s1 === s3; // false
s1 === n;  // false
s1 === b;  // false
```

```
s1 !== s2; // false
s1 !== s3; // true
s1 !== n;  // true
s1 !== b;  // true
```

### Consulte também

`==` (igualdade), `!=` (diferença), `===` (igualdade estrita)

## > (maior que)

### Disponibilidade

Flash Player 5.

### Uso

*expressão1* > *expressão2*

### Parâmetros

*expressão1*, *expressão2* Um inteiro, um número de ponto flutuante ou uma sequência de dados.

### Retorna

Nada.

### 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).

## >= (maior ou igual a)

### Disponibilidade

Flash Player 4.

Arquivo do Flash 4:

`x > y`

Arquivo do Flash 5 convertido:

`Number(x) > Number(y)`

### Uso

*expressão1* >= *expressão2*

### Parâmetros

*expressão1*, *expressão2* Uma sequência de dados, um inteiro ou um número de ponto flutuante.

### Retorna

Nada.

### 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).

No Flash 5, maior ou igual a (>) é um operador de comparação com capacidade para gerenciar vários 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.

## >> (deslocamento para a direita bit a bit)

### Disponibilidade

Flash Player 5.

### Uso

*expressão1* >> *expressão2*

### Parâmetros

*expressão1* Número ou expressão a ser deslocada para a direita.

*expressão2* Um número ou expressão que converte em um inteiro de 0 a 31.

### Retorna

Nada.

### 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 a direita são descartados. Para preservar o sinal da *expressão* original, os bits na esquerda serão preenchidos com 0, se o bit mais significativo (o bit mais à esquerda) de *expressão1* for 0, e preenchido com 1, se o bit mais significativo for 1. O deslocamento de um valor em uma posição para a direita equivale à divisão por 2 e ao descarte do resto.

### Exemplo

O exemplo a seguir converte 65535 em um inteiro de 32 bits e o desloca 8 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 (dezesesseis 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 1 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 de 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)

## >>= (deslocamento para a direita bit a bit e atribuição)

### Disponibilidade

Flash Player 5.

### Uso

*expressão1* ==>>*expressão2*

### Parâmetros

*expressão1* Número ou expressão a ser deslocada para a esquerda.

*expressão2* Um número ou expressão que converte em um inteiro de 0 a 31.

### Retorna

Nada.

### Descrição

Operador (atribuição composta bit a bit); este operador executa uma operação de deslocamento para direita bit a bit e armazena o conteúdo como um resultado em *expressão1*.

### 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;
}
trace(convertToBinary(479));
// Retorna a seqüência de caracteres 00000000000000000000000111011111
//A seqüência de caracteres acima é a representação binária do número decimal
// número 479
```

### Consulte também

<< (deslocamento para a esquerda bit a bit)



## >>> (deslocamento para a direita não assinado bit a bit)

### Disponibilidade

Flash Player 5.

### Uso

*expressão1* >>> *expressão2*

### Parâmetros

*expressão1* Número ou expressão a ser deslocada para a direita.

*expressão2* Um número ou uma expressão que converte em um inteiro de 0 a 31.

### Retorna

Nada.

### 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.

### Exemplo

O exemplo a seguir converte -1 em um inteiro de 32 bits e o desloca 1 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)

## >>>= (deslocamento para a direita não assinado bit a bit e atribuição)

### Disponibilidade

Flash Player 5.

### Uso

*expressão1* >>>= *expressão2*

### Parâmetros

*expressão1* Número ou expressão a ser deslocada para a esquerda.

*expressão2* Um número ou expressão que converte em um inteiro de 0 a 31.

### Retorna

Nada.

### Descrição

Operador (atribuição composta bit a bit); executa uma operação de deslocamento para direita bit a bit não assinada e armazena o conteúdo como um resultado em *expressão1*. As duas expressões a seguir são equivalentes:

```
A >>>= B  
A = (A >>> B)
```

### Consulte também

>>> (deslocamento para a direita não assinado bit a bit), >>= (deslocamento para a direita bit a bit e atribuição)

## Accessibility (objeto)

O objeto Accessibility é um conjunto de métodos usado para criar conteúdo que pode ser acessado com o ActionScript. No Flash MX, só existe um método.

Este objeto está disponível no Flash Player 6.

## Resumo de métodos do objeto Arguments

Propriedade	Descrição
Accessibility.isActive	Indica se um programa leitor de tela está ativo.

## Accessibility.isActive

### Disponibilidade

Flash Player 6.

### Uso

Accessibility.isActive()

### Parâmetros

Nenhum.

### Retorna

Um valor booleano.

### Descrição

Método; indica se um programa leitor de tela está ativo ou não atualmente. Use este método quando desejar que seu filme tenha um comportamento diferente diante de um leitor de tela.

### Consulte também

System.capabilities.hasAccessibility

## add

### Disponibilidade

Flash Player 4.

### Uso

*seq\_caract1* add *seq\_caract2*

### Parâmetros

*seq\_ência de caracteres1*, *seq\_ência de caracteres2* Uma seq\_ência de caracteres.

### Retorna

Nada.

### Descrição

Operador; concatena (combina) duas ou mais seq\_ências de caracteres. O operador add substitui o operador add (&) do Flash 4; os arquivos do Flash 4 que usam o operador & são convertidos automaticamente para usar o operador add na concatenação de seq\_ências de caracteres quando trazidos para o ambiente de criação do Flash 5. Entretanto, o operador add está obsoleto no Flash 5 e recomenda-se o uso do operador + na criação de conteúdo para o Flash 5 Player ou Flash Player 6. Use o operador add para concatenar seq\_ências de caracteres se estiver criando conteúdo para o Flash 4 ou versões anteriores do Player.

### Consulte também

+ (adição)

## and

### Disponibilidade

Flash Player 4.

### Uso

*condi\_ão1* and *condi\_ão2*

### Parâmetros

*condi\_ão1*, *condi\_ão2* Condições ou expressões que avaliam como true ou false.

### Retorna

Nada.

### Descrição

Operador; executa uma operação lógica AND no Flash Player 4. Se ambas as expressões forem avaliadas como true, toda a expressão é true. Esse operador está obsoleto no Flash 5 e os usuários são incentivados a usar o novo operador && .

### Consulte também

&& (AND de curto-circuito)

## arguments (objeto)

O objeto Arguments é uma matriz que contém os valores passados como parâmetros para qualquer função. Toda vez que uma função é chamada no ActionScript, um objeto Arguments é criado automaticamente para essa função. Além disso, é criada uma variante local, `arguments`, que permite a consulta ao objeto Arguments.

O objeto Arguments está disponível no Flash Player 6.

### Resumo de propriedades do objeto Arguments

Propriedade	Descrição
<code>arguments.callee</code>	Refere-se à função sendo chamada.
<code>arguments.caller</code>	Refere-se ao chamamento da função.
<code>arguments.length</code>	O número de parâmetros passados para uma função.

## arguments.callee

### Disponibilidade

Flash Player 5.

### Uso

`arguments.callee`

### Descrição

Propriedade; refere-se à função que está sendo chamada atualmente.

### Exemplo

É possível usar a propriedade `arguments.callee` para tornar uma função anônima repetitiva, como no exemplo a seguir:

```
factorial = function (x) {  
    if (x <= 1) {  
        return 1;  
    }  
    else {  
        return x * arguments.callee(x-1);  
    }  
};
```

A seguir é apresentada uma função repetitiva nomeada:

```
function factorial (x) {  
    if (x <= 1) {  
        return 1;  
    }  
    else {  
        return x * factorial(x-1);  
    }  
}
```

## arguments.caller

### Disponibilidade

Flash Player 6.

### Uso

`arguments.caller`

### Descrição

Propriedade; refere-se ao objeto Arguments da função chamada.

## arguments.length

### Disponibilidade

Flash Player 6.

### Uso

`arguments.length`

### Descrição

Propriedade; o número de parâmetros realmente passados para uma função.

## 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. 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.

```
myArray[0] = "Janeiro"
myArray[1] = "Fevereiro"
myArray[2] = "Março"
myArray[3] = "Abril"
```

Para criar um objeto Array, use o construtor `new Array` ou o operador de acesso de matriz (`[]`). Para acessar os elementos de uma matriz, use o operador de acesso de matriz (`[]`).

No Flash MX, o objeto Array se tornou um objeto nativo. Assim, você poderá observar uma melhora radical no desempenho.

## Resumo de métodos do objeto Array

Método	Descrição
<code>Array.concat</code>	Concatena os parâmetros e os retorna como uma nova matriz.
<code>Array.join</code>	Reúne todos os elementos de uma matriz em uma sequência de caracteres.
<code>Array.pop</code>	Remove o último elemento de uma matriz e retorna seu valor.
<code>Array.push</code>	Adiciona um ou mais elementos ao fim de uma matriz e retorna o novo tamanho da matriz.
<code>Array.reverse</code>	Inverte a direção de uma matriz.
<code>Array.shift</code>	Remove o primeiro elemento de uma matriz e retorna seu valor.
<code>Array.slice</code>	Extrai uma seção de uma matriz e a retorna como uma nova matriz.
<code>Array.sort</code>	Classifica uma matriz no local.

Método	Descrição
<code>Array.sortOn</code>	Classifica uma matriz com base em um campo da matriz.
<code>Array.splice</code>	Adiciona e/ou remove elementos de uma matriz.
<code>Array.toString</code>	Retorna um valor de sequência de caracteres que representa os elementos no objeto <code>Array</code> .
<code>Array.unshift</code>	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
<code>Array.length</code>	Retorna o tamanho da matriz.

## Construtor do objeto `Array`

### Disponibilidade

Flash Player 5.

### Uso

```
new Array()
new Array(tamanho)
new Array(elemento0, elemento1, elemento2,...elementoN)
```

### Parâmetros

*tamanho* Um inteiro que especifica o número de elementos na matriz. No caso de elementos não contíguos, o parâmetro *tamanho* especifica o número do índice do último elemento na matriz mais 1.

*elemento0...elementoN* Uma lista de dois ou mais valores arbitrários. Os valores podem ser números, seqüências de caracteres, objetos ou outras matrizes. O primeiro elemento em uma matriz sempre tem um índice, ou posição 0.

### Retorna

Nada.

### Descrição

Construtor; permite a criação de uma matriz. Use o construtor para criar diferentes tipos de matrizes: uma matriz vazia, uma matriz com um tamanho específico, mas cujos elementos não têm valores, ou uma cujos elementos têm valores específicos.

Uso 1: Se os parâmetros não forem especificados, será criada uma matriz com tamanho 0.

Uso 2: Se apenas um tamanho for especificado, será criada uma matriz com o *tamanho* do número de elementos, sem valores.

Uso 3: Se os parâmetros de *elemento* forem usados para especificar os valores, será criada uma matriz com valores específicos.

### Exemplo

Uso 1: O exemplo a seguir cria um novo objeto `Array` com um tamanho inicial 0.

```
myArray = new Array();
```

Uso 3: O exemplo a seguir cria o objeto `new Array go_gos`, com o tamanho inicial 5.

```
go_gos = new Array("Belinda", "Gina", "Kathy", "Charlotte", "Jane");
trace(go_gos.join(" + "));
```

Os elementos iniciais da matriz `go_gos` são estes:

```
go_gos[0] = "Belinda";
go_gos[1] = "Gina";
go_gos[2] = "Kathy";
go_gos[3] = "Charlotte";
go_gos[4] = "Jane";
```

O código a seguir adiciona o quinto elemento à matriz `go_gos` e altera o primeiro elemento:

```
go_gos[5] = "Donna";
go_gos[1] = "Nina";
trace(go_gos.join(" + "));
```

#### Consulte também

`Array.length, []` (acesso de matriz)

## Array.concat

### Disponibilidade

Flash Player 5.

### Uso

```
myArray.concat(valor0,valor1,...valorN)
```

### Parâmetros

*valor0,...valorN* Números, elementos ou seqüências de caracteres a serem concatenados em uma nova matriz.

### Retorna

Nada.

### Descrição

Método; concatena os elementos especificados nos parâmetros, se houver, com os elementos em *myArray*, e cria uma nova matriz. Se os parâmetros *valor* especificarem uma matriz, os elementos dessa matriz serão concatenados, em vez da própria matriz. A matriz *myArray* permanece inalterada.

### Exemplo

O código a seguir concatena duas matrizes:

```
alpha = new Array("a","b","c");
numeric = new Array(1,2,3);
alphaNumeric=alpha.concat(numeric);
trace(alphaNumeric);
// cria 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)
trace(nums);
// cria matriz [1,3,5,2,4,6,7,8,9]
```

As matrizes aninhadas não são achatadas da mesma forma que as matrizes comuns. Os elementos de uma matriz aninhada não são desmembrados em elementos separados na matriz `x`, como observado no exemplo abaixo:

```
a = new array ("a","b","c");
n = new array(1, [2, 3], 4);
// 2 e 3 são elementos de uma matriz aninhada
x = a.concat(n);
x[0] = "a"
x[1] = "b"
x[2] = "c"
x[3] = 1
x[4] = 2, 3
x[5] = 4
```

## Array.join

### Disponibilidade

Flash Player 5.

### Uso

```
myArray.join([separador])
```

### Parâmetros

*separador* Um caractere ou uma seqüência de caracteres que separa elementos da matriz na seqüência de caracteres retornada. A omissão desse parâmetro resulta no uso de uma vírgula como separador padrão.

### Retorna

Nada.

### Descrição

Método; converte os elementos de uma matriz em seqüências de caracteres, insere o separador especificado entre eles, concatena esses elementos e retorna a seqüência de caracteres resultante. A matriz aninhada é sempre separada por uma vírgula, não pelo separador passado para o método `join`.

### Exemplo

O exemplo a seguir cria uma matriz com três elementos. Depois, reúne a matriz três vezes—usando o separador padrão, uma vírgula e um espaço, e um sinal de mais—e os exibe na janela Saída:

```
a = new Array("Terra","Lua","Sol")
trace(a.join());
// retorna Terra, Lua, Sol
trace(a.join(" - "));
// retorna Terra - Lua - Sol
trace(a.join(" + "));
// retorna Terra + Lua + Sol
```



## Array.length

### Disponibilidade

Flash Player 5.

### Uso

*myArray.length*

### Descrição

Propriedade; contém o tamanho da matriz. Essa propriedade é atualizada automaticamente quando são adicionados novos elementos à matriz. Ao atribuir um valor ao elemento de uma matriz (por exemplo, *myArray[index] = valor*), se *índice* for um número e *índice+1* for maior do que a propriedade *length*, a propriedade *length* será atualizada para *índice + 1*.

### Exemplo

O código a seguir explica como a propriedade *length* é atualizada.

```
// tamanho inicial é 0
myArray = new Array();
myArray[0] = 'a';
//myArray.length é atualizada para 1
myArray[1] = 'b';
//myArray.length é atualizada para 2
myArray[9] = 'c';
//myArray.length é atualizada para 10
```

## Array.pop

### Disponibilidade

Flash Player 5.

### Uso

*myArray.pop()*

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Método; remove o último elemento de uma matriz e retorna o valor desse elemento.

### 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();
trace(popped);
// retorna peixe
```

## Array.push

### Disponibilidade

Flash Player 5.

### Uso

```
myArray.push(valor,...)
```

### Parâmetros

*valor* Um ou mais valores a serem anexados à matriz.

### Retorna

O tamanho da nova matriz.

### Descrição

Método; adiciona um ou mais elementos ao fim de uma matriz e retorna o novo tamanho da matriz.

### Exemplo

O exemplo a seguir cria a matriz `myPets` com dois elementos, `gato` e `cachorro`. A segunda linha adiciona dois elementos à matriz. Depois de chamar o método `push`, a variável `pushed` contém quatro elementos. Como o método `push` retorna o novo tamanho da matriz, a ação `trace` na última linha envia o novo tamanho de `myPets` (4) para a janela Saída:

```
myPets = ["gato", "cachorro"];  
pushed = myPets.push("pássaro", "peixe");  
trace(pushed);
```

## Array.reverse

### Disponibilidade

Flash Player 5.

### Uso

```
myArray.reverse()
```

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Método; reverte a matriz no local.

### 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

### Disponibilidade

Flash Player 5.

### Uso

```
myArray.shift()
```

### Parâmetros

Nenhum.

### Retorna

O primeiro elemento de uma matriz.

### Descrição

Método; remove o primeiro elemento de uma matriz e retorna esse elemento.

### Exemplo

O código a seguir cria a matriz `myPets` e, em seguida, remove o primeiro elemento da matriz e o atribui à variável `shifted`.

```
myPets = ["gato", "cachorro", "pássaro", "peixe"];
shifted = myPets.shift();
trace(shifted);
// retorna gato
```

### Consulte também

`Array.pop`

## Array.slice

### Disponibilidade

Flash Player 5.

### Uso

```
myArray.slice(início, fim)
```

### Parâmetros

*início* Um número que especifica o índice do ponto inicial da fatia. 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 da fatia. Se esse argumento for omitido, a fatia incluirá todos os elementos do início 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.

### Retorna

Nada.

### 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 *início* e todos os elementos até, mas não incluindo, o elemento *fim*.

# Array.sort

## Disponibilidade

Flash Player 5.

## Uso

```
myArray.sort([compareFunction])
```

## Parâmetros

*compareFunction* Uma função de comparação opcional usada para determinar a ordem de classificação de elementos de uma matriz. Dados os elementos A e B, o parâmetro *orderfunc* pode ter um dos três seguintes valores:

- -1 se A aparecer antes de B na sequência classificada
- 0 se A = B
- 1 se A aparecer depois de B na sequência classificada

## Retorna

Nada.

## Descrição

Método; classifica a matriz no local, sem fazer uma cópia. Se o argumento *orderfunc* for omitido, o Flash classificará os elementos no local com o operador de comparação <.

## Exemplo

O exemplo a seguir usa `Array.sort` sem especificar o parâmetro *compareFunction*.

```
var fruits = ["oranges", "apples", "strawberries", "pineapples", "cherries"];
trace(fruits.join());
fruits.sort()
trace(fruits.join());
```

## Saída:

oranges,apples,strawberries,pineapples,cherries  
é exibido 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 ("Classificado:");
for (var i=0; i< password.length; i++) {
    trace (passwords.join());
}
```

A execução do código anterior exibe o seguinte resultado na janela Saída.

```
daniel:1234
gary:foo
john:snafu
mike:bar
steve:yuck
```

## Array.sortOn

### Disponibilidade

Flash Player 6.

### Uso

`Array.sortOn(fieldName)`

### Parâmetros

*fieldName* Uma sequência de caracteres que identifica um campo em um elemento do Array para usar valor de classificação.

### Retorna

Nenhum.

### Descrição

Método; classifica os elementos de uma matriz com base em um campo da matriz. Se nenhum parâmetro *fieldName* for passado, a função falhará. Se vários parâmetros *fieldName* forem passados, o primeiro campo será convertido em um valor de sequência de caracteres e os parâmetros remanescentes serão ignorados.

Se algum dos elementos comparados não tiver o campo especificado no parâmetro *fieldName*, a classificação será o padrão do comportamento no método `Array.sort`.

### Exemplo

O exemplo a seguir cria uma nova matriz e a classifica com base no campo `city`:

```
var recArray = new Array();
recArray.push( { name: "bob", city: "omaha", zip: 68144 } );
recArray.push( { name: "greg", city: "kansas city", zip: 72345 } );
recArray.push( { name: "chris", city: "burlingame", zip: 94010 } );
recArray.sortOn("city");
// resulta no seguinte:
recArray[0] = name: "chris", city: "burlingame", zip: 94010
recArray[1] = name: "greg", city: "kansas city", zip: 72345
recArray[2] = name: "bob", city: "omaha", zip: 68144
```

### Consulte também

`Array.sort`

## Array.splice

### Disponibilidade

Flash Player 5.

### Uso

```
myArray.splice(início, deleteCount, valor0, valor1...valorN)
```

### Parâmetros

*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 parâmetro *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. Se o valor for 0, nenhum elemento será excluído.

*valor* Zero ou mais valores a serem inseridos na matriz no ponto de inserção especificado no parâmetro *início*. Este parâmetro é opcional.

### Retorna

Nada.

### Descrição

Método; adiciona e remove elementos de uma matriz. Esse método modifica a matriz sem fazer uma cópia.

## Array.toString

### Disponibilidade

Flash Player 5.

**Uso**

`myArray.toString()`

**Parâmetros**

Nenhum.

**Retorna**

Uma seqüência de caracteres.

**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`, são convertidos em uma seqüência de caracteres concatenados e separados por vírgulas.

**Exemplo**

O exemplo a seguir cria `myArray`, a converte em uma seqüência de caracteres, e exibe 1,2,3,4,5 na janela Saída.

```
myArray = new Array();  
myArray[0] = 1;  
myArray[1] = 2;  
myArray[2] = 3;  
myArray[3] = 4;  
myArray[4] = 5;  
  
trace(myArray.toString());
```

## Array.unshift

**Disponibilidade**

Flash Player 5.

**Uso**

`myArray.unshift(valor1, valor2, ... valorN)`

**Parâmetros**

`valor1, ... valorN` Um ou mais números, elementos ou variáveis a serem inseridos no início da matriz.

**Retorna**

O novo tamanho da matriz.

**Descrição**

Método; adiciona um ou mais elementos ao início de uma matriz e retorna o novo tamanho da matriz.

## asfunction

**Disponibilidade**

Flash Player 5.

**Uso**

`asfunction:function, "parâmetro"`

## Parâmetros

*função* Um identificador para uma função.

*parâmetro* Uma sequência de caracteres que é passada para a função identificada no parâmetro *function*.

## Retorna

Nada.

## Descrição

Protocolo; um protocolo especial para URLs em campos de texto HTML. Nos campos de texto HTML, o texto pode ter um hiperlink usando a marca A de HTML. O atributo HREF da marca A contém um URL que pode servir para um protocolo padrão como HTTP, HTTPS ou FTP. O protocolo *asfunction* é um protocolo adicional, específico do Flash, que faz com que o link chame uma função do ActionScript.

## Exemplo

Neste exemplo, a função MyFunc é definida nas três primeiras linhas de código. A variável *textField* é associada a um campo de texto HTML. O texto "Clique em mim!" é um hiperlink dentro do campo de texto. A função MyFunc será chamada quando o usuário clicar no hiperlink:

```
function MyFunc(arg){
    trace ("Você clicou em mim!0 argumento era "+arg);
}
myTextField.text ="<A HREF=\"asfunction:MyFunc,Foo \">Clique em mim!</A>";
```

Ao clicar no hiperlink, os resultados a seguir são exibidos na janela Saída:

Você clicou em mim! 0 parâmetro foi Foo

# Boolean (função)

## Disponibilidade

Flash Player 5.

## Uso

Booleano(*expressão*)

## Parâmetros

*expressão* Uma expressão a ser convertida em um valor booleano.

## Retorna

Nada.

## Descrição

Função; converte a *expressão* do parâmetro em um valor booleano e retorna um valor da seguinte maneira:

- Se *expressão* for um valor booleano, o valor de retorno será *expressão*.
- Se *expressão* for um número e esse não for zero, o valor de retorno será *true*, caso contrário, o valor de retorno será *false*.
- Se *expressão* for uma sequência de caracteres, o método *toNumber* será chamado e o valor de retorno será *true* se o número não for zero, caso contrário, o valor de retorno será *false*.
- Se *expressão* for indefinida, o valor de retorno será *false*.
- Se *expressão* for um clipe de filme ou um objeto, o valor de retorno será *true*.



## Boolean (objeto)

O objeto Boolean é um objeto envoltório que funciona da mesma forma que o objeto Boolean JavaScript padrão. Use o objeto Boolean para recuperar o tipo de dados primitivo ou a representação de uma sequência de caracteres do objeto Boolean. No Flash MX, o objeto Boolean se tornou um objeto nativo. Assim, você poderá observar uma melhora radical no desempenho.

Use o construtor `new Boolean()` para criar uma instância do objeto Boolean antes de chamar seus métodos.

### Resumo de métodos do objeto Boolean

Método	Descrição
<code>Boolean.toString</code>	Retorna a representação da sequência de caracteres ( <code>true</code> ) ou ( <code>false</code> ) do objeto Boolean.
<code>Boolean.valueOf</code>	Retorna o tipo de valor primitivo do objeto Boolean especificado.

### Construtor do objeto Boolean

#### Disponibilidade

Flash Player 5.

#### Uso

```
new Boolean(x)
```

#### Parâmetros

*x* Qualquer expressão. Este parâmetro é opcional.

#### Retorna

Nada.

#### Descrição

Construtor; cria uma instância do objeto Boolean. Se o parâmetro *x* for omitido, o objeto Boolean será inicializado com um valor `false`. Caso especifique um valor para o parâmetro *x*, o método o avalia e retorna o resultado como um valor booleano de acordo com as regras estabelecidas na função Boolean (função).

**Observação:** Para manter a compatibilidade com o Flash Player 4, a manipulação de sequências de caracteres pelo objeto Boolean não usa o padrão ECMA-262.

#### Exemplo

O código a seguir cria um novo objeto Boolean vazio denominado `myBoolean`.

```
myBoolean = new Boolean();
```

## Boolean.toString

### Disponibilidade

Flash Player 5.

### Uso

*myBoolean.toString()*

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Método; retorna a representação da sequência de caracteres, `true` ou `false`, do objeto Boolean.

## Boolean.valueOf

### Disponibilidade

Flash Player 5.

### Uso

`Boolean.valueOf()`

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Método; retorna o tipo de valor primitivo do objeto Boolean especificado.

## break

### Disponibilidade

Flash Player 4.

### Uso

`break`

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Ação; é exibida em um loop (for, for..in, do while ou while) ou em um bloco de comandos associados a um case particular em uma ação switch. 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. Ao usar a ação break, o interpretador Flash ignora o resto dos comandos nesse bloco case e vai para o primeiro comando subsequente à ação switch anexada. Use a ação break para interromper uma série de loops aninhados.

### Exemplo

O exemplo a seguir usa a ação break para sair de um loop infinito.

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

### Consulte também

for, for..in, do while, while, switch, case

## Button (objeto)

Todos os símbolos de botão em um filme do Flash são instâncias do objeto Button. É possível dar um nome de instância a um botão no inspetor de propriedades e usar os métodos e as propriedades do objeto Button para manipular botões com o ActionScript. Nomes de instância de botão são exibidos no Movie Explorer e na caixa de diálogo Inserir caminho de destino no painel Actions.

O objeto Button herda propriedades do objeto Object.

O Flash Player 6 oferece suporte ao objeto Button.

## Resumo de métodos do objeto Button

Método	Descrição
Button.getDepth	Retorna a profundidade de uma instância de botão.

## Resumo de propriedades do objeto Button

Propriedade	Descrição
Button._alpha	O valor de transparência de uma instância de botão.
Button.enabled	Indica se o botão está ativo.
Button._focusrect	Indica se um botão focalizado tem um retângulo amarelo em volta dele.
Button._height	A altura de uma instância de botão, em pixels.
Button._highquality	Indica a qualidade do filme.
Button._name	O nome de uma instância de botão.
Button._parent	Uma referência à instância do clipe de filme que é o pai dessa instância.
Button._quality	Indica a qualidade do filme.

Propriedade	Descrição
<code>Button._rotation</code>	O grau de rotação de uma instância de botão.
<code>Button._soundbuftime</code>	Número de segundos para que um som seja pré-carregado.
<code>Button.tabEnabled</code>	Indica se um botão está incluído na ordenação de guia automática.
<code>Button.tabIndex</code>	Indica a ordem de guias de um objeto.
<code>Button._target</code>	O caminho de destino de uma instância de botão.
<code>Button.trackAsMenu</code>	Indica se outros botões podem receber eventos de liberação de mouse.
<code>Button._url</code>	O URL do arquivo SWF que criou a instância de botão.
<code>Button.useHandCursor</code>	Indica se o cursor mão é exibido quando o mouse passar sobre um botão.
<code>Button._visible</code>	Um valor booleano que determina se a instância de botão está oculta ou visível.
<code>Button._width</code>	A largura de uma instância de botão, em pixels.
<code>Button._x</code>	A coordenada x de uma instância de botão.
<code>Button._xmouse</code>	A coordenada x do cursor relativa à instância de um botão.
<code>Button._xscale</code>	O valor que especifica a porcentagem para o dimensionamento horizontal de uma instância de botão.
<code>Button._y</code>	A coordenada y de uma instância de botão.
<code>Button._ymouse</code>	A coordenada y do cursor relativa à instância de um botão.
<code>Button._yscale</code>	O valor que especifica a porcentagem para o dimensionamento vertical de uma instância de botão.

## Resumo de eventos do objeto Button

A tabela a seguir lista os resumos de eventos do objeto Button.

Método	Descrição
<code>Button.onDragOut</code>	Chamado enquanto o ponteiro está fora do botão, o botão do mouse é pressionado e rolado para fora da área do botão.
<code>Button.onDragOver</code>	Chamada 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.
<code>Button.onKeyUp</code>	Chamada quando uma tecla é liberada.
<code>Button.onKillFocus</code>	Chamada quando o foco é removido de um botão.
<code>Button.onPress</code>	Chamada quando o mouse é pressionado enquanto o ponteiro está sobre um botão.
<code>Button.onRelease</code>	Chamada quando o mouse é liberado enquanto o ponteiro está sobre um botão.
<code>Button.onReleaseOutside</code>	Chamada quando o mouse é liberado enquanto o ponteiro está fora de um botão, depois que o botão é pressionado enquanto o ponteiro está dentro do botão.
<code>Button.onRollOut</code>	Chamada quando o ponteiro rola para fora da área de um botão.
<code>Button.onRollOver</code>	Chamada quando o ponteiro do mouse rola sobre um botão.
<code>Button.onSetFocus</code>	Chamada quando um botão tem o foco de entrada e uma tecla é liberada.

## Button.\_alpha

### Disponibilidade

Flash Player 6.

### Uso

*myButton.\_alpha*

### Descrição

Propriedade; define ou recupera a transparência alfa (*valor*) do botão especificado por *Button*. A faixa de valores válidos vai de 0 (totalmente transparente) a 100 (totalmente opaco). Os objetos em um botão com *\_alpha* definido como 0 são ativos, apesar de invisíveis.

### Exemplo

O exemplo a seguir define a propriedade *\_alpha* de um botão denominado *star* como 30%.

```
on(release) {  
    star._alpha = 30;  
}
```

## Button.enabled

### Disponibilidade

Flash Player 6.

### Uso

*myButton.enabled*

### Descrição

Propriedade; um valor booleano que especifica se um botão está ativado. O valor padrão é *true*.

## Button.\_focusrect

### Disponibilidade

Flash Player 6.

### Uso

*myButton.\_focusrect*

### Descrição

Propriedade; um valor booleano que especifica se um botão tem um retângulo amarelo em volta dele quando tiver foco de teclado. Esta propriedade pode substituir a propriedade global *\_focusrect*.

## Button.getDepth

### Disponibilidade

Flash Player 6.

### Uso

*myButton.getDepth()*

### Retorna

Um inteiro.

### Descrição

Método; retorna a profundidade de uma instância de botão.

## Button.\_height

### Disponibilidade

Flash Player 6.

### Uso

*myButton.\_height*

### Descrição

Propriedade; define e recupera a altura do botão, em pixels.

### Exemplo

O exemplo de código a seguir define a altura e a largura de um botão quando o usuário clicar com o mouse:

```
myButton._width = 200;  
myButton._height = 200;
```

## Button.\_highquality

### Disponibilidade

Flash Player 6.

### Uso

*myButton.\_highquality*

### Descrição

Propriedade (global); especifica o nível de sem serrilhado aplicado no filme atual. 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.

### Exemplo

```
_highquality = 1;
```

### Consulte também

*\_quality*, *toggleHighQuality*

## Button.\_name

### Disponibilidade

Flash Player 6.

### Uso

*myButton.\_name*

### Descrição

Propriedade; retorna o nome de instância do botão especificado por *myButton*.

## Button.onDragOut

### Disponibilidade

Flash Player 6.

### Uso

*myButton.onDragOut*

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Manipulador de eventos; chamado quando o botão do mouse estiver pressionado sobre o botão e o ponteiro rolar para fora do botão.

## Button.onDragOver

### Disponibilidade

Flash Player 6.

### Uso

*myButton.onDragOver*

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Manipulador de eventos; chamado quando o usuário pressiona e arrasta o botão do mouse para fora e sobre o botão.

É necessário definir uma função que seja executada quando o evento é chamado.

### Exemplo

O exemplo a seguir define uma função para o método `onKeyDown` que envia uma ação `trace` à janela Saída.

```
myButton.onDragOver = function () {  
    trace ("onDragOver chamado");  
};
```

### Consulte também

[Button.onKeyUp](#)

## Button.onKeyDown

### Disponibilidade

Flash Player 6.

### Uso

*myButton.onKeyDown*

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Manipulador de eventos; chamado quando um botão tiver foco de teclado e uma tecla for pressionada. O evento `onKeyDown` é chamado sem nenhum parâmetro. Use os métodos `Key.getAscii` e `Key.getCode` para determinar qual tecla foi pressionada.

É necessário definir uma função que seja executada quando o evento é chamado.

### Exemplo

O exemplo a seguir define uma função para o método `onKeyDown` que envia uma ação `trace` à janela Saída.

```
myButton.onKeyDown = function () {  
    trace ("onKeyDown chamado");  
};
```

### Consulte também

[Button.onKeyUp](#)

## Button.onKeyUp

### Disponibilidade

Flash Player 6.

### Uso

*myButton.onKeyUp*

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Manipulador de eventos; chamado quando um botão tiver foco de entrada e uma tecla for liberada. O evento `onKeyUp` é chamado sem nenhum parâmetro. Use os métodos `Key.getAscii` e `Key.getCode` para determinar qual tecla foi pressionada.

É necessário definir uma função que seja executada quando o evento é chamado.



### Exemplo

O exemplo a seguir define uma função para o método `onKeyPress` que envia uma ação `trace` à janela Saída.

```
myButton.onKeyUp = function () {  
    trace ("onKeyUp chamado");  
};
```

## Button.onKillFocus

### Disponibilidade

Flash Player 6.

### Uso

```
myButton.onKillFocus = function (newFocus) {  
    comandos;  
};
```

### Parâmetros

*newFocus* O objeto em foco.

### Retorna

Nada.

### Descrição

Manipulador de eventos; um evento que é chamado quando um botão perde o foco do teclado. O método `onKillFocus` recebe um parâmetro, *newFocus*, que é um objeto representando o novo objeto a receber o foco. Se nenhum objeto receber o foco, *newFocus* conterá o valor `null`.

## Button.onPress

### Disponibilidade

Flash Player 6.

### Uso

```
myButton.onPress
```

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Manipulador de eventos; chamado quando um botão for pressionado. É necessário definir uma função que seja executada quando o evento é chamado.

### Exemplo

O exemplo a seguir define uma função para o método `onPress` que envia uma ação `trace` à janela Saída.

```
myButton.onPress = function () {  
    trace ("onPress chamado");  
};
```

## Button.onRelease

### Disponibilidade

Flash Player 6.

### Uso

*myButton.onRelease*

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Manipulador de eventos; chamado na liberação de um botão.

É necessário definir uma função que seja executada quando o evento é chamado.

### Exemplo

O exemplo a seguir define uma função para o método `onRelease` que envia uma ação `trace` para a janela Saída.

```
myButton.onRelease = function () {  
    trace ("onRelease chamado");  
};
```

## Button.onReleaseOutside

### Disponibilidade

Flash Player 6.

### Uso

*myButton.onReleaseOutside*

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Manipulador de eventos; chamado na liberação do mouse enquanto o ponteiro estiver fora do botão depois de pressionar o botão enquanto o ponteiro estiver dentro do botão.

É necessário definir uma função que seja executada quando o evento é chamado.

### Exemplo

O exemplo a seguir define uma função para o método `onReleaseOutside` que envia uma ação `trace` à janela Saída.

```
myButton.onReleaseOutside = function () {  
    trace ("onReleaseOutside chamado");  
};
```

## Button.onRollOut

### Disponibilidade

Flash Player 6.

### Uso

*myButton.onRollOut*

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Manipulador de eventos; chamado quando o ponteiro rolar para fora da área de um botão.

É necessário definir uma função que seja executada quando o evento é chamado.

### Exemplo

O exemplo a seguir define uma função para o método `onRollOut` que envia uma ação `trace` à janela Saída.

```
myButton.onRollOut = function () {  
    trace ("onRollOut chamado");  
};
```

## Button.onRollOver

### Disponibilidade

Flash Player 6.

### Uso

*myButton.onRollOver*

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Identificador de eventos; chamado na passagem do ponteiro sobre a área de um botão.

É necessário definir uma função que seja executada quando o evento é chamado.

### Exemplo

O exemplo a seguir define uma função para o método `onRollOver` que envia uma ação `trace` à janela Saída.

```
myButton.onRollOver = function () {  
    trace ("onRollOver chamado");  
};
```

## Button.onSetFocus

### Disponibilidade

Flash Player 6.

### Uso

```
myButton.onSetFocus = function(oldFocus){  
    comandos;  
};
```

### Parâmetros

*oldFocus* O objeto a perder o foco de teclado.

### Retorna

Nada.

### Descrição

Manipulador de eventos; chamado quando um botão recebe foco de teclado. O parâmetro *oldFocus* é o objeto que perde o foco. Por exemplo, se o usuário pressionar a tecla Tab para mover o foco de entrada de um campo de texto para um botão, o parâmetro *oldFocus* conterá a instância do campo de texto.

Se nenhum objeto possuía o foco anteriormente, *oldFocus* conterá um valor `null`.

## Button.\_parent

### Disponibilidade

Flash Player 6.

### Uso

```
_parent.property
```

### Descrição

Propriedade; especifica ou retorna uma referência ao clipe de filme ou objeto que contém o clipe de filme ou objeto atual. O objeto atual é aquele que contém o código do ActionScript que faz referência a `_parent`. Use `_parent` para especificar um caminho relativo para clipes de filme ou objetos que estiverem acima do clipe de filme ou objeto atual. É possível usar `_parent` para subir vários níveis na lista de exibição, conforme mostrado a seguir:

```
_parent._parent._alpha = 20;
```

### Consulte também

`_root`, `targetPath`

## Button.\_quality

### Disponibilidade

Flash Player 6.

### Uso

```
myButton._quality
```

### Descrição

Propriedade (global); define ou recupera a qualidade usada para um filme. As fontes de dispositivo são sempre serrilhadas, sendo assim não são afetadas pela propriedade `_quality`.

A propriedade `_quality` pode ser definida nos seguintes valores:

- "LOW" Qualidade baixa. Os gráficos não são apresentados sem serrilhado, os bitmaps não são suavizados.
- "MEDIUM" Qualidade média. Os gráficos são apresentados sem serrilhado usando uma grade de pixels 2 x 2, 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 de pixels 4 x 4 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 de pixels 4 x 4 e os bitmaps sempre são suavizados.

#### **Exemplo**

O exemplo a seguir define a qualidade como LOW:

```
myButton._quality = "LOW";
```

#### **Consulte também**

`_highquality`, `toggleHighQuality`

## **Button.\_rotation**

#### **Disponibilidade**

Flash Player 6.

#### **Uso**

```
myButton._rotation
```

#### **Descrição**

Propriedade; especifica a rotação do botão em graus.

## **Button.\_soundbuftime**

#### **Disponibilidade**

Flash Player 6.

#### **Uso**

```
myButton._soundbuftime
```

#### **Descrição**

Propriedade (global); um inteiro que especifica o número de segundos em que um som é armazenado em pré-buffer antes de começar a fluir.

## Button.tabEnabled

### Disponibilidade

Flash Player 6.

### Uso

`myButton.tabEnabled`

### Descrição

Propriedade; pode ser definida em uma instância dos objetos MovieClip, Button ou TextField. Por padrão, não é definido.

Se a propriedade `tabEnabled` for `undefined` ou `true`, o objeto será incluído na ordenação automática de guias. Se a propriedade `tabIndex` também estiver definida com um valor, o objeto estará incluído em uma ordenação de guia personalizada. Se `tabEnabled` for `false`, o objeto não será incluído na ordenação automática de guias. No caso de um clipe de filme, se `tabEnabled` for `false`, os filhos do clipe de filme ainda poderão ser incluídos na ordenação automática de guias, a menos que a propriedade `tabChildren` também seja definida como `false`.

Se `tabEnabled` for `undefined` ou `true`, e se a propriedade `tabIndex` for definida, então o objeto será incluído na ordenação de guia personalizada. Se `tabEnabled` for `false`, então o objeto não será incluído na ordenação de guia personalizada, ainda que a propriedade `tabIndex` seja definida. Se `tabEnabled` for definido como `false` em um clipe de filme, os filhos do clipe de filme ainda poderão ser incluídos na ordenação de guia personalizada.

### Consulte também

`Button.tabIndex`

## Button.tabIndex

### Disponibilidade

Flash Player 6.

### Uso

`myButton.tabIndex`

### Descrição

Propriedade; permite personalizar a ordenação de guias dos objetos em um filme. É possível definir a propriedade `tabIndex` em um botão, clipe de filme ou instância de campo de texto. Por padrão, ela é `undefined`.

Se algum objeto sendo exibido atualmente no filme do Flash tiver uma propriedade `tabIndex`, a ordenação de guia automática será desativada e a ordenação de guia será calculada nas propriedades `tabIndex` de objetos do filme. A ordenação personalizada de guias inclui apenas os objetos que têm propriedades `tabIndex`.

A propriedade `tabIndex` pode ser um inteiro não negativo. Os objetos são ordenados de acordo com suas propriedades `tabIndex`, em ordem ascendente. Um objeto com um `tabIndex` 1 vem antes de um objeto com `tabIndex` 2. Se dois objetos tiverem o mesmo `tabIndex`, aquele que vier antes do outro na ordenação de guia será `undefined`.

A ordenação de guia personalizada definida pela propriedade `tabIndex` é *flat*. Isso significa que as relações hierárquicas de objetos são ignoradas no filme do Flash. Todos os objetos no filme do Flash com propriedades `tabIndex` são colocados na ordem de guia. Por sua vez, essa é determinada pela ordem dos valores de `tabIndex`. Se dois objetos têm o mesmo valor de `tabIndex`, aquele que vier primeiro será `undefined`. Você não deve usar o mesmo valor de `tabIndex` para vários objetos.

## Button.\_target

### Disponibilidade

Flash Player 6.

### Uso

*myButton.\_target*

### Descrição

Propriedade (somente leitura); retorna o caminho de destino da instância de botão especificada no parâmetro *Botão*.

## Button.trackAsMenu

### Disponibilidade

Flash Player 6.

### Uso

*myButton.trackAsMenu*

### Descrição

Propriedade; uma propriedade booleana que indica se outros botões ou cliques de filme podem ou não receber eventos de liberação de mouse. Permite a criação de menus. Você pode definir a propriedade *trackAsMenu* em qualquer botão ou objeto de clipe de filme. Se a propriedade *trackAsMenu* não existir, o comportamento padrão será *false*.

É possível alterar a qualquer momento a propriedade *trackAsMenu*; o botão modificado assume o novo comportamento logo em seguida.

### Consulte também

*MovieClip.trackAsMenu*

## Button.\_url

### Disponibilidade

Flash Player 6.

### Uso

*myButton.\_url*

### Descrição

Propriedade (somente leitura); recupera o URL do arquivo SWF que criou o botão.

## Button.useHandCursor

### Disponibilidade

Flash Player 6.

### Uso

*myButton.useHandCursor*

### Descrição

Propriedade; um valor booleano que, ao ser definido como `true`, indica que um cursor não será exibido quando o usuário passar o cursor sobre um botão. O valor padrão de `useHandCursor` é `true`. Se a propriedade `useHandCursor` for definida como `false`, então será usado o cursor de seta no lugar.

Se a qualquer momento a propriedade `useHandCursor` for alterada; o botão modificado logo assumirá o comportamento do novo cursor. A propriedade `useHandCursor` pode ser lida de um objeto de protótipo.

## Button.\_visible

### Disponibilidade

Flash Player 6.

### Uso

```
myButton._visible
```

### Descrição

Propriedade; um valor booleano que indica se o botão especificado pelo parâmetro *Botão* é visível. Os botões invisíveis (propriedade `_visible` definida como `false`) são desativados.

## Button.\_width

### Disponibilidade

Flash Player 6.

### Uso

```
myButton._width
```

### Descrição

Propriedade; define e recupera a largura do botão, em pixels.

### Exemplo

O exemplo a seguir define as propriedades de altura e de largura de um botão.

```
myButton._width=200;  
myButton._height=200;
```

### Consulte também

`MovieClip._width`



## Button.\_x

### Disponibilidade

Flash Player 6.

### Uso

*myButton.\_x*

### Descrição

Propriedade; um inteiro que define a coordenada  $x$  de um botão em relação às coordenadas locais do clipe de filme pai. Se um botão estiver na Linha de tempo principal, seu sistema de coordenadas refere-se ao canto superior esquerdo do Palco como (0, 0). Se o botão estiver dentro de outro clipe de filme que tenha transformações, o botão está no sistema de coordenadas local do clipe de filme anexado. Assim, para que um clipe de filme gire 90° no sentido anti-horário, o botão anexado herda um sistema de coordenadas que é girado 90° no sentido anti-horário. As coordenadas do botão referem-se à posição do ponto do registro.

### Consulte também

*Button.\_xscale*, *Button.\_y*, *Button.\_yscale*

## Button.\_xmouse

### Disponibilidade

Flash Player 6.

### Uso

*myButton.\_xmouse*

### Descrição

Propriedade (somente leitura); retorna a coordenada  $x$  da posição do mouse em relação ao botão.

### Consulte também

*Button.\_ymouse*

## Button.\_xscale

### Disponibilidade

Flash Player 6.

### Uso

*myButton.\_xscale*

### Descrição

Propriedade; determina o dimensionamento horizontal (*porcentagem*) do botão conforme aplicado do ponto do registro do botão. 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 for dimensionado em 50%, definir a propriedade *\_x* move um objeto no botão pela metade do número de pixels, como se o filme tivesse sido dimensionado em 100%.

### Consulte também

*Button.\_x*, *Button.\_y*, *Button.\_yscale*

## Button.\_y

### Disponibilidade

Flash Player 6.

### Uso

*myButton.\_y*

### Descrição

Propriedade; define a coordenada *y* do botão em relação às coordenadas locais do clipe de filme pai. Se um botão estiver na Linha de tempo principal, seu sistema de coordenadas irá referir-se ao canto superior esquerdo do Palco como (0, 0). Se o botão estiver dentro de outro clipe de filme que tem transformações, o botão está no sistema de coordenadas local do clipe de filme anexado. Assim, para que um clipe de filme gire 90° no sentido anti-horário, o botão anexado herda um sistema de coordenadas que é girado 90° no sentido anti-horário. As coordenadas do botão referem-se à posição do ponto do registro.

### Consulte também

Button.\_x, Button.\_xscale, Button.\_yscale

## Button.\_ymouse

### Disponibilidade

Flash Player 6.

### Uso

*myButton.\_ymouse*

### Descrição

Propriedade (somente leitura); indica a coordenada *y* da posição do mouse em relação ao botão.

### Consulte também

Button.\_xmouse

## Button.\_yscale

### Disponibilidade

Flash Player 6.

### Uso

*myButton.\_yscale*

### Descrição

Propriedade; define a escala vertical (*porcentagem*) do botão conforme aplicado do ponto de registro do botão. O ponto de registro padrão é (0,0).

### Consulte também

Button.\_y, Button.\_x, Button.\_xscale

## call

### Disponibilidade

Flash Player 4. Esta ação tornou-se obsoleta no Flash 5. Recomenda-se o uso da ação `function` em seu lugar.

### Uso

`call(quadro)`

### Parâmetros

*quadro* O rótulo ou número de um quadro na Linha de tempo.

### Retorna

Nada.

### Descrição

Ação; executa o script no quadro chamado, sem mover a reprodução para esse quadro. Não haverá variáveis locais, uma vez que a execução do script é concluída.

### Consulte também

`function`

## chamar função

### Disponibilidade

Flash Player 6

### Uso

`objeto.função([parâmetros])`

### Parâmetros

*objeto* Um objeto (pode ser um clipe de filme) em que a função foi definida.

*função* Um identificador que especifica uma função definida pelo usuário.

*parâmetros* Um parâmetro opcional que indica um parâmetro necessário à função.

### Retorna

Nada.

### Descrição

Ação; permite o uso de campos de parâmetros para chamar uma função definida pelo usuário no modo Normal no painel Ações.

## case

### Disponibilidade

Flash Player 4.

### Uso

*expressão case: comandos*

### Parâmetros

*expressão* Qualquer expressão.

*comandos* Qualquer comando.

**Retorna**

Nada.

**Descrição**

Palavra-chave; define uma condição para a ação `switch`. Os comandos no parâmetro *comandos* são executados se o parâmetro *expressão* subsequente à palavra-chave `case` for igual ao parâmetro *expressão* da ação `switch` que usa a igualdade estrita (`===`)

O uso da ação `case` fora de um comando `switch` apresenta erro e o script não é compilado.

**Consulte também**

`switch`, `default`, `break`, `===` (igualdade estrita)

## chr

**Disponibilidade**

Flash Player 4. Esta função foi reprovada no Flash 5 e substituída pelo método `String.fromCharCode`.

**Uso**

`chr(número)`

**Parâmetros**

*número* Um número de código ASCII.

**Retorna**

Nada.

**Descrição**

Função de sequência de caracteres; converte código ASCII em caracteres.

**Exemplo**

O exemplo a seguir converte o número 65 na letra *A* e o atribui à variável `myVar`.

```
myVar = chr(65);
```

**Consulte também**

`String.fromCharCode`

## clearInterval

**Disponibilidade**

Flash Player 6.

**Uso**

`clearInterval( intervalID )`

**Parâmetros**

*intervalID* Um objeto retornado de uma chamada à função `setInterval`.

**Retorna**

Nada.

**Descrição**

Ação; limpa a chamada da função `setInterval`.

### Exemplo

O exemplo a seguir em primeiro lugar define uma chamada de intervalo, depois limpa-a:

```
function callback() {  
    trace("interval chamado");  
}  
var intervalID;  
intervalID = setInterval( callback, 1000 );  
  
// depois de algum tempo  
clearInterval( intervalID );
```

### Consulte também

`setInterval`

## Color (objeto)

O objeto `Color` define o valor de cor RGB e a transformação de cor de cliques de filmes e recupera esses valores depois da definição.

É necessário usar o construtor `new Color()` para criar uma instância do objeto `Color` antes de chamar os métodos.

Somente o Flash 5 e as versões mais recentes do Flash Player oferecem suporte ao objeto `Color`.

### Resumo de métodos do objeto Color

Método	Descrição
<code>Color.getRGB</code>	Retorna o valor RGB definido pela última chamada <code>setRGB</code> .
<code>Color.getTransform</code>	Retorna a informação de transformação definida pela última chamada <code>setTransform</code> .
<code>Color.setRGB</code>	Define a representação hexadecimal do valor RGB de um objeto <code>Color</code> .
<code>Color.setTransform</code>	Define a transformação de cor de um objeto <code>Color</code> .

### Construtor do objeto Color

#### Disponibilidade

Flash Player 5.

#### Uso

```
new Color(destino);
```

#### Parâmetros

*destino* O nome da instância de um clipe de filme.

#### Retorna

Nada.

#### Descrição

Construtor; cria uma instância do objeto `Color` para o clipe de filme especificado pelo parâmetro *destino*. Depois será possível usar os métodos desse objeto `Color` para alterar a cor de todo o clipe de filme de destino.

### Exemplo

O exemplo a seguir cria uma instância do objeto `Color` denominado `myColor` para o clipe de filme `myMovieClip` e define seu valor RGB:

```
myColor = new Color(myMovieClip);  
myColor.setRGB(0xff9933);
```

## Color.getRGB

### Disponibilidade

Flash Player 5.

### Uso

```
myColor.getRGB()
```

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Método; retorna os valores numéricos definidos pela última chamada `setRGB`.

### Exemplo

O código a seguir recupera o valor RGB da instância `myColor` do objeto `Color`, converte-o em uma sequência de caracteres hexadecimal e a atribui à variável `value`.

```
value = myColor.getRGB().toString(16);
```

### Consulte também

`Color.setRGB`

## Color.getTransform

### Disponibilidade

Flash Player 5.

### Uso

```
myColor.getTransform()
```

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Método; retorna os valores de transformação definidos pela última chamada `setTransform`.

### Consulte também

`Color.setTransform`

## Color.setRGB

### Disponibilidade

Flash Player 5.

### Uso

```
myColor.setRGB(0xRRGGBB)
```

### Parâmetros

*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. A sequência *0x* informa ao compilador do ActionScript que o número é um valor hexadecimal.

### Descrição

Método; especifica uma cor RGB para uma instância do objeto Color. Quando este método é chamado, todas as definições anteriores são substituídas pelo método `setTransform`.

### Retorna

Nada.

### Exemplo

Este exemplo define o valor de cor RGB do clipe de filme `myMovie`. Para observar o funcionamento deste código, coloque um clipe de filme no Palco com o nome da instância, `myMovie`. Depois, coloque o código a seguir no Quadro 1 na Linha de tempo principal e escolha Controlar > Testar filme.

```
myColor = new Color(myMovie);  
myColor.setRGB(0x993366);
```

### Consulte também

`Color.setTransform`

## Color.setTransform

### Disponibilidade

Flash Player 5.

### Uso

```
myColor.setTransform(objetoTransformCor);
```

### Parâmetros

*colorTransformObject* Um objeto criado com o construtor `new Object`. Esta instância do objeto `Object` deve ter as seguintes propriedades que especificam valores de transformação de cor: *ra*, *rb*, *ga*, *gb*, *ba*, *bb*, *aa*, *ab*. Essas propriedades são explicadas a seguir.

### Retorna

Nada.

### Descrição

Método; define as informações de transformação de cor para uma instância do objeto Color. O parâmetro *colorTransformObject* é um objeto genérico criado a partir do construtor `new Object`. Ele possui parâmetros que especificam os valores de porcentagem e de deslocamento dos componentes vermelho, verde, azul e alfa (transparência) de uma cor, inseridos no formato *0xRRGGBBAA*.

Os parâmetros de um objeto de transformação de cor correspondem às configurações na caixa de diálogo Efeito avançado e são definidos da seguinte forma:

- *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).

Crie um parâmetro *colorTransformObject* 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;
```

Além disso, é possível usar a seguinte sintaxe para criar um parâmetro *colorTransformObject*:

```
myColorTransform = { ra: '50', rb: '244', ga: '40', gb: '112', ba: '12', bb:
    '90', aa: '40', ab: '70' }
```

### Exemplo

Este exemplo cria uma nova instância do objeto Color para um filme de destino. Além disso, cria um objeto genérico denominado *myColorTransform* com as propriedades definidas acima e usa o método *setTransform* para passar o *colorTransformObject* para um objeto Color. Para usar esse código em um documento Flash (FLA), coloque-o no Quadro 1 na Linha de tempo principal e coloque um clipe de filme no Palco com o nome de instância *myMovie*, como no código a seguir:

```
//Crie um objeto de cor chamado myColor para o destino myMovie
myColor = new Color(myMovie);
// Crie um objeto de transformação de cor chamado myColorTransform usando
// o objeto genérico Object
myColorTransform = new Object();
// Defina os valores de myColorTransform
myColorTransform = { ra: '50', rb: '244', ga: '40', gb: '112', ba: '12', bb:
    '90', aa: '40', ab: '70' };
// Associe o objeto de transformação de cor ao objeto Color
// criado para myMovie
myColor.setTransform(myColorTransform);
```



## continue

### Disponibilidade

Flash Player 4.

### Uso

`continue`

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Ação; aparece em vários tipos de comandos de loop; em cada um dos tipos tem um comportamento diferente.

Em um loop `while`, `continue` faz o interpretador do 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 interpretador do 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 interpretador do 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 interpretador do Flash ignorar o resto do corpo do loop e voltar ao início do loop, onde o próximo valor na enumeração é processado.

### Consulte também

`do while`, `for`, `for...in`, `while`

## CustomActions (objeto)

Os métodos do objeto `CustomActions` permitem que um filme do Flash seja executado na ferramenta de criação do Flash para gerenciar quaisquer ações personalizadas registradas na ferramenta de criação. Um filme do Flash pode instalar e desinstalar ações personalizadas, recuperar a definição XML de uma ação personalizada e recuperar a lista de ações personalizadas registradas.

Use esses métodos para montar filmes do Flash que sejam extensões da ferramenta de criação do Flash. Um filme de extensão como esse poderia, por exemplo, usar o protocolo do aplicativo Flash para navegar em um repositório UDDI e fazer download de serviços da Web na caixa de ferramentas Ações.

## Resumo de métodos do objeto CustomActions

Método	Descrição
<code>CustomActions.get</code>	Lê o conteúdo de um arquivo de definição XML de ações personalizadas.
<code>CustomActions.install</code>	Instala um novo arquivo de definição XML de ações personalizadas.
<code>CustomActions.list</code>	Retorna uma lista de todas as ações personalizadas registradas.
<code>CustomActions.uninstall</code>	Remove um arquivo de definição XML de ações personalizadas.

## CustomActions.get

### Disponibilidade

Flash Player 6.

### Uso

```
CustomActions.get(customActionsName)
```

### Parâmetros

*customActionsName*    O nome da definição de ações personalizadas a ser recuperada.

### Retorna

Nada.

### Descrição

Método; lê o conteúdo do arquivo de definição XML de ações personalizadas denominado *customActionsName*.

O nome do arquivo de definições deve ser simples, sem a extensão de arquivo .xml e sem qualquer separador de diretório (':', '/' ou '\').

Se o arquivo de definições especificado por *customActionsName* não puder ser encontrado, será retornado o valor `undefined`. Se a definição XML de ações personalizadas especificada pelo parâmetro *customActionsName* for localizada, ela será lida por inteiro e retornada como uma sequência de caracteres.

## CustomActions.install

### Disponibilidade

Flash Player 6.

### Uso

```
CustomActions.install(customActionsName, customXMLDefinition)
```

### Parâmetros

*customActionsName*    O nome da definição de ações personalizadas a ser instalada.

*customXMLDefinition*    O texto da definição XML a ser instalada.

### Retorna

Nada.

### Descrição

Método; instala um novo arquivo de definição XML de ações personalizadas indicado pelo parâmetro *customActionsName*. O conteúdo do arquivo é especificado pela sequência de caracteres *customXMLDefinition*.

O nome do arquivo de definições deve ser simples, sem a extensão de arquivo .xml e sem qualquer separador de diretório (':', '/' ou '\').

Se já existir um arquivo de ações personalizadas com o nome *customActionsName*, ele será substituído.

Se ocorrer um erro durante a instalação, será retornado o valor `false`; caso contrário, será retornado o valor `true` para indicar que a ação personalizada foi instalada com êxito.

Se o diretório Configuração/ActionsPanel/CustomActions for inexistente quando esse método for chamado, então o diretório será criado.

## CustomActions.list

### Disponibilidade

Flash Player 6.

### Uso

```
CustomActions.list()
```

### Parâmetros

Nenhum.

### Retorna

Uma matriz.

### Descrição

Método; retorna um objeto Array que contém os nomes de todas as ações personalizadas registradas na ferramenta de criação do Flash. Os elementos da matriz são nomes simples, sem a extensão de arquivo .xml e sem qualquer separador de diretório ( por exemplo, “:”, “/” ou “\”). Se não houver ações personalizadas registradas, o método `list` retornará uma matriz de tamanho zero. Se ocorrer um erro, o método `list` retornará o valor `undefined`.

## CustomActions.uninstall

### Disponibilidade

Flash Player 6.

### Uso

```
CustomActions.uninstall(customActionsName)
```

### Parâmetros

*customActionsName*    O nome da definição de ações personalizadas a ser desinstalada.

### Retorna

Nada.

### Descrição

Método; remove o arquivo de definição XML de ações personalizadas denominado *customActionsName*.

O nome do arquivo de definições deve ser simples, sem a extensão de arquivo .xml e sem qualquer separador de diretório (':', '/' ou '\').

Se não forem localizadas ações personalizadas com o nome `customActionsName`, será retornado o valor `false`. Se a remoção das ações personalizadas for bem sucedida, será retornado o valor `true`.

## Date (objeto)

O objeto `Date` permite a recuperação dos 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. Os métodos do objeto `Date` não são estáticos, mas se aplicam somente à instância individual do objeto `Date` especificado quando o método é chamado. O método `Date.UTC` é uma exceção, ele é estático.

O objeto `Date` trata do horário de verão de modo diferente, em função do sistema operacional em uso, e da versão do Flash Player. O Flash Player 6 trata do horário de verão nestes sistemas operacionais das seguintes maneiras:

- **Windows**—a saída do objeto `Date` é ajustada automaticamente para o horário de verão. O objeto `Date` detecta se o horário de verão está em vigor no local atual. Em caso positivo, ele detecta a data e hora da transição do horário padrão para o horário de verão. Contudo, as datas de mudança atualmente em vigor são aplicadas a datas passadas e futuras, sendo assim, as diferenças de horário de verão podem ser calculadas de forma errada em datas passadas quando o local tiver datas de mudança diferentes.
- **Mac OS 8 e 9**—o objeto `Date` usa a diferença relativa ao horário de verão atual, independentemente da data ou hora que está sendo calculada. Por exemplo, nos EUA, no fuso horário do Pacífico em agosto, quando o horário de verão (DST) está em vigor, um objeto `Date` que tenha a data 1º jan. 2001 ainda informará o horário de verão, embora ele não esteja mais em vigor no mês de janeiro. Este problema não pode ser solucionado no Mac OS 8 ou 9, pois o banco de dados com as informações de fuso horário não está disponível.
- **Mac OS X**—o objeto `Date` ajusta automaticamente a saída para horário de verão. O banco de dados com as informações de fuso horário no Mac OS X é usado para determinar se a diferença de horário de verão deve ser aplicada a alguma data ou hora atual ou passada.

O Flash Player 5 gerencia o horário de verão nestes sistemas operacionais da seguinte maneira:

- **Mac OS 8 e 9**—o comportamento é o mesmo descrito para o Flash Player 6.
- **Windows**—as regras norte-americanas de horário de verão sempre são aplicadas, o que leva a mudanças incorretas nos países da Europa e de outras regiões que adotam o horário de verão mas com horas de transição diferentes dos Estados Unidos. O Flash detecta corretamente se o horário de verão está em vigor no local atual.

Para chamar os métodos do objeto `Date`, crie primeiro uma instância do objeto `Date` com o construtor adequado.

O objeto `Date` requer o Flash Player 5.

## Resumo de métodos do objeto `Date`

Método	Descrição
<code>Date.getDate</code>	Retorna o dia do mês de acordo com a hora local.
<code>Date.getDay</code>	Retorna o dia da semana de acordo com a hora local.
<code>Date.getFullYear</code>	Retorna o ano com quatro dígitos de acordo com a hora local.
<code>Date.getHours</code>	Retorna a hora de acordo com a hora local.
<code>Date.getMilliseconds</code>	Retorna os milissegundos de acordo com a hora local.
<code>Date.getMinutes</code>	Retorna os minutos de acordo com a hora local.
<code>Date.getMonth</code>	Retorna o mês de acordo com a hora local.
<code>Date.getSeconds</code>	Retorna os segundos de acordo com a hora local.
<code>Date.getTime</code>	Retorna o número de milissegundos desde a meia-noite de 1º de janeiro de 1970, hora universal.
<code>Date.getTimezoneOffset</code>	Retorna a diferença, em minutos, entre o a hora local do computador e a hora universal.
<code>Date.getUTCDate</code>	Retorna o dia (data) do mês de acordo com a hora universal.

<b>Método</b>	<b>Descrição</b>
<code>Date.getUTCDay</code>	Retorna o dia da semana de acordo com a hora universal.
<code>Date.getUTCFullYear</code>	Retorna o ano com quatro dígitos de acordo com a hora universal.
<code>Date.getUTCHours</code>	Retorna a hora de acordo com a hora universal.
<code>Date.getUTCMilliseconds</code>	Retorna os milissegundos de acordo com a hora universal.
<code>Date.getUTCMinutes</code>	Retorna os minutos de acordo com a hora universal.
<code>Date.getUTCMonth</code>	Retorna o mês de acordo com a hora universal.
<code>Date.getUTCSeconds</code>	Retorna os segundos de acordo com a hora universal.
<code>Date.getYear</code>	Retorna o ano de acordo com a hora local.
<code>Date.setDate</code>	Define o dia do mês de acordo com a hora local. Retorna a nova hora em milissegundos.
<code>Date.setFullYear</code>	Define o ano completo de acordo com a hora local. Retorna a nova hora em milissegundos.
<code>Date.setHours</code>	Define a hora de acordo com a hora local. Retorna a nova hora em milissegundos.
<code>Date.setMilliseconds</code>	Define os milissegundos de acordo com a hora local. Retorna a nova hora em milissegundos.
<code>Date.setMinutes</code>	Define os minutos de acordo com a hora local. Retorna a nova hora em milissegundos.
<code>Date.setMonth</code>	Define o mês de acordo com a hora local. Retorna a nova hora em milissegundos.
<code>Date.setSeconds</code>	Define os segundos de acordo com a hora local. Retorna a nova hora em milissegundos.
<code>Date.setTime</code>	Define a data em milissegundos. Retorna a nova hora em milissegundos.
<code>Date.setUTCDate</code>	Define a data de acordo com a hora universal. Retorna a nova hora em milissegundos.
<code>Date.setUTCFullYear</code>	Define o ano de acordo com a hora universal. Retorna a nova hora em milissegundos.
<code>Date.setUTCHours</code>	Define a hora de acordo com a hora universal. Retorna a nova hora em milissegundos.
<code>Date.setUTCMilliseconds</code>	Define os milissegundos de acordo com a hora universal. Retorna a nova hora em milissegundos.
<code>Date.setUTCMinutes</code>	Define os minutos de acordo com a hora universal. Retorna a nova hora em milissegundos.
<code>Date.setUTCMonth</code>	Define o mês de acordo com a hora universal. Retorna a nova hora em milissegundos.
<code>Date.setUTCSeconds</code>	Define os segundos de acordo com a hora universal. Retorna a nova hora em milissegundos.
<code>Date.setYear</code>	Define o ano de acordo com a hora local.
<code>Date.toString</code>	Retorna um valor de seqüência de caracteres representando a data e a hora armazenadas no objeto Date especificado.
<code>Date.UTC</code>	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

### Disponibilidade

Flash Player 5.

### Uso

```
new Date()
```

```
new Date(ano [, mês [, data [, hora [, minutos [, segundos [, milissegundos  
]]]])
```

### Parâmetros

*ano* Um valor de 0 a 99 indica 1900 a 1999; caso contrário, especifique todos os 4 dígitos do ano.

*mês* Um inteiro entre 0 (janeiro) e 11 (dezembro).

*data* Um inteiro de 1 a 31. Este parâmetro é opcional.

*hora* Um inteiro entre 0 (meia-noite) e 23 (11 p.m.).

*minutos* Um inteiro de 0 a 59. Este parâmetro é opcional.

*segundos* Um inteiro de 0 a 59. Este parâmetro é opcional.

*milissegundos* Um inteiro de 0 a 999. Este parâmetro é opcional.

### Retorna

Um inteiro.

### Descrição

Objeto; constrói um objeto new Date que mantém a data e hora atuais ou a data especificada.

### 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() + 1) + "/" + myDate.getDate() + "/" +  
myDate.getFullYear());
```

## Date.getDate

### Disponibilidade

Flash Player 5.

### Uso

```
myDate.getDate()
```

### Parâmetros

Nenhum.

**Retorna**

Um inteiro.

**Descrição**

Método; retorna o dia do mês (um inteiro de 1 a 31) 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.

## Date.getDay

**Disponibilidade**

Flash Player 5.

**Uso**

```
myDate.getDay()
```

**Parâmetros**

Nenhum.

**Retorna**

Um inteiro.

**Descrição**

Método; retorna o dia da semana (0 para domingo, 1 para segunda-feira, 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.

## Date.getFullYear

**Disponibilidade**

Flash Player 5.

**Uso**

```
myDate.getFullYear()
```

**Parâmetros**

Nenhum.

**Retorna**

Um inteiro.

**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.

**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

### Disponibilidade

Flash Player 5.

### Uso

```
myDate.getHours()
```

### Parâmetros

Nenhum.

### Retorna

Um inteiro.

### 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.

## Date.getMilliseconds

### Disponibilidade

Flash Player 5.

### Uso

```
myDate.getMilliseconds()
```

### Parâmetros

Nenhum.

### Retorna

Um inteiro.

### 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.

## Date.getMinutes

### Disponibilidade

Flash Player 5.

### Uso

```
myDate.getMinutes()
```

### Parâmetros

Nenhum.

### Retorna

Um inteiro.

### 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.



## Date.getMonth

### Disponibilidade

Flash Player 5.

### Uso

```
myDate.getMonth()
```

### Parâmetros

Nenhum.

### Retorna

Um inteiro.

### 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.

## Date.getSeconds

### Disponibilidade

Flash Player 5.

### Uso

```
myDate.getSeconds()
```

### Parâmetros

Nenhum.

### Retorna

Um inteiro.

### 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.

## Date.getTime

### Disponibilidade

Flash Player 5.

### Uso

```
myDate.getTime()
```

### Parâmetros

Nenhum.

### Retorna

Um inteiro.

### Descrição

Método; retorna o número de milissegundos 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 quando comparar dois ou mais objetos Date.

## Date.getTimezoneOffset

### Disponibilidade

Flash Player 5.

### Uso

```
mydate.getTimezoneOffset()
```

### Parâmetros

Nenhum.

### Retorna

Um inteiro.

### Descrição

Método; retorna a diferença, em minutos, entre a hora local do computador e a hora universal.

### Exemplo

O exemplo a seguir retorna a diferença entre o horário de verão de São Francisco e a hora universal. O horário de verão será incluído no resultado apresentado somente se a data definida no objeto Date constar do período do horário de verão.

```
trace(new Date().getTimezoneOffset());  
// 420 é exibido na janela Saída  
// (7 horas * 60 minutos/hora = 420 minutos)  
// Este é um exemplo de horário de verão do Pacífico (PDT (Pacific  
// Daylight Time) GMT-0700).  
// O resultado varia em função do local e da hora do ano.
```

## Date.getUTCDate

### Disponibilidade

Flash Player 5.

### Uso

```
myDate.getUTCDate()
```

### Parâmetros

Nenhum.

### Retorna

Um inteiro.

### Descrição

Método; retorna o dia (data) do mês do objeto Date especificado, de acordo com a hora universal.

## Date.getUTCDay

### Disponibilidade

Flash Player 5.

### Uso

*myDate*.getUTCDate()

### Parâmetros

Nenhum.

### Retorna

Um inteiro.

### Descrição

Método; retorna o dia da semana do objeto Date especificado, de acordo com a hora universal.

## Date.getUTCFullYear

### Disponibilidade

Flash Player 5.

### Uso

*myDate*.getUTCFullYear()

### Parâmetros

Nenhum.

### Retorna

Um inteiro.

### Descrição

Método; retorna o ano com quatro dígitos do objeto Date especificado, de acordo com a hora universal.

## Date.getUTCHours

### Disponibilidade

Flash Player 5.

### Uso

*myDate*.getUTCHours()

### Parâmetros

Nenhum.

### Retorna

Um inteiro.

### Descrição

Método; retorna a hora do objeto Date especificado, de acordo com a hora universal.

## Date.getUTCMilliseconds

### Disponibilidade

Flash Player 5.

### Uso

```
myDate.getUTCMilliseconds()
```

### Parâmetros

Nenhum.

### Retorna

Um inteiro.

### Descrição

Método; retorna os milissegundos do objeto Date especificado, de acordo com a hora universal.

## Date.getUTCMinutes

### Disponibilidade

Flash Player 5.

### Uso

```
myDate.getUTCMinutes()
```

### Parâmetros

Nenhum.

### Retorna

Um inteiro.

### Descrição

Método; retorna os minutos do objeto Date especificado, de acordo com a hora universal.

## Date.getUTCMonth

### Disponibilidade

Flash Player 5.

### Uso

```
myDate.getUTCMonth()
```

### Parâmetros

Nenhum.

### Retorna

Um inteiro.

### Descrição

Método; retorna o mês do objeto Date especificado, de acordo com a hora universal.

## Date.getUTCSeconds

### Disponibilidade

Flash Player 5.

### Uso

```
myDate.getUTCSeconds()
```

### Parâmetros

Nenhum.

### Retorna

Um inteiro.

### Descrição

Método; retorna os segundos do objeto Date especificado, de acordo com a hora universal.

## Date.getYear

### Disponibilidade

Flash Player 5.

### Uso

```
myDate.getFullYear()
```

### Parâmetros

Nenhum.

### Retorna

Um inteiro.

### 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 completo menos 1900. Por exemplo, o ano 2000 é representado como 100.

## Date.setDate

### Disponibilidade

Flash Player 5.

### Uso

```
myDate.setDate(data)
```

### Parâmetros

*data* Um inteiro de 1 a 31.

### Retorna

Um inteiro.

### Descrição

Método; define o dia do mês do objeto Date especificado, de acordo com a hora local, e retorna a nova hora em milissegundos. A hora local é determinada pelo sistema operacional em que o Flash Player está sendo executado.

## Date.setFullYear

### Disponibilidade

Flash Player 5.

### Uso

```
myDate.setFullYear(ano [, mês [, data]] )
```

### Parâmetros

*ano* Um número de quatro dígitos que especifica 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). Este parâmetro é opcional.

*data* Um número de 1 a 31. Este parâmetro é opcional.

### Retorna

Um inteiro.

### Descrição

Método; define o ano do objeto Date especificado, de acordo com a hora local, e retorna a nova hora em milissegundos. Se os parâmetros *mês* e *data* forem especificados, esses também serão definidos para a hora local. A hora local é determinada pelo sistema operacional em que o Flash Player está sendo executado.

Quando este método é chamado, os outros campos do objeto Date especificado não são modificados, mas, se o dia da semana for alterado em decorrência do chamamento a esse método, os métodos `getUTCDay` e `getDay` podem reportar um novo valor.

## Date.setHours

### Disponibilidade

Flash Player 5.

### Uso

```
myDate.setHours(hora)
```

### Parâmetros

*hora* Um inteiro entre 0 (meia-noite) e 23 (11 p.m.).

### Retorna

Um inteiro.

### Descrição

Método; define as horas do objeto Date especificado, de acordo com a hora local, e retorna a nova hora em milissegundos. A hora local é determinada pelo sistema operacional em que o Flash Player está sendo executado.

## Date.setMilliseconds

### Disponibilidade

Flash Player 5.

### Uso

```
myDate.setMilliseconds(milissegundos)
```

### Parâmetros

*milissegundos* Um inteiro de 0 a 999.

### Retorna

Um inteiro.

### Descrição

Método; define os milissegundos do objeto Date especificado, de acordo com a hora local, e retorna a nova hora em milissegundos. A hora local é determinada pelo sistema operacional em que o Flash Player está sendo executado.

## Date.setMinutes

### Disponibilidade

Flash Player 5.

### Uso

```
myDate.setMinutes(minutos)
```

### Parâmetros

*minutos* Um inteiro de 0 a 59.

### Retorna

Um inteiro.

### Descrição

Método; define os minutos do objeto Date especificado, de acordo com a hora local, e retorna a nova hora em milissegundos. A hora local é determinada pelo sistema operacional em que o Flash Player está sendo executado.

## Date.setMonth

### Disponibilidade

Flash Player 5.

### Uso

```
myDate.setMonth(mês [, data ])
```

### Parâmetros

*mês* Um inteiro entre 0 (janeiro) e 11 (dezembro).

*data* Um inteiro de 1 a 31. Este parâmetro é opcional.

### Retorna

Um inteiro.

**Descrição**

Método; define o mês do objeto Date especificado, de acordo com a hora local, e retorna a nova hora em milissegundos. A hora local é determinada pelo sistema operacional em que o Flash Player está sendo executado.

## Date.setSeconds

**Disponibilidade**

Flash Player 5.

**Uso**

```
myDate.setSeconds(segundos)
```

**Parâmetros**

*segundos* Um inteiro de 0 a 59.

**Retorna**

Um inteiro.

**Descrição**

Método; define os segundos do objeto Date especificado, de acordo com a hora local, e retorna a nova hora em milissegundos. A hora local é determinada pelo sistema operacional em que o Flash Player está sendo executado.

## Date.setTime

**Disponibilidade**

Flash Player 5.

**Uso**

```
myDate.setTime(milissegundos)
```

**Parâmetros**

*milissegundos* Um valor inteiro onde 0 é 0:00 GMT 1º jan. 1970.

**Retorna**

Um inteiro.

**Descrição**

Método; define a data do objeto Date especificado, em milissegundos, desde a meia-noite de 1º de janeiro de 1970, e retorna a nova hora em milissegundos.

## Date.setUTCDate

**Disponibilidade**

Flash Player 5.

**Uso**

```
myDate.setUTCDate(data)
```

**Parâmetros**

*data* Um inteiro de 1 a 31.

**Retorna**

Um inteiro.



### Descrição

Método; define a data do objeto Date especificado, de acordo com a hora universal, e retorna a nova hora em milissegundos. Quando este método é chamado, os outros campos do objeto Date especificado não são modificados, mas, se o dia da semana for alterado em decorrência do chamamento a esse método, os métodos `getUTCDay` e `getDay` podem reportar um novo valor.

## Date.setUTCFullYear

### Disponibilidade

Flash Player 5.

### Uso

```
myDate.setUTCFullYear(ano [, mês [, data]])
```

### Parâmetros

*ano* O ano especificado com quatro dígitos completos; por exemplo, 2000.

*mês* Um inteiro entre 0 (janeiro) e 11 (dezembro). Este parâmetro é opcional.

*data* Um inteiro de 1 a 31. Este parâmetro é opcional.

### Retorna

Um inteiro.

### Descrição

Método; define o ano do objeto Date especificado (*mydate*), de acordo com a hora universal, e retorna a nova hora em milissegundos.

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.

## Date.setUTCHours

### Disponibilidade

Flash Player 5.

### Uso

```
myDate.setUTCHours(hora [, minutos [, segundos [, milissegundos]])
```

### Parâmetros

*hora* Um inteiro entre 0 (meia-noite) e 23 (11 p.m.).

*minutos* Um inteiro de 0 a 59. Este parâmetro é opcional.

*segundos* Um inteiro de 0 a 59. Este parâmetro é opcional.

