

扩展

# ADOBE® FLASH® PROFESSIONAL CS5 和 CS5.5

## 法律声明

有关法律声明，请参阅 [http://help.adobe.com/zh\\_CN/legalnotices/index.html](http://help.adobe.com/zh_CN/legalnotices/index.html)。

# 目录

## 第 1 章：简介

使用 JavaScript API .....	1
JavaScript API 中的新增内容 .....	4
JavaScript API 对象 .....	6
实现范例 .....	11

## 第 2 章：顶级函数和方法

activate() .....	13
alert() .....	14
configureTool() .....	14
confirm() .....	15
deactivate() .....	16
keyDown() .....	16
keyUp() .....	17
mouseDoubleClick() .....	17
mouseDown() .....	18
mouseMove() .....	19
mouseUp() .....	19
notifySettingsChanged() .....	20
prompt() .....	20
setCursor() .....	21

## 第 3 章：actionsPanel 对象

actionsPanel.getClassForObject() .....	22
actionsPanel.getScriptAssistMode() .....	23
actionsPanel.getSelectedText() .....	23
actionsPanel.getText() .....	24
actionsPanel.hasSelection() .....	24
actionsPanel.replaceSelectedText() .....	25
actionsPanel.setScriptAssistMode() .....	26
actionsPanel.setSelection() .....	26
actionsPanel.setText() .....	27

## 第 4 章：BitmapInstance 对象

bitmapInstance.getBits() .....	28
bitmapInstance.hPixels .....	29
bitmapInstance.setBits() .....	29
bitmapInstance.vPixels .....	30

**第 5 章 : BitmapItem 对象**

bitmapItem.allowSmoothing	31
bitmapItem.compressionType	32
bitmapItem.exportToFile()	32
bitmapItem.fileLastModifiedDate	33
bitmapItem.originalCompressionType	33
bitmapItem.quality	34
bitmapItem.sourceFileExists	34
bitmapItem.sourceFileIsCurrent	34
bitmapItem.sourceFilePath	35
bitmapItem.useDeblocking	35
bitmapItem.useImportedJPEGQuality	36

**第 6 章 : CompiledClipInstance 对象**

compiledClipInstance.accName	37
compiledClipInstance.actionScript	38
compiledClipInstance.description	38
compiledClipInstance.forceSimple	38
compiledClipInstance.shortcut	39
compiledClipInstance.silent	39
compiledClipInstance.tabIndex	40

**第 7 章 : compilerErrors 对象**

compilerErrors.clear()	41
compilerErrors.save()	42

**第 8 章 : ComponentInstance 对象**

componentInstance.parameters	43
------------------------------	----

**第 9 章 : componentsPanel 对象**

componentsPanel.addItemToDocument()	44
componentsPanel.reload()	45

**第 10 章 : Contour 对象**

contour.fill	46
contour.getHalfEdge()	47
contour.interior	48
contour.orientation	48

**第 11 章 : Document 对象**

document.accName	56
document.addDataToDocument()	56
document.addDataToSelection()	57
document.addFilter()	57

document.addItem()	58
document.addNewLine()	59
document.addNewOval()	59
document.addNewPrimitiveOval()	60
document.addNewPrimitiveRectangle()	61
document.addNewPublishProfile()	61
document.addNewRectangle()	62
document.addNewScene()	63
document.addNewText()	64
document.align()	64
document.allowScreens()	65
document.arrange()	65
document.as3AutoDeclare	66
document.as3Dialect	66
document.as3ExportFrame	67
document.as3StrictMode	67
document.as3WarningsMode	68
document.asVersion	68
document.autoLabel	69
document.backgroundColor	69
document.breakApart()	70
document.canEditSymbol()	70
document.canRevert()	71
document.canTestMovie()	71
document.canTestScene()	72
document.changeFilterOrder()	72
document.clipCopy()	73
document.clipCut()	74
document.clipPaste()	74
document.close()	75
document.convertLinesToFills()	75
document.convertToSymbol()	76
document.crop()	76
document.currentPublishProfile	77
document.currentTimeline	77
document.debugMovie()	78
document.deleteEnvelope()	78
document.deletePublishProfile()	79
document.deleteScene()	79
document.deleteSelection()	80
document.description	80
document.disableAllFilters()	81

document.disableFilter()	81
document.disableOtherFilters()	82
document.distribute()	82
document.distributeToLayers()	83
document.docClass	84
document.documentHasData()	84
document.duplicatePublishProfile()	85
document.duplicateScene()	85
document.duplicateSelection()	86
document.editScene()	86
document.enableAllFilters()	87
document.enableFilter()	87
document.enterEditMode()	88
document.exitEditMode()	88
document.exportPNG()	89
document.exportPublishProfile()	90
document.exportPublishProfileString()	90
document.exportSWF()	91
document.externalLibraryPath	91
document.forceSimple	92
document.frameRate	92
document.getAlignToDocument()	93
document.getBlendMode()	93
document.getCustomFill()	94
document.getCustomStroke()	95
document.getDataFromDocument()	95
document.getElementProperty()	96
document.getElementTextAttr()	96
document.getFilters()	97
document.getMetadata()	98
document.getMobileSettings()	98
document.getPlayerVersion()	99
document.getSelectionRect()	100
document.getTextString()	100
document.getTimeline()	101
document.getTransformationPoint()	102
document.group()	102
document.height	103
document.id	103
document.importFile()	104
document.importPublishProfile()	104
document.importPublishProfileString()	105

document.importSWF()	105
document.intersect()	106
document.library	106
document.libraryPath	107
document.livePreview	107
document.loadCuepointXML()	108
document.match()	108
document.mouseClick()	109
document.mouseDbClk()	109
document.moveSelectedBezierPointsBy()	110
document.moveSelectionBy()	111
document.name	111
document.optimizeCurves()	112
document.path	112
document.pathURI	113
document.publish()	113
document.publishProfiles	114
document.punch()	114
document.removeAllFilters()	115
document.removeDataFromDocument()	115
document.removeDataFromSelection()	116
document.removeFilter()	116
document.renamePublishProfile()	117
document.renameScene()	117
document.reorderScene()	118
document.resetOvalObject()	118
document.resetRectangleObject()	119
document.resetTransformation()	119
document.revert()	120
document.rotate3DSelection()	120
document.rotateSelection()	121
document.save()	122
document.saveAndCompact()	122
document.scaleSelection()	123
document.screenOutline	124
document.selectAll()	124
document.selection	125
document.selectNone()	126
document.setAlignToDocument()	127
document.setBlendMode()	127
document.setCustomFill()	128
document.setCustomStroke()	128

document.setElementProperty()	129
document.setElementTextAttr()	129
document.setFillColor()	130
document.setFilterProperty()	131
document.setFilters()	131
document.setInstanceAlpha()	132
document.setInstanceBrightness()	133
document.setInstanceTint()	133
document.setMetadata()	134
document.setMobileSettings()	135
document.setOvalObjectProperty()	136
document.setPlayerVersion()	136
document.setRectangleObjectProperty()	137
document.setSelectionBounds()	137
document.setSelectionRect()	138
document.setStageVanishingPoint()	139
document.setStageViewAngle()	139
document.setStroke()	140
document.setStrokeColor()	140
document.setStrokeSize()	141
document.setStrokeStyle()	142
document.setTextRectangle()	142
document.setTextSelection()	143
document.setTextString()	143
document.setTransformationPoint()	144
document.silent	145
document.skewSelection()	145
document.smoothSelection()	146
document.sourcePath	146
document.space()	147
document.straightenSelection()	147
document.swapElement()	148
document.swapStrokeAndFill()	148
document.testMovie()	149
document.testScene()	149
document.timelines	150
document.traceBitmap()	151
document.translate3DCenter()	151
document.translate3DSelection()	152
document.transformSelection()	152
document.unGroup()	153
document.union()	153



document.unlockAllElements()	154
document.viewMatrix	154
document.width	155
document.xmlPanel()	155
document.zoomFactor	156
<b>第 12 章 : drawingLayer 对象</b>	
drawingLayer.beginDraw()	157
drawingLayer.beginFrame()	158
drawingLayer.cubicCurveTo()	158
drawingLayer.curveTo()	159
drawingLayer.drawPath()	160
drawingLayer.endDraw()	160
drawingLayer.endFrame()	161
drawingLayer.lineTo()	161
drawingLayer.moveTo()	162
drawingLayer.newPath()	162
drawingLayer.setColor()	163
drawingLayer.setFill()	163
drawingLayer.setStroke()	163
<b>第 13 章 : Edge 对象</b>	
edge.cubicSegmentIndex	164
edge.getControl()	165
edge.getHalfEdge()	165
edge.id	166
edge.isLine	166
edge.setControl()	166
edge.splitEdge()	167
edge.stroke	168
<b>第 14 章 : Element 对象</b>	
element.depth	170
element.elementType	170
element.getPersistentData()	171
element.getTransformationPoint()	171
element.hasPersistentData()	172
element.height	173
element.layer	173
element.left	173
element.locked	174
element.matrix	174
element.name	175

element.removePersistentData()	175
element.rotation	175
element.scaleX	176
element.scaleY	176
element.selected	177
element.setPersistentData()	177
element.setTransformationPoint()	178
element.skewX	178
element.skewY	179
element.top	179
element.transformX	180
element.transformY	180
element.width	180
element.x	181
element.y	181
<b>第 15 章 : Fill 对象</b>	
fill.bitmapIsClipped	182
fill.bitmapPath	183
fill.color	183
fill.colorArray	184
fill.focalPoint	184
fill.linearRGB	185
fill.matrix	185
fill.overflow	186
fill.posArray	186
fill.style	187
<b>第 16 章 : Filter 对象</b>	
filter.angle	189
filter.blurX	189
filter.blurY	190
filter.brightness	190
filter.color	191
filter.contrast	191
filter.distance	192
filter.enabled	192
filter.hideObject	193
filter.highlightColor	193
filter.hue	194
filter.inner	194
filter.knockout	195
filter.name	195