*milissegundos* Um inteiro de 0 a 999. Este parâmetro é opcional.

### Retorna

Um inteiro.

### Descrição

Método; define a hora do objeto Date especificado, de acordo com a hora universal, e retorna a nova hora em milissegundos.

## Date.setUTCMilliseconds

### Disponibilidade

Flash Player 5.

### Uso

*myDate.setUTCMilliseconds(milissegundos)*

### Parâmetros

*milissegundos* Um inteiro de 0 a 999.

### Retorna

Um inteiro.

### Descrição

Método; define os milissegundos do objeto Date especificado, de acordo com a hora universal, e retorna a nova hora em milissegundos.

## Date.setUTCMinutes

### Disponibilidade

Flash Player 5.

### Uso

*myDate.setUTCMinutes(minutos [, segundos [, milissegundos]])*

### Parâmetros

*minutos* Um inteiro de 0 a 59.

*segundos* Um inteiro de 0 a 59. Este parâmetro é opcional.

*milissegundos* Um inteiro de 0 a 999. Este parâmetro é opcional.

### Retorna

Um inteiro.

### Descrição

Método; define os minutos do objeto Date especificado, de acordo com a hora universal, e retorna a nova hora em milissegundos.

## Date.setUTCMonth

### Disponibilidade

Flash Player 5.

### Uso

*myDate.setUTCMonth(mês [, data])*

### Parâmetros

*mês* Um inteiro entre 0 (janeiro) e 11 (dezembro).

*data* Um inteiro de 1 a 31. Este parâmetro é opcional.

### Retorna

Um inteiro.

### Descrição

Método; define o mês e, opcionalmente, o dia (data), do objeto Date especificado, de acordo com a hora universal, e retorna a nova hora em milissegundos. Quando este método é chamado, os outros campos do objeto Date especificado não são modificados, mas, se o dia da semana for alterado em decorrência do resultado da especificação do parâmetro *data* quando for `setUTCMonth` for chamado, os métodos `getUTCDay` e `getDay` podem reportar um novo valor.

## Date.setUTCSeconds

### Disponibilidade

Flash Player 5.

### Uso

```
myDate.setUTCSeconds(segundos [, milissegundos])
```

### Parâmetros

*segundos* Um inteiro de 0 a 59.

*milissegundos* Um inteiro de 0 a 999. Este parâmetro é opcional.

### Retorna

Um inteiro.

### Descrição

Método; define os segundos do objeto Date especificado, de acordo com a hora universal, e retorna a nova hora em milissegundos.

## Date.setYear

### Disponibilidade

Flash Player 5.

### Uso

```
myDate.setYear(ano)
```

### Parâmetros

*ano* Se *ano* for um número inteiro entre 0–99, `setYear` definirá o ano como 1900 + *ano*; caso contrário, o ano será o valor do parâmetro *ano*.

### Retorna

Um inteiro.

### Descrição

Método; define o ano do objeto Date especificado, de acordo com a hora local, e retorna a nova hora em milissegundos. A hora local é determinada pelo sistema operacional em que o Flash Player está sendo executado.

## Date.toString

### Disponibilidade

Flash Player 5.

### Uso

```
myDate.toString()
```

### Parâmetros

Nenhum.

### Retorna

Uma seqüência de caracteres.

### Descrição

Método; retorna, em um formato legível, o valor de uma seqüência de caracteres do objeto Date especificado, e retorna a nova hora em milissegundos.

### Exemplo

O exemplo a seguir retorna as informações no objeto Date `dateOfBirth` como uma seqüência de caracteres.

```
var dateOfBirth = 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

### Disponibilidade

Flash Player 5.

### Uso

```
Date.UTC(ano, mês [, data [, hora [, minutos [, segundos [, milissegundos  
]]]]]);
```

### Parâmetros

*ano* Um número de quatro dígitos; por exemplo, 2000.

*mês* Um inteiro entre 0 (janeiro) e 11 (dezembro).

*data* Um inteiro de 1 a 31. Este parâmetro é opcional.

*hora* Um inteiro entre 0 (meia-noite) e 23 (11 p.m.).

*minutos* Um inteiro de 0 a 59. Este parâmetro é opcional.

*segundos* Um inteiro de 0 a 59. Este parâmetro é opcional.

*milissegundos* Um inteiro de 0 a 999. Este parâmetro é opcional.

### Retorna

Um inteiro.

### Descrição

Método; retorna o número de milissegundos entre a meia-noite de 1º de janeiro de 1970, hora universal, e a hora especificada nos parâmetros. Este é um método estático chamado pelo construtor do objeto Date, não por um objeto Date específico. Este método permite criar um objeto Date que assuma uma hora universal, enquanto o construtor de Date assume a hora local.

### 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));
```

## default

### Disponibilidade

Flash Player 6.

### Uso

`default`: *comandos*

### Parâmetros

*comandos* Qualquer comando.

### Retorna

Nada.

### Descrição

Palavra-chave; define o case padrão de uma ação `switch`. Os comandos serão executados se o parâmetro *Expressão* da ação `switch` for diferente (usando a igualdade estrita) de algum dos parâmetros *Expressão* subsequentes às palavras-chaves de case de uma determinada ação `switch`.

Para ter um case `default`, não é necessária uma ação `switch`. Um case `default` não precisa ser o último da lista. O uso de uma ação `default` fora de uma ação `switch` representa um erro e o script não pode ser compilado.

### Exemplo

No exemplo a seguir, a expressão A é diferente das expressões B ou D, portanto o comando que segue a palavra-chave padrão é executado e a ação `trace` é enviada para a janela Saída.

```
switch ( A ) {  
    case B:  
        C;  
        break;  
    case D:  
        E;  
        break;  
    default:  
        trace ("nenhum caso específico foi encontrado");  
}
```

### Consulte também

`switch`, `case`, `break`

# delete

## Disponibilidade

Flash Player 5.

## Uso

`delete referência`

## Parâmetros

*referência* O nome da variável ou do objeto a ser eliminado.

## Retorna

Nada.

## Descrição

Operador; elimina o objeto ou a variável especificada pelo parâmetro *referência* e retorna `true` se o objeto for excluído com êxito; do contrário, retorna `false`. Este operador é útil para liberar memória usada pelos scripts. Embora `delete` seja um operador, normalmente ele é usado como um comando, como no exemplo a seguir:

```
delete x;
```

O operador `delete` pode falhar e retornar `false` se o parâmetro de *referência* não existir ou não puder ser excluído. Objetos e propriedades predefinidos e variáveis declaradas com `var` não podem ser excluídas. Não é possível usar o operador `delete` para remover clipes de filme.

## Exemplo

O exemplo a seguir cria um objeto, usa-o e o exclui quando não for mais necessário.

```
account = new Object();
account.name = 'Jon';
account.balance = 10000;
```

```
delete account;
```

## Exemplo

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;
```

## Exemplo

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. Se `ref2` for excluído, não haverá mais referências ao objeto; ele será eliminado e a memória que estava sendo usada ficará disponível.

#### Consulte também

`var`

## do while

#### Disponibilidade

Flash Player 4.

#### Uso

```
do {
    comando(s)
} while (condição)
```

#### Parâmetros

*condição* A condição a ser avaliada.

*comando(s)* O(s) comando(s) a ser(em) executado(s) desde que o parâmetro *condição* seja avaliado como `true`.

#### Retorna

Nada.

#### Descrição

Ação; executa os comandos e, a seguir, avalia a condição em um loop, pelo tempo em que a condição `for true`.

#### Consulte também

`break`, `continue`

## duplicateMovieClip

#### Disponibilidade

Flash Player 4.

#### Uso

```
duplicateMovieClip(destino, novonome, profundidade)
```

#### Parâmetros

*destino* O caminho de destino do clipe de filme a ser duplicado.

*novonome* Um identificador exclusivo do clipe de filme duplicado.

*profundidade* Um nível de profundidade exclusivo para o clipe de filme duplicado. O nível de profundidade é uma ordem de empilhamento para os cliques de filmes duplicados. Essa ordem de empilhamento é muito parecida com a ordem de empilhamento das camadas na Linha de tempo; os cliques de filmes com um nível de profundidade inferior ficam ocultos abaixo de cliques com uma ordem de empilhamento superior. Você deve atribuir a cada clipe de filme duplicado um nível de profundidade exclusivo para evitar que ele substitua filmes em níveis ocupados.

#### **Retorna**

Nada.

#### **Descrição**

Ação; cria uma instância de um clipe de filme enquanto o filme é reproduzido. A reprodução em cliques de filme duplicados sempre começa pelo Quadro 1, independente da posição da reprodução no clipe de filme original (ou “pai”). 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. Use a ação ou método `removeMovieClip` para excluir uma instância de clipe de filme criada com `duplicateMovieClip`.

#### **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 exclusiva para cada clipe de filme duplicado.

```
on (release) {  
    amount = 10;  
    while (valor>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++;  
        valor--;  
    }  
}
```

#### **Consulte também**

`MovieClip.duplicateMovieClip`, `removeMovieClip`, `MovieClip.removeMovieClip`

## **else**

#### **Disponibilidade**

Flash Player 4.

#### **Uso**

Comando `else`

`else {...comando(s)...}`

#### **Parâmetros**

*condição* Uma expressão que seja avaliada como `true` ou `false`.

*comando(s)* Uma série alternativa de comandos a ser executada se a condição especificada no comando `if` for `false`.



**Retorna**

Nada.

**Descrição**

Ação; especifica os comandos a serem executados se a condição no comando `if for false`.

**Consulte também**

`if`

## else if

**Disponibilidade**

Flash Player 4.

**Uso**

```
if (condição){  
    comando(s);  
} else if (condição){  
    comando(s);  
}
```

**Parâmetros**

*condição* Uma expressão que seja avaliada como `true` ou `false`.

*comando(s)* Uma série alternativa de comandos a ser executada se a condição especificada no comando `if for false`.

**Retorna**

Nada.

**Descrição**

Ação; avalia uma condição e especifica os comandos a serem executados se a condição no comando `if` inicial `for false`. Se a condição `else if for true`, o interpretador Flash executará os comandos entre chaves (`{}`) que seguem a condição. Se a condição `else if for false`, o Flash não considerará os comandos entre chaves e executará os comandos após as chaves. Use a ação `else if` para criar uma lógica ramificada em seus scripts.

**Exemplo**

O exemplo seguinte faz uso de ações `else if` para verificar se cada lado de um objeto está dentro de um limite específico.

```
// se o objeto ultrapassar os limites,  
// mande-o de volta e inverta a velocidade de percurso  
if (this._x > rightBound) {  
    this._x = rightBound;  
    xInc = -xInc;  
} else if (this._x < leftBound) {  
    this._x = leftBound;  
    xInc = -xInc;  
} else if (this._y > bottomBound) {  
    this._y = bottomBound;  
    yInc = -yInc;  
} else if (this._y < topBound) {  
    this._y = topBound;  
    yInc = -yInc;  
}
```

**Consulte também**

`if`

## #endinitclip

### Disponibilidade

Flash Player 6.

### Uso

```
#endinitclip
```

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Ação; indica o término de um bloco de ações de inicialização de componentes.

### Exemplo

```
#initclip  
...ações de inicialização de componentes entram aqui...  
#endinitclip
```

### Consulte também

```
#initclip
```

## eq (igual – específico de sequência de caracteres)

### Disponibilidade

Flash Player 4. Esse operador foi substituído no Flash 5 pelo operador == (igualdade).

### Uso

```
expressão1 eq expressão2
```

### Parâmetros

*expression1*, *expression2* Números, seqüências de caracteres ou variáveis.

### Retorna

Nada.

### Descrição

Operador de comparação; avalia se duas expressões são iguais e retorna o valor `true` se a representação da seqüência de caracteres da *expressão1* for igual à representação da seqüência de caracteres da *expressão2*; caso contrário, a operação retorna o valor `false`.

### Consulte também

== (igualdade)

## escape

### Disponibilidade

Flash Player 5.

### Uso

`escape(expressão)`

### Parâmetros

*expressão* A expressão a ser convertida em sequência de caracteres e codificada em formato de URL.

### Retorna

Nada.

### Descrição

Função; converte o parâmetro em uma sequência de caracteres e o codifica em formato de URL, onde todos os caracteres que não são alfanuméricos são substituídos por seqüências hexadecimais de %.

### Exemplo

A execução do código a seguir apresenta o resultado 0i%7B%5BMundo%5D%7D.

```
escape("0i{[Mundo]}");
```

### Consulte também

`unescape`

## eval

### Disponibilidade

Flash Player 5 ou posterior com funcionalidade completa. Você pode usar a função `eval` ao exportar para o Flash Player 4, mas deve usar a notação de barra e só pode acessar variáveis, mas não propriedades ou objetos.

### Uso

`eval(expressão);`

### Parâmetros

*expressão* Uma seqüência de caracteres que contém o nome de uma variável, propriedade, objeto ou clipe de filme a ser recuperado.

### Retorna

Nada.

### Descrição

Função; acessa variáveis, propriedades, objetos ou clipes de filmes 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 citado na *expressão*, será retornado o valor `undefined`.

No Flash 4, a função `eval` era usada para simular matrizes, ao passo que no Flash 5, recomenda-se o uso do objeto `Array` para esse fim.

Também é possível usar a função `eval` para definir e recuperar de forma dinâmica o valor de uma variável ou um nome de instância. Porém, também é possível fazer isso através do operador de acesso a 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.

### Exemplo

O exemplo a seguir faz uso da função `eval` para determinar o valor da expressão `"piece" + x`. Como o resultado é um nome de variável, `piece3`, a função `eval` retorna o valor da variável e o atribui a `y`:

```
piece3 = "perigoso";  
x = 3;  
  
y = eval("piece" + x);  
trace(y);  
  
// Saída: perigoso
```

### Consulte também

Array (objeto)

## evaluate

### Disponibilidade

Flash Player 5.

### Uso

*comando*

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Ação; cria uma nova linha vazia e insere um ponto-e-vírgula (;) para a criação de comandos a serem avaliados no painel Ações.

## false

### Disponibilidade

Flash Player 5.

### Uso

`true`

### Descrição

Um valor Booleano exclusivo que representa o oposto de `true`.

### Consulte também

`true`

## FCheckBox (componente)

O componente CheckBox no ambiente de criação Flash oferece o recurso de arrastar e soltar para adicionar as caixas de seleção a documentos Flash; ele também oferece uma interface de usuário para a definição de parâmetros básicos. Os métodos do componente FCheckBox permitem controlar as caixas de seleção durante a execução: você pode criar caixas de seleção, controlar as caixas de seleção criadas no ambiente de criação, definir ou cancelar parâmetros básicos e definir opções adicionais de tempo de execução. Não é preciso usar um construtor para acessar os métodos de componentes.

O componente CheckBox tem suporte do Flash Player 6.

Os métodos do componente não realizam verificação de erros de tipo, como outros objetos e ações nativos do ActionScript; portanto, recomenda-se a validação dos parâmetros antes de passá-los para métodos.

Para obter informações sobre o uso do componente CheckBox, como definir parâmetros durante o processo de criação e como alterar as cores e a aparência de componentes, consulte “Personalizando cores e texto do componente” e “Personalizando aparências de componentes” no capítulo “Usando componentes” de *Usando o Flash*.

### Resumo dos métodos do componente FCheckBox

Método	Descrição
<code>FCheckBox.setEnabled</code>	Retorna <code>true</code> se a caixa de seleção estiver ativada, <code>false</code> se estiver desativada.
<code>FCheckBox.getLabel</code>	Retorna o rótulo aplicado à caixa de seleção como uma sequência de caracteres.
<code>FCheckBox.getValue</code>	Retorna <code>true</code> se a caixa de seleção estiver selecionada, <code>false</code> se não estiver selecionada.
<code>FCheckBox.registerSkinElement</code>	Registra um elemento de aparência em uma propriedade.
<code>FCheckBox.setChangeHandler</code>	Especifica um identificador de alteração a ser chamado quando o valor da caixa de seleção é alterado.
<code>FCheckBox.setEnabled</code>	Determina se a caixa de seleção está ativada ou desativada.
<code>FCheckBox.setLabel</code>	Especifica texto para o rótulo da caixa de seleção.
<code>FCheckBox.setLabelPlacement</code>	Especifica se o rótulo é exibido à esquerda ou à direita da caixa de seleção.
<code>FCheckBox.setSize</code>	Define a largura da caixa de seleção, em pixels, e redesenha a caixa.
<code>FCheckBox.setStyleProperty</code>	Define uma única propriedade de estilo para um componente.
<code>FCheckBox.setValue</code>	Marca ou desmarca a caixa de seleção e inicia a função do identificador de alteração.

## FCheckBox.setEnabled

### Disponibilidade

Flash Player 6.

### Uso

```
myCheckBox.setEnabled()
```

### Parâmetros

Nenhum.

### Retorna

Um valor Booleano que indica se a instância da caixa de seleção está ativada (`true`) ou desativada (`false`).

### Descrição

Método; indica se a instância da caixa de seleção está ativada ou desativada.

### Exemplo

O código a seguir retorna o estado ativado de `checkBox1` na janela Saída.

```
trace(checkBox1.setEnabled());
```

### Consulte também

`FCheckBox.setValue`

## FCheckBox.getLabel

### Disponibilidade

Flash Player 6.

### Uso

```
myCheckBox.getLabel()
```

### Parâmetros

Nenhum.

### Retorna

Uma seqüência de caracteres de texto.

### Descrição

Método; recupera o rótulo da caixa de seleção.

### Exemplo

O código a seguir retorna o rótulo de `checkBox1`.

```
checkBox1.getLabel();
```

### Consulte também

`FCheckBox.setLabel`

## FCheckBox.getValue

### Disponibilidade

Flash Player 6.

### Uso

```
myCheckBox.getValue()
```

### Parâmetros

Nenhum.

### Retorna

Um valor Booleano que indica se a instância da caixa de seleção está ativada (`true`) ou desativada (`false`).

### Descrição

Método; indica se a caixa de seleção está selecionada.

### Exemplo

O código a seguir retorna o valor selecionado de `KowalczykBox` na janela Saída.

```
trace(KowalczykBox.getValue());
```

### Consulte também

`FCheckBox.setValue`

## FCheckBox.registerSkinElement

### Disponibilidade

Flash Player 6.

### Uso

```
myCheckBox.registerSkinElement(element, styleProperty)
```

### Parâmetros

*element* Uma instância de clipe de filme.

*styleProperty* O nome de uma propriedade de `FStyleFormat`.

### Retorna

Nada.

### Descrição

Método; registra um elemento de aparência em uma propriedade de estilo. Elementos de aparência são registrados em propriedades no primeiro quadro da camada ReadMe de cada aparência na biblioteca.

Os componentes são compostos de aparências e cada aparência é composta de vários elementos de aparência, cada um dos quais pode ser registrado em uma propriedade do objeto `FStyleFormat`. Essas propriedades são valores atribuídos pelo formato de estilo atribuído a um componente. Como padrão, o objeto `globalStyleFormat` é atribuído a todos os componentes de interface do Flash. Esse objeto é uma instância do objeto `FStyleFormat`.

Use este método para registrar propriedades e elementos de aparência personalizados na interface do Flash ou aparências personalizadas de componentes editando o código no primeiro quadro da camada ReadMe de uma aparência na biblioteca.

O componente FCheckBox usa as aparências na pasta FCheckBox Skins depois que o componente for adicionado ao documento Flash.

Para obter mais informações, consulte “Personalizando aparências de componentes” no capítulo “Usando componentes” de *Usando o Flash*.

#### Exemplo

O código a seguir registra o elemento de aparência personalizado `customChk_mc` na propriedade `check` no primeiro quadro da camada ReadMe da aparência `fcx_check` na pasta FCheckBox Skins na biblioteca.

```
check1.registerSkinElement(customChk_mc, "check");
```

## FCheckBox.setChangeHandler

### Disponibilidade

Flash Player 6.

### Uso

```
myCheckBox.setChangeHandler(functionName, [location])
```

### Parâmetros

*functionName* Uma sequência de caracteres que especifica o nome da função do identificador a ser executada quando o valor da caixa de seleção é alterado. Se o parâmetro *location* não for especificado, esta função deverá estar na mesma Linha de tempo da instância do componente.

*location* Uma referência de caminho até um objeto de dados, clipe de filme ou Linha de tempo que contém a função especificada. Este parâmetro é opcional e tem como padrão a Linha de tempo pai do componente.

### Retorna

Nada.

### Descrição

Método; especifica um identificador de alteração a ser chamado quando o valor da caixa de seleção é alterado. Você pode especificar a mesma função de identificador de alteração para mais de um componente; a função sempre aceita a instância do componente que foi alterada como um parâmetro. Se este método for chamado, valor do parâmetro Identificador de alteração especificado na criação será cancelado.

Para obter mais informações, consulte “Criando funções do identificador de alteração para componentes” no capítulo “Usando componentes” de *Usando o Flash*.

### Exemplo

O código a seguir especifica `myHandler` como a função chamada quando o valor de `checkBox1` é alterado. Como o parâmetro *location* não foi especificado, `myHandler` deverá estar na mesma Linha de tempo da instância do componente.



O parâmetro `component` em `myHandler` é automaticamente preenchido com a instância de um componente (o componente que foi alterado como resultado de uma entrada do usuário e que especifica `myHandler` como seu identificador de alteração). As ações definidas em `myHandler` especificam que quando o usuário marca uma caixa de seleção, o nome do componente é exibido na janela Saída juntamente com “foi selecionado”.

```
checkBox1.setChangeHandler("myHandler");  
function myHandler(component){  
    trace(component._name + " foi selecionado ");  
}
```

Se, no exemplo acima, `myHandler` fosse uma função localizada na Linha de tempo bisavó da Linha de tempo do componente, a primeira linha de código seria da seguinte forma:

```
check1.setChangeHandler("myHandler", _parent._parent._parent);
```

O código a seguir cria a função `myHandler` em uma instância de `myObject` (que é da classe `Object`) e, a seguir, especifica `myHandler` como a função de `check1`.

```
myObject = new Object();  
myObject.myHandler = function(component){  
    trace(component._name + " foi selecionado ");  
}  
  
check1.setChangeHandler("myHandler", myObject);
```

## FCheckBox.setEnabled

### Disponibilidade

Flash Player 6.

### Uso

```
myCheckBox.setEnabled(enable)
```

### Parâmetros

*enable* Um valor Booleano que especifica se a caixa de seleção está ativada (`true`) ou desativada (`false`).

### Retorna

Nada.

### Descrição

Método; especifica se a caixa de seleção está ativada (`true`) ou desativada (`false`). Se uma caixa de seleção estiver desativada, ela não aceitará a interação do mouse nem do teclado do usuário. Se esse parâmetro for omitido, o método utilizará o padrão `true`.

### Exemplo

O código a seguir desativa `checkBox1`.

```
checkBox1.setEnabled(false);
```

## FCheckBox.setLabel

### Disponibilidade

Flash Player 6.

### Uso

```
myCheckBox.setLabel(label)
```

### Parâmetros

*label* Uma seqüência de caracteres que especifica o rótulo de texto da caixa de seleção.

### Retorna

Nada.

### Descrição

Método; especifica o rótulo de texto da caixa de seleção. Como padrão, o rótulo é exibido à direita da caixa de seleção. Se este método for chamado, o parâmetro *label* especificado na criação será cancelado.

### Exemplo

O código a seguir aplica o rótulo “Enviar mais informações” a `checkBox1`.

```
checkBox1.setLabel("Enviar mais informações");
```

### Consulte também

`FCheckBox.getLabel`, `FCheckBox.setLabelPlacement`

## FCheckBox.setLabelPlacement

### Disponibilidade

Flash Player 6.

### Uso

```
myCheckBox.setLabelPlacement(labelPosition)
```

### Parâmetros

*labelPosition* Uma seqüência de caracteres de texto; especifica "left" ou "right".

### Retorna

Nada.

### Descrição

Método; especifica se o rótulo é exibido à esquerda ou à direita da caixa de seleção. Se este método for chamado, o valor do parâmetro Label Placement definido durante o processo de criação será cancelado.

### Exemplo

O código a seguir posiciona o rótulo de `checkBox1` à esquerda da caixa de seleção.

```
checkBox1.setLabelPlacement("left");
```

### Consulte também

`FCheckBox.setLabel`

## FCheckBox.setSize

### Disponibilidade

Flash Player 6.

### Uso

```
myCheckBox.setSize(width)
```

### Parâmetros

*width* Um número inteiro que especifica a largura da caixa de seleção, em pixels.

### Retorna

Nada.

### Descrição

Método; especifica a largura da caixa de seleção e redesenha a caixa. Não é possível definir a altura dos componentes da caixa de seleção. Se este método for chamado, o dimensionamento de largura aplicado durante o processo de criação será cancelado.

Para obter mais informações, consulte “Dimensionando componentes CheckBox” no capítulo “Usando componentes” de *Usando o Flash*.

### Exemplo

O código a seguir define a largura da `checkBox1` como sendo 200 pixels.

```
checkBox1.setSize(200);
```

## FCheckBox.setStyleProperty

### Disponibilidade

Flash Player 6.

### Uso

```
myCheckBox.setStyleProperty(styleProperty, value)
```

### Parâmetros

*styleProperty* Uma sequência de caracteres que especifica uma propriedade do objeto `FStyleFormat`.

*value* O valor definido para a propriedade.

### Retorna

Nada.

### Descrição

Método; define uma propriedade `FStyleFormat` para uma determinada caixa de seleção. Chamar este método para especificar uma propriedade cancela as configurações dessa propriedade no formato de estilo atribuído ao componente. Se o valor `undefined` for atribuído a uma propriedade, todos os estilos dessa propriedade serão removidos.

Para definir as propriedades `FStyleFormat` para vários componentes, crie um formato de estilo personalizado. Para obter mais informações sobre a criação de formatos de estilo personalizados, consulte “Personalizando cores e texto do componente” no capítulo “Usando componentes” de *Usando o Flash*.

### Exemplo

O código a seguir define a propriedade `shadow` de `checkBox1` como sendo `0x000000` (preto).

```
checkBox1.setStyleProperty("shadow", 0x000000);
```

### Consulte também

`FStyleFormat` (object)

## FCheckBox.setValue

### Disponibilidade

Flash Player 6.

### Uso

```
myCheckBox.setValue(select)
```

### Parâmetros

*select* Um valor Booleano que especifica se a caixa de seleção está marcada (`true`) ou não (`false`).

### Retorna

Nada.

### Descrição

Método; marca ou desmarca *myCheckBox* e inicia a função do identificador de alteração especificada (caso haja alguma) durante a execução. O valor padrão é `true`.

Embora o fato de chamar este método cancele o valor do parâmetro `Initial Value` especificado na criação, não use o método com este objetivo, pois ele também inicia a função do identificador de alteração associada. Para definir o parâmetro `Initial Value` de uma caixa de seleção durante a execução, use `FCheckBox.setStyleProperty`.

### Exemplo

O código a seguir seleciona a instância de `checkBox1` e inicia a função do identificador de alteração que estiver especificada.

```
checkBox1.setValue(true);
```

### Consulte também

`FCheckBox.getValue`

## FComboBox (component)

O componente `ComboBox` no ambiente de criação Flash oferece o recurso de arrastar e soltar para adicionar listas suspensas de seleção simples a documentos do Flash; ele também oferece uma interface de usuário para a definição de parâmetros básicos. Os métodos do componente `FComboBox` permitem controlar as caixas de combinação durante a execução: você pode criar caixas de combinação, controlar as caixas de combinação criadas no ambiente de criação, definir ou cancelar os parâmetros básicos e definir as opções adicionais de tempo de execução. Não é preciso usar um construtor para acessar os métodos de componentes.

O componente ComboBox cria caixas de combinação estáticas e editáveis. A caixa de combinação estática é uma lista suspensa rolável que permite aos usuários selecionar itens. Uma caixa de combinação editável é uma lista suspensa rolável com um campo de texto na parte superior. Você pode permitir que os usuários insiram texto no campo de texto para fazer a caixa de combinação rolar até o item desejado ou pode usar o campo de texto para definir o texto exibido durante a execução.

Tanto a versão estática quanto a editável do componente ComboBox relaciona itens de cima para baixo usando um sistema de indexação baseado no zero. Se o número de itens na lista da caixa de combinação criar uma lista suspensa que ultrapasse o espaço disponível abaixo do componente, a lista se abrirá para cima e não para baixo.

Os métodos do componente não realizam verificação de erros de tipo, como outros objetos e ações nativos do ActionScript; portanto, recomenda-se a validação dos parâmetros antes de passá-los para métodos.

O componente ComboBox tem suporte do Flash Player 6 e de suas versões posteriores.

Para obter informações sobre o uso do componente ComboBox, como definir parâmetros durante o processo de criação e como alterar as cores e a aparência de componentes, consulte “Personalizando cores e texto do componente” e “Personalizando aparências de componentes” no capítulo “Usando componentes” de *Usando o Flash*.

## Resumo dos métodos do componente FComboBox

Método	Descrição
FComboBox.addItem	Adiciona um novo item ao final da lista da caixa de combinação.
FComboBox.addItemAt	Adiciona um novo à lista da caixa de combinação no índice especificado.
FComboBox.setEnabled	Retorna true se a caixa de combinação estiver ativada, false se estiver desativada.
FComboBox.getItemAt	Retorna o item no índice especificado como um objeto com as propriedades label e data.
FComboBox.getLength	Retorna o número de itens relacionados na caixa de combinação.
FComboBox.getRowCount	Retorna o número de linhas visíveis na caixa de combinação.
FComboBox.getScrollPosition	Retorna o índice do item na parte superior da caixa de combinação.
FComboBox.getSelectedIndex	Retorna o índice do item atualmente selecionado.
FComboBox.getSelectedItem	Retorna o item atualmente selecionado como um objeto com as propriedades label e data.
FComboBox.getValue	Retorna o texto no campo de entrada no caso de caixas de combinação editáveis; retorna o rótulo ou os dados do item selecionado no caso de caixas de combinação estáticas.
FComboBox.registerSkinElement	Registra um elemento de aparência em uma propriedade.
FComboBox.removeAll	Remove todos os itens da caixa de combinação.
FComboBox.removeItemAt	Remove o item no índice especificado.
FComboBox.replaceItemAt	Substitui o rótulo e os dados de um item no índice especificado.
FComboBox.setChangeHandler	Atribui uma função a ser chamada todas as vezes que um item é selecionado ou que o usuário insere texto no campo de texto.

Método	Descrição
<code>FComboBox.setDataProvider</code>	Registra um objeto externo no componente como uma fonte de dados.
<code>FComboBox.setEditable</code>	Determina se a caixa de combinação é editável ( <code>true</code> ) ou estática ( <code>false</code> ).
<code>FComboBox.setEnabled</code>	Especifica se a caixa de combinação está ativada ( <code>true</code> ) ou desativada ( <code>false</code> ).
<code>FComboBox.setItemSymbol</code>	Registra o identificador de vinculação de um símbolo a ser usado para exibir itens de listagem de uma caixa de combinação.
<code>FComboBox.setRowCount</code>	Determina o número de itens exibidos na caixa de combinação sem uma barra de rolagem.
<code>FComboBox.setSelectedIndex</code>	Seleciona o item no índice especificado.
<code>FComboBox.setSize</code>	Define a largura em pixels da caixa de combinação.
<code>FComboBox.setStyleProperty</code>	Define uma única propriedade de estilo para a instância de um componente.
<code>FComboBox.setValue</code>	Especifica o texto exibido no campo de texto na parte superior da caixa de combinação editável.
<code>FComboBox.sortItemsBy</code>	Classifica os itens na caixa de listagem em ordem alfabética ou numérica por rótulo ou por dados.

## FComboBox.addItem

### Disponibilidade

Flash Player 6.

### Uso

```
myComboBox.addItem(label [, data])
```

### Parâmetros

*label* Uma seqüência de caracteres de texto a ser exibida na listagem da caixa de combinação.

*data* O valor a ser associado ao item da listagem. Este parâmetro é opcional.

### Retorna

Nada.

### Descrição

Método; adiciona um novo item com o rótulo e os dados especificados ao final da listagem da caixa de combinação e atualiza a listagem. O parâmetro *data* pode ser qualquer objeto do Flash, seqüência de caracteres, valor Booleano, número inteiro, objeto ou clipe de filme.

Para obter melhor desempenho e menor tempo de carregamento, não adicione mais de 400 itens a cada quadro. Isso se aplica esteja você adicionando os itens a uma única listagem de caixa de combinação ou a várias.

### Exemplo

O código a seguir adiciona o item `Kenny` com o valor associado `Keen` ao final da listagem na caixa de combinação `teacherList`.

```
teacherList.addItem("Kenny", Keen);
```

O código a seguir adiciona o número máximo de itens recomendado em um único quadro (400 itens) à `comboBox1`:

```
for (i=0; i<400; i++) {  
    comboBox1.addItem(i);  
}
```

O código a seguir adiciona o número máximo de itens recomendado em um único quadro (400 itens) a `listBox1` e a `comboBox2`:

```
for (i=0; i<200; i++) {  
    listBox1.addItem(i);  
    comboBox2.addItem(i);  
}
```

#### Consulte também

`FComboBox.addItemAt`, `FComboBox.getItemAt`, `FComboBox.replaceItemAt`,  
`FComboBox.setDataProvider`, `FComboBox.sortItemsBy`

## FComboBox.addItemAt

#### Disponibilidade

Flash Player 6.

#### Uso

```
myComboBox.addItemAt(index, label [,data])
```

#### Parâmetros

*index* Um número inteiro que especifica a posição onde inserir o item.

*label* Uma sequência de caracteres que identifica o item da listagem na caixa de combinação.

*data* O valor a ser associado ao item da listagem. Este parâmetro é opcional.

#### Retorna

Nada.

#### Descrição

Método; adiciona um novo item com o rótulo especificado e os dados opcionais associados à listagem da caixa de combinação na posição de índice especificada. O parâmetro `Data` pode ser qualquer objeto do Flash, sequência de caracteres, valor Booleano, número inteiro, objeto ou clipe de filme. À medida que cada item é adicionado, a listagem é atualizada e a barra de rolagem é redimensionada.

O componente `ComboBox` usa um índice com base no zero, onde o item no índice 0 é exibido no topo da listagem.

Para obter melhor desempenho e menor tempo de carregamento, não adicione mais de 400 itens a cada quadro. Isso se aplica esteja você adicionando os itens a uma única listagem de caixa de combinação ou a várias.

#### Exemplo

O código a seguir adiciona o item `Justin` com o valor associado `Ace` como quinto item na listagem da caixa de combinação `Favorites`.

```
Favorites.addItemAt(4, "Justin", Ace);
```

Para obter exemplos de como carregar um grande número de itens, consulte `FComboBox.addItem`.

#### Consulte também

`FComboBox.getItemAt`, `FComboBox.removeItemAt`, `FComboBox.replaceItemAt`,  
`FComboBox.setDataProvider`, `FComboBox.sortItemsBy`

## FComboBox.setEnabled

### Disponibilidade

Flash Player 6.

### Uso

```
myComboBox.setEnabled()
```

### Parâmetros

Nenhum.

### Retorna

Um valor Booleano que indica se a caixa de combinação está ativada (`true`) ou desativada (`false`).

### Descrição

Método; indica se a caixa de combinação está ativada.

### Exemplo

O código a seguir usa `setEnabled` para determinar se `comboBox1` está ativada ou desativada e exibe o resultado na janela Saída.

```
trace(comboBox1.setEnabled());
```

### Consulte também

`FComboBox.setEnabled`

## FComboBox.getItemAt

### Disponibilidade

Flash Player 6.

### Uso

```
myComboBox.getItemAt(index)
```

### Parâmetros

*index* Um número inteiro que especifica a posição de um item na caixa de combinação.

### Retorna

Um objeto.

### Descrição

Método; retorna o item no índice especificado como um objeto com as propriedades `label` e `data`.

O componente `ComboBox` usa um índice com base no zero, onde o item no índice 0 é exibido no topo da listagem.

### Exemplo

O código apresentado a seguir retorna o rótulo do item no índice 4 na `comboBox1` como uma sequência de caracteres.

```
trace(comboBox1.getItemAt(4).label);
```



O código a seguir retorna os dados associados ao item no índice 4 na `comboBox2`. O valor de retorno depende do tipo de dado e pode ser um objeto, sequência de caracteres, referência de clipe de filme ou outro valor.

```
trace(comboBox2.getItemAt(4).data);
```

O código a seguir retorna um objeto contendo o rótulo e o valor de dados associado ao item no índice 4 na `comboBox3`.

```
trace(comboBox3.getItemAt(4));
```

#### Consulte também

`FComboBox.getSelectedItem`

## FComboBox.getLength

#### Disponibilidade

Flash Player 6.

#### Uso

```
myComboBox.getLength()
```

#### Parâmetros

Nenhum.

#### Retorna

Um inteiro.

#### Descrição

Método; retorna o número de itens na listagem da caixa de combinação.

#### Exemplo

O código a seguir recupera o número de itens na lista de `listMain` e armazena esse valor na variável `len`.

```
var len = listMain.getLength();
```

#### Consulte também

`FComboBox.addItem`, `FComboBox.addItemAt`

## FComboBox.getRowCount

#### Disponibilidade

Flash Player 6.

#### Uso

```
myComboBox.getRowCount()
```

#### Parâmetros

Nenhum.

#### Retorna

Um inteiro.

#### Descrição

Método; retorna o número de linhas visíveis na caixa de combinação.

**Exemplo**

O código a seguir retorna o número de linhas visíveis em `toyList` e define o valor para a variável `rowCount`.

```
var rowCount = toyList.getRowCount();
```

**Consulte também**

`FComboBox.setRowCount`

## **FComboBox.getScrollPosition**

**Disponibilidade**

Flash Player 6.

**Uso**

```
myComboBox.getScrollPosition()
```

**Parâmetros**

Nenhum.

**Retorna**

Um inteiro.

**Descrição**

Método; retorna o índice do item exibido no momento na parte superior da caixa de combinação.

O componente `ComboBox` usa um índice com base no zero, onde o item no índice 0 é exibido no topo da listagem.

**Exemplo**

O código a seguir recupera o índice do item que, no momento, está na parte superior da lista em `toyList` e armazena esse valor na variável `scrollPos`.

```
var scrollPos = toyList.getScrollPosition();
```

**Consulte também**

`FComboBox.setSelectedIndex`

## **FComboBox.getSelectedIndex**

**Disponibilidade**

Flash Player 6.

**Uso**

```
myComboBox.getSelectedIndex()
```

**Parâmetros**

Nenhum.

**Retorna**

Um número inteiro ou `undefined`.

**Descrição**

Método; retorna o índice do item selecionado no momento na caixa de combinação, ou retorna `undefined`, se não houver um item selecionado.

Os itens são apresentados na caixa de combinação da parte superior para a inferior usando um índice baseado no zero.

**Exemplo**

O código a seguir recupera o índice do item atualmente selecionado em `toyList` e armazena esse valor na variável `selectedIndex`.

```
var selectedIndex = toyList.getSelectedIndex();
```

**Consulte também**

`FComboBox.setSelectedIndex`

## **FComboBox.getSelectedItem**

**Disponibilidade**

Flash Player 6.

**Uso**

```
myComboBox.getSelectedItem()
```

**Parâmetros**

Nenhum.

**Retorna**

Um objeto ou `undefined`.

**Descrição**

Método; retorna o item atualmente selecionado como um objeto com as propriedades `label` e `data`, ou retorna `undefined`, se não houver um item selecionado.

**Exemplo**

O código a seguir recupera o rótulo e os dados do item atualmente selecionados na `comboBox1`.

```
trace(comboBox1.getSelectedItem());
```

O código a seguir recupera o rótulo do item atualmente selecionado na `comboBox2`.

```
trace(comboBox2.getSelectedItem().label);
```

O código a seguir recupera os dados do item atualmente selecionado na `comboBox3`.

```
trace(comboBox3.getSelectedItem().data);
```

**Consulte também**

`FComboBox.setSelectedIndex`

## **FComboBox.getValue**

**Disponibilidade**

Flash Player 6.

**Uso**

```
myComboBox.getValue()
```

**Parâmetros**

Nenhum.

**Retorna**

Uma seqüência de caracteres de texto.

### Descrição

Método; retorna o texto do campo na parte superior da caixa de combinação, se a caixa de combinação for editável. Se a caixa de combinação for estática (não editável), este método retornará os dados associados ao item selecionado, ou o rótulo do item, se não houver dados associados.

### Exemplo

O código a seguir retorna os dados ou o rótulo do item atualmente selecionado em `menuMain`.

```
trace(menuMain.getValue())
```

### Consulte também

`FComboBox.setValue`

## FComboBox.registerSkinElement

### Disponibilidade

Flash Player 6.

### Uso

```
myComboBox.registerSkinElement(element, styleProperty)
```

### Parâmetros

*element* Uma instância de clipe de filme.

*styleProperty* O nome de uma propriedade de `FStyleFormat`.

### Retorna

Nada.

### Descrição

Método; registra um elemento de aparência em uma propriedade de estilo. Elementos de aparência são registrados em propriedades no primeiro quadro da camada `ReadMe` de cada aparência na biblioteca.

Os componentes são compostos de aparências e cada aparência é composta de vários elementos de aparência, cada um dos quais pode ser registrado em uma propriedade do objeto `FStyleFormat`. Essas propriedades são valores atribuídos pelo formato de estilo atribuído a um componente. Como padrão, o objeto `globalStyleFormat` é atribuído a todos os componentes de interface do Flash. Esse objeto é uma instância do objeto `FStyleFormat`.

Use este método para registrar propriedades e elementos de aparência personalizados na interface do Flash ou aparências personalizadas de componentes editando o código no primeiro quadro da camada `ReadMe` de uma aparência na biblioteca.

O componente `FComboBox` usa as aparências na pasta `FComboBox Skins` depois que o componente for adicionado ao documento Flash.

Para obter mais informações, consulte “Personalizando aparências de componentes” no capítulo “Usando componentes” de *Usando o Flash*.

### Exemplo

O código a seguir registra o elemento de aparência personalizado `boundBox_mc` na propriedade `background` no primeiro quadro da camada `ReadMe` da aparência `FBoundingBox` na pasta `Global Skins`.

```
toysMenu.registerSkinElement(boundBox_mc, "background");
```

## FComboBox.removeAll

### Disponibilidade

Flash Player 6.

### Uso

```
myComboBox.removeAll();
```

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Método; remove todos os itens na listagem da caixa de combinação, atualiza a listagem e redimensiona a barra de rolagem. As caixas de combinação sem itens são exibidas sem barra de rolagem. Este método não pode ser usado se a caixa de combinação estiver desativada.

### Exemplo

O código a seguir remove todos os itens de menuMain.

```
menuMain.removeAll();
```

### Consulte também

FComboBox.removeItemAt

## FComboBox.removeItemAt

### Disponibilidade

Flash Player 6.

### Uso

```
myComboBox.removeItemAt(index)
```

### Parâmetros

*index* Um número inteiro que especifica o índice do item a ser removido.

### Retorna

Um objeto que contém o item removido.

### Descrição

Método; retorna o item removido no índice especificado e atualiza a lista. Quando um item é removido da lista, os índices dos itens subsequentes são atualizados para refletir suas novas posições. Se não houver item algum no índice especificado, este método retornará `undefined`.

O componente ComboBox usa um índice com base no zero, onde o item no índice 0 é exibido no topo da listagem.

### Exemplo

O código a seguir remove o quinto item da lista em menuMain.

```
menuMain.removeItemAt(4);
```

### Consulte também

FComboBox.removeAll

## FComboBox.replaceItemAt

### Disponibilidade

Flash Player 6.

### Uso

```
myComboBox.replaceItemAt(index, label [,data])
```

### Parâmetros

*index* Um número inteiro que especifica a posição de um item de listagem.

*label* Uma sequência de caracteres que especifica um novo rótulo para o item de listagem.

*data* O novo valor a ser associado ao item de listagem. Este parâmetro é opcional; se você não especificá-lo, qualquer dado atualmente especificado para o item permanece no lugar.

### Retorna

Nada.

### Descrição

Método; atualiza o item no índice especificado com o rótulo e os dados especificados. Se o item no índice especificado tiver um valor de dado associado e você não especificar um valor para o parâmetro *data*, o valor dos dados do item de listagem não será alterado.

O componente ComboBox usa um índice com base no zero, onde o item no índice 0 é exibido no topo da listagem.

### Exemplo

O código a seguir atualiza o quinto item na caixa de combinação Favorites com o rótulo Nigel e o valor de dados 7439. Se nenhum dado tiver sido especificado para o item de listagem, Nigel assume o valor de dados do item de listagem existente.

```
Favorites.replaceItemAt(4, "Nigel", "7439");
```

### Consulte também

FComboBox.addItemAt, FComboBox.getItemAt, FComboBox.setDataProvider, FComboBox.sortItemsBy

## FComboBox.setChangeHandler

### Disponibilidade

Flash Player 6.

### Uso

```
myComboBox.setChangeHandler(functionName, [location])
```

### Parâmetros

*functionName* Uma sequência de caracteres que especifica o nome da função do identificador a ser executada quando a seleção na caixa de combinação é alterada. Se o parâmetro *location* não for especificado, esta função deverá estar na mesma Linha de tempo da instância do componente.

*location* Uma referência de caminho até um objeto de dados, clipe de filme ou Linha de tempo que contém a função especificada. Este parâmetro é opcional e tem como padrão a Linha de tempo pai do componente.

**Retorna**

Nada.

**Descrição**

Método; especifica um identificador de alteração a ser chamado quando a seleção na caixa de seleção é alterada. Você pode especificar a mesma função de identificador de alteração para mais de um componente; a função sempre aceita a instância do componente que foi alterada como um parâmetro. Se este método for chamado, valor do parâmetro Identificador de alteração especificado na criação será cancelado.

Para obter mais informações, consulte “Criando funções do identificador de alteração para componentes” no capítulo “Usando componentes” de *Usando o Flash*.

**Exemplo**

O código a seguir especifica `myHandler` como a função chamada quando o valor de `toyList` é alterado. Como o parâmetro `location` não foi especificado, `myHandler` deverá estar na mesma Linha de tempo da instância do componente.

O parâmetro `component` em `myHandler` é automaticamente preenchido com a instância de um componente (o componente que foi alterado como resultado de uma entrada do usuário e que especifica `myHandler` como seu identificador de alteração). As ações definidas em `myHandler` especificam que o rótulo do item é exibido na janela Saída quando o usuário seleciona um item na listagem.

```
toyList.setChangeHandler("myHandler");
function myHandler(component){
    trace(toyList.getSelectedItem().label);
}
```

Se, no exemplo acima, `myHandler` fosse uma função localizada na Linha de tempo bisavô da Linha de tempo do componente, a primeira linha de código seria da seguinte forma:

```
toyList.setChangeHandler("myHandler", _parent._parent._parent);
```

O código a seguir cria a função `myHandler` em uma instância de `myObject` (que é da classe `Object`) e, a seguir, especifica `myHandler` como a função de `toyList`.

```
myObject = new Object();
myObject.myHandler = function(component){
    trace(toyList.getSelectedItem().label);
}

toyList.setChangeHandler("myHandler", myObject);
```

## FComboBox.setDataProvider

**Disponibilidade**

Flash Player 6.

**Uso**

```
myComboBox.setDataProvider(dataProvider)
```

**Parâmetros**

*dataProvider* Uma matriz de seqüências de caracteres de texto que lista itens para adicionar, uma instância do objeto `Array` que especifica os itens a serem adicionados ou uma instância da classe `DataProvider`.

**Retorna**

Nada.

## Descrição

Método; registra um objeto externo (*dataProvider*) como a fonte de dados para o componente da caixa de combinação. Se *dataProvider* for uma instância do objeto Array, o objeto poderá especificar *label*, *data* ou ambos, pois as propriedades de objeto e o conteúdo da matriz poderão ser copiados para a caixa de combinação como rótulos, dados ou ambos. Se *dataProvider* for uma instância da classe *DataProvider*, ele deverá implementar a API *DataProvider* definida no símbolo *DataProvider* na pasta *FlashUIComponents/Core Assets/ClassTree* da biblioteca. A API *DataProvider* é somente para usuários e programadores avançados; todos os outros usuários devem usar uma matriz ou um objeto Array.

## Exemplo

O código a seguir especifica o objeto Array *peopleList1* como o provedor de dados para *comboBox1*.

```
comboBox1.setDataProvider(peopleList1);
```

O código a seguir cria a matriz *peopleList* para exibir os rótulos dos itens listados em *comboBox1*.

```
peopleList = new Array();
peopleList[0] = "BHall";
peopleList[1] = "CMooock";
peopleList[2] = "MWobensmith";
peopleList[3] = "MShepherd";
```

O código a seguir cria a matriz *itemList1*, que especifica tanto o rótulo quanto os dados para itens de listagem. Esse objeto Array poderia ser usado como um provedor de dados alternativo para *comboBox1*.

```
itemList1 = new Array();
for (i=0; i<10; i++) {
    // criar um item real
    var myItem = new Object();
    myItem.label = "Item" + i;
    myItem.data = 75;
    // coloca-lo na matriz
    itemList1[i] = myItem;
}
```

O código a seguir especifica *comboData*, uma instância da classe *DataProvider*, como o provedor de dados de *comboBox1*.

```
comboBox1.setDataProvider(comboData);
```

O código a seguir cria uma nova instância da classe *DataProvider* e, a seguir, adiciona os rótulos do item usando o método *addItem* de *DataProvider*.

**Observação:** O método *addItem* é somente um método da classe *DataProvider*. Os programadores interessados em usar a classe *DataProvider* devem consultar o símbolo *DataProvider* na pasta *FlashUIComponents/CoreAssets/ClassTree* na biblioteca antes de tentar usar os métodos.

```
comboData = new DataProviderClass();

comboData.addItem("Devra");
comboData.addItem("Delia");
comboData.addItem("Vashti");
comboData.addItem("Alicia");
```

## Consulte também

*FComboBox.addItem*, *FComboBox.replaceItemAt*, *FComboBox.sortItemsBy*



## FComboBox.setEditable

### Disponibilidade

Flash Player 6.

### Uso

```
myComboBox.setEditable(editable)
```

### Parâmetros

*editable* Um valor Booleano que especifica se a caixa de combinação é editável (`true`) ou estática (`false`).

### Retorna

Nada.

### Descrição

Método; determina se a caixa de combinação é editável (`true`) ou estática (`false`). Uma caixa de combinação editável tem um campo de texto; quando o usuário insere texto, a caixa de combinação rola até o item com o mesmo texto. O campo de texto também pode ser utilizado para exibir texto usando `FComboBox.setValue`. Se este método for chamado, o valor do parâmetro `Editable` definido durante o processo de criação será cancelado.

### Exemplo

O código a seguir ativa um campo de texto de entrada na parte superior de `menuMain`.

```
menuMain.setEditable(true);
```

### Consulte também

`FComboBox.setValue`

## FComboBox.setEnabled

### Disponibilidade

Flash Player 6.

### Uso

```
myComboBox.setEnabled(enable)
```

### Parâmetros

*enable* Um valor Booleano que especifica se a caixa de combinação está ativada (`true`) ou desativada (`false`).

### Retorna

Nada.

### Descrição

Método; determina se a caixa de combinação está ativada (`true`) ou desativada (`false`). Se uma caixa de combinação estiver desativada, ela não aceitará interação de mouse nem de teclado do usuário. Se esse parâmetro for omitido, o método utilizará o padrão `true`.

### Exemplo

O código a seguir desativa `menuMain`.

```
menuMain.setEnabled(false);
```

### Consulte também

`FListBox.setEnabled`

## FComboBox.setItemSymbol

### Disponibilidade

Flash Player 6.

### Uso

```
myComboBox.setItemSymbol(symbolID)
```

### Parâmetros

*symbolID* O identificador de vinculação de símbolo para exibir o conteúdo da caixa de combinação.

### Retorna

Nada.

### Descrição

Método; registra um símbolo gráfico para exibir os itens de listagem da caixa de combinação. O valor padrão é o símbolo FComboBoxItem na biblioteca. Este método é destinado a usuários e programadores avançados.

## FComboBox.setRowCount

### Disponibilidade

Flash Player 6.

### Uso

```
myComboBox.setRowCount(rows)
```

### Parâmetros

*rows* O número máximo de linhas que a listagem suspensa pode exibir sem rolagem.

### Retorna

Nada.

### Descrição

Método; define o número de itens que podem ser vistos na listagem suspensa da caixa de combinação sem rolagem. O valor mínimo do parâmetro *rows* é 3. Se este método for chamado, o valor do parâmetro Row Count definido durante o processo de criação será cancelado.

### Exemplo

O código a seguir define o número de itens exibidos na listagem suspensa de menuMain como 4.

```
menuMain.setRowCount(4);
```

### Consulte também

FComboBox.setSize

## FComboBox.setSelectedIndex

### Disponibilidade

Flash Player 6.

### Uso

```
myComboBox.setSelectedIndex(index)
```

### Parâmetros

*index* Um número inteiro que especifica o índice do item a ser selecionado.

### Retorna

Nada.

### Descrição

Método; seleciona o item especificado e atualiza a caixa de combinação para exibir o item conforme selecionado. Se este método for chamado, o estado atual de aberto ou fechado da listagem suspensa não será alterado. Este método não pode ser utilizado se a caixa de combinação estiver desativada.

O componente ComboBox usa um índice com base no zero, onde o item no índice 0 é exibido no topo da listagem.

### Exemplo

O código a seguir seleciona o quinto item na lista de `menuMain`.

```
menuMain.setSelectedIndex(3);
```

### Consulte também

`FComboBox.setRowCount`

## FComboBox.setSize

### Disponibilidade

Flash Player 6.

### Uso

```
myComboBox.setSize(width)
```

### Parâmetros

*width* Um número inteiro que especifica a largura da caixa de combinação, em pixels.

### Retorna

Nada.

### Descrição

Método; ajusta a caixa de combinação à largura especificada. (Não é possível definir a altura do componente de uma caixa de combinação.) Use este método para redimensionar a caixa de combinação e atualizá-la durante a execução.

### Exemplo

O código a seguir define ou redimensiona a largura do `menuMain` como sendo 100 pixels.

```
menuMain.setSize(100);
```

### Consulte também

`FComboBox.setRowCount`

## FComboBox.setStyleProperty

### Disponibilidade

Flash Player 6.

### Uso

```
myComboBox.setStyleProperty(styleProperty, value)
```

### Parâmetros

*styleProperty* Uma seqüência de caracteres que especifica uma propriedade do objeto FStyleFormat.

*value* O valor definido para a propriedade.

### Retorna

Nada.

### Descrição

Método; define uma propriedade FStyleFormat para uma determinada caixa de combinação. Chamar este método para especificar uma propriedade cancela as configurações dessa propriedade no formato de estilo atribuído ao componente. Se o valor `undefined` for atribuído a uma propriedade, todos os estilos dessa propriedade serão removidos.

Para definir as propriedades FStyleFormat para vários componentes, crie um formato de estilo personalizado. Para obter mais informações sobre a criação de formatos de estilo personalizados, consulte “Personalizando cores e texto do componente” no capítulo “Usando componentes” de *Usando o Flash*.

### Exemplo

O código a seguir define a propriedade `arrow` de `comboBox1` como sendo `0x000000` (preto).

```
comboBox1.setStyleProperty("arrow", 0x000000);
```

### Consulte também

FStyleFormat (object)

## FComboBox.setValue

### Disponibilidade

Flash Player 6.

### Uso

```
myComboBox.setValue(editableText)
```

### Parâmetros

*editableText* Uma seqüência de caracteres que especifica o texto a ser exibido no campo de texto de uma caixa de combinação editável.

### Retorna

Nada.

### Descrição

Método; especifica oS texto exibido no campo de entrada na parte superior da caixa de combinação editável. Se você chamar este método, o usuário poderá ainda inserir texto no campo.

Este método só pode ser usado com caixas de combinação editáveis. Antes de chamar este método, você deve especificar `true` para o parâmetro `Editable` (cujo padrão é `false`) durante o processo de criação ou utilizar `FComboBox.setEditable` para definir o parâmetro como `true`.

### Exemplo

O código a seguir insere a sequência de caracteres `Gabino` no campo superior da caixa de combinação `surnameMenu`.

```
surnameMenu.setValue("Gabino");
```

### Consulte também

`FComboBox.getValue`

## FComboBox.sortItemsBy

### Disponibilidade

Flash Player 6.

### Uso

```
myComboBox.sortItemsBy(fieldName, order)
```

### Parâmetros

*fieldName* Uma sequência de caracteres que especifica o nome do campo usado para classificação. Este será normalmente `"label"` ou `"data"`.

*order* Uma sequência de caracteres que especifica se os itens devem ser classificados em ordem crescente (`"ASC"`) ou decrescente (`"DESC"`).

### Retorna

Nada.

### Descrição

Método; classifica os itens na caixa de combinação em ordem alfabética ou numérica, na ordem especificada, usando o nome de campo especificado. Se os itens de *fieldName* forem uma combinação de sequências de caracteres de texto e números inteiros, os itens inteiros serão apresentados primeiro. O parâmetro *fieldName* é geralmente `label` ou `data`, mas pode ser especificado qualquer valor de dado primitivo que atenda às necessidades.

### Exemplo

O código a seguir classifica os itens na caixa de combinação `surnameMenu` em ordem crescente usando os rótulos dos itens de listagem.

```
surnameMenu.sortItemsBy("label", "ASC");
```

### Consulte também

`FComboBox.addItemAt`, `FComboBox.replaceItemAt`, `FComboBox.setDataProvider`

## FListBox (component)

O componente `ListBox` no ambiente de criação `Flash` oferece recurso de arrastar e soltar para adicionar caixas de listagem roláveis de seleção única e seleção múltipla a documentos `Flash`; ele também oferece uma interface de usuário para definição de parâmetros básicos. Os métodos do componente `FListBox` permitem controlar caixas de listagem durante a execução: você pode criar caixas de listagem, controlar as caixas de listagem criadas no ambiente de criação, definir ou cancelar parâmetros básicos e definir opções adicionais de tempo de execução. Não é preciso usar um construtor para acessar os métodos de componentes.

Os métodos do componente não realizam verificação de erros de tipo, como outros objetos e ações nativos do `ActionScript`; portanto, recomenda-se a validação dos parâmetros antes de passá-los para métodos.

O componente `ListBox` tem suporte do `Flash Player 6` e de suas versões posteriores.

Para obter informações sobre o uso do componente `ListBox`, como definir parâmetros durante o processo de criação e como alterar as cores e a aparência de componentes, consulte “Personalizando cores e texto do componente” e “Personalizando aparências de componentes” no capítulo “Usando componentes” de *Usando o Flash*.

### Resumo dos métodos do componente FListBox

Método	Descrição
<code>FListBox.addItem</code>	Adiciona um novo item ao final da lista da caixa de listagem.
<code>FListBox.addItemAt</code>	Adiciona um novo item à lista da caixa de listagem no índice especificado.
<code>FListBox.setEnabled</code>	Retorna <code>true</code> se a caixa de listagem estiver ativada, <code>false</code> se estiver desativada.
<code>FListBox.getItemAt</code>	Retorna o rótulo e o valor do item no índice especificado.
<code>FListBox.getLength</code>	Retorna o número de itens na caixa de listagem.
<code>FListBox.getRowCount</code>	Retorna o número de itens visíveis na caixa de listagem.
<code>FListBox.getScrollPosition</code>	Retorna o índice do item na parte superior da caixa de listagem.
<code>FListBox.getSelectedIndex</code>	Retorna o índice do item selecionado por último.
<code>FListBox.getSelectedIndices</code>	Retorna os índices dos vários itens selecionados.
<code>FListBox.getSelectedItem</code>	Retorna o rótulo e o valor do item selecionado.
<code>FListBox.getSelectedItems</code>	Retorna o rótulo e o valor dos vários itens selecionados.
<code>FListBox.selectMultiple</code>	Retorna <code>true</code> se for permitida seleção múltipla, <code>false</code> se for permitida seleção única.
<code>FListBox.getValue</code>	Retorna o rótulo do item selecionado ou quaisquer outras informações associadas.
<code>FListBox.registerSkinElement</code>	Registra um elemento de aparência em uma propriedade.
<code>FListBox.removeAll</code>	Remove todos os itens da caixa de listagem.
<code>FListBox.removeItemAt</code>	Remove o item no índice especificado.
<code>FListBox.replaceItemAt</code>	Substitui o rótulo e os dados de um item em um índice especificado por um novo rótulo e novos dados.
<code>FListBox.setAutoHideScrollBar</code>	Determina se a barra de rolagem fica oculta ( <code>true</code> ) ou aparente ( <code>false</code> ) quando o número de itens na caixa de listagem não exigir rolagem.

Método	Descrição
<code>FListBox.setChangeHandler</code>	Atribui uma função a ser chamada todas as vezes que a seleção for alterada.
<code>FListBox.setDataProvider</code>	Associa um objeto externo à caixa de listagem.
<code>FListBox.setEnabled</code>	Especifica se a caixa de listagem está ativada ( <code>true</code> ) ou desativada ( <code>false</code> ).
<code>FListBox.setItemSymbol</code>	Registra o identificador de vinculação de um símbolo a ser usado para exibir itens na caixa de listagem.
<code>FListBox.setRowCount</code>	Retorna o número de itens exibidos na caixa de listagem.
<code>FListBox.setScrollPosition</code>	Faz a caixa de listagem rolar até que o item no índice especificado seja exibido na parte superior da lista.
<code>FListBox.setSelectedIndex</code>	Seleciona o item no índice especificado e atualiza a caixa de listagem.
<code>FListBox.setSelectedIndices</code>	Seleciona os itens nos índices especificados e atualiza a caixa de listagem.
<code>FListBox.setSelectMultiple</code>	Determina se o usuário pode selecionar mais de um item na lista ( <code>true</code> ) ou não ( <code>false</code> ).
<code>FListBox.setSize</code>	Define a largura e a altura da caixa de listagem, em pixels.
<code>FListBox.setStyleProperty</code>	Define uma única propriedade de estilo para um componente.
<code>FListBox.setWidth</code>	Define a largura da caixa de listagem, em pixels.
<code>FListBox.sortItemsBy</code>	Classifica os itens na caixa de listagem em ordem alfabética ou numérica usando o rótulo ou os dados.

## FListBox.addItem

### Disponibilidade

Flash Player 6.

### Uso

```
myListBox.addItem(label [, data])
```

### Parâmetros

*label* Uma seqüência de caracteres de texto que especifica o item a ser adicionado à lista.

*data* Um valor a ser associado ao item de listagem. Este parâmetro é opcional.

### Retorna

Nada.

### Descrição

Método; adiciona um novo item com o rótulo e os dados (opcional) especificados ao final da caixa de listagem, atualiza a caixa de listagem e redimensiona a barra de rolagem. O parâmetro *Data* pode ser qualquer objeto do Flash, seqüência de caracteres, valor Booleano, número inteiro, objeto ou clipe de filme.

Para obter melhor desempenho e menor tempo de carregamento, não adicione mais de 400 itens a cada quadro. Isso se aplica esteja você adicionando os itens a uma única caixa de listagem ou a várias.

### Exemplo

O código a seguir adiciona Lyvia à lista de itens exibidos na caixa de listagem coolGirls.

```
coolGirls.addItem("Lyvia");
```

O código a seguir adiciona o número máximo de itens recomendado em um único quadro (400 itens) a listBox1:

```
for (i=0; i<400; i++) {  
    listBox1.addItem(i);  
}
```

O código a seguir adiciona o número máximo de itens recomendado em um único quadro (400 itens) a listBox1 e a comboBox2:

```
for (i=0; i<200; i++) {  
    listBox1.addItem(i);  
    comboBox2.addItem(i);  
}
```

### Consulte também

`FListBox.addItemAt`, `FListBox.getItemAt`, `FListBox.removeItemAt`,  
`FListBox.replaceItemAt`, `FListBox.sortItemsBy`

## FListBox.addItemAt

### Disponibilidade

Flash Player 6.

### Uso

```
myListBox.addItemAt(index, label [, data])
```

### Parâmetros

*index* Um número inteiro que especifica a posição onde inserir o item.

*label* Uma sequência de caracteres de texto que especifica o rótulo do item.

*data* Um valor a ser associado ao item de listagem. Este parâmetro é opcional.

### Retorna

Nada.

### Descrição

Método; adiciona um novo item com o rótulo especificado e os dados (opcionais) associados no índice especificado e atualiza a caixa de listagem. O parâmetro Data pode ser qualquer objeto do Flash, sequência de caracteres, valor Booleano, número inteiro, objeto ou clipe de filme.

O componente ListBox usa um índice com base no zero, onde o item no índice 0 é exibido no começo da lista.

Para obter melhor desempenho e menor tempo de carregamento, não adicione mais de 400 itens a cada quadro. Isso se aplica esteja você adicionando os itens a uma única caixa de listagem ou a várias.



### Exemplo

O código a seguir adiciona o item Dave com o valor associado friend como quinto item na caixa de listagem peopleList.

```
peopleList.addItemAt(4, "Dave", friend);
```

Para obter exemplos de como carregar um grande número de itens, consulte `FListBox.addItem`.

### Consulte também

`FListBox.getSelectedItem`, `FListBox.removeItemAt`, `FListBox.replaceItemAt`,  
`FListBox.sortItemsBy`

## FListBox.setEnabled

### Disponibilidade

Flash Player 6.

### Uso

```
myListBox.setEnabled()
```

### Parâmetros

Nenhum.

### Retorna

Um valor Booleano que indica se a caixa de listagem está ativada (`true`) ou desativada (`false`).

### Descrição

Método; indica se a caixa de listagem está ativada.

### Exemplo

O código a seguir usa `setEnabled` para determinar se `listMenu` está ativado ou desativado e exibe o resultado na janela Saída.

```
trace(listMenu.setEnabled());
```

### Consulte também

`FListBox.setEnabled`

## FListBox.getItemAt

### Disponibilidade

Flash Player 6.

### Uso

```
myListBox.getItemAt(index)
```

### Parâmetros

*index* Um número inteiro que especifica o índice do item a ser recuperado.

### Retorna

Um objeto ou `undefined`.

### Descrição

Método; retorna o item no índice especificado como um objeto com as propriedades `label` e `data`. Se não houver item algum no índice especificado, este método retornará `undefined`.

O componente `Listbox` usa um índice com base no zero, onde o item no índice 0 é exibido no começo da lista.

**Exemplo**

O código a seguir retorna o rótulo do item no índice 4 em `listMenu1` na janela Saída.

```
trace(listMenu1.getItemAt(4).label);
```

O código a seguir retorna os dados ou o valor associado ao item no índice 4 em `listMenu2` na janela Saída.

```
trace(listMenu2.getItemAt(4).data);
```

O código a seguir retorna um objeto contendo o rótulo e o valor de dados associado ao item no índice 4 em `listMenu3` na janela Saída.

```
trace(listMenu3.getItemAt(4));
```

**Consulte também**

`FListBox.getSelectedItem`

## **FListBox.getLength**

**Disponibilidade**

Flash Player 6.

**Uso**

```
myListBox.getLength()
```

**Parâmetros**

Nenhum.

**Retorna**

Um inteiro.

**Descrição**

Método; retorna o número de itens na caixa de listagem.

**Exemplo**

O código a seguir retorna o número de itens em `phoneList`.

```
phoneList.getLength();
```

**Consulte também**

`FListBox.setSize`

## **FListBox.getRowCount**

**Disponibilidade**

Flash Player 6.

**Uso**

```
myListBox.getRowCount()
```

**Parâmetros**

Nenhum.

**Retorna**

Um inteiro.

**Descrição**

Método; retorna o número de linhas visíveis na caixa de listagem. Este método é útil para determinar quantas linhas são exibidas em uma caixa de listagem dimensionada em pixels.

**Exemplo**

O código a seguir retorna o número de linhas visíveis em `toyList` e define o valor para a variável `rowCount`.

```
var rowCount = toyList.getRowCount();
```

**Consulte também**

`FListBox.setRowCount`, `FListBox.setSize`

## FListBox.getScrollPosition

**Disponibilidade**

Flash Player 6.

**Uso**

```
myListBox.getScrollPosition()
```

**Parâmetros**

Nenhum.

**Retorna**

Um inteiro.

**Descrição**

Método; retorna o índice do item que está atualmente no topo da exibição da caixa de listagem.

**Exemplo**

O código a seguir retorna o índice do item no topo de `staffList`.

```
staffList.getScrollPosition();
```

**Consulte também**

`FListBox.setScrollPosition`

## FListBox.getSelectedIndex

**Disponibilidade**

Flash Player 5

**Uso**

```
myListBox.getSelectedIndex()
```

**Parâmetros**

Nenhum.

**Retorna**

Um número inteiro ou `undefined`.

**Descrição**

Método; retorna o índice do item atualmente selecionado em uma caixa de listagem de seleção única, o item selecionado por último em uma caixa de listagem de seleção múltipla, ou `undefined`, se não houver nenhum item selecionado. Para recuperar os índices de todos os itens selecionados em uma caixa de listagem de seleção múltipla, use `FListBox.getSelectedIndices`.

**Exemplo**

O código a seguir retorna o índice do item atualmente selecionado na caixa de listagem de seleção única `nationList`.

```
nationList.getSelectedIndex();
```

**Consulte também**

`FListBox.setSelectedIndices`, `FListBox.setSelectMultiple`

## FListBox.getSelectedIndices

**Disponibilidade**

Flash Player 6.

**Uso**

```
myListBox.getSelectedIndices()
```

**Parâmetros**

Nenhum.

**Retorna**

Uma matriz ou `undefined`.

**Descrição**

Método; retorna os índices dos itens atualmente selecionados em uma caixa de listagem de seleção múltipla como uma matriz, ou retorna `undefined`, caso nenhum item esteja selecionado.

**Exemplo**

O código a seguir retorna os índices dos itens atualmente selecionados na caixa de listagem de seleção múltipla `groceryList`.

```
groceryList.getSelectedIndices();
```

**Consulte também**

`FListBox.getSelectedIndex`, `FListBox.setSelectMultiple`

## FListBox.getSelectedItem

**Disponibilidade**

Flash Player 6.

**Uso**

```
myListBox.getSelectedItem()
```

**Parâmetros**

Nenhum.

**Retorna**

Um objeto ou `undefined`.

**Descrição**

Método, retorna o item atualmente selecionado como um objeto com as propriedades `label` e `data`. Se houver mais de um item selecionado, o método retornará o item selecionado por último na lista; se nenhum item estiver selecionado, o método retornará `undefined`. Para obter informações sobre todos os itens selecionados em uma caixa de listagem de seleção múltipla, use `FListBox.getSelectedItems`.

**Exemplo**

O código a seguir recupera o rótulo do item atualmente selecionado em `listBox1`.

```
trace(listBox1.getSelectedItem().label);
```

O código a seguir retorna os dados ou valores associados ao item atualmente selecionado em `listBox2`.

```
trace(listBox2.getSelectedItem().data);
```

O código a seguir retorna um objeto contendo o rótulo e o valor de dados associado ao item atualmente selecionado em `listBox3`.

```
trace(listBox3.getSelectedItem());
```

**Consulte também**

`FListBox.getItemAt`

## **FListBox.getSelectedItems**

**Disponibilidade**

Flash Player 6.

**Uso**

```
myListBox.getSelectedItems()
```

**Parâmetros**

Nenhum.

**Retorna**

Uma matriz ou `undefined`.

**Descrição**

Método; retorna os itens atualmente selecionados como uma matriz de objetos com as propriedades `label` e `data`, ou retorna `undefined`, se não houver item selecionado. Este método só pode ser usado para obter os itens selecionados em uma caixa de listagem de seleção múltipla. Para obter informações sobre o item atualmente selecionado em uma caixa de listagem de seleção única, use `FListBox.getSelectedItem`.

**Exemplo**

O código a seguir recupera os itens atualmente selecionados em `wishList` e armazena esses valores na variável `myObjArray`.

```
var myObjArray = wishList.getSelectedItems();
```

**Consulte também**

`FListBox.getSelectedItem`, `FListBox.setSelectMultiple`

## FListBox.getSelectMultiple

### Disponibilidade

Flash Player 6.

### Uso

```
myListBox.getSelectMultiple()
```

### Parâmetros

Nenhum.

### Retorna

Um valor booleano.

### Descrição

Método; indica se os usuários podem selecionar vários itens (`true`) ou somente um único item (`false`) na caixa de listagem.

### Exemplo

O código a seguir testa se `wishList` permite seleção múltipla.

```
if (wishList.getSelectMultiple()) {  
}
```

### Consulte também

`FListBox.setSelectMultiple`

## FListBox.getValue

### Disponibilidade

Flash Player 6.

### Uso

```
myListBox.getValue()
```

### Parâmetros

Nenhum.

### Retorna

O rótulo ou os dados associados ao item selecionado.

### Descrição

Método; retorna informações sobre o item atualmente selecionado na caixa de listagem. Se o item não tiver dados especificados, este método retornará o rótulo do item; se o item tiver dados associados, este método retornará os dados.

### Exemplo

O código a seguir retorna o rótulo do item selecionado em `nationList`.

```
trace(nationList.getValue());
```

### Consulte também

`FListBox.getItemAt`

# FListBox.registerSkinElement

## Disponibilidade

Flash Player 6.

## Uso

```
myListBox.registerSkinElement(element, styleProperty)
```

## Parâmetros

*element* Uma instância de clipe de filme.

*styleProperty* O nome de uma propriedade de FStyleFormat.

## Retorna

Nada.

## Descrição

Método; registra um elemento de aparência em uma propriedade de estilo. Elementos de aparência são registrados em propriedades no primeiro quadro da camada ReadMe de cada aparência na biblioteca.

Os componentes são compostos de aparências e cada aparência é composta de vários elementos de aparência, cada um dos quais pode ser registrado em uma propriedade do objeto FStyleFormat. Essas propriedades são valores atribuídos pelo formato de estilo atribuído a um componente. Como padrão, o objeto `globalStyleFormat` é atribuído a todos os componentes de interface do Flash. Esse objeto é uma instância do objeto FStyleFormat.

Use este método para registrar propriedades e elementos de aparência personalizados na interface do Flash ou aparências personalizadas de componentes editando o código no primeiro quadro da camada ReadMe de uma aparência na biblioteca.

O componente FListBox usa as aparências na pasta FListBox Skins depois que o componente for adicionado ao documento Flash.

Para obter mais informações, consulte “Personalizando aparências de componentes” no capítulo “Usando componentes” de *Usando o Flash*.

## Exemplo

O código a seguir registra o elemento de aparência personalizado `boundBox_mc` na propriedade `background` no primeiro quadro da camada ReadMe da aparência `FBoundingBox` na pasta `Global Skins` na biblioteca.

```
toysMenu.registerSkinElement(boundBox_mc, "background");
```

## Consulte também

FStyleFormat (object)

## FListBox.removeAll

### Disponibilidade

Flash Player 6.

### Uso

```
myListBox.removeAll()
```

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Método; remove todos os itens da caixa de listagem, atualiza a caixa e redimensiona a barra de rolagem.

### Exemplo

O código a seguir remove todos os itens de `wishList`.

```
wishList.removeAll();
```

### Consulte também

`FListBox.removeItemAt`

## FListBox.removeItemAt

### Disponibilidade

Flash Player 6.

### Uso

```
myListBox.removeItemAt(index)
```

### Parâmetros

*index* Um número inteiro que especifica o índice do item a ser removido.

### Retorna

Nada ou `undefined`.

### Descrição

Método; remove o item no índice especificado, atualiza os índices dos itens da lista seguintes ao item removido para refletir suas novas posições e, a seguir, atualiza a caixa de listagem e redimensiona a barra de rolagem. Se não houver item algum no índice especificado, este método retornará `undefined`.

O componente `ListBox` usa um índice com base no zero, onde o item no índice 0 é exibido no começo da lista.

### Exemplo

O código a seguir remove o quinto item na lista de `wishList`.

```
wishList.removeItemAt(4);
```

### Consulte também

`FListBox.addItemAt`



## FListBox.replaceItemAt

### Disponibilidade

Flash Player 6.

### Uso

```
myListBox.replaceItemAt(index, label [,data])
```

### Parâmetros

*index* Um número inteiro que especifica a posição de um item de listagem.

*label* Uma sequência de caracteres que especifica um novo rótulo para o item de listagem.

*data* O novo valor a ser associado ao item de listagem. Este parâmetro é opcional; se você deixar de especificá-lo, qualquer dado atualmente associado ao item permanece no lugar.

### Retorna

Nada.

### Descrição

Método; atualiza o item no índice especificado com o rótulo e os dados especificados. Se o item no índice especificado tiver um valor de dado associado e você não especificar um valor para o parâmetro *data*, o valor de dado do item de listagem não será alterado.

O componente List Box usa um índice com base no zero, onde o item no índice 0 é exibido no topo da listagem.

### Exemplo

O código a seguir atualiza o quinto item na caixa de listagem Favorites com o novo rótulo Lucky e o novo valor Cat. Se o valor de dados Cat não for especificado e o dado associado ao quinto item da lista for Dog, o valor de dados de Lucky será Dog (o que estaria errado, pois Lucky é um gato ("cat")).

```
Favoritos.replaceItemAt(4, "Lucky", "Cat");
```

### Consulte também

FListBox.addItemAt, FListBox.getItemAt

## FListBox.setAutoHideScrollBar

### Disponibilidade

Flash Player 6.

### Uso

```
myListBox.setAutoHideScrollBar(hideScroll)
```

### Parâmetros

*hideScroll* Um valor Booleano que especifica se a barra de rolagem deve ficar oculta quando não for necessária (*true*) ou deve ser sempre exibida (*false*).

### Retorna

Nada.

### Descrição

Método; especifica se a barra de rolagem deve ficar oculta quando o número de itens na caixa de listagem puder ser visualizado sem uma barra de rolagem (`true`) ou se a barra de rolagem será sempre exibida (`false`). Se este método for definido como `false` e o número de itens não exigir uma barra de rolagem, a barra de rolagem será exibida como desativada (esmaecida).

### Exemplo

O código a seguir oculta a barra de rolagem de `wishList` quando o número de itens na caixa de listagem puder ser visualizado sem rolagem.