filter.quality	196
filter.saturation	196
filter.shadowColor	197
filter.strength	198
filter.type	198
<b>第 17 章 : flash 对象 (fl)</b>	
fl.actionsPanel	202
fl.addEventListener()	203
fl.as3PackagePaths	203
fl.browseForFileURL()	204
fl.browseForFolderURL()	204
fl.clearPublishCache()	205
fl.clipCopyString()	206
fl.closeAll()	206
fl.closeAllPlayerDocuments()	207
fl.closeDocument()	207
fl.compilerErrors	208
fl.componentsPanel	208
fl.configDirectory	208
fl.configURI	209
fl.contactSensitiveSelection	209
fl.createDocument()	210
fl.createNewDocList	210
fl.createNewDocListType	211
fl.createNewTemplateList	211
fl.documents	211
fl.drawingLayer	212
fl.exportPublishProfileString()	212
fl.externalLibraryPath	213
fl.fileExists()	213
fl.findDocumentDOM()	214
fl.findDocumentIndex()	214
fl.findObjectInDocByName()	215
fl.findObjectInDocByType()	216
fl.flexSDKPath	217
fl.getAppMemoryInfo()	218
fl.getDocumentDOM()	218
fl.getSwfPanel()	219
fl.installedPlayers	220
fl.isFontInstalled()	220
fl.languageCode	221

fl.libraryPath	221
fl.mapPlayerURL()	222
fl.Math	222
fl.mruRecentFileList	223
fl.mruRecentFileListType	223
fl.objectDrawingMode	224
fl.openDocument()	224
fl.openScript()	225
fl.outputPanel	225
fl.packagePaths	226
fl.presetPanel	226
fl.publishCacheDiskSizeMax	226
fl.publishCacheEnabled	227
fl.publishCacheMemoryEntrySizeLimit	227
fl.publishCacheMemorySizeMax	228
fl.publishDocument()	228
fl.quit()	229
fl.reloadEffects()	229
fl.reloadTools()	230
fl.removeEventListener()	230
fl.resetAS3PackagePaths()	231
fl.resetPackagePaths()	231
fl.revertDocument()	232
fl.runScript()	232
fl.saveAll()	233
fl.saveDocument()	234
fl.saveDocumentAs()	235
fl.scriptURI	235
fl.selectElement()	236
fl.selectTool()	236
fl.setActiveWindow()	237
fl.showIdleMessage()	238
fl.sourcePath	238
fl.swfPanels	239
fl.toggleBreakpoint()	239
fl.tools	240
fl.trace()	240
fl.version	241
fl.xmlui	241

**第 18 章 : FLfile 对象**

FLfile.copy()	243
FLfile.createFolder()	244
FLfile.exists()	244
FLfile.getAttributes()	245
FLfile.getCreationDate()	246
FLfile.getCreationDateObj()	247
FLfile.getModificationDate()	247
FLfile.getModificationDateObj()	248
FLfile.getSize()	249
FLfile.listFolder()	249
FLfile.platformPathToURI()	250
FLfile.read()	251
FLfile.remove()	251
FLfile.setAttributes()	252
FLfile.uriToPlatformPath()	253
FLfile.write()	254

**第 19 章 : folderItem 对象**

**第 20 章 : fontItem 对象**

fontItem.bitmap	256
fontItem.bold	257
fontItem.embeddedCharacters	257
fontItem.embedRanges	257
fontItem.embedVariantGlyphs	258
fontItem.font	259
fontItem.isDefineFont4Symbol	260
fontItem.italic	260
fontItem.size	261

**第 21 章 : Frame 对象**

frame.convertMotionObjectTo2D()	263
frame.convertMotionObjectTo3D()	264
frame.actionScript	264
frame.duration	265
frame.elements	265
frame.getCustomEase()	265
frame.getMotionObjectXML()	266
frame.hasCustomEase	267
frame.hasMotionPath()	267
frame.is3DMotionObject()	268
frame.isMotionObject()	268

frame.labelType	269
frame.motionTweenOrientToPath	269
frame.motionTweenRotate	270
frame.motionTweenRotateTimes	270
frame.motionTweenScale	270
frame.motionTweenSnap	271
frame.motionTweenSync	271
frame.name	271
frame.selectMotionPath()	272
frame.setCustomEase()	272
frame.setMotionObjectDuration()	273
frame.setMotionObjectXML()	274
frame.shapeTweenBlend	274
frame.soundEffect	274
frame.soundLibraryItem	275
frame.soundLoop	275
frame.soundLoopMode	276
frame.soundName	276
frame.soundSync	276
frame.startFrame	277
frame.tweenEasing	277
frame.tweenInstanceName	278
frame.tweenType	278
frame.useSingleEaseCurve	278
<b>第 22 章 : HalfEdge 对象</b>	
halfEdge.getEdge()	280
halfEdge.getNext()	281
halfEdge.getOppositeHalfEdge()	281
halfEdge.getPrev()	282
halfEdge.getVertex()	282
halfEdge.id	283
halfEdge.index	283
<b>第 23 章 : Instance 对象</b>	
instance.instanceType	285
instance.libraryItem	285
<b>第 24 章 : Item 对象</b>	
item.addData()	287
item.getData()	288
item.hasData()	288
item.itemType	289

item.linkageBaseClass .....	289
item.linkageClassName .....	290
item.linkageExportForAS .....	290
item.linkageExportForRS .....	291
item.linkageExportInFirstFrame .....	291
item.linkageIdentifier .....	292
item.linkageImportForRS .....	292
item.linkageURL .....	292
item.name .....	293
item.removeData() .....	293
<b>第 25 章 : Layer 对象</b>	
layer.color .....	295
layer.frameCount .....	296
layer.frames .....	296
layer.height .....	297
layer.layerType .....	297
layer.locked .....	297
layer.name .....	298
layer.outline .....	298
layer.parentLayer .....	298
layer.visible .....	299
<b>第 26 章 : library 对象</b>	
library.addItemToDocument() .....	301
library.addNewItem() .....	301
library.deleteItem() .....	302
library.duplicateItem() .....	302
library.editItem() .....	303
library.expandFolder() .....	303
library.findItemIndex() .....	304
library.getItemProperty() .....	304
library.getItemType() .....	305
library.getSelectedItems() .....	305
library.importEmbeddedSWF() .....	306
library.itemExists() .....	307
library.items .....	307
library.moveToFolder() .....	307
library.newFolder() .....	308
library.renameItem() .....	309
library.selectAll() .....	309
library.selectItem() .....	310
library.selectNone() .....	310

library.setItemProperty()	311
library.updateItem()	311
<b>第 27 章 : Math 对象</b>	
Math.concatMatrix()	313
Math.invertMatrix()	313
Math.pointDistance()	314
<b>第 28 章 : Matrix 对象</b>	
matrix.a	315
matrix.b	316
matrix.c	316
matrix.d	317
matrix.tx	317
matrix.ty	317
<b>第 29 章 : outputPanel 对象</b>	
outputPanel.clear()	319
outputPanel.save()	320
outputPanel.trace()	320
<b>第 30 章 : Oval 对象</b>	
OvalObject.closePath	322
OvalObject.endAngle	323
OvalObject.innerRadius	323
OvalObject.startAngle	323
<b>第 31 章 : Parameter 对象</b>	
parameter.category	325
parameter.insertItem()	326
parameter.listIndex	326
parameter.name	327
parameter.removeItem()	327
parameter.value	328
parameter.valueType	328
parameter.verbose	329
<b>第 32 章 : Path 对象</b>	
path.addCubicCurve()	330
path.addCurve()	331
path.addPoint()	332
path.clear()	332
path.close()	333
path.makeShape()	333



path.newContour()	334
path.nPts	334
<b>第 33 章 : presetItem 对象</b>	
presetItem.isDefault	336
presetItem.isFolder	337
presetItem.level	337
presetItem.name	337
presetItem.open	338
presetItem.path	338
<b>第 34 章 : presetPanel 对象</b>	
presetPanel.addNewItem()	339
presetPanel.applyPreset()	340
presetPanel.deleteFolder()	341
presetPanel.deleteItem()	341
presetPanel.expandFolder()	342
presetPanel.exportItem()	343
presetPanel.findItemIndex()	343
presetPanel.getSelectedItems()	344
presetPanel.importItem()	345
presetPanel.items	345
presetPanel.moveToFolder()	346
presetPanel.newFolder()	346
presetPanel.renameItem()	347
presetPanel.selectItem()	348
<b>第 35 章 : Rectangle 对象</b>	
RectangleObject.bottomLeftRadius	349
RectangleObject.bottomRightRadius	350
RectangleObject.lockFlag	350
RectangleObject.topLeftRadius	350
RectangleObject.topRightRadius	351
<b>第 36 章 : Shape 对象</b>	
shape.beginEdit()	353
shape.contours	353
shape.deleteEdge()	353
shape.edges	354
shape.endEdit()	354
shape.getCubicSegmentPoints()	355
shape.isDrawingObject	355
shape.isGroup	356
shape.isOvalObject	356

shape.isRectangleObject	357
shape.members	357
shape.numCubicSegments	358
shape.vertices	358
<b>第 37 章 : SoundItem 对象</b>	
soundItem.bitRate	359
soundItem.bits	360
soundItem.compressionType	360
soundItem.convertStereoToMono	361
soundItem.exportToFile()	361
soundItem.fileLastModifiedDate	362
soundItem.originalCompressionType	362
soundItem.quality	363
soundItem.sampleRate	363
soundItem.sourceFileExists	364
soundItem.sourceFileIsCurrent	364
soundItem.sourceFilePath	365
soundItem.useImportedMP3Quality	365
<b>第 38 章 : Stroke 对象</b>	
stroke.breakAtCorners	367
stroke.capType	367
stroke.color	368
stroke.curve	368
stroke.dash1	369
stroke.dash2	369
stroke.density	369
stroke.dotSize	370
stroke.dotSpace	370
stroke.hatchThickness	371
stroke.jiggle	371
stroke.joinType	372
stroke.length	372
stroke.miterLimit	372
stroke.pattern	373
stroke.rotate	373
stroke.scaleType	374
stroke.shapeFill	374
stroke.space	375
stroke.strokeHinting	375
stroke.style	375
stroke.thickness	376

stroke.variation	376
stroke.waveHeight	377
stroke.waveLength	377
<b>第 39 章 : swfPanel 对象</b>	
swfPanel.call()	379
swfPanel.name	381
swfPanel.path	382
swfPanel.setFocus()	382
<b>第 40 章 : SymbolInstance 对象</b>	
symbolInstance.accName	384
symbolInstance.actionScript	384
symbolInstance.backgroundColor	385
symbolInstance.bitmapRenderMode	385
symbolInstance.blendMode	386
symbolInstance.buttonTracking	386
symbolInstance.cacheAsBitmap	387
symbolInstance.colorAlphaAmount	387
symbolInstance.colorAlphaPercent	387
symbolInstance.colorBlueAmount	388
symbolInstance.colorBluePercent	388
symbolInstance.colorGreenAmount	388
symbolInstance.colorGreenPercent	389
symbolInstance.colorMode	389
symbolInstance.colorRedAmount	389
symbolInstance.colorRedPercent	390
symbolInstance.description	390
symbolInstance.filters	391
symbolInstance.firstFrame	391
symbolInstance.forceSimple	391
symbolInstance.loop	392
symbolInstance.shortcut	392
symbolInstance.silent	393
symbolInstance.symbolType	393
symbolInstance.tabIndex	394
symbolInstance.usesBackgroundColor	394
symbolInstance.visible	394
<b>第 41 章 : SymbolItem 对象</b>	
symbolItem.convertToCompiledClip()	396
symbolItem.exportSWC()	397
symbolItem.exportSWF()	397

symbolItem.scalingGrid .....	398
symbolItem.scalingGridRect .....	398
symbolItem.sourceAutoUpdate .....	399
symbolItem.sourceFilePath .....	399
symbolItem.sourceLibraryName .....	399
symbolItem.symbolType .....	400
symbolItem.timeline .....	400
<b>第 42 章 : Text 对象</b>	
text.accName .....	402
text.antiAliasSharpness .....	403
text.antiAliasThickness .....	403
text.autoExpand .....	403
text.border .....	404
text.description .....	404
text.embeddedCharacters .....	405
text.embedRanges .....	405
text.embedVariantGlyphs .....	406
text.fontRenderingMode .....	406
text.getTextAttr() .....	407
text.getTextString() .....	408
text.length .....	408
text.lineType .....	409
text.maxCharacters .....	409
text.orientation .....	409
text.renderAsHTML .....	410
text.scrollable .....	410
text.selectable .....	410
text.selectionEnd .....	411
text.selectionStart .....	411
text.setTextAttr() .....	412
text.setTextString() .....	413
text.shortcut .....	413
text.silent .....	414
text.tabIndex .....	414
text.textRuns .....	415
text.textType .....	415
text.useDeviceFonts .....	415
text.variableName .....	416
<b>第 43 章 : TextAttrs 对象</b>	
textAttrs.aliasText .....	417
textAttrs.alignment .....	418

textAttrs.autoKern	418
textAttrs.bold	419
textAttrs.characterPosition	419
textAttrs.characterSpacing	419
textAttrs.face	420
textAttrs.fillColor	420
textAttrs.indent	420
textAttrs.italic	421
textAttrs.leftMargin	421
textAttrs.letterSpacing	422
textAttrs.lineSpacing	422
textAttrs.rightMargin	422
textAttrs.rotation	423
textAttrs.size	423
textAttrs.target	423
textAttrs.url	424
<b>第 44 章 : TextRun 对象</b>	
textRun.textAttrs	425
textRun.characters	425
<b>第 45 章 : Timeline 对象</b>	
timeline.addMotionGuide()	429
timeline.addNewLayer()	429
timeline.clearFrames()	430
timeline.clearKeyframes()	430
timeline.convertToBlankKeyframes()	431
timeline.convertToKeyframes()	432
timeline.copyFrames()	432
timeline.copyLayers()	433
timeline.copyMotion()	434
timeline.copyMotionAsAS3()	434
timeline.createMotionObject()	435
timeline.createMotionTween()	435
timeline.currentFrame	436
timeline.currentLayer	436
timeline.cutFrames()	437
timeline.cutLayers()	437
timeline.deleteLayer()	438
timeline.duplicateLayers()	439
timeline.expandFolder()	439
timeline.findLayerIndex()	440
timeline.frameCount	441

timeline.getFrameProperty()	441
timeline.getGuidelines()	442
timeline.getLayerProperty()	442
timeline.getSelectedFrames()	443
timeline.getSelectedLayers()	443
timeline.insertBlankKeyframe()	444
timeline.insertFrames()	444
timeline.insertKeyframe()	445
timeline.layerCount	446
timeline.layers	446
timeline.libraryItem	447
timeline.name	447
timeline.pasteFrames()	448
timeline.pasteLayers()	448
timeline.pasteMotion()	449
timeline.removeFrames()	449
timeline.removeMotionObject()	450
timeline.reorderLayer()	451
timeline.reverseFrames()	451
timeline.selectAllFrames()	452
timeline setFrameProperty()	452
timeline.setGuidelines()	453
timeline.setLayerProperty()	454
timeline.setSelectedFrames()	454
timeline.setSelectedLayers()	455
timeline.showLayerMasking()	456
timeline.startPlayback()	456
timeline.stopPlayback()	457
<b>第 46 章 : ToolObj 对象</b>	
toolObj.depth	458
toolObj.enablePIControl()	459
toolObj.iconID	460
toolObj.position	460
toolObj.setIcon()	461
toolObj.setMenuString()	461
toolObj.setOptionsFile()	462
toolObj.setPI()	463
toolObj.setToolName()	463
toolObj.setToolTip()	464
toolObj.showPIControl()	464
toolObj.showTransformHandles()	465

**第 47 章 : Tools 对象**

tools.activeTool	467
tools.altIsDown	468
tools.constrainPoint()	468
tools.ctrlIsDown	469
tools.getKeyDown()	469
tools.mouseIsDown	469
tools.penDownLoc	470
tools.penLoc	470
tools.setCreatingBbox()	471
tools.setCursor()	471
tools.shiftIsDown	472
tools.snapPoint()	472
tools.toolObjs	473

**第 48 章 : Vertex 对象**

vertex.getHalfEdge()	474
vertex.setLocation()	475
vertex.x	475
vertex.y	476

**第 49 章 : VideoItem 对象**

videoItem.exportToFLV()	477
videoItem.fileLastModifiedDate	478
videoItem.sourceFileExists	478
videoItem.sourceFileIsCurrent	479
videoItem.sourceFilePath	479
videoItem.videoType	480

**第 50 章 : XMLUI 对象**

xmlui.accept()	481
xmlui.cancel()	482
xmlui.get()	482
xmlui.getControlItemElement()	483
xmlui.setEnabled()	483
xmlui.getVisible()	484
xmlui.set()	484
xmlui.setControlItemElement()	485
xmlui.setControlItemElements()	486
xmlui.setEnabled()	486
xmlui.setVisible()	487

**第 51 章 : C 级可扩展性**

关于可扩展性 .....	488
集成 C 函数 .....	488
数据类型 .....	493
C 级 API .....	494



# 第 1 章：简介

作为 Adobe® Flash® Professional CS5 或 CS5.5 的用户，您可能熟悉 Adobe® ActionScript®，它可用于创建运行时在 Adobe® Flash® Player 中执行的脚本。本文档中介绍的 Flash JavaScript 应用程序编程接口 (JavaScript API) 是一个辅助编程工具，该工具可用于创建在创作环境中运行的脚本。

本文档介绍 JavaScript API 中提供的对象、方法和属性，并假定您了解在创作环境中工作时如何使用文档化的命令。如果您对特定命令的作用持有疑问，请使用 Flash 帮助中的其他文档（如《使用 Flash》）来查找该信息。

本文档还假定您熟悉 JavaScript 或 ActionScript 语法以及函数、参数和数据类型等基本编程概念。

## 使用 JavaScript API

利用 Flash JavaScript API 可以编写在 Flash 创作环境中（即用户保持 Flash 程序打开期间）执行多个动作的脚本。此功能与 ActionScript 语言不同，利用 ActionScript 语言可以编写能够在 Flash Player 环境中（即播放 SWF 文件期间）执行动作的脚本。此功能与 JavaScript 命令也不同，JavaScript 命令可以在用 Web 浏览器显示的页面中使用。

利用 JavaScript API，您可以编写有助于简化创作过程的 Flash 应用程序脚本。例如，您可以编写脚本来自动执行重复性任务，或向“工具”面板添加自定义工具。

Flash JavaScript API 在设计上类似于 Adobe® Dreamweaver® 和 Adobe® Fireworks® JavaScript API（它们基于 Netscape JavaScript API 而设计）。Flash JavaScript API 基于文档对象模型 (DOM)，该模型允许使用 JavaScript 对象访问 Flash 文档。Flash JavaScript API 包含 Netscape JavaScript API 及 Flash DOM 的所有元素。本文档将对这些新增的对象及其方法和属性进行说明。您可以在 Flash 脚本中使用本地 JavaScript 语言的任何元素，但只有在 Flash 文档上下文中有意义的元素才有效。

此外，JavaScript API 还包含一些方法，使您可组合使用 JavaScript 和自定义 C 代码来实现可扩展性。有关详细信息，请参阅第 488 页的“C 级可扩展性”。

Flash 中的 JavaScript 解释程序采用的是 Mozilla SpiderMonkey 引擎 1.6 版，该版本可从网络中找到：[www.mozilla.org/js/spidermonkey/](http://www.mozilla.org/js/spidermonkey/)。SpiderMonkey 是 Mozilla.org 开发的 JavaScript 语言的两种引用实现之一。它与嵌入 Mozilla 浏览器中的引擎相同。

SpiderMonkey 按照 ECMAScript (ECMA-262) 版本 3 语言规范中的定义，实现了核心 JavaScript 语言，并完全符合该规范的要求。只有那些特定于浏览器的主机对象（未包含在 ECMA-262 规范中）不受支持。同样，许多 JavaScript 参考指南都区分核心 JavaScript 和客户端（与浏览器相关的）JavaScript。只有核心 JavaScript 才适用于 Flash JavaScript 解释程序。

## 创建 JSFL 文件

可以使用 Adobe Flash Professional 或喜欢使用的文本编辑器来编写和编辑 Flash JavaScript (JSFL) 文件。如果使用 Flash，则这些文件的默认扩展名为 .jsfl。若要编写脚本，请选择“文件”>“新建”>“Flash JavaScript 文件”。

另外，还可以在“历史记录”面板中选择命令，从而创建 JSFL 文件。然后，单击“历史记录”面板中的“保存”按钮，或在面板菜单中选择“另存为命令”。命令 (JSFL) 文件保存在 Commands 文件夹中（请参阅第 2 页的“保存 JSFL 文件”）。这样就可以像打开和编辑任何其他脚本文件一样处理该文件。

“历史记录”面板还提供了其他一些有用的选项。可以将所选命令复制到剪贴板，并且可以查看在 Flash 中工作时生成的 JavaScript 命令。

将命令从“历史记录”面板复制到剪贴板：

- 1 在“历史记录”面板中选择一个或多个命令。

2 请执行下列操作之一：

- 单击“复制”按钮。
- 从面板菜单中选择“复制步骤”。

在“历史记录”面板中查看 **JavaScript** 命令：

- 从面板菜单中选择“查看”>“在面板中显示 JavaScript”。

## 保存 JSFL 文件

您可以通过将 JSFL 脚本存储在 **Configuration** 文件夹内的多个文件夹之一中，使它们可用于 **Flash** 创作环境。默认情况下，**Configuration** 文件夹位于以下位置：