```
wishList.setAutoHideScrollBar(true);
```

## FListBox.setChangeHandler

### Disponibilidade

Flash Player 6.

### Uso

```
myListBox.setChangeHandler(functionName, [location])
```

### Parâmetros

*functionName* Uma seqüência de caracteres que especifica o nome da função do identificador a ser executada quando a seleção na caixa de listagem é alterada. Se o parâmetro *location* não for especificado, esta função deverá estar na mesma Linha de tempo da instância do componente.

*location* Uma referência de caminho até um objeto de dados, clipe de filme ou Linha de tempo que contém a função especificada. Este parâmetro é opcional e tem como padrão a Linha de tempo pai do componente.

### Retorna

Nada.

### Descrição

Método; especifica um identificador de alteração a ser chamado quando a seleção na caixa de listagem é alterada. Você pode especificar a mesma função de identificador de alteração para mais de um componente; a função sempre aceita a instância do componente que foi alterada como um parâmetro. Se este método for chamado, valor do parâmetro Identificador de alteração especificado na criação será cancelado.

Para obter mais informações, consulte “Criando funções do identificador de alteração para componentes” no capítulo “Usando componentes” de *Usando o Flash*.

### Exemplo

O código a seguir especifica `myHandler` como a função chamada quando o valor de `listBox1` é alterado. Como o parâmetro *location* não foi especificado, `myHandler` deverá estar na mesma Linha de tempo da instância do componente.

O parâmetro `component` em `myHandler` é automaticamente preenchido com a instância de um componente (o componente que foi alterado como resultado de uma entrada do usuário e que especifica `myHandler` como seu identificador de alteração). As ações definidas em `myHandler` especificam que o rótulo do item é exibido na janela Saída quando o usuário seleciona um item na listagem.

```
listBox1.setChangeHandler("myHandler");
function myHandler(component){
    trace(listBox1.getSelectedItem().label);
}
```

Se, no exemplo acima, `myHandler` fosse uma função localizada na Linha de tempo bisavô da Linha de tempo do componente, a primeira linha de código seria da seguinte forma:

```
listBox1.setChangeHandler("myHandler", _parent._parent._parent);
```

O código a seguir cria a função `myHandler` em uma instância de `myObject` (que é da classe `Object`) e, a seguir, especifica `myHandler` como a função de `listBox1`.

```
myObject = new Object();
myObject.myHandler = function(component){
    trace(listBox1.getSelectedItem().label);
}

listBox1.setChangeHandler("myHandler", myObject);
```

## FListBox.setDataProvider

### Disponibilidade

Flash Player 6.

### Uso

```
myListBox.setDataProvider(dataProvider)
```

### Parâmetros

*dataProvider* Uma matriz de seqüências de caracteres de texto que lista itens para adicionar, uma instância do objeto `Array` que especifica os itens a serem adicionados ou uma instância da classe `DataProvider`.

### Retorna

Nada.

### Descrição

Método; registra um objeto externo (*dataProvider*) como a fonte de dados para o componente da caixa de listagem. Se *dataProvider* for uma instância do objeto `Array`, o objeto poderá especificar `label`, `data` ou ambos, pois as propriedades de objeto e o conteúdo da matriz poderão ser copiados para a caixa de listagem como rótulos, dados ou ambos. Se *dataProvider* for uma instância da classe `DataProvider`, ele deverá implementar a API `DataProvider` definida no símbolo `DataProvider` na pasta `FlashUIComponents/Core Assets/ClassTree` da biblioteca. A API `DataProvider` é somente para usuários e programadores avançados; todos os outros usuários devem usar uma matriz ou um objeto `Array`.

### Exemplo

O código a seguir especifica o objeto `Array` `writerList` como o provedor de dados para `listBox1`.

```
listBox1.setDataProvider(writerList);
```

O código a seguir cria a matriz `writerList` para exibir os rótulos dos itens listados em `listBox1`.

```
writerList = new Array();
writerList[0] = "Jody";
writerList[1] = "Mary";
writerList[2] = "Marcelle";
writerList[3] = "Dale";
writerList[4] = "Stephanie";
writerList[5] = "Barbara";
```

O código a seguir cria a matriz `itemList1`, que especifica tanto o rótulo quanto os dados para itens de listagem. Esse objeto `Array` poderia ser usado como um provedor de dados alternativo para `listBox1`.

```
itemList1 = new Array();
for (i=0; i<10; i++) {

    // criar um item real
    var myItem = new Object();
    myItem.label = "Item" + i;
    myItem.data = 75;

    // coloca-lo na matriz
    itemList1[i] = myItem;
}
```

O código a seguir especifica `editorList`, uma instância da classe `DataProvider`, como o provedor de dados para `listBox1`.

```
listBox1.setDataProvider(editorList);
```

O código a seguir cria uma nova instância da classe `DataProvider` e, a seguir, adiciona os rótulos do item usando o método `addItem` de `DataProvider`.

**Observação:** O método `addItem` é somente um método da classe `DataProvider`. Os programadores interessados em usar a classe `DataProvider` devem consultar o símbolo `DataProvider` na pasta `FlashUIComponents/CoreAssets/ClassTree` na biblioteca antes de tentar usar os métodos.

```
editorList = new DataProviderClass();
editorList.addItem("Anne");
editorList.addItem("Rosana");
editorList.addItem("Lisa");
editorList.addItem("Rebecca");
```

#### Consulte também

`FListBox.addItem`, `FListBox.replaceItemAt`, `FListBox.sortItemsBy`

## FListBox.setEnabled

### Disponibilidade

Flash Player 6.

### Uso

```
myListBox.setEnabled(enable)
```

### Parâmetros

*enable* Um valor Booleano que especifica se a caixa de listagem está ativada (`true`) ou desativada (`false`).

**Retorna**

Nada.

**Descrição**

Método; especifica se a caixa de listagem está ativada (`true`) ou desativada (`false`). Se uma caixa de seleção estiver desativada, ela não aceitará interação de mouse nem de teclado do usuário. Se esse parâmetro for omitido, o método utilizará o padrão `true`.

**Exemplo**

O código a seguir desativa `interestList`.

```
interestList.setEnabled(false);
```

**Consulte também**

`FListBox.setEnabled`

## **FListBox.setItemSymbol**

**Disponibilidade**

Flash Player 6.

**Uso**

```
myListBox.setItemSymbol(symbolID)
```

**Parâmetros**

*symbolID* O identificador de vinculação de símbolo de um símbolo gráfico para exibir o conteúdo da caixa de listagem.

**Retorna**

Nada.

**Descrição**

Método; registra um símbolo gráfico para exibir os itens na caixa de listagem. O valor padrão é o símbolo `FListBoxItem` na biblioteca. Este método é destinado a usuários e programadores avançados.

## **FListBox.setRowCount**

**Disponibilidade**

Flash Player 6.

**Uso**

```
myListBox.setRowCount(rows)
```

**Parâmetros**

*rows* O número máximo de linhas exibidas na caixa de listagem.

**Retorna**

Nada.

**Descrição**

Método; retorna o número de itens exibidos na caixa de listagem. Se você usar este método, utilize `FListBox.setWidth`, e não `FListBox.setSize`, para definir a largura da caixa de listagem. Chamar `FListBox.setSize` cancela o valor do parâmetro Row Count definido durante o processo de criação. Portanto, se você chamar este método depois de chamar `FListBox.setRowCount`, seu filme desconsiderará a configuração de `rowCount` e definirá a altura da caixa de listagem em pixels.

**Exemplo**

O código a seguir define o número de itens exibidos em `toyList` como 4.

```
toyList.setRowCount(4);
```

**Consulte também**

`FListBox.getRowCount`, `FListBox.setSize`

## **FListBox.setScrollPosition**

**Disponibilidade**

Flash Player 6.

**Uso**

```
myListBox.setScrollPosition(index)
```

**Parâmetros**

*index* Um número inteiro que especifica o índice do item a ser exibido na parte superior da caixa de listagem.

**Retorna**

Nada.

**Descrição**

Método; faz a caixa de listagem rolar até que o item especificado seja exibido na parte superior da caixa.

O componente `ListBox` usa um índice com base no zero, onde o item no índice 0 é exibido no começo da lista.

**Exemplo**

O código a seguir exibe o quinto item em `toyList` no topo da listagem.

```
toyList.setScrollPosition(4);
```

**Consulte também**

`FListBox.getScrollPosition`

## FListBox.setSelectedIndex

### Disponibilidade

Flash Player 6.

### Uso

```
myListBox.setSelectedIndex(index)
```

### Parâmetros

*index* Um número inteiro que especifica o índice do item a ser selecionado na caixa de listagem.

### Retorna

Nada.

### Descrição

Método; seleciona o item no índice especificado e atualiza a caixa de listagem.

O componente ListBox usa um índice com base no zero, onde o item no índice 0 é exibido no começo da lista.

### Exemplo

O código a seguir seleciona o quinto item na caixa de listagem.

```
toyList.setSelectedIndex(4);
```

### Consulte também

FListBox.getSelectedIndex, FListBox.getSelectedIndices

## FListBox.setSelectedIndices

### Disponibilidade

Flash Player 6.

### Uso

```
myListBox.setSelectedIndices(indexArray)
```

### Parâmetros

*indexArray* Uma matriz de índices de itens a ser selecionada na caixa de listagem.

### Retorna

Nada.

### Descrição

Método; seleciona os itens especificados na matriz de índices e atualiza a caixa de listagem.

O componente ListBox usa um índice com base no zero, onde o item no índice 0 é exibido no começo da lista.

### Exemplo

O código a seguir cria uma matriz denominada *myArray* que especifica os itens que deverão ser selecionados em *toyList* e, a seguir, passa a matriz para o método *setSelectedIndices*.

```
var myArray = new Array (1,4,5,7);  
toyList.setSelectedIndices(myArray);
```

### Consulte também

FListBox.getSelectedIndices

## FListBox.setSelectedMultiple

### Disponibilidade

Flash Player 6.

### Uso

```
myListBox.setSelectedMultiple(multipleSelect)
```

### Parâmetros

*multipleSelect* Um valor Booleano que especifica o modo de seleção múltipla (`true`) ou o modo de seleção única (`false`).

### Retorna

Nada.

### Descrição

Método; especifica se os usuários podem selecionar vários itens (`true`) ou somente um único item (`false`) na caixa de listagem. A configuração padrão é `false`. Se este método for chamado, o valor do parâmetro `Select Multiple` definido durante o processo de criação será cancelado.

### Exemplo

O código a seguir ativa a seleção múltipla para `toyList`.

```
toyList.setSelectedMultiple(true);
```

## FListBox.setSize

### Disponibilidade

Flash Player 6.

### Uso

```
myListBox.setSize(width, height)
```

### Parâmetros

*width* Um número inteiro que especifica a largura da caixa de listagem, em pixels.

*height* Um número inteiro que especifica a altura da caixa de listagem, em pixels.

### Retorna

Nada.

### Descrição

Método; ajusta o tamanho da caixa de listagem durante a execução segundo a largura e a altura especificadas. Se este método for chamado, o valor do parâmetro `Row Count` definido durante o processo de criação será cancelado. Portanto, se você chamar este método depois de chamar `FListBox.setRowCount`, seu filme definirá a altura da caixa de listagem em pixels e desconsiderará a configuração de `rowCount`. Para definir a largura de uma caixa de listagem ao utilizar `setRowCount`, use `FListBox.setWidth`.

### Exemplo

O código a seguir faz com que `phoneList` passe a medir 200 pixels de largura e 50 pixels de altura.

```
phoneList.setSize(200, 50);
```

### Consulte também

`FListBox.setRowCount`, `FListBox.setWidth`



## FListBox.setStyleProperty

### Disponibilidade

Flash Player 6.

### Uso

```
myListBox.setStyleProperty(styleProperty, value)
```

### Parâmetros

*styleProperty* Uma seqüência de caracteres que especifica uma propriedade do objeto FStyleFormat.

*value* O valor definido para a propriedade.

### Retorna

Nada.

### Descrição

Método; define uma propriedade FStyleFormat para uma determinada caixa de listagem. Chamar este método para especificar uma propriedade cancela as configurações dessa propriedade no formato de estilo atribuído ao componente. Se o valor `undefined` for atribuído a uma propriedade, todos os estilos dessa propriedade serão removidos.

Para definir as propriedades FStyleFormat para vários componentes, crie um formato de estilo personalizado. Para obter mais informações sobre a criação de formatos de estilo personalizados, consulte “Personalizando cores e texto do componente” no capítulo “Usando componentes” de *Usando o Flash*.

### Exemplo

O código a seguir define a propriedade `shadow` de `listBox1` como `0x000000` (preto).

```
listBox1.setStyleProperty("shadow", 0x000000);
```

### Consulte também

FStyleFormat (object)

## FListBox.setWidth

### Disponibilidade

Flash Player 6.

### Uso

```
myListBox.setWidth(width)
```

### Parâmetros

*width* Um número inteiro que especifica a largura da caixa de listagem, em pixels.

### Retorna

Nada.

### Descrição

Método; especifica a largura da caixa de listagem, em pixels. Este método é útil para definir a largura da caixa de listagem quando `FListBox.setRowCount` for usado para determinar a altura.

### Exemplo

O código a seguir define a largura da caixa de listagem `toyList` como 500 pixels.

```
toyList.setWidth(500);
```

### Consulte também

`FListBox.setSize`

## FListBox.sortItemsBy

### Disponibilidade

Flash Player 6.

### Uso

```
myListBox.sortItemsBy(fieldName, order)
```

### Parâmetros

*fieldName* Uma sequência de caracteres que especifica o nome do campo usado para classificação. Este será normalmente "label" ou "data".

*order* Uma sequência de caracteres que especifica se os itens devem ser classificados em ordem crescente ("ASC") ou decrescente ("DESC").

### Retorna

Nada.

### Descrição

Método; classifica os itens na caixa de listagem em ordem alfabética ou numérica, na ordem especificada, usando o nome de campo especificado. Se os itens de *fieldName* forem uma combinação de sequências de caracteres de texto e números inteiros, os itens inteiros serão apresentados primeiro. O parâmetro *fieldName* é geralmente "label" ou "data", mas os usuários e programadores avançados podem especificar qualquer primitivo que atenda às suas necessidades.

### Exemplo

O código a seguir classifica os itens na caixa de listagem `surnameMenu` em ordem crescente usando os rótulos dos itens de listagem.

```
surnameMenu.sortItemsBy("label", "ASC");
```

### Consulte também

`FListBox.addItemAt`, `FListBox.replaceItemAt`

## **\_focusrect**

### **Disponibilidade**

Flash Player 4.

### **Uso**

```
_focusrect = Booleano;
```

### **Descrição**

Propriedade (global); especifica se é exibido um retângulo amarelo em volta do botão que tem foco de teclado. O valor padrão, *true*, exibe um retângulo amarelo em volta do botão ou campo de texto com foco no momento quando o usuário pressiona a tecla Tab para navegar pelos objetos em um filme. Especifique *false* se você não desejar exibir o retângulo amarelo. Essa é uma propriedade global que pode ser cancelada para instâncias específicas.

### **Consulte também**

Button.\_focusrect

## **for**

### **Disponibilidade**

Flash Player 5.

### **Uso**

```
for(início; condição; próxima) {  
    comando(s);  
}
```

### **Parâmetros**

*início* Uma expressão a ser avaliada antes do início da seqüência de loop, geralmente uma expressão de atribuição. O comando *var* também é permitido para este parâmetro.

*condição* Uma expressão 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*.

*próxima* Uma expressão que é avaliada após cada iteração do loop; geralmente uma expressão de atribuição com os operadores ++ (aumento) ou -- (diminuição).

*comando(s)* Uma instrução ou instruções a ser(em) executada(s) no corpo do loop.

### **Descrição**

Ação; um construtor de loop que avalia a expressão *início* (inicializar) uma vez e começa a seqüência 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* nem *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.

### 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;  
    trace(array[i]);  
}
```

Os seguintes resultados são exibidos na janela Saída:

```
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), `--` (decrement), `for..in`, `var`

## for..in

### Disponibilidade

Flash Player 5.

### Uso

```
for(variableIterant in objeto){  
    comando(s);  
}
```

### Parâmetros

*variableIterant* 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 repetido.

*comando(s)* Uma instrução a ser executada para cada iteração.

### Retorna

Nada.

### Descrição

Ação; realiza um loop pelas propriedades de um objeto ou elemento em uma matriz e executa o *comando* para cada propriedade de um objeto.

Algumas propriedades não podem ser enumeradas pelas ações `for` nem `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.

A construção `for..in` faz a iteração das propriedades de objetos na cadeia protótipo do objeto iterado. Se o protótipo do filho `for` `pai`, a iteração das propriedades do filho com `for..in` também fará a iteração das propriedades do `pai`.

A ação `for..in` enumera todos os objetos na cadeia protótipo de um objeto. As propriedades do objeto são enumeradas primeiro, a seguir, as propriedades de seu protótipo imediato, a seguir, as propriedades do protótipo do protótipo e assim por diante. A ação `for..in` não enumera o mesmo nome de propriedade duas vezes. Se o objeto filho tiver um pai protótipo e ambos contiverem a propriedade `prop`, a ação `for..in` iniciada sobre o filho enumerará `prop` a partir do filho, mas ignorará a que está no pai.

### Exemplo

O exemplo a seguir mostra o uso de `for..in` para iteração das 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 um tipo específico 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 pai de vários filhos, `_RedRadioButton`, `_GreenRadioButton` e `_BlueRadioButton`.

```
for (var name in RadioButtonGroup) {
    RadioButtonGroup[name].gotoAndStop(2);
}
```

## FPushButton (component)

O componente `PushButton` no ambiente de criação `Flash` oferece recurso de arrastar e soltar para adicionar botões a documentos `Flash`; ele também oferece uma interface de usuário para definição de parâmetros básicos. Os métodos do componente `FPushButton` permitem controlar botões durante a execução: você pode criar botões, controlar botões criados no ambiente de criação, definir ou cancelar parâmetros básicos e definir opções adicionais de tempo de execução. Não é preciso usar um construtor para acessar os métodos de componentes.

O componente `PushButton` aceita todas as interações padrão de mouse e teclado. Você pode usar os métodos de `FPushButton` para especificar uma função do identificador para os botões de ação, desativar ou ativar botões e redimensionar botões sem distorção durante a execução.

Os métodos do componente não realizam verificação de erros de tipo, como outros objetos e ações nativos do ActionScript; portanto, recomenda-se a validação dos parâmetros antes de passá-los para métodos.

O componente PushButton tem suporte do Flash Player 6 e de suas versões posteriores.

Para obter informações sobre o uso do componente PushButton, como definir parâmetros durante o processo de criação e como alterar as cores e a aparência de componentes, consulte “Personalizando cores e texto do componente” e “Personalizando aparências de componentes” no capítulo “Usando componentes” de *Usando o Flash*.

## Resumo dos métodos do componente FPushButton

Método	Descrição
<code>FPushButton.setEnabled</code>	Retorna <code>true</code> se o botão estiver ativado, <code>false</code> se estiver desativado.
<code>FPushButton.getLabel</code>	Retorna o rótulo do botão como uma sequência de caracteres.
<code>FPushButton.registerSkinElement</code>	Registra um elemento de aparência em uma propriedade.
<code>FPushButton.setClickHandler</code>	Especifica a função chamada quando o usuário libera o botão.
<code>FPushButton.setEnabled</code>	Determina se o botão está ativado ou desativado.
<code>FPushButton.setLabel</code>	Define o rótulo do botão durante a execução.
<code>FPushButton.setSize</code>	Define a altura e a largura do botão, em pixels.
<code>FPushButton.setStyleProperty</code>	Define uma única propriedade de estilo para um componente.

## FPushButton.setEnabled

### Disponibilidade

Flash Player 6.

### Uso

```
myPushButton.setEnabled()
```

### Parâmetros

Nenhum.

### Retorna

Um valor booleano.

### Descrição

Método; retorna `true` se a instância do botão de ação estiver ativada, `false` se estiver desativada.

### Exemplo

O código a seguir retorna o estado ativado do botão de ação `submit` na janela Saída.

```
trace(submit.setEnabled());
```

### Consulte também

`FPushButton.setEnabled`

## FPushButton.getLabel

### Disponibilidade

Flash Player 6.

### Uso

```
myPushButton.getLabel()
```

### Parâmetros

Nenhum.

### Retorna

Uma seqüência de caracteres.

### Descrição

Método; retorna o rótulo de texto no botão de ação como uma seqüência de caracteres.

### Exemplo

O código a seguir retorna o rótulo do botão de ação `buttonPage1` na janela Saída.

```
trace(buttonPage1.getLabel());
```

### Consulte também

`FPushButton.setLabel`

## FPushButton.registerSkinElement

### Disponibilidade

Flash Player 6.

### Uso

```
myPushButton.registerSkinElement(element, styleProperty)
```

### Parâmetros

*element* Uma instância de clipe de filme.

*styleProperty* O nome de uma propriedade de `FStyleFormat`.

### Retorna

Nada.

### Descrição

Método; registra um elemento de aparência em uma propriedade de estilo. Elementos de aparência são registrados em propriedades no primeiro quadro da camada ReadMe de cada aparência na biblioteca.

Os componentes são compostos de aparências e cada aparência é composta de vários elementos de aparência, cada um dos quais pode ser registrado em uma propriedade do objeto `FStyleFormat`. Essas propriedades são valores atribuídos pelo formato de estilo atribuído a um componente. Como padrão, o objeto `globalStyleFormat` é atribuído a todos os componentes de interface do Flash. Esse objeto é uma instância do objeto `FStyleFormat`.

Use este método para registrar propriedades e elementos de aparência personalizados na interface do Flash ou aparências personalizadas de componentes editando o código no primeiro quadro da camada ReadMe de uma aparência na biblioteca.

O componente `FPushButton` usa as aparências na pasta `FPushButton Skins` e a aparência `FLabel` na pasta `Global Skins` depois que você adiciona o componente ao documento Flash.

Para obter mais informações, consulte “Personalizando aparências de componentes” no capítulo “Usando componentes” de *Usando o Flash*.

#### Exemplo

O código a seguir registra o elemento de aparência personalizado `newFace_mc` na propriedade `face` no primeiro quadro da camada `ReadMe` da aparência `FLabel`. A aparência `FLabel` está na pasta `Component Skins/Global` na biblioteca.

```
submitButton.registerSkinElement(newFace_mc, "face");
```

#### Consulte também

`FStyleFormat` (object)

## FPushButton.setClickHandler

#### Disponibilidade

Flash Player 6.

#### Uso

```
myPushButton.setClickHandler(functionName, [location])
```

#### Parâmetros

*functionName* Uma sequência de caracteres que especifica o nome da função do identificador a ser executada quando o usuário libera o botão de ação. Se o parâmetro *location* não for especificado, esta função deverá estar na mesma Linha de tempo da instância do componente.

*location* Uma referência de caminho até um objeto de dados, clipe de filme ou Linha de tempo que contém a função especificada. Este parâmetro é opcional e tem como padrão a Linha de tempo pai do componente.

#### Retorna

Nada.

#### Descrição

Método; especifica a função do identificador a ser chamada quando o usuário libera o botão de ação. Você pode especificar a mesma função de identificador para mais de um componente; a função sempre aceita a instância do componente que foi alterada como um parâmetro. Se este método for chamado, o valor do parâmetro Identificador de clique especificado na criação será cancelado.

Para obter mais informações, consulte “Criando funções do identificador de alteração para componentes” no capítulo “Usando componentes” de *Usando o Flash*.



### Exemplo

O código a seguir especifica `onClick` como a função chamada quando o valor de `button1` é alterado. Como o parâmetro *location* não foi especificado, `onClick` deverá estar na mesma Linha de tempo da instância do componente. O parâmetro `component` em `onClick` é automaticamente preenchido com a instância de um componente (o componente que foi alterado como resultado de uma entrada do usuário e que especifica `onClick` como seu identificador de alteração). As ações definidas em `onClick` especificam que quando o usuário libera um botão, o rótulo do botão é exibido na janela Saída.

```
button1.setClickHandler("onClick");  
  
function onClick(component){  
    trace(component._name);  
}
```

Se, no exemplo acima, `onClick` fosse uma função localizada na Linha de tempo bisavô da Linha de tempo do componente, a primeira linha de código seria da seguinte forma:

```
button1.setChangeHandler("onClick", _parent._parent._parent);
```

O código a seguir cria a função `onClick` em uma instância de `myObject` (que é da classe `Object`) e, a seguir, especifica `onClick` como a função de `button1`.

```
myObject = new Object();  
myObject.onClick = function(component){  
    trace(component._name);  
}  
  
button1.setChangeHandler("onClick", myObject);
```

## FPushButton.setEnabled

### Disponibilidade

Flash Player 6.

### Uso

```
myPushButton.setEnabled(enable)
```

### Parâmetros

*enable* Um valor Booleano que especifica se o botão de ação está ativado (`true`) ou desativado (`false`).

### Retorna

Nada.

### Descrição

Método; determina se o botão de ação está ativado. Um botão de ação desativado não aceita interação de mouse nem de teclado do usuário, e o texto sobre ele fica esmaecido. A omissão do parâmetro é o mesmo que passar `true`.

### Exemplo

O código a seguir desativa `button1`.

```
button1.setEnabled(false);
```

### Consulte também

`FPushButton.setEnabled`

## FPushButton.setLabel

### Disponibilidade

Flash Player 6.

### Uso

```
myPushButton.setLabel(label)
```

### Parâmetros

*label* Uma sequência de caracteres que contém o texto a ser exibido no botão de ação.

### Retorna

Nada.

### Descrição

Método; aplica um rótulo de texto ao botão de ação durante a execução. Se este método for chamado, o valor do parâmetro `label` especificado na criação será cancelado.

### Exemplo

O código a seguir aplica o rótulo `Cleveland Rocks!` a `voteButton`.

```
voteButton.setLabel("Cleveland Rocks!");
```

### Consulte também

`FPushButton.getLabel`

## FPushButton.setSize

### Disponibilidade

Flash Player 6.

### Uso

```
myPushButton.setSize(width, height)
```

### Parâmetros

*width* Um número inteiro que especifica a largura do botão de ação, em pixels.

*height* Um número inteiro que especifica a altura do botão de ação, em pixels.

### Retorna

Nada.

### Descrição

Método; define a largura e a altura do botão de ação durante a execução. Se este método for chamado, todo dimensionamento aplicado durante o processo de criação será cancelado. Para obter mais informações, consulte “Dimensionando componentes PushButton” do capítulo “Usando componentes” de *Usando o Flash*.

### Exemplo

O código a seguir redimensiona `submitButton` para 100 x 50 pixels durante a exibição.

```
submitButton.setSize(100, 50);
```

## FPushButton.setStyleProperty

### Disponibilidade

Flash Player 6.

### Uso

```
myPushButton.setStyleProperty(styleProperty, value)
```

### Parâmetros

*styleProperty* Uma seqüência de caracteres que especifica uma propriedade do objeto `FStyleFormat`.

*value* O valor definido para a propriedade.

### Retorna

Nada.

### Descrição

Método; define uma propriedade `FStyleFormat` para um determinado botão de ação. Chamar este método para especificar uma propriedade cancela as configurações dessa propriedade no formato de estilo atribuído ao componente. Se o valor `undefined` for atribuído a uma propriedade, todos os estilos dessa propriedade serão removidos.

Para definir as propriedades `FStyleFormat` para vários componentes, crie um formato de estilo personalizado. Para obter mais informações sobre a criação de formatos de estilo personalizados, consulte “Personalizando cores e texto do componente” no capítulo “Usando componentes” de *Usando o Flash*.

### Exemplo

O código a seguir define a propriedade `face` do `submitButton` como `0xffffffff` (branco).

```
submitButton.setStyleProperty("face", 0xffffffff);
```

### Consulte também

`FStyleFormat` (object)

## FRadioButton (component)

Os botões de opção são grupos de botões selecionáveis dos quais somente um botão pode ser selecionado por vez. O componente `RadioButton` no ambiente de criação Flash oferece recurso de arrastar e soltar para adicionar grupos de botões de opção a documentos Flash; ele também oferece uma interface de usuário para definição de parâmetros básicos. Os métodos do componente `FRadioButton` permitem controlar botões de opção durante a execução: você pode criar botões, controlar botões de opção criados no ambiente de criação, definir ou cancelar parâmetros básicos e definir mais opções de tempo de execução. Não é preciso usar um construtor para acessar os métodos de componentes.

O componente `RadioButton` tem suporte do Flash Player 6 e de suas versões posteriores.

Os métodos do componente não realizam verificação de erros de tipo, como outros objetos e ações nativos do `ActionScript`; portanto, recomenda-se a validação dos parâmetros antes de passá-los para métodos.

Para obter informações sobre o uso do componente `RadioButton`, como definir parâmetros durante o processo de criação e como alterar as cores e a aparência de componentes, consulte “Personalizando cores e texto do componente” e “Personalizando aparências de componentes” no capítulo “Usando componentes” de *Usando o Flash*.

## Resumo dos métodos do componente FRadioButton

Método	Descrição
<code>FRadioButton.getData</code>	Retorna um valor de dados para uma instância do botão de opção.
<code>FRadioButton.setEnabled</code>	Retorna <code>true</code> se o botão de opção estiver ativado, <code>false</code> se estiver desativado.
<code>FRadioButton.getLabel</code>	Retorna o rótulo do botão de opção como uma sequência de caracteres.
<code>FRadioButton.getState</code>	Retorna o estado selecionado de uma instância do botão de opção.
<code>FRadioButton.getValue</code>	Retorna o valor de dados do botão de opção selecionado em um grupo, ou retorna o rótulo, caso nenhum dado tenha sido especificado.
<code>FRadioButton.registerSkinElement</code>	Registra um elemento de aparência em uma propriedade de estilo.
<code>FRadioButton.setChangeHandler</code>	Especifica uma função a ser chamada quando a seleção do botão de opção é alterada.
<code>FRadioButton.setData</code>	Define os dados associados a uma instância do botão de opção.
<code>FRadioButton.setEnabled</code>	Determina se o botão de opção está ativado ou desativado.
<code>FRadioButton.setGroupName</code>	Especifica um nome de grupo para uma instância de botão de opção ou define um novo nome para um grupo de botões de opção.
<code>FRadioButton.setLabel</code>	Aplica um rótulo ao botão de opção durante a execução.
<code>FRadioButton.setLabelPlacement</code>	Especifica se o rótulo é exibido à esquerda ou à direita do botão de opção.
<code>FRadioButton.setSize</code>	Define a largura do botão de opção, em pixels.
<code>FRadioButton.setState</code>	Define o estado selecionado da instância do botão de opção.
<code>FRadioButton.setStyleProperty</code>	Define uma única propriedade de estilo para uma instância do componente.
<code>FRadioButton.setValue</code>	Seleciona um botão de opção em um grupo de botões de opção durante a execução.

## FRadioButton.getData

### Disponibilidade

Flash Player 6.

### Uso

```
myRadioButton.getData()
```

### Parâmetros

Nenhum.

### Retorna

Uma sequência de caracteres.

**Descrição**

Método; retorna os dados associados à instância de botão de opção especificada. Use `FRadioButton.getValue` para obter os dados associados ao botão de opção selecionado em um grupo de botões de opção.

**Exemplo**

O código a seguir retorna os dados associados ao botão de opção `flashRadio` na janela Saída.

```
trace(flashRadio.getData());
```

**Consulte também**

`FRadioButton.setData`

## FRadioButton.setEnabled

**Disponibilidade**

Flash Player 6.

**Uso**

```
myRadioButton.setEnabled()  
myRadioButtonGroup.setEnabled()
```

**Parâmetros**

Nenhum.

**Retorna**

Um valor Booleano ou `undefined`.

**Descrição**

Método; indica se uma instância de botão de opção ou grupo de botões de opção está ativada(o).

Uso 1: Indica se *myRadioButton* está ativado (`true`) ou desativado (`false`).

Uso 2: Indica se os botões em *myRadioButtonGroup* estão ativados (`true`) ou desativados (`false`). Se alguns dos botões no grupo estiverem ativados e alguns desativados, o método retornará `undefined`.

**Exemplo**

O código a seguir retorna o estado ativado de `radio1` na janela Saída.

```
trace(radio1.setEnabled());
```

**Consulte também**

`FRadioButton.setEnabled`

## FRadioButton.getLabel

**Disponibilidade**

Flash Player 6.

**Uso**

```
myRadioButton.getLabel()
```

**Parâmetros**

Nenhum.

**Retorna**

Uma seqüência de caracteres.

**Descrição**

Método; retorna o rótulo do botão de opção especificado como uma seqüência de caracteres. Não é possível usar este método para obter rótulos de um grupo de botões de opção; a sintaxe *radioButtonGroup.getLabel* não é válida.

**Exemplo**

O código a seguir retorna o rótulo da instância *radio2* na janela Saída.

```
trace(radio2.getLabel());
```

**Consulte também**

*FRadioButton.setLabel*

## FRadioButton.getState

**Disponibilidade**

Flash Player 6.

**Uso**

```
myRadioButton.getState()
```

**Parâmetros**

Nenhum.

**Retorna**

Um valor Booleano que indica o estado selecionado do botão de opção.

**Descrição**

Método; retorna um valor Booleano que indica se *myRadioButton* está selecionado (*true*) ou não (*false*).

**Exemplo**

O código a seguir retorna o estado selecionado do botão de opção *radio1* na janela Saída.

```
trace(radio1.getState());
```

**Consulte também**

*FRadioButton.setState*

## FRadioButton.getValue

**Disponibilidade**

Flash Player 6.

**Uso**

```
myRadioButtonGroup.getValue()
```

**Parâmetros**

Nenhum.

**Retorna**

Uma seqüência de caracteres ou *undefined*.

### Descrição

Método; retorna os dados associados ao botão de opção selecionado em *myRadioButtonGroup*, ou o rótulo do botão de opção, caso nenhum dado tenha sido especificado. Se nenhum botão tiver sido selecionado, o método retornará *undefined*.

### Exemplo

O código a seguir retorna os dados associados ao botão de opção selecionado no grupo *radioGroup1* na janela Saída.

```
trace(radioGroup1.getValue());
```

### Consulte também

*FRadioButton.setValue*

## FRadioButton.registerSkinElement

### Disponibilidade

Flash Player 6.

### Uso

```
myRadioButton.registerSkinElement(element, styleProperty)
```

### Parâmetros

*element* Uma instância de clipe de filme.

*styleProperty* O nome de uma propriedade de *FStyleFormat*.

### Retorna

Nada.

### Descrição

Método; registra um elemento de aparência em uma propriedade de estilo. Elementos de aparência são registrados em propriedades no primeiro quadro da camada ReadMe de cada aparência na biblioteca.

Os componentes são compostos de aparências e cada aparência é composta de vários elementos de aparência, cada um dos quais pode ser registrado em uma propriedade do objeto *FStyleFormat*. Essas propriedades são valores atribuídos pelo formato de estilo atribuído a um componente. Como padrão, o objeto *globalStyleFormat* é atribuído a todos os componentes de interface do Flash. Esse objeto é uma instância do objeto *FStyleFormat*.

Use este método para registrar propriedades e elementos de aparência personalizados na interface do Flash ou aparências personalizadas de componentes editando o código no primeiro quadro da camada ReadMe de uma aparência na biblioteca.

O componente *FRadioButton* usa as aparências na pasta *FPushButton Skins* e a aparência *FLabel* na pasta *Global Skins* depois que você adiciona o componente ao documento Flash.

Para obter mais informações, consulte “Personalizando aparências de componentes” no capítulo “Usando componentes” de *Usando o Flash*.

### Exemplo

O código a seguir registra o elemento de aparência personalizado `myDot_mc` na propriedade `radioDot` de `FStyleFormat` no arquivo `ReadMe` da aparência `frb_dot` localizada na pasta `FRadioButton Skins` na biblioteca.

```
radio1.registerSkinElement(myDot_mc, "radioDot");
```

### Consulte também

`FStyleFormat` (object)

## FRadioButton.setChangeListener

### Disponibilidade

Flash Player 6.

### Uso

```
myRadioButton.setChangeListener(functionName, [location])
```

```
myRadioButtonGroup.setChangeListener(functionName, [location])
```

### Parâmetros

*functionName* Uma seqüência de caracteres que especifica o nome da função do identificador a ser executada quando o valor de um botão de opção é alterado. Se o parâmetro *location* não for especificado, esta função deverá estar na mesma Linha de tempo da instância do componente.

*location* Uma referência a um objeto de dados, clipe de filme ou Linha de tempo que contém a função especificada. Este parâmetro é opcional e tem como padrão a Linha de tempo pai do componente.

### Retorna

Nada.

### Descrição

Método; especifica uma função do identificador de alteração a ser chamada quando a seleção do botão de opção é alterada. Você pode especificar a mesma função de identificador de alteração para mais de um componente; a função sempre aceita a instância do componente que foi alterada como um parâmetro. Se este método for chamado, o valor do parâmetro Identificador de alteração especificado na criação será cancelado.

Uso 1: Especifica a função a ser chamada se a instância do botão de opção *myRadioButton* for marcada ou desmarcada.

Uso 2: Especifica a função a ser chamada se o botão de opção selecionado no grupo *radioButtonGroup* for alterado. Especificar uma função para um grupo de botões de opção equivale a especificar a mesma função para cada um dos botões de opção nesse grupo separadamente com *myRadioButton.setChangeListener*.

Para obter mais informações, consulte “Criando funções do identificador de alteração para componentes” no capítulo “Usando componentes” de *Usando o Flash*.

### Exemplo

Uso 1: O código a seguir especifica *myHandler* como a função chamada quando *radio1* é selecionado.

```
radio1.setChangeListener("myHandler");
```



Uso 2: O código a seguir especifica `onChange` como a função chamada quando um botão de opção no grupo `radioGroup1` é selecionado.

```
radioGroup1.setChangeHandler("onChange");
```

O código a seguir especifica `onChange` como a função chamada quando o usuário seleciona um botão de opção em `radioGroup1`. Como o parâmetro *location* não foi especificado, `onChange` deverá estar na mesma Linha de tempo da instância do componente. O parâmetro `component` em `onChange` é automaticamente definido com o componente (o componente que foi alterado como resultado de uma entrada do usuário e que especifica `onChange` como seu identificador de alteração) — nesse caso, um botão de opção no grupo. As ações definidas em `onChange` especificam que quando o usuário seleciona um botão de opção, o nome da instância é exibido na janela Saída.

```
radioGroup1.setChangeHandler("onChange");  
function onChange(component){  
    trace(component._name);  
}
```

Se, no exemplo acima, `onChange` fosse uma função localizada na Linha de tempo bisavô da Linha de tempo do componente, a primeira linha de código seria da seguinte forma:

```
radioGroup1.setChangeHandler("onChange", _parent._parent._parent);
```

O código a seguir cria a função `onChange` em uma instância de `myObject` (que é da classe `Object`) e, a seguir, especifica `onChange` como a função de `radioGroup1`.

```
myObject = new Object();  
myObject.onChange = function(component){  
    trace(component._name);  
}  
  
radioGroup1.setChangeHandler("onChange", myObject);
```

## FRadioButton.setData

### Disponibilidade

Flash Player 6.

### Uso

```
myRadioButton.setData("data")
```

### Parâmetros

*data* Os dados a serem associados à instância do botão de opção.

### Retorna

Nada.

### Descrição

Método; especifica os dados a serem associados à instância do botão de opção. Se este método for chamado, o valor do parâmetro `data` definido durante o processo de criação será cancelado.

### Exemplo

O código a seguir especifica os dados `ActionScript` para a instância do botão de opção `flashRadio`.

```
flashRadio.setData("ActionScript");
```

### Consulte também

`FRadioButton.getData`, `FRadioButton.setValue`

## FRadioButton.setEnabled

### Disponibilidade

Flash Player 6.

### Uso

```
myRadioButton.setEnabled(enable)
```

```
myRadioButtonGroup.setEnabled(enable)
```

### Parâmetros

*enable* Um valor Booleano que especifica se um botão de opção ou todos os botões em um grupo está(estão) ativado(s) (*true*) ou desativado(s) (*false*).

### Retorna

Nada.

### Descrição

Método; ativa e desativa botões de opção durante a execução.

Uso 1: Especifica se *myRadioButton* está ativado (*true*) ou desativado (*false*).

Uso 2: Especifica se todos os botões de opção com o nome de grupo *radioButtonGroup* estão ativados (*true*) ou desativados (*false*). Chamar este método sem passar um parâmetro é o mesmo que passar o parâmetro *true*.

### Exemplo

Uso 1: O código a seguir desativa somente o botão de opção *radiol* sem desativar os outros botões no grupo.

```
radiol.setEnabled(false);
```

Uso 2: O código a seguir desativa todos os botões de opção no grupo *radioGroup1*.

```
radioGroup1.setEnabled(false);
```

### Consulte também

*FRadioButton.setEnabled*

## FRadioButton.setGroupName

### Disponibilidade

Flash Player 6.

### Uso

```
myRadioButton.setGroupName(groupName)
```

```
myRadioButtonGroup.setGroupName(groupName)
```

### Parâmetros

*groupName* Uma sequência de caracteres que especifica o nome de um grupo de botões de opção.

### Retorna

Nada.

### Descrição

Método; aplica um nome de grupo a uma instância de botão de opção ou grupo de botões de opção durante a execução. Se este método for chamado, o valor do parâmetro Group Name definido durante o processo de criação será cancelado.

Uso 1: Especifica *myRadioButton* como integrante do grupo de botões de opção *groupName*.

Uso 2: Especifica um novo nome de grupo para todos os botões de opção em *myRadioButtonGroup*.

### Exemplo

Uso 1: O código a seguir especifica *Colors* como o nome do grupo para a instância de botão de opção *radioRed*.

```
radioRed.setGroupName("Colors");
```

Uso2: O código a seguir especifica *radioGroupToys* como o novo nome de grupo para todos os botões de opção em *radioGroupGames*.

```
radioGroupGames.setGroupName("radioGroupToys");
```

## FRadioButton.setLabel

### Disponibilidade

Flash Player 6.

### Uso

```
myRadioButton.setLabel(label)
```

### Parâmetros

*label* Uma seqüência de caracteres de texto que especifica o rótulo exibido à direita do botão de opção.

### Retorna

Nada.

### Descrição

Método; aplica um rótulo à instância de botão de opção *myRadioButton* durante a execução. Se este método for chamado, o valor do parâmetro *label* especificado na criação será cancelado. Não é possível usar este método para definir rótulos para grupos de botões de opção; a sintaxe *radioButtonGroup.getLabel* não é válida.

### Exemplo

O código a seguir aplica o rótulo *Olhos castanhos* a *radiol*.

```
radiol.setLabel("Olhos castanhos");
```

### Consulte também

*FRadioButton.getLabel*

## FRadioButton.setLabelPlacement

### Disponibilidade

Flash Player 6.

### Uso

```
myRadioButton.setLabelPlacement(labelPosition)
```

```
myRadioButtonsGroup.setLabelPlacement(labelPosition)
```

### Parâmetros

*labelPosition* Uma sequência de caracteres de texto; especifica "left" ou "right".

### Descrição

Método; especifica se o rótulo é exibido à esquerda ou à direita do botão de opção. Se este método for chamado, o valor do parâmetro Label Placement definido durante o processo de criação será cancelado.

Uso 1: especifica o posicionamento do rótulo de um único botão de opção.

Uso 2: especifica o posicionamento dos rótulos de todos os botões de opção em um grupo.

### Exemplo

Uso 1: O código a seguir posiciona o rótulo de `radio1` à esquerda do botão de opção.

```
radio1.setLabelPlacement("left");
```

Uso 2: O código a seguir posiciona os rótulos dos botões de opção no grupo `Colors` à direita dos botões.

```
Colors.setLabelPlacement("right");
```

### Consulte também

`FRadioButton.setLabel`, `FRadioButton.setLabelPlacement`

## FRadioButton.setSize

### Disponibilidade

Flash Player 6.

### Uso

```
myRadioButton.setSize(width)
```

```
myRadioButtonsGroup.setSize(width)
```

### Parâmetros

*width* Um número inteiro que especifica o tamanho do botão de opção, em pixels.

### Retorna

Nada.

### Descrição

Método; especifica a largura do botão de opção, em pixels, e redesenha o botão de opção. (Não é possível definir a altura de componentes do botão de opção.) Se este método for chamado, o dimensionamento de largura aplicado durante o processo de criação será cancelado.

Uso 1: Define o tamanho de um botão de opção.

Uso 2: Define o tamanho de todos os botões de opção em um grupo.

Para obter mais informações, consulte “Dimensionando componentes RadioButton” no capítulo “Usando componentes” de *Usando o Flash*.

#### Exemplo

O código a seguir define a largura de `radiol` como sendo 200 pixels.

```
radiol.setSize(200);
```

## FRadioButton.setState

### Disponibilidade

Flash Player 6.

### Uso

```
myRadioButton.setState("select")
```

### Parâmetros

*select* Um valor Booleano que indica se o botão de opção está selecionado (`true`) ou não (`false`).

### Retorna

Nada.

### Descrição

Método: especifica se *myRadioButton* está selecionado (`true`) ou não (`false`). Somente um botão de opção em um grupo (todos tendo o mesmo parâmetro Nome do grupo) pode ter um estado inicial de `true` (selecionado). Se mais de um botão de opção tiver `true` especificado para este parâmetro, o último botão de opção com um parâmetro de estado inicial `true` será selecionado. O valor padrão para este parâmetro é `false`.

Se este método for chamado, o valor do parâmetro Initial State definido durante o processo de criação será cancelado. Se você chamar este método e também chamar `FRadioButton.setValue` para selecionar um botão de opção durante a execução, e os botões de opção forem botões diferentes no mesmo grupo, o botão de opção especificado no último método chamado será selecionado.

### Exemplo

O código a seguir seleciona o botão de opção `radiol` durante a execução.

```
radiol.setState(true));
```

### Consulte também

`FRadioButton.getState`, `FRadioButton.getValue`, `FRadioButton.setValue`

## FRadioButton.setStyleProperty

### Disponibilidade

Flash Player 6.

### Uso

```
myRadioButton.setStyleProperty(styleProperty, value)
```

```
myRadioGroup.setStyleProperty(styleProperty, value)
```

### Parâmetros

*styleProperty* Uma sequência de caracteres que especifica uma propriedade do objeto `FStyleFormat`.

*value* O valor definido para a propriedade.

### Retorna

Nada.

### Descrição

Método; define uma propriedade `FStyleFormat` para um determinado botão de opção. Chamar este método para especificar uma propriedade cancela as configurações dessa propriedade no formato de estilo atribuído ao componente. Se o valor `undefined` for atribuído a uma propriedade, todos os estilos dessa propriedade serão removidos.

Para definir as propriedades `FStyleFormat` para vários componentes, crie um formato de estilo personalizado. Para obter mais informações sobre a criação de formatos de estilo, consulte “Personalizando cores e texto de componentes” no capítulo “Usando componentes” de *Usando o Flash*.

### Exemplo

O código a seguir define a propriedade `radioDot` para `radioButton1` como `0xFF12AC` (rosa).

```
radioButton1.setStyleProperty("radioDot", 0xFF12AC);
```

O código a seguir define a propriedade `radioDot` para todos os botões em `radioGroup1` como `0xFF12AC` (rosa).

```
radioGroup1.setStyleProperty("radioDot", 0xFF12AC);
```

### Consulte também

`FStyleFormat` (object)

## FRadioButton.setValue

### Disponibilidade

Flash Player 6.

### Uso

```
myRadioGroup.setValue("data")
```

### Parâmetros

*data* Os dados associados ao botão de opção a ser selecionado.

### Retorna

Nada.

### Descrição

Método; marca o botão de opção associado aos dados especificados e desmarca os botões eventualmente selecionados no mesmo grupo.

Se este método for chamado, o valor do parâmetro Initial Value definido durante o processo de criação será cancelado. Se você chamar este método e também chamar `FRadioButton.setState` para selecionar um botão de opção durante a execução, e os botões de opção forem botões diferentes no mesmo grupo, o botão de opção especificado no último método chamado será selecionado.

### Exemplo

O código a seguir seleciona o botão de opção com o dado associado `red` no grupo de botões de opção denominado `Colors`.

```
Colors.setValue("red");
```

### Consulte também

`FRadioButton.getData`, `FRadioButton.getValue`, `FRadioButton.setState`

## FScrollBar (component)

O componente `ScrollBar` no ambiente de criação Flash oferece recurso de arrastar e soltar para adicionar barras de rolagem a campos de texto dinâmicos e de entrada em documentos Flash; ele também oferece uma interface de usuário para definição de parâmetros básicos. Os métodos do componente `FScrollBar` permitem controlar barras de rolagem durante a execução: você pode criar barras de rolagem, controlar barras de rolagem criadas no ambiente de criação, definir ou cancelar parâmetros básicos e definir opções adicionais de tempo de execução. Não é preciso usar um construtor para acessar os métodos de componentes. Alguns dos métodos do componente `FScrollBar` não são recomendados para uso com barras de rolagem anexadas a campos de texto. Consulte as entradas individuais dos métodos para obter detalhes.

Usuários e programadores avançados podem usar o componente `ScrollBar` com outros elementos do Flash para criar interfaces de usuário personalizadas.

Os métodos do componente não realizam verificação de erros de tipo, como outros objetos e ações nativos do `ActionScript`; portanto, recomenda-se a validação dos parâmetros antes de passá-los para métodos.

O componente `ScrollBar` tem suporte do Flash Player 6 e de suas versões posteriores.

Para obter informações sobre o uso do componente `ScrollBar`, como definir parâmetros durante o processo de criação e como alterar as cores e a aparência de componentes, consulte “Personalizando cores e texto do componente” e “Personalizando aparências de componentes” no capítulo “Usando componentes” de *Usando o Flash*.

## Resumo dos métodos do componente FScrollBar.

Método	Descrição
<code>FScrollBar.setEnabled</code>	Retorna <code>true</code> se a barra de rolagem estiver ativada, <code>false</code> se estiver desativada.
<code>FScrollBar.getScrollPosition</code>	Retorna um número inteiro que representa a posição atual da caixa de rolagem (direcionador).
<code>FScrollBar.registerSkinElement</code>	Registra um elemento de aparência em uma propriedade definida para uma aparência no <code>ReadMe</code> localizado no Quadro 1 de um clipe de filme de aparência na biblioteca.

Método	Descrição
<code>FScrollBar.setChangeHandler</code>	Especifica uma função a ser chamada todas as vezes que a posição de rolagem for alterada. (Este método não pode ser usado com campos de texto.)
<code>FScrollBar.setEnabled</code>	Especifica se a barra de rolagem está ativada ( <code>true</code> ) ou desativada ( <code>false</code> ).
<code>FScrollBar.setHorizontal</code>	Especifica se a barra de rolagem é horizontal ( <code>true</code> ) ou vertical ( <code>false</code> ).
<code>FScrollBar.setLargeScroll</code>	Especifica o número de posições roladas quando o usuário clica na trilha.
<code>FScrollBar.setScrollContent</code>	Especifica a instância de campo de texto à qual a barra de rolagem se aplica.
<code>FScrollBar.setScrollPosition</code>	Define a posição da caixa de rolagem como um número inteiro entre <code>minPos</code> e <code>maxPos</code> .
<code>FScrollBar.setScrollProperties</code>	Define as propriedades <code>pageSize</code> , <code>minPos</code> e <code>maxPos</code> da barra de rolagem. (Este método não pode ser usado com campos de texto.)
<code>FScrollBar.setScrollTarget</code>	Especifica um campo de texto como o destino da barra de rolagem.
<code>FScrollBar.setSize</code>	Define o comprimento da barra de rolagem, em pixels.
<code>FScrollBar.setSmallScroll</code>	Especifica o número de posições roladas quando o usuário clica na seta de rolagem.
<code>FScrollBar.setStyleProperty</code>	Define uma única propriedade de estilo para um componente.

## FScrollBar.setEnabled

### Disponibilidade

Flash Player 6.

### Uso

```
myScrollBar.setEnabled()
```

### Parâmetros

Nenhum.

### Retorna

Um valor booleano.

### Descrição

Método; indica se a barra de rolagem está ativada (`true`) ou desativada (`false`).

### Exemplo

O código a seguir retorna um valor na janela Saída que indica se `scroll1` está ativada (`true`) ou desativada (`false`).

```
trace(scroll1.setEnabled());
```

### Consulte também

`FScrollBar.setEnabled`



## FScrollBar.getScrollPosition

### Disponibilidade

Flash Player 6.

### Uso

```
myScrollBar.getScrollPosition()
```

### Parâmetros

Nenhum.

### Retorna

Um inteiro.

### Descrição

Método; retorna um número inteiro que especifica a posição da caixa de rolagem (direcionador). O valor apresentado na faixa definida pelas propriedades `minPos` e `maxPos` que determina os limites de rolagem da barra de rolagem. Para determinar os parâmetros `minPos` e `maxPos`, use `FScrollBar.setScrollProperties`.

### Exemplo

O código a seguir retorna a posição atual da caixa de rolagem da barra de rolagem `scroll2` na janela Saída. Se a configuração de `scroll2` em `minPos` for 2 e em `maxPos` for 25, um valor de retorno equivalente a 12 indicará que a caixa de rolagem está no meio da barra de rolagem.

```
trace(scroll2.getPosition());
```

Consulte `FScrollBar.setChangeHandler` para obter outro exemplo que utilize este método.

### Consulte também

`FScrollBar.setChangeHandler`, `FScrollBar.setScrollPosition`

## FScrollBar.registerSkinElement

### Disponibilidade

Flash Player 6.

### Uso

```
myScrollBar.registerSkinElement(element, styleProperty)
```

### Parâmetros

*element* Uma instância de clipe de filme.

*styleProperty* Uma seqüência de caracteres que especifica uma propriedade `FStyleFormat`.

### Retorna

Nada.

### Descrição

Método; registra um elemento de aparência em uma propriedade de estilo. Elementos de aparência são registrados em propriedades no primeiro quadro da camada ReadMe de cada aparência na biblioteca.

Os componentes são compostos de aparências e cada aparência é composta de vários elementos de aparência, cada um dos quais pode ser registrado em uma propriedade do objeto `FStyleFormat`. Essas propriedades são valores atribuídos pelo formato de estilo atribuído a um componente. Como padrão, o objeto `globalStyleFormat` é atribuído a todos os componentes de interface do Flash. Esse objeto é uma instância do objeto `FStyleFormat`.

Use este método para registrar propriedades e elementos de aparência personalizados na interface do Flash ou aparências personalizadas de componentes editando o código no primeiro quadro da camada `ReadMe` de uma aparência na biblioteca.

O componente `FScrollBar` usa as aparências na pasta `FRadioButton Skins` e a aparência `FLabel` na pasta `Global Skins` depois que você adiciona o componente ao documento Flash. A edição de qualquer aparência na pasta `FScrollBar Skins` afeta todos os componentes que usam barras de rolagem (`ComboBox`, `Listbox`, `ScrollBar` e `ScrollPane`).

Para obter mais informações, consulte “Personalizando aparências de componentes” no capítulo “Usando componentes” de *Usando o Flash*.

### Exemplo

O código a seguir registra o elemento de aparência personalizado `NewArrow_mc` na propriedade `arrow` no primeiro quadro da camada `ReadMe` da aparência `fsb_downArrow` na pasta `FScrollBar Skins` na biblioteca.

```
Scroll11.registerSkinElement(NewArrow_mc, "arrow");
```

### Consulte também

`FStyleFormat` (object)

## FScrollBar.setChangeHandler

### Disponibilidade

Flash Player 6.

### Uso

```
myScrollBar.setChangeHandler(functionName, [location])
```

### Parâmetros

*functionName* Uma sequência de caracteres que especifica o nome da função do identificador a ser executada quando o usuário move a caixa de rolagem. Se o parâmetro *location* não for especificado, esta função deverá estar na mesma Linha de tempo da instância do componente.

*location* Uma referência de caminho até um objeto de dados, clipe de filme ou Linha de tempo que contém a função especificada. Este parâmetro é opcional e tem como padrão a Linha de tempo pai do componente.

### Retorna

Nada.

### Descrição

Método; especifica um identificador de alteração a ser chamado quando o usuário move a caixa de rolagem (direcionador) da barra de rolagem. Você pode especificar a mesma função de identificador de alteração para mais de um componente; a função sempre aceita a instância do componente que foi alterada como um parâmetro. Se este método for chamado, valor do parâmetro Identificador de alteração especificado na criação será cancelado.

Este método é para usuários e programadores avançados que criam aplicativos e componentes personalizados usando o componente Flash UI ScrollBar; o método não pode ser usado com barras de rolagem anexadas a campos de texto.

### Exemplo

O código a seguir cria uma caixa preenchida no Palco, aplica uma barra de rolagem horizontal, define as propriedades de rolagem e especifica a função `mover` como o identificador de alteração. O identificador de alteração `mover` usa a posição de rolagem da barra de rolagem para alterar a posição `_x` do clipe de filme entre 50 e 250.

```
root.createEmptyMovieClip("quadrado", 1);
_root.square._x = 50;
_root.square._y = 50;

with (_root.square) {
    moveTo(0, 0);
    beginFill(0x0066CC);
    lineTo(20, 0);
    lineTo(20, 20);
    lineTo(0, 20);
    lineTo(0, 0);
    endFill();
}

scrollBar._x = 50;
scrollBar.setHorizontal (true);
scrollBar.setScrollProperties (1, 50, 250);
scrollBar.setChangeHandler ("mover");

function mover () {
    _root.square._x = scrollBar.getScrollPosition();
}
```

O código a seguir especifica uma função de identificador de alteração para uma instância do componente de barra de rolagem anexado a um componente de caixa de listagem personalizada. O identificador de alteração define `scroll1` para obter a posição de rolagem atual usando `FScrollBar.getScrollPosition` e, a seguir, `customListBox` usa `FScrollBar.setScrollPosition` para redefinir a posição de rolagem de forma que o item na posição de rolagem atual seja exibido no topo da exibição da caixa de listagem personalizada. O parâmetro `component` é automaticamente preenchido com a instância de um componente (o componente que foi alterado como resultado de uma entrada do usuário e que especifica `myHandler` como seu identificador de alteração).

```
rolagem1.setChangeHandler("myHandler");

function myHandler(component)
{
    customListBox.setScrollPosition(component.getScrollPosition());
}
```

Se, no exemplo acima, `myHandler` fosse uma função localizada na Linha de tempo bisavô da Linha de tempo do componente, a primeira linha de código seria da seguinte forma:

```
scroll1.setChangeHandler("myHandler", _parent._parent._parent);
```

O código a seguir cria a função `myHandler` em uma instância de `myObject` (que é da classe `Object`) e, a seguir, especifica `myHandler` como a função de `scroll1`.

```
myObject = new Object();
myObject.myHandler = function(component){
    customListBox.setScrollPosition(component.getScrollPosition());
}

scroll1.setChangeHandler("myHandler", myObject);
```

#### Consulte também

`FScrollBar.getScrollPosition`, `FScrollBar.setScrollPosition`,  
`FScrollBar.setScrollProperties`

## FScrollBar.setEnabled

#### Disponibilidade

Flash Player 6.

#### Uso

```
myScrollBar.setEnabled(enable)
```

#### Parâmetros

*enable* Um valor Booleano que especifica se a barra de rolagem está ativada (`true`) ou desativada (`false`).

#### Retorna

Nada.

#### Descrição

Método; determina se a barra de rolagem está ativada (`true`) ou desativada (`false`). Se uma barra de rolagem estiver desativada, ela não aceitará interação de mouse nem de teclado do usuário, e ficará esmaecida (indisponível). Chamar este método sem passar um parâmetro é o mesmo que passar o parâmetro `true`.

#### Exemplo

O código a seguir desativa a barra de rolagem `scroll2`.

```
scroll2.setEnabled(false);
```

#### Consulte também

`FScrollBar.setEnabled`

## FScrollBar.setHorizontal

#### Disponibilidade

Flash Player 6.

#### Uso

```
myScrollBar.setHorizontal(horizontalScroll)
```

#### Parâmetros

*horizontalScroll* Um valor Booleano que especifica se a barra de rolagem será horizontal (`true`) ou vertical (`false`).

**Retorna**

Nada.

**Descrição**

Método; especifica se a barra de rolagem é aplicada ao destino horizontalmente (`true`) ou verticalmente (`false`). O padrão deste método é `false`.

**Exemplo**

O código a seguir especifica que a barra de rolagem `scrollText` seja aplicada horizontalmente no destino.

```
scrollText.setHorizontal(true);
```

**Consulte também**

`FScrollbar.setSize`

## FScrollbar.setLargeScroll

**Disponibilidade**

Flash Player 6.

**Uso**

```
myScrollbar.setLargeScroll(largeScroll)
```

**Parâmetros**

*largeScroll* Um número inteiro que especifica o número de posições a serem roladas quando o usuário clica uma vez na trilha. O valor padrão é o valor definido para `pageSize` com `FScrollbar.setScrollProperties`.

**Retorna**

Nada.

**Descrição**

Método; define a propriedade `largeScroll` da instância da barra de rolagem durante a execução. Quando o usuário clica uma vez na trilha de rolagem, a caixa de rolagem (direcionador) é movida na distância especificada em uma posição de `largeScroll`.

**Exemplo**

O código a seguir define que quando o usuário clica na trilha, `scrollText1` é rolado 20 posições.

```
scrollText1.setLargeScroll(20);
```

**Consulte também**

`FScrollbar.setSmallScroll`

## FScrollbar.setScrollContent

**Disponibilidade**

Flash Player 6.

**Uso**

```
myScrollbar.setScrollContent(target)
```

**Parâmetros**

*target* Uma referência ao campo de texto da barra de rolagem.

**Retorna**

Nada.

**Descrição**

Método; especifica a instância de campo de texto à qual a barra de rolagem se aplica. A instância deve ser definida na mesma Linha de tempo e no mesmo nível da barra de rolagem. Se este método for chamado, o valor do parâmetro Target Text Field definido durante o processo de criação será cancelado. Se o valor `undefined` for passado para o parâmetro *target*, a barra de rolagem será dissociada do campo de texto.

**Exemplo**

O código a seguir anexa `scrollText1` ao campo de texto com o nome de instância `textField1`.  
`scrollText1.setScrollContent("textField1");`

## FScrollBar.setScrollPosition

**Disponibilidade**

Flash Player 6.

**Uso**

`myScrollBar.setScrollPosition(position)`

**Parâmetros**

*position* Um número inteiro entre as configurações de `minPos` e `maxPos` da barra de rolagem. Consulte `FScrollBar.setScrollProperties` para obter mais informações sobre como configurar as propriedades `minPos` e `maxPos`.

**Retorna**

Nada.

**Descrição**

Método; especifica a posição da caixa de rolagem (direcionador) na barra de rolagem e executa a função especificada com `FScrollBar.setChangeHandler`.

**Exemplo**

O código a seguir define a posição da barra de rolagem para `scrollText1` como 5.

`scrollText1.setScrollPosition(5);`

Consulte `FScrollBar.setChangeHandler` para obter outro exemplo que utilize este método.

**Consulte também**

`FScrollBar.setChangeHandler`, `FScrollBar.setScrollProperties`

## FScrollBar.setScrollProperties

### Disponibilidade

Flash Player 6.

### Uso

```
myScrollBar.setScrollProperties(pageSize, minPos, maxPos)
```

### Parâmetros

*pageSize* Um número inteiro que representa o número de posições exibidas na página.

*minPos* Um número inteiro que representa a posição de rolagem mínima.

*maxPos* Um número inteiro que representa a posição de rolagem máxima.

### Retorna

Nada.

### Descrição

Método; especifica as propriedades *pageSize*, *minPos* e *maxPos* da barra de rolagem e define a caixa de rolagem (direcionador) da barra de rolagem no tamanho adequado.

Este método é para usuários e programadores avançados que criam componentes personalizados; o método não pode ser usado com barras de rolagem anexadas a campos de texto. Quando uma barra de rolagem é anexada a um campo de texto, as propriedades de rolagem são automaticamente definidas de acordo com as propriedades do campo de texto e a execução desse método quebra a barra de rolagem do campo de texto.

### Exemplo

O código a seguir define as propriedades *pageSize*, *minPos* e *maxPos* de uma barra de rolagem anexada a um componente de caixa de listagem personalizada. A caixa de listagem tem 5 linhas visíveis e um total de 20 itens na lista. Como a caixa é indexada de 0 a 19, a propriedade *maxPos* será igual ao número total de itens na caixa menos o número de itens visíveis.

```
scrollBar.setScrollProperties(5, 0, 15);
```

### Consulte também

FScrollBar.setScrollPosition

## FScrollBar.setScrollTarget

### Disponibilidade

Flash Player 6.

### Uso

```
myScrollBar.setScrollTarget(target)
```

### Parâmetros

*target* Uma referência ao campo de texto da barra de rolagem.

### Retorna

Nada.

**Descrição**

Método; especifica a instância de campo de texto à qual a barra de rolagem se aplica. A instância deve ser definida na mesma Linha de tempo e no mesmo nível da barra de rolagem. Se este método for chamado, o valor do parâmetro Target Text Field definido durante o processo de criação será cancelado. Se o valor `undefined` for passado para o parâmetro *target*, a barra de rolagem será dissociada do campo de texto.

**Exemplo**

O código a seguir anexa `scrollText1` ao campo de texto com o nome de instância `textField1`.

```
scrollText1.setScrollTarget("textField1");
```

## FScrollBar.setSize

**Disponibilidade**

Flash Player 6.

**Uso**

```
myScrollBar.setSize(length)
```

**Parâmetros**

*length* Um número inteiro que especifica o comprimento da barra de rolagem, em pixels.

**Retorna**

Nada.

**Descrição**

Método; define o comprimento, em pixels, da barra de rolagem durante a execução. (Não é possível definir a largura dos componentes da barra de rolagem.) Se este método for chamado, qualquer dimensionamento aplicado durante o processo de criação será cancelado.

Este método não deve ser usado com barras de rolagem anexadas a campos de texto; a barra de rolagem automaticamente se ajusta ao tamanho do campo de texto durante o processo de criação.

Para obter mais informações, consulte “Dimensionando componentes do ScrollBar” no capítulo “Usando componentes” de *Usando o Flash*.

**Exemplo**

O código a seguir define o comprimento de `scrollText1` como sendo 200 pixels.

```
scrollText1.setSize(200);
```

## FScrollBar.setSmallScroll

**Disponibilidade**

Flash Player 6.

**Uso**

```
myScrollBar.setSmallScroll(smallScroll)
```

**Parâmetros**

*smallScroll* Um número inteiro que especifica o número de posições a serem roladas quando o usuário clica em uma seta de rolagem. O valor padrão é 1.



**Retorna**

Nada.

**Descrição**

Método; define a propriedade `smallScroll` da instância da barra de rolagem durante a execução, se o campo de texto tiver o foco. Quando o usuário clica nas setas da barra de rolagem ou em uma tecla de seta do teclado, a caixa de rolagem (direcionador) se move na distância especificada por uma posição de `smallScroll`.

**Exemplo**

O código a seguir especifica que quando o usuário clica em uma seta de rolagem, `scrollText1` é rolado 5 posições.

```
scrollText1.setSmallScroll(5);
```

**Consulte também**

`FScrollBar.setLargeScroll`

## **FScrollBar.setStyleProperty**

**Disponibilidade**

Flash Player 6.

**Uso**

```
myScrollBar.setStyleProperty(styleProperty, value)
```

**Parâmetros**

*styleProperty* Uma sequência de caracteres que especifica uma propriedade do objeto `FStyleFormat`.

*value* O valor definido para a propriedade.

**Retorna**

Nada.

**Descrição**

Método; define uma propriedade `FStyleFormat` para uma determinada barra de rolagem. Chamar este método para especificar uma propriedade cancela as configurações dessa propriedade no formato de estilo atribuído ao componente. Se o valor `undefined` for atribuído a uma propriedade, todos os estilos dessa propriedade serão removidos.

Para definir as propriedades `FStyleFormat` para vários componentes, crie um formato de estilo personalizado. Para obter mais informações sobre a criação de formatos de estilo personalizados, consulte “Personalizando cores e texto do componente” no capítulo “Usando componentes” de *Usando o Flash*.

**Exemplo**

O código a seguir define a propriedade `arrow` da `scrollBar1` como `0x000000` (preto).

```
scrollBar1.setStyleProperty("arrow", 0x000000);
```

**Consulte também**

`FStyleFormat` (object)

## FScrollPane (component)

O componente ScrollPane no ambiente de criação Flash oferece recurso de arrastar e soltar para adicionar painéis de rolagem para exibição de clipes de filmes em documentos Flash; ele também oferece uma interface de usuário para definição de parâmetros básicos. Os métodos do componente FScrollPane permitem controlar painéis de rolagem durante a execução: você pode criar painéis de rolagem, controlar painéis de rolagem criados no ambiente de criação, definir ou cancelar parâmetros básicos e definir opções adicionais de tempo de execução. Não é preciso usar um construtor para acessar os métodos de componentes.

O componente ScrollPane oferece barras de rolagem verticais e horizontais que permitem exibir clipes de filmes grandes sem ocupar muito espaço do Palco. Controles padrão de mouse e teclado são incorporados.

**Observação:** O componente ScrollPane só exibe clipes de filmes; para adicionar barras de rolagem a campos de texto dinâmicos e de entrada, use o componente ScrollBar. O componente ScrollPane não pode exibir nenhum conteúdo que utilize fontes de dispositivo.

Os métodos do componente não realizam verificação de erros de tipo, como outros objetos e ações nativos do ActionScript; portanto, recomenda-se a validação dos parâmetros antes de passá-los para métodos.

O componente ScrollPane tem suporte do Flash Player 6 e de suas versões posteriores.

Para obter informações sobre o uso do componente ScrollPane, como definir parâmetros durante o processo de criação e como alterar as cores e a aparência de componentes, consulte “Personalizando cores e texto do componente” e “Personalizando aparências de componentes” no capítulo “Usando componentes” de *Usando o Flash*.

### Resumo dos métodos do componente FScrollPane.

Método	Descrição
FScrollPane.getPaneHeight	Retorna a altura do painel de rolagem.
FScrollPane.getPaneWidth	Retorna a largura do painel de rolagem.
FScrollPane.getScrollContent	Retorna uma instância do conteúdo exibido no painel de rolagem.
FScrollPane.getScrollPosition	Retorna as coordenadas x e y da posição de rolagem atual.
FScrollPane.loadScrollContent	Carrega um SWF ou JPEG no painel de rolagem.
FScrollPane.refreshPane	Redimensiona as barras de rolagem no painel de rolagem quando o conteúdo muda de tamanho.
FScrollPane.registerSkinElement	Registra um elemento de aparência em uma propriedade definida para uma aparência na camada ReadMe localizada no Quadro 1 de um clipe de filme de aparência na biblioteca.
FScrollPane.setDragContent	Define o conteúdo do painel de rolagem como arrastável.
FScrollPane.setHScroll	Define o estilo de rolagem horizontal do painel de rolagem.
FScrollPane.setScrollContent	Define um clipe de filme como destino do painel de rolagem.
FScrollPane.setScrollPosition	Faz o painel rolar até as coordenadas x, y especificadas.
FScrollPane.setSize	Define a largura e a altura do painel de rolagem, em pixels.
FScrollPane.setStyleProperty	Define uma única propriedade de estilo para um componente.
FScrollPane.setVScroll	Define o estilo de rolagem vertical do painel de rolagem.

## FScrollPane.getPaneHeight

### Disponibilidade

Flash Player 6.

### Uso

```
myScrollPane.getPaneHeight()
```

### Parâmetros

Nenhum.

### Retorna

Um número inteiro que especifica a altura da exibição do painel de rolagem.

### Descrição

Método; retorna a altura da exibição do painel de rolagem. Você só pode usar este método para obter a altura de um painel de rolagem que foi dimensionado com `FScrollPane.setSize`. Este método só funciona se o painel de rolagem tiver sido dimensionado com `FScrollPane.setSize`. Ele não funciona se você tiver definido o tamanho através das propriedades `_width` e `_height`.

### Exemplo

O código a seguir obtém a altura e a largura do painel de rolagem `display1` e usa os valores apresentados para redimensionar o painel de rolagem com `FScrollPane.setSize`.

```
var h = display1.getPaneHeight();  
var w = display1.getPaneWidth();  
display1.setSize(w+10, h+10);
```

### Consulte também

`FScrollPane.getPaneWidth`, `FScrollPane.setSize`

## FScrollPane.getPaneWidth

### Disponibilidade

Flash Player 6.

### Uso

```
myScrollPane.getPaneWidth()
```

### Parâmetros

Nenhum.

### Retorna

Um número inteiro que especifica a largura da exibição do painel de rolagem.

### Descrição

Método; retorna a largura da exibição do painel de rolagem. Você só pode usar este método para obter a largura de um painel de rolagem que foi dimensionado com `FScrollPane.setSize`. Este método só funciona se o painel de rolagem tiver sido dimensionado com `FScrollPane.setSize`. Ele não funciona se você tiver definido o tamanho através das propriedades `_width` e `_height`.

### Exemplo

O código a seguir obtém a altura e a largura do painel de rolagem `display1` e usa os valores apresentados para redimensionar o painel de rolagem com `FScrollPane.setSize`.

```
var h = display1.getPaneHeight();
var w = display1.getPaneWidth();
display1.setSize(w+10, h+10);
```

### Consulte também

`FScrollPane.getPaneHeight`, `FScrollPane.setSize`

## FScrollPane.getScrollContent

### Disponibilidade

Flash Player 6.

### Uso

```
myScrollPane.getScrollContent()
```

### Parâmetros

Nenhum.

### Retorna

Uma referência ao clipe de filme no painel de rolagem.

### Descrição

Método; retorna uma instância do conteúdo exibido no painel de rolagem.

### Exemplo

O código a seguir recupera uma referência ao clipe de filme dentro de `display1`, armazena-a em uma variável `e`, a seguir, faz o clipe de filme ir para o quadro 4.

```
var content = display1.getScrollContent();
content.gotoAndStop(4);
```

### Consulte também

`FScrollPane.setScrollContent`

## FScrollPane.getScrollPosition

### Disponibilidade

Flash Player 6.

### Uso

```
myScrollPane.getScrollPosition()
```

### Parâmetros

Nenhum.

### Retorna

Um objeto.

### Descrição

Método; retorna um objeto com os campos `.x` ou `.y` especificando a posição de rolagem vertical ou horizontal atual da exibição do painel de rolagem.

### Exemplo

O código a seguir retorna a posição de rolagem atual do painel de rolagem `scroll2` na janela Saída.

```
trace(scroll2.getScrollPosition());
```

### Consulte também

`FScrollPane.setScrollPosition`

## FScrollPane.loadScrollContent

### Disponibilidade

Flash Player 6.

### Uso

```
myScrollPane.loadScrollContent(URL [, functionName, location])
```

### Parâmetros

*URL* Uma sequência de caracteres que especifica o URL de um arquivo SWF ou JPEG a ser carregado no painel de rolagem.

*functionName* Uma sequência de caracteres que especifica o nome da função do identificador a ser executada quando o conteúdo do painel de rolagem é carregado. Se o parâmetro *location* não for especificado, esta função deverá estar na mesma Linha de tempo da instância do componente.

*location* Uma referência de caminho até um objeto de dados, clipe de filme ou Linha de tempo que contém a função especificada. Este parâmetro é opcional e tem como padrão a Linha de tempo pai do componente.

### Retorna

Nada.

### Descrição

Método; especifica o URL de um arquivo SWF ou JPEG a ser exibido no painel de rolagem. Os parâmetros opcionais *funcName* e *location* permitem especificar uma função de identificador de alteração a ser chamada quando o conteúdo é carregado.

O URL deve estar no mesmo subdomínio que o URL onde o filme Flash reside no momento. Para usar arquivos SWF ou JPEG no Flash Player ou testar o filme no ambiente de criação Flash, você deve armazenar todos os arquivos SWF ou JPEG na mesma pasta, e os nomes dos arquivos não podem conter especificações de pasta nem de unidade de disco.

Se este método for chamado, o valor do parâmetro Scroll Content definido durante o processo de criação será cancelado.

Consulte `FScrollBar.setChangeHandler` para obter mais informações e exemplos de como usar funções do identificador de alteração.

### Exemplo

O código a seguir carrega em `display1` um JPEG localizado em um servidor.

```
display1.loadScrollContent("http://www.YourWebServer.com/Nice.jpg");
```

O código a seguir carrega um JPEG localizado em um servidor e especifica a função do identificador de alteração `load` localizada na Linha de tempo avô do componente `display1`.

```
display1.loadScrollContent("http://www.YourWebServer.com/Nice.jpg" , "load" ,
    _parent._parent);

function load(component){
    //conteúdo está carregado
    component.setScrollPostion(10,10);
}
```

**Consulte também**

`FScrollPane.getPaneHeight`, `FScrollPane.setScrollContent`

## FScrollPane.refreshPane

**Disponibilidade**

Flash Player 6.

**Uso**

```
myScrollPane.refreshPane()
```

**Parâmetros**

Nenhum.

**Retorna**

Nada.

**Descrição**

Método; redimensiona as barras de rolagem do painel de rolagem quando o conteúdo dentro do painel de rolagem é alterado. Chame este método se você redimensionar o conteúdo na janela do painel de rolagem usando `_width` ou `_height`.

**Exemplo**

O código a seguir atualiza as barras de rolagem de `moviePane` depois de aumentar o tamanho do clipe de filme `myContent` no painel de rolagem.

```
var myContent = moviePane.getScrollContent();
myContent._width = 400;
moviePane.refreshPane();
```

**Consulte também**

`FScrollPane.getScrollContent`

## FScrollPane.registerSkinElement

**Disponibilidade**

Flash Player 6.

**Uso**

```
myScrollPane.registerSkinElement(element, styleProperty)
```

**Parâmetros**

*element* Uma instância de clipe de filme.

*styleProperty* O nome de uma propriedade de `FStyleFormat`.

**Retorna**

Nada.

**Descrição**

Método; registra um elemento de aparência em uma propriedade de estilo. Elementos de aparência são registrados em propriedades no primeiro quadro da camada ReadMe de cada aparência na biblioteca.

Os componentes são compostos de aparências e cada aparência é composta de vários elementos de aparência, cada um dos quais pode ser registrado em uma propriedade do objeto `FStyleFormat`. Essas propriedades são valores atribuídos pelo formato de estilo atribuído a um componente. Como padrão, o objeto `globalStyleFormat` é atribuído a todos os componentes de interface do Flash. Esse objeto é uma instância do objeto `FStyleFormat`.

Use este método para registrar propriedades e elementos de aparência personalizados na interface do Flash ou aparências personalizadas de componentes editando o código no primeiro quadro da camada ReadMe de uma aparência na biblioteca.

O componente `FScrollPane` usa as aparências na pasta `FScrollBar Skins` e a aparência `FLabel` na pasta `Global Skins` depois que você adiciona o componente ao documento Flash. A edição de qualquer aparência na pasta `FScrollBar Skins` afeta todos os componentes que usam barras de rolagem (`ComboBox`, `ListBox`, `ScrollBar` e `ScrollPane`).

Para obter mais informações, consulte “Personalizando aparências de componentes” no capítulo “Usando componentes” de *Usando o Flash*.

**Exemplo**

O código a seguir registra o elemento de aparência personalizado `NewArrow_mc` na propriedade `arrow` no primeiro quadro da camada ReadMe da aparência `fsb_downArrow` na pasta `FScrollBar Skins` na biblioteca.