- Windows® 7™：

引导驱动器 \Users\ 用户名 \AppData\Local\Adobe\Flash CS5 或 Flash CS5.5\ 语言 \Configuration\

- Windows® Vista™：

引导驱动器 \Users\ 用户名 \Local Settings\Application Data\Adobe\Flash CS5 或 Flash CS5.5\ 语言 \Configuration\

- Windows XP：

引导驱动器 \Documents and Settings\ 用户名 \Local Settings\Application Data\Adobe\Flash CS5 或 Flash CS5.5\ 语言 \Configuration\

- Mac OS® X：

Macintosh HD/Users/ 用户名 /Library/Application Support/Adobe/Flash CS5 或 Flash CS5.5/ 语言 /Configuration/

若要确定 **Configuration** 文件夹的位置，请使用 `fl.configDirectory` 或 `fl.configURI`，如下面的示例所示：

```
// store directory to a variable
var configDir = fl.configDirectory;
// display directory in the Output panel
fl.trace(fl.configDirectory);
```

**Configuration** 文件夹中的以下文件夹可能包含您可在创作环境中访问的脚本：**Behaviors**（用于支持行为的用户界面）；**Commands**（用于在“命令”菜单上显示的脚本）；**JavaScript**（脚本助手用来填充用户界面控件的脚本）；**Tools**（用于“工具”面板中的可扩展工具）和 **WindowSWF**（用于 **Windows** 菜单中显示的面板）。本文档主要介绍用于命令和工具的脚本。

如果编辑 **Commands** 文件夹中的脚本，新脚本随即便可用于 **Flash**。如果编辑用于可扩展工具的脚本，请关闭并重新启动 **Flash**，或者使用 `fl.reloadTools()` 命令。不过，如果您曾使用脚本向“工具”面板中添加可扩展工具，然后编辑脚本，请删除该工具，然后重新向“工具”面板中添加该工具，或者关闭并重新启动 **Flash**，以使修改后的工具可用。

您可以将命令和工具文件存储在两个位置，以便在创作环境中访问这些文件。

- 对于作为菜单项显示在“命令”菜单中的脚本，请将 JSFL 文件保存在以下位置的 **Commands** 文件夹中：

操作系统	位置
Windows 7	引导驱动器 \Users\ 用户名 \AppData\Local\Adobe\Flash CS5 或 Flash CS5.5\ 语言 \Configuration\Commands
Windows Vista	引导驱动器 \Users\ 用户名 \Local Settings\Application Data\Adobe\Flash CS5\ 语言 \Configuration\Commands
Windows XP	引导驱动器 \Documents and Settings\ 用户 \Local Settings\Application Data\Adobe\Flash CS5\ 语言 \Configuration\Commands
Mac OS X	Macintosh HD/Users/ 用户名 /Library/Application Support/Adobe/Flyout CS5/ 语言 /Configuration/Commands

- 对于作为可扩展工具显示在“工具”面板中的脚本，请将 JSFL 文件保存在以下位置的 Tools 文件夹中：

操作系统	位置
Windows 7	引导驱动器 \Users\ 用户名 \AppData\Local\Adobe\Flyout CS5 或 Flash CS5.5\ 语言 \Configuration\Tools
Windows Vista	引导驱动器 \Users\ 用户名 \Local Settings\Application Data\Adobe\Flyout CS5\ 语言 \Configuration\Tools
Windows XP	引导驱动器 \Documents and Settings\ 用户 \Local Settings\Application Data\Adobe\Flyout CS5\ 语言 \Configuration\Tools
Mac OS X	Macintosh HD/Users/ 用户名 /Library/Application Support/Adobe/Flyout CS5/ 语言 /Configuration/Tools

如果 JSFL 文件附带了其他文件（如 XML 文件），请将这些文件与 JSFL 文件存储在同一目录中。

## 运行脚本

有多种方法可以运行脚本。本节介绍几种最常用的方法。

运行您当前查看或编辑的脚本：

- 右键单击（在 Macintosh 中按住 Command 单击），然后选择“运行脚本”。
- 在“脚本”窗口工具栏中单击“运行脚本”图标。

使用此选项可以在保存脚本之前运行脚本。即便没有打开任何 FLA 文件，也可利用此选项运行脚本。

若要运行 **Commands** 文件夹中的脚本，请按以下方法之一操作：

- 在创作环境中，选择“命令”>“脚本名称”。
- 使用分配给脚本的快捷键。若要分配快捷键，请使用“编辑”>“快捷键”，然后从“命令”弹出菜单中选择“绘画菜单命令”。展开菜单树中的“命令”节点以查看可用脚本的列表。

若要运行不在 **Commands** 文件夹中的命令脚本，请按以下方法之一操作：

- 在创作环境中，选择“命令”>“运行命令”，然后选择要运行的脚本。
- 在脚本中，使用 `fl.runScript()` 命令。
- 在文件系统中，双击脚本文件。

将在 JSFL 文件中实现的工具添加到“工具”面板中：

- 1 将该工具的 JSFL 文件和所有其他关联文件复制到 Tools 文件夹中（请参阅第 2 页的“保存 JSFL 文件”）。
- 2 选择“编辑”>“自定义工具面板”(Windows) 或“Flash”>“自定义工具面板”(Macintosh)。

3 将该工具添加到可用工具的列表中。

4 单击“确定”。

您可以通过使用 `MMEExecute()` 函数，将各个 JavaScript API 命令添加到 ActionScript 文件中，具体说明请参阅《ActionScript 3.0 语言和组件参考》。不过，`MMEExecute()` 函数仅在自定义用户界面元素（如创作环境中的组件属性检查器或 SWF 面板）的上下文中使用时才有效。在 Flash Player 中或在创作环境外部，即使从 ActionScript 进行了调用，JavaScript API 命令也无效。

从 ActionScript 脚本发出命令：

- 使用以下语法（可以将多个命令连接成一个字符串）：

```
MMEExecute(Javascript command string);
```

还可以从命令行运行脚本。

从 Windows 命令行运行脚本：

- 使用以下语法（根据需要添加路径信息）：

```
"flash.exe" myTestFile.jsfl
```

从 Macintosh 的“终端”应用程序中运行脚本：

- 使用以下语法（根据需要添加路径信息）：

```
osascript -e 'tell application "flash" to open alias "Mac OS X:Users:user:myTestFile.jsfl" '
```

osascript 命令还可以运行文件中的 AppleScript。例如，可以在名为 myScript 的文件中放置以下文本：

```
tell application "flash"
open alias "Mac OS X:Users:user:myTestFile.jsfl"
end tell
```

以后在运行脚本时，可以使用下面的命令：

```
osascript myScript
```

## JavaScript API 中的新增内容

在 Flash CS5 和 Flash CS5.5 中，新增和删除了一些对象、方法和属性。下面概述了这些更改。

如果您以前未使用过 JavaScript API，则最好跳过这一节，直接转到第 6 页的“JavaScript API 对象”。

### 新增方法和属性

Flash Pro CS5 中为现有对象新增了以下方法和属性：

- [Document 对象](#)
  - `document.debugMovie()`
  - `document.loadCuepointXML()`
- [flash 对象 \(fl\)](#)
  - `fl.languageCode`
  - `fl.toggleBreakpoint`
- [Frame 对象](#)
  - `frame.convertMotionObjectTo2D()`

- `frame.convertMotionObjectTo3D()`
- `frame.getMotionObjectXML()`
- `frame.hasMotionPath()`
- `frame.isMotionObject()`
- `frame.is3DMotionObject()`
- `frame.selectMotionPath()`
- `frame.setMotionObjectDuration()`
- `frame.setMotionObjectXML()`
- `frame.tweenInstanceName`
- **Timeline 对象**
  - `timeline.createMotionObject()`
  - `timeline.libraryItem`
  - `timeline.removeMotionObject()`
  - `timeline.startPlayback`
  - `timeline.stopPlayback`

Flash Pro CS5.5 中为现有对象新增了以下方法和属性：

- **SymbolInstance 对象**
  - `symbolInstance.bitmapRenderMode`
  - `symbolInstance.backgroundColor`
  - `symbolInstance.usesBackgroundColor`
  - `symbolInstance.visible`
- **Timeline 对象**
  - `timeline.copyLayers()`
  - `timeline.cutLayers()`
  - `timeline.duplicateLayers()`
  - `timeline.pasteLayers()`
- **flash 对象 (fl)**
  - `fl.getSwfPanel()`
  - `fl.installedPlayers()`
  - `fl.publishCacheEnabled`
  - `fl.publishCacheDiskSizeMax`
  - `fl.publishCacheMemorySizeMax`
  - `fl.publishCacheMemoryEntrySizeLimit`
  - `fl.clearPublishCache()`
- **swfPanel 对象**
  - `swfPanel.setFocus()`

## 其他更改

Flash CS5 中更新了下列方法和属性：

- fl.openScript()
- fl.publishDocument()
- fontItem.embedRanges
- fontItem.embeddedCharacters
- fontItem.embedVariantGlyphs

Flash CS5 中不再提供下列对象和方法：

- Screen object
- ScreenOutline object
- document.canSaveAVersion()
- document.revertToLastVersion()
- document.saveAVersion()
- document.synchronizeWithHeadVersion()
- fl.downloadLatestVersion()
- fl.revertDocumentToLastVersion()
- fl.saveAVersionOfDocument()
- fl.synchronizeDocumentWithHeadVersion()

## JavaScript API 对象

本节概要介绍 Flash JavaScript API 中的可用对象，以及如何开始使用它们。使用 JavaScript API 时，所有标准 JavaScript 命令均可用。

下表简要说明 JavaScript API 中的每个对象。这些对象按字母顺序列出。

对象	描述
actionsPanel 对象	actionsPanel 对象表示当前显示的“动作”面板。
BitmapInstance 对象	BitmapInstance 对象是 Instance 对象的子类，它表示帧中的一个位图。
BitmapItem 对象	BitmapItem 对象是指文档库中的位图。BitmapItem 对象是 Item 对象的子类。
CompiledClipInstance 对象	CompiledClipInstance 对象是 Instance 对象的子类。
compilerErrors 对象	compilerErrors 对象表示“编译器错误”面板。它是 flash 对象的属性 (fl.compilerErrors)。
ComponentInstance 对象	ComponentInstance 对象是 SymbolInstance 对象的子类，表示帧中的一个组件。
componentsPanel 对象	componentsPanel 对象表示“组件”面板，它是 flash 对象 (fl.componentsPanel) 的属性。
Contour 对象	Contour 对象表示由形状边界上的半边缘组成的封闭路径。
Document 对象	Document 对象表示舞台。
drawingLayer 对象	drawingLayer 对象可以从 JavaScript 作为 flash 对象的子对象访问。
Edge 对象	Edge 对象表示舞台上一个形状的边缘。

对象	描述
Element 对象	出现在舞台上的所有对象都是 <b>Element</b> 类型。
Fill 对象	<b>Fill</b> 对象包含“工具”面板或某一选定形状的填充颜色设置的所有属性。
Filter 对象	<b>Filter</b> 对象包含所有滤镜的全部属性。
flash 对象 (fl)	<b>flash</b> 对象表示 <b>Flash</b> 应用程序。
FLfile 对象	<b>FLfile</b> 对象允许您编写可对本地文件系统中的文件和文件夹进行访问、修改和删除操作的 <b>Flash</b> 扩展。
folderItem 对象	<b>folderItem</b> 对象是 <b>Item</b> 对象的子类。
fontItem 对象	<b>fontItem</b> 对象是 <b>Item</b> 对象的子类。
Frame 对象	<b>Frame</b> 对象表示图层中的帧。
HalfEdge 对象	<b>Shape</b> 对象的边缘的有向侧。
Instance 对象	<b>Instance</b> 对象是 <b>Element</b> 对象的子类。
Item 对象	<b>Item</b> 对象是一种抽象基类。
Layer 对象	<b>Layer</b> 对象表示时间轴中的图层。
library 对象	<b>library</b> 对象表示“库”面板。
Math 对象	<b>Math</b> 对象可用作 <b>flash</b> 对象 ( <b>fl.Math</b> ) 的只读属性。
Matrix 对象	<b>Matrix</b> 对象表示一个变形矩阵。
outputPanel 对象	<b>outputPanel</b> 对象表示“输出”面板，它用来显示语法错误等疑难解答信息。它是 <b>flash</b> 对象 ( <b>fl.outputPanel</b> ) 的属性。
Oval 对象	<b>Oval</b> 对象是使用椭圆工具绘制的形状。若要确定一个项目是否为 <b>Oval</b> 对象，请使用 <b>shape.isOvalObject</b> 。
Parameter 对象	<b>Parameter</b> 对象类型是从 <b>screen.parameters</b> 数组（对应于 <b>Flash</b> 创作工具中的屏幕“属性”检查器）或由 <b>componentInstance.parameters</b> 数组（对应于创作工具中的组件“属性”检查器）访问的。
Path 对象	<b>Path</b> 对象定义线段（直线、曲线或两者）的序列，通常在创建可扩展工具时使用。
presetItem 对象	<b>presetItem</b> 对象表示“动画预设”面板中的项目（预设或文件夹）。
presetPanel 对象	<b>presetPanel</b> 对象表示“动画预设”面板（“窗口”>“动画预设”）。它是 <b>flash</b> 对象 ( <b>fl.presetPanel</b> ) 的属性。
Rectangle 对象	<b>Rectangle</b> 对象是使用“矩形”工具绘制的形状。若要确定一个项目是否为 <b>Rectangle</b> 对象，请使用 <b>shape.isRectangleObject</b> 。
Screen 对象	<b>Screen</b> 对象表示幻灯片或表单文档中的单个屏幕。
ScreenOutline 对象	<b>ScreenOutline</b> 对象表示幻灯片或表单文档中的一组屏幕。
Shape 对象	<b>Shape</b> 对象是 <b>Element</b> 对象的子类。当在舞台上操作或创建几何形状时， <b>Shape</b> 对象提供的控制比绘图 API 提供的控制更精确。
SoundItem 对象	<b>SoundItem</b> 对象是 <b>Item</b> 对象的子类。它表示一个用于创建声音的库项目。
Stroke 对象	<b>Stroke</b> 对象包含笔触的所有设置（包括自定义设置）。
swfPanel 对象	<b>swfPanel</b> 对象表示窗口 SWF 面板。窗口 SWF 面板是一些 SWF 文件，它们实现的应用程序可以在 <b>Flash</b> 创作环境中运行。 <b>swfPanel</b> 对象的数组为 <b>flash</b> 对象 ( <b>fl.swfPanels</b> ) 的属性。
SymbolInstance 对象	<b>SymbolInstance</b> 对象是 <b>Instance</b> 对象的子类，它表示帧中的一个元件。

对象	描述
SymbolItem 对象	SymbolItem 对象是 Item 对象的子类。
Text 对象	Text 对象表示文档中单独的文本项。
TextAttrs 对象	TextAttrs 对象包含能应用于部分选定的文本的所有属性。此对象为 Text 对象的一个子类。
TextRun 对象	TextRun 对象表示一串字符，其属性与 TextAttrs 对象中的所有属性相匹配。
Timeline 对象	Timeline 对象表示 Flash 时间轴，可通过 fl.getDocumentDOM().getTimeline() 访问当前文档的时间轴。
ToolObj 对象	一个 ToolObj 对象表示“工具”面板中的单个工具。
Tools 对象	可从 Flash 对象 (fl.tools) 访问 Tools 对象。
Vertex 对象	Vertex 对象是形状数据结构中保存坐标数据的部分。
VideoItem 对象	VideoItem 对象是 Item 对象的子类。
XMLUI 对象	XMLUI 对象能够获取和设置 XMLUI 对话框的属性，并能接受或取消其中的某个功能。

## Flash 文档对象模型

用于 Flash JavaScript API 的 Flash 文档对象模型 (DOM) 包含一组顶级函数（请参阅第 13 页的“[顶级函数和方法](#)”）及两个顶级对象（FLfile 对象和 flash 对象 (fl)）。由于每个对象在 Flash 创作环境打开时总是存在，因此在脚本中肯定可用。有关详细信息，请参阅 [FLfile 对象](#) 和 [flash 对象 \(fl\)](#)。

在引用 flash 对象时，可以使用 flash 或 fl。例如，若要关闭所有打开的 FLA 文件，可以使用以下任一语句：

```
flash.closeAll();
fl.closeAll();
```

flash 对象包含以下子对象：

对象	访问方法
actionsPanel 对象	使用 fl.actionsPanel 访问 actionsPanel 对象。该对象与 Flash 创作环境中的“动作”面板相对应。
compilerErrors 对象	使用 fl.compilerErrors 访问 compilerErrors 对象。该对象与 Flash 创作环境中的“编译器错误”面板相对应。
componentsPanel 对象	使用 fl.componentsPanel 访问 componentsPanel 对象。该对象与 Flash 创作环境中的“组件”面板相对应。
Document 对象	使用 fl.documents 检索由所有打开文档组成的数组；使用 fl.documents[index] 访问特定文档；使用 fl.getDocumentDOM() 访问当前文档（具有焦点的文档）。
drawingLayer 对象	使用 fl.drawingLayer 访问 drawingLayer 对象。
Math 对象	使用 fl.Math 访问 Math 对象。
outputPanel 对象	使用 fl.outputPanel 访问 outputPanel 对象。该对象与 Flash 创作环境中的“输出”面板相对应。
presetPanel 对象	使用 fl.presetPanel 访问 presetPanel 对象。此对象对应于“动画预设”面板（“窗口”>“动画预设”）。
swfPanel 对象	使用 fl.swfPanels 访问由 swfPanel 对象组成的数组。这些对象对应于“窗口 SWF”面板。
Tools 对象	使用 fl.tools 访问由 Tools 对象组成的数组。
XMLUI 对象	使用 fl.xmlui 访问 XML 用户界面 (XMLUI) 对象。XMLUI 对象能获取和设置 XMLUI 对话框的属性。



## Document 对象

`fl.documents` 属性是顶级 `flash` 对象的一个重要属性。此属性包含一个由 `Document` 对象组成的数组，每个对象代表创作环境中当前打开的一个 `FLA` 文件。每个 `Document` 对象的属性代表一个 `FLA` 文件可包含的大多数元素。因此，`DOM` 主要由 `Document` 对象的子对象和属性组成。有关详细信息，请参阅 [Document 对象](#)。

例如，若要引用首个打开的文档，请使用语句 `flash.documents[0]` 或 `fl.documents[0]`。首个文档是在创作环境中的当前会话期间打开的首个 `Flash` 文档。在关闭首个打开的文档后，其他已打开文档的索引将递减。

若要查找特定文档的索引，请使用 `flash.findDocumentIndex(nameOfDocument)` 或 `fl.findDocumentIndex(nameOfDocument)`。请参阅 [fl.findDocumentIndex\(\)](#)。

若要访问当前具有焦点的文档，请使用语句 `flash.getDocumentDOM()` 或 `fl.getDocumentDOM()`。请参阅 [fl.getDocumentDOM\(\)](#)。后一语句是本文档的示例中最常用的语法。

若要查找 `fl.documents` 数组中的特定文档，请遍历此数组并测试每个文档的 `document.name` 属性。请参阅 [fl.documents](#) 和 [document.name](#)。

`DOM` 中未在上表中列出的所有对象（请参阅第 8 页的“[Flash 文档对象模型](#)”）均从 `Document` 对象访问。例如，若要访问一个文档的库，请使用 `document.library` 属性，该属性可检索一个库对象：

```
fl.getDocumentDOM().library
```

若要访问库中项目的数组，请使用 `library.items` 属性；该数组中的每个元素均是一个 `Item` 对象：

```
fl.getDocumentDOM().library.items
```

若要访问库中的特定项目，请指定 `library.items` 数组中的一个成员：

```
fl.getDocumentDOM().library.items[0]
```

也就是说，`library` 对象是 `Document` 对象的子级，而 `Item` 对象是 `library` 对象的子级。有关详细信息，请参阅 [document.library](#)、[library 对象](#)、[library.items](#)、[library.items](#) 和 [Item 对象](#)。

## 指定动作目标

除非以其他方式指定，否则方法将影响当前焦点或选择范围。例如，由于未指定特定的对象，下面的脚本会将当前选择范围扩大一倍：

```
fl.getDocumentDOM().scaleSelection(2, 2);
```

在某些情况下，您可能希望某一动作专门针对 `Flash` 文档中当前选定的项。为此，请使用 `document.selection` 属性包含的数组（请参阅 [document.selection](#)）。数组中的首个元素表示当前选定的项，如下面的示例所示：

```
var accDescription = fl.getDocumentDOM().selection[0].description;
```

以下脚本将舞台上存储在元素数组中的首个元素（而不是当前选定范围）的大小扩大一倍：

```
var element = fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0];  
if (element) {  
    element.width = element.width*2;  
    element.height = element.height*2;  
}
```

此外，您还可以执行一些操作（如遍历舞台上的所有元素），并按指定的数量增加宽度和高度，如下面的示例所示：

```
var elementArray =  
    fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements;  
for (var i=0; i < elementArray.length; i++) {  
    var offset = 10;  
    elementArray[i].width += offset;  
    elementArray[i].height += offset;  
}
```