```
Panel.registerSkinElement(NewArrow_mc, "arrow");
```

**Consulte também**

`FStyleFormat` (object)

## **FScrollPane.setDragContent**

**Disponibilidade**

Flash Player 6.

**Uso**

```
myScrollPane.setDragContent(drag)
```

**Parâmetros**

*drag* Um valor Booleano; `true` que o usuário pode alterar a exibição arrastando o conteúdo no painel de rolagem; `false` especifica que o usuário pode alterar a exibição somente com o uso das barras de rolagem.

**Retorna**

Nada.

**Descrição**

Método; especifica se o usuário pode alterar a exibição do painel de rolagem arrastando seu conteúdo, além de usar as barras de rolagem. Se este método for chamado, o valor do parâmetro Drag Content definido durante o processo de criação será cancelado.

**Exemplo**

O exemplo a seguir especifica que o conteúdo no painel de rolagem `display1` pode ser arrastado.

```
display1.setDragContent(true);
```

## FScrollPane.setHScroll

**Disponibilidade**

Flash Player 6.

**Uso**

```
myScrollPane.setHScroll(display)
```

**Parâmetros**

*display* Um valor Booleano que especifica se a barra de rolagem é exibida sempre (`true`) ou nunca é exibida (`false`), ou uma sequência de caracteres que determina que a barra de rolagem seja exibida somente quando necessário ("`auto`").

Se este método for chamado, o valor do parâmetro Horizontal Scroll definido durante o processo de criação será cancelado.

**Retorna**

Nada.

**Descrição**

Método; determina se a barra de rolagem horizontal é exibida sempre (`true`), nunca é exibida (`false`) ou somente quando necessário ("`auto`"). O valor padrão é `auto`.

**Exemplo**

O código a seguir oculta a barra de rolagem horizontal em `display1`.

```
display1.setHScroll(false);
```

**Consulte também**

```
FScrollPane.setVScroll
```

## FScrollPane.setScrollContent

**Disponibilidade**

Flash Player 6.

**Uso**

```
myScrollPane.setScrollContent(target)
```

**Parâmetros**

*target* Uma sequência de caracteres de texto que especifica a ID de vinculação de símbolo de um clipe de filme na biblioteca ou uma instância de um clipe de filme.

**Retorna**

Nada.



**Descrição**

Método; especifica um clipe de filme a ser exibido no painel de rolagem. Se este método for chamado, o valor do parâmetro Scroll Content definido durante o processo de criação será cancelado.

**Exemplo**

O exemplo a seguir especifica a instância de clipe de filme BetsyTacy como o destino de `display1`.

```
display1.setScrollContent("BetsyTacy");
```

**Consulte também**

`FScrollPane.getPaneHeight`, `FScrollPane.loadScrollContent`

## FScrollPane.setScrollPosition

**Disponibilidade**

Flash Player 6.

**Uso**

```
myScrollPane.setScrollPosition(x, y)
```

**Parâmetros**

*x* Um número inteiro que especifica o número de pixels (a partir de 0) para a rolagem para a direita.

*y* Um número inteiro que especifica o número de pixels (a partir de 0) para a rolagem para baixo.

**Retorna**

Nada.

**Descrição**

Método; define a posição de rolagem segundo as posições das coordenadas *x*, *y* especificadas.

**Exemplo**

O exemplo a seguir faz o conteúdo de `display1` rolar 14 pixels para baixo e 40 pixels para a direita.

```
display1.setScrollPosition(14,40);
```

**Consulte também**

`FScrollPane.getScrollPosition`

## FScrollPane.setSize

**Disponibilidade**

Flash Player 6.

**Uso**

```
myScrollPane.setSize(width, height)
```

**Parâmetros**

*width* Um número inteiro que especifica a largura do painel de rolagem, em pixels.

*height* Um número inteiro que especifica a altura do painel de rolagem, em pixels.

**Retorna**

Nada.

### Descrição

Método; define a largura e a altura, em pixels, da exibição do painel de rolagem durante a execução. Se este método for chamado, o dimensionamento aplicado durante o processo de criação será cancelado.

Para obter mais informações, consulte “Personalizando aparências de componentes” no capítulo “Usando componentes” de *Usando o Flash*.

### Exemplo

O código a seguir define a largura e a altura de `display1` como sendo 500 x 300 pixels.

```
display1.setSize(500, 300);
```

### Consulte também

`FScrollPane.getPaneHeight`, `FScrollPane.getPaneWidth`

## FScrollPane.setStyleProperty

### Disponibilidade

Flash Player 6.

### Uso

```
myScrollPane.setStyleProperty(styleProperty, value)
```

### Parâmetros

*styleProperty* Uma sequência de caracteres que especifica uma propriedade do objeto `FStyleFormat`.

*value* O valor definido para a propriedade.

### Retorna

Nada.

### Descrição

Método; define uma propriedade `FStyleFormat` para um determinado painel de rolagem. Chamar este método para especificar uma propriedade cancela as configurações dessa propriedade no formato de estilo atribuído ao componente. Se o valor `undefined` for atribuído a uma propriedade, todos os estilos dessa propriedade serão removidos.

Para definir as propriedades `FStyleFormat` para vários componentes, crie um formato de estilo personalizado. Para obter mais informações sobre a criação de formatos de estilo personalizados, consulte “Personalizando cores e texto do componente” no capítulo “Usando componentes” de *Usando o Flash*.

### Exemplo

O código a seguir define a propriedade `arrow` de `ScrollPane2` como sendo 0x000000 (preto).

```
ScrollPane2.setStyleProperty("arrow", 0x000000);
```

### Consulte também

`FStyleFormat` (object)

## FScrollPane.setVScroll

### Disponibilidade

Flash Player 6.

### Uso

```
myScrollBar.setVScroll(display)
```

### Parâmetros

*display* Um valor Booleano que especifica se a barra de rolagem é exibida sempre (`true`) ou nunca é exibida (`false`), ou uma sequência de caracteres que determina que a barra de rolagem seja exibida somente quando necessário ("`auto`").

### Retorna

Nada.

### Descrição

Método; determina se a barra de rolagem vertical é exibida sempre (`true`), nunca é exibida (`false`) ou somente quando necessário ("`auto`"). O valor padrão é `auto`.

Se este método for chamado, o valor do parâmetro Vertical Scroll definido durante o processo de criação será cancelado.

### Exemplo

O código a seguir especifica que a barra de rolagem vertical de `display1` seja exibida sempre.

```
display1.setVScroll(true);
```

### Consulte também

FScrollPane.setHScroll

## FStyleFormat (object)

O objeto `FStyleFormat` permite definir ou alterar propriedades no formato de estilo global atribuídas a todos os componentes de interface de usuário Flash por padrão, ou criar novos formatos de estilo personalizados para uso com componentes de interface de usuário Flash ou componentes personalizados que você cria ou obtém de outras fontes. O formato de estilo global, ou objeto `globalStyleFormat`, é uma instância do objeto `FStyleFormat` que define as propriedades de formatação de cores e texto usadas para exibir todos os componentes de interface de usuário Flash.

Para criar um novo formato de estilo personalizado, você cria uma nova instância do objeto `FStyleFormat` usando o construtor `new FStyleFormat()`, define as propriedades de `FStyleFormat` que deseja incluir no formato de estilo e, a seguir, usa o método `FStyleFormat.addListener` para registrar instâncias de componentes no novo formato de estilo. Uma instância de componente pode “ouvir” mais de um formato de estilo, mas só pode obter o valor de um formato de estilo de uma propriedade específica. Se você adicionar um componente como um ouvinte de um formato de estilo, ele utilizará o novo formato de estilo para propriedades especificadas no formato e utilizará o antigo formato de estilo para todas as outras propriedades.

Não será preciso usar o construtor `FStyleFormat` para adicionar ou remover ouvintes ou definir ou alterar propriedades no formato de estilo global, porque o objeto `globalStyleFormat` existe no momento em que qualquer componente de interface de usuário Flash é inserido no Palco.

Você pode definir qualquer propriedade de `FStyleFormat` para uma única instância de um componente usando o método `setStyleProperty` disponível a todos os componentes de interface de usuário Flash. O uso de `setStyleProperty` permite definir uma propriedade para um componente sem criar uma instância do objeto `FStyleFormat`. O uso de `setStyleProperty` cancela a configuração de uma propriedade de formato de estilo específica atribuída ao componente sem alterar as outras configurações de propriedades. Para obter mais informações, consulte as entradas de `setStyleProperty` de componentes individuais.

Ao atribuir um valor de cor a uma propriedade de `FStyleFormat`, especifique uma cor RGB no formato `0xRRGGBB`.

Para obter mais informações sobre o formato de estilo global e sobre a criação de formatos de estilo personalizados, consulte “Personalizando cores e texto do componente” no capítulo “Usando componentes” de *Usando o Flash*.

## Resumo de métodos do objeto `FStyleFormat`

Método	Descrição
<code>FStyleFormat.addListener</code>	Registra um componente em formato de estilo.
<code>FStyleFormat.applyChanges</code>	Aplica as alterações efetuadas em valores de propriedades de um formato de estilo.
<code>FStyleFormat.removeListener</code>	Remove um componente como um ouvinte de um formato de estilo.

## Resumo das propriedades do objeto `FStyleFormat`

As tabelas a seguir listam os resumos das propriedades do objeto `FStyleFormat`.

Propriedade	Descrição
<code>FStyleFormat.arrow</code>	A cor da seta usada em barras de rolagem e listagens suspensas.
<code>FStyleFormat.background</code>	A cor da porção de fundo de um componente.
<code>FStyleFormat.backgroundDisabled</code>	A cor da porção de fundo de um componente desativado.
<code>FStyleFormat.check</code>	A cor da marca de seleção em uma caixa de seleção assinalada.
<code>FStyleFormat.darkshadow</code>	A cor da borda interna ou da porção de sombreamento mais escuro de um componente.
<code>FStyleFormat.face</code>	A cor principal do componente.
<code>FStyleFormat.foregroundDisabled</code>	A cor de primeiro plano do componente.
<code>FStyleFormat.highlight</code>	A cor da borda interna ou da porção de sombreamento mais escuro de um componente quando selecionado.
<code>FStyleFormat.highlight3D</code>	A cor da borda externa ou da porção de sombreamento mais claro de um componente quando selecionado.
<code>FStyleFormat.radioDot</code>	A cor do ponto em um botão de opção assinalado.
<code>FStyleFormat.scrollTrack</code>	A cor da trilha em uma barra de rolagem.
<code>FStyleFormat.selection</code>	A cor da barra de seleção que destaca um item de listagem em um componente.
<code>FStyleFormat.selectionDisabled</code>	A cor da barra de seleção que destaca um item de listagem em um componente desativado.
<code>FStyleFormat.selectionUnfocused</code>	A cor da barra de seleção quando o componente não tem foco de teclado.

Propriedade	Descrição
<code>FStyleFormat.shadow</code>	A cor da borda externa ou da porção de sombreamento claro de um componente.
<code>FStyleFormat.textAlign</code>	O alinhamento (esquerda, direita ou centro) do texto exibido em um componente.
<code>FStyleFormat.textBold</code>	Especifica se o texto será em negrito ( <code>true</code> ) ou não ( <code>false</code> ).
<code>FStyleFormat.textColor</code>	A cor de texto padrão em todos os componentes atribuída ao formato de estilo.
<code>FStyleFormat.textDisabled</code>	A cor do texto em um componente desativado.
<code>FStyleFormat.textFont</code>	O nome da fonte para exibição de texto.
<code>FStyleFormat.textIndent</code>	O recuo do texto a partir da margem esquerda até o primeiro caractere de texto, em pixels.
<code>FStyleFormat.textItalic</code>	Especifica se o texto será em itálico ( <code>true</code> ) ou não ( <code>false</code> ).
<code>FStyleFormat.textLeftMargin</code>	A margem de parágrafo esquerda do texto, em pixels.
<code>FStyleFormat.textRightMargin</code>	A margem de parágrafo direita do texto, em pixels.
<code>FStyleFormat.textSelected</code>	A cor de um item de listagem selecionado em um componente.
<code>FStyleFormat.textSize</code>	O tamanho do texto, em pontos.
<code>FStyleFormat.textUnderline</code>	Especifica se o texto será sublinhado ( <code>true</code> ) ou não ( <code>false</code> ).

## Construtor do objeto FStyleFormat

### Disponibilidade

Flash Player 6.

### Uso

```
new FStyleFormat()
```

### Parâmetros

Nenhum.

### Retorna

Uma instância do objeto `FStyleFormat`.

### Descrição

Método; cria um novo objeto `FStyleFormat`. Você cria novos objetos `FStyleFormat` para definir propriedades de texto e cores de formatos de estilo personalizados usados com componentes personalizados ou com os componentes de interface de usuário Flash. Como padrão, o objeto `globalStyleFormat` é atribuído a todos os componentes de interface de usuário Flash. Esse objeto é uma instância do objeto `FStyleFormat`. Não é preciso criar uma nova instância do objeto `FStyleFormat` para alterar propriedades no formato de estilo global, pois ela já existe. Você também pode usar `setStyleProperty` para alterar propriedades de instâncias específicas de componentes sem usar um construtor.

Para obter mais informações, consulte o método `setStyleProperty` disponível para cada componente—`FCheckBox.setStyleProperty`, `FComboBox.setStyleProperty` etc. Consulte também “Personalizando cores e texto do componente” no capítulo “Usando componentes” de *Usando o Flash*.

### Exemplo

O exemplo a seguir cria o novo formato de estilo `formStyleFormat`.

```
formStyleFormat = new StyleFormat();
```

## FStyleFormat.addListener

### Disponibilidade

Flash Player 6.

### Uso

```
myStyleFormat.addListener(component1 [, component2, ...componentN])
```

### Parâmetros

*component1 ... componentN* As instâncias de componentes a serem registradas em *myStyleFormat*.

### Retorna

Nada.

### Descrição

Método; registra os componentes especificados em *myStyleFormat*. Use este método para registrar instâncias de componentes de interface de usuário Flash ou componentes personalizados em um formato de estilo personalizado. Você também pode usar este método com a sintaxe a seguir para registrar um componente personalizado em um formato de estilo global usado por todos os componentes de interface de usuário Flash como padrão.

```
globalStyleFormat.addListener(customComponent);
```

### Exemplo

O código a seguir registra `formStyleFormat` com os componentes `myListBox`, `myComboBox` e `myScrollBar`.

```
formStyleFormat.addListener(myListBox, myComboBox, myScrollBar);
```

### Consulte também

```
FStyleFormat.applyChangesFStyleFormat.applyChanges, FStyleFormat.removeListener
```

## FStyleFormat.applyChanges

### Disponibilidade

Flash Player 6.

### Uso

```
myStyleFormat.applyChanges([propertyName1, ...propertyNameN])
```

```
myStyleFormat.applyChanges()
```

### Parâmetros

*propertyName1...propertyNameN* Uma série de seqüências de caracteres de texto que especificam as propriedades a serem atualizadas para todos os componentes atribuídos a *myStyleFormat*.

### Retorna

Nada.

### Descrição

Método; atualiza a instância do objeto de formato de estilo especificado e aplica as alterações a todos os componentes atribuídos ao formato. Você deve chamar este método ao adicionar ou remover ouvintes e definir ou alterar propriedades. Ao atualizar propriedades, recomenda-se o uso da primeira sintaxe para atualizar somente as propriedades para as quais você estiver especificando um novo valor.

Uso 1: Atualiza somente as propriedades especificadas nos parâmetros.

Uso 2: Atualiza todas as informações no formato de estilo (ou seja, componentes e propriedades atribuídos) tenham eles sido alterados ou não.

### Exemplo

Uso 1: O exemplo a seguir atualiza as propriedades `arrow` e `background`, mas não as propriedades `check` e `highlight`, em `formStyleFormat`.

```
formStyleFormat.arrow = 0x00ffaa;  
formStyleFormat.background = 0xaabbcc;  
formStyleFormat.check = 0x000000;  
formStyleFormat.highlight = 0xffffffff;  
formStyleFormat.applyChanges("arrow", "background");
```

Uso 2: O exemplo a seguir atualiza todas as propriedades em `formStyleFormat`—`arrow`, `background`, `check` e `highlight`.

```
formStyleFormat.arrow = 0x00ffaa;  
formStyleFormat.background = 0xaabbcc;  
formStyleFormat.check = 0x000000;  
formStyleFormat.highlight = 0xffffffff;  
formStyleFormat.applyChanges();
```

### Consulte também

`FStyleFormat.addListener`, `FStyleFormat.removeListener`

## FStyleFormat.arrow

### Disponibilidade

Flash Player 6.

### Uso

*myStyleFormat.arrow*

### Descrição

Propriedade; o valor de cor RGB da propriedade `arrow` usado em barras de rolagem e listagens suspensas em componentes como barras de rolagem, caixas de listagem e caixas de combinação. O valor da cor deverá estar no formato *0xRRGGBB*.

Você deve usar `FStyleFormat.applyChanges` ao atualizar propriedades com um novo valor.

### Exemplo

O código a seguir atribui o valor `0x800080` à propriedade `arrow` em `formStyleFormat`, gerando um seta roxa.

```
formStyleFormat.arrow = 0x800080;
```

## FStyleFormat.background

### Disponibilidade

Flash Player 6.

### Uso

*myStyleFormat.background*

### Descrição

Propriedade; o valor de cor RGB da porção de fundo de um componente. Por exemplo, em um botão de opção ou uma caixa de seleção, a porção de fundo é o espaço dentro da área de seleção; em uma caixa de listagem ou de combinação, a porção de fundo é a área de exibição. O valor da cor deverá estar no formato *0xRRGGBB*.

Você deve usar `FStyleFormat.applyChanges` ao atualizar propriedades com um novo valor.

### Exemplo

O código a seguir atribui o valor `0xFFE4E1` à propriedade `background` de `formStyleFormat`, produzindo um fundo rosa claro quando o componente é ativado.

```
formStyleFormat.background = 0xFFE4E1;
```

### Consulte também

`FStyleFormat.face`

## FStyleFormat.backgroundDisabled

### Disponibilidade

Flash Player 6.

### Uso

*myStyleFormat.backgroundDisabled*

### Descrição

Propriedade; o valor de cor RGB da porção de fundo de um componente desativado. Geralmente, a cor de fundo dos elementos desativados da interface de usuário é cinza claro. O valor da cor deverá estar no formato *0xRRGGBB*.

Você deve usar `FStyleFormat.applyChanges` ao atualizar propriedades com um novo valor.

### Exemplo

O código a seguir atribui o valor `0x808080` à propriedade `backgroundDisabled` de `formStyleFormat`, produzindo um fundo cinza quando o componente é desativado.

```
formStyleFormat.backgroundDisabled = 0x808080;
```

### Consulte também

`FStyleFormat.foregroundDisabled`



## FStyleFormat.check

### Disponibilidade

Flash Player 6.

### Uso

*myStyleFormat.check*

### Descrição

Propriedade; o valor de cor RGB da marca de seleção em uma caixa de seleção assinalada. O valor da cor deverá estar no formato *0xRRGGBB*.

Você deve usar `FStyleFormat.applyChanges` ao atualizar propriedades com um novo valor.

### Exemplo

O código a seguir atribui o valor `0x228B22` à propriedade `check` de `formStyleFormat`, produzindo uma seta verde escura.

```
formStyleFormat.check = 0x228B22;
```

## FStyleFormat.darkshadow

### Disponibilidade

Flash Player 6.

### Uso

*myStyleFormat.darkshadow*

### Descrição

Propriedade; o valor de cor RGB da borda interna ou da porção de sombreado mais escuro de um componente — por exemplo, a extremidade interna do círculo de um botão de opção desmarcado ou de uma caixa de seleção desmarcada. O valor da cor deverá estar no formato *0xRRGGBB*.

Você deve usar `FStyleFormat.applyChanges` ao atualizar propriedades com um novo valor.

### Exemplo

O código a seguir atribui o valor `0x0000CD` à propriedade `darkshadow` de `formStyleFormat`, produzindo uma borda interna na cor azul, em tom médio.

```
formStyleFormat.darkshadow = 0x0000CD;
```

### Consulte também

`FStyleFormat.highlight`, `FStyleFormat.shadow`

## FStyleFormat.face

### Disponibilidade

Flash Player 6.

### Uso

`myStyleFormat.face`

### Descrição

Propriedade; o valor RGB da cor principal de um componente — por exemplo, o cinza usado no componente PushButton ou ScrollBar. O valor da cor deverá estar no formato *0xRRGGBB*.

Você deve usar `FStyleFormat.applyChanges` ao atualizar propriedades com um novo valor.

### Exemplo

O código a seguir atribui o valor `0x32CD32` à propriedade `face` de `formStyleFormat`, produzindo botões de ação e barras de rolagem na cor verde limão.

```
formStyleFormat.face = 0x32CD32;
```

## FStyleFormat.foregroundDisabled

### Disponibilidade

Flash Player 6.

### Uso

`myStyleFormat.foregroundDisabled`

### Descrição

Propriedade; o valor de cor RGB para o primeiro plano de um componente desativado.

Geralmente, a cor de primeiro plano dos elementos desativados da interface de usuário é cinza médio. O valor da cor deverá estar no formato *0xRRGGBB*.

Você deve usar `FStyleFormat.applyChanges` ao atualizar propriedades com um novo valor.

### Exemplo

O código a seguir atribui o valor `0x708090` à propriedade `foregroundDisabled` de `formStyleFormat`, produzindo um primeiro plano cinza para os componentes desativados.

```
formStyleFormat.foregroundDisabled = 0x708090;
```

### Consulte também

`FStyleFormat.backgroundDisabled`

## FStyleFormat.highlight

### Disponibilidade

Flash Player 6.

### Uso

*myStyleFormat.highlight*

### Descrição

Propriedade; o valor de cor RGB da borda interna ou da porção de sombreado mais escuro de um componente quando selecionado — por exemplo, a extremidade interna do círculo de um botão de opção ou de uma caixa de seleção. O valor da cor deverá estar no formato *0xRRGGBB*.

Você deve usar `FStyleFormat.applyChanges` ao atualizar propriedades com um novo valor.

### Exemplo

O código a seguir atribui o valor `0xFF00FF` à propriedade `highlight` de `formStyleFormat`, produzindo uma borda interna na cor amarelo brilhante quando o componente é selecionado.

```
formStyleFormat.highlight = 0xFF00FF;
```

### Consulte também

`FStyleFormat.darkshadow`

## FStyleFormat.highlight3D

### Disponibilidade

Flash Player 6.

### Uso

*myStyleFormat.highlight3D*

### Descrição

Propriedade; o valor de cor RGB da borda externa ou da porção de sombreado mais clara de um componente quando selecionado — por exemplo, a extremidade externa do círculo de um botão de opção ou de uma caixa de seleção. O valor da cor deverá estar no formato *0xRRGGBB*.

Você deve usar `FStyleFormat.applyChanges` ao atualizar propriedades com um novo valor.

### Exemplo

O código a seguir atribui o valor `0x40E0D0` à propriedade `highlight3D` de `formStyleFormat`, produzindo uma borda externa na cor turquesa brilhante quando o componente é selecionado.

```
formStyleFormat.highlight3D = 0x40E0D0;
```

### Consulte também

`FStyleFormat.shadow`

## FStyleFormat.radioDot

### Disponibilidade

Flash Player 6.

### Uso

*myStyleFormat.radioDot*

### Descrição

Propriedade; o valor de cor RGB do ponto de seleção do botão de opção de um componente. O valor da cor deverá estar no formato *0xRRGGBB*.

Você deve usar `FStyleFormat.applyChanges` ao atualizar propriedades com um novo valor.

### Exemplo

O código a seguir atribui o valor `0xFF12AC` à propriedade `radioDot` de `formStyleFormat`, produzindo um ponto de seleção rosa.

```
formStyleFormat.radioDot = 0xFF12AC;
```

## FStyleFormat.removeListener

### Disponibilidade

Flash Player 6.

### Uso

*myStyleFormat.removeListener(component)*

### Parâmetros

*component* O componente a ser removido do formato de estilo.

### Descrição

Método; remove um componente atribuído ao formato de estilo.

- Se você remover um componente de interface de usuário Flash como um ouvinte do formato de estilo global e não o atribuir (não o adicionar como um ouvinte) a um formato de estilo personalizado, os cliques de filme dos elementos de aparência serão exibidos como foram originalmente criados pelo designer de componentes sem um valor de propriedade atribuído.
- Se você remover um componente de interface de usuário Flash como um ouvinte de um formato de estilo personalizado, o componente não mais usará os valores de propriedade no formato de estilo personalizado e, em vez disso, usará os valores especificados para essas propriedades no objeto de formato de estilo global.
- Se você remover um componente personalizado como um ouvinte de um formato de estilo personalizado sem adicioná-lo a um novo formato de estilo personalizado, o componente usará os valores definidos para as propriedades no formato de estilo global sempre que possível e, caso não seja possível, exibirá os elementos de aparência sem um valor de propriedade.

### Exemplo

O exemplo a seguir remove o componente `check1` de `globalStyleFormat`.

```
globalStyleFormat.removeListener(check1);
```

### Consulte também

`FStyleFormat.addListener`, `FStyleFormat.applyChanges`

## FStyleFormat.scrollTrack

### Disponibilidade

Flash Player 6.

### Uso

`myStyleFormat.scrollTrack`

### Descrição

Propriedade; o valor de cor RGB da porção de trilha de uma barra de rolagem. O componente ScrollBar é usado pelos componentes ScrollPane, ListBox e ComboBox, e a alteração do valor da propriedade `scrollTrack` no formato de estilo global altera a cor da trilha de rolagem em todos os componentes que usam barras de rolagem. O valor da cor deverá estar no formato *0xRRGGBB*.

Você deve usar `FStyleFormat.applyChanges` ao atualizar propriedades com um novo valor.

### Exemplo

O código a seguir atribui o valor `0xA0522D` à propriedade `scrollTrack` em `formStyleFormat`, gerando uma trilha de rolagem marrom.

```
formStyleFormat.scrollTrack = 0xA0522D;
```

## FStyleFormat.selection

### Disponibilidade

Flash Player 6.

### Uso

`myStyleFormat.selection`

### Descrição

Propriedade; o valor de cor RGB da barra usada para realçar o item selecionado na listagem de um componente. Essa propriedade atua em conjunto com a propriedade `FStyleFormat.textSelected` para exibir itens selecionados e você deve coordenar as cores para facilitar a leitura dos textos. Por exemplo, o formato de estilo global atribui um valor de cor azul à propriedade `selection`, usada para exibir a barra de seleção nos componentes ListBox e ComboBox, e atribui um valor de cor branca à propriedade `textSelected`; essa combinação de cores permite que o usuário visualize facilmente o texto selecionado. O valor da cor deverá estar no formato *0xRRGGBB*.

Você deve usar `FStyleFormat.applyChanges` ao atualizar propriedades com um novo valor.

### Exemplo

O código a seguir atribui o valor `0x87CEEB` à propriedade `selection` em `formStyleFormat`, gerando uma barra de seleção azul celeste.

```
formStyleFormat.selection = 0x87CEEB;
```

### Consulte também

`FStyleFormat.textSelected`

## FStyleFormat.selectionDisabled

### Disponibilidade

Flash Player 6.

### Uso

*myStyleFormat.selectionDisabled*

### Descrição

Propriedade; o valor de cor RGB da barra de seleção usada para realçar um item de listagem em um componente desativado. O valor da cor deverá estar no formato *0xRRGGBB*.

Você deve usar `FStyleFormat.applyChanges` ao atualizar propriedades com um novo valor.

### Exemplo

O código a seguir atribui o valor 0x708090 à propriedade `selectionDisabled` de `formStyleFormat`, produzindo uma barra de seleção na cor cinza azulada quando o componente é desativado.

```
formStyleFormat.selectionDisabled = 0x708090;
```

### Consulte também

`FStyleFormat.selection`

## FStyleFormat.selectionUnfocused

### Disponibilidade

Flash Player 6.

### Uso

*myStyleFormat.selectionUnfocused*

### Descrição

Propriedade; o valor de cor RGB da barra de seleção (realce) na listagem de um componente quando o componente não tem o foco do teclado. O valor da cor deverá estar no formato *0xRRGGBB*.

Você deve usar `FStyleFormat.applyChanges` ao atualizar propriedades com um novo valor.

### Exemplo

O código a seguir atribui o valor 0xaabbcc à propriedade `selectionUnfocused` de `formStyleFormat`.

```
formStyleFormat.selectionUnfocused = 0xaabbcc;
```

### Consulte também

`FStyleFormat.selection`

## FStyleFormat.shadow

### Disponibilidade

Flash Player 6.

### Uso

*myStyleFormat.shadow*

### Descrição

Propriedade; o valor de cor RGB da borda externa ou da porção de sombreamento clara de um componente — por exemplo, a extremidade externa do círculo de um botão de opção desmarcado ou de uma caixa de seleção desmarcada. O valor da cor deverá estar no formato *0xRRGGBB*.

Você deve usar `FStyleFormat.applyChanges` ao atualizar propriedades com um novo valor.

### Exemplo

O código a seguir atribui o valor `0x008080` à propriedade `shadow` de `formStyleFormat`, produzindo uma borda externa na cor azul esverdeada para componentes de botões de opção e caixas de seleção desmarcadas.

```
formStyleFormat.shadow = 0x008080;
```

### Consulte também

`FStyleFormat.check`

## FStyleFormat.textAlign

### Disponibilidade

Flash Player 6.

### Uso

*myStyleFormat.textAlign*

### Descrição

Propriedade; uma sequência de caracteres de texto que especifica o alinhamento à direita, à esquerda ou no centro para o texto exibido em todos os componentes atribuídos ao formato de estilo. A configuração padrão é `left`.

Você deve usar `FStyleFormat.applyChanges` ao atualizar propriedades com um novo valor.

### Exemplo

O código a seguir alinha à direita todo o texto em componentes usando `formStyleFormat`.

```
formStyleFormat.textAlgin = "right";
```

### Consulte também

`FStyleFormat.textIndent`, `FStyleFormat.textLeftMargin`,  
`FStyleFormat.textRightMargin`

## FStyleFormat.textBold

### Disponibilidade

Flash Player 6.

### Uso

`myStyleFormat.textBold`

### Descrição

Propriedade; um valor Booleano que especifica se todo o texto exibido em componentes que usam o formato de estilo ficará em negrito (`true`) ou não (`false`). A configuração padrão é `false`.

Você deve usar `FStyleFormat.applyChanges` ao atualizar propriedades com um novo valor.

### Exemplo

O código a seguir determina que todo o texto exibido em componentes atribuídos a `formStyleFormat` fique em negrito.

```
formStyleFormat.textBold = true;
```

### Consulte também

`FStyleFormat.textItalic`, `FStyleFormat.textUnderline`

## FStyleFormat.textColor

### Disponibilidade

Flash Player 6.

### Uso

`myStyleFormat.textColor`

### Descrição

Propriedade; o valor de cor RGB para a cor de texto padrão em todos os componentes atribuídos ao formato de estilo. O valor da cor deverá estar no formato *0xRRGGBB*.

Você deve usar `FStyleFormat.applyChanges` ao atualizar propriedades com um novo valor.

### Exemplo

O código a seguir atribui o valor `0x000000` à propriedade `textColor` em `formStyleFormat`, produzindo texto em preto.

```
formStyleFormat.textColor = 0x000000;
```

### Consulte também

`FStyleFormat.textDisabled`, `FStyleFormat.textSelected`



## FStyleFormat.textDisabled

### Disponibilidade

Flash Player 6.

### Uso

*myStyleFormat.textDisabled*

### Descrição

Propriedade; o valor de cor RGB para a cor de texto padrão usada para exibir texto em componentes desativados atribuídos ao formato de estilo. O valor da cor deverá estar no formato *0xRRGGBB*.

Você deve usar `FStyleFormat.applyChanges` ao atualizar propriedades com um novo valor.

### Exemplo

O código a seguir atribui o valor `0xC0C0C0` à propriedade `textDisabled` de `formStyleFormat`, produzindo um texto prateado quando o componente é desativado.

```
formStyleFormat.textDisabled = 0xC0C0C0;
```

### Consulte também

`FStyleFormat.textAlign`, `FStyleFormat.textSelected`

## FStyleFormat.textFont

### Disponibilidade

Flash Player 6.

### Uso

*myStyleFormat.textFont*

### Descrição

Propriedade; uma sequência de caracteres de texto que especifica a fonte usada para exibir texto em todos os componentes atribuídos ao formato de estilo.

Você deve usar `FStyleFormat.applyChanges` ao atualizar propriedades com um novo valor.

### Exemplo

O código a seguir atribui o valor `Arial` à propriedade `textFont` de `formStyleFormat`.

```
formStyleFormat.textFont = "Arial";
```

## FStyleFormat.textIndent

### Disponibilidade

Flash Player 6.

### Uso

*myStyleFormat.textIndent*

### Descrição

Propriedade; um número inteiro que especifica o recuo, em pixels, a partir da margem esquerda até o primeiro caractere de texto para todo o texto exibido usando o formato de estilo.

### Exemplo

O código a seguir recua todo o texto exibido por `formStyleFormat` em 5 pixels.

```
formStyleFormat.textIndent = 5;
```

### Consulte também

`FStyleFormat.textAlign`, `FStyleFormat.textLeftMargin`

## FStyleFormat.textItalic

### Disponibilidade

Flash Player 6.

### Uso

```
myStyleFormat.textItalic
```

### Descrição

Propriedade; um valor Booleano que especifica se todo o texto exibido em componentes que usam o formato de estilo ficará em itálico (`true`) ou não (`false`). A configuração padrão é `false`.

Você deve usar `FStyleFormat.applyChanges` ao atualizar propriedades com um novo valor.

### Exemplo

O código a seguir determina que todo o texto exibido em componentes atribuídos a `formStyleFormat` fique em itálico.

```
formStyleFormat.textItalic = true;
```

### Consulte também

`FStyleFormat.textBold`

## FStyleFormat.textLeftMargin

### Disponibilidade

Flash Player 6.

### Uso

```
myStyleFormat.textLeftMargin
```

### Descrição

Propriedade; um número inteiro que especifica a margem esquerda do parágrafo, em pixels, para todo o texto exibido em componentes atribuído ao formato de estilo.

Você deve usar `FStyleFormat.applyChanges` ao atualizar propriedades com um novo valor.

### Exemplo

O código a seguir especifica um valor de 4 pixels para a propriedade `textLeftMargin` de `formStyleFormat`.

```
formStyleFormat.textLeftMargin = 4;
```

### Consulte também

`FStyleFormat.textRightMargin`

## FStyleFormat.textRightMargin

### Disponibilidade

Flash Player 6.

### Uso

*myStyleFormat.textRightMargin*

Propriedade; um número inteiro que especifica a margem direita do parágrafo, em pixels, para todo o texto exibido em componentes atribuído ao formato de estilo.

Você deve usar `FStyleFormat.applyChanges` ao atualizar propriedades com um novo valor.

### Exemplo

O código a seguir especifica um valor de 4 pixels para a propriedade `textRightMargin` de `formStyleFormat`.

```
formStyleFormat.textRightMargin = 4;
```

### Consulte também

`FStyleFormat.textLeftMargin`

## FStyleFormat.textSelected

### Disponibilidade

Flash Player 6.

### Uso

*myStyleFormat.textSelected*

### Descrição

Propriedade; o valor de cor RGB que especifica a cor do texto selecionado em componentes atribuídos ao formato de estilo. Essa propriedade atua em conjunto com a propriedade `FStyleFormat.selection` para exibir itens de listagem selecionados e você deve coordenar as cores para facilitar a leitura dos textos. Por exemplo, o formato de estilo global atribui um valor de cor azul à propriedade `selection`, usada para exibir a barra de seleção nos componentes `ListBox` e `ComboBox`, e atribui um valor de cor branca à propriedade `textSelected`; essa combinação de cores permite que o usuário visualize facilmente o texto selecionado. O valor da cor deverá estar no formato *0xRRGGBB*.

Você deve usar `FStyleFormat.applyChanges` ao atualizar propriedades com um novo valor.

### Exemplo

O código a seguir atribui o valor `0xffffffff` à propriedade `textSelected` de `formStyleFormat`, produzindo um texto branco quando o componente é selecionado.

```
formStyleFormat.textSelected = 0xffffffff;
```

### Consulte também

`FStyleFormat.selection`, `FStyleFormat.textDisabled`

## FStyleFormat.textSize

### Disponibilidade

Flash Player 6.

### Uso

*myStyleFormat.textSize*

### Descrição

Propriedade; um número inteiro que especifica o tamanho do ponto do texto exibido em componentes atribuídos ao formato de estilo. A configuração padrão para esta propriedade é texto de 12 pontos.

Você deve usar `FStyleFormat.applyChanges` ao atualizar propriedades com um novo valor.

### Exemplo

O código a seguir especifica texto de 10 pontos para todos os componentes atribuídos a `formStyleFormat`.

```
formStyleFormat.textSize = 10;
```

### Consulte também

`FStyleFormat.textFont`

## FStyleFormat.textUnderline

### Disponibilidade

Flash Player 6.

### Uso

*myStyleFormat.textUnderline*

### Descrição

Propriedade; especifica se o texto exibido em componentes que usam o formato de estilo especificado será sublinhado (`true`) ou não (`false`). A configuração padrão é `false`.

Você deve usar `FStyleFormat.applyChanges` ao atualizar propriedades com um novo valor.

### Exemplo

O código a seguir determina que todo o texto exibido em componentes atribuídos a `formStyleFormat` seja sublinhado.

```
formStyleFormat.textUnderline = true;
```

### Consulte também

`FStyleFormat.textBold`, `FStyleFormat.textItalic`

## Function (objeto)

O objeto Function encontra-se disponível no Flash MX.

### Resumo das propriedades do objeto Function

Método	Descrição
<code>Function.prototype</code>	Refere-se a um objeto que é o protótipo de uma classe.

### Resumo dos métodos do objeto Function

Método	Descrição
<code>Function.apply</code>	Ativa o código ActionScript para chamar uma função.
<code>Function.call</code>	Invoca a função representada por um objeto Function.

## Function.apply

### Disponibilidade

Flash Player 6.

### Uso

```
myFunction.apply(thisObject, argumentsObject)
```

### Parâmetros

*thisObject* O objeto ao qual *myFunction* é aplicada.

*argumentsObject* Uma matriz cujos elementos são passados para *myFunction* como parâmetros.

### Retorna

Qualquer valor que a função chamada especifica.

### Descrição

Método; especifica o valor de `this` para ser usado em qualquer função que ActionScript chame. Este método também especifica os parâmetros a serem passados a qualquer função chamada. Como `apply` é um método do objeto Function, ele também é um método de todo objeto de função em ActionScript.

Os parâmetros são especificados como um objeto Array. Em geral, isso é útil quando o número de parâmetros a serem passados só é conhecido quando o script é realmente executado.

### Exemplo

As invocações de função a seguir são equivalentes.

```
Math.atan2(1, 0)  
Math.atan2.apply(null, [1, 0])
```

Você pode construir um filme Flash que contenha campos de entrada que permitam ao usuário inserir o nome de uma função a ser invocada e nenhum ou qualquer número de parâmetros a serem passados à função. O botão “Chamar” usaria então o método `apply` para chamar a função, especificando os parâmetros.

No exemplo, o usuário especifica um nome de função em um campo de texto de entrada denominado `functionName`. A quantidade de parâmetros é especificada em um campo de texto de entrada denominado `numParameters`. Até 10 parâmetros são especificados em campos de texto denominados `parameter1`, `parameter2`, até `parameter10`.

```
on (release) {
    callTheFunction();
}
...
function callTheFunction()
{
    var theFunction = eval(functionName.text);
    var n = Number(numParameters);
    var parameters = [];
    for (var i = 0; i < n; i++) {
        parameters.push(eval("parameter" + i));
    }
    theFunction.apply(null, parameters);
}
```

## Function.call

### Disponibilidade

Flash Player 6.

### Uso

```
myFunction.call(thisObject, parameter1, ..., parameterN)
```

### Parâmetros

*thisObject*    Especifica o valor de `this` no corpo da função.

*parameter1*    Um parâmetro a ser passado a *myFunction*. Você pode especificar zero ou mais parâmetros.

*parameterN*

### Retorna

Nada.

### Descrição

Método; invoca a função representada por um objeto `Function`. Toda função em `ActionScript` é representada por um objeto `Function` para que todas as funções ofereçam suporte ao método `call`.

Em quase todos os casos, o operador de chamada de função (`()`) pode ser usado no lugar do método `call`. O operador de chamada de função produz um código conciso e de fácil leitura. O método `call` é útil principalmente quando o parâmetro `this` da invocação da função precisa ser explicitamente controlado. Normalmente, se uma função é invocada como um método de um objeto, no corpo da função, `this` é definido para `myObject` como no exemplo a seguir:

```
myObject.myMethod(1, 2, 3);
```

Em algumas situações, pode ser preferível que `this` aponte para algum outro lugar; por exemplo, se uma função tiver que ser invocada como um método de um objeto, mas não estiver realmente armazenada como um método desse objeto.

```
myObject.myMethod.call(myOtherObject, 1, 2, 3);
```

Você pode passar o valor `null` para o parâmetro *thisObject* para invocar uma função como uma função regular e não como um método de um objeto. Por exemplo, as invocações de função a seguir são equivalentes:

```
Math.sin(Math.PI / 4)
Math.sin.call(null, Math.PI / 4)
```

### Exemplo

Este exemplo usa o método `call` para fazer uma função se comportar como um método de outro objeto, sem armazenar a função no objeto.

```
function MyObject() {
}
function MyMethod(obj) {
    trace("this == obj? " + (this == obj));
}
var obj = new MyObject();
MyMethod.call(obj, obj);
```

A ação `trace` envia o seguinte código para a janela Saída:

```
this == obj? true
```

## Function.prototype

### Disponibilidade

Flash Player 6.

### Uso

```
myFunction.prototype
```

### Descrição

Propriedade; em uma função de construtor, a propriedade `prototype` refere-se a um objeto que é o protótipo da classe construída. Cada instância da classe que é criada pela função de construtor herda todas as propriedades e métodos do objeto protótipo.

## fscommand

### Disponibilidade

Flash Player 3.

### Uso

```
fscommand("command", "parameters")
```

### Parâmetros

*comand* Uma sequência de caracteres passada ao aplicativo host para qualquer uso ou um comando passado ao Flash Player independente.

*parameters* Uma sequência de caracteres passada ao aplicativo host para qualquer uso ou um valor passado ao Flash Player.

### Retorna

Nada.

## Descrição

Ação; permite que o filme Flash se comunique com o Flash Player ou com o programa que hospeda o Flash Player, como um navegador da Web. Você também pode usar a ação `fscommand` para passar mensagens para o Macromedia Director ou para o Visual Basic, Visual C++ e outros programas que possam hospedar controles ActiveX.

Uso 1: Para enviar uma mensagem ao Flash Player, você deve usar comandos e parâmetros predefinidos. A tabela a seguir mostra os valores que podem ser especificados para os parâmetros *command* e *parameters* da ação `fscommand` para controlar um filme reproduzido na versão independente do Flash Player (incluindo projetores):

Comando	Parâmetros	Objetivo
<code>quit</code>	Nenhum	Fecha o projetor.
<code>fullscreen</code>	<code>true</code> ou <code>false</code>	A especificação de <code>true</code> define o Flash Player no modo de tela cheia. A especificação de <code>false</code> faz o exibidor voltar à exibição de menu normal.
<code>allowscale</code>	<code>true</code> ou <code>false</code>	A especificação de <code>false</code> define o exibidor para que o filme seja sempre desenhado em seu tamanho original e nunca escalado. A especificação de <code>true</code> força o filme a ser escalado para 100% do exibidor.
<code>showmenu</code>	<code>true</code> ou <code>false</code>	A especificação de <code>true</code> ativa o conjunto completo de itens do menu de contexto. A especificação de <code>false</code> torna esmaecidos todos os itens do menu de contexto, exceto Sobre o Flash Player.
<code>exec</code>	Caminho para o aplicativo	Executa um aplicativo no projetor.
<code>trapallkeys</code>	<code>true</code> ou <code>false</code>	A especificação de <code>true</code> envia todos os eventos de teclas, incluindo as teclas de aceleração, para o identificador <code>onClipEvent(keyDown/keyUp)</code> no Flash Player.

Uso 2: Para usar a ação `fscommand` para enviar uma mensagem para uma linguagem de script como JavaScript em um navegador da Web, você pode passar dois parâmetros quaisquer nos parâmetros *command* e *parameters*. Esses parâmetros podem ser seqüências de caracteres ou expressões e serão usados em uma função JavaScript que “captura”, ou manipula, a ação `fscommand`.

Em um navegador da Web, a ação `fscommand` chama a função JavaScript `movienam_DoFSCommand` na página HTML que contém o filme Flash. O `movienam` é o nome do Flash Player conforme atribuído pelo atributo `NAME` da tag `EMBED` ou pela propriedade `ID` da tag `OBJECT`. Se o nome atribuído ao Flash Player for `myMovie`, a função JavaScript chamada será `myMovie_DoFSCommand`.

Uso 3: 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.

Uso 4: 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 tratadas na linguagem de programação do ambiente. Para obter mais informações, use as palavras-chave `Flash method` e faça uma pesquisa no Flash Support Center.



## Exemplo

Uso 1: No exemplo a seguir, a ação `fscommand` determina que o Flash Player ajuste o filme para toda a tela do monitor quando o botão for liberado.

```
on(release){
    fscommand("fullscreen", true);
}
```

Uso 2: O exemplo a seguir usa a ação `fscommand` aplicada a um botão em Flash para abrir uma caixa de mensagem JavaScript em uma página HTML. A mensagem em si é enviada para JavaScript como o parâmetro de `fscommand`.

Você deve adicionar uma função à página HTML que contém o filme Flash. Esta função `myMovie_DoFSCommand` repousa na página HTML e aguarda uma ação `fscommand` no Flash. Quando um `fscommand` é disparado no Flash (por exemplo, quando um usuário pressiona o botão), as seqüências de caracteres `command` e `parameter` são passadas para a função `myMovie_DoFSCommand`. Você pode usar as seqüências de caracteres passadas no seu código JavaScript ou VBScript da maneira que desejar. Neste exemplo, a função contém um comando condicional `if` que verifica se a seqüência de caracteres de comando é "messagebox". Se for, uma caixa de alerta JavaScript (ou "messagebox") é aberta e exibe o conteúdo da seqüência de caracteres `parameters`.

```
function myMovie_DoFSCommand(command, args) {
    if (command == "messagebox") {
        alert(args);
    }
}
```

No documento Flash, adicione a ação `fscommand` a um botão:

```
fscommand("messagebox", "Esta é uma caixa de mensagem chamada de dentro do  
Flash.")
```

Você também pode usar expressões para a ação `fscommand` e seus parâmetros, como no exemplo a seguir:

```
fscommand("messagebox", "Olá, " + nome + ", bem-vindo(a) ao nosso site na  
Web!")
```

Para testar o filme, escolha Arquivo > Visualizar Publicação > HTML.

**Observação:** Se você publicar o seu filme usando o Flash com o modelo FSCCommand nas Configurações de Publicação em HTML, a função `myMovie_DoFSCommand` será inserida automaticamente. Os atributos `NAME` e `ID` do filme serão o nome do arquivo. Por exemplo, para o arquivo `myMovie.fla`, os atributos seriam definidos como `myMovie`.

## function

### Disponibilidade

Flash Player 5.

### Uso

```
function functionname ([parameter0, parameter1,...parameterN]){
    comando(s)
}
function ([parameter0, parameter1,...parameterN]){
    comando(s)
}
```

## Parâmetros

*functionname* O nome da nova função.

*parameter* Um identificador que representa um parâmetro a ser passado para a função. Esses parâmetros são opcionais.

*comando(s)* Qualquer instrução ActionScript definida para o corpo da função.

## Retorna

Nada.

## 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 parâmetros para ela. Os parâmetros são espaços reservados para valores sobre os quais a função fará suas operações. Você pode passar parâmetros diferentes para uma função todas as vezes que chamá-la. Isso permite reutilizar uma função em diversas situações diferentes.

Use a ação *return no(s) comando(s)* de uma função para fazer a função apresentar, ou gerar, um valor.

Uso 1: Declara uma função com o *nome da função*, os *parâmetros* 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 apresenta a mesma. Esta sintaxe é usada em expressões e é particularmente útil para a instalação de métodos em objetos.

## Exemplo

Uso 1: O exemplo a seguir define a função *sqr*, que aceita um parâmetro e retorna o quadrado ( $x \times x$ ) do parâmetro. 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)

### Disponibilidade

Flash Player 4. Esse operador foi substituído no Flash 5 pelo operador >= (maior ou igual a).

### Uso

*expressão1* ge *expressão2*

### Parâmetros

*expression1*, *expression2* Números, seqüências de caracteres ou variáveis

### Retorna

Nada.

### Descrição

Operador (comparação); compara a representação em seqüência de caracteres da *expressão1* com a representação em seqüência de caracteres da *expressão2* e retorna true se a *expressão1* for maior ou igual à *expressão2*; caso contrário, retorna false.

### Consulte também

>= (maior ou igual a)

## getProperty

### Disponibilidade

Flash Player 4.

### Uso

getProperty(*instancename* , *property*)

### Parâmetros

*instancename* O nome da instância de um clipe de filme para o qual a propriedade está sendo recuperada.

*property* Uma propriedade de um clipe de filme.

### Retorna

Nada.

### Descrição

Função; retorna o valor de *property* especificada para o clipe de filme *instancename*.

### Exemplo

O exemplo a seguir recupera a coordenada do eixo horizontal (\_x) do clipe de filme myMovie e atribui a coordenada à variável myMovieX:

```
myMovieX = getProperty(_root.myMovie, _x);
```

## getTimer

### Disponibilidade

Flash Player 4.

### Uso

```
getTimer()
```

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Função; retorna o número de milissegundos decorridos desde o início da reprodução do filme.

## getURL

### Disponibilidade

Flash 2. As opções GET e POST só estão disponíveis no Flash 4 e versões posteriores.

### Uso

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

### Parâmetros

*url* O URL de onde o documento será obtido.

*window* Um parâmetro opcional que especifica a janela ou quadro HTML em que o documento deve ser carregado. Você pode inserir o nome de uma janela específica ou escolher um dos seguintes nomes de destino reservados:

- `_self` especifica o quadro atual na janela atual.
- `_blank` especifica uma nova janela.
- `_parent` especifica a origem do quadro atual.
- `_top` especifica o quadro de nível superior na janela atual.

*variables* Um método GET ou POST para envio de variáveis. Caso não haja variáveis, omitir esse parâmetro. O método GET anexa as variáveis ao final do URL e é usado para pequenos números. O método POST envia as variáveis em um cabeçalho HTTP separado e é usado com longas seqüências de caracteres.

### Retorna

Nada.

### 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.

### 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`, `XML.send`, `XML.sendAndLoad`, `XMLSocket.send`

## getVersion

### Disponibilidade

Flash Player 5.

### Uso

```
getVersion()
```

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Função; retorna uma sequência de caracteres contendo informações sobre a versão e plataforma do Flash Player.

A função `getVersion` só retorna informações para Flash Player 5 ou versões posteriores.

### Exemplo

O exemplo a seguir mostra uma sequência de caracteres apresentada como resposta 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).

## \_global

### Disponibilidade

Flash Player 6.

### Uso

```
_global.identifier
```

### Parâmetros

Nenhum.

**Retorna**

Uma referência ao objeto global que possui as principais classes ActionScript, como String, Object, Math e Array.

**Descrição**

Identificador; cria variáveis globais, objetos ou classes. Por exemplo, você poderia criar uma biblioteca que é exposta como um objeto global ActionScript, muito parecido com o objeto Math ou Date. Diferentemente das variáveis declaradas na Linha de tempo ou em nível local, as variáveis e funções globais são visíveis em todas as Linhas de Tempo e escopos no filme Flash, desde que não sejam encobertas por identificadores com nomes idênticos em escopos internos.

**Exemplo**

O exemplo a seguir cria uma função de nível superior `factorial` que está disponível para todas as Linhas de tempo e escopos no filme Flash:

```
_global.factorial = function (n) {  
    if (n <= 1) {  
        return 1;  
    }  
    else {  
        return n * factorial(n-1);  
    }  
}
```

**Consulte também**

`var`, `set variable`

## globalStyleFormat

**Disponibilidade**

Flash Player 6.

**Uso**

`globalStyleFormat.styleProperty`

**Parâmetros**

*styleProperty* Uma propriedade do objeto `FStyleFormat`.

**Retorna**

Nada.

**Descrição**

Instância de objeto; uma instância do objeto `FStyleFormat` que define as propriedades de formato de estilo para componentes de interface de usuário Flash. A instância `globalStyleFormat` fica disponível quando um componente de interface de usuário Flash é inserido no Palco. Você define ou altera propriedades de formato de estilo para componentes de interface de usuário Flash editando as propriedades na instância de objeto `globalStyleFormat`. Para obter mais informações, consulte “Personalizando cores e texto do componente” no capítulo “Usando componentes” de *Usando o Flash*.

**Exemplo**

O código a seguir define a propriedade `arrow` da propriedade `FStyleFormat` para a instância `globalStyleFormat`.

```
globalStyleFormat.arrow = 0x800080;
```

**Consulte também**

`FStyleFormat` (object)

## gotoAndPlay

### Disponibilidade

Flash 2.

### Uso

```
gotoAndPlay(scene, frame)
```

### Parâmetros

*scene* O nome da cena para onde a reprodução é enviada.

*frame* O número do quadro ou rótulo para onde a reprodução é enviada.

### Retorna

Nada.

### 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.

### Exemplo

Quando o usuário clica em um botão com a ação `gotoAndPlay` atribuída, a reprodução é enviada para o Quadro 16 e começa a reproduzir.

```
on(release) {  
    gotoAndPlay(16);  
}
```

## gotoAndStop

### Disponibilidade

Flash 2.

### Uso

```
gotoAndStop(scene, frame)
```

### Parâmetros

*scene* O nome da cena para onde a reprodução é enviada.

*frame* O número do quadro ou rótulo para onde a reprodução é enviada.

### Retorna

Nada.

### Descrição

Ação; envia a reprodução para o quadro especificado em uma cena e a interrompe. Se não for especificada uma cena, a reprodução segue para o quadro especificado na cena atual.

### 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)

### Disponibilidade

Flash Player 4. Este operador foi reprovado no Flash 5 e substituído pelo novo operador > (maior que).

### Uso

*expressão1* gt *expressão2*

### Parâmetros

*expression1*, *expression2* Números, seqüências de caracteres ou variáveis.

### Descrição

Operador (comparação); compara a representação em seqüência de caracteres da *expressão1* com a representação em seqüência de caracteres da *expressão2* e retorna true se a *expressão1* for maior que a *expressão2*; caso contrário, retorna false.

### Consulte também

> (maior que)

## \_highquality

### Disponibilidade

Flash Player 4.

### Uso

\_highquality

### Descrição

Propriedade (global); especifica o nível de sem serrilhado aplicado no filme atual. 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.

### Exemplo

```
_highquality = 1;
```

### Consulte também

\_quality, toggleHighQuality

## if

### Disponibilidade

Flash Player 4.

### Uso

```
if(condição) {  
    comando(s);  
}
```

### Parâmetros

*condição* Uma expressão que seja avaliada como true ou false.

*comando(s)* As instruções a serem executadas se ou quando a condição for avaliada como true.



## Retorna

Nada.

## Descrição

Ação; avalia uma condição para determinar a próxima ação em um filme. Se a condição for `true`, o Flash executará os comandos após a condição dentro das chaves (`{}`). Se a condição for `false`, o Flash ignorará os comandos contidos nas chaves e executará os comandos posteriores a elas. Use a ação `if` para criar lógica ramificada em seus scripts.

## Exemplo

No exemplo a seguir, a condição entre parênteses avalia a variável `name` para verificar se ela tem o valor literal “Erica”. Caso tenha, a ação `play` entre as chaves é executada.

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

## Exemplo

O exemplo a seguir utiliza uma ação `if` para avaliar quando o usuário libera um objeto arrastável no filme. Se o objeto tiver sido liberado menos de 300 milissegundos depois de arrastado, a condição será avaliada como `true` e os comandos entre as chaves serão executados. Esses comandos definem variáveis para armazenar informações como a nova posição do objeto, a força e a velocidade com que ele foi lançado. A variável `timePressed` também é redefinida. Se o objeto tiver sido liberado mais de 300 milissegundos depois de arrastado, a condição será avaliada como `false` e nenhum comando será executado.

```
if (getTimer()<timePressed+300) {  
    // se a condição for true,  
    // o objeto foi lançado.  
    // qual a nova posição deste objeto?  
    xNewLoc = this._x;  
    yNewLoc = this._y;  
    //qual a força empregada para seu lançamento?  
    xTravel = xNewLoc-xLoc;  
    yTravel = yNewLoc-yLoc;  
    // a definição da velocidade do objeto depende  
    // da distância percorrida  
    xInc = xTravel/2;  
    yInc = yTravel/2;  
    timePressed = 0;  
}
```

## Consulte também

`else`

## ifFrameLoaded

### Disponibilidade

Flash Player 3. A ação `ifFrameLoaded` foi reprovada no Flash 5; o uso da ação `MovieClip._framesloaded` é encorajado.

### Uso

```
ifFrameLoaded(cena, quadro) {  
    comando;  
}  
  
ifFrameLoaded(quadro) {  
    comando(s);  
}
```

### Parâmetros

*cena* A cena que deve ser carregada.

*quadro* O número ou rótulo do quadro que deve ser carregado antes da execução do próximo comando.

*comando(s)* As instruções a serem executadas se uma determinada cena (ou cena e quadro específicos) for carregada.

### Retorna

Nada.

### 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 o uso de `_framesloaded` e `ifFrameLoaded` é que `_framesloaded` permite que você adicione seus próprios comandos `if` ou `else`.

### Consulte também

`MovieClip._framesloaded`

## #include

### Disponibilidade

N/A

### Uso

```
#include "filename.as"
```

### Parâmetros

*filename.as* O nome do arquivo para o script a ser adicionado ao painel Ações; `.as` é a extensão de arquivo recomendada.

### Retorna

Nada.

### Descrição

Ação; inclui o conteúdo do arquivo especificado no parâmetro 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.

## #initclip

### Disponibilidade

Flash Player 6.

### Uso

```
#initclip order
```

### Parâmetros

*order* Um número inteiro que especifica a ordem de execução de blocos de código `#initclip`. Este é um parâmetro opcional.

### Descrição

Ação; indica o início de um bloco de ações de inicialização de componentes. Quando vários cliques são inicializados ao mesmo tempo, você pode usar o parâmetro *order* (ordem) para especificar a inicialização que ocorrerá primeiro. As ações de inicialização de componentes são executadas quando um símbolo de clipe de filme é definido. Se o clipe de filme for um símbolo exportado, as ações de inicialização de componentes serão executadas antes das ações no Quadro 1 do arquivo SWF. Caso contrário, elas serão executadas imediatamente antes das ações do quadro que contém a primeira instância do símbolo de clipe de filme associado.

As ações de inicialização de componentes são executadas apenas uma vez durante a reprodução de um filme e você deve usá-las para inicializações de uma etapa, como a definição e o registro de classes.

### Exemplo

O exemplo de código a seguir foi atribuído ao primeiro quadro de um filme que é um componente de caixa de seleção. As ações `#initclip` e `#endinitclip` indicam o bloco de condições que delimitam como ações de inicialização de componente. Os comandos delimitados registram a classe e os métodos de armazenamento em um objeto de protótipo.

```
#initclip
if (typeof(CheckBox) == "indefinido") {
    // Definir o construtor para (e, portanto, definir) a classe CheckBox
    function CheckBox() {
        //Configurar nossas vinculações de dados
        this.watch('value', function(id, oldval, newval) { ... });
        this.watch('label', function(id, oldval, newval) { ... });
    }
    // Definir que a cadeia de protótipos CheckBox herda de MovieClip
    CheckBox.prototype = new MovieClip();
    // Registrar CheckBox como a classe do símbolo de "Caixa de seleção"
    Object.registerClass("Caixa de seleção", CheckBox);
    // Configurar alguns métodos
    CheckBox.prototype.enable = function() { ... };
    CheckBox.prototype.show = function() { ... };
    CheckBox.prototype.hide = function() { ... };
    // Configurar uma função conveniente para criar
    // caixas de seleção
    CheckBox.create = function(parentMovieClip, instanceName, depth) {
        parentMovieClip.attachMovie("CheckBox", instanceName, depth);
    };
}
#endinitclip
```

**Observação:** Se você copiar e colar esse código no painel Ações, será gerado um erro quando o script for compilado por causa das funções indefinidas (`{ ... }`)

### Consulte também

```
#endinitclip
```

# instanceof

## Disponibilidade

Flash Player 6.

## Uso

*object* instanceof *class*

## Parâmetros

**object** Um objeto `ActionScript`.

**class** Uma referência a uma função construtora `ActionScript`, como `String` ou `Date`.

## Retorna

Se o *objeto* for uma instância de *classe*, `instanceof` retornará `true`; caso contrário, `instanceof` retornará `false`.

## Descrição

Operador; determina se um objeto pertence a uma determinada classe. Testa se o *objeto* é uma instância de *classe*.

Um objeto `ActionScript` é considerado uma instância de uma classe quando o objeto de protótipo da função construtora está na cadeia de protótipos do objeto `ActionScript`.

O operador `instanceof` não converte tipos primitivos em objetos envoltórios. Por exemplo, o código a seguir retorna `true`:

```
new String("Olá") instanceof String
```

Já o código a seguir retorna `false`:

```
"Olá" instanceof String
```

## Exemplo

Para ilustrar o algoritmo do operador `instanceof`, o exemplo a seguir mostra a aparência possível desse operador `instanceof` caso seja codificado como uma função do `ActionScript`.

```
function instanceof (theObject, theClass){
    while ((theObject = theObject.__proto__) != null) {
        if (theObject == theClass.prototype) {
            return true;
        }
    }
    return false;
}
```

## Consulte também

`typeof`

## int

### Disponibilidade

Flash Player 4. Esta função foi reprovada no Flash 5 e substituída pelo método `Math.round`.

### Uso

```
int(valor)
```

### Parâmetros

*valor* Um número a ser arredondado para um inteiro.

### Retorna

Nada.

### Descrição

Função; converte um número decimal no valor inteiro mais próximo.

### Consulte também

`Math.floor`

## isFinite

### Disponibilidade

Flash Player 5.

### Uso

```
isFinite(expressão)
```

### Parâmetros

*expressão* Um valor booleano, variável ou outra expressão a ser avaliada.

### Retorna

Nada.

### Descrição

Função de alto nível; avalia a *expressão* 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.

### Exemplo

A seguir são mostrados exemplos dos valores retornados por `isFinite`:

```
isFinite(56)  
// retorna true
```

```
isFinite(Number.POSITIVE_INFINITY)  
// retorna false
```

# isNaN

## Disponibilidade

Flash Player 5.

## Uso

`isNaN(expressão)`

## Parâmetros

*expressão* Um valor booleano, variável ou outra expressão a ser avaliada.

## Retorna

Nada.

## Descrição

Função de alto nível; avalia o parâmetro e retorna `true` se o valor não for um número (NaN), indicando a presença de erros matemáticos.

## Exemplo

O código a seguir demonstra valores de retorno para a função `isNaN`.

```
isNaN("Árvore")
// 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. Para obter uma lista completa de valores do código de tecla, consulte o apêndice “Teclas do teclado e valores de códigos de teclas” em *Usando o Flash*.

## Exemplo

O script a seguir usa o objeto `Key` para identificar teclas em qualquer teclado, de forma que o usuário possa controlar um clipe de filme.

```
onClipEvent (enterFrame) {
    if(Key.isDown(Key.RIGHT)) {
        this._x=_x+10;
    } else if (Key.isDown(Key.DOWN)) {
        this._y=_y+10;
    }
}
```

## Resumo dos métodos do objeto Key

Método	Descrição
<code>Key.addListener</code>	Registra um objeto para receber a notificação quando os métodos <code>onKeyDown</code> e <code>onKeyUp</code> são chamados.
<code>Key.getAscii</code>	Retorna o valor ASCII da última tecla pressionada.
<code>Key.getCode</code>	Retorna o código de tecla virtual da última tecla pressionada.
<code>Key.isDown</code>	Retorna <code>true</code> se a tecla especificada no parâmetro for pressionada.
<code>Key.isToggled</code>	Retorna <code>true</code> se a tecla Num Lock ou Caps Lock estiver ativada.
<code>Key.removeListener</code>	Remove um objeto que foi registrado anteriormente com <code>addListener</code> .

## Resumo das propriedades do objeto Key

Todas as propriedades do objeto Key são constantes.

Propriedade	Descrição
<code>Key.BACKSPACE</code>	Constante associada ao valor do código da tecla Backspace (8).
<code>Key.CAPSLock</code>	Constante associada ao valor do código da tecla Caps Lock (20).
<code>Key.CONTROL</code>	Constante associada ao valor do código da tecla Control (17).
<code>Key.DELETEKEY</code>	Constante associada ao valor do código da tecla Delete (46).
<code>Key.DOWN</code>	Constante associada ao valor do código da tecla Seta para baixo (40).
<code>Key.END</code>	Constante associada ao valor do código da tecla End (35).
<code>Key.ENTER</code>	Constante associada ao valor do código da tecla Enter (13).
<code>Key.ESCAPE</code>	Constante associada ao valor do código da tecla Escape (27).
<code>Key.HOME</code>	Constante associada ao valor do código da tecla Home (36).
<code>Key.INSERT</code>	Constante associada ao valor do código da tecla Insert (45).
<code>Key.LEFT</code>	Constante associada ao valor do código da tecla Seta para esquerda (37).
<code>Key.PGDN</code>	Constante associada ao valor do código da tecla Page Down (34).
<code>Key.PGUP</code>	Constante associada ao valor do código da tecla Page Up (33).
<code>Key.RIGHT</code>	Constante associada ao valor do código da tecla Seta para direita (39).
<code>Key.SHIFT</code>	Constante associada ao valor do código da tecla Shift (16).
<code>Key.SPACE</code>	Constante associada ao valor do código de tecla da Barra de espaços (32).
<code>Key.TAB</code>	Constante associada ao valor do código da tecla Tab (9).
<code>Key.UP</code>	Constante associada ao valor do código da tecla Seta para cima (38).

## Resumo dos ouvintes do objeto Key

Método	Descrição
<code>Key.onKeyDown</code>	Notificado quando uma tecla é pressionada.
<code>Key.onKeyUp</code>	Notificado quando uma tecla é liberada

## Key.addListener

### Disponibilidade

Flash Player 6.

### Uso

`Key.addListener (newListener)`

### Parâmetros

`newListener` Um objeto com os métodos `onKeyDown` e `onKeyUp`.

### Retorna

Nada.

### Descrição

Método; registra um objeto para receber a notificação `onKeyDown` e `onKeyUp`. Quando uma tecla é pressionada ou liberada, independentemente do foco de entrada, todos os objetos ouvintes registrados com `addListener` têm o método `onKeyDown` ou `onKeyUp` chamado. Vários objetos podem ouvir notificações de teclado. Se o ouvinte `newListener` já estiver registrado, nenhuma alteração ocorrerá.

### Exemplo

Este exemplo cria um novo objeto ouvinte e define uma função para `onKeyDown` e `onKeyUp`. A última linha usa o método `addListener` para registrar o ouvinte com o objeto `Key` para que possa receber notificações de eventos de tecla pressionada e tecla liberada.

```
myListener = new Object();
myListener.onKeyDown = function () {
    trace ("Você pressionou uma tecla.");
}
myListener.onKeyUp = function () {
    trace ("Você liberou uma tecla.");
}
Key.addListener(myListener);
```

## Key.BACKSPACE

### Disponibilidade

Flash Player 5.

### Uso

`Key.BACKSPACE`

### Descrição

Propriedade; constante associada ao valor do código da tecla Backspace (8).

## Key.CAPSLOCK

### Disponibilidade

Flash Player 5.

### Uso

`Key.CAPSLOCK`

### Descrição

Propriedade; constante associada ao valor do código da tecla Caps Lock (20).



## Key.CONTROL

### Disponibilidade

Flash Player 5.

### Uso

Key.CONTROL

### Descrição

Propriedade; constante associada ao valor do código da tecla Control (17).

## Key.DELETEKEY

### Disponibilidade

Flash Player 5.

### Uso

Key.DELETEKEY

### Descrição

Propriedade; constante associada ao valor do código da tecla Delete (46).

## Key.DOWN

### Disponibilidade

Flash Player 5.

### Uso

Key.DOWN

### Descrição

Propriedade; constante associada ao valor do código da tecla Seta para baixo (40).

## Key.END

### Disponibilidade

Flash Player 5.

### Uso

Key.END

### Descrição

Propriedade; constante associada com o valor do código de chave da tecla End (35).

## Key.ENTER

### Disponibilidade

Flash Player 5.

### Uso

Key.ENTER

### Descrição

Propriedade; constante associada ao valor do código da tecla Enter (13).

## Key.ESCAPE

### Disponibilidade

Flash Player 5.

### Uso

`Key.ESCAPE`

### Descrição

Propriedade; constante associada ao valor do código da tecla Escape (27).

## Key.getAscii

### Disponibilidade

Flash Player 5.

### Uso

`Key.getAscii();`

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Método; retorna o código ASCII da última tecla pressionada ou liberada. Os valores ASCII retornados correspondem aos valores do teclado inglês. Por exemplo, se você pressionar Shift+2, `Key.getAscii` retornará @ em um teclado japonês, da mesma forma que ocorre com um teclado inglês.

## Key.getCode

### Disponibilidade

Flash Player 5.

### Uso

`Key.getCode();`

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Método; retorna o valor do código de tecla da última tecla pressionada. Para associar o valor do código de tecla retornado à tecla virtual em um teclado padrão, utilize as informações do Apêndice C, “Teclas do teclado e valores de códigos de teclas” de *Usando o Flash*.

## Key.HOME

### Disponibilidade

Flash Player 5.

### Uso

`Key.HOME`

### Descrição

Propriedade; constante associada ao valor do código da tecla Home (36).

## Key.INSERT

### Disponibilidade

Flash Player 5.

### Uso

`Key.INSERT`

### Descrição

Propriedade; constante associada ao valor do código da tecla Insert (45).

## Key.isDown

### Disponibilidade

Flash Player 5.

### Uso

`Key.isDown(keycode);`

### Parâmetros

*keycode* O valor do código de tecla atribuído a uma determinada tecla ou à propriedade do objeto `Key` associada a uma tecla específica. Para obter uma lista de todos os códigos associados às teclas de um teclado padrão, consulte o Apêndice C, “Teclas do teclado e valores de códigos de teclas” de *Usando o Flash*.

### Retorna

Nada.

### 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 são idênticos.

## Key.isToggled

### Disponibilidade

Flash Player 5.

### Uso

`Key.isToggled(keycode)`

### Parâmetros

*keycode* O código da tecla Caps Lock (20) ou Num Lock (144).

**Retorna**

Nada.

**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.

## Key.LEFT

**Disponibilidade**

Flash Player 5.

**Uso**

`Key.LEFT`

**Descrição**

Propriedade; constante associada ao valor do código de tecla para a tecla Seta para esquerda (37).

## Key.onKeyDown

**Disponibilidade**

Flash Player 6.

**Uso**

`someListener.onKeyDown`

**Descrição**

Ouvinte; notificado quando uma tecla é pressionada. É necessário criar um objeto ouvinte para usar `onKeyDown`. Em seguida, é possível definir uma função para `onKeyDown` e usar o método `addListener` para registrar o ouvinte com o objeto `Key`, como mostrado a seguir:

```
someListener = new Object();
someListener.onKeyDown = function () { ... };
Key.addListener(someListener);
```

Os ouvintes permitem a cooperação de partes diferentes de código. Isso ocorre porque vários ouvintes podem receber notificações sobre um único evento.

**Consulte também**

`Key.addListener`

## Key.onKeyUp

**Disponibilidade**

Flash Player 6.

**Uso**

`someListener.onKeyUp`

### Descrição

Ouvinte; notificado quando uma tecla é liberada. É necessário criar um objeto ouvinte para usar `onKeyUp`. Em seguida, é possível definir uma função para `onKeyUp` e usar o método `addListener` para registrar o ouvinte no objeto `Key`, como mostrado a seguir:

```
someListener = new Object();
someListener.onKeyUp = function () { ... };
Key.addListener(someListener);
```

Os ouvintes permitem a cooperação de partes diferentes de código. Isso ocorre porque vários ouvintes podem receber notificações sobre um único evento.

### Consulte também

`Key.addListener`

## Key.PGDN

### Disponibilidade

Flash Player 5.

### Uso

`Key.PGDN`

### Descrição

Propriedade; constante associada ao valor do código de tecla para a tecla Page Down (34).

## Key.PGUP

### Disponibilidade

Flash Player 5.

### Uso

`Key.PGUP`

### Descrição

Propriedade; constante associada ao valor do código da tecla Page Up (33).

## Key.removeListener

### Disponibilidade

Flash Player 6.

### Uso

```
Key.removeListener (ouvinte)
```

### Parâmetros

*ouvinte* Um objeto.

### Retorna

Se o *ouvinte* tiver sido removido com êxito, o método retornará `true`. Caso o *ouvinte* não tenha sido removido com êxito, por exemplo, se *ele* não estava na lista de ouvintes do objeto `Key`, o método retornará `false`.

### Descrição

Método; remove um objeto anteriormente registrado com o método `addListener`.

## Key.RIGHT

### Disponibilidade

Flash Player 5.

### Uso

Key.RIGHT

### Descri

ção

Propriedade; constante associada ao valor do código da tecla Seta para direita (39).

## Key.SHIFT

### Disponibilidade

Flash Player 5.

### Uso

Key.SHIFT

### Descrição

Propriedade; constante associada ao valor do código da tecla Shift (16).

## Key.SPACE

### Disponibilidade

Flash Player 5.

### Uso

Key.SPACE

### Descrição

Propriedade; constante associada ao valor do código de tecla da Barra de espaços (32).

## Key.TAB

### Disponibilidade

Flash Player 5.

### Uso

Key.TAB

### Descrição

Propriedade; constante associada ao valor do código da tecla Tab (9).

## Key.UP

### Disponibilidade

Flash Player 5.

### Uso

Key.UP

### Descrição

Propriedade; constante associada ao valor do código da tecla Seta para cima (38).

## le (menor que ou igual a – específico da sequência de caracteres)

### Disponibilidade

Flash Player 4. Este operador foi reprovado no Flash 5 e substituído pelo operador `<=` (menor ou igual a) .

### Uso

*expression1* le *expression2*

### Parâmetros

*expression1*, *expression2* Números, seqüências de caracteres ou variáveis.

### Retorna

Nada.

### Descrição

Operador (comparação); compara *expression1* com *expression2* e retorna um valor `true` se *expression1* for menor ou igual a *expression2*; de outra forma, retorna um valor `false`.

### Consulte também

`<=` (menor ou igual a)

## length

### Disponibilidade

Flash Player 4. Esta função, assim como todas as funções de seqüências de caracteres, foi reprovada no Flash 5. Recomenda-se utilizar os métodos e a propriedade `length` do objeto `String` para a realização das mesmas operações.

### Uso

`length(expressão)`

`length(variável)`

### Parâmetros

*expressão* Uma seqüência de caracteres.

*variável* O nome de uma variável.

### Retorna

Nada.

### Descrição

Função de seqüência de caracteres; retorna o comprimento da seqüência de caracteres ou do nome da variável especificada.

### Exemplo

O exemplo a seguir retorna o valor da seqüência de caracteres "Olá".

```
length("Olá");
```

O resultado é 4.

### Consulte também

`" "` (delimitador de seqüência de caracteres), `String.length`

## **\_level**

### **Disponibilidade**

Flash Player 4.

### **Uso**

`_levelN`

### **Descrição**

Propriedade; uma referência ao filme raiz Linha de tempo de `_levelN`. É necessário empregar a ação `loadMovieNum` para carregar filmes no Flash Player antes de usar a propriedade `_level` para especificá-los. Também é possível utilizar `_levelN` para especificar um filme carregado no nível atribuído por *N*.

O filme inicial carregado em uma instância do Flash Player é carregado automaticamente em `_level0`. O filme em `_level0` define a taxa de quadros, a cor de fundo e o tamanho do quadro para todos os outros filmes carregados. Em seguida, os filmes são empilhados em níveis que recebem números mais altos do que o do filme em `_level0`.

É necessário atribuir um nível para cada filme carregado no Flash Player com a ação `loadMovieNum`. É possível atribuir níveis em qualquer ordem. Se você atribuir um nível que já contenha um arquivo SWF (inclusive `_level0`), o filme nesse nível será descarregado e substituído pelo novo.

### **Exemplo**

O exemplo a seguir interrompe a reprodução na Linha de tempo principal do filme em `_level9`.  
`_level9.stop();`

O exemplo a seguir envia a reprodução na Linha de tempo principal do filme em `_level4` para o quadro 5. É necessário que o filme em `_level4` tenha sido carregado anteriormente com uma ação `loadMovieNum`.

```
_level4.gotoAndStop(5);
```

### **Consulte também**

`loadMovie`, `MovieClip.swapDepths`

## **loadMovie**

### **Disponibilidade**

Flash Player 3.

### **Uso**

```
loadMovie("url",nível/destino[, variáveis])
```

### **Parâmetros**

*url* O URL absoluto ou relativo do arquivo SWF ou JPEG que deve ser carregado. Um caminho relativo deve ser relativo ao arquivo SWF no nível 0. O URL deve estar no mesmo subdomínio que o URL onde o filme reside no momento. Para uso no Flash Player ou para verificações no modo de teste do aplicativo de criação 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.



*destino* Um caminho para um clipe de filme de destino. O clipe de filme de destino será substituído pelo filme ou pela imagem carregada. É necessário especificar um clipe de filme de *destino* ou um *nível* de filme de destino; não é possível especificar as duas opções.

*nível* Um inteiro que especifica o nível no qual o filme foi carregado no Flash Player. Quando você carrega um filme ou imagem em um determinado nível, a ação `loadMovie` do painel Ações no modo normal alterna para `loadMovieNum`; no modo Especialista, é necessário especificar `loadMovieNum` ou escolher essa opção na caixa de ferramentas Ações.

*variáveis* Um parâmetro opcional que especifica um método HTTP para o envio de variáveis. O parâmetro deve ser a sequência de caracteres `GET` ou `POST`. Se não houver nenhuma variável a ser enviada, omita esse parâmetro. O método `GET` anexa as variáveis ao final do URL e é usado para pequenos números de variáveis. O método `POST` envia as variáveis em um cabeçalho HTTP separado e é usado para sequências de caracteres maiores de variáveis.

## **Retorna**

Nada.

## **Descrição**

Ação; carrega um arquivo SWF ou JPEG no Flash Player durante a reprodução do filme original. A ação `loadMovie` permite que você exiba vários filmes de uma vez ou alterne entre os filmes sem carregar outro documento HTML. Sem a ação `loadMovie`, o Flash Player exibe um único filme (arquivo SWF) e é encerrado em seguida.

Quando você usa a ação `loadMovie`, é necessário especificar um nível no Flash Player ou um clipe de filme de destino no qual o filme será carregado. Se você especificar um nível, a ação será alterada para `loadMovieNum`. Se um filme for carregado em um clipe de filme de destino, você poderá usar o caminho de destino desse clipe para especificar o filme carregado.

Um filme ou imagem carregada em um destino herda as propriedades de posição, rotação e dimensionamento do clipe de filme de destino. O canto superior esquerdo da imagem ou filme carregado é alinhado ao ponto de registro do clipe de filme de destino. Como alternativa, se o destino for a Linha de tempo `_root`, o canto superior esquerdo da imagem ou filme é alinhado ao canto superior esquerdo do Palco.

Use a ação `unloadMovie` para remover os filmes carregados com a ação `loadMovie`.

## **Exemplo**

O comando `loadMovie` a seguir 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);  
}
```

O exemplo a seguir carrega uma imagem JPEG do mesmo diretório que o arquivo SWF que chama a ação `loadMovie`:

```
loadMovie("image45.jpeg", "nosso_clipe_de_filme");
```

## **Consulte também**

`loadMovieNum`, `unloadMovie`, `unloadMovieNum`, `_level`

# loadMovieNum

## Disponibilidade

Flash Player 4. Os arquivos do Flash 4 abertos no Flash 5 são convertidos para que utilizem a sintaxe correta.

## Uso

```
loadMovieNum("url",nível[, variáveis])
```

## Parâmetros

*url* O URL absoluto ou relativo do arquivo SWF ou JPEG a ser carregado. Um caminho relativo deve ser relativo ao arquivo SWF no nível 0. O URL deve estar no mesmo subdomínio que o URL onde o filme reside no momento. Para uso no Flash Player independente ou para verificações no modo de teste de filme no aplicativo de criação 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.

*nível* Um inteiro que especifica o nível no qual o filme foi carregado no Flash Player.

*variáveis* Um parâmetro opcional que especifica um método HTTP para o envio de variáveis. O parâmetro deve ser a seqüência de caracteres GET ou POST. Se não houver nenhuma variável a ser enviada, omite esse parâmetro. O método GET anexa as variáveis ao final do URL e é usado para pequenos números de variáveis. O método POST envia as variáveis em um cabeçalho HTTP separado e é usado para seqüências de caracteres maiores de variáveis.

## Retorna

Nada.

## Descrição

Ação; carrega um arquivo SWF ou JPEG em um nível do Flash Player durante a reprodução do filme carregado originalmente. Quando você carrega um filme em um nível em vez de fazê-lo em um destino, a ação `loadMovie` do painel Ações no modo normal alterna para `loadMovieNum`; no modo Especialista, é necessário especificar `loadMovieNum` ou escolher essa opção na caixa de ferramentas Ações. Normalmente, o Flash Player exibe um único filme (arquivo SWF) e em seguida é encerrado. A ação `loadMovieNum` permite que você exiba vários filmes de uma vez ou alterne entre os filmes sem carregar outro documento HTML.

O Flash Player apresenta uma ordem de empilhamento de níveis iniciada em 0. Esses níveis são como camadas de acetato: transparentes, a não ser pelos objetos em cada nível. Quando você usa a ação `loadMovieNum`, é necessário especificar um nível do Flash Player no qual o filme será carregado. Quando um filme é carregado em um determinado nível, é possível usar a sintaxe `_levelN`, na qual *N* é o número do nível para especificar o filme.

Quando você carrega um filme, pode especificar qualquer número de nível, além de carregar filmes em um nível que já tenha um arquivo SWF carregado. Se você o fizer, o novo filme irá substituir o arquivo SWF existente. Se você carregar um filme no nível 0, todos os níveis do Flash Player serão descarregados. Além disso, o nível 0 será substituído pelo novo arquivo. O filme no nível 0 define a taxa de quadros, a cor de fundo e o tamanho do quadro para todos os outros filmes carregados.

A ação `loadMovieNum` também permite carregar arquivos JPEG em um filme durante sua reprodução. No caso de imagens e arquivos SWF, o canto superior esquerdo da imagem é alinhado com o canto superior esquerdo do Palco durante o carregamento do arquivo. Além disso, nos dois casos o arquivo carregado herda a rotação e o dimensionamento, sendo que o conteúdo original é substituído.

Use a ação `unloadMovieNum` para remover filmes ou imagens carregadas com a ação `loadMovieNum`.

### Exemplo

Este exemplo carrega a imagem JPEG “image45.jpg” no nível 2 do Flash Player.

```
loadMovieNum("http://www.blag.com/image45.jpg", 2); //
```

### Consulte também

`loadMovie`, `unloadMovie`, `unloadMovieNum`, `_level`

## loadVariables

### Disponibilidade

Flash Player 4.

### Uso

```
loadVariables ("url" ,nível/"destino" [, variáveis])
```

### Parâmetros

*url* Um URL absoluto ou relativo no qual as variáveis estão localizadas. Se você acessar o filme com um navegador da Web, o host do URL deverá estar no mesmo subdomínio do filme.

*nível* Um inteiro que especifica o nível que receberá as variáveis no Flash Player. Quando você carrega variáveis em um nível, a ação do painel Ações no modo normal torna-se `loadVariablesNum`; no modo Especialista é necessário especificar `loadVariablesNum` ou escolher essa opção na caixa de ferramentas Ações.

*destino* O caminho de destino para um clipe de filme que recebe as variáveis carregadas. É necessário especificar um clipe de filme de *destino* ou um *nível* (nível) no Flash Player; não é possível especificar as duas opções.

*variáveis* Um parâmetro opcional que especifica um método HTTP para o envio de variáveis. O parâmetro deve ser a sequência de caracteres `GET` ou `POST`. Se não houver nenhuma variável a ser enviada, omita esse parâmetro. O método `GET` anexa as variáveis ao final do URL e é usado para pequenos números de variáveis. O método `POST` envia as variáveis em um cabeçalho HTTP separado e é usado para seqüências de caracteres maiores de variáveis.

### Retorna

Nada.

### Descrição

Ação; lê dados de um arquivo externo, como um arquivo de texto ou texto gerado por um script CGI, Active Server Pages (ASP) ou PHP, ou ainda um script Perl e define os valores das variáveis em um nível do Flash Player ou um clipe de filme de destino. Essa ação também pode ser usada para atualizar as variáveis no filme ativo com novos valores.

O texto no URL especificado deve ter o formato MIME padrão *aplicativo/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
```

O primeiro filme a ser aberto em uma instância do Flash Player é carregado no nível inferior (identificado no código como `_level0`). Quando você usa a ação `loadMovie` ou `loadMovieNum` para carregar filmes subsequentes no Flash Player, é necessário atribuir um número de nível no Flash Player ou um clipe de filme de destino no qual cada filme será carregado. Quando você usa a ação `loadVariables`, é necessário especificar um nível do Flash Player ou um clipe de filme de destino no qual as variáveis serão carregadas.

### Exemplo

Este exemplo carrega informações de um arquivo de texto em campos de texto no clipe de filme `varTarget` da Linha de tempo principal. Os nomes das variáveis dos campos de texto devem corresponder aos nomes das variáveis no arquivo `data.txt`.

```
on(release) {  
    loadVariables("data.txt", "_root.varTarget");  
}
```

### Consulte também

`loadVariablesNum`, `loadMovie`, `loadMovieNum`, `getURL`, `MovieClip.loadMovie`, `MovieClip.loadVariables`

## loadVariablesNum

### Disponibilidade

Flash Player 4. Os arquivos do Flash 4 abertos no Flash 5 são convertidos para que utilizem a sintaxe correta.

### Uso

```
loadVariables ("url" ,nível [, variáveis])
```

### Parâmetros

**url** Um URL absoluto ou relativo no qual as variáveis estão localizadas. Se você acessar o filme com um navegador da Web, o host do URL deverá estar no mesmo subdomínio do filme.

**nível** Um inteiro que especifica o nível que receberá as variáveis no Flash Player.

**variáveis** Um parâmetro opcional que especifica um método HTTP para o envio de variáveis. O parâmetro deve ser a sequência de caracteres `GET` ou `POST`. Se não houver nenhuma variável a ser enviada, omita esse parâmetro. O método `GET` anexa as variáveis ao final do URL e é usado para pequenos números de variáveis. O método `POST` envia as variáveis em um cabeçalho HTTP separado e é usado para seqüências de caracteres maiores de variáveis.

### Retorna

Nada.

## 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 PHP e define os valores das variáveis em um nível do Flash Player. Essa ação também pode ser usada para atualizar as variáveis no filme ativo com novos valores. Quando você carrega variáveis em um nível, a ação do painel Ações no modo normal torna-se `loadVariablesNum`; no modo Especialista é necessário especificar `loadVariablesNum` ou escolher essa opção na caixa de ferramentas Ações.

O texto no URL especificado deve ter o formato MIME padrão *aplicativo/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
```

O primeiro filme a ser aberto em uma instância do Flash Player é carregado no nível inferior (identificado no código como `_level0`). Quando você usa a ação `loadMovie` ou `loadMovieNum` para carregar filmes subsequentes no Flash Player, é necessário atribuir um número de nível no Flash Player ou um clipe de filme de destino no qual cada filme será carregado. Quando você usa a ação `loadVariablesNum`, é necessário especificar um nível do Flash Player no qual as variáveis serão carregadas.

## Exemplo

Este exemplo carrega informações de um arquivo de texto em campos de texto na Linha de tempo principal do filme no nível 0 do Flash Player. Os nomes das variáveis dos campos de texto devem corresponder aos nomes das variáveis no arquivo `data.txt`.

```
on(release) {  
    loadVariablesNum("data.txt", 0);  
}
```

## Consulte também

`getUrl`, `loadMovie`, `loadMovieNum`, `loadVariables`, `MovieClip.loadMovie`, `MovieClip.loadVariables`

## LoadVars (objeto)

O objeto `LoadVars` é uma alternativa à ação `loadVariables` para a transferência de variáveis entre um filme do Flash e um servidor.

Você pode usar o objeto `LoadVars` para obter informações sobre erros, indicações de progresso e fluxos de dados durante seu download. O objeto `LoadVars` funciona de maneira muito semelhante ao objeto XML; ele utiliza os métodos `load`, `send` e `sendAndLoad` para estabelecer comunicações com um servidor. A diferença principal entre os objetos `LoadVars` e XML é que o primeiro transfere o nome e pares de valores ActionScript, em vez de uma árvore XML DOM armazenada no objeto XML.

O objeto `LoadVars` segue as mesmas restrições de segurança do objeto XML.

É necessário usar o construtor `new LoadVars()` para criar uma instância do objeto `LoadVars` antes de chamar seus métodos.

O objeto `LoadVars` é suportado pelo Flash Player 6 e versões posteriores.

## Resumo dos métodos do objeto LoadVars

Método	Descrição
<code>LoadVars.load</code>	Faz download de variáveis de um URL especificado.
<code>LoadVars.getBytesTotal</code>	Retorna o número de bytes carregados de um método <code>load</code> ou <code>sendAndLoad</code> .
<code>LoadVars.getBytesTotal</code>	Retorna o número total de bytes que serão descarregados por um método <code>load</code> ou <code>sendAndLoad</code> .
<code>LoadVars.send</code>	Envia variáveis de um objeto <code>LoadVars</code> para um URL.
<code>LoadVars.sendAndLoad</code>	Envia variáveis de um objeto <code>LoadVars</code> para um URL e faz o download da resposta do servidor para um objeto de destino.
<code>LoadVars.toString</code>	Retorna uma sequência de caracteres codificados de URL que contém todas as variáveis enumeráveis do objeto <code>LoadVars</code> .

## Resumo das propriedades do objeto LoadVars

Todas as propriedades do objeto `Key` são constantes.

Propriedade	Descrição
<code>LoadVars.contentType</code>	Indica um tipo de dados MIME.
<code>LoadVars.load</code>	Um valor booleano que indica se foi realizada uma operação <code>load</code> ou <code>sendAndLoad</code> .

## Resumo dos eventos do objeto LoadVars

Método	Descrição
<code>LoadVars.onLoad</code>	Chamado quando uma operação <code>load</code> or <code>sendAndLoad</code> é concluída.

## Construtor do objeto LoadVars

### Disponibilidade

Flash Player 6.

### Uso

```
new LoadVars()
```

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Construtor; cria uma instância do objeto `LoadVars`. Em seguida, você pode usar os métodos desse objeto `LoadVars` para enviar e carregar dados.

### Exemplo

O exemplo a seguir cria uma instância do objeto `LoadVars` chamada `myLoadVars`:

```
myLoadVars = new LoadVars();
```

## LoadVars.contentType

### Disponibilidade

Flash Player 6.

### Uso

```
myLoadVars.contentType
```

### Descrição

Propriedade; o tipo MIME enviado ao servidor quando você chama o método `LoadVars.send` ou `LoadVars.sendAndLoad`. O padrão é `aplicativo/x-www-codificado` na forma de url.

### Consulte também

`LoadVars.send`, `LoadVars.sendAndLoad`

## LoadVars.getBytesLoaded

### Disponibilidade

Flash Player 6.

### Uso

```
myLoadVars.getBytesLoaded()
```

### Parâmetros

Nenhum.

### Retorna

Um inteiro.

### Descrição

Método; retorna o número de bytes descarregados por um método `load` ou `sendAndLoad`. O método `getBytesLoaded` retorna `undefined` se não houver nenhuma operação `load` em andamento ou se `load` ainda não tiver sido iniciada

## LoadVars.getBytesTotal

### Disponibilidade

Flash Player 6.

### Uso

```
myLoadVars.getBytesTotal()
```

### Parâmetros

Nenhum.

### Retorna

Um inteiro.

### Descrição

Método; retorna o número total de bytes descarregados por uma operação `load` ou `sendAndLoad`. O método `getBytesTotal` retorna `undefined` se não houver nenhuma operação `load` em andamento ou se `load` ainda não tiver sido iniciada. O método `getBytesTotal` também retorna `undefined` se não for possível determinar o número total de bytes; por exemplo, quando o download foi iniciado, mas o servidor não transmitiu um item de tamanho do conteúdo HTTP.

## LoadVars.load

### Disponibilidade

Flash Player 6.

### Uso

```
myLoadVars.load(url)
```

### Parâmetros

*url* O URL de onde será feito o download das variáveis.

### Retorna

Uma sequência de caracteres.

### Descrição

Método; faz download de variáveis do URL especificado, analisa os dados da variável e coloca as variáveis resultantes em `loadVarsObject`. As propriedades em `loadVarsObject` com o mesmo nome de variáveis descarregadas são substituídas. As propriedades em `loadVarsObject` com nomes diferentes das variáveis descarregadas não são excluídas. Esta é uma ação assíncrona.

Os dados descarregados devem estar codificados com o tipo de conteúdo MIME *aplicativo/x-www-codificado na forma de url*. Este é o mesmo formato utilizado por `loadVariables`.

Este método é semelhante ao método `XML.load` do objeto XML.

## LoadVars.loaded

### Disponibilidade

Flash Player 6.

### Uso

```
myLoadVars.loaded
```

### Descrição

Propriedade; indefinida por padrão. Quando uma operação `load` ou `sendAndLoad` é iniciada, a propriedade `loaded` é definida como `false`. Quando a operação `load` ou `sendAndLoad` é concluída, a propriedade `loaded` é definida como `true`. Se a operação `load` ainda não tiver sido concluída ou tiver apresentado erros, a propriedade `loaded` permanece definida como `false`.

A operação `LoadVars.loaded` é semelhante à propriedade `XML.loaded` do objeto XML.



## LoadVars.onLoad

### Disponibilidade

Flash Player 6.

### Uso

```
myLoadVars.onLoad(êxito)
```

### Parâmetros

*êxito* O parâmetro indica se a operação de carregamento foi concluída com êxito (`true`) ou não (`false`).

### Retorna

Um valor booleano.

### Descrição

Manipulador de eventos; chamado quando uma operação `load` ou `sendAndLoad` é concluída. Se a operação tiver sido bem-sucedida, *loadVarsObject* será preenchido com variáveis descarregadas pela operação `load` ou `sendAndLoad`, sendo que essas variáveis estarão disponíveis quando `onLoad` for chamado.

Este método permanece indefinido por padrão, mas é possível defini-lo através da atribuição de uma função de retorno de chamada a ele.

Este método é semelhante ao método `XML.onLoad` do objeto `XML`.

## LoadVars.send

### Disponibilidade

Flash Player 6.

### Uso

```
loadVarsObject.send(url [, destino, método] )
```

### Parâmetros

*loadVarsObject* O objeto `LoadVars` a partir do qual as variáveis devem ser carregadas.

*url* O URL no qual as variáveis devem ser carregadas.

*destino* A janela de quadro do navegador na qual as respostas serão exibidas.

*método* O método "GET" ou "POST" do protocolo HTTP.

### Retorna

Uma seqüência de caracteres.

### Descrição

Método; envia as variáveis do objeto *myLoadVars* para o URL especificado. Todas as variáveis enumeráveis do objeto *myLoadVars* são concatenadas em uma seqüência de caracteres no formato *aplicativo/x-www-codificado na forma de url* por padrão e essa seqüência é enviada para o URL que utiliza o método HTTP POST. Este é o mesmo formato usado pela ação `loadVariables`. O tipo de conteúdo MIME enviado nos cabeçalhos de solicitações HTTP é o valor de `myLoadVars.contentType` ou o padrão *aplicativo/x-www-codificado na forma de url*. O método "POST" é usado, a menos que "GET" seja especificado.

Se o parâmetro *destino* for especificado, a resposta do servidor será exibida na janela de quadro do navegador chamada destino. Se o parâmetro *destino* for omitido, a resposta do servidor será descartada.

Este método é semelhante ao método `XML.send` do objeto `XML`.

## LoadVars.sendAndLoad

### Disponibilidade

Flash Player 6.

### Uso

```
myLoadVars.sendAndLoad(url, targetObject[, método])
```

### Parâmetros

*loadVarsObject* O objeto `LoadVars` a partir do qual as variáveis devem ser carregadas.

*url* O URL no qual as variáveis devem ser carregadas.

*targetObject* O objeto `LoadVars` que recebe as variáveis descarregadas.

*método* O método "GET" ou "POST" do protocolo HTTP.

### Retorna

Uma sequência de caracteres.

### Descrição

Método; envia variáveis do objeto *myLoadVars* para o URL especificado. A resposta do servidor é descarregada e analisada como dados variáveis. As variáveis resultantes são colocadas no objeto *targetObject*.

As variáveis são enviadas da mesma forma que `LoadVars.send`. As variáveis são descarregadas em *targetObject* da mesma forma que `LoadVars.load`.

Este método é semelhante ao método `XML.sendAndLoad` do objeto `XML`.

## LoadVars.toString

### Disponibilidade

Flash Player 6.

### Uso

```
loadVarsObject.toString()
```

### Parâmetros

Nenhum.

### Retorna

Uma sequência de caracteres.

### Descrição

Método; retorna uma sequência de caracteres que contém todas as variáveis enumeráveis do objeto `LoadVars`, no formato de conteúdo MIME *aplicativo/x-www-codificado na forma de url*.

### Exemplo

```
var myVars = new LoadVars();
myVars.name = "Gary";
myVars.age = 26;
trace (myVars.toString());
// would output
// name=Gary&age=26
```

## lt (menor que – sequência de caracteres específica)

### Disponibilidade

Flash Player 4. Este operador foi reprovado no Flash 5 e substituído pelo novo operador < (menor que).

### Uso

*expression1* lt *expression2*

### Parâmetros

*expression1*, *expression2* Números, seqüências de caracteres ou variáveis

### Descrição

Operador (comparação); compara a *expression1* com a *expression2* e retorna true se *expression1* for menor do que *expression2*; caso contrário, retorna false.

### Consulte também

< (menor que)

## 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 estáticas e devem ser chamadas com a sintaxe `Math.method(parâmetro)` ou `Math.constant`. Em ActionScript, as constantes são definidas com a precisão máxima de números de ponto flutuante IEEE-754 de dupla precisão.

Vários métodos do objeto Math usam o radiano de um ângulo como parâmetro. Você pode usar a equação abaixo para calcular os valores radianos ou simplesmente passar a equação (inserindo um valor para graus) para o parâmetro 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 parâmetro para calcular o seno de um ângulo de 45 graus:

```
Math.SIN(Math.PI/180 * 45) é o mesmo que Math.SIN(.7854)
```

O objeto Math é totalmente suportado no Flash Player 5. Você pode usar métodos do objeto Math no Flash Player 4, mas eles são emulados com aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash Player 5.

## Resumo dos métodos do objeto Math

Método	Descrição
<code>Math.abs</code>	Calcula um valor absoluto.
<code>Math.acos</code>	Calcula um arco cosseno.
<code>Math.asin</code>	Calcula um arco seno.
<code>Math.atan</code>	Calcula um arco tangente.
<code>Math.atan2</code>	Calcula um ângulo do eixo x ao ponto.
<code>Math.ceil</code>	Arredonda um número para o inteiro mais próximo
<code>Math.cos</code>	Calcula um cosseno.
<code>Math.exp</code>	Calcula um valor exponencial.
<code>Math.floor</code>	Arredonda um número para o inteiro mais próximo
<code>Math.log</code>	Calcula um logaritmo natural.
<code>Math.max</code>	Retorna o maior de dois inteiros.
<code>Math.min</code>	Retorna o menor de dois inteiros.
<code>Math.pow</code>	Calcula $x$ elevado à potência de $y$ .
<code>Math.random</code>	Retorna um número pseudo-aleatório entre 0.0 e 1.0.
<code>Math.round</code>	Arredonda para o inteiro mais próximo.
<code>Math.sin</code>	Calcula um seno.
<code>Math.sqrt</code>	Calcula uma raiz quadrada.
<code>Math.tan</code>	Calcula uma tangente.

## Resumo das propriedades do objeto Math

Todas as propriedades do objeto Math são constantes.

Propriedade	Descrição
<code>Math.E</code>	Constante de Euler e a base de logaritmos naturais (aproximadamente 2,718).
<code>Math.LN2</code>	O logaritmo natural de 2 (aproximadamente 0,693).
<code>Math.LOG2E</code>	O logaritmo de base 2 de $e$ (aproximadamente 1,442).
<code>Math.LN10</code>	O logaritmo natural de 10 (aproximadamente 2,302).
<code>Math.LOG10E</code>	O logaritmo de base 10 de $e$ (aproximadamente 0,434).
<code>Math.PI</code>	A razão entre a circunferência de um círculo e o seu diâmetro (aproximadamente 3,14159).
<code>Math.SQRT1_2</code>	O inverso da raiz quadrada de 1/2 (aproximadamente 0,707).
<code>Math.SQRT2</code>	A raiz quadrada de 2 (aproximadamente 1,414).

## Math.abs

### Disponibilidade

Flash Player 5. No Flash Player 4, os métodos e propriedades do objeto Math são emulados com aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash Player 5.

### Uso

```
Math.abs(x)
```

### Parâmetros

*x* Um número.

### Retorna

Um número.

### Descrição

Método; calcula e retorna um valor absoluto do número especificado pelo parâmetro *x*.

## Math.acos

### Disponibilidade

Flash Player 5. No Flash Player 4, os métodos e propriedades do objeto Math são emulados com aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash Player 5.

### Uso

```
Math.acos(x)
```

### Parâmetros

*x* Um número de -1,0 a 1,0.

### Retorna

Nada.

### Descrição

Método; calcula e retorna o arco cosseno do número especificado no parâmetro *x*, em radianos.

## Math.asin

### Disponibilidade

Flash Player 5. No Flash Player 4, os métodos e propriedades do objeto Math são emulados com aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash Player 5.

### Uso

```
Math.asin(x);
```

### Parâmetros

*x* Um número de -1,0 a 1,0.

### Retorna

Um número.

### Descrição

Método; calcula e retorna o arco seno de um número especificado no parâmetro *x*, em radianos.

## Math.atan

### Disponibilidade

Flash Player 5. No Flash Player 4, os métodos e propriedades do objeto Math são emulados com aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash Player 5.

### Uso

```
Math.atan(x)
```

### Parâmetros

*x* Um número.

### Retorna

Um número.

### Descrição

Método; calcula e retorna o arco tangente do número especificado no parâmetro *x*. O valor retornado está entre o pi negativo dividido por 2 e o pi positivo dividido por 2.

## Math.atan2

### Disponibilidade

Flash Player 5. No Flash Player 4, os métodos e propriedades do objeto Math são emulados com aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash Player 5.

### Uso

```
Math.atan2(y, x)
```

### Parâmetros

*x* Um número que especifica a coordenada *x* do ponto.

*y* Um número que especifica a coordenada *y* do ponto.

### Retorna

Um número.

### 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.

## Math.ceil

### Disponibilidade

Flash Player 5. No Flash Player 4, os métodos e propriedades do objeto Math são emulados com aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash Player 5.

### Uso

```
Math.ceil(x)
```

### Parâmetros

*x* Um número ou expressão.

**Retorna**

Um número.

**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.

## Math.cos

**Uso**

`Math.cos(x)`

**Parâmetros**

*x* Um ângulo medido em radianos.

**Retorna**

Um número.

**Descrição**

Método; retorna o cosseno (um valor de -1,0 a 1,0) do ângulo especificado pelo parâmetro *x*. O ângulo *x* deve ser especificado em radianos. Use as informações descritas na introdução do objeto Math para calcular um radiano.

## Math.E

**Disponibilidade**

Flash Player 5. No Flash Player 4, os métodos e propriedades do objeto Math são emulados com aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash Player 5.

**Uso**

`Math.E`

**Parâmetros**

Nenhum.

**Retorna**

Nada.

**Descrição**

Constante; uma constante matemática para a base de logaritmos naturais, apresentados como *e*. O valor aproximado de *e* é 2,71828.

**Disponibilidade**

Flash Player 5. No Flash Player 4, os métodos e propriedades do objeto Math são emulados com aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash Player 5.

## Math.exp

### Disponibilidade

Flash Player 5. No Flash Player 4, os métodos e propriedades do objeto Math são emulados com aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash Player 5.

### Uso

```
Math.exp(x)
```

### Parâmetros

*x* O expoente; um número ou expressão.

### Retorna

Um número.

### Descrição

Método; retorna o valor de base do logaritmo natural (*e*), à potência do expoente especificado no parâmetro *x*. A constante `Math.E` pode fornecer o valor de *e*.

## Math.floor

### Disponibilidade

Flash Player 5. No Flash Player 4, os métodos e propriedades do objeto Math são emulados com aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash Player 5.

### Uso

```
Math.floor(x)
```

### Parâmetros

*x* Um número ou expressão.

### Retorna

Um número.

### Descrição

Método; retorna o piso do número ou expressão especificada no parâmetro *x*. O piso é o inteiro mais próximo menor ou igual ao número ou expressão especificada.

### Exemplo

O exemplo de código a seguir retorna um valor 12:

```
Math.floor(12.5);
```

## Math.log

### Disponibilidade

Flash Player 5. No Flash Player 4, os métodos e propriedades do objeto Math são emulados com aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash Player 5.

### Uso

```
Math.log(x)
```



**Parâmetros**

$x$  Um número ou expressão com um valor maior que 0.

**Retorna**

Um número.

**Descrição**

Método; retorna o logaritmo natural do parâmetro  $x$ .

## Math.LOG2E

**Disponibilidade**

Flash Player 5. No Flash Player 4, os métodos e propriedades do objeto Math são emulados com aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash Player 5.

**Uso**

```
Math.LOG2E
```

**Parâmetros**

Nenhum.

**Retorna**

Nada.

**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.

## Math.LOG10E

**Disponibilidade**

Flash Player 5. No Flash Player 4, os métodos e propriedades do objeto Math são emulados com aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash Player 5.

**Uso**

```
Math.LOG10E
```

**Parâmetros**

Nenhum.

**Retorna**

Nada.

**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.

## Math.LN2

### Disponibilidade

Flash Player 5. No Flash Player 4, os métodos e propriedades do objeto Math são emulados com aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash Player 5.

### Uso

`Math.LN2`

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Constante; uma constante matemática do logaritmo natural de 2, expressa como  $\log_e 2$ , com um valor aproximado de 0,69314718055994528623.

## Math.LN10

### Disponibilidade

Flash Player 5. No Flash Player 4, os métodos e propriedades do objeto Math são emulados com aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash Player 5.

### Uso

`Math.LN10`

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Constante; uma constante matemática do logaritmo natural de 10, expressa como  $\log_e 10$ , com um valor aproximado de 2,3025850929940459011.

## Math.max

### Disponibilidade

Flash Player 5. No Flash Player 4, os métodos e propriedades do objeto Math são emulados com aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash Player 5.

### Uso

`Math.max(x, y)`

### Parâmetros

*x* Um número ou expressão.

*y* Um número ou expressão.

**Retorna**

Um número.

**Descrição**

Método; avalia  $x$  e  $y$  e retorna o maior valor.

## Math.min

**Disponibilidade**

Flash Player 5. No Flash Player 4, os métodos e propriedades do objeto Math são emulados com aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash Player 5.

**Uso**

`Math.min( $x$  ,  $y$ )`

**Parâmetros**

$x$  Um número ou expressão.

$y$  Um número ou expressão.

**Retorna**

Nada.

**Descrição**

Método; avalia  $x$  e  $y$  e retorna o menor valor.

## Math.PI

**Disponibilidade**

Flash Player 5. No Flash Player 4, os métodos e propriedades do objeto Math são emulados com aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash Player 5.

**Uso**

`Math.PI`

**Parâmetros**

Nenhum.

**Retorna**

Nada.

**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

## Math.pow

### Disponibilidade

Flash Player 5. No Flash Player 4, os métodos e propriedades do objeto Math são emulados com aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash Player 5.

### Uso

```
Math.pow(x , y)
```

### Parâmetros

*x* Um número a ser elevado a uma potência.

*y* Um número que especifica a potência à qual o parâmetro *x* é elevado.

### Retorna

Um número.

### Descrição

Método; calcula e retorna *x* à potência de *y*,  $x^y$ .

## Math.random

### Disponibilidade

Flash Player 5. No Flash Player 4, os métodos e propriedades do objeto Math são emulados com aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash Player 5.

### Uso

```
Math.random()
```

### Parâmetros

Nenhum.

### Retorna

Um número.

### Descrição

Método; retorna *n*, onde  $0 \leq n < 1$ .

### Consulte também

random

## Math.round

### Disponibilidade

Flash Player 5. No Flash Player 4, os métodos e propriedades do objeto Math são emulados com aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash Player 5.

### Uso

```
Math.round(x)
```

### Parâmetros

*x* Um número.

### Retorna

Um número.

### Descrição

Método; arredonda o valor do parâmetro *x* para cima ou para baixo para o inteiro mais próximo e retorna esse valor.

## Math.sin

### Disponibilidade

Flash Player 5. No Flash Player 4, os métodos e as propriedades do objeto Math são emulados através de aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash Player 5.

### Uso

```
Math.sin(x)
```

### Parâmetros

*x* Um ângulo medido em radianos.

### Retorna

Nada.

### 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.

## Math.sqrt

### Disponibilidade

Flash Player 5. No Flash Player 4, os métodos e propriedades do objeto Math são emulados com aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash Player 5.

### Uso

```
Math.sqrt(x)
```

### Parâmetros

*x* Um número ou expressão maior que ou igual a 0.

**Retorna**

Um número.

**Descrição**

Método; calcula e retorna a raiz quadrada do número especificado.

## Math.SQRT1\_2

**Disponibilidade**

Flash Player 5. No Flash Player 4, os métodos e propriedades do objeto Math são emulados com aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash Player 5.

**Uso**

```
Math.SQRT1_2
```

**Parâmetros**

Nenhum.

**Retorna**

Nada.

**Descrição**

Constante; uma constante matemática do inverso da raiz quadrada de meio (1/2), com um valor aproximado de 0,707106781186.

## Math.SQRT2

**Disponibilidade**

Flash Player 5. No Flash Player 4, os métodos e propriedades do objeto Math são emulados com aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash Player 5.

**Uso**

```
Math.SQRT2
```

**Parâmetros**

Nenhum.

**Descrição**

Constante; uma constante matemática para a raiz quadrada de 2, com um valor aproximado de 1,414213562373.

## Math.tan

### Disponibilidade

Flash Player 5. No Flash Player 4, os métodos e propriedades do objeto Math são emulados com aproximações e podem não ser tão precisos quanto as funções matemáticas não emuladas suportadas pelo Flash Player 5.

### Uso

```
Math.tan(x)
```

### Parâmetros

*x* Um ângulo medido em radianos.

### Retorna

Um número.

### Descrição

Método; calcula e retorna a tangente do ângulo especificado. Para calcular um radiano, use as informações apresentadas na introdução de Math (objeto).

## maxscroll

### Disponibilidade

Flash Player 4.

### Uso

```
variable_name.maxscroll
```

### Descrição

Propriedade (somente leitura); uma propriedade reprovada que indica o número da linha visível mais acima do texto em um campo quando a linha mais inferior desse campo também está visível. A propriedade `maxscroll` funciona em conjunto com a propriedade `scroll` para controlar a exibição de informações em um campo de texto. Esta propriedade pode ser recuperada, mas não modificada.

### Consulte também

`TextField.maxscroll`, `TextField.scroll`

## mbchr

### Disponibilidade

Flash Player 4. Esta função foi reprovada e substituída pelo método `String.fromCharCode`.

### Uso

```
mbchr(número)
```

### Parâmetros

*número* O número a ser convertido em um caractere de vários bytes.

### Retorna

Uma seqüência de caracteres.

**Descrição**

Função de seqüência de caracteres; converte um número de código ASCII em um caractere de vários bytes.

**Consulte também**

`String.fromCharCode`

## **mblength**

**Disponibilidade**

Flash Player 4. Esta função foi reprovada e substituída por `String` (objeto).

**Uso**

`mblength(seqüência de caracteres)`

**Parâmetros**

*seqüência de caracteres*    Uma seqüência de caracteres.

**Retorna**

Um número.

**Descrição**

Função de seqüência de caracteres; retorna o tamanho da seqüência de caracteres de vários bytes.

## **mbord**

**Disponibilidade**

Flash Player 4. Esta função foi reprovada no Flash 5 e substituída pelo método `String.charCodeAt`.

**Uso**

`mbord(caractere)`

**Parâmetros**

*caractere*    O caractere a ser convertido em um número de vários bytes.

**Retorna**

Um número.

**Descrição**

Função de seqüência de caracteres; converte o caractere especificado em um número de vários bytes.

**Consulte também**

`String.fromCharCode`



## mbsubstring

### Disponibilidade

Flash Player 4. Esta função foi substituída no Flash 5 pelo método `String.substr`.

### Uso

```
mbsubstring(valor, índice, contagem)
```

### Parâmetros

*valor* A sequência de caracteres de vários bytes da qual extrair uma nova sequência de caracteres de vários bytes.

*índice* O número do primeiro caractere a ser extraído.

*contagem* O número de caracteres a ser incluído na sequência de caracteres extraída, sem incluir o caractere índice.

### Retorna

Uma sequência de caracteres.

### Descrição

Função de sequência de caracteres; extrai uma nova sequência de caracteres de vários bytes de uma sequência de caracteres de vários bytes.

### Consulte também

`String.substr`

## método

### Disponibilidade

Flash Player 6.

### Uso

```
object.method = function ([parâmetros]) {  
    ...corpo da função...  
};
```

### Parâmetros

*object* Um identificador de um objeto.

*method* Um identificador de um método.

*parâmetros* Parâmetros que devem ser passados para a função. Um parâmetro opcional.

### Retorna

Nada.

### Descrição

Ação (modo normal apenas); permite a definição dos métodos para os objetos através do painel Ações no modo Normal. Para mais informações sobre a definição de métodos para objetos, consulte *Usando o Flash*.

## Mouse (objeto)

O objeto Mouse é um objeto de alto nível que você pode acessar sem usar um construtor. Use os métodos do objeto Mouse para ocultar e mostrar o cursor no filme. Por padrão, o ponteiro do mouse fica visível, mas é possível ocultá-lo e implementar um ponteiro personalizado usando um clipe de filme.

### Resumo do método Mouse

Método	Descrição
Mouse.addListener	Registra um objeto para receber as notificações onMouseDown, onMouseMove e onMouseUp.
Mouse.hide	Oculta o ponteiro do mouse no filme.
Mouse.removeListener	Remove um objeto registrado com o método addListener.
Mouse.show	Exibe o ponteiro do mouse no filme.

### Resumo de ouvintes de Mouse

Método	Descrição
MovieClip.onMouseDown	Notificado quando o botão do mouse é pressionado.
MovieClip.onMouseMove	Notificado quando o botão do mouse é movido.
MovieClip.onMouseUp	Notificado quando o botão do mouse é liberado.

## Mouse.addListener

### Disponibilidade

Flash Player 6.

### Uso

Mouse.addListener (*newListener*)

### Parâmetros

*newListener* Um objeto.

### Retorna

Nada.

### Descrição

Método; registra um objeto para receber notificações dos manipuladores de retorno de chamada onMouseDown, onMouseMove e onMouseUp.

O parâmetro *newListener* deve conter um objeto com métodos definidos para os eventos onMouseDown, onMouseMove e onMouseUp.

Quando o mouse é pressionado, movido ou liberado, independentemente do foco de entrada, todos os objetos ouvintes registrados com o método addListener têm o método onMouseDown, onMouseMove ou onMouseUp chamado. Vários objetos podem ouvir notificações de teclado. Se o ouvinte *newListener* já estiver registrado, nenhuma alteração ocorrerá.

## Mouse.hide

### Disponibilidade

Flash Player 5.

### Uso

```
Mouse.hide()
```

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Método; oculta o cursor em um filme. Por padrão, o cursor fica visível.

### 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

`Mouse.show`, `MovieClip._xmouse`, `MovieClip._ymouse`

## Mouse.onMouseDown

### Disponibilidade

Flash Player 6.

### Uso

```
someListener.onMouseDown
```

### Descrição

Ouvinte; notificado quando o mouse é pressionado. Para usar o ouvinte `onMouseDown`, é necessário criar um objeto ouvinte. Em seguida, você pode definir uma função para `onMouseDown` e usar o método `addListener` para registrar o ouvinte com o objeto `Mouse`, como mostrado no código a seguir:

```
someListener = new Object();  
someListener.onMouseDown = function () { ... };  
Mouse.addListener(someListener);
```

Os ouvintes permitem a cooperação de partes diferentes de código. Isso ocorre porque vários ouvintes podem receber notificações sobre um único evento.

### Consulte também

`Mouse.addListener`

## Mouse.onMouseMove

### Disponibilidade

Flash Player 6.

### Uso

*someListener.onMouseMove*

### Descrição

Ouvinte; notificado quando o mouse é movido. Para usar o ouvinte `onMouseMove`, é necessário criar um objeto ouvinte. Em seguida, é possível definir uma função para `onMouseMove` e usar o método `addListener` para registrar o ouvinte com o objeto `Mouse`, como mostrado no código a seguir:

```
someListener = new Object();
someListener.onMouseMove = function () { ... };
Mouse.addListener(someListener);
```

Os ouvintes permitem a cooperação de partes diferentes de código. Isso ocorre porque vários ouvintes podem receber notificações sobre um único evento.

### Consulte também

`Mouse.addListener`

## Mouse.onMouseUp

### Disponibilidade

Flash Player 6.

### Uso

*someListener.onMouseUp*

### Descrição

Ouvinte; notificado quando o mouse é liberado. Para usar o ouvinte `onMouseUp`, é necessário criar um objeto ouvinte. Em seguida, você pode definir uma função para `onMouseUp` e usar o método `addListener` para registrar o ouvinte com o objeto `Mouse`, como mostrado no código a seguir:

```
someListener = new Object();
someListener.onMouseUp = function () { ... };
Mouse.addListener(someListener);
```

Os ouvintes permitem a cooperação de partes diferentes de código. Isso ocorre porque vários ouvintes podem receber notificações sobre um único evento.

### Consulte também

`Mouse.addListener`

## Mouse.removeListener

### Disponibilidade

Flash Player 6.

### Uso

`Mouse.removeListener (ouvinte)`

### Parâmetros

*ouvinte* Um objeto.

### Retorna

Se o objeto *ouvinte* tiver sido removido com êxito, o método retornará `true`; se a remoção do *ouvinte* não tiver sido bem-sucedida (por exemplo, se o *ouvinte* não estava na lista de ouvintes do objeto `Mouse`), o método retornará `false`.

### Descrição

Método; remove um objeto registrado anteriormente com o método `addListener`.

## Mouse.show

### Disponibilidade

Flash Player 5.

### Uso

`Mouse.show()`

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Método; exibe o cursor em um filme. Por padrão, o cursor fica visível.

### Consulte também

`Mouse.show`, `MovieClip._xmouse`, `MovieClip._ymouse`

## MovieClip (objeto)

Os métodos do objeto `MovieClip` oferecem a mesma funcionalidade que as ações dos clipes de filme de destino. Também existem métodos adicionais que não têm ações equivalentes na caixa de ferramentas Ações do painel 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:

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

## Resumo dos métodos do objeto MovieClip

Método	Descrição
<code>MovieClip.attachMovie</code>	Anexa um filme à biblioteca.
<code>MovieClip.createEmptyMovieClip</code>	Cria um clipe de filme vazio.
<code>MovieClip.createTextField</code>	Cria um campo de texto vazio.
<code>MovieClip.duplicateMovieClip</code>	Duplica o clipe de filme especificado.
<code>MovieClip.getBounds</code>	Retorna as coordenadas x e y mínimas e máximas de um filme em um espaço de coordenadas especificado.
<code>MovieClip.getBytesLoaded</code>	Retorna o número de bytes carregados do clipe de filme especificado.
<code>MovieClip.getBytesTotal</code>	Retorna o tamanho do clipe de filme em bytes.
<code>MovieClip.getDepth</code>	Retorna a profundidade de um clipe de filme.
<code>MovieClip.getURL</code>	Recupera um documento de um URL.
<code>MovieClip.globalToLocal</code>	Converte o objeto Ponto das coordenadas do Palco nas coordenadas locais do clipe de filme especificado.
<code>MovieClip.gotoAndPlay</code>	Envia a reprodução para um quadro em específico no clipe de filme e reproduz o filme.
<code>MovieClip.gotoAndStop</code>	Envia a reprodução para um quadro em específico no clipe de filme e encerra o filme.
<code>MovieClip.hitTest</code>	Retorna <code>true</code> se há interseção entre a caixa delimitadora do clipe de filme especificado e a caixa delimitadora do clipe de filme de destino.
<code>MovieClip.loadMovie</code>	Carrega o filme no clipe de filme.
<code>MovieClip.loadVariables</code>	Carrega variáveis de um URL ou outro local no clipe de filme
<code>MovieClip.localToGlobal</code>	Converte um objeto Ponto das coordenadas locais do clipe de filme nas coordenadas globais do Palco.
<code>MovieClip.nextFrame</code>	Envia a reprodução para o próximo quadro do clipe de filme.
<code>MovieClip.play</code>	Reproduz o clipe de filme especificado.
<code>MovieClip.prevFrame</code>	Envia a reprodução para o quadro anterior do clipe de filme.
<code>MovieClip.removeMovieClip</code>	Remove o clipe de filme da Linha de tempo se ele foi criado com uma ação <code>duplicateMovieClip</code> ou com o método <code>attachMovie</code> .
<code>MovieClip.setMask</code>	Especifica um clipe de filme como uma máscara para outro clipe de filme.
<code>MovieClip.startDrag</code>	Especifica um clipe de filme como arrastável e começa a arrastá-lo.
<code>MovieClip.stop</code>	Pára o filme que está sendo reproduzido no momento.
<code>MovieClip.stopDrag</code>	Pára o arraste de qualquer clipe de filme que esteja sendo arrastado.
<code>MovieClip.swapDepths</code>	Troca o nível de profundidade de dois filmes.
<code>MovieClip.unloadMovie</code>	Remove um filme que foi carregado com a ação <code>loadMovie</code> .

## Resumo de métodos de desenho de MovieClip

Método	Descrição
<code>MovieClip.beginFill</code>	Começa a desenhar um preenchimento no Palco.
<code>MovieClip.beginGradientFill</code>	Começa a desenhar um preenchimento de gradiente no Palco.
<code>MovieClip.clear</code>	Remove todos os comandos de desenho associados a uma instância de clipe de filme.
<code>MovieClip.curveTo</code>	Desenha uma curva utilizando o último estilo de linha.
<code>MovieClip.endFill</code>	Conclui o preenchimento especificado por <code>beginFill</code> ou <code>beginGradientFill</code> .
<code>MovieClip.lineStyle</code>	Define o traço das linhas criadas com os métodos <code>lineTo</code> e <code>curveTo</code> .
<code>MovieClip.lineTo</code>	Desenha uma linha utilizando o estilo de linha atual.
<code>MovieClip.moveTo</code>	Move a posição do desenho especificado para determinadas coordenadas.

## Resumo das propriedades do objeto MovieClip

Propriedade	Descrição
<code>MovieClip._alpha</code>	O valor de transparência de uma instância de clipe de filme.
<code>MovieClip._currentframe</code>	O número do quadro no qual a reprodução está localizada no momento.
<code>MovieClip._droptarget</code>	O caminho absoluto em notação de sintaxe de barra da instância do clipe de filme na qual um clipe de filme arrastável foi solto.
<code>MovieClip.enabled</code>	Indica se um clipe de filme de botão está ativado.
<code>MovieClip.focusEnabled</code>	Permite que um clipe de filme receba o foco.
<code>MovieClip._focusrect</code>	Indica se um clipe de filme focalizado tem um retângulo amarelo ao seu redor.
<code>MovieClip._framesloaded</code>	O número de quadros que foram carregados de um filme em fluxo.
<code>MovieClip._height</code>	A altura de uma instância de clipe de filme em pixels.
<code>MovieClip.hitArea</code>	Designa outro clipe de filme para atuar como a área de clique de um clipe de filme de botão.
<code>MovieClip._highquality</code>	Define a qualidade de processamento de um filme.
<code>MovieClip._name</code>	O nome da instância de um clipe de filme.
<code>MovieClip._parent</code>	Uma referência ao clipe de filme que inclui outro clipe de filme.
<code>MovieClip._rotation</code>	O grau de rotação de uma instância de clipe de filme.
<code>MovieClip._soundbuftime</code>	O número de segundos decorridos antes de um som começar a ser reproduzido.
<code>MovieClip.tabChildren</code>	Indica se os filhos de um clipe de filme são incluídos na ordenação automática de guias.
<code>MovieClip.tabEnabled</code>	Indica se um clipe de filme é incluído na ordenação de guias.
<code>MovieClip.tabIndex</code>	Indica a ordem de guias de um objeto.
<code>MovieClip._target</code>	O caminho de destino de uma instância de clipe de filme.
<code>MovieClip._totalframes</code>	O número total de quadros de uma instância de clipe de filme.

Propriedade	Descrição
<code>MovieClip.trackAsMenu</code>	Indica se outros botões podem receber eventos de liberação de mouse.
<code>MovieClip._url</code>	O URL do arquivo SWF a partir do qual um clipe de filme foi descarregado.
<code>MovieClip.useHandCursor</code>	Determina se a mão é exibida quando um usuário rola o cursor do mouse sobre um clipe de filme de botão.
<code>MovieClip._visible</code>	Um valor booleano que determina se uma instância de clipe de filme está oculta ou visível.
<code>MovieClip._width</code>	A largura de uma instância de clipe de filme em pixels.
<code>MovieClip._x</code>	A coordenada x de uma instância de clipe de filme.
<code>MovieClip._xmouse</code>	A coordenada x do cursor em uma instância de clipe de filme.
<code>MovieClip._xscale</code>	O valor que especifica a porcentagem para o dimensionamento horizontal de um clipe de filme.
<code>MovieClip._y</code>	A coordenada y de uma instância de clipe de filme.
<code>MovieClip._ymouse</code>	A coordenada y do cursor em uma instância de clipe de filme.
<code>MovieClip._yscale</code>	O valor que especifica a porcentagem para o dimensionamento vertical de um clipe de filme.

## Resumo de manipuladores de evento do objeto **MovieClip**

Propriedade	Descrição
<code>MovieClip.onData</code>	Chamada quando todos os dados são carregados em um clipe de filme.
<code>MovieClip.onDragOut</code>	Chamada enquanto o ponteiro está fora do botão, o botão do mouse é pressionado no interior e rola para fora da área do botão.
<code>MovieClip.onDragOver</code>	Chamada enquanto o ponteiro está sobre o botão, o botão do mouse foi pressionado, rolando para fora do botão e, a seguir, rolando de volta sobre o botão.
<code>MovieClip.onEnterFrame</code>	Chamada continuamente na taxa de quadros do filme. As ações associadas ao evento do clipe <code>enterFrame</code> são processadas depois das ações que tenham sido anexadas aos quadros afetados.
<code>MovieClip.onKeyDown</code>	Chamada quando uma tecla é pressionada. Use os métodos <code>Key.getCode</code> e <code>Key.getAscii</code> para recuperar informações sobre a última tecla pressionada.
<code>MovieClip.onKeyUp</code>	Chamada quando uma tecla é liberada.
<code>MovieClip.onKillFocus</code>	Chamada quando o foco é removido de um botão.
<code>MovieClip.onLoad</code>	Chamada quando o clipe de filme é criado e aparece na Linha de tempo.
<code>MovieClip.onMouseDown</code>	Chamada quando o botão esquerdo do mouse é pressionado.
<code>MovieClip.onMouseMove</code>	Chamada sempre que o mouse é movido.
<code>MovieClip.onMouseUp</code>	Chamada quando o botão esquerdo do mouse é liberado.
<code>MovieClip.onPress</code>	Chamada quando o mouse é pressionado enquanto o ponteiro está sobre um botão.
<code>MovieClip.onRelease</code>	Chamada quando o mouse é liberado enquanto o ponteiro está sobre um botão.



Propriedade	Descrição
<code>MovieClip.onReleaseOutside</code>	Chamada quando o mouse é liberado enquanto o ponteiro está fora de um botão, depois que o botão é pressionado enquanto o ponteiro está dentro do botão.
<code>MovieClip.onRollOut</code>	Chamada quando o ponteiro rola para fora da área de um botão.
<code>MovieClip.onRollOver</code>	Chamada quando o ponteiro do mouse rola sobre um botão.
<code>MovieClip.onSetFocus</code>	Chamada quando um botão tem o foco de entrada e uma tecla é liberada.
<code>MovieClip.onUnload</code>	Chamada no primeiro quadro depois que o clipe de filme é 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.

## MovieClip.\_alpha

### Disponibilidade

Flash Player 4.

### Uso

*myMovieClip.\_alpha*

### Descrição

Propriedade; define ou recupera a transparência alfa (*valor*) do clipe de filme especificado por *MovieClip*. A faixa de valores válidos vai 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, você ainda pode clicar em um botão em um clipe de filme que tenha a propriedade *\_alpha* definida como 0.

### Exemplo

Os comandos a seguir definem como 30% a propriedade *\_alpha* de um clipe de filme chamado *star* quando o usuário clica no botão.

```
on(release) {
    star._alpha = 30;
}
```

## MovieClip.attachMovie

### Disponibilidade

Flash Player 5.

### Uso

*myMovieClip.attachMovie( idName, newName, profundidade [, initObject] )*

### Parâmetros

*idName* O nome de vinculação do símbolo do clipe de filme na biblioteca a ser anexada a um clipe de filme no Palco. É o nome inserido no campo Identificador na caixa de diálogo Propriedades de Vinculação do Símbolo.

*newname* Um nome de instância único para o clipe de filme que está sendo anexado ao clipe de filme.

*profundidade* Um inteiro que especifica o nível de profundidade no qual o filme é colocado.

*initObject* Um objeto que contém propriedades que devem ser utilizadas para preencher o clipe de filme recém-anexado. Este parâmetro permite que os cliques de filme criados dinamicamente recebam parâmetros de clipe. Se *initObject* não for um objeto, ele será ignorado. Todas as propriedades de *initObject* são copiadas na nova instância. As propriedades especificadas com *initObject* estão disponíveis para a função construtora. Este parâmetro é opcional.

#### **Retorna**

Nada.

#### **Descrição**

Método; pega um símbolo da biblioteca e o anexa ao filme no Palco especificado por *MovieClip*. Use a ação ou método `removeMovieClip` ou `unloadMovie` para remover um clipe de filme anexado com `attachMovie`.

#### **Exemplo**

O exemplo a seguir anexa o símbolo com o identificador de vinculação “círculo” à instância de clipe de filme localizada no Palco, no filme.

```
on (release) {  
    thing.attachMovie( "círculo", "círculo1", 2 );  
}
```

#### **Consulte também**

`MovieClip.removeMovieClip`, `MovieClip.unloadMovie`, `Object.registerClass`, `removeMovieClip`

## **MovieClip.beginFill**

#### **Disponibilidade**

Flash Player 6.

#### **Uso**

*myMovieClip.beginFill* ([*rgb*[, *alfa*]])

#### **Parâmetro**

*rgb* Um valor de cor hexadecimal (por exemplo, vermelho corresponde a 0xFF0000, azul a 0x0000FF e assim por diante). Caso este valor não seja fornecido ou esteja indefinido, nenhum preenchimento será criado.

*alfa* Um número inteiro entre 0 e 100 que especifica o valor alfa do preenchimento. Se este valor não for informado, a opção 100 (sólido) será usada. Se o valor for menor do que 0, o Flash usará 0. Se o valor for maior do que 100, o Flash usará 100.

#### **Retorna**

Nada.

#### **Descrição**

Método; indica o início de um novo caminho de desenho. Se houver um caminho aberto (isto é, se a posição atual do desenho não for igual à posição anterior especificada em um método `moveTo`) e se houver um preenchimento associado a ele, esse caminho será fechado com uma linha e preenchido em seguida. Trata-se de um processo semelhante ao que ocorre quando o método `endFill` é chamado.

## Consulte também

MovieClip.beginGradientFill, MovieClip.endFill

# MovieClip.beginGradientFill

## Disponibilidade

Flash Player 6.

## Uso

*myMovieClip.beginGradientFill (fillType, cores, alfas, proporções, matriz)*

## Parâmetro

*fillType* A sequência de caracteres "linear" ou "radial".

*cores* Uma matriz de valores de cores hexadecimais RGB a ser utilizada no gradiente (por exemplo, vermelho corresponde a 0xFF0000, azul a 0x0000FF e assim por diante).

*alfas* Uma matriz de valores alfa para as cores correspondentes na matriz *cores*; a faixa de valores válidos vai de 0 a 100. Se o valor for menor do que 0, o Flash usará 0. Se o valor for maior do que 100, o Flash usará 100.

*proporções* Uma matriz de razões de distribuição de cores; a faixa de valores válidos vai de 0 a 255. Este valor define a porcentagem de largura em que o exemplo de cor é realizado a 100 por cento.

*matriz* Uma matriz de transformação que é um objeto que possui um dos dois conjuntos de propriedades a seguir:

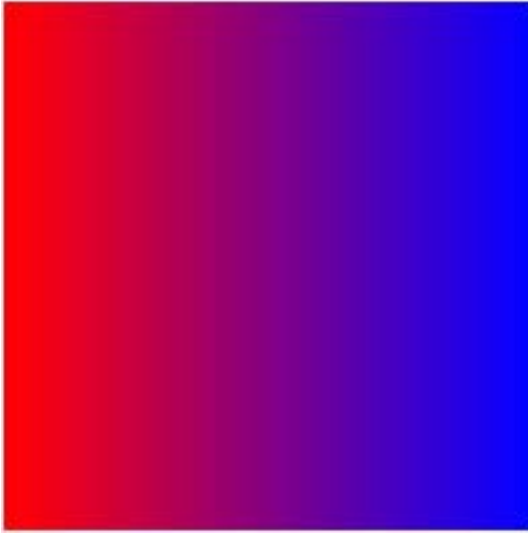
- *a, b, c, d, e, f, g, h, i*, que pode ser usado para descrever uma matriz do tipo 3 x 3 da seguinte forma:

```
a b c
d e f
g h i
```

O exemplo a seguir usa um método `beginGradientFill` com um parâmetro *matrix* que é um objeto com essas propriedades.

```
_root.createEmptyMovieClip( "grad", 1 );
    with ( _root.grad )
    {
        colors = [ 0xFF0000, 0x0000FF ];
        alphas = [ 100, 100 ];
        ratios = [ 0, 0xFF ];
        matrix = { a:200, b:0, c:0, d:0, e:200, f:0, g:200, h:200, i:1
    };
        beginGradientFill( "linear", cores, alfas, proporções, matriz
    );
        moveto(100,100);
        lineto(100,300);
        lineto(300,300);
        lineto(300,100);
        lineto(100,100);
        endFill();
    }
```

Se não houver uma propriedade *matrixType*, todos os outros parâmetros serão necessários; a função falhará se faltar qualquer parâmetro. Esta matriz dimensiona, converte, gira e inclina o gradiente da unidade definido em (-1,-1) e (1,1).<



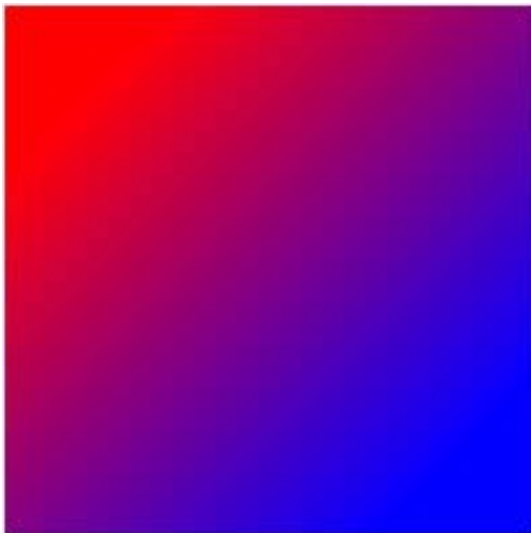
- *matrixType*, *x*, *y*, *w*, *h*, *r*.

As propriedades indicam o seguinte: *matrixType* é a sequência de caracteres "box", *x* é a posição horizontal relativa ao ponto de registro do clipe pai do canto superior esquerdo do gradiente, *y* é a posição vertical relativa ao ponto de registro do clipe pai do canto superior esquerdo do gradiente, *w* é a largura do gradiente, *h* é sua altura e *r* é a sua rotação em radianos.

O exemplo a seguir usa um método `beginGradientFill` com um parâmetro *matrix* que é um objeto com essas propriedades.

```
_root.createEmptyMovieClip( "grad", 1 );
    with ( _root.grad )
    {
        colors = [ 0xFF0000, 0x0000FF ];
        alphas = [ 100, 100 ];
        ratios = [ 0, 0xFF ];
        matrix = { matrixType:"box", x:100, y:100, w:200, h:200, r:(45/
180)*Math.PI };
        beginGradientFill( "linear", cores, alfas, proporções, matriz
);
        moveto(100,100);
        lineto(100,300);
        lineto(300,300);
        lineto(300,100);
        lineto(100,100);
        endFill();
    }
```

Se houver uma propriedade *matrixType*, ela deverá ser igual a "box" e todos os outros parâmetros serão necessários. A função falhará se uma dessas condições não for atendida.



#### **Retorna**

Nada.

#### **Descrição**

Método; indica o início de um novo caminho de desenho. Se o primeiro parâmetro estiver indefinido, ou se nenhum parâmetro tiver sido passado, o caminho não terá preenchimento. Se houver um caminho aberto (isto é, se a posição atual do desenho não for igual à posição anterior especificada em um método `moveTo`) e se houver um preenchimento associado a ele, esse caminho será fechado com uma linha e preenchido em seguida. Trata-se de um processo semelhante ao que ocorre quando você chama o método `endFill`.

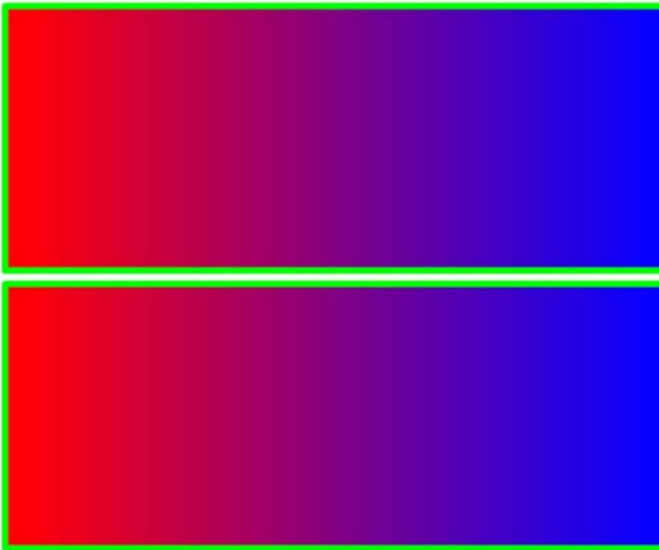
Este método falhará se qualquer uma das condições a seguir for encontrada:

- O número de itens nos parâmetros *cores*, *alfas* e *proporções* não é igual.
- O parâmetro *fillType* não é "linear" ou "radial".
- Um dos campos no objeto para o parâmetro *matrix* está ausente ou é inválido.

### Exemplo

O código a seguir usa os dois métodos para desenhar dois retângulos empilhados com um preenchimento de gradiente vermelho e azul e um traço verde sólido de 5 pontos.

```
_root.createEmptyMovieClip("goober",1);
with ( _root.goober )
{
    colors = [ 0xFF0000, 0x0000FF ];
    alphas = [ 100, 100 ];
    ratios = [ 0, 0xFF ];
    lineStyle( 5, 0x00ff00 );
    matrix = { a:500,b:0,c:0,d:0,e:200,f:0,g:350,h:200,i:1};
    beginGradientFill( "linear", cores, alfas, proporções, matriz );
    moveto(100,100);
    lineto(100,300);
    lineto(600,300);
    lineto(600,100);
    lineto(100,100);
    endFill();
    matrix = { matrixType:"box", x:100, y:310, w:500, h:200, r:(0/180)*Math.PI
    };
    beginGradientFill( "linear", cores, alfas, proporções, matriz );
    moveto(100,310);
    lineto(100,510);
    lineto(600,510);
    lineto(600,310);
    lineto(100,310);
    endFill();
}
```



### Consulte também

`MovieClip.beginFill`, `MovieClip.endFill`, `MovieClip.lineStyle`, `MovieClip.lineTo`, `MovieClip.moveTo`

## MovieClip.clear

### Disponibilidade

Flash Player 6.

### Uso

```
myMovieClip.clear()
```

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Método; remove todos os comandos de desenho associados a um clipe de filme. As formas e linhas criadas com as ferramentas de desenho do Flash não são afetadas. Quando o método `clear` é chamado, o estilo de linha atual também é removido.

### Consulte também

`MovieClip.lineStyle`

## MovieClip.createEmptyMovieClip

### Disponibilidade

Flash Player 6.

### Uso

```
myMovieClip.createEmptyMovieClip (instanceName, profundidade)
```

### Parâmetro

*instanceName* Uma sequência de caracteres que identifica o nome da instância do novo clipe de filme.

*profundidade* Um inteiro que especifica a profundidade do novo clipe de filme.

### Retorna

Nada.

### Descrição

Método; cria um clipe de filme vazio como filho de outro clipe de filme existente. Este método apresenta um comportamento semelhante ao de `attachMovie`, só que não é preciso fornecer um nome de vinculação externo para o novo clipe de filme. O ponto de registro de um clipe de filme vazio recém-criado é o canto superior esquerdo. Este método falhará se um dos parâmetros estiver faltando.

### Consulte também

`MovieClip.attachMovie`

# MovieClip.createTextField

## Disponibilidade

Flash Player 6.

## Uso

*myMovieClip.createTextField (instanceName, profundidade, x, y, largura, altura)*

## Parâmetros

*instanceName* Uma seqüência de caracteres que identifica o nome da instância do novo campo de texto.

*profundidade* Um inteiro positivo que especifica a profundidade do novo campo de texto.

*x* Um inteiro que especifica a coordenada x do novo campo de texto.

*y* Um inteiro que especifica a coordenada y do novo campo de texto.

*largura* Um inteiro positivo que especifica a largura do novo campo de texto.

*altura* Um inteiro positivo que especifica a altura do novo campo de texto.

## Retorna

Nada.

## Descrição

Método; cria um novo campo de texto vazio como filho do clipe de filme especificado pelo parâmetro *MovieClip*. Use o método `createTextField` para criar campos de texto durante a reprodução de um filme. O campo de texto é posicionado em (*x*, *y*) com as dimensões de *largura* por *altura*. Os parâmetros *x* e *y* são relativos ao clipe de filme recipiente; esses parâmetros correspondem às propriedades `_x` e `_y` do campo de texto. Os parâmetros *largura* e *altura* correspondem às propriedades `_width` e `_height` do campo de texto.

As propriedades padrão de um campo de texto são as seguintes:

```
type = "dynamic",
border = false,
background = false,
password = false,
multiline = false,
html = false,
embedFonts = false,
variable = null,
maxChars = null
```



Um campo de texto criado com `createTextField` recebe o seguinte objeto `TextFormat` padrão:

```
font = "Times New Roman"
size = 12
textColor = 0x000000
bold = false
italic = false,
underline = false
url = ""
target = ""
align = "left"
leftMargin = 0
rightMargin = 0
indent = 0
leading = 0
bullet = false
tabStops = [] (matriz vazia)
```

### Exemplo

O exemplo a seguir cria um campo de texto com uma largura de 300, uma altura de 100, uma coordenada *x* de 100, uma coordenada *y* de 100, sem borda, com texto vermelho e sublinhado.

```
_root.createTextField("meutexto",1,100,100,300,100);
mytext.multiline = true;
mytext.wordWrap = true;
mytext.border = false;

myformat = new TextFormat();
myformat.color = 0xff0000;
myformat.bullet = false;
myformat.underline = true;

mytext.text = "este é o meu primeiro texto de objeto de campo de teste";
mytext.setTextFormat(myformat);
```

### Consulte também

`TextFormat` (objeto)

## MovieClip.\_currentframe

### Disponibilidade

Flash Player 4.

### Uso

*myMovieClip.\_currentframe*

### Descrição

Propriedade (somente leitura); retorna o número do quadro no qual a reprodução está localizada na Linha de tempo especificada por *MovieClip*.

### Exemplo

O exemplo a seguir usa a propriedade `_currentframe` para orientar a reprodução de `actionClip` do clipe de filme para avançar cinco quadros em relação à sua posição atual.

```
actionClip.gotoAndStop(_currentframe + 5);
```

# MovieClip.curveTo

## Disponibilidade

Flash Player 6.

## Uso

*myMovieClip.curveTo (controlX, controlY, anchorX, anchorY)*

## Parâmetros

*controlX* Um inteiro que especifica uma posição horizontal relativa ao ponto de registro do clipe de filme pai do ponto de controle.

*controlY* Um inteiro que especifica uma posição vertical relativa ao ponto de registro do clipe de filme pai do ponto de controle.

*anchorX* Um inteiro que especifica uma posição horizontal relativa ao ponto de registro do clipe de filme pai do próximo ponto de ancoragem.

*anchorY* Um inteiro que especifica uma posição vertical relativa ao ponto de registro do clipe de filme pai do próximo ponto de ancoragem.

## Retorna

Nada.

## Descrição

Métodos; desenha uma curva utilizando o estilo de linha atual da posição de desenho para (*anchorX*, *anchorY*) com o ponto de controle especificado por (*controlX*, *controlY*). Em seguida, a posição atual do desenho é definida para (*anchorX*, *anchorY*). Se o clipe de filme que você está criando apresentar conteúdo elaborado com as ferramentas de desenho do Flash, as chamadas para *curveTo* serão desenhadas sob esse conteúdo. Se você chamar *curveTo* antes de realizar qualquer chamada a *moveTo*, o padrão de posição atual do desenho será (0, 0). Se faltar algum parâmetro, o método falhará e a posição atual do desenho não será alterada.

## Exemplo

O exemplo a seguir desenha um círculo com uma linha azul sólida fina e um preenchimento vermelho sólido.

```
_root.createEmptyMovieClip( "círculo", 1 );
with ( _root.círculo )
{
    lineStyle( 0, 0x0000FF, 100 );
    beginFill( 0xFF0000 );
    moveTo( 500, 500 );
    curveTo( 600, 500, 600, 400 );
    curveTo( 600, 300, 500, 300 );
    curveTo( 400, 300, 400, 400 );
    curveTo( 400, 500, 500, 500 );
    endFill();
}
```

## Consulte também

MovieClip.beginFill, MovieClip.createEmptyMovieClip, MovieClip.endFill,  
MovieClip.lineStyle, MovieClip.lineTo, MovieClip.moveTo

## MovieClip.\_droptarget

### Disponibilidade

Flash Player 4.

### Uso

*myMovieClip.\_droptarget*

### Descrição

Propriedade (somente leitura); retorna o caminho absoluto, em notação de sintaxe de barra, da instância do clipe de filme em que *MovieClip* foi solto. A propriedade `_droptarget` sempre retorna um caminho iniciado com uma barra (/). Para comparar a propriedade `_droptarget` de uma instância a uma referência, use a função `eval` para converter o valor retornado de sintaxe de barra para uma referência de sintaxe de ponto.

### 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`

## MovieClip.cloneMovieClip

### Disponibilidade

Flash Player 5.

### Uso

*myMovieClip.cloneMovieClip(newname, profundidade [,initObject])*

### Parâmetros

*newname* Um identificador exclusivo do clipe de filme duplicado.

*profundidade* Um número exclusivo que especifica o nível de profundidade no qual o filme especificado deve ser colocado.

*initObject* Um objeto que contém propriedades com as quais o clipe de filme duplicado deve ser preenchido. Este parâmetro permite que os cliques de filme criados dinamicamente recebam parâmetros de clipe. Se *initObject* não for um objeto, ele será ignorado. Todas as propriedades de *initObject* são copiadas na nova instância. As propriedades especificadas com *initObject* estão disponíveis para a função construtora. Este parâmetro é opcional.

**Retorna**

Nada.

**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 `duplicateMovieClip` é chamado. As variáveis no clipe de filme pai não são copiadas para o clipe de filme duplicado. Os clipes de filme criados com o método `duplicateMovieClip` não são duplicados quando você chama `duplicateMovieMethod` em seus pais. Se o clipe de filme pai for excluído, o clipe de filme duplicado também o será. Os clipes de filme adicionados com `duplicateMovieClip` podem ser excluídos com a ação ou o método `removeMovieClip`.

**Consulte também**

`duplicateMovieClip`, `MovieClip.removeMovieClip`, `removeMovieClip`

## MovieClip.enabled

**Disponibilidade**

Flash Player 6.

**Uso**

```
myMovieClip.enabled
```

**Descrição**

Propriedade; um valor booleano que indica se um clipe de filme de botão está ativado. O valor padrão de `enabled` é `true`. Se `enabled` for definida como `false`, os métodos de retorno de chamada do clipe de filme de botão e os eventos de ação `on` não serão mais chamados. Além disso, os quadros `Over`, `Down` e `Up` são desativados. A propriedade `enabled` não afeta a Linha de tempo do clipe de filme de botão; se um clipe de filme estiver sendo reproduzido, esse processo não será interrompido. O clipe de filme continua a receber eventos de `Movieclip` (por exemplo, `mouseDown`, `mouseUp`, `keyDown` e `keyUp`).

A propriedade `enabled` regula apenas as propriedades de botão de um clipe de filme de botão. É possível alterar a propriedade `enabled` a qualquer momento; o clipe de filme de botão modificado é ativado ou desativado imediatamente. A propriedade `enabled` pode ser lida a partir de um objeto de protótipo. Se `enabled` estiver definida como `false`, o objeto não será incluído na ordenação automática de guias.

## MovieClip.endFill

**Disponibilidade**

Flash Player 6.

**Uso**

```
myMovieClip.endFill()
```

**Parâmetros**

Nenhum.

**Retorna**

Nada.

### Descrição

Método; aplica um preenchimento às linhas e curvas adicionadas desde a última chamada ao método `beginFill` ou `beginGradientFill`. O Flash usa o preenchimento especificado na chamada anterior a `beginFill` ou `beginGradientFill`. Se a posição de desenho atual não for igual à posição anterior especificada em um método `moveTo` e um preenchimento for definido, o caminho será fechado com uma linha e preenchido em seguida.

## MovieClip.focusEnabled

### Disponibilidade

Flash Player 6.

### Uso

*myMovieClip.focusEnabled*

### Descrição

Propriedade; se o valor for `undefined` ou `false`, um clipe de filme não poderá receber o foco de entrada, a menos que seja um clipe de filme de botão. Se o valor da propriedade `focusEnabled` for `true`, um clipe de filme poderá receber o foco de entrada mesmo que não seja do tipo de botão.

## MovieClip.\_focusrect

### Disponibilidade

Flash Player 6.

### Uso

*myMovieClip.\_focusrect*

### Descrição

Propriedade; um valor booleano que especifica se um clipe de filme apresenta um retângulo amarelo ao seu redor quando tem o foco do teclado. Esta propriedade pode substituir a propriedade global `_focusrect`.

## MovieClip.\_framesloaded

### Disponibilidade

Flash Player 4.

### Uso

*myMovieClip.\_framesloaded*

### 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. 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.

### Exemplo

O exemplo a seguir utiliza a propriedade `_framesloaded` para iniciar um filme quando todos os quadros estão carregados. Se nem todos os quadros estiverem carregados, a propriedade `_xscale` da instância do clipe de filme `loader` será aumentada proporcionalmente para criar uma barra de progresso.

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

## MovieClip.getBounds

### Disponibilidade

Flash Player 5.

### Uso

```
myMovieClip.getBounds(targetCoordinateSpace)
```

### Parâmetros

*targetCoordinateSpace* O caminho de destino da Linha de tempo cujo sistema de coordenadas você deseja usar como ponto de referência.

### Retorna

Um objeto com as propriedades `xMin`, `xMax`, `yMin` e `yMax`.

### Descrição

Método; retorna as propriedades correspondentes aos valores de coordenadas mínimos e máximos *x* e *y* da instância especificada por *MovieClip* para o parâmetro *targetCoordinateSpace*.

**Observação:** 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.

### Exemplo

No exemplo a seguir, o objeto retornado pelo método `getBounds` é atribuído ao identificador `clipBounds`. Em seguida, é possível acessar os valores de cada propriedade e utilizá-los em um script. Outra instância de clipe de filme, chamada `clip2`, é colocada junto de `clip` neste script.

```
clipBounds = clip.getBounds(_root);
clip2._x = clipBounds.xMax;
```

### Consulte também

`MovieClip.globalToLocal`, `MovieClip.localToGlobal`

## MovieClip.getBytesLoaded

### Disponibilidade

Flash Player 6.

### Uso

*myMovieClip.getBytesLoaded()*

### Parâmetros

Nenhum.

### Retorna

Um inteiro que indica o número de bytes carregados.

### Descrição

Método; retorna o número de bytes carregados (enviados) para o objeto Movieclip especificado. É possível comparar o valor do método `getBytesLoaded` com o do método `getBytesTotal` para determinar a porcentagem de um clipe de filme que já foi carregada.

### Consulte também

`MovieClip.getBytesTotal`

## MovieClip.getBytesTotal

### Disponibilidade

Flash Player 5.

### Uso

*myMovieClip.getBytesTotal()*

### Parâmetros

Nenhum.

### Retorna

Um número inteiro que indica o tamanho total, em bytes, do objeto MovieClip especificado.

### Descrição

Método; retorna o tamanho, em bytes, do objeto MovieClip especificado. No caso de clipes de filme externos (o filme 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.

### Consulte também

`MovieClip.getBytesLoaded`

## MovieClip.getDepth

### Disponibilidade

Flash Player 6.

### Uso

*myMovieClip.getDepth*

### Parâmetros

Nenhum.

**Retorna**

Um inteiro.

**Descrição**

Método; retorna a profundidade de uma instância de clipe de filme.

## MovieClip.getURL

**Disponibilidade**

Flash Player 5.

**Uso**

```
myMovieClip.getURL(URL [,janela, variáveis])
```

**Parâmetros**

*URL* O URL a partir do qual obter o documento.

*janela* Um parâmetro opcional que especifica o nome, quadro ou expressão que determina a janela ou quadro HTML no qual o documento foi carregado. Também é possível usar um dos seguintes 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 da janela atual.

*variáveis* Um parâmetro opcional que especifica um método para o envio de variáveis associado ao filme a ser carregado. Se não houver variáveis, omita esse parâmetro; caso contrário, especifique se deseja carregar as variáveis usando um método GET ou POST. GET anexa as variáveis ao final do 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.

**Retorna**

Nada.

**Descrição**

Método; carrega um documento do URL especificado na janela especificada. O método `getURL` também pode ser usado para passar variáveis para outro aplicativo definido no URL usando o método GET ou POST.

**Consulte também**

`getURL`

## MovieClip.globalToLocal

**Disponibilidade**

Flash Player 5.

**Uso**

```
myMovieClip.globalToLocal(ponto)
```

**Parâmetros**

*ponto* O nome ou identificador de um objeto criado com o objeto genérico `Object` especificando as coordenadas *x* e *y* como propriedades.

**Retorna**

Nada.



**Descrição**

Método; converte o objeto *Ponto* das coordenadas do Palco (global) em coordenadas do clipe de filme (local).

**Exemplo**

O exemplo a seguir converte as coordenadas *x* e *y* globais do objeto *Ponto* em coordenadas locais do clipe de filme.