## DOM 结构摘要

以下列表以大纲形式显示了 DOM 结构。每行开头的数字表示对象级别。例如，以“03”开头的对象是上一级“02”对象的子级，而后者又是上一级“01”对象的子级。

有些情况下，某对象可以通过指定其父对象的属性提供。例如，`document.timelines` 属性包含由 **Timeline** 对象组成的数组。下面的大纲说明了这些属性。

有些对象是其他对象的子类，而不是其他对象的子对象。除了父对象（超类）的方法和属性之外，作为其他对象子类的对象还有自己的方法和 / 或属性。子类与其超类在层次结构中共享同一级别。例如，**Item** 对象是 **BitmapItem** 对象的超类。下面的大纲阐释了这些关系：

```
01 Top-Level Functions and Methods
01 FLfile object
01 flash object (fl)
  02 compilerErrors object
  02 componentsPanel object
  02 Document object (fl.documents array)
    03 Filter object
    03 Matrix object
    03 Fill object
    03 Stroke object
    03 library object
      04 Item object (library.items array)
      04 BitmapItem object (subclass of Item object)
      04 folderItem object (subclass of Item object)
      04 fontItem object (subclass of Item object)
      04 SoundItem object (subclass of Item object)
      04 SymbolItem object (subclass of Item object)
      04 VideoItem object (subclass of Item object)
    03 Timeline object (document.timelines array)
      04 Layer object (timeline.layers array)
        05 Frame object (layer.frames array)
          06 Element object (frame.elements array)
            07 Matrix object (element.matrix)
          06 Instance object (abstract class, subclass of Element object)
          06 BitmapInstance object (subclass of Instance object)
          06 CompiledClipInstance object (subclass of Instance object)
          06 ComponentInstance object (subclass of SymbolInstance object)
            07 Parameter object (componentInstance.parameters array)
          06 SymbolInstance object (subclass of Instance object)
          06 Text object (subclass of Element object)
            07 TextRun object (text.textRuns array)
              08 TextAttrs object (textRun.textAttrs array)
        06 Shape object (subclass of Element object)
          07 Oval object
          07 Rectangle object
          07 Contour object (shape.contours array)
            08 HalfEdge object
              09 Vertex object
              09 Edge object
```

```
07 Edge object (shape.edges array)
  08 HalfEdge object
  09 Vertex object
  09 Edge object
07 Vertex object (shape.vertices array)
  08 HalfEdge object
  09 Vertex object
  09 Edge object
05 Parameter object (screen.parameters array)
02 drawingLayer object
  03 Path object
  04 Contour object
02 Math object
02 outputPanel object
02 presetPanel object
  03 presetItem object (presetPanel.items array)
02 swfPanel object
02 Tools object (fl.tools array)
  03 ToolObj object (tools.toolObjs array)
02 XMLUI object
```

## 实现范例

Adobe Flash Professional CS5 和 CS5.5 提供了几个示例 JSFL 实施。您可查看并安装这些文件以熟悉 JavaScript API。这些范例位于 Samples.zip 文件（位于 [www.adobe.com/go/learn\\_fl\\_samples\\_cn](http://www.adobe.com/go/learn_fl_samples_cn)）内的 Samples/ExtendingFlash 文件夹中。

### “形状”命令范例

在 ExtendingFlash/Shape 文件夹中有一个名为 Shape.jsfl 的 JavaScript API 脚本范例（请参阅上述“实现范例”）。该脚本在“输出”面板中显示有关形状轮廓的信息。

安装并运行 **Shape** 脚本：

- 1 将 Shape.jsfl 文件复制到 Configuration/Commands 文件夹中（请参阅第 2 页的“保存 JSFL 文件”）。
- 2 在 Flash 文档（FLA 文件）中，选择一个形状对象。
- 3 选择“命令”>“形状”来运行此脚本。

### 获取和设置“滤镜”命令的范例

在 ExtendingFlash/filtersGetSet 文件夹中有一个名为 filtersGetSet.jsfl 的 JavaScript API 脚本范例（请参阅上述“实现范例”）。该脚本为选中的对象添加滤镜，并在“输出”面板中显示有关所添加滤镜的信息。

安装并运行 **filtersGetSet** 脚本：

- 1 将 filtersGetSet.jsfl 文件复制到 Configuration/Commands 文件夹中（请参阅第 2 页的“保存 JSFL 文件”）。
- 2 在 Flash 文档（FLA 文件）中，选择一个文本、影片剪辑或按钮对象。
- 3 选择“命令”>“filtersGetSet”来运行脚本。

### “多角星形”工具范例

在 ExtendingFlash/PolyStar 文件夹中有一个名为 PolyStar.jsfl 的 JavaScript API 脚本范例（请参阅上述“实现范例”）。

PolyStar.jsfl 将复制位于 Flash“工具”面板中的“多角星形”工具。该脚本演示如何使用 JavaScript API 来生成“多角星形”工具，并包含详细的注释来描述代码的作用。请阅读此文件，以便更好地了解如何使用 JavaScript API。此外，您还应阅读 Tools 目录下的 PolyStar.xml 文件，以便更好地了解如何生成您自己的工具。

## “转换位图为矢量图”面板范例

在 ExtendingFlash/TraceBitmapPanel 文件夹中有一组名为 TraceBitmap fla 和 TraceBitmap.swf 的文件（请参阅上述“实现范例”）。这些文件阐释了如何设计和建立面板来控制 Flash 的功能。此外，这些文件还显示了如何使用 MMExecute() 函数从 ActionScript 脚本调用 JavaScript 命令。

运行“转换位图为矢量图”范例：

- 1 如果 Flash 正在运行，请退出 Flash。
- 2 将 TraceBitmap.swf 文件复制到 WindowSWF 文件夹，该文件夹为 Configuration 文件夹的子目录（请参阅第 2 页的“保存 JSFL 文件”）。例如，在 Windows XP 中，该文件夹位于引导驱动器 \Documents and Settings\用户\Local Settings\Application Data\Adobe\Flash CS5\语言\Configuration\WindowSWF 中。
- 3 启动 Flash。
- 4 创建或打开一个 Flash 文档（FLA 文件），然后将一个位图或 JPEG 图像导入到该文件中。  
可以使用 TraceBitmapPanel 文件夹中提供的 flower.jpg 文件，或者使用您选择的其他图像。
- 5 选定要导入的图像后，选择“窗口”>“其他面板”>“转换位图为矢量图”。
- 6 单击“提交”。

该图像将转换为一组形状。

## DLL 范例

在 ExtendingFlash/dllSampleComputeSum 文件夹中有一个 DLL 实现范例（请参阅上述“实现范例”）。有关生成 DLL 的详细信息，请参阅第 488 页的“C 级可扩展性”。

## 第 2 章：顶级函数和方法

### 本节内容

本节介绍在使用 Adobe Flash JavaScript 应用程序编程接口 (JavaScript API) 时可用的顶级函数和方法。有关 JavaScript API 文件存储位置的信息，请参阅第 2 页的“[保存 JSFL 文件](#)”。

### 全局方法

可从任何 JavaScript API 脚本调用以下方法：

```
alert()
confirm()
prompt()
```

### 可扩展工具

在创建可扩展工具的脚本中，可调用以下函数：

```
activate()
configureTool()
deactivate()
keyDown()
keyUp()
mouseDoubleClick()
mouseDown()
mouseMove()
mouseUp()
notifySettingsChanged()
setCursor()
```

## activate()

### 可用性

Flash MX 2004。

### 用法

```
function activate() {
    // statements
}
```

### 参数

无。

### 返回

无。

### 描述

函数；当可扩展工具变为活动状态（即在“工具”面板中选中该工具）时调用。使用此函数来执行工具要求的所有初始化任务。

示例

下面的示例在“工具”面板中选择可扩展工具时设置 `tools.activeTool` 的值：

```
function activate() {  
    var theTool = fl.tools.activeTool  
}
```

另请参见

[tools.activeTool](#)

## alert()

可用性

Flash MX 2004。

用法

```
alert ( alertText )
```

参数

**alertText** 一个字符串，指定要在“警告”对话框中显示的消息。

返回

无。

描述

方法；在模式“警告”对话框中显示一个字符串和一个“确定”按钮。

示例

下面的示例在“警告”对话框中显示消息“Process Complete”：

```
alert("Process Complete");
```

另请参见

[confirm\(\)](#)、[prompt\(\)](#)

## configureTool()

可用性

Flash MX 2004。

用法

```
function configureTool() {  
    // statements  
}
```

参数

无。

返回  
无。

描述

函数；在 **Flash** 打开并将可扩展工具加载到“工具”面板时调用。使用此函数，可以设置 **Flash** 关于该工具所要了解的任何信息。

示例

下面的示例显示此函数的两种可能的实现：

```
function configureTool() {  
    theTool = fl.tools.activeTool;  
    theTool.setToolName("myTool");  
    theTool.setIcon("myTool.png");  
    theTool.setMenuString("My Tool's menu string");  
    theTool.setToolTip("my tool's tool tip");  
    theTool.setOptionsFile( "mtTool.xml" );  
}
```

```
function configureTool() {  
    theTool = fl.tools.activeTool;  
    theTool.setToolName("ellipse");  
    theTool.setIcon("Ellipse.png");  
    theTool.setMenuString("Ellipse");  
    theTool.setToolTip("Ellipse");  
    theTool.showTransformHandles( true );  
}
```

## confirm()

可用性

Flash 8。

用法

```
confirm ( strAlert )
```

参数

**strAlert** 一个字符串，指定要在“警告”对话框中显示的消息。

返回

一个布尔值：如果用户单击“确定”，则返回 **true**；如果用户单击“取消”，则返回 **false**。

说明

方法；在模式“警告”对话框中显示一个字符串，同时还显示一个“确定”按钮和一个“取消”按钮。

注：如果未打开任何文档（**FLA** 文件），此方法将失败并显示错误条件。

示例

下面的示例在“警告”对话框中显示消息“Sort data?”：

```
confirm("Sort data?");
```

另请参见

[alert\(\)](#)、[prompt\(\)](#)

## deactivate()

可用性

Flash MX 2004。

用法

```
function deactivate() {  
    // statements  
}
```

参数

无。

返回

无。

描述

函数；在可扩展工具变成非活动状态（即：当处于活动状态的工具从该工具变为另一个工具）时调用。此函数用于执行该工具所需的任何清除工作。

示例

下面的示例在工具变为非活动状态时，会在“输出”面板中显示一条消息：

```
function deactivate() {  
    fl.trace( "Tool is no longer active" );  
}
```