```
onClipEvent(mouseMove) {  
    point = new object();  
    point.x = _root._xmouse;  
    point.y = _root._ymouse;  
    globalToLocal(point);  
    trace(_root._xmouse + " " + _root._ymouse);  
    trace(point.x + " " + point.y);  
    updateAfterEvent();  
}
```

**Consulte também**

`MovieClip.getBounds`, `MovieClip.localToGlobal`

## MovieClip.gotoAndPlay

**Disponibilidade**

Flash Player 5.

**Uso**

*myMovieClip.gotoAndPlay(quadro)*

**Parâmetros**

*quadro* O número do quadro para o qual a reprodução é enviada.

**Retorna**

Nada.

**Descrição**

Método; inicia a reprodução do filme no quadro especificado.

**Consulte também**

`gotoAndPlay`

## MovieClip.gotoAndStop

**Disponibilidade**

Flash Player 5.

**Uso**

*myMovieClip.gotoAndStop(quadro)*

**Parâmetros**

*quadro* O número do quadro para o qual a reprodução é enviada.

**Retorna**

Nada.

**Descrição**

Método; envia a reprodução para o quadro especificado do clipe de filme e a interrompe nessa posição.

**Consulte também**

`gotoAndStop`

## MovieClip.\_height

**Disponibilidade**

Flash Player 4.

**Uso**

*myMovieClip.\_height*

**Descrição**

Propriedade; define e recupera a altura do clipe de filme, em pixels.

**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;  
}
```

## MovieClip.\_highquality

**Disponibilidade**

Flash Player 6.

**Uso**

*myMovieClip.\_highquality*

**Descrição**

Propriedade (global); especifica o nível de sem serrilhado aplicado no filme atual. 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. Esta propriedade pode substituir a propriedade global `_highquality`.

**Exemplo**

```
myMovieClip._highquality = 1;
```

**Consulte também**

`_quality`, `toggleHighQuality`

## MovieClip.hitArea

### Disponibilidade

Flash Player 6.

### Uso

*myMovieClip.hitArea*

### Retorna

Uma referência a um clipe de filme.

### Descrição

Propriedade; designa outro clipe de filme para atuar como a área de clique de um clipe de filme de botão. Se não houver uma propriedade *hitArea* ou se ela for *null* ou *undefined*, o próprio clipe de filme de botão será usado como a área de clique. O valor da propriedade *hitArea* pode ser uma referência para um objeto de clipe de filme.

É possível alterar a propriedade *hitArea* a qualquer momento; o clipe de filme de botão modificado assume imediatamente o comportamento da nova área de clique. O clipe de filme designado como área de clique não precisa ficar visível; sua forma gráfica é testada mesmo que esteja invisível. A propriedade *hitArea* pode ser lida de um objeto de protótipo.

## MovieClip.hitTest

### Disponibilidade

Flash Player 5.

### Uso

*myMovieClip.hitTest(x, y, shapeFlag)*

*myMovieClip.hitTest(destino)*

### Parâmetros

*x* A coordenada *x* da área de clique no Palco.

*y* A coordenada *y* da área de clique no Palco.

As coordenadas *x* e *y* são definidas no espaço de coordenadas globais.

*destino* O caminho de destino da área de clique que pode entrar em interseção ou se sobrepor à instância especificada por *MovieClip*. Normalmente, *destino* representa um botão ou um campo de entrada de texto.

*shapeFlag* Um valor booleano que determina se será avaliada a forma completa da instância especificada (*true*) ou apenas a caixa delimitadora (*false*). Esse parâmetro só pode ser especificado se a área de clique for identificada com os parâmetros das coordenadas *x* e *y*.

### Retorna

Nada.

### Descrição

Método; avalia a instância especificada por *MovieClip* para ver se ela se sobrepõe ou entra em interseção com a área de clique identificada pelos parâmetros de *destino* ou das coordenadas *x* e *y*.

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 clicagem ou de ponto ativo especificada.

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.

#### 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 o item *ball* do clipe de filme se sobrepõe ou entra em interseção com o item *square* do clipe de filme:

```
if(_root.ball, hitTest(_root.square)){  
    trace("ball intersects square");  
}
```

#### Consulte também

*MovieClip.getBounds*, *MovieClip.globalToLocal*, *MovieClip.localToGlobal*

## MovieClip.lineStyle

#### Disponibilidade

Flash Player 6.

#### Uso

```
myMovieClip.lineStyle ([espessura[, rgb[, alfa]])
```

#### Parâmetros

*espessura* Um inteiro que indica a espessura da linha em pontos; a faixa de valores válidos vai de 0 a 255. Se nenhum número for especificado ou se o parâmetro estiver indefinido, nenhuma linha será desenhada. Se um valor menor do que 0 for passado, o Flash usará 0. O valor 0 indica a espessura fina; 255 indica a espessura máxima. Se um valor maior do que 255 for passado, o interpretador do Flash usará 255.

*rgb* Um valor de cor hexadecimal da linha (por exemplo, vermelho corresponde a 0xFF0000, azul corresponde a 0x0000FF e assim por diante). Se nenhum valor for indicado, o Flash usará 0x000000 (preto).

*alfa* Um número inteiro que indica o valor alfa da cor da linha; a faixa de valores válidos vai de 0 a 100. Se nenhum valor for indicado, o Flash usará 100 (sólido). Se o valor for menor do que 0, o Flash usará 0 e se for maior do que 100, o Flash usará 100.

#### Retorna

Nada.

#### Descrição

Método; especifica um estilo de linha que será usado pelo Flash para as chamadas subsequentes aos métodos *lineTo* e *curveTo* até que *lineStyle* seja chamado com parâmetros diferentes. É possível chamar o método *lineStyle* durante o desenho de um caminho para especificar estilos diferentes para segmentos de linha distintos em um caminho.

**Observação:** As chamadas para *clear* redefinem o método *lineStyle* como indefinido.

### Exemplo

O código a seguir desenha um triângulo com uma linha magenta sólida de 5 pontos e sem preenchimento.

```
_root.createEmptyMovieClip( "triângulo", 1 );
with ( _root.triangle )
{
    lineStyle( 5, 0xff00ff, 100 );
    moveTo( 200, 200 );
    lineTo( 300, 300 );
    lineTo( 100, 300 );
    lineTo( 200, 200 );
}
```

### Consulte também

MovieClip.beginFill, MovieClip.beginGradientFill, MovieClip.clear,  
MovieClip.curveTo, MovieClip.lineTo, MovieClip.moveTo,

## MovieClip.lineTo

### Disponibilidade

Flash Player 6.

### Uso

*myMovieClip.lineTo (x, y)*

### Parâmetros

x Um inteiro que indica a posição horizontal relativa ao ponto de registro do clipe de filme pai.

y Um inteiro que indica a posição vertical relativa ao ponto de registro do clipe de filme pai.

### Retorna

Nada.

### Descrição

Método; desenha uma linha utilizando o estilo atual a partir da posição do desenho no momento em (x, y); a posição atual do desenho é então definida como (x, y). Se o clipe de filme no qual você está desenhando apresentar conteúdo criado com as ferramentas de desenho do Flash, - as chamadas para `lineTo` serão feitas sob o conteúdo. Se você chamar o método `lineTo` antes de realizar qualquer chamada a `moveTo`, o padrão da posição atual do desenho será (0, 0). Se faltar algum parâmetro, o método falhará e a posição atual do desenho não será alterada.

### Exemplo

O exemplo a seguir desenha um triângulo sem nenhuma linha e um preenchimento azul parcialmente transparente.

```
_root.createEmptyMovieClip( "triângulo", 1);
with (_root.triangle){
    beginFill (0x0000FF, 50);
    lineStyle (5, 0xFF00FF, 100);
    moveTo (200, 200);
    lineTo (300, 300);
    lineTo (100, 300);
    lineTo (200, 200);
    endFill();
}
```

### Consulte também

`MovieClip.beginFill`, `MovieClip.createEmptyMovieClip`, `MovieClip.endFill`,  
`MovieClip.lineStyle`, `MovieClip.moveTo`

## MovieClip.loadMovie

### Disponibilidade

Flash Player 5.

### Uso

```
myMovieClip.loadMovie("url" [,variáveis])
```

### Parâmetros

*url* Um URL absoluto ou relativo do arquivo SWF ou JPEG a ser carregado. Um caminho relativo deve ser relativo ao arquivo SWF em `_level0`. O URL deve estar no mesmo subdomínio que o URL onde o filme reside no momento. Para uso no Flash Player independente ou para teste no modo de teste de filme no aplicativo 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 parâmetro opcional que especifica um método HTTP para o envio ou carregamento de variáveis. O parâmetro deve ser a sequência de caracteres `GET` ou `POST`. Se não houver nenhuma variável a ser enviada, omita esse parâmetro. O método `GET` anexa as variáveis ao final do URL e é usado para pequenos números de variáveis. O método `POST` envia as variáveis em um cabeçalho HTTP separado e é usado para seqüências de caracteres maiores de variáveis.

### Retorna

Nada.

### Descrição

Método; carrega arquivos SWF ou JPEG em um clipe de filme no Flash Player durante a reprodução do filme original. Sem o método `loadMovie`, o Flash Player exibe um único filme (arquivo SWF) e é encerrado em seguida. O método `loadMovie` permite que você exiba vários filmes de uma vez e alterne entre os filmes sem carregar outro documento HTML.

Um filme ou imagem carregada em um clipe de filme herda as propriedades de posição, rotação e dimensionamento do clipe de filme. Utilize o caminho de destino do clipe de filme para especificar o filme carregado.

Use o método `unloadMovie` para remover filmes ou imagens carregadas com o método `loadMovie`. Use o método `loadVariables` para manter o filme ativo e atualizar as variáveis com os novos valores.

### Consulte também

`loadMovie`, `loadMovieNum`, `MovieClip.loadVariables`, `MovieClip.unloadMovie`,  
`unloadMovie`, `unloadMovieNum`

## MovieClip.loadVariables

### Disponibilidade

Flash Player 5.

### Uso

```
myMovieClip.loadVariables("url", variáveis)
```

### Parâmetros

*url* O URL absoluto ou relativo para o arquivo externo que contém as variáveis a serem carregadas. O host do URL deve estar no mesmo subdomínio que o clipe de filme.

*variáveis* Um parâmetro opcional que especifica um método HTTP para o envio de variáveis. O parâmetro deve ser a sequência de caracteres GET ou POST. Se não houver nenhuma variável a ser enviada, omita esse parâmetro. O método GET anexa as variáveis ao final do URL e é usado para pequenos números de variáveis. O método POST envia as variáveis em um cabeçalho HTTP separado e é usado para sequências de caracteres maiores de variáveis.

### Retorna

Nada.

### Descrição

Método; lê dados de um arquivo externo e define os valores das variáveis em *MovieClip*. 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 clipe de filme ativo com novos valores.

Esse método exige que o texto no URL esteja no formato MIME padrão: *aplicativo/x-www-formato de url codificado* (formato de script CGI).

### Consulte também

loadMovie, loadVariables, loadVariablesNum, MovieClip.unloadMovie

## MovieClip.localToGlobal

### Disponibilidade

Flash Player 5.

### Uso

```
myMovieClip.localToGlobal(ponto)
```

### Parâmetros

*ponto* O nome ou identificador de um objeto criado com o objeto Object, que especifica as coordenadas *x* e *y* como propriedades.

### Retorna

Nada.

### Descrição

Método; converte o objeto *Ponto* das coordenadas do clipe de filme (locais) em coordenadas do Palco (globais).

### Exemplo

O exemplo a seguir converte as coordenadas *x* e *y* do objeto *Ponto* das coordenadas do clipe de filme (local) em coordenadas do Palco (globais). As coordenadas *x* e *y* locais são especificadas com as propriedades `_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`

## MovieClip.moveTo

### Disponibilidade

Flash Player 6.

### Uso

*myMovieClip.moveTo* (*x*, *y*)

### Parâmetros

- x* Um inteiro que indica a posição horizontal relativa ao ponto de registro do clipe de filme pai.
- y* Um inteiro que indica a posição vertical relativa ao ponto de registro do clipe de filme pai.

### Retorna

Nada.

### Descrição

Método; move a posição atual do desenho para (*x*, *y*). Se faltar algum parâmetro, o método falhará e a posição atual do desenho não será alterada.

### Exemplo

Este exemplo desenha um triângulo com linhas magenta sólidas de 5 pontos e sem preenchimento. A primeira linha cria um clipe de filme vazio para o desenho. Um tipo de linha é definido no comando `with` e, em seguida, a posição inicial do desenho é indicada pelo método `moveTo`.

```
_root.createEmptyMovieClip( "triângulo", 1 );  
with ( _root.triangle )  
{  
    lineStyle( 5, 0xff00ff, 100 );  
    moveTo( 200, 200 );  
    lineTo( 300, 300 );  
    lineTo( 100, 300 );  
    lineTo( 200, 200 );  
}
```

### Consulte também

`MovieClip.createEmptyMovieClip`, `MovieClip.lineStyle`, `MovieClip.lineTo`



## MovieClip.\_name

### Disponibilidade

Flash Player 4.

### Uso

*myMovieClip.\_name*

### Descrição

Propriedade; retorna o nome da instância do clipe de filme especificado por *MovieClip*.

## MovieClip.nextFrame

### Disponibilidade

Flash Player 5.

### Uso

*myMovieClip.nextFrame()*

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Método; envia a reprodução para o próximo quadro e a encerra.

### Consulte também

`nextFrame`

## MovieClip.onData

### Disponibilidade

Flash Player 6.

### Uso

*myMovieClip.onData*

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Manipulador de eventos; chamado quando um clipe de filme recebe dados de uma chamada `loadVariables` ou `loadMovie`.

É necessário definir uma função que seja executada quando o evento é chamado.

### Exemplo

O exemplo a seguir define uma função para o método `onData` que envia uma ação `trace` à janela Saída.

```
myMovieClip.onData = function () {  
    trace ("onData chamado");  
};
```

## MovieClip.onDragOut

### Disponibilidade

Flash Player 6.

### Uso

*myMovieClip.onDragOver*

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Manipulador de eventos; chamado quando o ponteiro é pressionado e arrastado fora e, em seguida, sobre o clipe de filme.

É necessário definir uma função que seja executada quando o evento é chamado.

### Exemplo

O exemplo a seguir define uma função para o método `onDragOut` que envia uma ação `trace` à janela Saída.

```
myMovieClip.onDragOut = function () {  
    trace ("onDragOut chamado");  
};
```

### Consulte também

*MovieClip.onDragOver*

## MovieClip.onDragOver

### Disponibilidade

Flash Player 6.

### Uso

*myMovieClip.onDragOver*

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Manipulador de eventos; chamado quando o ponteiro é pressionado e arrastado fora e, em seguida, sobre o clipe de filme.

É necessário definir uma função que seja executada quando o evento é chamado.

### Exemplo

O exemplo a seguir define uma função para o método `onDragOut` que envia uma ação `trace` à janela Saída.

```
myMovieClip.onDragOver = function () {  
    trace ("onDragOver chamado");  
};
```

### Consulte também

`MovieClip.onDragOut`

## MovieClip.onEnterFrame

### Disponibilidade

Flash Player 6.

### Uso

*myMovieClip.onEnterFrame*

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Manipulador de eventos; chamado continuamente à taxa de quadros do filme. As ações associadas ao evento do clipe `enterFrame` são processadas depois das ações que tenham sido anexadas aos quadros afetados.

É necessário definir uma função que seja executada quando o evento é chamado.

### Exemplo

O exemplo a seguir define uma função para o método `onEnterFrame` que envia `trace` à janela Saída.

```
myMovieClip.onEnterFrame = function () {  
    trace ("onEnterFrame chamado");  
};
```

## MovieClip.onKeyDown

### Disponibilidade

Flash Player 6.

### Uso

*myMovieClip.onKeyDown*

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Manipulador de eventos; chamado quando um clipe de filme tem o foco de entrada e uma tecla é pressionada. O evento `onKeyDown` é chamado sem nenhum parâmetro. Use os métodos `Key.getAscii` e `Key.getCode` para determinar qual tecla foi pressionada.

É necessário definir uma função que seja executada quando o evento é chamado.

### Exemplo

O exemplo a seguir define uma função para o método `onKeyDown` que envia uma ação `trace` à janela Saída.

```
myMovieClip.onKeyDown = function () {  
    trace ("onKeyDown chamado");  
};
```

### Consulte também

`MovieClip.onKeyUp`

## MovieClip.onKeyUp

### Disponibilidade

Flash Player 6.

### Uso

*myMovieClip.onKeyUp*

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Evento; chamado quando uma tecla é liberada. O evento `onKeyUp` é chamado sem nenhum parâmetro. Use os métodos `Key.getAscii` e `Key.getCode` para determinar qual tecla foi pressionada.

É necessário definir uma função que seja executada quando o evento é chamado.

### Exemplo

O exemplo a seguir define uma função para o método `onKeyPress` que envia uma ação `trace` à janela Saída.

```
myMovieClip.onKeyUp = function () {  
    trace ("onKeyUp chamado");  
};
```

## MovieClip.onKillFocus

### Disponibilidade

Flash Player 6.

### Uso

```
myMovieClip.onKillFocus = function (newFocus) {  
    comandos;  
};
```

**Parâmetros**

*newFocus* O objeto que recebe o foco do teclado.

**Retorna**

Nada.

**Descrição**

Manipulador de eventos; um evento que é chamado quando um clipe de filme perde o foco do teclado. O método `onKillFocus` recebe um parâmetro, *newFocus*, que é um objeto representando o novo objeto a receber o foco. Se nenhum objeto receber o foco, *newFocus* conterá o valor `null`.

## MovieClip.onLoad

**Disponibilidade**

Flash Player 6.

**Uso**

*myMovieClip.onLoad*

**Parâmetros**

Nenhum.

**Retorna**

Nada.

**Descrição**

Manipulador de eventos; chamado quando o clipe de filme é criado e aparece na Linha de tempo.

É necessário definir uma função que seja executada quando o evento é chamado.

**Exemplo**

O exemplo a seguir define uma função para o método `onLoad` que envia uma ação `trace` à janela Saída:

```
myMovieClip.onLoad = function () {  
    trace ("onLoad chamado");  
};
```

## MovieClip.onMouseDown

**Disponibilidade**

Flash Player 6.

**Uso**

*myMovieClip.onMouseDown*

**Parâmetros**

Nenhum.

**Retorna**

Nada.

**Descrição**

Manipulador de eventos; chamado quando o botão do mouse é pressionado.

É necessário definir uma função que seja executada quando o evento é chamado.

**Exemplo**

O exemplo a seguir define uma função para o método `onMouseDown` que envia uma ação trace à janela Saída:

```
myMovieClip.onMouseDown = function () {  
    trace ("onMouseDown chamado");  
}
```

## MovieClip.onMouseMove

**Disponibilidade**

Flash Player 6.

**Uso**

*myMovieClip.onMouseMove*

**Parâmetros**

Nenhum.

**Retorna**

Nada.

**Descrição**

Manipulador de eventos; chamado quando o mouse é movido.

É necessário definir uma função que seja executada quando o evento é chamado.

**Exemplo**

O exemplo a seguir define uma função para o método `onMouseMove` que envia uma ação trace à janela Saída.

```
myMovieClip.onMouseMove = function () {  
    trace ("onMouseMove chamado");  
};
```

## MovieClip.onMouseUp

**Disponibilidade**

Flash Player 6.

**Uso**

*myMovieClip.onMouseUp*

**Parâmetros**

Nenhum.

**Retorna**

Nada.

**Descrição**

Manipulador de eventos; chamado quando o mouse é liberado.

É necessário definir uma função que seja executada quando o evento é chamado.

**Exemplo**

O exemplo a seguir define uma função para o método `onMouseUp` que envia uma ação `trace` à janela Saída.

```
myMovieClip.onMouseUp = function () {  
    trace ("onMouseUp chamado");  
};
```

## MovieClip.onPress

**Disponibilidade**

Flash Player 6.

**Uso**

*myMovieClip.onPress*

**Parâmetros**

Nenhum.

**Retorna**

Nada.

**Descrição**

Identificador de eventos; chamado quando o ponteiro do mouse é clicado sobre um clipe de filme.

É necessário definir uma função que seja executada quando o evento é chamado.

**Exemplo**

O exemplo a seguir define uma função para o método `onPress` que envia uma ação `trace` à janela Saída.

```
myMovieClip.onPress = function () {  
    trace ("onPress chamado");  
};
```

## MovieClip.onRelease

**Disponibilidade**

Flash Player 6.

**Uso**

*myMovieClip.onRelease*

**Parâmetros**

Nenhum.

**Retorna**

Nada.

**Descrição**

Manipulador de eventos; chamado quando um clipe de filme de botão é liberado.

É necessário definir uma função que seja executada quando o evento é chamado.

### Exemplo

O exemplo a seguir define uma função para o método `onPress` que envia uma ação `trace` à janela Saída.

```
myMovieClip.onRelease = function () {  
    trace ("onRelease chamado");  
};
```

## MovieClip.onReleaseOutside

### Disponibilidade

Flash Player 6.

### Uso

*myMovieClip.onReleaseOutside*

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Manipulador de eventos; chamado quando o mouse é liberado enquanto o ponteiro está fora do clipe de filme, depois que o botão do mouse é pressionado dentro do clipe de filme.

É necessário definir uma função que seja executada quando o evento é chamado.

### Exemplo

O exemplo a seguir define uma função para o método `onReleaseOutside` que envia uma ação `trace` à janela Saída.

```
myMovieClip.onReleaseOutside = function () {  
    trace ("onReleaseOutside chamado");  
};
```

## MovieClip.onRollOut

### Disponibilidade

Flash Player 6.

### Uso

*myMovieClip.onRollOut*

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Manipulador de eventos; chamado quando o ponteiro rola fora da área de um clipe de filme.

É necessário definir uma função que seja executada quando o evento é chamado.



### Exemplo

O exemplo a seguir define uma função para o método `onRollOut` que envia uma ação trace à janela Saída.

```
myMovieClip.onRollOut = function () {  
    trace ("onRollOut chamado");  
};
```

## MovieClip.onRollOver

### Disponibilidade

Flash Player 6.

### Uso

*myMovieClip.onRollOver*

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Manipulador de eventos; chamado quando o ponteiro rola sobre uma área do clipe de filme.

É necessário definir uma função que seja executada quando o evento é chamado.

### Exemplo

O exemplo a seguir define uma função para o método `onRollOver` que envia uma ação trace à janela Saída.

```
myMovieClip.onRollOver = function () {  
    trace ("onRollOver chamado");  
};
```

## MovieClip.onSetFocus

### Disponibilidade

Flash Player 6.

### Uso

```
myMovieClip.onSetFocus = function(oldFocus){  
    comandos;  
};
```

### Parâmetros

*oldFocus* O objeto que perde o foco.

### Retorna

Nada.

### Descrição

Manipulador de eventos; chamado quando um clipe de filme recebe o foco do teclado. O parâmetro *oldFocus* é o objeto que perde o foco. Por exemplo, se o usuário pressionar a tecla Tab para mover o foco de entrada de um clipe de filme para um campo de texto, *oldFocus* conterá a instância do clipe de filme.

Se nenhum objeto possuía o foco anteriormente, *oldFocus* conterá um valor `null`.

## MovieClip.onUnload

### Disponibilidade

Flash Player 6.

### Uso

*myMovieClip.onUnload*

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Manipulador de eventos; chamado no primeiro quadro depois que o clipe de filme é removido da Linha de tempo. As ações associadas ao evento `onUnload` são processadas antes de qualquer ação anexada ao quadro em questão. É necessário definir uma função a ser executada quando o evento for chamado.

### Exemplo

O exemplo a seguir define uma função para o método `onUnload` que envia uma ação `trace` à janela Saída.

```
myMovieClip.onUnload = function () {  
    trace ("onUnload chamado");  
};
```

## MovieClip.\_parent

### Disponibilidade

Flash Player 6.

### Uso

*myMovieClip.\_parent.property*  
*\_parent.\_parent.property*

### Descrição

Propriedade; especifica ou retorna uma referência ao clipe de filme ou objeto que contém o clipe de filme ou objeto atual. O objeto atual é o que contém o código ActionScript que faz referência a `_parent`. Use a propriedade `_parent` para especificar um caminho relativo para clipes de filme ou objetos que estiverem acima do clipe de filme ou objeto atual.

### Consulte também

`_root`, `targetPath`

## MovieClip.play

### Disponibilidade

Flash Player 5.

### Uso

*myMovieClip.play()*

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Método; move a reprodução na Linha de tempo do clipe de filme.

### Consulte também

`play`

## MovieClip.prevFrame

### Disponibilidade

Flash Player 5.

### Uso

*myMovieClip.prevFrame()*

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Método; envia a reprodução do quadro anterior e o pára.

### Consulte também

`prevFrame`

## MovieClip.removeMovieClip

### Disponibilidade

Flash Player 5.

### Uso

*myMovieClip.removeMovieClip()*

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Método; remove uma instância de clipe de filme criado com a ação `duplicateMovieClip` ou os métodos `duplicateMovieClip` ou `attachMovie` do objeto `MovieClip`.

### Consulte também

`MovieClip.attachMovie`, `MovieClip.attachMovie`, `removeMovieClip`,  
`MovieClip.attachMovie`

## MovieClip.\_rotation

### Disponibilidade

Flash Player 4.

### Uso

*myMovieClip.\_rotation*

### Descrição

Propriedade; especifica a rotação do clipe de filme em graus.

## MovieClip.setMask

### Disponibilidade

Flash Player 6.

### Uso

*myMovieClip.setMask (maskMovieClip)*

### Parâmetros

*myMovieClip* O nome de instância de um clipe de filme a ser mascarado.

*maskMovieClip* O nome de instância de um clipe de filme que será uma máscara.

### Retorna

Nada.

### Descrição

Método; transforma o clipe de filme do parâmetro *maskMovieClip* em uma máscara que revela o clipe de filme especificado pelo parâmetro *myMovieClip*.

O método `setMask` permite que clipes de filme com vários quadros e conteúdo complexo com diversas camadas atuem como máscaras. É possível ativar e desativar máscaras em tempo de execução. Entretanto, você não pode usar a mesma máscara para vários itens com máscara (o que é possível através do uso de camadas de máscara). Se houver fontes de dispositivo em um clipe de filme mascarado, elas serão desenhadas, mas não mascaradas. Não é possível definir um clipe de filme como sua própria máscara, por exemplo `mc.setMask(mc)`.

Se você criar uma camada de máscara que contenha um clipe de filme e, em seguida, aplicar o método `setMask` a ele, a chamada `setMask` terá prioridade e não será possível reverter isso. Por exemplo, se houver um clipe de filme em uma camada de máscara chamada `UIMask`, esta mascara outra camada que contém outro clipe de filme chamado `UIMaskee`. Se, à medida que o filme for reproduzido, você chamar `UIMask.setMask(UIMaskee)`, `UIMask` será mascarado por `UIMaskee` desse ponto em diante.

Para cancelar uma máscara criada com ActionScript, passe o valor `null` ao método `setMask`. O código a seguir cancela a máscara sem afetar a camada da máscara na Linha de tempo.

```
UIMask.setMask(null)
```

#### Exemplo

O exemplo de código a seguir usa o clipe de filme `circleMask` para mascarar o clipe de filme `theMaskee`.

```
theMaskee.setMask(circleMask);
```

## MovieClip.\_soundbuftime

#### Disponibilidade

Flash Player 6.

#### Uso

```
myMovieClip._soundbuftime
```

#### Descrição

Propriedade (global); um inteiro que especifica o número de segundos em que um som é armazenado em pré-buffer antes de começar a fluir.

## MovieClip.startDrag

#### Disponibilidade

Flash Player 5.

#### Uso

```
myMovieClip.startDrag([bloqueio, [esquerdo, superior, direito, inferior]])
```

#### Parâmetros

*bloqueio* Um valor booleano que especifica se o clipe de filme a ser arrastado está bloqueado no centro da posição do mouse (*true*) ou no ponto onde o usuário clicou pela primeira vez no clipe de filme (*false*). Este parâmetro 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 parâmetros são opcionais.

#### Retorna

Nada.

#### 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.

#### Consulte também

`MovieClip._droptarget`, `MovieClip.startDrag`, `MovieClip.stopDrag`

## MovieClip.stop

### Disponibilidade

Flash Player 5.

### Uso

```
myMovieClip.stop()
```

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Método ; pára o clipe de filme em execução no momento.

### Consulte também

`stop`

## MovieClip.stopDrag

### Disponibilidade

Flash Player 5.

### Uso

```
myMovieClip.stopDrag()
```

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Método; finaliza um método `startDrag`. Um filme que adquire a característica de arrastável com o método `startDrag`, permanece arrastável até a adição de um método `stopDrag` ou até que outro filme se torne arrastável. Somente um clipe de filme é arrastável de cada vez.

### Consulte também

`MovieClip._droptarget`, `MovieClip.startDrag`, `stopDrag`

## MovieClip.swapDepths

### Disponibilidade

Flash Player 5.

### Uso

```
myMovieClip.swapDepths(profundidade)
```

```
myMovieClip.swapDepths(destino)
```

### Parâmetros

*destino* A instância do clipe de filme cuja profundidade está sendo trocada pela instância especificada em *myMovieClip*. As duas instâncias devem ter o mesmo clipe de filme pai.

*profundidade* Um número que especifica o nível de profundidade no qual *MovieClip* deve ser colocado.

### Retorna

Nada.

### Descrição

Método; troca a ordem do empilhamento ou *z* (nível de profundidade) da instância especificada (*MovieClip*) pelo filme determinado pelo parâmetro *target* ou pelo filme que ocupa atualmente o nível de profundidade indicado no parâmetro *depth*. Os dois filmes devem ter o mesmo clipe de filme pai. Trocar o nível de profundidade 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.

### Consulte também

`_level`

## MovieClip.tabChildren

### Disponibilidade

Flash Player 6.

### Uso

*myMovieClip.tabChildren*

### Descrição

Propriedade; indefinida por padrão. Se *tabChildren* for *undefined* ou *true*, os filhos de um clipe de filme serão incluídos na ordenação automática de guias. Se o valor de *tabChildren* for *false*, os filhos de um clipe de filme não serão incluídos na ordenação automática de guias.

### Exemplo

Um dispositivo de IU de caixa de listagem criado como um clipe de filme contém vários itens. É possível clicar em cada um desses itens para selecioná-los, portanto, estes itens são botões. Todavia, somente a própria caixa de listagem é considerada uma parada de tabulação. Os itens contidos na caixa de listagem devem ser excluídos da ordenação de guias. Para fazer isso, defina a propriedade *tabChildren* da caixa de listagem como *false*.

A propriedade *tabChildren* não apresenta nenhum efeito quando a propriedade *tabIndex* é usada; ela só afeta a ordenação automática de guias.

### Consulte também

`Button.tabIndex`, `TextField.tabIndex`

## MovieClip.tabEnabled

### Disponibilidade

Flash Player 6.

### Uso

*MovieClip.tabEnabled*

### Descrição

Propriedade; pode ser definida em uma instância dos objetos MovieClip, Button ou TextField. A opção `undefined` é utilizada por padrão.

Se a propriedade `tabEnabled` for `undefined` ou `true`, o objeto será incluído na ordenação automática de guias. Se a propriedade `tabIndex` também estiver definida com um determinado valor, o objeto será incluído na ordenação personalizada de guias também. Se `tabEnabled` for `false`, o objeto não será incluído na ordenação automática de guias. No caso de um clipe de filme, se `tabEnabled` for `false`, os filhos do clipe de filme ainda poderão ser incluídos na ordenação automática de guias, a menos que a propriedade `tabChildren` também seja definida como `false`.

### Consulte também

*MovieClip.tabChildren*, *MovieClip.tabIndex*

## MovieClip.tabIndex

### Disponibilidade

Flash Player 6.

### Uso

*myMovieClip.tabIndex*

### Descrição

Propriedade; permite personalizar a ordenação de guias dos objetos em um filme. A propriedade `tabIndex` permanece indefinida por padrão. Você pode definir `tabIndex` em um botão, clipe de filme ou instância de campo de texto.

Se um objeto de um filme do Flash contém uma propriedade `tabIndex`, a ordenação automática de guias é desativada. Nesse caso, a ordenação é calculada com base nas propriedades `tabIndex` dos objetos no filme. A ordenação personalizada de guias inclui apenas os objetos que têm propriedades `tabIndex`.

A propriedade `tabIndex` deve ser um inteiro positivo. Os objetos são ordenados de acordo com suas propriedades `tabIndex`, em ordem ascendente. Um objeto com `tabIndex` de 1 precede um objeto com `tabIndex` de 2. A ordenação personalizada de guias ignora os relacionamentos hierárquicos dos objetos em um filme do Flash. Todos os objetos contidos no filme do Flash que possuem a propriedade `tabIndex` são colocados na ordenação de guias. Você não deve usar o mesmo valor de `tabIndex` para vários objetos.



## MovieClip.\_target

### Disponibilidade

Flash Player 4.

### Uso

*myMovieClip.\_target*

### Descrição

Propriedade (somente leitura); retorna o caminho de destino da instância do clipe de filme especificada no parâmetro *MovieClip*.

## MovieClip.\_totalframes

### Disponibilidade

Flash Player 4.

### Uso

*myMovieClip.\_totalframes*

### Descrição

Propriedade (somente leitura); retorna o número total de quadros na instância de clipe de filme especificada no parâmetro *MovieClip*.

## MovieClip.trackAsMenu

### Disponibilidade

Flash Player 6.

### Uso

*myMovieClip.trackAsMenu*

### Descrição

Propriedade; uma propriedade booleana que indica se outros botões ou clipes de filme podem ou não receber eventos de liberação de mouse. Permite a criação de menus. Você pode definir a propriedade *trackAsMenu* em qualquer botão ou objeto de clipe de filme. Se a propriedade *trackAsMenu* não existir, o comportamento padrão será *false*.

Você pode alterar a propriedade *trackAsMenu* a qualquer momento; o clipe de filme de botão modificado adquire imediatamente o novo comportamento.

### Consulte também

*Button.trackAsMenu*

## MovieClip.unloadMovie

### Disponibilidade

Flash Player 5.

### Uso

*myMovieClip.unloadMovie()*

### Parâmetros

Nenhum.

**Retorna**

Nada.

**Descrição**

Método; remove um clipe de filme carregado com os métodos do MovieClip `loadMovie` ou `attachMovie`.

**Consulte também**

`MovieClip.attachMovie`, `MovieClip.loadMovie`, `unloadMovie`, `unloadMovieNum`

## MovieClip.\_url

**Disponibilidade**

Flash Player 4.

**Uso**

*myMovieClip.\_url*

**Descrição**

Propriedade (somente leitura); recupera o URL do arquivo SWF do qual o clipe de filme foi descarregado.

## MovieClip.useHandCursor

**Disponibilidade**

Flash Player 6.

**Uso**

*myMovieClip.useHandCursor*

**Descrição**

Propriedade; um valor booleano que indica se o cursor em forma de mão é exibido quando um usuário rola o mouse sobre um clipe de filme de botão. O valor padrão de `useHandCursor` é `true`. Se `useHandCursor` estiver definida como `true`, o cursor em forma de mão padrão usado para botões é exibido quando um usuário rola o mouse sobre um clipe de filme de botão. Se `useHandCursor` for `false`, o cursor em forma de seta será usado.

Você pode alterar a propriedade `useHandCursor` a qualquer momento; o clipe de filme de botão modificado adquire imediatamente o comportamento do novo cursor. A propriedade `useHandCursor` pode ser lida de um objeto de protótipo.

## MovieClip.\_visible

### Disponibilidade

Flash Player 4.

### Uso

*myMovieClip.\_visible*

### Descrição

Propriedade; um valor booleano que indica se o filme especificado pelo parâmetro *MovieClip* está visível. Os cliques de filme que não são visíveis (propriedade `_visible` 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.

## MovieClip.\_width

### Disponibilidade

Flash Player 4 como uma propriedade somente leitura.

### Uso

*myMovieClip.\_width*

### Descrição

Propriedade; define e recupera a largura do clipe de filme, em pixels.

### Exemplo

O exemplo de código a seguir define a altura e a largura das propriedades de um clipe de filme quando o usuário clica com o mouse.

```
onClipEvent(mouseDown) {  
    _width=200;  
    _height=200;  
}
```

### Consulte também

*MovieClip.\_height*

## MovieClip.\_x

### Disponibilidade

Flash Player 3.

### Uso

*myMovieClip.\_x*

### Descrição

Propriedade; um inteiro que 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 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. Assim, para um clipe de filme girado 90° no sentido anti-horário, os filhos desse clipe herdam um sistema de coordenadas que é girado 90° no mesmo sentido. As coordenadas do clipe de filme referem-se à posição do ponto do registro.

### Consulte também

*MovieClip.\_xscale*, *MovieClip.\_y*, *MovieClip.\_yscale*

## MovieClip.\_xmouse

### Disponibilidade

Flash Player 5.

### Uso

*myMovieClip.\_xmouse*

### Descrição

Propriedade (somente leitura); retorna a coordenada *x* da posição do mouse.

### Consulte também

Mouse (objeto), MovieClip.\_ymouse

## MovieClip.\_xscale

### Disponibilidade

Flash Player 4.

### Uso

*myMovieClip.\_xscale*

### 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 da propriedade *\_x* e *\_y*, que são definidas em pixels. Por exemplo, se o clipe de filme pai é dimensionado em 50%, a definição da propriedade *\_x* move um objeto no clipe de filme pela metade do número de pixels, como se o filme tivesse sido definido em 100%.

### Consulte também

MovieClip.\_x, MovieClip.\_y, MovieClip.\_yscale

## MovieClip.\_y

### Disponibilidade

Flash Player 3.

### Uso

*myMovieClip.\_y*

### 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. Assim, para um clipe de filme girado 90° no sentido anti-horário, os filhos do clipe de filme herdam um sistema de coordenadas que é girado 90° no mesmo sentido. As coordenadas do clipe de filme referem-se à posição do ponto de registro.

### Consulte também

MovieClip.\_x, MovieClip.\_xscale, MovieClip.\_yscale

## MovieClip.\_ymouse

### Disponibilidade

Flash Player 5.

### Uso

`myMovieClip._ymouse`

### Descrição

Propriedade (somente leitura); indica a coordenada *y* da posição do mouse.

### Consulte também

Mouse (objeto), MovieClip.\_xmouse

## MovieClip.\_yscale

### Disponibilidade

Flash Player 4.

### Uso

`myMovieClip._yscale`

### 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%.

### Consulte também

MovieClip.\_x, MovieClip.\_xscale, MovieClip.\_y

## NaN

### Disponibilidade

Flash Player 5.

### Uso

NaN

### Descrição

Variável; uma variável predefinida com o valor IEEE 754 para NaN (Não Número).

## ne (diferente – específico de sequência de caracteres)

### Disponibilidade

Flash Player 4. Este operador foi reprovado e substituído pelo operador `!=` (diferença).

### Uso

*expression1* ne *expression2*

### Parâmetros

*expression1*, *expression2* Números, seqüências de caracteres ou variáveis.

### Retorna

Nada.

### Descrição

Operador (comparação); compara a *expression1* com a *expression2* e retorna `true` se a *expression1* não for igual à *expression2*; caso contrário, retorna `false`.

### Consulte também

`!=` (diferença)

## new

### Disponibilidade

Flash Player 5.

### Uso

novo *construtor*()

### Parâmetros

*construtor* Uma função seguida por parâmetros opcionais em parênteses. Normalmente, a função é o nome do tipo de objeto (por exemplo, `Array`, `Math`, `Number` ou `Object`) a ser criado.

### Retorna

Nada.

### Descrição

Operador; cria um novo objeto, inicialmente anônimo, e chama a função identificada pelo parâmetro *construtor*. O novo operador passa à função quaisquer parâmetros opcionais em parênteses, bem como o objeto recém-criado, que é referenciado com a palavra-chave `this`. Em seguida, a função construtora pode usar `this` para definir as variáveis do objeto.

A propriedade `prototype` da função construtora é copiada na propriedade `__proto__` do novo objeto. Como resultado, o novo objeto suporta todos os métodos e propriedades especificados no objeto `Prototype` da função construtora.

### Exemplo

O exemplo a seguir cria a função `Book` e, em seguida, usa o operador `new` para criar os objetos `book1` e `book2`.

```
function Book(nome, preço){
    this.name = nome;
    this.price = price;
}

book1 = new Book("Confederacy of Dunces", 19.95);
book2 = new Book("The Floating Opera", 10.95);
```

### Exemplo

O exemplo a seguir usa o novo operador para criar uma instância do objeto `Array` com 18 elementos:

```
golfCourse = new Array(18);
```

### Consulte também

`[]` (acesso de matriz), `{}` (inicializador de objeto)

A seção do método constructor em cada entrada do objeto.

## newline

### Disponibilidade

Flash Player 4.

### Uso

`newline`

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Constante; inclui um caractere de retorno de carro () que insere 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.

## nextFrame

### Disponibilidade

Flash 2.

### Uso

`nextFrame()`

### Parâmetros

Nenhum.

### Retorna

Nada.

**Descrição**

Ação; envia a reprodução para o próximo quadro e o encerra.

**Exemplo**

Neste exemplo, quando o usuário clica no botão, a reprodução passa para o próximo quadro e é encerrada.

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

## nextScene

**Disponibilidade**

Flash 2.

**Uso**

```
nextScene()
```

**Parâmetros**

Nenhum.

**Retorna**

Nada.

**Descrição**

Ação; envia a reprodução para o Quadro 1 da próxima cena e faz uma interrupção.

**Exemplo**

Neste exemplo, quando o usuário libera o botão, a reprodução é enviada para o Quadro1 da próxima cena.

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

**Consulte também**

prevScene

## not

**Disponibilidade**

Flash Player 4. Este operador foi substituído pelo operador ! (NOT lógico).

**Uso**

```
not expressão
```

**Parâmetros**

*expressão* Uma variável ou outra expressão que seja convertida em um valor booleano.

**Descrição**

Operador; executa uma operação NOT lógica no Flash Player 4.

**Consulte também**

! (NOT lógico)



## null

### Disponibilidade

Flash Player 5.

### Uso

`null`

### Parâmetros

Nenhum.

### Retorna

Nada.

### 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.

### Exemplo

Em um contexto numérico, `null` é avaliado como 0. É possível realizar testes de igualdade 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)

### Disponibilidade

Flash Player 4.

### Uso

`Number(expressão)`

### Parâmetros

*expressão* Uma expressão a ser convertida em um número.

### Retorna

Nada.

### Descrição

Função; converte o parâmetro *expressão* em um número e retorna um valor como a seguir:

Se *expressão* for um número, o valor de retorno será *expressão*.

Se *expressão* for um valor booleano, o valor de retorno será 1 se *expressão* for `true` ou 0 se *expressão* for `false`.

Se *expressão* for uma sequência de caracteres, a função tenta analisar *expressão* como um número decimal com um expoente inicial opcional, isto é, 1,57505e-3.

Se *expressão* for indefinida, o valor de retorno será 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.

#### **Consulte também**

Number (objeto)

## **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. Este objeto é idêntico ao objeto Number do JavaScript. No Flash MX, o objeto Number tornou-se um objeto nativo. Assim, você poderá observar uma melhora radical no desempenho.

É necessário usar um construtor durante a chamada aos métodos do objeto Number, mas não é preciso utilizá-lo durante a chamada às propriedades deste objeto. Os exemplos a seguir especificam a sintaxe para chamar os métodos e propriedades do objeto Number.

O exemplo a seguir chama o método `toString` do objeto Number, que retorna a sequência de caracteres "1234".

```
myNumber = new Number(1234);  
myNumber.toString();
```

Este exemplo chama a propriedade `MIN_VALUE` (também chamada constante) do objeto Number:

```
smallest = Number.MIN_VALUE
```

## Resumo de métodos do objeto Number

Método	Descrição
<code>Number.toString</code>	Retorna a representação da sequência de caracteres do objeto Number.
<code>Number.valueOf</code>	Retorna o valor primitivo do objeto Number.

## Resumo de propriedades do objeto Number

Propriedade	Descrição
<code>Number.MAX_VALUE</code>	Constante que representa o maior número representável (IEEE 754 de dupla precisão). Esse número é aproximadamente 1,7976931348623158e+308.
<code>Number.MIN_VALUE</code>	Constante que representa o menor número representável (IEEE 754 de dupla precisão). Esse número é aproximadamente 5e-324.
<code>Number.NaN</code>	Constante que representa o valor Não Número (NaN).
<code>Number.NEGATIVE_INFINITY</code>	Constante que representa o valor do infinito negativo.
<code>Number.POSITIVE_INFINITY</code>	Constante que representa o valor do infinito positivo. Este valor é o mesmo da variável global <code>Infinity</code> .

## Construtor do objeto Number

### Disponibilidade

Flash Player 5.

### Uso

```
myNumber = new Number(valor)
```

### Parâmetros

*valor* O valor numérico do objeto Number que está sendo criado ou um valor a ser convertido em um número.

### Retorna

Nada.

### Descrição

Construtor; cria um novo objeto Number. Você deve usar o construtor Number quando estiver utilizando 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 parâmetro em um valor primitivo.

### Exemplo

O código a seguir cria objetos `new Number`.

```
n1 = new Number(3.4);  
n2 = new Number(-10);
```

### Consulte também

Number (função)

## Number.MAX\_VALUE

### Disponibilidade

Flash Player 5.

### Uso

`Number.MAX_VALUE`

### Descrição

Propriedade; o maior número representável (IEEE 754 de dupla precisão). Esse número é aproximadamente  $1,79E+308$ .

## Number.MIN\_VALUE

### Disponibilidade

Flash Player 5.

### Uso

`Number.MIN_VALUE`

### Descrição

Propriedade; o menor número representável (IEEE 754 de dupla precisão). Esse número é aproximadamente  $5e-324$ .

## Number.NaN

### Disponibilidade

Flash Player 5.

### Uso

`Number.NaN`

### Descrição

Propriedade; o valor IEEE-754 que representa Não Número (NaN).

## Number.NEGATIVE\_INFINITY

### Disponibilidade

Flash Player 5.

### Uso

`Number.NEGATIVE_INFINITY`

### Descrição

Propriedade; retorna o valor IEEE 754 que representa o infinito negativo.

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.

## Number.POSITIVE\_INFINITY

### Disponibilidade

Flash Player 5.

### Uso

`Number.POSITIVE_INFINITY`

### 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.

## Number.toString

### Disponibilidade

Flash Player 5.

### Uso

`myNumber.toString(raiz)`

### Parâmetros

*raiz* Especifica a base numérica (de 2 a 36) a ser usada para a conversão de número em sequência de caracteres. Se você não especificar o parâmetro *raiz*, o valor padrão será 10.

### Retorna

Nada.

### Descrição

Método; retorna a representação da sequência de caracteres do objeto `Number` especificado (*myNumber*).

### Exemplo

O exemplo a seguir usa o método `Number.toString`, especificando 2 para o parâmetro *radix* e retorna uma sequência de caracteres que contém a representação binária do número 1000.

```
myNumber = new Number (1000);  
myNumber.toString(2);
```

## Number.valueOf

### Disponibilidade

Flash Player 5.

### Uso

`myNumber.valueOf()`

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Método; retorna o tipo de valor primitivo do objeto `Number` especificado.

## Object (objeto)

O objeto `Object` genérico está na raiz da hierarquia de classes do ActionScript. O objeto `Object` genérico do ActionScript contém um pequeno subconjunto de recursos fornecido pelo objeto `Object` do JavaScript. No Flash MX, o objeto `Object` tornou-se um objeto nativo. Assim, você poderá observar uma melhora radical no desempenho.

O objeto `Object` genérico é suportado no Flash Player 5.

### Resumo de métodos do objeto `Object`

Método	Descrição
<code>Object.addProperty</code>	Cria uma propriedade de apanhador/definidor em um objeto.
<code>Object.registerClass</code>	Atribui uma classe do ActionScript a uma instância de clipe de filme.
<code>Object.toString</code>	Converte o objeto especificado em uma sequência de caracteres e o retorna.
<code>Object.unwatch</code>	Remove o registro criado por um método <code>Object.watch</code> .
<code>Object.valueOf</code>	Retorna o valor primitivo do objeto <code>Object</code>
<code>Object.watch</code>	Registra uma função de chamada a ser ativada quando uma determinada propriedade de um objeto do ActionScript é alterada.

### Resumo de propriedades do objeto `Object`

Propriedade	Descrição
<code>Object.__proto__</code>	Uma referência à propriedade <code>prototype</code> da função construtora do objeto.

### Construtor do objeto `Object`

#### Disponibilidade

Flash Player 5.

#### Uso

```
new Object(valor)
```

#### Parâmetros

*valor* Um número, valor booleano ou sequência de caracteres a ser convertido em um objeto. Este parâmetro é opcional. Se você não especificar *valor*, o construtor cria um novo objeto com propriedades não definidas.

#### Descrição

Construtor; cria um novo objeto `Object`.

# Object.addProperty

## Disponibilidade

Flash Player 6.

## Uso

```
myObject.addProperty( prop, getFunc, setFunc )
```

## Parâmetros

*prop* O nome da propriedade de objeto a ser criada.

*getFunc* A função chamada para recuperar o valor da propriedade; este parâmetro é um objeto de função.

*setFunc* A função chamada para definir o valor da propriedade; este parâmetro é um objeto de função. Se você passar o valor `null` para este parâmetro, a propriedade será do tipo somente leitura.

## Retorna

Retorna um valor de `true` se a propriedade for criada com êxito; caso contrário, retorna `false`.

## Descrição

Método; cria uma propriedade de apanhador/definidor. Quando o Flash lê uma propriedade de apanhador/definidor, chama a função `get` e o valor de retorno da função torna-se um valor de *prop*. Quando o Flash grava uma propriedade de apanhador/definidor, chama a função `set` e passa a ela o novo valor como um parâmetro. Se já houver uma propriedade com o mesmo nome, a nova propriedade a substituirá.

Uma função `get` não possui nenhum parâmetro. Seu valor de retorno pode ser de qualquer tipo. Seu tipo pode ser alterado entre as chamadas. O valor de retorno é tratado como o valor atual da propriedade.

Uma função `set` utiliza um parâmetro, que é o novo valor da propriedade. Por exemplo, se a propriedade `x` for atribuída pelo comando `x = 1`, a função `set` receberá o parâmetro `1` do tipo `number`. O valor de retorno da função `setter` é ignorado.

É possível adicionar propriedades de apanhador/definidor aos objetos de protótipo. Se você adicionar uma propriedade de apanhador/definidor a um objeto de protótipo, todas as instâncias de objeto que herdam o objeto de protótipo também herdam esta propriedade. Isso torna possível adicionar uma propriedade de apanhador/definidor em um local, o objeto de protótipo, e fazer com que ela seja propagada para todas as instâncias de uma classe (muito semelhante à adição de métodos a objetos de protótipo). Se uma função `get/set` for chamada para uma propriedade de apanhador/definidor em um objeto de protótipo herdado, a referência passada à função `get/set` será o objeto referenciado originalmente, e não o objeto de protótipo.

Se chamado incorretamente, `Object.addProperty` poderá apresentar um erro. A tabela a seguir descreve os erros que podem ocorrer:

Condição de erro	O que acontece
<code>prop</code> não é um nome de propriedade válido; por exemplo, uma sequência de caracteres vazia.	Retorna <code>false</code> e a propriedade não é adicionada.
<code>getFunc</code> não é um objeto de função válido.	Retorna <code>false</code> e a propriedade não é adicionada.
<code>setFunc</code> não é um objeto de função válido.	Retorna <code>false</code> e a propriedade não é adicionada.

## Exemplo

Uso 1: As propriedades internas `TextField.scroll` e `TextField.maxscroll` são do tipo apanhador/definidor. O objeto `TextField` possui os métodos internos `getScroll`, `setScroll` e `getMaxScroll`. O construtor `TextField` cria as propriedades de apanhador/definidor e as aponta para os métodos `get/set` internos, como mostrado a seguir:

```
this.addProperty("scroll", this.getScroll, this.setScroll);
this.addProperty("maxscroll", this.getMaxScroll, null);
```

Quando um script recupera o valor de `myTextField.scroll`, o interpretador do `ActionScript` chama `myTextField.getScroll` automaticamente. Quando um script modifica o valor de `myTextField.scroll`, o interpretador chama `myTextField.setScroll`. A propriedade `maxscroll` não especifica uma função `set`, portanto, as tentativas feitas para modificar `maxscroll` são ignoradas.

Uso 2: O exemplo anterior de `TextField.scroll` e `TextField.maxscroll` funciona, mas as propriedades `scroll` e `maxscroll` são adicionadas a todas as instâncias do objeto `TextField`. Isso significa que o custo para estabelecer as propriedades é de dois slots de propriedades para cada instância do objeto. Se houver muitas propriedades como `scroll` e `maxscroll` em uma classe, é possível que elas consumam uma grande quantidade de memória. Nesse caso, você pode adicionar as propriedades `scroll` e `maxscroll` a `TextField.prototype`:

```
TextField.prototype.addProperty("scroll", this.getScroll, this.setScroll);
TextField.prototype.addProperty("maxscroll", this.getMaxScroll, null);
```

Agora, as propriedades `scroll` e `maxscroll` existem apenas em um local: o objeto `TextField.prototype`. Entretanto, o efeito é o mesmo do código anterior que adicionou `scroll` e `maxscroll` diretamente a todas as instâncias. Se `scroll` ou `maxscroll` for acessada em uma instância de `TextField`, a cadeia de protótipos será percorrida para cima e a propriedade de apanhador/definidor de `TextField.prototype` será encontrada.

## Object.\_\_proto\_\_

### Disponibilidade

Flash Player 5.

### Uso

`myObject.__proto__`

### Parâmetros

Nenhum.

### Descrição

Propriedade; refere-se à propriedade `prototype` da função construtora que criou `myObject`. A propriedade `__proto__` é atribuída automaticamente a todos os objetos durante sua criação. O interpretador do `ActionScript` usa a propriedade `__proto__` para acessar a propriedade `prototype` da função construtora do objeto e assim descobrir quais propriedades e métodos o objeto herda de sua classe.



# Object.registerClass

## Disponibilidade

Flash Player 6

## Uso

`Object.registerClass(symbolID, theClass)`

## Parâmetros

*symbolID* O identificador de vinculação do símbolo de clipe de filme ou o identificador de sequência de caracteres da classe do ActionScript.

*theClass* Uma referência à função construtora da classe do ActionScript ou `null` para cancelar o registro do símbolo.

## Retorna

Se o registro da classe for bem-sucedido, será retornado um valor de `true`; caso contrário, `false` será retornado.

## Descrição

Método; associa um símbolo de clipe de filme a uma classe de objeto do ActionScript. Se não houver um símbolo, o Flash criará uma associação entre um identificador de sequência de caracteres e uma classe de objeto.

Quando uma instância do símbolo de clipe de filme especificado for colocada pela Linha de tempo, ela será registrada na classe indicada pelo parâmetro *theClass* e não na classe `MovieClip`.

Quando uma instância do símbolo de clipe de filme especificado for criada com o método `attachMovie` ou `duplicateMovieClip`, ela será registrada na classe indicada pelo parâmetro *theClass* e não na classe `MovieClip`.

Se *theClass* for `null`, `Object.registerClass` removerá qualquer definição de classe de ActionScript associada ao símbolo de clipe de filme ou identificador de classe especificado. No caso de símbolos de clipe de filme, qualquer instância existente do clipe de filme permanecerá inalterada, mas as novas instâncias do símbolo serão associadas à classe `MovieClip` padrão.

Se um símbolo já estiver registrado em uma classe, o método `Object.registerClass` o substituirá pelo novo registro.

Quando uma instância de clipe de filme é colocada pela Linha de tempo ou criada com `attachMovie` ou `duplicateMovieClip`, ActionScript chama o construtor da classe apropriada com a palavra-chave `this` apontando para o objeto. A função construtora é chamada sem nenhum parâmetro.

Se o método `Object.registerClass` for usado para registrar um clipe de filme com uma classe ActionScript diferente de `MovieClip`, o símbolo do clipe de filme não herdar os métodos, propriedades e eventos da classe `MovieClip` incorporada, a menos que a classe `MovieClip` seja incluída na cadeia protótipo da nova classe. O código a seguir cria uma nova classe ActionScript denominada *theClass* que herda as propriedades da classe `MovieClip`:

```
theClass.prototype = new MovieClip();
```

## Exemplo

Este exemplo cria um componente para um dispositivo de IU de caixa de seleção padrão.

Primeiro crie um símbolo de clipe de filme chamado Caixa de seleção na biblioteca. Em seguida, crie uma classe `CheckBox` no `ActionScript` que representará a caixa de seleção.

```
// Define o construtor para (e assim define)
Classe CheckBox

function CheckBox() {
    ...
}

// Define que a cadeia de protótipos CheckBox
herdará de MovieClip

CheckBox.prototype = new MovieClip();

// Define os métodos da classe CheckBox

CheckBox.prototype.setLabel = function (newLabel) {
    this.label = newLabel;
    ...
};
CheckBox.prototype.setValue = function (newValue) {
    this.value = value;
    ...
};
CheckBox.prototype.getValue = function () {
    return this.value;
};
CheckBox.prototype.getLabel = function () {
    return this.label;
};
```

Agora é necessário associar a classe `CheckBox` ao símbolo de clipe de filme de Caixa de seleção. Primeiro, você precisa da capacidade de identificar o símbolo de clipe de filme de Caixa de seleção com o `ActionScript`. Para fazer isso, insira um identificador na caixa de diálogo Vinculação da biblioteca e selecione Exportar para `ActionScript`.

Em seguida, faça com que o `ActionScript` associe a classe `CheckBox` ao símbolo `CheckBox`:

```
Object.registerClass("CheckBox" /*symbolID*/, CheckBox /*theClass*/ );
```

**Uso 1 (colocação de Linha de tempo):** Agora é possível colocar instâncias de `CheckBox` no palco na ferramenta de criação, sendo que em tempo de execução, as instâncias receberão automaticamente a classe do `ActionScript` `CheckBox`. Se você inserir duas instâncias, `myCheckBox1` e `myCheckBox2`, poderá controlá-las chamando métodos, como mostrado a seguir:

```
myCheckBox1.setValue(true);
myCheckBox2.setValue(false);
myCheckBox2.setLabel("new label for #2");
```

Uso 2 (Instâncias dinâmicas): Você pode usar o método `attachMovie` para criar uma nova instância da caixa de seleção no Palco, à medida que o filme é reproduzido. Como o símbolo `CheckBox` é registrado na classe do ActionScript `CheckBox`, a nova instância dinâmica receberá essa classe automaticamente.

```
// createCheckBox é uma função assistente que
// cria CheckBoxes dinamicamente
function createCheckBox(nome, profundidade) {
    attachMovie("CheckBox", nome, profundidade);
}
createCheckBox("myCheckBox3", 100);
myCheckBox3.setValue(false);
myCheckBox3.setLabel("novo rótulo para #3");
```

#### Consulte também

`MovieClip.attachMovie`, `MovieClip.duplicateMovieClip`

## Object.toString

#### Disponibilidade

Flash Player 5.

#### Uso

```
myObject.toString()
```

#### Parâmetros

Nenhum.

#### Retorna

Nada.

#### Descrição

Método; converte o objeto especificado em uma sequência de caracteres e o retorna.

## Object.unwatch

#### Disponibilidade

Flash Player 6.

#### Uso

```
myObject.unwatch (prop)
```

#### Parâmetros

*prop* O nome da propriedade de objeto que não deverá mais ser observada, como uma sequência de caracteres.

#### Retorna

Um valor booleano.

#### Descrição

Método; remove um ponto de controle criado pelo método `Object.watch`. Este método retornará um valor de `true` se o ponto de controle tiver sido removido com êxito; caso contrário, retornará `false`.

## Object.valueOf

### Disponibilidade

Flash Player 5.

### Uso

```
myObject.valueOf()
```

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Método; retorna o valor primitivo do objeto especificado. Se o objeto não tiver um valor primitivo, o objeto é retornado.

## Object.watch

### Disponibilidade

Flash Player 6.

### Uso

```
myObject.watch( prop, callback [, userData] )
```

### Parâmetros

*prop* Uma sequência de caracteres que indica o nome da propriedade do objeto a ser observada.

*callback* A função a ser chamada quando a propriedade observada é alterada. Este parâmetro é um objeto de função e não um nome de função como uma sequência de caracteres. O formato de *callback* é `callback(prop, oldval, newval, userData)`.

*userData* Uma parte arbitrária dos dados de ActionScript passada ao método *callback*. Se o parâmetro *userData* for omitido, `undefined` será passado ao método de retorno de chamada. Este parâmetro é opcional.

### Retorna

Um valor de `true` se o ponto de controle tiver sido criado com êxito, caso contrário, retorna um valor `false`.

### Descrição

Método; registra uma função de retorno de chamada a ser ativada quando uma determinada propriedade de um objeto do ActionScript é alterada. Quando a propriedade é alterada, a função de retorno de chamada é ativada com `myObject` como o objeto recipiente.

Um ponto de controle pode filtrar (ou anular) a atribuição de valor retornando um `newval` modificado (ou `oldval`). Se você excluir uma propriedade para a qual tenha sido definido um ponto de controle, este ponto não desaparecerá. Se, posteriormente, você recriar a propriedade, o ponto de controle ainda estará valendo. Para remover o ponto de controle, use o método `Object.unwatch`.

Apenas um único ponto de controle pode ser registrado em uma propriedade. As chamadas subsequentes a `Object.watch` na mesma propriedade substituem o ponto de controle original.

O método `Object.watch` apresenta um comportamento semelhante à função `Object.watch` no Netscape JavaScript 1.2 ou posterior. A diferença principal é o parâmetro `userData`, que é uma adição do Flash a `Object.watch` não suportada pelo Netscape Navigator. Você pode passar o parâmetro `userData` à função de retorno de chamada e usá-lo nesta função.

O método `Object.watch` não pode observar propriedades de apanhador/definidor. As propriedades de apanhador/definidor funcionam através de “avaliação preguiçosa”—o valor da propriedade não é determinado até que ela seja realmente consultada. Com frequência, a “avaliação preguiçosa” é eficiente porque a propriedade não é atualizada constantemente; assim, ela é avaliada quando necessário. Entretanto, `Object.watch` precisa avaliar uma propriedade para acionar pontos de controle nela. Para funcionar com uma propriedade de apanhador/definidor, `Object.watch` precisa avaliá-la constantemente, o que é um processo ineficaz.

Geralmente, as propriedades predefinidas de `ActionScript`, como `_x`, `_y`, `_width` e `_height`, são propriedades de apanhador/definidor e, portanto, não podem ser observadas com `Object.watch`.

### Exemplo

Este exemplo mostra um componente `CheckBox` com métodos que definem o rótulo ou valor de cada instância de caixa de seleção:

```
myCheckBox1.setValue(true);
myCheckBox1.setLabel("novo rótulo");
...
```

É adequado considerar o valor e o rótulo de uma caixa de seleção como propriedades. É possível utilizar `Object.watch` para fazer com que o acesso ao valor e ao rótulo pareça com o acesso à propriedade em vez de chamada a método, como mostrado a seguir:

```
// Definir o construtor para (e, portanto, definir) a classe CheckBox
function CheckBox() {
    ...
    this.watch('value', function (id, oldval, newval)) {
        ...
    }
    this.watch('label', function (id, oldval, newval)) {
        ...
    }
}
```

Quando a propriedade do valor ou rótulo é modificada, a função especificada pelo componente é chamada para realizar as tarefas necessárias para atualização da aparência e do estado do componente, de forma a refletir seus novos parâmetros. Portanto, o comando de atribuição a seguir usa um manipulador `Object.watch` para notificar o componente de que a variável foi alterada e fazer com que ele atualize sua representação gráfica

```
myCheckBox1.value = false;
```

Esta sintaxe é mais concisa do que a anterior:

```
myCheckBox1.setValue(false);
```

### Consulte também

`Object.addProperty`, `Object.unwatch`

# onClipEvent

## Disponibilidade

Flash Player 5.

## Uso

```
onClipEvent(movieEvent){  
    comando(s);  
}
```

## Parâmetros

*movieEvent* é um dispositivo de ativação chamado *evento*. Quando o evento ocorre, são executados os comandos posteriores a ele entre chaves. Qualquer um dos valores a seguir pode ser especificado pelo parâmetro *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 disparada continuamente à taxa de quadros do filme. As ações associadas ao evento do clipe **enterFrame** 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 do mouse atual.
- **mouseDown** A ação é iniciada quando o botão esquerdo 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 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.

*comando(s)* Os comandos a serem executados quando o evento *movieEvent* ocorre.

## Descrição

Manipulador de eventos; dispara ações definidas por uma instância específica de um clipe de filme.

## Exemplo

O comando a seguir inclui o script de um arquivo externo quando o filme é exportado; as ações no script incluído são executadas quando o clipe de filme ao qual elas estão anexadas é carregado:

```
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 do objeto `Key`. O script a seguir usa o método `Key.getCode` para descobrir qual tecla foi pressionada pelo usuário; se a tecla pressionada corresponder à propriedade `Key.RIGHT`, o filme será enviado ao quadro seguinte, se corresponder à propriedade `Key.LEFT`, o filme será enviado ao quadro anterior.

```
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 sempre que ele é movido.

```
onClipEvent(mouseMove) {
    stageX=_root.xmouse;
    stageY=_root.ymouse;
}
```

### Consulte também

`Key` (objeto), `MovieClip._xmouse`, `MovieClip._ymouse`, `on`

## on

### Disponibilidade

Flash 2. Nem todos os eventos são suportados no Flash 2.

### Uso

```
on(mouseEvent) {
    comando(s);
}
```

### Parâmetros

*comando(s)* Os comandos a serem executados quando o evento *mouseEvent* ocorre.

Um *mouseEvent* é um dispositivo de ativação chamado “evento”. Quando o evento ocorre, os comandos posteriores a ele entre chaves são executados. Qualquer um dos valores a seguir pode ser especificado pelo parâmetro *movieEvent*:

- `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, depois que o botão foi pressionado enquanto o ponteiro estava dentro do botão.
- `rollOut` O ponteiro rola fora da área do botão.
- `rollOver` O ponteiro do mouse rola 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.
- `dragOver` Com o ponteiro sobre o botão, o botão do mouse é pressionado, rolado para fora do botão e, a seguir, rolado de volta sobre o botão.
- `keyPress (“tecla”)` A *tecla* especificada é pressionada. A parte *tecla* do parâmetro é especificada por qualquer código de tecla listado no Apêndice C, “Teclas do teclado e valores de códigos de teclas” de *Usando o Flash* ou qualquer uma das constantes listadas em Resumo das propriedades do objeto `Key`.

### Descrição

Manipulador de eventos; especifica o evento do mouse ou o pressionamento de tecla que dispara uma ação.

### 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) {  
    trace(_root.rabbit._y);  
    trace(_root.rabbit._x);  
    stopDrag();  
}
```

### Consulte também

`onClipEvent`

## or

### Disponibilidade

Flash 4. Este operador foi reprovado e substituído pelo operador `||` (OR lógico).

### Uso

*condition1* ou *condition2*

### Parâmetros

*condition1,2* Uma expressão que pode receber o valor `true` ou `false`.

### Retorna

Nada.

### Descrição

Operador; avalia *condition1* e *condition2* e se alguma das expressões for `true`, toda a expressão será `true`.

### Consulte também

`||` (OR lógico), `|` (OR bit a bit)

## ord

### Disponibilidade

Flash Player 4. Esta função foi reprovada e substituída pelos métodos e propriedades de `String` (objeto).

### Uso

`ord(caractere)`

### Parâmetros

*caractere* O caractere a ser convertido em um número de código ASCII.

### Retorna

Nada.



**Descrição**

Função de sequência de caracteres; converte caracteres em números de código ASCII.

**Consulte também**

String (objeto)

## **\_parent**

**Disponibilidade**

Flash Player 4.

**Uso**

`_parent.property`

`_parent._parent.property`

**Descrição**

Propriedade; especifica ou retorna uma referência ao clipe de filme ou objeto que contém o clipe de filme ou objeto atual. O objeto atual é o que contém o código ActionScript que faz referência a `_parent`. Use `_parent` para especificar um caminho relativo para clipes de filme ou objetos que estiverem acima do clipe de filme ou objeto atual.

**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 salta para o Quadro 10 na Linha de tempo do clipe de filme `classroom`.

```
_parent.gotoAndStop(10);
```

**Consulte também**

`_root`, `targetPath`

## **parseFloat**

**Disponibilidade**

Flash Player 5.

**Uso**

`parseFloat(seqüência de caracteres)`

**Parâmetros**

*seqüência de caracteres* A seqüência de caracteres a ser lida e convertida em um número de ponto flutuante.

**Retorna**

Nada.

**Descrição**

Função; converte uma seqüência de caracteres em um número de ponto flutuante. A função lê ou "analisa" e retorna os números em uma seqüência de caracteres até alcançar um caractere que não seja parte do número inicial. Se a seqüência de caracteres não começar com um número que possa ser analisado, `parseFloat` retornará NaN. O espaço em branco que precede os inteiros válidos é ignorado, pois são caracteres precedentes não numéricos.

### Exemplo

O exemplo a seguir usa a função `parseFloat` para avaliar vários tipos de números.

```
parseFloat("-2") retorna -2
parseFloat("2.5") retorna 2.5
parseFloat("3.5e6") retorna 3.5e6, or 3500000
parseFloat("foobar") retorna NaN
parseFloat(" 5.1") retorna 5.1
parseFloat("3.75math") retorna 3.75
parseFloat("0garbage") retorna 0
```

## parseInt

### Disponibilidade

Flash Player 5.

### Uso

```
parseInt(expressão, [raiz])
```

### Parâmetros

*expressão* Uma seqüência de caracteres a ser convertida em um inteiro.

*raiz* Um inteiro que representa a raiz (base) do número a ser analisado. Os valores permitidos vão de 2 a 36. Este parâmetro é opcional.

### Retorna

Nada.

### Descrição

Função; converte uma seqüência de caracteres em um inteiro. Se não for possível converter a seqüência de caracteres especificada nos parâmetros em um número, a função retornará NaN. Os inteiros que começam com 0 ou que especificam uma raiz de 8 são interpretados como números octais. As seqüências de caracteres que começam com 0x são interpretadas como números hexadecimais. O espaço em branco que precede os inteiros válidos é ignorado, pois são caracteres precedentes não numéricos.

### Exemplo

Os exemplos a seguir usam a função `parseInt` para avaliar vários tipos de números.

```
parseInt("3.5")
// retorna 3.5

parseInt("barra")
// retorna NaN

parseInt("4foo")
// retorna 4
```

A seguir são mostrados exemplos de conversões hexadecimais:

```
parseInt("0x3F8")  
// retorna 1016
```

```
parseInt("3E8", 16)  
// retorna 1000
```

A seguir são mostrados exemplos de uma conversão binária:

```
parseInt("1010", 2)  
// retorna 10 (a representação decimal do binário 1010)
```

A seguir é mostrado um exemplo de análise de número octal (neste caso, o número octal é identificado pela raiz, 8):

```
parseInt("777", 8)  
// retorna 511 (a representação decimal do octal 777)
```

## play

### Disponibilidade

Flash 2.

### Uso

```
play()
```

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Ação; move a reprodução para frente na Linha de tempo.

### 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="Você não é Steve!";  
    }  
}
```

## prevFrame

### Disponibilidade

Flash 2.

### Uso

```
prevFrame()
```

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Ação; envia a reprodução para o quadro anterior e o encerra. Se o quadro atual for 1, a reprodução não será movida.

### Exemplo

Quando o usuário clica em um botão que tem o manipulador a seguir anexado, a reprodução é enviada ao quadro anterior.

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

### Consulte também

MovieClip.prevFrame

## prevScene

### Disponibilidade

Flash 2.

### Uso

```
prevScene()
```

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Ação; envia a reprodução para o Quadro 1 da cena anterior e faz uma ininterrupção.

### Consulte também

nextScene

# print

## Disponibilidade

Flash Player 4.20.

## Uso

```
print (nível)
print (nível, "Caixa delimitadora")
print ("destino")
print ("destino", "Caixa delimitadora")
printAsBitmap (nível)
printAsBitmap (nível, "Caixa delimitadora")
printAsBitmap ("destino")
printAsBitmap ("destino", "Caixa delimitadora")
```

## Parâmetros

*print* No modo normal do painel Ações, escolha Como vetores para imprimir quadros que não contenham imagens de bitmap nem usem transparência (alfa) ou efeitos de cores; escolha Como bitmap para imprimir quadros que contenham imagens de bitmap, transparência ou efeitos de cores. Se você escolher o parâmetro de impressão Como bitmap, a sintaxe AsBitmap será anexada à ação print no painel Ações.

*nível* O nível a ser impresso no Flash Player. Se você escolher um nível no modo normal do painel Ações, a ação print alternará para printNum ou printAsBitmapNum; no modo especialista, é necessário especificar printNum ou printAsBitmapNum. Por padrão, todos os quadros do nível são impressos. Se você quiser imprimir quadros específicos do nível, atribua um rótulo de quadro #p aos quadros desejados.

*destino* O nome da instância do clipe de filme a ser impresso. Por padrão, todos os quadros na instância de destino são impressos. Se você quiser imprimir quadros específicos do clipe de filme, atribua um rótulo de quadro #p a esses quadros.

*Caixa delimitadora* Um modificador que define a área de impressão do filme. Este parâmetro é opcional. Você pode escolher um dos itens a seguir:

- *bmovie* Indica a caixa delimitadora de um quadro específico em um filme como a área de impressão de todos os quadros imprimíveis no filme. Atribua um rótulo de quadro #b ao 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 parâmetro 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.

## Retorna

Nenhum.

## Descrição

Ação; imprime o clipe de filme de *destino* de acordo com os limites especificados no parâmetro (bmovie, bmax ou bframe). Para imprimir quadros específicos do filme de destino, é necessário anexar um rótulo de quadro #P a esses quadros. Embora a ação `print` resulte em impressões de mais qualidade do que a ação `printAsBitmap`, não é possível usá-la para imprimir filmes que utilizem transparências alfa ou efeitos especiais de cores.

Se você não especificar um parâmetro de área de impressão, ela será 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 parâmetros 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.

## Exemplo

O exemplo a seguir imprimirá 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 imprimirá todos os quadros imprimíveis em `myMovie` com uma área de impressão definida pela caixa delimitadora de cada quadro:

```
print("myMovie", "bframe");
```

## Consulte também

`printNum`, `printAsBitmap`, `printAsBitmapNum`

# printAsBitmap

## Disponibilidade

Flash Player 4.20.

## Uso

```
printAsBitmap(destino, "Caixa delimitadora")
```

## Parâmetros

*destino* O nome da instância do clipe de filme a ser impresso. Por padrão, todos os quadros do filme são impressos. Para imprimir quadros específicos do filme, é necessário anexar um rótulo de quadro #P a esses quadros.

*Caixa delimitadora* Um modificador que define a área de impressão do filme. Você pode escolher um dos seguintes parâmetros:

- **bmovie** Indica a caixa delimitadora de um quadro específico em um filme como a área de impressão de todos os quadros imprimíveis no filme. Atribua um rótulo de quadro #b ao 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 parâmetro **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.

## Retorna

Nenhum.

## Descrição

Ação; imprime o clipe de filme de *destino* como um bitmap. Use a ação `printAsBitmap` para imprimir filmes que contenham quadros com objetos que usem transparência ou efeitos de cor. A ação `printAsBitmap` imprime na resolução mais alta disponível da impressora para manter a maior definição e qualidade possível.

Se o seu filme não contiver transparências alfa ou efeitos de cor, use a ação `print` para obter resultados de melhor qualidade.

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 parâmetros `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.

## Consulte também

`print`, `printAsBitmapNum`, `printNum`

# printAsBitmapNum

## Disponibilidade

Flash Player 5.

## Uso

```
printAsBitmapNum(nível)  
printAsBitmapNum(nível, "Caixa delimitadora")
```

## Parâmetros

*nível* O nível a ser impresso no Flash Player. Por padrão, todos os quadros do nível são impressos. Se você quiser imprimir quadros específicos do nível, atribua um rótulo de quadro `#p` aos quadros desejados.

*Caixa delimitadora* Um modificador que define a área de impressão do filme. Este parâmetro é opcional. Você pode escolher um dos seguintes parâmetros:

- `bmovie` Indica a caixa delimitadora de um quadro específico em um filme como a área de impressão de todos os quadros imprimíveis no filme. Atribua um rótulo de quadro `#b` ao 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 parâmetro `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.

## Retorna

Nenhum.

## Descrição

Ação; imprime um nível no Flash Player como um bitmap. Use a ação `printAsBitmapNum` para imprimir filmes que contenham quadros com objetos que usem transparência ou efeitos de cor. A ação `printAsBitmapNum` imprime na resolução mais alta disponível da impressora para manter a melhor 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 alfa ou efeitos de cor, use a ação `printNum` para obter resultados de melhor qualidade.

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 parâmetros `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.

## Consulte também

`print`, `printAsBitmap`, `printNum`

# printNum

## Disponibilidade

Flash Player 5.

## Uso

```
printNum (nível, "Caixa delimitadora")
```

## Parâmetros

**nível** O nível a ser impresso no Flash Player. Por padrão, todos os quadros do nível são impressos. Se você quiser imprimir quadros específicos do nível, atribua um rótulo de quadro `#p` aos quadros desejados.

**Caixa delimitadora** Um modificador que define a área de impressão do filme. Você pode escolher um dos seguintes parâmetros:

- **bmovie** Indica a caixa delimitadora de um quadro específico em um filme como a área de impressão de todos os quadros imprimíveis no filme. Atribua um rótulo de quadro `#b` ao 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 parâmetro `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.



## Retorna

Nenhum.

## Descrição

Ação; imprime o nível no Flash Player conforme os limites especificados no parâmetro *Caixa delimitadora* ("bmovie", "bmax", "bframe"). Para imprimir quadros específicos do filme de destino, é necessário anexar um rótulo de quadro #P a esses quadros. Embora a ação `printNum` ofereça impressões de qualidade superior às da ação `printAsBitmapNum`, não é possível usar `printNum` para imprimir filmes com transparências alfa ou efeitos de cor especiais.