## keyDown()

可用性

Flash MX 2004。

用法

```
function keyDown() {  
    // statements  
}
```

参数

无。

返回

无。



#### 描述

函数；在可扩展工具处于活动状态且用户按下一个键时调用。此脚本应调用 [tools.getKeyDown\(\)](#) 来确定按下了哪个键。

#### 示例

下面的示例在可扩展工具处于活动状态且用户按下一个键时，显示哪个键被按下的相关信息。

```
function keyDown() {  
    fl.trace("key " + fl.tools.getKeyDown() + " was pressed");  
}
```

另请参见

[keyUp\(\)](#)、[tools.getKeyDown\(\)](#)

## keyUp()

#### 可用性

Flash MX 2004。

#### 用法

```
function keyUp() {  
    // statements  
}
```

#### 参数

无。

#### 返回

无。

#### 描述

函数；在可扩展工具处于活动状态且释放一个键时调用。

#### 示例

下面的示例在可扩展工具处于活动状态且释放一个键时，会在“输出”面板中显示一条消息。

```
function keyUp() {  
    fl.trace("Key is released");  
}
```

另请参见

[keyDown\(\)](#)

## mouseDoubleClick()

#### 可用性

Flash MX 2004。

#### 用法

```
function mouseDoubleClick() {  
    // statements  
}
```

#### 参数

无。

#### 返回

无。

#### 描述

函数；在可扩展工具处于活动状态且在舞台上双击鼠标按钮时调用。

#### 示例

下面的示例在可扩展工具处于活动状态且双击鼠标按钮时，会在“输出”面板中显示一条消息。

```
function mouseDoubleClick() {  
    fl.trace("Mouse was double-clicked");  
}
```

## mouseDown()

#### 可用性

Flash MX 2004。

#### 用法

```
function mouseDown( [ pt ] ) {  
    // statements  
}
```

#### 参数

**pt** 一个点，当按下鼠标按钮时指定鼠标位置。当按下鼠标按钮时，该点被传递给此函数。此参数是可选的。

#### 返回

无。

#### 描述

函数；在可扩展工具处于活动状态且指针在舞台上时按下鼠标按钮时调用。

#### 示例

下面的示例显示在可扩展工具处于活动状态时，如何使用此函数。第一个示例在按下鼠标按钮时，会在“输出”面板中显示一条消息。第二个示例显示按下鼠标按钮时鼠标位置的 **x** 和 **y** 坐标。

```
function mouseDown() {  
    fl.trace("Mouse button has been pressed");  
}  
function mouseDown(pt) {  
    fl.trace("x = "+ pt.x+" :: y = "+pt.y);  
}
```

## mouseMove()

可用性

Flash MX 2004。

用法

```
function mouseMove( [ pt ] ) {  
    // statements  
}
```

参数

**pt** 一个点，指定鼠标的当前位置。该点跟踪鼠标位置，只要鼠标移动，它就被传递给此函数。如果舞台处于“编辑”或者“在当前位置编辑”模式，则点坐标相对于正被编辑的对象。否则，该点坐标相对于舞台。此参数是可选的。

返回

无。

描述

函数；每当可扩展工具处于活动状态且鼠标滑过舞台上指定的点时调用。不论鼠标按钮是否被按下。

示例

下面的示例说明如何使用此函数。第一个示例在移动鼠标时，会在“输出”面板中显示一条消息。第二个示例显示鼠标移动时鼠标位置的 **x** 和 **y** 坐标。

```
function mouseMove() {  
    fl.trace("moving");  
}  
  
function mouseMove(pt) {  
    fl.trace("x = " + pt.x + " :: y = " + pt.y);  
}
```

## mouseUp()

可用性

Flash MX 2004。

用法

```
function mouseUp() {  
    // statements  
}
```

参数

无。

返回

无。

#### 描述

函数；每当可扩展工具处于活动状态且鼠标按钮在舞台上按下后被释放时调用。

#### 示例

下面的示例在可扩展工具处于活动状态且释放鼠标按钮时，会在“输出”面板中显示一条消息。

```
function mouseUp() {  
    fl.trace("mouse is up");  
}
```

## notifySettingsChanged()

#### 可用性

Flash MX 2004。

#### 用法

```
function notifySettingsChanged() {  
    // statements  
}
```

#### 参数

无。

#### 返回

无。

#### 描述

函数；在可扩展工具处于活动状态且用户在“属性”检查器中更改其选项时调用。您可以使用 `tools.activeTool` 属性来查询选项的当前值（请参阅 [tools.activeTool](#)）。

#### 示例

下面的示例在可扩展工具处于活动状态且用户在“属性”检查器中更改其选项时，会在“输出”面板中显示一条消息。

```
function notifySettingsChanged() {  
    var theTool = fl.tools.activeTool;  
    var newValue = theTool.myProp;  
}
```

## prompt()

#### 可用性

Flash MX 2004。

#### 用法

```
prompt(promptMsg [,text])
```

#### 参数

`promptMsg` 显示在“提示”对话框中的字符串（Mac OS X 中不超过 256 个字符）。

**text** 一个可选字符串，显示为文本字段的默认值。

返回

如果用户单击“确定”，则为用户键入的字符串；如果用户单击“取消”，则为 **null**。

说明

方法；在模式“警告”对话框中显示一个提示和可选文本，同时还显示一个“确定”按钮和一个“取消”按钮。

示例

下面的示例提示用户输入用户名。如果用户键入名称并单击“确定”，该名称将显示在“输出”面板中。

```
var userName = prompt("Enter user name", "Type user name here");  
fl.trace(userName);
```

另请参见

[alert\(\)](#)、[confirm\(\)](#)

## setCursor()

可用性

Flash MX 2004。

用法

```
function setCursor() {  
    // statements  
}
```

参数

无。

返回

无。

描述

函数；在可扩展工具处于活动状态且鼠标在移动时调用，以便允许脚本设置自定义指针。此脚本应调用 **tools.setCursor()** 来指定要使用的指针。有关指针与整数值的对应关系的列表，请参阅 [tools.setCursor\(\)](#)。

示例

```
function setCursor() {  
    fl.tools.setCursor( 1 );  
}
```

## 第 3 章 : actionsPanel 对象

可用性

Flash CS3 Professional。

描述

`actionsPanel` 对象表示当前显示的“动作”面板，是 `flash` 对象的属性（请参阅 [fl.actionsPanel](#)）。

方法摘要

以下方法可用于 `actionsPanel` 对象：

方法	描述
<a href="#">actionsPanel.getClassForObject()</a>	返回指定变量的类。
<a href="#">actionsPanel.getScriptAssistMode()</a>	指定是否启用“脚本助手”模式。
<a href="#">actionsPanel.getSelectedText()</a>	返回“动作”面板中当前选定的文本。
<a href="#">actionsPanel.getText()</a>	返回“动作”面板中的文本。
<a href="#">actionsPanel.hasSelection()</a>	指定在“动作”面板中当前是否选定了任何文本。
<a href="#">actionsPanel.replaceSelectedText()</a>	用指定的文本替换当前选定的文本。
<a href="#">actionsPanel.setScriptAssistMode()</a>	启用或禁用“脚本助手”模式。
<a href="#">actionsPanel.setSelection()</a>	选择“动作”面板中的指定字符集。
<a href="#">actionsPanel.setText()</a>	清除“动作”面板中的任何文本，然后添加指定文本。

### actionsPanel.getClassForObject()

可用性

Flash CS3 Professional。

用法

```
actionsPanel.getClassForObject(ASvariableName)
```

参数

`ASvariableName` 表示 `ActionScript` 变量名称的字符串。

返回

表示 `ASvariableName` 为其成员的类的字符串。

说明

方法；返回指定变量的类，该类必须在当前显示的“动作”面板中定义。另外，“动作”面板中的光标或所选文本必须位于变量定义之后。

#### 示例

下例显示分配给变量 `myVar` 的类（如果“动作”面板中的光标位于 `var myVar:ActivityEvent;` 语句之后）。

```
// Place the following code in the Actions panel,  
// and position the cursor somewhere after the end of the line  
var myVar:ActivityEvent;  
// Place the following code in the JSFL file  
var theClass = fl.actionsPanel.getClassForObject("myVar");  
fl.trace(theClass); // traces: "ActivityEvent"
```

## actionsPanel.getScriptAssistMode()

#### 可用性

Flash CS3 Professional。

#### 用法

```
actionsPanel.getScriptAssistMode()
```

#### 参数

无。

#### 返回

一个布尔值，指定“脚本助手”模式是处于启用状态 (`true`) 还是禁用状态 (`false`)。

#### 说明

方法；指定是否启用“脚本助手”模式。

#### 示例

如果“脚本助手”模式未启用，下例显示一条消息。

```
mAssist = fl.actionsPanel.getScriptAssistMode();  
if (!mAssist) {  
    alert("For more guidance when writing ActionScript code, try Script Assist mode");  
}
```

#### 另请参见

[actionsPanel.setScriptAssistMode\(\)](#)

## actionsPanel.getSelectedText()

#### 可用性

Flash CS3 Professional。

#### 用法

```
actionsPanel.getSelectedText()
```

#### 参数

无。

**actionsPanel** 对象

返回

包含“动作”面板中当前选定文本的字符串。

说明

方法；返回“动作”面板中当前选定的文本。

示例

下例显示“动作”面板中当前选定的文本。

```
var apText = fl.actionsPanel.getSelectedText();  
fl.trace(apText);
```

另请参见

[actionsPanel.getText\(\)](#)、[actionsPanel.hasSelection\(\)](#)、[actionsPanel.replaceSelectedText\(\)](#)、[actionsPanel.setSelection\(\)](#)

## actionsPanel.getText()

可用性

Flash CS3 Professional。

用法

```
actionsPanel.getText()
```

参数

无。

返回

包含“动作”面板中所有文本的字符串。

说明

方法；返回“动作”面板中的文本。

示例

下例显示“动作”面板中的文本。

```
var apText = fl.actionsPanel.getText();  
fl.trace(apText);
```

另请参见

[actionsPanel.getSelectedText\(\)](#)、[actionsPanel.setText\(\)](#)

## actionsPanel.hasSelection()

可用性

Flash CS3 Professional。



#### 用法

```
actionsPanel.hasSelection()
```

#### 参数

无。

#### 返回

一个布尔值，指定“动作”面板中是否选定了任何文本 (**true**) 还是没有选定文本 (**false**)。

#### 说明

方法；指定“动作”面板中当前是否选定了任何文本。

#### 示例

下例显示“动作”面板中当前选定的文本。如果未选择任何文本，则会显示“动作”面板中的所有文本。

```
if (fl.actionsPanel.hasSelection()) {  
    var apText = fl.actionsPanel.getSelectedText();  
}  
else {  
    var apText = fl.actionsPanel.getText();  
}  
fl.trace(apText);
```