Se você não especificar um parâmetro de área de impressão, ela será 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 parâmetros `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.

## Consulte também

`print`, `printAsBitmap`, `printAsBitmapNum`

## **\_quality**

### Disponibilidade

Flash Player 5.

### Uso

`_quality`

### Descrição

Propriedade (global); define ou recupera a qualidade usada para um filme. As fontes de dispositivo são sempre serrilhadas, sendo assim não são afetadas pela propriedade `_quality`.

A propriedade `_quality` pode ser definida nos seguintes valores:

- "LOW" Qualidade baixa. Os gráficos não são apresentados sem serrilhado, os bitmaps não são suavizados.
- "MEDIUM" Qualidade média. Os gráficos são apresentados sem serrilhado usando uma grade de 2 x 2, em pixels, 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 de 4 x 4, em pixels, e os bitmaps são suavizados quando o filme é estático. Essa é a configuração de qualidade padrão usada pelo Flash.
- "BEST" Qualidade muito alta. Os gráficos são apresentados sem serrilhado usando uma grade de 4 x 4, em pixels, e os bitmaps sempre são suavizados.

### Exemplo

O exemplo a seguir define a qualidade como LOW:

```
_quality = "LOW";
```

### Consulte também

`_highquality`, `toggleHighQuality`

## random

### Disponibilidade

Flash Player 4. Esta função está obsoleta no Flash 5; é recomendável usar o método `Math.random`.

### Uso

```
random(valor)
```

### Parâmetros

*valor* Um inteiro.

### Retorna

Um inteiro.

### Descrição

Função; retorna um inteiro aleatório entre 0 e um a menos que o inteiro especificado no parâmetro *valor*.

### Exemplo

O seguinte uso de `random` retorna um valor de 0, 1, 2, 3 ou 4:

```
random(5);
```

### Consulte também

`Math.random`

## removeMovieClip

### Disponibilidade

Flash Player 4.

### Uso

```
removeMovieClip(destino)
```

### Parâmetros

*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 `duplicateMovieClip` do objeto `MovieClip`.

### Retorna

Nenhum.

### Descrição

Ação; exclui uma instância de clipe de filme criada com os métodos `attachMovie` ou `duplicateMovieClip` do objeto `MovieClip`, ou com a ação `duplicateMovieClip`.

### Consulte também

`duplicateMovieClip`, `MovieClip.duplicateMovieClip`, `MovieClip.attachMovie`, `MovieClip.removeMovieClip`

## return

### Disponibilidade

Flash Player 5.

### Uso

```
return[expressão]  
return
```

### Parâmetros

*expressão* Uma sequência de caracteres, um número, uma matriz ou um objeto a ser avaliado e retornado como um valor da função. Este parâmetro é opcional.

### Retorna

O parâmetro avaliado *expressão*, se fornecido.

### Descrição

Ação; especifica o valor retornado pela função. A ação `return` avalia *expressão* e retorna o resultado como o valor da função em que é executada. A ação `return` faz com que a função pare de ser executada e a substitui pelo valor retornado. Se o comando `return` for usado isoladamente, retornará `null`.

### Exemplo

O exemplo a seguir usa a ação `return` dentro do corpo da função `sum` para retornar o valor adicionado dos três parâmetros. A próxima linha de código chama a função `sum` e atribui o valor retornado à variável `newValue`:

```
function sum(a, b, c){  
    return a + b + c;  
}  
  
newValue = sum(4, 32, 78);  
trace(newValue);  
// envia 114 à janela Saída
```

### Consulte também

`function`

## \_root

### Disponibilidade

Flash Player 4.

### Uso

```
_root.movieClip  
_root.action  
_root.property
```

### Parâmetros

*movieClip* O nome da instância de um clipe de filme.

*action* Uma ação ou método.

*property* Uma propriedade do objeto `MovieClip`.

### 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`, será retornado `_level1`.

Especificar `_root` é o mesmo que usar a notação de barra (/) para especificar um caminho absoluto dentro do nível atual.

### Exemplo

O exemplo a seguir interrompe a Linha de tempo do nível que contém o script sendo executado no momento:

```
_root.stop();
```

O exemplo a seguir envia a Linha de tempo no nível atual para o quadro 3:

```
_root.gotoAndStop(3);
```

### Consulte também

`_parent`, `targetPath`

## scroll

### Disponibilidade

Flash Player 4.

### Uso

```
textFieldVariableName.scroll = x
```

### Descrição

Propriedade; uma propriedade obsoleta que 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.

### Exemplo

O código a seguir é anexado a um botão Para cima que rola o campo de texto `myText`:

```
on (release) {  
    myText.scroll = myText.scroll + 1;  
}
```

### Consulte também

`TextField.maxscroll`, `TextField.scroll`

## Selection (objeto)

O objeto `Selection` permite definir e controlar em que campo de texto o cursor é localizado em um filme do Flash. O campo de texto que supostamente está em “foco” é aquele em que o cursor está atualmente localizado. Os índices do intervalo de seleção são baseados em zero (por exemplo, a primeira posição é 0, a segunda é 1 e assim por diante).

Não há nenhum método construtor para o objeto `Selection`, pois só pode haver um campo focalizado por vez.

## Resumo de métodos do objeto Selection

Método	Descrição
<code>Selection.addListener</code>	Registra um objeto para receber notificação quando o método <code>onSetFocus</code> é chamado.
<code>Selection.getBeginIndex</code>	Retorna o índice no começo do intervalo da seleção. Retorna -1 se não houver índice ou campo selecionado no momento.
<code>Selection.getCaretIndex</code>	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.
<code>Selection.getEndIndex</code>	Retorna o índice no final do intervalo de seleção. Retorna -1 se não houver índice ou campo selecionado no momento.
<code>Selection.getFocus</code>	Retorna o nome da variável do campo de texto em foco no momento. Retorna <code>null</code> caso não haja campo de texto em foco no momento.
<code>Selection.removeListener</code>	Remove um objeto que foi registrado com <code>addListener</code> .
<code>Selection.setFocus</code>	Põe em foco o campo de texto associado à variável especificada no parâmetro.
<code>Selection.setSelection</code>	Define os índices de início e de fim do intervalo de seleção.

## Resumo de ouvintes do objeto Selection

Método	Descrição
<code>Selection.onSetFocus</code>	Notificado quando o foco de entrada é alterado.

## Selection.addListener

### Disponibilidade

Flash Player 6.

### Uso

```
Selection.addListener(novo_Ouvinte)
```

### Parâmetros

*novo\_Ouvinte* Objeto com um método `onSetFocus`.

### Retorna

Nenhum.

### Descrição

Método; registra um objeto para receber notificações de alteração de foco do teclado. Quando o foco é alterado (por exemplo, sempre que o método `Selection.SetFocus` é chamado), todos os objetos ouvintes registrados com `addListener` têm seu método `onSetFocus` chamado. Vários objetos podem ouvir notificações de alteração de foco. Se o ouvinte *newListener* já estiver registrado, nenhuma alteração ocorrerá.

## Selection.getBeginIndex

### Disponibilidade

Flash Player 5.

### Uso

```
Selection.getBeginIndex()
```

### Parâmetros

Nenhum.

### Retorna

Um inteiro.

### Descrição

Método; retorna o índice no início do intervalo de seleção. Se nenhum índice existir ou nenhum campo de texto estiver em foco no momento, o método retornará -1. Os índices do intervalo de seleção são baseados em zero (por exemplo, a primeira posição é 0, a segunda é 1 e assim por diante).

## Selection.getCaretIndex

### Disponibilidade

Flash Player 5.

### Uso

```
Selection.getCaretIndex()
```

### Parâmetros

Nenhum.

### Retorna

Um inteiro.

### Descrição

Método; retorna o índice da posição do cursor intermitente. Se nenhum cursor intermitente for exibido, o método retornará -1. Os índices do intervalo de seleção são baseados em zero (por exemplo, a primeira posição é 0, a segunda é 1 e assim por diante).

## Selection.getEndIndex

### Disponibilidade

Flash Player 5.

### Uso

```
Selection.getEndIndex()
```

### Parâmetros

Nenhum.

### Retorna

Um inteiro.

### Descrição

Método; retorna o índice final do intervalo de seleção focalizado no momento. Se nenhum índice existir ou nenhum intervalo de seleção estiver em foco no momento, o método retornará -1. Os índices do intervalo de seleção são baseados em zero (por exemplo, a primeira posição é 0, a segunda é 1 e assim por diante).

## Selection.setFocus

### Disponibilidade

Flash Player 5. Os nomes de instância para botões e campos de texto funcionam no Flash Player 6.

### Uso

```
Selection.setFocus()
```

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Método; retorna o nome da variável do campo de texto que está evidenciado. Se nenhum campo de texto estiver evidenciado, o método retornará `null`. Se o foco atual for um botão que seja uma instância do objeto `Button`, `getFocus` retornará o caminho de destino como uma seqüência de caracteres. Se o foco atual for um campo de texto que seja uma instância do objeto `TextField`, `getFocus` retornará o caminho de destino como uma seqüência de caracteres.

Se um clipe de filme com botões for o botão atualmente em foco, `Selection.setFocus` retornará o caminho de destino do clipe de filme com botões. Se um campo de texto com um nome de instância estiver atualmente em foco, `Selection.setFocus` retornará o caminho de destino do objeto `TextField`. Caso contrário, retornará o nome da variável do campo de texto.

## Selection.onSetFocus

### Disponibilidade

Flash Player 6.

### Uso

```
someListener.onSetFocus = function(oldFocus, newFocus){  
comandos;  
}
```

### Descrição

Ouvinte; notificado quando o foco de entrada é alterado. Para usar `onSetFocus`, crie um objeto ouvinte. Em seguida, é possível definir uma função para `onSetFocus` e usar o método `addListener` para registrar o ouvinte com o objeto `Selection`, como a seguir:

```
someListener = new Object();  
someListener.onSetFocus = function () { ... };  
Selection.addListener(someListener);
```

Os ouvintes permitem a cooperação de partes diferentes de código. Isso ocorre porque vários ouvintes podem receber notificações sobre um único evento.

### Consulte também

```
Selection.addListener
```

## Selection.removeListener

### Disponibilidade

Flash Player 6.

### Uso

```
Selection.removeListener(ouvinte)
```

### Parâmetros

*ouvinte* O objeto que deixará de receber notificações de foco.

### Retorna

Se o *ouvinte* tiver sido removido com êxito, o método retornará um valor `true`. Se o *ouvinte* não tiver sido removido com êxito, por exemplo, caso o *ouvinte* não estivesse na lista de ouvintes do objeto Selection, o método retornará um valor `false`.

### Descrição

Método; remove um objeto anteriormente registrado com `addListener`.

## Selection.setFocus

### Disponibilidade

Flash Player 5. Os nomes de instância para botões e campos de texto funcionam apenas no Flash Player 6.

### Uso

```
Selection.setFocus("caminho_variável")
```

### Parâmetros

*caminho\_variável* Uma sequência de caracteres especificando o caminho do nome de uma variável associada a um campo de texto.

### Retorna

Um evento.

### Descrição

Método; põe em foco o campo de texto editável associado à variável especificada pelo *caminho\_variável*. O parâmetro *caminho\_variável* deve ser uma sequência de caracteres literal do caminho dessa variável. É possível usar a notação de ponto ou barra para especificar o caminho. Também é possível usar um caminho relativo ou absoluto.

Se um caminho de destino de uma instância de botão for passado como o parâmetro *caminho\_variável*, esse botão se tornará o novo foco. Se um caminho de destino de uma instância de campo de texto for passado como o parâmetro *caminho\_variável*, esse campo de texto se tornará o novo foco. Se `null` for passado, o foco atual será removido.

Se um clipe de filme com botões for passado para `Selection.setFocus`, ele se tornará o botão atualmente em foco. Se um objeto `TextField` for especificado, ele se tornará o foco atual. Se um objeto `Button` for especificado, ele se tornará o botão atualmente em foco.



### Exemplo

O exemplo a seguir põe em foco um campo de texto na Linha de tempo principal que é associado à variável *myVar*. O parâmetro *caminho\_variável* é um caminho absoluto; portanto, é possível chamar a ação de qualquer Linha de tempo.

```
Selection.setFocus("_root.myVar");
```

No exemplo a seguir, o campo de texto associado a *myVar* está em um clipe de filme chamado *myClip* na Linha de tempo principal. É possível usar um dos caminhos a seguir para definir o foco. O primeiro é relativo e o segundo é absoluto.

```
Selection.setFocus("myClip.myVar");  
Selection.setFocus("_root.myClip.myVar");
```

## Selection.setSelection

### Disponibilidade

Flash Player 5.

### Uso

```
Selection.setSelection(início, fim)
```

### Parâmetros

*início* O índice inicial do intervalo de seleção.

*fim* O índice final do intervalo de seleção.

### Retorna

Nada.

### Descrição

Método; define o intervalo de seleção do campo de texto focalizado no momento. O novo intervalo de seleção começará no índice especificado no parâmetro *início* e terminará no índice especificado no parâmetro *fim*. Os índices do intervalo de seleção são baseados em zero (por exemplo, a primeira posição é 0, a segunda é 1 e assim por diante). Esse método não tem efeito se não houver campo de texto focalizado no momento.

## set variable

### Disponibilidade

Flash Player 4.

### Uso

```
set(variável, expressão)
```

### Parâmetros

*variável* Um identificador para conter o valor do parâmetro *expressão*.

*expressão* Um valor atribuído à variável.

### Retorna

Nada.

## Descrição

Ação; atribui um valor a uma variável. Uma *variável* é um recipiente que contém dados. 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 qualquer tipo de dado (por exemplo, sequência de caracteres, número, Booleano, objeto ou clipe de filme). A Linha de tempo de cada filme e clipe de filme possui seu próprio conjunto de variáveis, e cada variável possui seu próprio valor, independentemente das variáveis em outras linhas de tempo.

ActionScript é uma linguagem com tipos dinâmicos. Cada variável tem um tipo. O tipo é atribuído no tempo de execução e pode mudar durante a execução. Isso é diferente de uma linguagem com tipos estáticos, como Java ou C++, onde o tipo é atribuído no tempo de compilação e não pode ser alterado no tempo de execução.

## 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(orig_x_pos, getProperty ("ship", _x ));  
}
```

O código anterior fornece o mesmo resultado que o código a seguir:

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

## Consulte também

`var`, `call`

# setInterval

## Disponibilidade

Flash Player 6.

## Uso

```
setInterval( função, intervalo[, arg1, arg2, ..., argn] )  
setInterval( objeto, nome_do_método, intervalo[, arg1, arg2, ..., argn] )
```

## Parâmetros

*função* Um nome de função ou uma referência a uma função anônima.

*objeto* Um objeto derivado do objeto `Object`.

*nome\_do\_método* O nome do método para chamar o parâmetro *objeto*.

*intervalo* O tempo em milissegundos entre chamadas para o parâmetro *função* ou *nome\_do\_método*.

*arg1, arg2, ..., argn* Parâmetros opcionais passados para o parâmetro *função* ou *nome\_do\_método*.

## Retorna

Um identificador de intervalo que pode ser passado para `clearInterval` a fim de cancelar o intervalo.

## Descrição

Ação; chama uma função, um método ou um objeto em intervalos periódicos enquanto um filme é reproduzido. É possível usar uma função de intervalo para atualizar variáveis de um banco de dados ou atualizar uma exibição de tempo.

Se *intervalo* for menor que a taxa de quadros do filme (por exemplo, 10 quadros por segundo (fps) equivalem a 100 milissegundos), a função de intervalo será chamada o mais próximo possível de *intervalo*. Use a função `updateAfterEvent` para certificar-se de que a tela seja atualizada com a frequência necessária. Se *intervalo* for maior que a taxa de quadros do filme, a função de intervalo só será chamada quando a reprodução entrar em um quadro para minimizar o impacto sempre que a tela for atualizada.

O exemplo da primeira sintaxe acima é a sintaxe padrão para a função `setInterval` no painel Ações no modo Normal. Para usar o exemplo da segunda sintaxe, use o painel Ações no modo Especialista.

## Exemplo

Uso 1: o exemplo a seguir chama uma função anônima a cada 1.000 milissegundos (a cada 1 segundo).

```
setInterval( function(){ trace("interval called"); }, 1000 );
```

Uso 2: o exemplo a seguir define duas funções de retorno de chamada e chama cada uma delas. Ambas as chamadas da função `setInterval` enviam a sequência de caracteres "interval called" para a janela Saída a cada 1.000 milissegundos. A primeira chamada de `setInterval` chama a função `callback1` que contém uma ação `trace`. A segunda chamada de `setInterval` passa a sequência de caracteres "interval called" para a função `callback2` como um parâmetro.

```
function callback1() {  
    trace("interval chamado");  
}  
  
function callback2(arg) {  
    trace(arg);  
}  
  
setInterval( callback1, 1000 );  
setInterval( callback2, 1000, "interval called" );
```

Uso 3: este exemplo usa um método de um objeto. Use esta sintaxe quando quiser chamar um método que seja definido para um objeto. Só é possível usar esta sintaxe no modo Especialista.

```
obj = new Object();  
obj.interval = function() {  
    trace("interval function called");  
}  
  
setInterval( obj, "interval", 1000 );  
  
obj2 = new Object();  
obj2.interval = function(s) {  
    trace(s);  
}  
setInterval( obj2, "interval", 1000, "interval function called" );
```

Use a segunda forma da sintaxe `setInterval` para chamar um método de um objeto, como a seguir:

```
setInterval( obj2, "interval", 1000, "interval function called" );
```

**Consulte também**

`clearInterval`, `updateAfterEvent`

## setProperty

**Disponibilidade**

Flash Player 4.

**Uso**

```
setProperty("destino", propriedade, valor/expressão)
```

**Parâmetros**

*destino* O caminho para o nome da instância do clipe de filme cuja propriedade será definida.

*propriedade* A propriedade que será definida.

*valor* O novo valor literal da propriedade.

*expressão* Uma equação que é avaliada como o novo valor da propriedade.

**Retorna**

Nada.

**Descrição**

Ação; altera o valor da propriedade de um clipe de filme enquanto o filme é reproduzido.

**Exemplo**

Este comando define a propriedade `_alpha` de um clipe de filme chamado `star` como 30% quando o botão é clicado:

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

**Consulte também**

`getProperty`

## Sound (objeto)

O objeto `Sound` permite controlar o som em um filme. É possível adicionar sons a um clipe de filme da Biblioteca enquanto o filme está sendo reproduzido e controlar esses sons. Se não for especificado um *destino* durante a criação de um novo objeto `Sound`, será possível usar os métodos para controlar o som de todo o filme. 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 no Flash Player 5 e no Flash Player 6.

## Resumo de métodos do objeto Sound

Método	Descrição
<code>Sound.attachSound</code>	Anexa o som especificado no parâmetro.
<code>Sound.getBytesLoaded</code>	Retorna o número de bytes carregados para o som especificado.
<code>Sound.getBytesTotal</code>	Retorna o tamanho do som em bytes.
<code>Sound.getPan</code>	Retorna o valor da chamada <code>setPan</code> anterior.
<code>Sound.getTransform</code>	Retorna o valor da chamada <code>setTransform</code> anterior.
<code>Sound.getVolume</code>	Retorna o valor da chamada <code>setVolume</code> anterior.
<code>Sound.loadSound</code>	Carrega um arquivo MP3 no Flash Player.
<code>Sound.setPan</code>	Define a distribuição esquerda/direita do som.
<code>Sound.setTransform</code>	Define a quantidade de cada canal, esquerdo e direito, a ser reproduzido em cada alto-falante.
<code>Sound.setVolume</code>	Define o nível de volume de um som.
<code>Sound.start</code>	Começa a reproduzir um som desde o início ou, opcionalmente, de um ponto de deslocamento definido no parâmetro.
<code>Sound.stop</code>	Pára o som especificado ou todos os sons em reprodução no momento.

## Resumo das propriedades do objeto Sound

Método	Descrição
<code>Sound.duration</code>	Tamanho de um som em milissegundos.
<code>Sound.position</code>	Número de milissegundos em que o som foi reproduzido.

## Resumo de identificadores de eventos do objeto Sound

Método	Descrição
<code>Sound.onLoad</code>	Chamado quando um som é carregado.
<code>Sound.onSoundComplete</code>	Chamado quando a reprodução de um som é interrompida.

## Construtor do objeto Sound

### Disponibilidade

Flash Player 5.

### Uso

```
new Sound([destino])
```

### Parâmetros

*destino* A instância de clipe de filme em que o objeto Sound opera. Este parâmetro é opcional.

### Retorna

Nada.

### Descrição

Construtor; cria um novo objeto Sound para um clipe de filme especificado. Se não for especificada uma instância de destino, o objeto Sound controlará todos os sons do filme.

### Exemplo

O exemplo a seguir cria uma nova instância do objeto Sound chamada `GlobalSound`. A segunda linha chama o método `setVolume` e ajusta o volume de todos os sons do filme em 50%.

```
globalsound = new Sound();  
globalsound.setVolume(50);
```

O exemplo a seguir cria uma nova instância do objeto Sound, passa para ela o clipe de filme de destino `meu_Filme` e chama o método `start` que inicia qualquer som em `meu_Filme`.

```
moviesound = new Sound(meu_Filme);  
moviesound.start();
```

## Sound.attachSound

### Disponibilidade

Flash Player 5.

### Uso

```
mySound.attachSound("Nome_id")
```

### Parâmetros

*Nome\_id* O identificador de um som exportado na Biblioteca. O identificador está localizado na caixa de diálogo Propriedades de vinculação de símbolo.

### Retorna

Nada.

### Descrição

Método; anexa o som especificado no parâmetro *Nome\_id* 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.

### Consulte também

`Sound.start`

## Sound.duration

### Disponibilidade

Flash Player 6.

### Uso

```
mySound.duration
```

### Descrição

Propriedade (somente leitura); a duração de um som em milissegundos.

## Sound.getBytesLoaded

### Disponibilidade

Flash Player 6.

### Uso

*Sound.getBytesLoaded()*

### Parâmetros

Nenhum.

### Retorna

Um inteiro que indica o número de bytes carregados.

### Descrição

Método; retorna o número de bytes carregados (transmitidos) do objeto Sound especificado. É possível comparar o valor de `getBytesLoaded` ao valor de `getBytesTotal` para determinar que porcentagem de um som foi carregada.

### Consulte também

`Sound.getBytesTotal`

## Sound.getBytesTotal

### Disponibilidade

Flash Player 6.

### Uso

*Sound.getBytesTotal()*

### Parâmetros

Nenhum.

### Retorna

Um inteiro indicando o tamanho total, em bytes, do objeto Sound especificado.

### Descrição

Método; retorna o tamanho, em bytes, do objeto Sound especificado.

### Consulte também

`Sound.getBytesLoaded`

## Sound.getPan

### Disponibilidade

Flash Player 5.

### Uso

*mySound.getPan();*

### Parâmetros

Nenhum.

**Retorna**

Nada.

**Descrição**

Método; retorna o nível de pan definido na última chamada `setPan` como um inteiro de -100 (esquerda) a 100 (direita). (0 define os canais esquerdo e direito igualmente.) 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`.

**Consulte também**

`Sound.setPan`

## Sound.getTransform

**Disponibilidade**

Flash Player 5.

**Uso**

```
mySound.getTransform();
```

**Parâmetros**

Nenhum.

**Retorna**

Nada.

**Descrição**

Método; retorna as informações de transformação do som do objeto `Sound` especificado na última chamada `setTransform`.

**Consulte também**

`Sound.setTransform`

## Sound.getVolume

**Disponibilidade**

Flash Player 5.

**Uso**

```
mySound.getVolume()
```

**Parâmetros**

Nenhum.

**Retorna**

Nada.

**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.

**Consulte também**

`Sound.setVolume`



## Sound.loadSound

### Disponibilidade

Flash Player 6.

### Uso

```
mySound.loadSound("url", isStreaming)
```

### Parâmetros

*url* Local de um arquivo de som MP3 no servidor.

*isStreaming* Valor booleano que indica se o som é um evento ou um fluxo de som.

### Retorna

Nada.

### Descrição

Método; carrega um arquivo MP3 em uma instância do objeto Sound. É possível usar o parâmetro *isStreaming* para indicar se o som é um evento ou um fluxo de som.

Os eventos de som são totalmente carregados antes de serem reproduzidos. Eles são gerenciados pelo objeto Sound do ActionScript e respondem a todos os métodos e propriedades desse objeto.

Os fluxos de som são reproduzidos durante o download. A reprodução começa após o recebimento de dados suficientes para iniciar o descompactador. Assim como nos eventos de som, os fluxos de som só existem na memória virtual e seu download não é feito para o disco rígido.

### Exemplo

O exemplo a seguir carrega um evento de som:

```
s.loadSound( "http://caminho_do_servidor:porta/nome_de_arquivo_mp3", false);
```

O exemplo a seguir carrega um fluxo de som:

```
loadSound( "http://caminho_do_servidor:porta/nome_de_arquivo_mp3", true);
```

## Sound.onLoad

### Disponibilidade

Flash Player 6.

### Uso

```
mySoundObject.onLoad = callbackFunction
```

### Parâmetros

*mySoundObject* Um objeto Sound.

*callbackFunction* Uma função.

### Retorna

Nada.

### Descrição

Identificador de eventos; chamado automaticamente quando um som é carregado. Crie uma função que seja executada quando o evento `onLoad` for chamado. É possível usar uma função anônima ou uma função nomeada.

### Consulte também

`Sound.onSoundComplete`

# Sound.onSoundComplete

## Disponibilidade

Flash Player 6.

## Uso

*mySoundObject.onSoundComplete = callbackFunction*

## Parâmetros

*mySoundObject* Um objeto Sound.

*callbackFunction* Uma função.

## Retorna

Nada.

## Descrição

Evento; chamado automaticamente quando a reprodução de um som é concluída. É possível usar o evento `onSoundComplete` para ativar eventos em um filme com base na conclusão de um som.

Crie uma função que seja executada quando o evento `onSoundComplete` for chamado. É possível usar uma função anônima ou uma função nomeada.

## Exemplo

Uso 1: o exemplo a seguir usa uma função anônima:

```
s = new Sound();
s.attachSound("mySound");
s.onSoundComplete = function() { trace("mySound completed"); };
s.start();
```

Uso 2: o exemplo a seguir usa uma função nomeada:

```
function callback1() {
    trace("mySound completed");
}

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

s.onSoundComplete = callback1;

s.start();
```

# Sound.position

## Disponibilidade

Flash Player 6.

## Uso

*mySound.position*

## Parâmetros

Nenhum.

## Retorna

Número de milissegundos em que o som foi reproduzido.

### Descrição

Propriedade (somente leitura); retorna o número de milissegundos em que um som foi reproduzido. Se o som for repetido, a posição será redefinida como 0 no início de cada loop.

## Sound.setPan

### Disponibilidade

Flash Player 5.

### Uso

```
mySound.setPan(pan);
```

### Parâmetros

*pan* Um inteiro que especifica a distribuição esquerda-direita de um som. O intervalo de valores válidos é de -100 a 100, no qual -100 usa somente o canal esquerdo, 100 usa somente o canal direito e 0 distribui o som uniformemente entre os dois canais.

### Retorna

Nada.

### Descrição

Método; determina como o som é reproduzido nos canais esquerdo e direito (alto-falantes). No caso de sons mono, *pan* determina o alto-falante (esquerdo ou direito) pelo qual o som passa.

### Exemplo

O exemplo a seguir cria uma instância do objeto `Sound` *s* e anexa um som com o Identificador L7 da Biblioteca. Também chama os métodos `setVolume` e `setPan` para controlar o som L7.

```
onClipEvent(mouseDown) {  
    // cria um objeto de som  
    s = new Sound(this);  
    // anexa um som da biblioteca  
    s.attachSound("L7");  
    //define o volume como 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.attachSound`, `Sound.setPan`, `Sound.setTransform`, `Sound.setVolume`, `Sound.start`

## Sound.setTransform

### Disponibilidade

Flash Player 5.

### Uso

```
mySound.setTransform(soundTransformObject)
```

### Parâmetros

*soundTransformObject* Um objeto criado com o construtor do objeto `Object` genérico.

## Retorna

Nada.

## Descrição

Método; define as informações de transformação ou de “distribuição” do som de um objeto Sound.

O parâmetro *soundTransformObject* é um objeto criado por meio do método construtor do objeto Object genérico com parâmetros que especificam como o som é distribuído para os canais esquerdo e direito (alto-falantes).

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 6 bits de 22 KHz. Você pode usar o método *setTransform* para reproduzir sons mono como estéreo, sons estéreo como mono e para adicionar efeitos de som interessantes.

Os parâmetros do *soundTransformObject* são os seguintes:

**ll** Uma porcentagem que especifica a quantidade de som do canal esquerdo a ser reproduzida no alto-falante esquerdo (de 0 a 100).

**lr** Uma porcentagem que especifica a quantidade de som do canal direito a ser reproduzida no alto-falante esquerdo (de 0 a 100).

**rr** Uma porcentagem que especifica a quantidade de som do canal direito a ser reproduzida no alto-falante direito (de 0 a 100).

**rl** Uma porcentagem que especifica a quantidade de som do canal esquerdo a ser reproduzida no alto-falante direito (de 0 a 100).

O resultado líquido dos parâmetros é representado pela seguinte fórmula:

```
leftOutput = left_input * ll + right_input * lr
rightOutput = right_input * rr + left_input * rl
```

Os valores para *left\_input* ou *right\_input* 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
```

## Exemplo

O exemplo a seguir ilustra uma configuração que pode ser obtida com o método *setTransform*, mas não com os métodos *setVolume* ou *setPan*, mesmo que estejam combinados.

O código abaixo cria um novo objeto `soundTransformObject` e define suas propriedades para que o som de ambos os canais seja reproduzido somente no canal esquerdo.

```
mySoundTransformObject = new Object;  
mySoundTransformObject.ll = 100;  
mySoundTransformObject.lr = 100;  
mySoundTransformObject.rr = 0;  
mySoundTransformObject.rl = 0;
```

Para aplicar o objeto `soundTransformObject` a um objeto `Sound`, é necessário passar o objeto para `Sound` usando o método `setTransform` da seguinte maneira:

```
mySound.setTransform(mySoundTransformObject);
```

O exemplo a seguir reproduz um som estéreo como mono; *`soundTransformObjectMono`* tem os seguintes parâmetros:

```
mySoundTransformObjectMono = new Object;  
mySoundTransformObjectMono.ll = 50;  
mySoundTransformObjectMono.lr = 50;  
mySoundTransformObjectMono.rr = 50;  
mySoundTransformObjectMono.rl = 50;  
  
mySound.setTransform(soundTransformObjectMono);
```

Este exemplo reproduz o canal esquerdo na metade de sua capacidade e adiciona o restante do canal esquerdo ao canal direito; *`soundTransformObjectHalf`* tem os seguintes parâmetros:

```
mySoundTransformObjectHalf = new Object;  
mySoundTransformObjectHalf.ll = 50;  
mySoundTransformObjectHalf.lr = 0;  
mySoundTransformObjectHalf.rr = 100;  
mySoundTransformObjectHalf.rl = 50;  
  
setTransform(soundTransformObjectHalf);
```

#### Consulte também

Construtor do objeto `Object`

## Sound.setVolume

### Disponibilidade

Flash Player 5.

### Uso

```
mySound.setVolume(volume)
```

### Parâmetros

*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.

### Retorna

Nada.

### Descrição

Método; define o volume do objeto `Sound`.

### Exemplo

O exemplo a seguir define o volume em 50% e, com o tempo, transfere o som do alto-falante esquerdo para o direito:

```
onClipEvent (load) {  
    i = -100;  
    s = new Sound();  
    s.setVolume(50);  
}  
onClipEvent (enterFrame) {  
    if (i <= 100) {  
        S.setPan(i++);  
    }  
}
```

### Consulte também

Sound.setPan, Sound.setTransform

## Sound.start

### Disponibilidade

Flash Player 5.

### Uso

```
mySound.start([deslocamento_Segundo, loop])
```

### Parâmetros

*deslocamento\_Segundo* Um parâmetro opcional que permite começar a reproduzir o som em um ponto específico. Por exemplo, no caso de um som de 30 segundos que deva iniciar a reprodução no meio, especifique 15 para o parâmetro *deslocamento\_Segundo*. O som não é atrasado 15 segundos; em vez disso, ele inicia a sua reprodução na marca de 15 segundos.

*loop* Um parâmetro opcional que permite especificar o número de vezes em que o som deve ser reproduzido consecutivamente.

### Retorna

Nada.

### Descrição

Método; inicia a reprodução do último som anexado desde o início se nenhum parâmetro estiver especificado, ou em um ponto especificado pelo parâmetro *deslocamento\_Segundo*.

### Consulte também

Sound.stop

## Sound.stop

### Disponibilidade

Flash Player 5.

### Uso

```
mySound.stop(["Nome_id"])
```

### Parâmetros

*Nome\_id* Uma parâmetro opcional que especifica a interrupção da reprodução de determinado som. O parâmetro *Nome\_id* deve ser colocado entre aspas (" ").

### Retorna

Nada.

### Descrição

Método; interrompe todos os sons em reprodução no momento se nenhum parâmetro estiver especificado, ou somente o som especificado no parâmetro *Nome\_id*.

### Consulte também

`Sound.start`

## **\_soundbuftime**

### Disponibilidade

Flash Player 4.

### Uso

`_soundbuftime = integer`

### Parâmetros

*integer* 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.

## **Stage (objeto)**

O objeto Stage é um objeto de alto nível que pode ser acessado sem o uso de um construtor.

Use os métodos e as propriedades deste objeto para acessar e manipular informações sobre os limites de um filme do Flash.

O objeto Stage está disponível no Flash Player 6 e posterior.

## **Resumo de métodos do objeto Stage**

Método	Descrição
<code>Stage.addListener</code>	Adiciona um objeto ouvinte ao objeto Stage.
<code>Stage.removeListener</code>	Remove um objeto ouvinte do objeto Stage.

## **Resumo das propriedades do objeto Stage**

Método	Descrição
<code>Stage.align</code>	Alinhamento do filme do Flash no navegador.
<code>Stage.height</code>	Altura do objeto Stage, em pixels.
<code>Stage.width</code>	Largura do objeto Stage, em pixels.
<code>Stage.scaleMode</code>	O dimensionamento atual do filme do Flash.

## Resumo de identificadores de eventos do objeto Stage

Método	Descrição
Stage.onResize	Indica que o filme foi redimensionado.

## Stage.addListener

### Disponibilidade

Flash Player 6.

### Uso

```
Stage.addListener(meu_Ouvinte)
```

### Parâmetros

*meu\_Ouvinte* Objeto que ouve uma notificação de retorno de chamada do evento onResize.

### Retorna

Nada.

### Descrição

Método; detecta quando um filme do Flash é redimensionado se Stage.scaleMode = "noScale". O método addListener não funciona com a configuração de dimensionamento de filme padrão ("showAll") nem com outras configurações de dimensionamento ("exactFit" e "noBorder").

Para usar addListener, crie primeiro um *objeto ouvinte*. O objeto ouvinte é aquele que recebe notificação de um evento quando este é ativado em um filme. Os objetos ouvintes do objeto Stage recebem notificação de Stage.onResize.

### Exemplo

Este exemplo cria um novo objeto ouvinte chamado meu\_Ouvinte. Em seguida, usa meu\_Ouvinte para chamar onResize e define uma função que será chamada quando onResize for ativado. Finalmente, o código adiciona o objeto meu\_Ouvinte à lista de retorno de chamada do objeto Stage. Os objetos ouvintes permitem que vários objetos ouçam notificações de redimensionamento.

```
myListener = new Object();  
myListener.onResize = function () { ... }  
Stage.addListener(meu_Ouvinte);
```

## Stage.align

### Disponibilidade

Flash Player 6.

### Uso

```
Stage.align
```

### Descrição

Propriedade; indica o alinhamento atual do filme do Flash dentro do Palco.



A tabela a seguir lista os valores da propriedade `align`. Os valores não listados aqui centralizam o filme na área do Palco.

Valor	Vertical	Horizontal
"T"	superior	centro
"B"	inferior	centro
"L"	centro	esquerda
"R"	centro	direita
"TL"	superior	esquerda
"TR"	superior	direita
"BL"	inferior	esquerda
"BR"	inferior	direita

## Stage.height

### Disponibilidade

Flash Player 6.

### Uso

`Stage.height`

### Descrição

Propriedade (somente leitura); indica a altura atual, em pixels, do palco do filme do Flash. Quando a propriedade `Stage.noScale` tem o valor `true`, *height* representa a altura do Flash Player. Quando o valor `Stage.noScale` é `false` (o filme é dimensionado quando a janela do exibidor é redimensionada), *height* representa a altura do filme do Flash.

## Stage.onResize

### Disponibilidade

Flash Player 6.

### Uso

```
Stage.onResize() = function() {...}
```

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Método de retorno de chamada; indica que o filme do Flash foi redimensionado. É possível usar este evento para gravar uma função que disponha os objetos no Palco quando um filme for redimensionado.

## Stage.removeListener

### Disponibilidade

Flash Player 6.

### Uso

```
Stage.removeListener(meu_Ouvinte)
```

### Parâmetros

*meu\_Ouvinte* Um objeto adicionado à lista de retorno de chamada de um objeto com o método `addListener`.

### Retorna

Nada.

### Descrição

Método; remove um objeto ouvinte criado com `addListener`.

### Consulte também

`Stage.addListener`

## Stage.scaleMode

### Disponibilidade

Flash Player 6.

### Uso

```
Stage.scaleMode = "value"
```

### Descrição

Propriedade; indica o dimensionamento atual do filme do Flash dentro do Palco. A propriedade `scaleMode` força o modo de dimensionamento específico do filme. Como padrão, o filme usa os parâmetros HTML definidos na caixa de diálogo Configurações de publicação.

A propriedade `scaleMode` pode usar os valores `"exactFit"`, `"showAll"`, `"noBorder"` e `"noScale"`. Qualquer outro valor define a propriedade `scaleMode` como o padrão `"showAll"`.

## Stage.width

### Disponibilidade

Flash Player 6.

### Uso

```
Stage.width
```

### Descrição

Propriedade (somente leitura); indica a largura atual, em pixels, do palco do filme do Flash. Quando o valor de `Stage.noScale` é `true`, a propriedade `width` representa a largura do Player. Quando o valor de `Stage.noScale` é `false` (o filme é dimensionado quando a janela do exibidor é redimensionada), `width` representa a largura do filme do Flash.

## startDrag

### Disponibilidade

Flash Player 4.

### Uso

```
startDrag(destino,[bloqueio ,esquerdo , superior , direito, inferior])
```

### Parâmetros

*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 no centro da posição do mouse (*true*) ou no ponto onde o usuário clicou pela primeira vez no clipe de filme (*false*). Este parâmetro 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 parâmetros são opcionais.

### Retorna

Nada.

### 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 local, anexe as ações *startDrag* e *stopDrag* a um botão dentro do clipe de filme.

```
on(press) {  
    startDrag(this, true);  
}  
on(release) {  
    stopDrag();  
}
```

### Consulte também

`MovieClip._droptarget`, `stopDrag`

## stop

### Disponibilidade

Flash 2.

### Uso

```
stop
```

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Ação; encerra o filme em exibição. A utilidade mais comum dessa ação é controlar clipes de filme com botões.

## stopAllSounds

### Disponibilidade

Flash Player 3.

### Uso

```
stopAllSounds()
```

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Ação; encerra a reprodução de todos os sons de um filme sem interromper a exibição do filme. Sons definidos como 'em streaming' voltarão a ser reproduzidos quando a reprodução for movida sobre os quadros em que se encontram.

### 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)

## stopDrag

### Disponibilidade

Flash Player 4.

### Uso

```
stopDrag()
```

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Ação; encerra a operação de arraste em andamento.

### Exemplo

Este comando interrompe a ação de arraste na instância `mc` quando o usuário libera o botão do mouse:

```
on(press) {  
    startDrag("mc");  
}  
on(release) {  
    stopdrag();  
}
```

### Consulte também

`MovieClip.stopDrag`, `MovieClip._droptarget`, `startDrag`,

## String (função)

### Disponibilidade

Flash Player 4.

### Uso

`String(expressão)`

### Parâmetros

*expressão* Expressão para converter em uma sequência de caracteres.

### Retorna

Nada.

### Descrição

Função; retorna uma representação de sequência de caracteres do parâmetro especificado da seguinte maneira:

Se *expressão* for um valor Booleano, a sequência de caracteres retornada será `true` ou `false`.

Se *expressão* for um número, a sequência de caracteres retornada será uma representação de texto do número.

Se *expressão* for uma sequência de caracteres, a sequência retornada será *expressão*.

Se *expressão* for um objeto, o valor retornado será uma representação de sequência de caracteres do objeto gerado pela chamada da propriedade da sequência de caracteres referente ao objeto ou pela chamada de `Object.toString`, se tal propriedade não existir.

Se *expressão* for um clipe de filme, o valor de retorno será o caminho de destino do clipe de filme em notação de barra (/).

Se *expressão* for `undefined`, o valor de retorno será uma sequência de caracteres vazia ("").

### Consulte também

`Number.toString`, `Object.toString`, `String(objeto)`, " " (delimitador de sequência de caracteres)

## " " (delimitador de seqüência de caracteres)

### Disponibilidade

Flash Player 4.

### Uso

*"texto"*

### Parâmetros

*texto* Um caractere.

### Retorna

Nada.

### Descrição

Delimitador de seqüência de caracteres; quando usadas antes e depois de caracteres, as aspas indicam que os caracteres têm um valor literal e são considerados como uma *seqüência de caracteres*, e não como variável, valor numérico ou outro elemento do ActionScript.

### Exemplo

Este exemplo usa aspas para indicar que o valor da variável *yourGuess* é a seqüência de caracteres literal "Prince Edward Island" e não o nome de uma variável. O valor *province* é uma variável, e não uma literal; para determinar o valor de *province*, é necessário localizar o valor de *yourGuess*.

```
yourGuess = "Prince Edward Island";  
on(release){  
    province = yourGuess  
    trace(province);  
}  
  
// exibe Prince Edward Island na janela Saída
```

### Consulte também

String (objeto), String (função)

## String (objeto)

O objeto String é um envoltório para o tipo de dados primitivo de seqüência de caracteres, que permite usar os métodos e as propriedades do objeto String para manipular tipos de valores primitivos de seqüências de caracteres. É possível converter o valor de qualquer objeto em uma seqüência de caracteres usando a função `String()`. No Flash MX, o objeto String tornou-se um objeto nativo. Assim, você poderá observar uma melhora radical no desempenho.

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.

Como todos os índices de seqüência de caracteres são baseados em zero, o índice do último caractere para qualquer seqüência *x* é o seguinte:

```
x.length - 1
```

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")
```

Use valores literais da sequência de caracteres, a menos seja necessário usar especificamente um objeto `String`.

## Resumo de métodos do objeto `String`

Método	Descrição
<code>String.charAt</code>	Retorna o caractere em um local específico em uma sequência de caracteres.
<code>String.charCodeAt</code>	Retorna o valor do caractere de um índice determinado como um inteiro de 16 bits entre 0 e 65535.
<code>String.concat</code>	Combina o texto de duas sequências de caracteres e retorna uma nova sequência de caracteres
<code>String.fromCharCode</code>	Retorna uma sequência de caracteres constituída de caracteres especificados nos parâmetros.
<code>String.indexOf</code>	Pesquisa a sequência de caracteres e retorna o índice da subsequência de caracteres especificada nos parâmetros. Se o valor ocorrer mais de uma vez, o índice da primeira ocorrência é retornado. Se o valor não for encontrado, -1 é retornado.
<code>String.lastIndexOf</code>	Retorna o índice da última subsequência de caracteres dentro da sequência de caracteres que aparece antes da posição de início especificada no parâmetro, ou retorna -1 se não encontrado.
<code>String.slice</code>	Extrai uma seção de uma sequência de caracteres e retorna uma nova sequência de caracteres.
<code>String.split</code>	Divide um objeto <code>String</code> em uma matriz de sequências de caracteres separando a sequência em subsequências.
<code>String.substr</code>	Retorna um número especificado de caracteres em uma sequência de caracteres, começando no local especificado no parâmetro.
<code>String.substring</code>	Retorna os caracteres entre dois índices, especificado nos parâmetros como uma sequência de caracteres.
<code>String.toLowerCase</code>	Converte a sequência de caracteres em minúsculas e retorna o resultado; não altera o conteúdo do objeto original.
<code>String.toUpperCase</code>	Converte a sequência de caracteres em maiúsculas e retorna o resultado; não altera o conteúdo do objeto original.

## Resumo de propriedades do objeto `String`

Propriedade	Descrição
<code>String.length</code>	Retorna o tamanho da sequência de caracteres

## Construtor do objeto String

### Disponibilidade

Flash Player 5.

### Uso

```
new String(valor)
```

### Parâmetros

*valor* O valor inicial do objeto new String.

### Retorna

Nada.

### Descrição

Construtor; cria um objeto new String.

### Consulte também

String (função), " " (delimitador de sequência de caracteres)

## String.charAt

### Disponibilidade

Flash Player 5.

### Uso

```
myString.charAt(índice)
```

### Parâmetros

*índice* O número do caractere a ser retornado na sequência de caracteres.

### Retorna

Nada.

### Descrição

Método; retorna o caractere na posição especificada pelo parâmetro *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`, será retornada uma sequência de caracteres vazia.

## String.charCodeAt

### Disponibilidade

Flash Player 5.

### Uso

```
myString.charCodeAt(índice)
```

### Parâmetros

*índice* Um inteiro que especifica a posição de um caractere na sequência de caracteres. O primeiro caractere é indicado por 0 e o último é indicado por `myString.length-1`.

### Retorna

Nada.



**Descrição**

Método; retorna um número inteiro de 16 bits de 0 a 65535 que representa o caractere especificado por *índice*.

Este método é semelhante a `string.charAt`, exceto pelo fato de o valor retornado ser um código de caracteres inteiro de 16 bits, e não um caractere.

**Exemplo**

No exemplo a seguir, o método `charCodeAt` é chamado na primeira letra da sequência de caracteres “Chris”.

```
s = new String("Chris");  
i = s.charCodeAt(0);  
// i = 67
```

## String.concat

**Disponibilidade**

Flash Player 5.

**Uso**

`myString.concat(valor1,...valorN)`

**Parâmetros**

*valor1,...valorN* Zero ou mais valores a serem concatenados.

**Retorna**

Nada.

**Descrição**

Método; combina o valor do objeto `String` aos parâmetros e retorna a sequência de caracteres recém-formada; o valor original, *myString*, é inalterado.

## String.fromCharCode

**Disponibilidade**

Flash Player 5.

**Uso**

`String.fromCharCode(c1,c2,...cN)`

**Parâmetros**

*c1,c2,...cN* Inteiros decimais que representam valores ASCII.

**Retorna**

Nada.

**Descrição**

Método; retorna uma sequência de caracteres constituída de caracteres representados pelos valores ASCII nos parâmetros.

### Exemplo

Este exemplo usa o método `fromCharCode` para inserir um caractere “@” no endereço eletrônico.

```
address = "dog" + String.fromCharCode(64) + "house.net";  
trace(address);  
// saída: dog@house.net
```

## String.indexOf

### Disponibilidade

Flash Player 5.

### Uso

`myString.indexOf(subseqüência de caracteres, [início_Índice])`

### Parâmetros

*subseqüência de caracteres* Um inteiro ou uma seqüência de caracteres que especifica a subseqüência de caracteres a ser procurada em *myString*.

*início\_Índice* Um inteiro que especifica o ponto inicial em *myString* para procurar pela subseqüência de caracteres. Este parâmetro é opcional.

### Retorna

Nada.

### Descrição

Método; pesquisa a seqüência de caracteres e retorna a posição da primeira ocorrência da *subseqüência de caracteres* especificada. Se o valor não for encontrado, o método retorna -1.

## String.lastIndexOf

### Disponibilidade

Flash Player 5.

### Uso

`myString.lastIndexOf(subseqüência de caracteres, [início_Índice])`

### Parâmetros

*subseqüência de caracteres* Um inteiro ou seqüência de caracteres que especifica a seqüência a ser procurada.

*início\_Índice* Um inteiro que especifica o ponto inicial para procurar pela *subseqüência de caracteres*. Este parâmetro é opcional.

### Retorna

Nada.

### Descrição

Método; procura a seqüência de caracteres da direita para a esquerda e retorna o índice da última ocorrência de *subseqüência de caracteres* localizada antes de *início\_Índice* na seqüência de caracteres de chamada. Se a *subseqüência de caracteres* não for encontrada, o método retorna -1.

## String.length

### Disponibilidade

Flash Player 5.

### Uso

`string.length`

### Parâmetros

Nenhum.

### Descrição

Propriedade; retorna o número de caracteres no objeto String especificado.

## String.slice

### Disponibilidade

Flash Player 5.

### Uso

`myString.slice(início, [fim])`

### Parâmetros

*início* Um número que especifica o índice do ponto inicial da fatia. 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.

*fim* Um número que especifica o índice do ponto final da fatia. Se *fim* não for especificado, a fatia incluirá 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.

### Retorna

Nada.

### Descrição

Método; extrai uma fatia, ou subsequência de caracteres, do objeto String especificado; em seguida, retorna-a como uma nova sequência, sem modificar o objeto String original. A sequência de caracteres retornada inclui o caractere de *início* e todos os caracteres até (mas não incluindo) o caractere de *fim*.

### Exemplo

O exemplo a seguir define uma variável, `text`, cria uma instância do objeto String, `s`, e a passa para a variável `text`. O método `slice` extrai uma seção da sequência de caracteres contida na variável e a ação `trace` a envia para a janela Saída.

```
text = "lexington";  
s = new String( text );  
trace(s.slice( 1, 3 ));  
trace(s);
```

A janela Saída exibe ex.

O código a seguir produz o mesmo resultado, mas o parâmetro passado para a função `String` é uma sequência de caracteres em vez de uma variável.

```
s = new String( "lexington" );
trace(s.slice( 1, 3 ));
trace(s);
```

A janela Saída exibe ex.

## String.split

### Disponibilidade

Flash Player 5.

### Uso

```
myString.split("delimitador", [limite])
```

### Parâmetros

*delimitador* O caractere ou a sequência de caracteres em que *myString* é dividido. Se o parâmetro *delimitador* não for definido, toda a sequência de caracteres será colocada no primeiro elemento da matriz.

*limite* O número de itens que devem ser colocados na matriz. Este parâmetro é opcional.

### Retorna

Uma matriz que contém as subsequências de caracteres de *myString*.

### Descrição

Método; divide um objeto `String` em subsequências de caracteres quebrando-o sempre que o parâmetro *delimitador* especificado ocorre e retorna as subsequências de caracteres em uma matriz. Se for usada uma sequência de caracteres vazia ("" ) como delimitador, cada caractere na sequência será colocado como um elemento na matriz, como no código a seguir.

```
myString = "Joe";
i = myString.split("");
trace (i);
```

A janela Saída exibe o seguinte:

J, O, E

Se o parâmetro *delimitador* não for definido, toda a sequência de caracteres será colocada no primeiro elemento da matriz retornada.

### Exemplo

O exemplo a seguir retorna uma matriz com cinco elementos.

```
myString = "P, A, T, S, Y";
myString.split(",");
```

Este exemplo retorna uma matriz com dois elementos.

```
myString.split(", ", 2);
```

## String.substr

### Disponibilidade

Flash Player 5.

### Uso

*myString*.substr(*início*, [*tamanho*])

### Parâmetros

*início* Um inteiro que indica a posição do primeiro caractere em *myString* a ser usado para criar a subsequência de caracteres. 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.

### Retorna

Nada.

### Descrição

Método; retorna os caracteres em uma sequência de caracteres do índice especificado no parâmetro *início* até o número de caracteres especificado no parâmetro *tamanho*. O método *substr* não altera a sequência de caracteres especificada por *myString*, mas retorna uma nova sequência de caracteres.

## String.substring

### Disponibilidade

Flash Player 5.

### Uso

*myString*.substring(*de*, *para*)

### Parâmetros

*de* Um inteiro que indica a posição do primeiro caractere de *myString* usado para criar a subsequência de caracteres. Os valores válidos referentes a *de* vão de 0 a *string.length* - 1. Se *de* for um valor negativo, 0 será usado.

*para* Um inteiro que é 1+ o índice do último caractere em *myString* a ser extraído. Os valores válidos referentes a *para* vão de 1 a *string.length*. O caractere indexado pelo parâmetro *para* não está incluído na sequência de caracteres extraída. Se este parâmetro for omitido, *string.length* será usado. Se este parâmetro for um valor negativo, 0 será usado.

### Retorna

Nada.

### Descrição

Método; retorna uma sequência de caracteres que consiste nos caracteres entre os pontos especificados pelos parâmetros *de* e *para*. Se o parâmetro *to* não for especificado, o fim da subsequência de caracteres será o fim da sequência de caracteres. Se o valor referente a *de* for igual ao valor referente a *para*, o método retornará uma sequência de caracteres vazia. Se o valor referente a *de* for maior que o valor referente a *para*, os parâmetros serão trocados automaticamente antes de a função ser executada e o valor original será inalterado.

## String.toLowerCase

### Disponibilidade

Flash Player 5.

### Uso

```
myString.toLowerCase()
```

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Método; retorna uma cópia do objeto String, com todos os caracteres em maiúsculas convertidos em minúsculas. O valor original não se altera.

## String.toUpperCase

### Disponibilidade

Flash Player 5.

### Uso

```
myString.toUpperCase()
```

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Método; retorna uma cópia do objeto String, com todos os caracteres em maiúsculas convertidos em minúsculas. O valor original não se altera.

## substring

### Disponibilidade

Flash Player 4. Esta função tornou-se obsoleta em favor de `String.substr`.

### Uso

```
substring("seqüência de caracteres", índice, contagem)
```

### Parâmetros

*seqüência de caracteres* A seqüência de caracteres da qual será extraída 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.

**Retorna**

Nada.

**Descrição**

Função String; extrai parte de uma sequência de caracteres. Esta função baseia-se em 1, enquanto os métodos do objeto String baseiam-se em 0.

**Consulte também**

String.substr

## super

**Disponibilidade**

Flash Player 6.

**Uso**

```
super.method([arg1, ..., argN])
```

```
super([arg1, ..., argN])
```

**Parâmetros**

*method* O método que será chamado na superclasse.

*arg1* Parâmetros opcionais que são passados para a versão da superclasse do método (sintaxe 1) ou para a função construtora da superclasse (sintaxe 2).

**Retorna**

Ambas as formas chamam uma função. A função pode retornar qualquer valor.

**Descrição**

Operador: o estilo da primeira sintaxe pode ser usado no corpo de um método de objeto para chamar a versão de superclasse de um método e pode, opcionalmente, passar parâmetros (*arg1 ... argN*) para o método de superclasse. Isso é útil para criar métodos de subclasse que adicionem um comportamento extra aos métodos de superclasse, mas que também chamem os métodos de superclasse para executar o comportamento original.

O estilo da segunda sintaxe pode ser usado dentro do corpo de uma função construtora para chamar a versão de superclasse da função construtora e pode, opcionalmente, passar parâmetros para ela. Isso é útil para criar uma subclasse que execute uma inicialização adicional, mas que também chame o construtor de superclasse para executar a inicialização da superclasse.

### Exemplo

O exemplo a seguir cria duas classes, `ParentClass` e `ChildClass`, e define um método chamado `method` para cada classe. Cada definição de método tem uma ação `trace` que envia uma mensagem para a janela Saída. A penúltima linha de código cria uma instância de `ChildClass` e chama seu método `method`:

```
function ParentClass() {  
}  
ParentClass.prototype.method = function () {  
    trace("ParentClass implementation of method");  
};  
function ChildClass() {  
}  
ChildClass.prototype = new ParentClass();  
ChildClass.prototype.method = function () {  
    trace("ChildClass implementation of method");  
    super.method();  
};  
x = new ChildClass();  
x.method();
```

O seguinte é exibido na janela Saída:

```
ChildClass implementation of method  
ParentClass implementation of method
```

O exemplo a seguir passa parâmetros para o super construtor:

```
function SuperClass(arg){  
    trace("SuperClass constructor was passed " + arg);  
}  
function SubClass(arg){  
    super(arg);  
    trace("SubClass constructor");  
}
```

## switch

### Disponibilidade

Flash Player 4.

### Uso

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

### Parâmetros

*expressão* Qualquer expressão.

*caseClause* Uma palavra-chave `case` seguida de uma expressão, dois-pontos e um grupo de comandos para serem executados se a expressão corresponder ao parâmetro *expressão* de troca usando igualdade estrita (`===`).

*defaultClause* Uma palavra-chave `default` seguida de comandos para serem executados se nenhuma das expressões com diferenciação de maiúsculas e minúsculas corresponder ao parâmetro *expressão* de troca usando igualdade estrita (`===`).

### Retorna

Nada.



## Descrição

Ação; cria uma estrutura ramificada para comandos do ActionScript. Como a ação `if`, a ação `switch` testa uma condição e executa comandos se a condição retornar um valor `true`.

## Exemplo

No exemplo a seguir, se o parâmetro `number` for avaliado como 1, a ação `trace` seguinte a `case 1` será executada; se o parâmetro `number` for avaliado como 2, a ação `trace` seguinte a `case 2` será executada e assim por diante. Se nenhuma expressão `case` corresponder ao parâmetro `number`, a ação `trace` seguinte à palavra-chave `default` será executada.

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

No exemplo a seguir, não há uma quebra no primeiro grupo `case`; portanto, se o número for 1, A e B serão enviados para a janela Safda:

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

## Consulte também

`===` (igualdade estrita), `break`, `case`, `default`, `if`

## System (objeto)

Este é um objeto de alto nível que contém o objeto `Capabilities`. É necessário usar o objeto `System` para utilizar o objeto `Capabilities` e suas propriedades. Por exemplo, o código a seguir verifica se um sistema tem recursos de áudio.

```
System.capabilites.hasAudio
```

## System.capabilities (objeto)

É possível usar o objeto `System.capabilites` para determinar os recursos do sistema e do exibidor que hospedam um filme do Flash. Isso permite ajustar o conteúdo para formatos diferentes. Por exemplo, a tela de um telefone celular (preto-e-branco, 100 pixels quadrados) é diferente da tela de PC colorida de 1.000 pixels quadrados. Para oferecer um conteúdo apropriado ao maior número possível de usuários, é possível usar o objeto `Capabilities` para determinar o tipo de dispositivo de um usuário. Em seguida, é possível especificar que o servidor envie diferentes arquivos SWFs com base nos recursos do dispositivo, ou solicitar que o filme do Flash altere sua apresentação com base nos recursos do dispositivo.

É possível enviar informações sobre recursos usando um método GET ou POST HTTP. A seguir, é exibido um exemplo de uma sequência de caracteres do servidor referente a um dispositivo que não tem suporte MP3 e tem uma tela de 400 x 200 pixels, 8 x 4 centímetros:

```
"A=t&MP3=f&AE=gsm&VE=h11&ACC=f&V=WIN%206%2C0%2C0%2C129&M=Macromedia%WINDOWS&R=400x200&DP=72&COL=co1or&AR=1.0&OS=WIND0WS%2000&L=en-US"
```

O objeto Capabilities está disponível no Flash Player 6.

Acesse todas as propriedades do objeto Capabilities por meio do objeto System.capabilities.

## Resumo das propriedades do objeto Capabilities

Propriedade	Descrição
System.capabilities.hasAudioEncoder	Indica os codificadores de áudio suportados.
System.capabilities.hasAccessibility	Indica se o dispositivo atende aos padrões de acessibilidade.
System.capabilities.hasAudio	Indica se o dispositivo tem recursos de áudio.
System.capabilities.hasMP3	Indica se o dispositivo tem um decodificador MP3.
System.capabilities.language	Indica o idioma suportado pelo Flash Player.
System.capabilities.manufacturer	Indica o fabricante do Flash Player.
System.capabilities.os	Indica o sistema operacional que hospeda o Flash Player.
System.capabilities.pixelAspectRatio	Indica a proporção de pixels da tela.
System.capabilities.screenColor	Indica se a tela é colorida, em preto-e-branco ou em tons de cinza.
System.capabilities.screenDPI	Indica os pontos por polegada da tela.
System.capabilities.screenResolution.x	Indica o tamanho horizontal da tela.
System.capabilities.screenResolution.y	Indica o tamanho vertical da tela.
System.capabilities.version	Indica a versão mais antiga do Flash Player para a qual há suporte.
System.capabilities.hasVideoEncoder	Indica os codificadores de vídeo suportados.

## System.capabilities.hasAudioEncoder

### Disponibilidade

Flash Player 6.

### Uso

```
System.capabilities.hasAudioEncoder
```

### Descrição

Propriedade; uma matriz de decodificadores de áudio. A sequência de caracteres do servidor é AE.

## System.capabilities.hasAccessibility

### Disponibilidade

Flash Player 6.

### Uso

`System.capabilities.hasAccessibility`

### Descrição

Propriedade; um valor booleano que indica se o dispositivo suporta ou não a comunicação entre o Flash Player e os auxílios de acessibilidade. O valor padrão é `false`. A sequência de caracteres do servidor é ACC.

## System.capabilities.hasAudio

### Disponibilidade

Flash Player 6.

### Uso

`System.capabilities.hasAudio`

### Descrição

Propriedade; um valor booleano que indica se o exibidor tem ou não recursos de áudio. O valor padrão é `true`. A sequência de caracteres do servidor é A.

## System.capabilities.hasMP3

### Disponibilidade

Flash Player 6.

### Uso

`System.capabilities.hasMP3`

### Descrição

Propriedade; um valor booleano que indica se o exibidor tem ou não um decodificador MP3. O valor padrão é `true`. A sequência de caracteres do servidor é MP3.

## System.capabilities.language

### Disponibilidade

Flash Player 6.

### Uso

`System.capabilities.language`

### Descrição

Propriedade; um código de idioma de duas letras minúsculas do ISO 639-1, e uma submarca de código de país de duas letras maiúsculas opcional do ISO 3166. Os próprios idiomas são nomeados com as marcas em inglês. Por exemplo, “pt” é o idioma do documento que você lê neste momento. A sequência de caracteres do servidor é LAN. O Flash suporta o seguinte subconjunto das marcas de idioma:

Idioma	Marca	Países e Marcas Suportados
Inglês	en	Estados Unidos = US, Reino Unido = UK
Francês	fr	
Coreano	ko	
Japonês	ja	
Sueco	sv	
Alemão	de	
Espanhol	es	
Italiano	it	
Chinês Simplificado	zh	República Popular da China (Chinês Simplificado) = CN
Chinês Tradicional	zh	Taiwan (Chinês Tradicional) = TW
Português	pt	
Polonês	pl	
Húngaro	hu	
Tcheco	cs	
Turco	tr	
Finlandês	fi	
Dinamarquês	da	
Norueguês	no	
Holandês	nl	
Russo	ru	
Outros/Desconhecidos	xu	

## System.capabilities.manufacturer

### Disponibilidade

Flash Player 6.

### Uso

```
System.capabilities.manufacturer
```

### Descrição

Propriedade; uma sequência de caracteres que indica o fabricante do Flash Player. O padrão é "Macromedia OSName" (*OSName* pode ser "Windows", "Macintosh" ou "Other OS Name"). A sequência de caracteres do servidor é M.

## System.capabilities.os

### Disponibilidade

Flash Player 6.

### Uso

`System.capabilities.os`

### Descrição

Propriedade; uma seqüência de caracteres que indica o fabricante do Flash Player. O padrão é uma seqüência de caracteres vazia (""). A propriedade `os` pode retornar as seguintes seqüências de caracteres: "Windows XP", "Windows 2000", "Windows NT", "Windows 98/ME", "Windows 95", "Windows CE" (disponível apenas em SDK, não na versão desktop) e "MacOS". A seqüência de caracteres do servidor é OS.

## System.capabilities.pixelAspectRatio

### Disponibilidade

Flash Player 6.

### Uso

`System.capabilities.hasVideoEncoder`

### Descrição

Propriedade; um inteiro que indica a proporção de pixels da tela. O valor padrão é 1.0. A seqüência de caracteres do servidor é PAR.

## System.capabilities.screenColor

### Disponibilidade

Flash Player 6.

### Uso

`System.capabilities.screenColor`

### Descrição

Propriedade; indica a cor da tela: colorida (`color`), cinza (`gray`) ou preto-e-branco (`bw`). O valor padrão é `color`. A seqüência de caracteres do servidor é SC.

## System.capabilities.screenDPI

### Disponibilidade

Flash Player 6.

### Uso

`System.capabilities.screenDPI`

### Descrição

Propriedade; indica os pontos por polegada (`dpi`) da tela, em pixels. O valor padrão é 72. A seqüência de caracteres do servidor é DPI.

## System.capabilities.screenResolution.x

### Disponibilidade

Flash Player 6.

### Uso

System.capabilities.screenResolution.x

### Descrição

Propriedade; um inteiro que indica a resolução horizontal máxima da tela. O valor padrão é 800 (pixels). A sequência de caracteres do servidor é SRX.

## System.capabilities.screenResolution.y

### Disponibilidade

Flash Player 6.

### Uso

System.capabilities.screenResolution.y

### Descrição

Propriedade; um inteiro que indica a resolução vertical máxima da tela. O valor padrão é 600 (pixels). A sequência de caracteres do servidor é SRY.

## System.capabilities.version

### Disponibilidade

Flash Player 6.

### Uso

System.capabilities.version

### Descrição

Propriedade; um inteiro que especifica a versão suportada do Flash Player. O padrão é 6.0. A sequência de caracteres do servidor é VER.

## System.capabilities.hasVideoEncoder

### Disponibilidade

Flash Player 6.

### Uso

System.capabilities.hasVideoEncoder

### Descrição

Propriedade; uma matriz de codificadores de vídeo. A sequência de caracteres do servidor é VE.

## targetPath

### Disponibilidade

Flash Player 5.

### Uso

`targetpath(objeto_MovieClip)`

### Parâmetros

*objeto\_MovieClip* Referência (por exemplo, `_root` ou `_parent`) ao clipe de filme cujo caminho de destino está sendo recuperado.

### Retorna

Nada.

### Descrição

Função; retorna uma sequência de caracteres que contém o caminho de destino de *objeto\_MovieClip*. O caminho de destino é retornado em notação com pontos. Para recuperar o caminho de destino em notação de barras, use a propriedade `_target`.

### Exemplo

Este exemplo exibe o caminho de destino de um clipe de filme assim que é carregado.

```
onClipEvent(load){
    trace(targetPath(this));
}
```

### Consulte também

`eval`

## tellTarget

### Disponibilidade

Flash Player 3. (Obsoleto no Flash 5; é recomendável usar a notação com pontos e a ação `with`.)

### Uso

```
tellTarget("destino") {
    comando(s);
}
```

### Parâmetros

*destino* Uma sequência de caracteres que especifica o caminho de destino da Linha de tempo a ser controlada.

*comando(s)* Os comandos que serão executados se a condição for avaliada como `true`.

### Retorna

Nada.

## Descrição

Ação; aplica os comandos especificados no parâmetro *statements* para a Linha de tempo especificada no parâmetro *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 encerrem ou comecem clipes de filme no Palco ou solicitem que os clipes de filme saltem para um quadro específico.

No Flash 5, é possível usar a notação com pontos em vez da ação `tellTarget`. Use a ação `with` a fim de emitir várias ações para a mesma Linha de tempo. É possível usar a ação `with` para especificar qualquer objeto, enquanto a ação `tellTarget` só pode especificar clipes de filme.

## Exemplo

O comando `tellTarget` controla na instância do clipe de filme `ball` na Linha de tempo principal. O quadro 1 da instância `ball` está em branco e tem uma ação `stop`, de maneira que não é visível no Palco. Quando o botão com a ação a seguir é clicado, `tellTarget` solicita que a reprodução em `ball` vá para o quadro 2, onde a animação começa.

```
on(release) {
    tellTarget("ball") {
        gotoAndPlay(15);
    }
}
```

O exemplo a seguir usa uma notação com pontos para alcançar os mesmos resultados.

```
on(release) {
    ball.gotoAndPlay(2);
}
```

Se for necessário emitir vários comandos para a instância `ball`, use a ação `with`, como no comando a seguir.

```
on(release) {
    with(ball) {
        gotoAndPlay(15);
        _alpha = 15;
        _xscale = 50;
        _yscale = 50;
    }
}
```

## Consulte também

`with`

## TextField (objeto)

Todos os campos de texto dinâmicos e de entrada em um filme do Flash são instâncias do objeto `TextField`. É possível dar a um campo de texto um nome de instância no Inspetor de propriedades e usar os métodos e as propriedades do objeto `TextField` para manipulá-lo com o `ActionScript`. Os nomes de instância do `TextField` são exibidos no `Movie Explorer` e na caixa de diálogo `Inserir caminho de destino` no painel `Ações`.

O objeto `TextField` herda do objeto `Object`.

Para criar um campo de texto dinamicamente, use o método `MovieClip.createTextField`.

O objeto `TextField` tem suporte do Flash Player 6 e de suas versões posteriores.



## Resumo de métodos do objeto TextField

Método	Descrição
<code>TextField.addListener</code>	Registra um objeto para receber notificação quando os eventos <code>onChanged</code> e <code>onScroller</code> forem chamados.
<code>TextField.getDepth</code>	Retorna a espessura de um campo de texto.
<code>TextField.getNewTextFormat</code>	Obtém o formato de texto padrão atribuído ao texto recém-inserido.
<code>TextField.removeListener</code>	Remove um objeto ouvinte.
<code>TextField.removeTextField</code>	Remove um campo de texto que foi criado com <code>MovieClip.createTextField</code> .
<code>TextField.setNewTextFormat</code>	Define um objeto de formato de texto para o texto que é inserido por um usuário ou por um método.
<code>TextField.replaceSel</code>	Substitui a seleção atual.
<code>TextField.setTextFormat</code>	Define o formato de texto padrão atribuído ao texto recém-inserido.

## Resumo das propriedades do objeto TextField

Propriedade	Descrição
<code>TextField._alpha</code>	O valor da transparência de uma instância de campo de texto.
<code>TextField.autoSize</code>	Controla o alinhamento automático e o dimensionamento de um campo de texto.
<code>TextField.background</code>	Indica se o campo de texto tem um preenchimento de fundo.
<code>TextField.backgroundColor</code>	Indica a cor do preenchimento de fundo.
<code>TextField.border</code>	Indica se o campo de texto tem uma borda.
<code>TextField.borderColor</code>	Indica a cor da borda.
<code>TextField.bottomScroll</code>	A linha visível mais inferior em um campo de texto.
<code>TextField.embedFonts</code>	Indica se o campo de texto usa contornos de fontes incorporadas ou fontes de dispositivo.
<code>TextField._highquality</code>	Indica a qualidade do filme.
<code>TextField._height</code>	A altura de uma instância de campo de texto em pixels. Afeta somente a caixa delimitadora do campo de texto, e não a espessura da borda nem o tamanho da fonte de texto.
<code>TextField.hscroll</code>	Indica o valor de rolagem horizontal de um campo de texto.
<code>TextField.html</code>	Indica a posição de rolagem máxima atual de um campo de texto.
<code>TextField.htmlText</code>	Contém uma representação HTML do conteúdo de um campo de texto.
<code>TextField.length</code>	O número de caracteres em um campo de texto.
<code>TextField.maxChars</code>	O número máximo de caracteres que um campo de texto pode conter.
<code>TextField.maxhscroll</code>	O valor máximo de <code>TextField.hscroll</code> .
<code>TextField.maxscroll</code>	O valor máximo de <code>TextField.scroll</code> .
<code>TextField.multiline</code>	Indica se o campo de texto contém várias linhas.
<code>TextField._name</code>	O nome de uma instância de campo de texto.

Propriedade	Descrição
<code>TextField._parent</code>	Uma referência à instância que é o pai desta instância; seja do tipo <code>Button</code> ou <code>MovieClip</code> .
<code>TextField.password</code>	Indica se um campo de texto oculta os caracteres de entrada.
<code>TextField._quality</code>	Indica a qualidade de um filme.
<code>TextField.restrict</code>	O conjunto de caracteres que um usuário pode digitar em um campo de texto.
<code>TextField._rotation</code>	O grau de rotação de uma instância de campo de texto.
<code>TextField.scroll</code>	Indica a posição de rolagem atual de um campo de texto.
<code>TextField.selectable</code>	Indica se um campo de texto pode ser selecionado.
<code>TextField._soundbuftime</code>	A quantidade de tempo em que um som deve ser armazenado em pré-buffer antes de ser reproduzido.
<code>TextField.tabEnabled</code>	Indica se um clipe de filme está incluído na ordenação de tabulação automática.
<code>TextField.tabIndex</code>	Indica a ordem de guias de um objeto.
<code>TextField.text</code>	O texto atual no campo de texto.
<code>TextField.textColor</code>	A cor do texto atual no campo de texto.
<code>TextField.textHeight</code>	A altura da caixa delimitadora do campo de texto.
<code>TextField.textWidth</code>	A largura da caixa delimitadora do campo de texto.
<code>TextField.type</code>	Indica se um campo de texto é dinâmico ou de entrada.
<code>TextField._url</code>	A URL do arquivo SWF que criou a instância de campo de texto.
<code>TextField.variable</code>	O nome da variável associada ao campo de texto.
<code>TextField._visible</code>	Um valor booleano que determina se uma instância de campo de texto está oculta ou visível.
<code>TextField._width</code>	A largura de uma instância de campo de texto em pixels. Afeta somente a caixa delimitadora do campo de texto, e não a espessura da borda nem o tamanho da fonte de texto.
<code>TextField.wordWrap</code>	Indica se o campo de texto faz quebra automática de linha.
<code>TextField._x</code>	A coordenada <i>x</i> de uma instância de campo de texto.
<code>TextField._xmouse</code>	A coordenada <i>x</i> do cursor relativo a uma instância de campo de texto.
<code>TextField._xscale</code>	O valor que especifica a porcentagem para dimensionar horizontalmente uma instância de campo de texto.
<code>TextField._y</code>	A coordenada <i>y</i> de uma instância de campo de texto.
<code>TextField._ymouse</code>	A coordenada <i>y</i> do cursor relativo a uma instância de campo de texto.
<code>TextField._yscale</code>	O valor que especifica a porcentagem para dimensionar verticalmente uma instância de campo de texto.

## Resumo de identificadores de eventos do objeto TextField

Método	Descrição
<code>TextField.onChangeed</code>	Chamado quando o campo de texto é alterado.
<code>TextField.onKillFocus</code>	Chamado quando o campo de texto perde o foco.
<code>TextField.onScroller</code>	Chamado quando a propriedade <code>scroll</code> , <code>maxscroll</code> , <code>hscroll</code> , <code>maxhscroll</code> ou <code>bottomscroll</code> de um campo de texto é alterada.
<code>TextField.onSetFocus</code>	Chamado quando o campo de texto recebe o foco.

## Resumo de ouvintes do objeto TextField

Método	Descrição
<code>TextField.onChangeed</code>	Notificado quando o campo de texto é alterado.
<code>TextField.onScroller</code>	Notificado quando a propriedade <code>scroll</code> ou <code>maxscroll</code> de um campo de texto é alterada.

## TextField.\_alpha

### Disponibilidade

Flash Player 6.

### Uso

*TextField.\_alpha*

### Descrição

Propriedade; define ou recupera a transparência alfa (*valor*) do campo de texto especificado por *TextField*. A faixa de valores válidos vai de 0 (totalmente transparente) a 100 (totalmente opaco).

### Exemplo

Os comandos a seguir definem a propriedade `_alpha` de um campo de texto chamado de `text1` como 30%.

```
on(release) {  
    text1._alpha = 30;  
}
```

## TextField.addListener

### Disponibilidade

Flash Player 6.

### Uso

*TextField.addListener(novo\_Ouvinte)*

### Parâmetros

*novo\_Ouvinte* Um objeto com notificações dos eventos `onChangeed` e `onScroller`.

### Retorna

Nada.

### Descrição

Método; registra um objeto para receber notificações de eventos. Quando o evento `onChanged` ou `onScroller` ocorre, os eventos `TextField.onChangeed` e `TextField.onScroller` são chamados, seguidos dos métodos `onChangeed` e `onScroller` de objetos ouvintes registrados com `addListener`. Vários objetos podem ouvir notificações de alteração. Se o ouvinte *newListener* já estiver registrado, nenhuma alteração ocorrerá.

## TextField.autoSize

### Disponibilidade

Flash Player 6.

### Uso

*TextField*.autoSize

### Descrição

Propriedade; controla o alinhamento e o dimensionamento automáticos de campos de texto. Se o valor de dimensionamento automático for `"none"`, o campo de texto se comportará normalmente e não será redimensionado ou alinhado automaticamente para corresponder ao texto. Se o valor for `"left"`, o campo de texto expandirá ou contrairá seus lados direito e inferior para se ajustar a todo o texto contido. Os lados esquerdo e superior permanecem nas mesmas posições. Se o valor de dimensionamento automático for `"center"`, o campo de texto será dimensionado automaticamente, mas seu centro horizontal permanecerá ancorado na posição central horizontal original do campo de texto. O lado inferior ainda será expandido para se ajustar a todo o texto contido. Se o valor de dimensionamento automático for `"right"`, o campo de texto será dimensionado automaticamente, mas os lados esquerdo e inferior serão expandidos ou contraídos. Os lados superior e direito permanecem nas mesmas posições. Ao definir a propriedade `autoSize`, `true` será sinônimo de `"esquerda"` e `false` de `"nenhum"`.

### Exemplo

O exemplo a seguir define a propriedade `autosize` do campo de texto `textField2` como `"center"`.

```
textField2.autosize = "center";
```

## TextField.background

### Disponibilidade

Flash Player 6.

### Uso

*TextField*.background

### Descrição

Propriedade; se `true`, o campo de texto terá um preenchimento de fundo. Se `false`, o campo de texto não terá nenhum preenchimento de fundo.

## TextField.backgroundColor

### Disponibilidade

Flash Player 6.

### Uso

`TextField.backgroundColor`

### Descrição

Propriedade; a cor do fundo do campo de texto. O padrão é `0xFFFFFF` (branco). Esta propriedade pode ser recuperada ou definida, mesmo que, no momento, não haja fundo, mas a cor só será visível se o campo de texto tiver uma borda.

### Consulte também

`TextField.background`

## TextField.border

### Disponibilidade

Flash Player 6.

### Uso

`TextField.border`

### Descrição

Propriedade; se `true`, o campo de texto terá uma borda. Se `false`, o campo de texto não terá borda.

## TextField.borderColor

### Disponibilidade

Flash Player 6.

### Uso

`TextField.borderColor`

### Descrição

Propriedade; a cor da borda do campo de texto, o padrão é `0x000000` (preto). Esta propriedade pode ser recuperada ou definida, mesmo que não haja borda no momento.

### Consulte também

`TextField.border`

## TextField.bottomScroll

### Disponibilidade

Flash Player 6.

### Uso

`TextField.bottomScroll`

### Descrição

Propriedade (somente leitura); um inteiro (índice baseado em 1) que indica a linha mais inferior atualmente visível em *TextField*. Imagine o campo de texto como “uma janela” para um bloco de texto. A propriedade `TextField.scroll` é o índice baseado em 1 da linha visível mais superior na janela.

Todo o texto entre as linhas `TextField.scroll` e `TextField.bottomScroll` está visível atualmente no campo de texto.

## TextField.embedFonts

### Disponibilidade

Flash Player 6.

### Uso

`TextField.embedFonts`

### Descrição

Propriedade; um valor booleano que, quando `true`, exibe o campo de texto usando contornos de fontes incorporadas. Se `false`, exibirá o campo de texto usando fontes de dispositivo.

## TextField.\_focusrect

### Disponibilidade

Flash Player 6.

### Uso

`TextField._focusrect`

### Descrição

Propriedade; um valor booleano que especifica se um campo de texto tem um retângulo amarelo em torno dele quando está em foco.

## TextField.getDepth

### Disponibilidade

Flash Player 6.

### Uso

`TextField.getDepth`

### Parâmetros

Nenhum.

**Retorna**

Um inteiro.

**Descrição**

Método; retorna a espessura de um campo de texto.

## TextField.getFontList

**Disponibilidade**

Flash Player 6.

**Uso**

```
TextField.getFontList
```

**Parâmetros**

Nenhum.

**Retorna**

Uma matriz.

**Descrição**

Método; retorna um objeto Array cujos elementos são os nomes de todas as fontes no sistema host do Flash Player, inclusive fontes no arquivo SWF e em quaisquer arquivos SWF de elementos carregados. Os nomes são do tipo sequência de caracteres.

## TextField.getNewTextFormat

**Disponibilidade**

Flash Player 6.

**Uso**

```
TextField.getNewTextFormat()
```

**Parâmetros**

Nenhum.

**Retorna**

Um objeto TextFormat.

**Descrição**

Método; retorna um objeto TextFormat que contém uma cópia do objeto de formato de texto do campo de texto. O objeto de formato de texto é o formato recebido pelo texto recém-inserido, como o texto inserido com o método `replaceSel` ou o texto digitado por um usuário. Quando `getNewTextFormat` é chamado, o objeto TextFormat retornado tem todas as suas propriedades definidas. Nenhuma propriedade é `null`.

## TextField.getTextFormat

### Disponibilidade

Flash Player 6.

### Uso

```
TextField.getTextFormat()
```

```
TextField.getTextFormat (índice)
```

```
TextField.getTextFormat (início_Índice, fim_Índice)
```

### Parâmetros

*índice* Um inteiro que especifica um caractere em uma sequência de caracteres.

### Retorna

Um objeto.

### Descrição

Método; (Uso 1) retorna um objeto `TextFormat` que contém informações de formatação para todo o texto em um campo de texto. Somente as propriedades comuns a todo o texto no campo de texto são definidas no objeto `TextFormat` resultante. Qualquer propriedade que seja *misturada*, ou seja, que tenha valores diferentes em pontos diferentes no texto, tem seu valor definido como `null`.

Uso 2: retorna um objeto `TextFormat` que contém uma cópia do formato de texto do campo de texto em *índice*.

Uso 3: retorna um objeto `TextFormat` que contém informações de formatação para o intervalo de texto de *início\_Índice* a *fim\_Índice*.

### Consulte também

`TextField.getNewTextFormat`, `TextField.setNewTextFormat`, `TextField.setTextFormat`

## TextField.\_height

### Disponibilidade

Flash Player 6.

### Uso

```
TextField._height
```

### Descrição

Propriedade; define e recupera a altura do campo de texto, em pixels.

### Exemplo

O exemplo de código a seguir define a altura e a largura de um campo de texto.

```
myTextField._width = 200;  
myTextField._height = 200;
```



## TextField.\_highquality

### Disponibilidade

Flash Player 6.

### Uso

*TextField.\_highquality*

### Descrição

Propriedade (global); especifica o nível de sem serrilhado aplicado no filme atual. 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.

### Exemplo

```
_highquality = 1;
```

### Consulte também

*\_quality*, *toggleHighQuality*

## TextField.hscroll

### Disponibilidade

Flash Player 6.

### Uso

*TextField.hscroll*

### Retorna

Um inteiro.

### Descrição

Propriedade; indica a posição de rolagem horizontal atual. Se a propriedade *hscroll* for 0, o texto não será rolado horizontalmente.

### Exemplo

O exemplo a seguir rola o texto horizontalmente.

```
on (release) {  
    myTextField.hscroll += 1;  
}
```

### Consulte também

*TextField.maxhscroll*, *TextField.scroll*

## TextField.html

### Disponibilidade

Flash Player 6.

### Uso

*TextField.html*

### Descrição

Propriedade; um sinalizador que indica se o campo de texto contém uma representação HTML. Se a propriedade `html` for `true`, o campo de texto será HTML. Se `html` for `false`, o campo de texto será não-HTML.

### Consulte também

`TextField.htmlText`

## TextField.htmlText

### Disponibilidade

Flash Player 6.

### Uso

*TextField.htmlText*

### Descrição

Propriedade; se o campo de texto for HTML, esta propriedade conterá a representação HTML do conteúdo do campo de texto. Se o campo de texto não for HTML, ele se comportará da mesma maneira que a propriedade `text`. É possível especificar que um campo de texto seja HTML no Inspetor de propriedades ou ao configurar a propriedade `html` do campo de texto como `true`.

### Exemplo

No exemplo a seguir, o texto do campo de texto `text2` está em negrito.

```
text2.html = true;  
text2.htmlText = "<b>texto em negrito </b>";
```

## TextField.length

### Disponibilidade

Flash Player 6.

### Uso

*TextField.length*

### Descrição

Propriedade (somente leitura); indica o número de caracteres em um campo de texto. Esta propriedade retorna o mesmo valor que `text.length`, mas é mais rápida. Um caractere como tabulação ("`\t`"), por exemplo, conta como um caractere.

## TextField.maxChars

### Disponibilidade

Flash Player 6.

### Uso

*TextField.maxChars*

### Descrição

Propriedade; indica o número máximo de caracteres que o campo de texto pode conter. Um script pode inserir mais texto que o permitido por `maxChars`; a propriedade `maxChars` só indica quanto texto um usuário pode digitar. Se o valor desta propriedade for `null`, não haverá limite para a quantidade de texto que um usuário pode digitar.

## TextField.maxhscroll

### Disponibilidade

Flash Player 6.

### Uso

*TextField.maxhscroll*

### Descrição

Propriedade (somente leitura); indica o valor máximo de `TextField.hscroll`.

### Consulte também

`TextField.hscroll`

## TextField.maxscroll

### Disponibilidade

Flash Player 6.

### Uso

*TextField.maxscroll*

### Descrição

Propriedade (somente leitura); indica o valor máximo de `TextField.scroll`.

### Consulte também

`TextField.scroll`

## TextField.multiline

### Disponibilidade

Flash Player 6.

### Uso

*TextField.multiline*

### Descrição

Propriedade; indica se o campo de texto contém várias linhas. Se o valor for `true`, o campo de texto conterá várias linhas; se for `false`, ele será um campo de texto de uma única linha.

## TextField.\_name

### Disponibilidade

Flash Player 6.

### Uso

*TextField.\_name*

### Descrição

Propriedade; retorna o nome da instância do campo de texto especificado por *TextField*.

## TextField.onChanged

### Disponibilidade

Flash Player 6.

### Uso

*TextField.onChanged*

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Identificador de eventos; chamado quando o conteúdo de um campo de texto é alterado. Como padrão, é indefinido; é possível defini-lo em um script.

## TextField.onKillFocus

### Disponibilidade

Flash Player 6.

### Uso

```
TextField.onKillFocus = function (newFocus) {  
comandos;  
};
```

### Parâmetros

*newFocus* O objeto em foco.

### Retorna

Nada.

### Descrição

Identificador de eventos; um evento que é chamado quando um campo de texto perde o foco do teclado. O método *onKillFocus* recebe um parâmetro, *newFocus*, que é um objeto representando o novo objeto a receber o foco. Se nenhum objeto receber o foco, *newFocus* conterá o valor `null`.

## TextField.onScroller

### Disponibilidade

Flash Player 6.

### Uso

`TextField.onScroller`

### Descrição

Identificador de eventos; um evento que é chamado quando uma das propriedades de rolagem do campo de texto é alterada.

### Consulte também

`TextField.hscroll`, `TextField.maxhscroll`, `TextField.maxscroll`, `TextField.scroll`

## TextField.onSetFocus

### Disponibilidade

Flash Player 6.

### Uso

```
TextField.onSetFocus = function(oldFocus){  
comandos;  
};
```

### Parâmetros

*oldFocus* O objeto que perde o foco.

### Retorna

Nada.

### Descrição

Identificador de eventos; chamado quando um campo de texto recebe o foco do teclado. O parâmetro *oldFocus* é o objeto que perde o foco. Por exemplo, se o usuário pressionar a tecla Tab para mover o foco de entrada de um botão para um campo de texto, *oldFocus* conterá a instância do campo de texto.

Se nenhum objeto possuía o foco anteriormente, *oldFocus* conterá um valor `null`.

## TextField.\_parent

### Disponibilidade

Flash Player 6.

### Uso

```
_parent.property  
_parent._parent.property
```

### Descrição

Propriedade; especifica ou retorna uma referência ao clipe de filme ou objeto que contém o clipe de filme ou objeto atual. O objeto atual é o que contém o código ActionScript que faz referência a `_parent`. Use `_parent` para especificar um caminho relativo para clipes de filme ou objetos que estiverem acima do clipe de filme ou objeto atual.

### Consulte também

`_root`, `targetPath`

## TextField.password

### Disponibilidade

Flash Player 6.

### Uso

`TextField.password`

### Descrição

Propriedade; se o valor de `password` for `true`, o campo de texto será de senha e ocultará os caracteres de entrada. Se for `false`, o campo de texto não será de senha.

## TextField.\_quality

### Disponibilidade

Flash Player 6.

### Uso

`TextField._quality`

### Descrição

Propriedade (global); define ou recupera a qualidade usada para um filme. As fontes de dispositivo são sempre serrilhadas, sendo assim não são afetadas pela propriedade `_quality`.

A propriedade `_quality` pode ser definida nos seguintes valores:

- "LOW"   Qualidade baixa. Os gráficos não são apresentados sem serrilhado, os bitmaps não são suavizados.
- "MEDIUM"   Qualidade média. Os gráficos são apresentados sem serrilhado usando uma grade de 2 x 2, em pixels, 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 de 4 x 4, em pixels, e os bitmaps são suavizados quando o filme é estático. Essa é a configuração de qualidade padrão usada pelo Flash.
- "BEST"   Qualidade muito alta. Os gráficos são apresentados sem serrilhado usando uma grade de 4 x 4, em pixels, e os bitmaps sempre são suavizados.

### Exemplo

O exemplo a seguir define a qualidade como LOW:

```
textfield._quality = "LOW";
```

### Consulte também

`_highquality`, `toggleHighQuality`

## TextField.removeListener

### Disponibilidade

Flash Player 6.

### Uso

```
Selection.removeListener(ouvinte)
```

### Parâmetros

*ouvinte* O objeto que deixará de receber notificações de foco.

### Retorna

Se o *ouvinte* tiver sido removido com êxito, o método retornará um valor `true`. Se o *ouvinte* não tiver sido removido com êxito (por exemplo, se o *ouvinte* não estava na lista de ouvintes do objeto `TextField`), o método retornará o valor `false`.

### Descrição

Método; remove um objeto ouvinte anteriormente registrado para uma instância de campo de texto com `addListener`.

## TextField.removeTextField

### Disponibilidade

Flash Player 6.

### Uso

```
TextField.removeTextField()
```

### Descrição

Método; remove o campo de texto especificado por *TextField*. Esta operação só pode ser executada em um campo de texto que tenha sido criado com o método `createTextField` do objeto `MovieClip`. Ela não funcionará em campos de texto inseridos pela Linha de tempo. Quando este método é chamado, o campo de texto é instruído a se remover. É semelhante ao método `MovieClip.removeMovieClip`.

### Consulte também

`MovieClip.createTextField`

## TextField.replaceSel

### Disponibilidade

Flash Player 6.

### Uso

```
TextField.replaceSel(texto)
```

### Parâmetros

*texto* Uma seqüência de caracteres.

### Retorna

Nada.

### Descrição

Método; substitui a seleção atual pelo conteúdo do parâmetro *texto*. O texto é inserido na posição da seleção atual, usando o formato de caracteres padrão atual e o formato de parágrafo padrão. O texto não é tratado como HTML, mesmo que o campo de texto seja HTML.

É possível usar o método `replaceSel` para inserir e excluir textos sem interromper a formatação de caractere e parágrafo do resto do texto.

## TextField.restrict

### Disponibilidade

Flash Player 6.

### Uso

*TextField.restrict*

### Descrição

Propriedade; indica o conjunto de caracteres que um usuário pode digitar no campo de texto. Se o valor da propriedade `restrict` for `null`, será possível digitar qualquer caractere. Se o valor da propriedade `restrict` for uma sequência de caracteres vazia, não será possível digitar nenhum caractere. Se o valor da propriedade `restrict` for uma sequência de caracteres, só será possível digitar os caracteres da sequência no campo de texto. A sequência de caracteres é rastreada da esquerda para a direita. É possível especificar um intervalo usando o traço (-). Isso restringe apenas a interação com o usuário; um script pode colocar qualquer texto no campo de texto. Esta propriedade não é sincronizada com as caixas de seleção Incorporar contornos de fonte no Inspetor de propriedades.

Se a sequência de caracteres começar com `^`, todos os caracteres serão aceitos inicialmente e os caracteres sucessores na sequência serão excluídos do conjunto de caracteres aceitos. Se a sequência de caracteres não começar com `^`, nenhum caractere será aceito inicialmente e os caracteres sucessores na sequência serão incluídos no conjunto de caracteres aceitos.

### Exemplo

O exemplo a seguir só permite que sejam digitados caracteres em maiúsculas, espaços e números em um campo de texto:

```
my_txt.restrict = "A-Z 0-9";
```

O exemplo a seguir inclui todos os caracteres, mas exclui letras minúsculas:

```
my_txt.restrict = "^a-z";
```

É possível usar uma barra invertida para digitar um `^` ou - literalmente. As sequências de barra invertida aceitas são `\`, `^` ou `\\`. A barra invertida deve ser um caractere real na sequência de caracteres para que, quando especificada no ActionScript, seja usada uma barra invertida dupla. Por exemplo, o código a seguir inclui somente o traço (-) e o circunflexo (^).

```
my_txt.restrict = "\\-\\^";
```

O `^` pode ser usado em qualquer parte da sequência de caracteres para alternar entre caracteres incluídos e caracteres excluídos. O código a seguir contém somente letras maiúsculas, mas exclui a letra maiúscula Q.

```
my_txt.restrict = "A-Z^Q";
```

É possível usar a sequência de escape `\u` para construir sequências de caracteres `restrict`. O código a seguir contém somente os caracteres de ASCII 32 (espaço) a ASCII 126 (til).

```
my_txt.restrict = "\u0020-\u007E";
```



## TextField.\_rotation

### Disponibilidade

Flash Player 6.

### Uso

*TextField.\_rotation*

### Descrição

Propriedade; especifica a rotação do campo de texto em graus.

## TextField.scroll

### Disponibilidade

Flash Player 6.

### Uso

*TextField.scroll*

### Descrição

Propriedade; define a posição vertical do texto em um 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.

### Exemplo

O código a seguir é anexado a um botão Para cima que rola pelo campo de texto `myText`.

```
on (release) {  
    myText.scroll = myText.scroll + 1;  
}
```

### Consulte também

`TextField.maxscroll`, `TextField.scroll`

## TextField.selectable

### Disponibilidade

Flash Player 6.

### Uso

*TextField.selectable*

### Descrição

Propriedade; um valor booleano que indica se o campo de texto pode ser selecionado. O valor `true` indica que o texto pode ser selecionado.

## TextField.setNewTextFormat

### Disponibilidade

Flash Player 6.

### Uso

```
TextField.setNewTextFormat(textFormat)
```

### Parâmetros

*textFormat* Uma instância do objeto TextFormat.

### Retorna

Nada.

### Descrição

Método; define um objeto TextFormat para o texto recém-inserido, como o texto inserido com o método `replaceSel` ou o texto digitado por um usuário em um campo de texto. Cada campo de texto tem um novo formato de texto. Quando o texto é inserido, o novo formato de texto é atribuído ao novo texto.

O formato de texto é definido em uma nova instância do objeto TextFormat. Ele contém informações de formatação de caractere e parágrafo. As informações sobre formatação de caractere descrevem a aparência de caracteres individuais. Por exemplo: o nome da fonte, o tamanho do ponto, a cor e a URL associada. As informações sobre formatação de parágrafo descrevem a aparência de um parágrafo. Por exemplo: margem esquerda, margem direita, recuo da primeira linha e alinhamento esquerdo, direito e centralizado.

### Consulte também

`TextField.getNewTextFormat`, `TextField.getTextFormat`, `TextField.setTextFormat`

## TextField.setTextFormat

### Disponibilidade

Flash Player 6.

### Uso

```
TextField.setTextFormat (textFormat)
```

```
TextField.setTextFormat (índice, textFormat)
```

```
TextField.setTextFormat (início_Índice, fim_Índice, textFormat)
```

### Parâmetros

*início\_Índice* Um inteiro.

*fim\_Índice* Um inteiro que especifica o primeiro caractere após o intervalo de texto desejado.

*textFormat* Uma instância do objeto TextFormat. Um objeto TextFormat que contém informações de formatação de caractere e parágrafo.

### Retorna

Nada.

### Descrição

Método; define um objeto de formato de texto para um intervalo especificado de texto em um campo de texto. É possível atribuir um formato de texto a cada caractere de um campo de texto. O formato de texto do primeiro caractere de um parágrafo é examinado para realizar formatação de parágrafo para todo o parágrafo. O método `setTextFormat` altera o formato de texto aplicado a caracteres isoladamente, a grupos de caracteres ou a todo o corpo de texto em um campo de texto.

O formato de texto é definido em uma nova instância do objeto `TextFormat`. Ele contém informações de formatação de caractere e parágrafo. As informações de formatação de caractere descrevem a aparência dos caracteres. Por exemplo: o nome da fonte, o tamanho do ponto, a cor e a URL associada. As informações de formatação de parágrafo descrevem a aparência de um parágrafo. Por exemplo: margem esquerda, margem direita, recuo da primeira linha e alinhamento esquerdo, direito e centralizado.

Uso 1: aplica as propriedades de `textFormat` a todo o texto no campo de texto.

Uso 2: aplica as propriedades de `textFormat` ao caractere na posição `indice`.

Uso 3: aplica as propriedades do parâmetro `textFormat` ao intervalo de texto do parâmetro `início_Índice` ao parâmetro `fim_Índice`.

### Exemplo

Este exemplo cria um novo objeto `TextFormat` chamado `myTextFormat` e define sua propriedade `bold` como `true`. Em seguida, chama o método `setTextFormat` e aplica o novo formato de texto ao campo de texto `myTextField`.

```
myTextFormat = new TextFormat();
myTextFormat.bold = true;
myTextField.setTextFormat(myTextFormat);
```

### Consulte também

`TextFormat` (objeto)

## TextField.\_soundbuftime

### Disponibilidade

Flash Player 6.

### Uso

`TextField._soundbuftime`

### Descrição

Propriedade (global); um inteiro que especifica o número de segundos em que um som é armazenado em pré-buffer antes de começar a fluir.

## TextField.tabEnabled

### Disponibilidade

Flash Player 6.

### Uso

*TextField.tabEnabled*

### Descrição

Propriedade; pode ser definida em uma instância dos objetos MovieClip, Button ou TextField. Por padrão, não é definido.

Se a propriedade `tabEnabled` for `undefined` ou tiver um valor `true`, o objeto será incluído na ordenação de tabulação automática, e será incluído na ordenação de tabulação personalizada se a propriedade `tabIndex` também for definida como um valor. Se `tabEnabled` for `false`, o objeto não será incluído na ordenação automática de guias. No caso de um clipe de filme, se `tabEnabled` for `false`, os filhos do clipe de filme ainda poderão ser incluídos na ordenação automática de guias, a menos que a propriedade `tabChildren` também seja definida como `false`.

Se `tabEnabled` for `undefined` ou `true`, o objeto será incluído na ordenação de tabulação personalizada se a propriedade `tabIndex` for definida. Se `tabEnabled` for `false`, então o objeto não será incluído na ordenação de guia personalizada, ainda que a propriedade `tabIndex` seja definida. Se `tabEnabled` for definido como `false` em um clipe de filme, os filhos do clipe de filme ainda poderão ser incluídos na ordenação de tabulação personalizada.

## TextField.tabIndex

### Disponibilidade

Flash Player 6.

### Uso

*TextField.tabIndex*

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Propriedade; permite personalizar a ordenação de guias dos objetos em um filme. É possível definir a propriedade `tabIndex` em um botão, clipe de filme ou instância de campo de texto. Por padrão, ela é `undefined`.

Se algum objeto sendo exibido atualmente no filme do Flash tiver uma propriedade `tabIndex`, a ordenação de guia automática será desativada e a ordenação de guia será calculada nas propriedades `tabIndex` de objetos do filme. A ordenação personalizada de guias inclui apenas os objetos que têm propriedades `tabIndex`.

A propriedade `tabIndex` deve ser um inteiro positivo. Os objetos são ordenados de acordo com suas propriedades `tabIndex`, em ordem ascendente. Um objeto com um `tabIndex` 1 vem antes de um objeto com `tabIndex` 2. Se dois objetos tiverem o mesmo valor `tabIndex`, aquele que preceder o outro na ordenação de tabulação será `undefined`.

A ordenação de tabulação personalizada definida pela propriedade `tabIndex` é simples. Isso significa que as relações hierárquicas de objetos são ignoradas no filme do Flash. Todos os objetos no filme do Flash com propriedades `tabIndex` são colocados na ordem de guia. Por sua vez, essa é determinada pela ordem dos valores de `tabIndex`. Se dois objetos tiverem o mesmo valor `tabIndex`, o primeiro será `undefined`. Você não deve usar o mesmo valor de `tabIndex` para vários objetos.

## TextField.\_target

### Disponibilidade

Flash Player 6.

### Uso

*TextField.\_target*

### Descrição

Propriedade (somente leitura); retorna o caminho de destino da instância de campo de texto especificada no parâmetro *TextField*.

## TextField.text

### Disponibilidade

Flash Player 6.

### Uso

*TextField.text*

### Descrição

Propriedade; indica o texto atual no campo de texto. As linhas são separadas pelo caractere de retorno de carro ('\r', ASCII 13). Esta propriedade contém o texto normal não formatado no campo de texto, sem marcas HTML, mesmo que o campo de texto seja HTML.

### Consulte também

*TextField.htmlText*

## TextField.textColor

### Disponibilidade

Flash Player 6.

### Uso

*TextField.textColor*

### Descrição

Propriedade; indica a cor do texto em um campo de texto.

## TextField.textHeight

### Disponibilidade

Flash Player 6.

### Uso

*TextField.textHeight*

### Descrição

Propriedade; indica a altura do texto.

## TextField.textWidth

### Disponibilidade

Flash Player 6.

### Uso

*TextField*.textWidth

### Descrição

Propriedade; indica a largura do texto.

## TextField.type

### Disponibilidade

Flash Player 6.

### Uso

*TextField*.type

### Descrição

Propriedade; especifica o tipo de campo de texto. Há dois valores: "dynamic", que especifica um campo de texto dinâmico (não pode ser editado pelo usuário) e "input", que especifica um campo de texto de entrada.

### Exemplo

```
TextField.type = "dynamic";
```

## TextField.\_url

### Disponibilidade

Flash Player 6.

### Uso

*TextField*.\_url

### Descrição

Propriedade (somente leitura); recupera a URL do arquivo SWF que criou o campo de texto.

## TextField.variable

### Disponibilidade

Flash Player 6.

### Uso

*TextField*.\_variable

### Descrição

Propriedade; o nome da variável à qual o campo de texto está associado. O tipo dessa propriedade é String.

## TextField.\_visible

### Disponibilidade

Flash Player 6.

### Uso

*TextField.\_visible*

### Descrição

Propriedade; um valor booleano que indica se o campo de texto especificado pelo parâmetro *TextField* é visível. Os campos de texto que não são visíveis (propriedade *\_visible* definida como *false*) são desativados.

## TextField.\_width

### Disponibilidade

Flash Player 6.

### Uso

*TextField.\_width*

### Descrição

Propriedade; define e recupera a largura do campo de texto, em pixels.

### Exemplo

O exemplo a seguir define as propriedades de altura e largura de um campo de texto:

```
myTextField._width=200;  
myTextField._height=200;
```

### Consulte também

*MovieClip.\_height*

## TextField.wordWrap

### Disponibilidade

Flash Player 6.

### Uso

*TextField.wordWrap*

### Descrição

Propriedade; um valor booleano que indica se o campo de texto tem quebra automática de linha. Se o valor de *wordWrap* for *true*, o campo de texto terá quebra automática de linha; se o valor for *false*, o campo de texto não terá quebra automática de linha.

## TextField.\_x

### Disponibilidade

Flash Player 6.

### Uso

*TextField.\_x*

### Descrição

Propriedade; um inteiro que define a coordenada *x* de um campo de texto em relação às coordenadas locais do clipe de filme pai. Se um campo de texto estiver na Linha de tempo principal, seu sistema de coordenadas será referente ao canto superior esquerdo do Palco como (0, 0). Se o campo de texto estiver dentro de um clipe de filme que tenha transformações, o campo de texto estará no sistema de coordenadas local do clipe de filme anexado. Assim, para um clipe de filme girado 90° no sentido anti-horário, o campo de texto anexado herda um sistema de coordenadas que é girado 90° no sentido anti-horário. As coordenadas do campo de texto referem-se à posição do ponto de registro.

### Consulte também

*TextField.\_xscale*, *TextField.\_y*, *TextField.\_yscale*

## TextField.\_xmouse

### Disponibilidade

Flash Player 6.

### Uso

*TextField.\_xmouse*

### Descrição

Propriedade (somente leitura); retorna a coordenada *x* da posição do mouse relativa ao campo de texto.

### Consulte também

*TextField.\_ymouse*

## TextField.\_xscale

### Disponibilidade

Flash Player 6.

### Uso

*TextField.\_xscale*

### Descrição

Propriedade; determina a escala horizontal (*porcentagem*) do campo de texto como aplicado no ponto de registro do campo de texto. O ponto de registro padrão é (0,0).

### Consulte também

*TextField.\_x*, *TextField.\_y*, *TextField.\_yscale*



## TextField.\_y

### Disponibilidade

Flash Player 6.

### Uso

*TextField.\_y*

### Descrição

Propriedade; define a coordenada *y* de um campo de texto relativa às coordenadas locais do clipe de filme pai. Se um campo de texto estiver na Linha de tempo principal, seu sistema de coordenadas será referente ao canto superior esquerdo do Palco como (0, 0). Se o campo de texto estiver dentro de outro clipe de filme que tenha transformações, o campo de texto estará no sistema de coordenadas local do clipe de filme anexado. Assim, para um clipe de filme girado 90° no sentido anti-horário, o campo de texto anexado herda um sistema de coordenadas que é girado 90° no sentido anti-horário. As coordenadas do campo de texto referem-se à posição do ponto de registro.

### Consulte também

*TextField.\_x*, *TextField.\_xscale*, *TextField.\_yscale*

## TextField.\_ymouse

### Disponibilidade

Flash Player 6.

### Uso

*TextField.\_ymouse*

### Descrição

Propriedade (somente leitura); indica a coordenada *y* da posição do mouse relativa ao campo de texto.

### Consulte também

*TextField.\_xmouse*

## TextField.\_yscale

### Disponibilidade

Flash Player 6.

### Uso

*TextField.\_yscale*

### Descrição

Propriedade; define a escala vertical (*porcentagem*) do campo de texto conforme aplicado no ponto de registro do campo de texto. O ponto de registro padrão é (0,0).

### Consulte também

*TextField.\_x*, *TextField.\_xscale*, *TextField.\_y*

## TextFormat (objeto)

O objeto `TextFormat` representa informações de formatação de caractere.

Use o construtor `new TextFormat` para criar uma instância do objeto `TextFormat` antes de chamar seus métodos.

É possível definir parâmetros `TextFormat` como `null` para indicar que eles são indefinidos. Quando um objeto `TextFormat` é aplicado a um campo de texto usando o método `setTextFormat`, apenas as propriedades definidas são aplicadas, como no seguinte exemplo:

```
myTextFormat = new TextFormat();
myTextFormat.bold = true;
myTextField.setTextFormat(myTextFormat);
```

Este código cria primeiro um objeto `TextFormat` vazio com todas as suas propriedades indefinidas; em seguida, define a propriedade `bold` como um valor definido.

O código `myTextField.setTextFormat(myTextFormat)` só altera a propriedade `bold` do formato de texto padrão do campo de texto, pois a propriedade `bold` é a única definida em `myTextFormat`. Todos os outros aspectos do formato de texto padrão do campo de texto permanecem inalterados.

Quando `getTextFormat` é chamado, um objeto `TextFormat` é retornado com todas as propriedades definidas; nenhuma propriedade é `null`.

## Resumo de métodos do objeto TextFormat

Método	Descrição
<code>TextFormat.getTextExtent</code>	Retorna um objeto com duas propriedades, <code>width</code> e <code>height</code> , que indicam o tamanho de um texto em um campo de texto.

## Resumo das propriedades do objeto TextFormat

Propriedade	Descrição
<code>TextFormat.align</code>	Indica o alinhamento de um parágrafo.
<code>TextFormat.blockIndent</code>	Indica o recuo de bloco em pontos.
<code>TextFormat.bold</code>	Indica se o texto está em negrito.
<code>TextFormat.bullet</code>	Indica se o texto está ou não em uma lista com marcadores.
<code>TextFormat.color</code>	Indica a cor do texto.
<code>TextFormat.font</code>	Indica o nome da fonte do texto com um formato de texto.
<code>TextFormat.indent</code>	Indica o recuo da margem esquerda ao primeiro caractere no parágrafo.
<code>TextFormat.italic</code>	Indica se o texto está em itálico.
<code>TextFormat.leading</code>	Indica a quantidade de espaço vertical entre as linhas.
<code>TextFormat.leftMargin</code>	Indica a margem esquerda do parágrafo, em pontos.
<code>TextFormat.rightMargin</code>	Indica a margem direita do parágrafo, em pontos.
<code>TextFormat.tabStops</code>	Especifica interrupções de tabulação personalizadas.
<code>TextFormat.target</code>	Indica a janela em um navegador na qual um hiperlink é exibido.
<code>TextFormat.size</code>	Indica o tamanho do ponto do texto.
<code>TextFormat.underline</code>	Indica se o texto está sublinhado.
<code>TextFormat.url</code>	Indica o URL ao qual o texto está vinculado.

## Construtor do objeto TextFormat

### Uso

```
new TextFormat([fonte, [tamanho, [cor, [negrito, [itálico, [sublinhado, [url,  
[destino, [alinhamento, [margem_Esquerda, [margem_Direita, [recuo,  
[entrelinhamento]]]]]]]]]]))
```

### Parâmetros

*fonte* O nome de uma fonte de texto como uma seqüência de caracteres.

*tamanho* Um inteiro que indica o tamanho do ponto.

*cor* A cor do texto que usa esse formato de texto. Um número que contém três componentes RGB de 8 bits; por exemplo, 0xFF0000 é vermelho, 0x00FF00 é verde.

*negrito* Um valor Booleano que indica se o texto está em negrito.

*itálico* Um valor Booleano que indica se o texto está em itálico.

*sublinhado* Um valor Booleano que indica se o texto está sublinhado.

*url* O URL ao qual o texto neste formato de texto se vincula por hiperlink. Se *url* for uma seqüência de caracteres vazia, o texto não terá um hiperlink.

*destino* A janela de destino em que o hiperlink é exibido. Se a janela de destino for uma seqüência de caracteres vazia, o texto será exibido na janela de destino padrão `_self`. Se a propriedade `TextFormat.url` for definida como uma seqüência de caracteres vazia ou como o valor `null`, ela poderá ser obtida ou definida, mas não terá efeito.

*alinhamento* O alinhamento do parágrafo, representado como uma seqüência de caracteres. Se for `"left"`, o parágrafo será alinhado à esquerda. Se for `"center"`, o parágrafo será centralizado. Se for `"right"`, o parágrafo será alinhado à direita.

*margem\_Esquerda* Indica a margem esquerda do parágrafo, em pontos.

*margem\_Direita* Indica a margem direita do parágrafo, em pontos.

*recuo* Um inteiro que indica o recuo da margem esquerda ao primeiro caractere no parágrafo.

*entrelinhamento* Um número que indica a quantidade de espaço vertical entre as linhas.

### Descrição

Construtor; cria uma instância do objeto `TextFormat` com as propriedades especificadas. Permite alterar as propriedades do objeto `TextFormat` para alterar a formatação de campos de texto.

É possível definir qualquer parâmetro como o valor `null` para indicar que não está definido. Todos os parâmetros são opcionais; qualquer parâmetro omitido é tratado como `null`.

### Disponibilidade

Flash Player 6.

## TextFormat.align

### Disponibilidade

Flash Player 6.

### Uso

*TextFormat.align*

### Descrição

Propriedade; indica o alinhamento do parágrafo, representado como uma sequência de caracteres. O alinhamento do parágrafo, representado como uma sequência de caracteres. Se for "left", o parágrafo será alinhado à esquerda. Se for "center", o parágrafo será centralizado. Se for "right", o parágrafo será alinhado à direita. O valor padrão é `null`, o que indica que a propriedade é indefinida.

## TextFormat.blockIndent

### Disponibilidade

Flash Player 6.

### Uso

*TextFormat.blockIndent*

### Descrição

Propriedade; um número que indica o recuo de bloco em pontos. O recuo de bloco é aplicado a um bloco de texto inteiro; ou seja, a todas as linhas do texto. Por outro lado, o recuo normal (*TextFormat.indent*) só afeta a primeira linha de cada parágrafo. Se esta propriedade for `null`, o objeto *TextFormat* não especificará um recuo de bloco.

## TextFormat.bold

### Disponibilidade

Flash Player 6.

### Uso

*TextFormat.bold*

### Descrição

Propriedade; um valor Booleano que indica se o texto está em negrito. O valor padrão é `null`, o que indica que a propriedade é indefinida.

## TextFormat.bullet

### Disponibilidade

Flash Player 6.

### Uso

*TextFormat.bullet*

### Descrição

Propriedade; um valor booleano que indica que o texto é parte de uma lista com marcadores. Em uma lista com marcadores, cada parágrafo de texto é recuado. À esquerda da primeira linha de cada parágrafo, é exibido um símbolo de marcador. Se esta propriedade for `null`, o objeto *TextFormat* não especificará que o texto tenha ou não marcadores.

## TextFormat.color

### Disponibilidade

Flash Player 6.

### Uso

*TextFormat.color*

### Descrição

Propriedade; indica a cor do texto. Um número que contém três componentes RGB de 8 bits; por exemplo, 0xFF0000 é vermelho, 0x00FF00 é verde.

## TextFormat.font

### Disponibilidade

Flash Player 6.

### Uso

*TextFormat.font*

### Descrição

Propriedade; o nome da fonte do texto nesse formato, como uma sequência de caracteres. O valor padrão é `null`, o que indica que a propriedade é indefinida.

## TextFormat.getTextExtent

### Disponibilidade

Flash Player 6.

### Uso

*TextFormat.getTextExtent* (*texto*)

### Parâmetros

*texto* Uma sequência de caracteres.

### Retorna

Um objeto com as propriedades `width` e `height`.

### Descrição

Método; retorna o tamanho da sequência de caracteres de texto especificada no parâmetro *texto* nesse formato de caractere. O valor retornado é um objeto da classe `Object` com duas propriedades, `width` e `height`. O *texto* é tratado como texto regular (não HTML). O *texto* é uma única linha de texto; os retornos de carro e alimentações de linha são ignorados e nenhuma quebra automática de linha é aplicada.

## TextFormat.indent

### Disponibilidade

Flash Player 6.

### Uso

*TextFormat.indent*

### Descrição

Propriedade; um inteiro que indica o recuo da margem esquerda ao primeiro caractere no parágrafo. O valor padrão é `null`, o que indica que a propriedade é indefinida.

## TextFormat.italic

### Disponibilidade

Flash Player 6.

### Uso

*TextFormat.italic*

### Descrição

Propriedade; um valor booleano que indica se o texto nesse formato está em itálico. O valor padrão é `null`, o que indica que a propriedade é indefinida.

## TextFormat.leading

### Disponibilidade

Flash Player 6.

### Uso

*TextFormat.leading*

### Descrição

Propriedade; a quantidade de espaço vertical entre as linhas. O valor padrão é `null`, o que indica que a propriedade é indefinida.

## TextFormat.leftMargin

### Disponibilidade

Flash Player 6.

### Uso

*TextFormat.leftMargin*

### Descrição

Propriedade; a margem esquerda do parágrafo, em pontos. O valor padrão é `null`, o que indica que a propriedade é indefinida.

## TextFormat.rightMargin

### Disponibilidade

Flash Player 6.

### Uso

*TextFormat.rightMargin*

### Descrição

Propriedade; a margem direita do parágrafo, em pontos. O valor padrão é `null`, o que indica que a propriedade é indefinida.

## TextFormat.size

### Disponibilidade

Flash Player 6.

### Uso

*TextFormat.size*

### Descrição

Propriedade; o tamanho do ponto do texto nesse formato. O valor padrão é `null`, o que indica que a propriedade é indefinida.

## TextFormat.tabStops

### Disponibilidade

Flash Player 6.

### Uso

*TextFormat.tabStops*

### Descrição

Propriedade; especifica interrupções de tabulação personalizadas como uma Matriz de inteiros não negativos. Cada interrupção de tabulação é especificada em pontos. Se as interrupções de tabulação personalizadas não forem especificadas (`null`), a interrupção de tabulação padrão será 4 (largura média de caractere).

## TextFormat.target

### Disponibilidade

Flash Player 6.

### Uso

*TextFormat.target*

### Descrição

Propriedade; indica a janela de destino em que o hiperlink é exibido. Se a janela de destino for uma sequência de caracteres vazia, o texto será exibido na janela de destino padrão `_self`. Se a propriedade `TextFormat.url` for definida como uma sequência de caracteres vazia ou como o valor `null`, ela poderá ser obtida ou definida, mas não terá efeito.

## TextFormat.underline

### Disponibilidade

Flash Player 6.

### Uso

*TextFormat.underline*

### Descrição

Propriedade; um valor Booleano que indica se o texto que usa este TextFormat está sublinhado. Se *underline* estiver definido como `true`, o texto nesse formato estará sublinhado. Se estiver definido como `false`, o texto nesse formato não estará sublinhado. Note que este é o mesmo sublinhado obtido pela marca `<U>`, que não é o sublinhado "verdadeiro", pois não ignora os descendentes corretamente. O valor padrão é `null`, o que indica que a propriedade é indefinida.

## TextFormat.url

### Disponibilidade

Flash Player 6.

### Uso

*TextFormat.url*

### Descrição

Propriedade; indica a URL à qual o texto nesse formato está vinculado. Se a propriedade *url* for uma seqüência de caracteres vazia, o texto não terá um hiperlink. O valor padrão é `null`, o que indica que a propriedade é indefinida.

## this

### Disponibilidade

Flash Player 5.

### Uso

*this*

### Descrição

Palavra-chave; faz referência a uma instância de objeto ou de clipe de filme. Quando um script é executado, *this* faz referência à instância do clipe de filme que contém o script. Quando um método é chamado, *this* contém uma referência ao objeto que contém o método chamado.

Dentro de uma ação identificadora de eventos *on* anexada a um botão, *this* refere-se à Linha de tempo que contém o botão. Dentro de uma ação identificadora de eventos *onClipEvent* anexada a um clipe de filme, *this* refere-se à Linha de tempo do próprio clipe de filme.

### 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`

## **toggleHighQuality**

### **Disponibilidade**

Flash 2.

### **Uso**

```
toggleHighQuality()
```

### **Parâmetros**

Nenhum.

### **Retorna**

Nada.

### **Descrição**

Ação; ativa e 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.

### **Exemplo**

O código a seguir pode ser aplicado a um botão que, quando clicado, ative e desative o modo sem serrilhado.

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

### **Consulte também**

`_quality`, `_highquality`

# trace

## Disponibilidade

Flash Player 4.

## Uso

`trace(expressão)`

## Parâmetros

*expressão* Uma expressão a ser avaliada. Quando um arquivo SWF é aberto na ferramenta de criação Flash (através do comando Testar filme), o valor do parâmetro *expressão* é exibido na janela Saída.

## Retorna

Nada.

## Descrição

Ação; avalia a *expressão* e exibe os resultados na janela Saída no modo de teste.

Use `trace` para registrar notas 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.

É possível usar o comando Omitir ações de traçagem em Configurações de publicação para remover ações `trace` do arquivo SWF exportado.

## 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 no fim do script para avaliar o local do clipe de filme `rabbi` e exibe o resultado 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();
}
```

## true

### Disponibilidade

Flash Player 5.

### Uso

true

### Descrição

Um valor booleano exclusivo que representa o oposto de false.

### Consulte também

false

## typeof

### Disponibilidade

Flash Player 5.

### Uso

typeof *expressão*

### Parâmetros

*expressão* Sequência de caracteres, clipe de filme, botão, objeto ou função.

### Descrição

Operador; um operador unário colocado antes de um único parâmetro. O operador typeof faz com que o interpretador Flash avalie *expressão*; o resultado é uma sequência de caracteres que especifica se a expressão é uma sequência de caracteres, um clipe de filme, um objeto ou uma função. A tabela a seguir mostra os resultados do operador typeof em cada tipo de expressão:

Parâmetro	Saída
String	seqüência de caracteres
MovieClip	movieclip
Button	objeto
Campo de texto	objeto
Number	number
Booleano	boolean
Objeto	objeto
Função	function

# undefined

## Disponibilidade

Flash Player 5.

## Uso

undefined

## Parâmetros

Nenhum.

## Retorna

Nada.

## Descrição

Um valor especial, geralmente usado para indicar que um valor ainda não foi atribuído a uma variável. Uma referência a um valor indefinido retorna o valor especial `undefined`. O código do `ActionScript` `typeof(undefined)` retorna a sequência de caracteres `"undefined"`. O único valor do tipo `undefined` é `undefined`.

Quando `undefined` é convertido em uma sequência de caracteres, ele é convertido na sequência vazia.

O valor `undefined` é semelhante ao valor especial `null`. De fato, quando `null` e `undefined` são comparados ao operador de igualdade, eles são comparados como iguais.

## Exemplo

Neste exemplo, a variável `x` não foi declarada e, portanto, tem o valor `undefined`. Na primeira seção de código, o operador de igualdade (`==`) compara o valor `x` ao valor `undefined` e o resultado apropriado é enviado à janela Saída. Na segunda seção de código, o operador de igualdade compara os valores `null` e `undefined`.

```
// x não foi declarado
trace ("The value of x is " + x);
if (x == undefined) {
    trace ("x is undefined");
} else {
    trace ("x is not undefined");
}

trace ("typeof (x) is " + typeof (x));
if (null == undefined) {
    trace ("null and undefined are equal");
} else {
    trace ("null and undefined are not equal");
}
```

O seguinte resultado é exibido na janela Saída:

```
The value of x is x is undefined
typeof (x) is undefined
null and undefined are equal
```

**Observação:** Na especificação ECMA-262, `undefined` é convertido na sequência de caracteres `"undefined"`, e não na sequência de caracteres vazia. Esta é uma diferença entre o `ActionScript` e a especificação ECMA-262.

## unescape

### Disponibilidade

Flash Player 5.

### Uso

`unescape(x)`

### Parâmetros

*x* Uma sequência de caracteres com seqüências hexadecimais de escape.

### Retorna

Nada.

### Descrição

Função de alto nível; avalia o parâmetro *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 a seqüência de caracteres.

### Exemplo

O exemplo a seguir ilustra o processo de conversão de 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

### Disponibilidade

Flash Player 3.

### Uso

`unloadMovie[Num](nível/"destino")`

### Parâmetros

*nível* O nível (`_levelN`) de um filme carregado. Quando um filme é descarregado de um nível, a ação `unloadMovie` no painel Ações no modo Normal muda para `unloadMovieNum`; no modo Especialista, é necessário especificar `unloadMovieNum` ou escolhê-lo na caixa de ferramentas Ações.

*destino* O caminho de destino de um clipe de filme.

### Retorna

Nenhum.

### Descrição

Ação; remove um filme carregado ou um clipe de filme do Flash Player. Para descarregar um filme que tenha sido carregado em um nível do Flash Player, use o parâmetro *nível*. Para descarregar um clipe de filme carregado, use o parâmetro *destino*.

**Exemplo**

O exemplo a seguir descarrega o clipe de filme `draggable` na Linha de tempo principal e carrega o filme `movie.swf` no nível 4.

```
on (press) {  
    unloadMovie ("_root.draggable");  
    loadMovieNum ("movie.swf", 4);  
}
```

O exemplo a seguir descarrega o filme carregado no nível 4:

```
on (press) {  
    unloadMovieNum (4);  
}
```

**Consulte também**

`loadMovie`, `loadMovieNum`, `unloadMovieNum`

## unloadMovieNum

**Disponibilidade**

Flash Player 3.

**Uso**

```
unloadMovieNum(nível)
```

**Parâmetros**

*nível* O nível (`_levelN`) de um filme carregado.

**Retorna**

Nada.

**Descrição**

Ação; remove um filme carregado do Flash Player.

**Consulte também**

`loadMovie`, `loadMovieNum`

## updateAfterEvent

**Disponibilidade**

Flash Player 5.

**Uso**

```
updateAfterEvent()
```

**Parâmetros**

Nenhum.

**Retorna**

Nada.

### Descrição

Ação; atualiza a exibição (independentemente dos quadros por segundo definidos para o filme) quando é chamada dentro de um identificador `onClipEvent` ou como parte de uma função ou método passado para `setInterval`. O Flash ignora as chamadas para `updateAfterEvent` que não estejam dentro de um identificador `onClipEvent` ou que não sejam parte de uma função ou método passado para `setInterval`.

### Consulte também

`onClipEvent`, `setInterval`

## var

### Disponibilidade

Flash Player 5.

### Uso

```
var variableName1 [= valor1] [...,variableNameN [=valorN]]
```

### Parâmetros

*variableName* Um identificador.

*valor* O valor atribuído à variável.

### Retorna

Nada.

### 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 são declaradas em um bloco e a lista de ações atuais não foi executada com a ação `call`, as variáveis não são locais.

### Exemplo

Os exemplos a seguir usam a ação `var` para declarar e atribuir variáveis:

```
var x;  
var y = 1;  
var z = 3, w = 4;  
var s, t, u = z;
```

## void

### Disponibilidade

Flash Player 5.

### Uso

```
void (expressão)
```

### Descrição

Operador; um operador unário que descarta o valor da *expressão* e retorna um valor indefinido. O operador `void` geralmente é usado em comparações que usem o operador `==` para testar os valores indefinidos.

# while

## Disponibilidade

Flash Player 4.

## Uso

```
while(condição) {  
    comando(s);  
}
```

## Parâmetros

*condição* A expressão que é reavaliada sempre que a ação `while` é executada. Se o comando for avaliado como `true`, *comando(s)* será executado.

*comando(s)* O código que será executado se a condição for avaliada como `true`.

## Retorna

Nada.

## Descrição

Ação; testa uma expressão e executa um comando ou série de comandos repetidamente em um loop, contanto que a expressão seja `true`.

Antes de o bloco de comando ser executado, a *condição* é testada; se o teste retornar `true`, o bloco de comando será executado. Se a condição for `false`, o bloco de comando será ignorado e o primeiro comando após o bloco de comando da ação `while` será executado.

O loop é normalmente usado para executar um ação enquanto uma variável de contador for menor do que um valor especificado. No final de cada loop, o contador é incrementado até que o valor especificado seja obtido. Nesse ponto, a *condição* não é mais `true` e o loop termina.

O comando `while` executa a série de etapas a seguir. Cada repetição das etapas de 1 a 4 é chamada de *iteração* do loop. A *condição* é testada novamente no início de cada iteração, como nas seguintes etapas:

- 1 A expressão *condição* é avaliada.
- 2 Se *condição* for avaliada como `true` ou como um valor conversível ao valor Booleano `true`, como um número diferente de zero, vá para a etapa 3.  
Caso contrário, o comando `while` será concluído e a execução continuará no próximo comando após o loop `while`.
- 3 Execute o bloco de comando *comando(s)*.
- 4 Vá para a etapa 1.



### 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 parâmetro *condição* é definido para que o loop `while` seja executado cinco vezes ou enquanto o valor da variável `foo` for menor 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` para que, quando o valor alcançar 5, o parâmetro *condição* seja avaliado como `false` e o loop não seja executado.

```
on(release) {
    foo = 0;
    while(foo < 5) {
        duplicateMovieClip("_root.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++;
    }
}
```

### Consulte também

do while, continue, for, for..in

## with

### Disponibilidade

Flash Player 5.

### Uso

```
with (objeto) {
    comando(s);
}
```

### Parâmetros

*objeto* Uma instância de um objeto ou clipe de filme do ActionScript.

*comando(s)* Uma ação ou grupo de ações entre chaves.

### Retorna

Nada.

### Descrição

Ação; permite especificar um objeto (como um clipe de filme) com o parâmetro *objeto* e avaliar expressões e ações dentro desse objeto com o parâmetro *comando(s)*. Isso evita que seja necessário escrever repetidamente o nome do objeto ou o caminho do objeto.

O parâmetro *objeto* torna-se o contexto em que as propriedades, variáveis e funções no parâmetro *comando(s)* são lidas. Por exemplo, se *objeto* for `myArray` e duas das propriedades especificadas forem `length` e `concat`, essas propriedades serão lidas automaticamente como `myArray.length` e `myArray.concat`. Em outro exemplo, se *objeto* for `state.california`, qualquer comando ou ação dentro da ação `with` será chamado de dentro do comando `california`.

Para localizar o valor de um identificador no parâmetro *comando(s)*, o ActionScript inicia no começo da cadeia do escopo especificado pelo *objeto* e procura pelo identificador em cada 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 item:

- O objeto especificado no parâmetro *objeto* na ação *with* mais interna.
- O objeto especificado no parâmetro *objeto* na ação *with* mais externa.
- O objeto Activation. (um objeto temporário que é criado automaticamente quando uma função é chamada e mantém as variáveis locais chamadas na função.)
- O clipe de filme que contém o script sendo executado no momento.
- O objeto Global (objetos internos como Math e String).

Para definir uma variável dentro de uma ação *with*, a variável deve ter sido declarada fora da ação *with* ou é necessário inserir o caminho completo para a Linha de tempo na qual a variável deve morar. Se for definida uma variável em uma ação *with* sem ser declarada, a ação *with* procurará pelo 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.

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 principal diferença entre as ações *with* e *tellTarget* é que *with* considera um clipe de filme ou outro objeto como seu parâmetro, enquanto *tellTarget* considera uma sequência de caracteres de caminho de destino que identifica um clipe de filme como seu parâmetro e não pode ser usada para especificar objetos.

### Exemplo

O exemplo a seguir define as propriedades *x* e *y* da instância *someOtherMovieClip* e instrui *someOtherMovieClip* a ir para o quadro 3 e parar:

```
with (someOtherMovieClip) {  
    _x = 50;  
    _y = 100;  
    gotoAndStop(3);  
}
```

O próximo trecho de código mostra como escrever o código anterior sem usar uma ação *with*.

```
someOtherMovieClip._x = 50;  
someOtherMovieClip._y = 100;  
someOtherMovieClip.gotoAndStop(3);
```

Também é possível escrever esse código usando a ação *tellTarget*. Entretanto, se *someOtherMovieClip* não fosse um clipe de filme, mas um objeto, não seria possível usar a ação *with*.

```
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 um objeto padrão resolve os identificadores `cos`, `sin` e `PI` como `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 são resolvidos como 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](#)

## XML (objeto)

Use os métodos e propriedades do objeto XML para carregar, analisar, enviar, montar e manipular árvores de documento XML. No Flash MX, o objeto XML tornou-se um objeto nativo. Assim, você poderá observar uma melhora radical no desempenho.

Você deve usar o construtor `new XML()` para criar uma instância do objeto XML antes de chamar qualquer um de seus métodos.

O Flash Player 5 e o Flash Player 6 oferecem suporte ao XML.

## Resumo dos métodos do objeto XML

Método	Descrição
<code>XML.appendChild</code>	Anexa um nó ao fim da lista filha do objeto especificado.
<code>XML.cloneNode</code>	Clona o nó especificado e, opcionalmente, clona recursivamente todos os filhos.
<code>XML.createElement</code>	Cria um novo elemento XML.
<code>XML.createTextNode</code>	Cria um novo nó de texto XML.
<code>XML.getBytesLoaded</code>	Retorna o número de bytes carregados do documento XML especificado.
<code>XML.getBytesTotal</code>	Retorna o tamanho do documento XML em bytes.

Método	Descrição
<code>XML.hasChildNodes</code>	Retorna <code>true</code> se o nó especificado tiver nós filhos; caso contrário, retorna <code>false</code> .
<code>XML.insertBefore</code>	Insere um nó na frente de um nó existente na lista de filhos do nó especificado.
<code>XML.load</code>	Carrega um documento (especificado pelo objeto XML) a partir de uma URL.
<code>XML.parseXML</code>	Analisa um documento XML na árvore de objeto XML especificada.
<code>XML.removeNode</code>	Remove o nó especificado de seu pai.
<code>XML.send</code>	Envia o objeto XML especificado para uma URL.
<code>XML.sendAndLoad</code>	Envia o objeto XML especificado para uma URL e carrega a resposta do servidor em outro objeto XML.
<code>XML.toString</code>	Converte o nó especificado e todos os seus filhos em texto XML.

## Resumo das propriedades do objeto XML

Propriedade	Descrição
<code>XML.contentType</code>	Indica o tipo de MIME transmitido para o servidor.
<code>XML.docTypeDecl</code>	Define e retorna informações sobre a declaração <code>DOCTYPE</code> de um documento XML.
<code>XML.firstChild</code>	Faz referência ao primeiro filho na lista do nó especificado.
<code>XML.ignoreWhite</code>	Quando definida como <code>true</code> , os nós de texto que só contêm espaço em branco são descartados durante o processo de análise.
<code>XML.lastChild</code>	Faz referência ao último filho na lista do nó especificado.
<code>XML.load</code>	Verifica se o objeto XML especificado foi carregado.
<code>XML.nextSibling</code>	Faz referência ao próximo irmão na lista de filhos do nó pai.
<code>XML.nodeName</code>	Retorna o nome da marca de um elemento XML.
<code>XML.nodeType</code>	Retorna o tipo do nó especificado (elemento XML ou nó de texto).
<code>XML.nodeValue</code>	Retorna o texto do nó especificado se o nó for um nó de texto.
<code>XML.parentNode</code>	Faz referência ao nó pai do nó especificado.
<code>XML.previousSibling</code>	Faz referência ao irmão anterior na lista de filhos do nó pai.
<code>XML.status</code>	Retorna um código de status numérico que indica o êxito ou a falha de uma operação de análise de um documento XML.
<code>XML.xmlDecl</code>	Define e retorna informações sobre uma declaração de um documento XML.

## Resumo de coleções do objeto XML

Método	Descrição
<code>XML.attributes</code>	Retorna um vetor associativo que contém todos os atributos do nó especificado.
<code>XML.childNodes</code>	Retorna um vetor que contém referências aos nós filhos do nó especificado.

## Resumo de identificadores de eventos do objeto XML

Método	Descrição
<code>XML.onData</code>	Uma função de retorno de chamada que é chamada quando o download de um texto XML foi totalmente feito do servidor, ou quando ocorre um erro ao fazer o download de um texto XML de um servidor.
<code>XML.onLoad</code>	Uma função de retorno de chamada para <code>load</code> e <code>sendAndLoad</code> .

## Construtor do objeto XML

### Disponibilidade

Flash Player 5.

### Uso

```
new XML([origem])
```

### Parâmetros

*origem* O texto XML analisado para criar o novo objeto XML.

### Retorna

Nada.

### 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.

**Observação:** Os métodos `createElement` e `createTextNode` são os métodos do 'construtor' para criar os elementos e nós de texto em uma árvore de documentos XML.

### Exemplo

Uso 1: o exemplo a seguir cria um novo objeto XML vazio.

```
myXML = new XML();
```

Uso 2: o exemplo a seguir cria um novo objeto XML analisando o texto XML especificado no parâmetro *origem* e preenche o objeto XML recém-criado com a árvore de documentos XML resultante.

```
anyOtherXML = new XML("<state>California<city>san francisco</city></state>");
```

### Consulte também

`XML.createElement`, `XML.createTextNode`

## XML.appendChild

### Disponibilidade

Flash Player 5.

### Uso

```
myXML.appendChild(nó_Filho)
```

### Parâmetros

*nó\_Filho* O nó filho a ser adicionado à lista de filhos do objeto XML especificado.

### Retorna

Nada.

### 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.

### 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

### Disponibilidade

Flash Player 5.

### Uso

*myXML.attributes*

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Coleção (leitura-gravação); retorna um vetor associativo que contém todos os atributos do objeto XML especificado.

### 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
first
```

## XML.childNodes

### Disponibilidade

Flash Player 5.

### Uso

*myXML.childNodes*

### Parâmetros

Nenhum.

### Retorna

Nada.

**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 manipulá-los.

Essa coleção não é definida para os nós de texto (`nodeType == 3`).

**Consulte também**

`XML.nodeType`

## XML.cloneNode

**Disponibilidade**

Flash Player 5.

**Uso**

```
myXML.cloneNode(profundidade)
```

**Parâmetros**

*profundidade* Valor booleano que especifica se os filhos do objeto XML especificado são clonados recursivamente.

**Retorna**

Nada.

**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.

O clone do nó que é retornado não é mais associado à árvore do item clonado.

Conseqüentemente, `nextSibling`, `parentNode` e `previousSibling` têm um valor `null`. Se uma cópia de clipe não for realizada, `firstChild` e `lastChild` também serão `null`.

## XML.contentType

**Disponibilidade**

Flash Player 6.

**Uso**

```
myXML.contentType
```

**Descrição**

Propriedade; o tipo de MIME que é enviado para o servidor quando o método `XML.send` ou `XML.sendAndLoad` é chamado. O padrão é *application/x-www-form-urlencoded*.

**Consulte também**

`XML.send`, `XML.sendAndLoad`

## XML.createElement

### Disponibilidade

Flash Player 5.

### Uso

```
myXML.createElement(nome)
```

### Parâmetros

*nome* O nome da marca do elemento XML que está sendo criado.

### Retorna

Nada.

### Descrição

Método; cria um novo elemento XML com o nome especificado no parâmetro. O novo elemento inicialmente não tem pai, filhos nem irmãos. 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.

## XML.createTextNode

### Disponibilidade

Flash Player 5.

### Uso

```
myXML.createTextNode(texto)
```

### Parâmetros

*texto* O texto usado para criar o novo nó de texto.

### Retorna

Nada.

### Descrição

Método; cria um novo nó de texto XML com o texto especificado. Inicialmente, o novo nó não tem pai e os nós de texto não podem ter filhos nem irmãos. 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.



## XML.docTypeDecl

### Disponibilidade

Flash Player 5.

### Uso

*myXML.XMLdocTypeDecl*

### Descrição

Propriedade; define e retorna informações sobre a declaração DOCTYPE do documento XML. Após o texto XML ter 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">`. Esta propriedade é definida usando uma representação de sequência de caracteres da declaração DOCTYPE, e 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 foi encontrada durante uma 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 for indefinido, nenhuma declaração DOCTYPE será mostrada.

### 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, XML.xmlDecl

## XML.firstChild

### Disponibilidade

Flash Player 5.

### Uso

*myXML.firstChild*

### 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 manipular nós filhos.

### Consulte também

XML.appendChild, XML.insertBefore, XML.removeNode

## XML.getBytesLoaded

### Disponibilidade

Flash Player 6.

### Uso

```
XML.getBytesLoaded()
```

### Parâmetros

Nenhum.

### Retorna

Um inteiro que indica o número de bytes carregados.

### Descrição

Método; retorna o número de bytes carregados (transmitidos) do documento XML. É possível comparar o valor de `getBytesLoaded` com o valor de `getBytesTotal` para determinar que porcentagem de um documento XML foi carregada.

### Consulte também

`XML.getBytesTotal`

## XML.getBytesTotal

### Disponibilidade

Flash Player 6.

### Uso

```
XML.getBytesTotal()
```

### Parâmetros

Nenhum.

### Retorna

Um inteiro.

### Descrição

Método; retorna o tamanho, em bytes, do documento XML.

### Consulte também

`XML.getBytesLoaded`

## XML.hasChildNodes

### Disponibilidade

Flash Player 5.

### Uso

```
myXML.hasChildNodes()
```

### Parâmetros

Nenhum.

**Retorna**

Nada.

**Descrição**

Método; retorna `true` se o objeto XML especificado tem nós filhos; caso contrário, retorna `false`.

**Exemplo**

O exemplo a seguir usa as informações do objeto XML em uma função definida pelo usuário.

```
if (rootNode.hasChildNodes()) {  
    myfunc (rootNode.firstChild);  
}
```

## XML.ignoreWhite

**Disponibilidade**

Flash Player 5.

**Uso**

```
myXML.ignoreWhite = boolean  
XML.prototype.ignoreWhite = boolean
```

**Parâmetros**

*boolean* Um valor Booleano (`true` ou `false`).

**Descrição**

Propriedade; a configuração padrão é `false`. Quando definida como `true`, os nós de texto que só contêm espaço em branco são descartados durante o processo de análise. Os nós de texto com espaço em branco inicial ou de rastro não são afetados.

Uso 1: é possível definir a propriedade `ignoreWhite` para objetos XML individuais, como no código a seguir:

```
myXML.ignoreWhite = true
```

## XML.insertBefore

**Disponibilidade**

Flash Player 5.

**Uso**

```
myXML.insertBefore(nó_Filho, nó_anterior)
```

**Parâmetros**

*nó\_Filho* O nó a ser inserido.

*nó\_anterior* O nó antes do ponto de inserção de *nó\_Filho*.

**Retorna**

Nada.

**Descrição**

Método; insere um novo nó filho na lista de filhos do objeto XML, antes do nó *nó\_anterior*. Se o parâmetro *nó\_anterior* for indefinido ou `null`, o nó será adicionado usando `appendChild`. Se *nó\_anterior* não for um filho de *myXML*, ocorrerá erro na inserção.

## XML.lastChild

### Disponibilidade

Flash Player 5.

### Uso

```
myXML.lastChild
```

### 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 manipular nós filhos.

### Consulte também

`XML.appendChild`, `XML.insertBefore`, `XML.removeNode`

## XML.load

### Disponibilidade

Flash Player 5.

### Uso

```
myXML.load(url)
```

### Parâmetros

*url* A URL em que o documento XML a ser carregado está localizado. O URL deve estar no mesmo subdomínio que o URL onde o filme reside no momento.

### Retorna

Nada.

### 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`.

### Exemplo

A seguir, um exemplo simples usando `XML.load`:

```
doc = new XML();  
doc.load ("theFile.xml");
```

### Consulte também

`XML.loaded`, `XML.onLoad`

## XML.loaded

### Disponibilidade

Flash Player 5.

### Uso

*myXML.loaded*

### 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`.

### Exemplo

O exemplo a seguir usa o `XML.loaded` em um script simples.

```
if (doc.loaded) {  
    gotoAndPlay(4);  
}
```

## XML.nextSibling

### Disponibilidade

Flash Player 5.

### Uso

*myXML.nextSibling*

### 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.

### Consulte também

`XML.appendChild`, `XML.insertBefore`, `XML.removeNode`

## XML.nodeName

### Disponibilidade

Flash Player 5.

### Uso

*myXML.nodeName*

### 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 de texto (`nodeType == 3`), o `nodeName` será `null`.

### Consulte também

`XML.nodeType`

## XML.nodeType

### Disponibilidade

Flash Player 5.

### Uso

*myXML*.nodeType

### Descrição

Propriedade (somente leitura); aceita ou apresenta um valor `nodeType` , onde 1 é um elemento XML e 3 é um nó de texto.

### Consulte também

`XML.nodeValue`

## XML.nodeValue

### Disponibilidade

Flash Player 5.

### Uso

*myXML*.nodeValue

### Descrição

Propriedade; retorna o valor do nó do objeto XML. Se o objeto XML for um nó de texto, `nodeType` será 3 e `nodeValue` será o texto do nó. Se o objeto XML for um elemento XML (tipo de nó é 1), será somente leitura e terá `null` como `nodeValue`.

### Consulte também

`XML.nodeType`

## XML.onData

### Disponibilidade

Flash Player 5

### Uso

*myXML*.onData()

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Identificador de eventos; chamado quando o download de um texto XML foi totalmente feito do servidor, ou quando ocorre um erro ao fazer o download do texto XML de um servidor. Esse identificador é chamado antes de o XML ser analisado e, portanto, pode ser usado para chamar uma rotina de análise personalizada em vez de usar o analisador XML do Flash. O método `XML.onData` retorna o valor `undefined` ou uma sequência de caracteres que contenha texto XML cujo download tenha sido feito do servidor. Se o valor retornado for `undefined`, ocorreu um erro durante o download do XML do servidor.

Por padrão, o método `XML.onData` chama o método `XML.onLoad`. É possível substituir o método `XML.onData` por seu próprio comportamento, mas o `XML.onLoad` não será mais chamado, a menos que seja na sua implementação do `XML.onData`.

### Exemplo

O exemplo a seguir mostra como é o método `onData` por padrão:

```
XML.prototype.onData = function (src) {  
    if (src == undefined) {  
        this.onLoad(false);  
    }  
    else {  
        this.parseXML(src);  
        this.loaded = true;  
        this.onLoad(true);  
    }  
}
```

É possível substituir o método `XML.onData` para interceptar o texto XML sem analisá-lo.

## XML.onLoad

### Disponibilidade

Flash Player 5.

### Uso

`myXML.onLoad(êxito)`

### Parâmetros

*êxito* Um valor booleano que indica se o objeto XML foi carregado com êxito por meio de uma operação `XML.load` ou `XML.sendAndLoad`.

### Retorna

Nada.

### Descrição

Método; chamado pelo Flash Player quando um documento XML é recebido do servidor. Se o documento XML for recebido com êxito, o parâmetro *êxito* será `true`. Se o documento não tiver sido recebido ou se tiver ocorrido algum erro ao receber a resposta do servidor, o parâmetro *êxito* será `false`. A implementação padrão deste método não está ativa. Para substituir a implementação padrão, atribua uma função que contém suas próprias ações.

### Exemplo

O exemplo a seguir cria um filme do Flash simples para um aplicativo de comércio eletrônico. O método `sendAndLoad` transmite um elemento XML que contém o nome e a senha do usuário, e instala um identificador `onLoad` para lidar com a resposta do servidor.

```
function myOnLoad(success) {
    if (success) {
        if (e.firstChild.nodeName == "LOGINREPLY" &&
            e.firstChild.attributes.status == "OK") {
            gotoAndPlay("loggedIn")
        } else {
            gotoAndStop("loginFailed")
        }
    } else {
        gotoAndStop("connectionFailed")
    }
}

var myLoginReply = new XML();
myLoginReply.onLoad = myOnLoad;
myXML.sendAndLoad("http://www.samplestore.com/login.cgi",
    myLoginReply);
```

### Consulte também

`function`, `XML.load`, `XML.sendAndLoad`

## XML.parentNode

### Disponibilidade

Flash Player 5.

### Uso

*myXML.parentNode*

### 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.

## XML.parseXML

### Disponibilidade

Flash Player 5.

### Uso

*myXML.parseXML(origem)*

### Parâmetros

*origem* O texto XML a ser analisado e passado para o objeto XML especificado.

### Retorna

Nada.

### Descrição

Método; analisa o texto XML especificado no parâmetro *origem* e preenche o objeto XML especificado com a árvore XML resultante. Quaisquer árvores existentes no objeto XML são descartadas.



## XML.previousSibling

### Disponibilidade

Flash Player 5.

### Uso

```
myXML.previousSibling
```

### Descrição

Propriedade (somente leitura); retorna uma referência do 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.

## XML.removeNode

### Disponibilidade

Flash Player 5.

### Uso

```
myXML.childNodes[1].removeNode()
```

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Método; remove o objeto XML especificado de seu pai. Todos os descendentes do nó também são excluídos.

## XML.send

### Disponibilidade

Flash Player 5.

### Uso

```
myXML.send(url, [janela])
```

### Parâmetros

*url* A URL de destino do objeto XML especificado.

*janela* A janela do navegador que exibe 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. Este parâmetro é opcional; se nenhum parâmetro *window* for especificado, será o mesmo que especificar `_self`.

### Retorna

Nada.

### Descrição

Método; codifica o objeto XML especificado em um documento XML e o envia para o URL especificado usando o método POST.

## XML.sendAndLoad

### Disponibilidade

Flash Player 5.

### Uso

```
myXML.sendAndLoad(url,objeto_XML_de_destino)
```

### Parâmetros

*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.

*objeto\_XML\_de\_destino* Um objeto XML criado com o método construtor XML que receberá as informações de retorno do servidor.

### Retorna

Nada.

### Descrição

Método; codifica o objeto XML especificado em um documento XML, envia-o para a URL especificada usando o método POST, faz o download da resposta do servidor e a carrega no objeto *objeto\_XML\_de\_destino* especificado nos parâmetros. A resposta do servidor é carregada da mesma maneira usada pelo método `load`.

### Consulte também

XML.load

## XML.status

### Disponibilidade

Flash Player 5.

### Uso

```
myXML.status
```

### Descrição

Propriedade; define e retorna automaticamente um valor numérico que indica se um documento XML foi analisado com êxito em um objeto XML. Os códigos de status numérico e uma descrição de cada um deles são listados da seguinte maneira:

- 0 Sem erro; a análise foi 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.

## XML.toString

### Disponibilidade

Flash Player 5.

### Uso

```
myXML.toString()
```

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Método; avalia o objeto XML especificado, constrói uma representação textual da estrutura XML incluindo nó, filhos e atributos, e retorna o resultado como uma seqüência de caracteres.

No caso de objetos XML de alto nível (os criados com o construtor), `XML.toString` gera a declaração XML do documento (armazenada em `XML.xmlDecl`), seguida da declaração DOCTYPE do documento (armazenada em `XML.docTypeDecl`), seguida da representação textual 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.

### Exemplo

O código a seguir é um exemplo do método `XML.toString` que envia `<h1>test</h1>` para a janela de saída.

```
node = new XML("<h1>test</h1>");  
trace(node.toString());
```

### Consulte também

`XML.docTypeDecl`, `XML.xmlDecl`

## XML.xmlDecl

### Disponibilidade

Flash Player 5.

### Uso

```
myXML.xmlDecl
```

### Descrição

Propriedade; define e retorna informações sobre uma declaração XML do documento. Depois de o documento XML ser analisado em um objeto XML, essa propriedade é definida como o texto da declaração XML do documento. Essa propriedade é definida usando uma representação de seqüência 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 `undefined.XML`. O método `toString` apresenta 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.

### 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\" ?>";
```

A seguir, um exemplo de declaração XML:

```
<?xml version="1.0" ?>
```

### Consulte também

`XML.docTypeDecl`, `XML.toString`

## 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 exigir 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` só pode conectar a números de porta TCP maiores que 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 que ou iguais a 1024. Os números de porta abaixo de 1024 geralmente são usados pelos serviços de sistema como FTP, Telnet e HTTP, barrando, dessa forma, o objeto XMLSocket dessas portas por motivos de segurança. 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>XMLSocket.close</code>	Fecha uma conexão de soquete aberta.
<code>XMLSocket.connect</code>	Estabelece uma conexão com o servidor especificado.
<code>XMLSocket.send</code>	Envia um objeto XML para o servidor.

## Resumo de identificadores de eventos do objeto XMLSocket

Método	Descrição
<code>XMLSocket.onClose</code>	Uma função de chamada que é chamada quando uma conexão XMLSocket é fechada.
<code>XMLSocket.onConnect</code>	Uma função de chamada que é chamada quando uma conexão XMLSocket é estabelecida.
<code>XMLSocket.onData</code>	Uma função de retorno de chamada que é chamada após o download de uma mensagem XML do servidor.
<code>XMLSocket.onXML</code>	Uma função de chamada que é chamada quando um objeto XML chega do servidor.

## Construtor do objeto XMLSocket

### Disponibilidade

Flash Player 5.

### Uso

```
new XMLSocket()
```

### Parâmetros

Nenhum.

### Retorna

Nada.

**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.

**Exemplo**

```
myXMLSocket = new XMLSocket();
```

**Consulte também**

XMLSocket.connect

## XMLSocket.close

**Disponibilidade**

Flash Player 5.

**Uso**

```
myXMLSocket.close()
```

**Parâmetros**

Nenhum.

**Retorna**

Nada.

**Descrição**

Método; fecha a conexão especificada pelo objeto XMLSocket.

**Consulte também**

XMLSocket.connect

## XMLSocket.connect

**Disponibilidade**

Flash Player 5.

**Uso**

```
myXMLSocket.connect(host, porta)
```

**Parâmetros**

*host* Um nome de domínio DNS totalmente qualificado ou um endereço IP no formato *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.

**Retorna**

Nada.

### 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 o controle ActiveX estiver sendo usado, o host especificado no parâmetro deverá ter o mesmo subdomínio do host do qual foi feito o download do filme.

Se for especificado `null` para o parâmetro *host*, o host contatado será aquele em que reside o filme que chama `XMLSocket.connect`. Por exemplo, se o download do filme foi feito de `http://www.seu_site.com`, especificar `null` para o parâmetro do host é o mesmo que digitar o endereço IP de `www.seu_site.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.

### 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, XMLSocket.onConnect`

## XMLSocket.onClose

### Disponibilidade

Flash Player 5.

### Uso

`myXMLSocket.onClose()`

### Parâmetros

Nenhum.

### Retorna

Nada.

### 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 ações. Para substituir a implementação padrão, atribua uma função que contém suas próprias ações.

### Consulte também

`function, XMLSocket.onConnect`

# XMLSocket.onConnect

## Disponibilidade

Flash Player 5.

## Uso

```
myXMLSocket.onConnect(êxito)
```

## Parâmetros

*êxito* Um valor booleano que indica se uma conexão de soquete foi estabelecida com êxito (true ou false).

## Retorna

Nada.

## 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 parâmetro *êxito* é true; caso contrário, o parâmetro *êxito* é false.

A implementação padrão desse método não executa ações. Para substituir a implementação padrão, atribua uma função que contém suas próprias ações.

## 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 conduzidos, dependendo do êxito da conexão estabelecida. Se a conexão for estabelecida com êxito, os usuários serão conduzidos 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 `connect` retornar true, o filme saltará para um quadro chamado `waitForConnection`, que é a tela “Aguarde”. 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

`function`, `XMLSocket.connect`



## XMLSocket.onData

### Disponibilidade

Flash Player 5.

### Uso

`XMLSocket.onData()`

### Parâmetros

Nenhum.

### Retorna

Nada.

### Descrição

Identificador de eventos; chamado após o download de uma mensagem XML do servidor, terminada por um byte zero.

Por padrão, o método `XMLSocket.onData` chama o método `XMLSocket.onXML`. Se você substituir `XMLSocket.onData` por seu próprio comportamento, `XMLSocket.onXML` não será mais chamado, a menos que seja na sua implementação de `XMLSocket.onData`.

```
XMLSocket.prototype.onData = function (src) {  
    this.onXML(new XML(src));  
}
```

No exemplo acima, o parâmetro *src* é uma sequência de caracteres que contém texto XML cujo download foi feito do servidor. O byte zero de término não é incluído na sequência de caracteres.

`XMLSocket.onData` pode ser substituído para interceptar o texto XML sem analisá-lo.

## XMLSocket.onXML

### Disponibilidade

Flash Player 5.

### Uso

*myXMLSocket.onXML(objeto)*

### Parâmetro

*objeto* Uma instância do objeto XML que contém um documento XML analisado recebido de um servidor.

### Retorna

Nada.

### 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 0 (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.

### 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 a reconhecer um único elemento XML, `MESSAGE`, no seguinte formato:

```
<MESSAGE USER="John" TEXT="Olá, meu nome é John!" />.
```

O manipulador `onXML` deve primeiro ser instalado no objeto `XMLSocket` da seguinte forma:

```
socket.onXML = myOnXML;
```

A função `displayMessage` é subentendida como uma função definida pelo usuário que exibe a mensagem recebida pelo usuário.

```
function myOnXML(doc) {  
    var e = doc.firstChild;  
    if (e != null && e.nodeName == "MESSAGE") {  
        displayMessage(e.attributes.user, e.attributes.text);  
    }  
}
```

### Consulte também

`function`

## XMLSocket.send

### Disponibilidade

Flash Player 5.

### Uso

```
myXMLSocket.send(objeto)
```

### Parâmetros

*objeto* Um objeto XML ou outros dados a serem transmitidos para o servidor.

### Retorna

Nada.

### Descrição

Método; converte o objeto XML ou os dados especificados no parâmetro *objeto* em uma sequência de caracteres e a transmite para o servidor, seguida de um byte zero. Se *objeto* for um objeto XML, a sequência de caracteres será a representação textual XML do objeto XML. A operação de envio é assíncrona; ela retorna imediatamente, mas os dados podem ser transmitidos posteriormente. O método `XMLSocket.send` não retorna nenhum valor que indique 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.

**Exemplo**

O exemplo a seguir ilustra como é possível especificar um nome e uma 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`