#### 另请参见

[actionsPanel.getSelectedText\(\)](#)、[actionsPanel.getText\(\)](#)、[actionsPanel.replaceSelectedText\(\)](#)、[actionsPanel.setSelection\(\)](#)

## actionsPanel.replaceSelectedText()

#### 可用性

Flash CS3 Professional。

#### 用法

```
actionsPanel.replaceSelectedText(replacementText)
```

#### 参数

**替换文本** 表示要用其替换“动作”面板中的选定文本的文本。

#### 返回

一个布尔值。如果找到“动作”面板，则为 **true**；否则为 **false**。

#### 说明

方法；用 **replacementText** 中指定的文本替换当前选定的文本。如果 **replacementText** 包含的字符多于选定文本包含的字符，则选定文本之后的所有字符现在将跟在 **replacementText** 之后；也就是说，不会覆盖这些字符。

#### 示例

下例替换“动作”面板中当前选定的文本。

```
if (fl.actionsPanel.hasSelection()) {  
    fl.actionsPanel.replaceSelectedText("// © 2006 Adobe Inc.");  
}
```

另请参见

[actionsPanel.getSelectedText\(\)](#)、[actionsPanel.hasSelection\(\)](#)、[actionsPanel.setSelection\(\)](#)、[actionsPanel.setText\(\)](#)

## actionsPanel.setScriptAssistMode()

可用性

Flash CS3 Professional。

用法

```
actionsPanel.setScriptAssistMode(bScriptAssist)
```

参数

**bScriptAssist** 一个布尔值，指定启用还是禁用“脚本助手”模式。

返回

一个布尔值，指定是否已成功启用或禁用“脚本助手”模式。

说明

方法；启用或禁用“脚本助手”模式。

示例

下例切换“脚本助手”模式的状态。

```
fl.trace(fl.actionsPanel.getScriptAssistMode());  
if (fl.actionsPanel.getScriptAssistMode()){  
    fl.actionsPanel.setScriptAssistMode(false);  
}  
else {  
    fl.actionsPanel.setScriptAssistMode(true);  
}  
fl.trace(fl.actionsPanel.getScriptAssistMode());
```

另请参见

[actionsPanel.getScriptAssistMode\(\)](#)

## actionsPanel.setSelection()

可用性

Flash CS3 Professional。

用法

```
actionsPanel.setSelection(startIndex, numberOfChars)
```

参数

**startIndex** 一个从零开始的整数，指定要选择的第一个字符。

**numberOfChars** 一个整数，指定要选择多少个字符。

#### 返回

一个布尔值，指定是可以选择所需字符 (**true**) 还是不可以选择 (**false**)。

#### 说明

方法；选择“动作”面板中的指定字符集。

#### 示例

下例用指定文本替换“动作”面板中的字符“2006”。

```
// Type the following as the first line in the Actions panel
// 2006 - Addresses user request 40196
// Type the following in the JSFL file
fl.actionsPanel.setSelection(3,4);
fl.actionsPanel.replaceSelectedText("// Last updated: 2007");
```

#### 另请参见

[actionsPanel.getSelectedText\(\)](#)、[actionsPanel.hasSelection\(\)](#)、[actionsPanel.replaceSelectedText\(\)](#)

## actionsPanel.setText()

#### 可用性

Flash CS3 Professional。

#### 用法

```
actionsPanel.setText(replacementText)
```

#### 参数

**替换文本** 表示要置于“动作”面板中的文本的字符串。

#### 返回

一个布尔值。如果指定的文本已置入“动作”面板，则为 **true**；否则为 **false**。

#### 说明

方法；清除“动作”面板中的任何文本，然后添加 **replacementText** 中指定的文本。

#### 示例

下例用指定的文本替换当前“动作”面板中的所有文本。

```
fl.actionsPanel.setText("// Deleted this code - no longer needed");
```

#### 另请参见

[actionsPanel.getText\(\)](#)、[actionsPanel.replaceSelectedText\(\)](#)

## 第 4 章 : BitmapInstance 对象

继承关系 [Element 对象](#) > [Instance 对象](#) > [BitmapInstance 对象](#)

可用性

Flash MX 2004。

描述

[BitmapInstance](#) 对象是 [Instance](#) 对象的子类，它表示帧中的一个位图（请参阅 [Instance 对象](#)）。

方法摘要

除 [Instance 对象](#) 方法外，对于 [BitmapInstance](#) 对象还可使用以下方法：

方法	描述
<a href="#">bitmapInstance.getBits()</a>	允许您通过提取位图中的位，对它们进行处理，然后将它们返回 <b>Flash</b> 的方式来创建位图特效。
<a href="#">bitmapInstance.setBits()</a>	设置现有位图元素的位。

属性摘要

除 [Instance 对象](#) 属性外，对于 [BitmapInstance](#) 对象还可使用以下属性：

属性	描述
<a href="#">bitmapInstance.hPixels</a>	只读；一个整数，表示位图的宽度（以像素为单位）。
<a href="#">bitmapInstance.vPixels</a>	只读；一个整数，表示位图的高度（以像素为单位）。

### bitmapInstance.getBits()

可用性

Flash MX 2004。

用法

```
bitmapInstance.getBits()
```

参数

无。

返回

一个对象，此对象包含 `width`、`height`、`depth` 和 `bits` 属性，并且，如果此位图有颜色表，则此对象还包含 `cTab` 属性。`bits` 元素是一个字节数组。`cTab` 元素是以 "#RRGGBB" 形式表示的颜色值数组。数组的长度就是颜色表的长度。

此字节数组只有在被 DLL 或共享库引用时才有意义。通常只有在创建可扩展工具或特效时才使用它。有关创建用于 **Flash JavaScript** 的 DLL 的信息，请参阅第 488 页的“[C 级可扩展性](#)”

**BitmapInstance** 对象

## 说明

方法；允许您如下创建位图特效：提取位图中的位，对它们进行处理，然后将它们返回给 **Flash**。

## 示例

下面的代码创建一个对当前选定对象的引用；测试此对象是否为位图；并跟踪位图的高度、宽度和深度：

```
var isBitmap = fl.getDocumentDOM().selection[0].instanceType;
if(isBitmap == "bitmap"){
    var bits = fl.getDocumentDOM().selection[0].getBits();
    fl.trace("height = " + bits.height);
    fl.trace("width = " + bits.width);
    fl.trace("depth = " + bits.depth);
}
```

另请参见

[bitmapInstance.setBits\(\)](#)

## bitmapInstance.hPixels

## 可用性

Flash MX 2004。

## 用法

`bitmapInstance.hPixels`

## 描述

只读属性；一个整数，表示位图的宽度，即水平尺寸的像素数。

## 示例

下面的代码获取位图的宽度（以像素为单位）：

```
// Get the number of pixels in the horizontal dimension.
var bmObj = fl.getDocumentDOM().selection[0];
var isBitmap = bmObj.instanceType;
if(isBitmap == "bitmap"){
    var numHorizontalPixels = bmObj.hPixels;
}
```

另请参见

[bitmapInstance.vPixels](#)

## bitmapInstance.setBits()

## 可用性

Flash MX 2004。

## 用法

`bitmapInstance.setBits(bitmap)`

#### 参数

**位图** 一个对象，包含 height、width、depth、bits 和 cTab 属性。height、width 和 depth 属性为整数。bits 属性是一个字节数组。cTab 属性仅对位图深度小于或等于 8 的位图是必需的，它是以 "#RRGGBB" 形式表示颜色值的字符串。

注：此字节数组只有在被外部库引用时才有意义。通常只有在创建可扩展工具或特效时才使用它。

#### 返回

无。

#### 描述

方法；设置现有位图元素的位。此方法允许您通过提取位图中的位，对它们进行处理，然后将此位图返回 Flash 的方式来创建位图特效。

#### 示例

下面的代码测试当前所选是否为位图，然后将位图的高度设置为 150 像素：

```
var isBitmap = fl.getDocumentDOM().selection[0].instanceType;
if(isBitmap == "bitmap"){
    var bits = fl.getDocumentDOM().selection[0].getBits();
    bits.height = 150;
    fl.getDocumentDOM().selection[0].setBits(bits);
}
```

另请参见

[bitmapInstance.getBits\(\)](#)

## bitmapInstance.vPixels

#### 可用性

Flash MX 2004。

#### 用法

```
bitmapInstance.vPixels
```

#### 描述

只读属性；一个整数，表示位图的高度，即垂直尺寸的像素数。

#### 示例

下面的代码获取位图的高度（以像素为单位）：

```
// Get the number of pixels in the vertical dimension.
var bmObj = fl.getDocumentDOM().selection[0];
var isBitmap = bmObj.instanceType;
if(isBitmap == "bitmap"){
    var numVerticalPixels = bmObj.vPixels;
}
```

另请参见

[bitmapInstance.hPixels](#)

## 第 5 章 : BitmapItem 对象

继承关系 [Item 对象](#) > [BitmapItem 对象](#)

可用性

Flash MX 2004。

描述

[BitmapItem](#) 对象是指文档库中的位图。[BitmapItem](#) 对象是 [Item](#) 对象的子类（请参阅 [Item 对象](#)）。

属性摘要

除 [Item 对象](#) 的属性外，[BitmapItem](#) 对象还具有以下属性：

属性	描述
<a href="#">bitmapItem.allowSmoothing</a>	一个布尔值，它指定是否允许对位图进行平滑处理。
<a href="#">bitmapItem.compressionType</a>	一个字符串，它确定应用于位图的图像压缩类型。
<a href="#">bitmapItem.fileLastModifiedDate</a>	从 1970 年 1 月 1 日至原始文件修改日期之间的秒数。
<a href="#">bitmapItem.originalCompressionType</a>	指定项目是否是以 jpeg 文件格式导入的。
<a href="#">bitmapItem.sourceFileExists</a>	指定之前导入库中的文件现在是否仍位于其导入时的源位置。
<a href="#">bitmapItem.sourceFileIsCurrent</a>	指定库项目的文件修改日期是否与其导入时在磁盘上的文件修改日期相同。
<a href="#">bitmapItem.sourceFilePath</a>	导入库中的文件的路径和名称。
<a href="#">bitmapItem.useDeblocking</a>	指定是否启用消除马赛克功能。
<a href="#">bitmapItem.useImportedJPEGQuality</a>	一个布尔值，它指定是否使用默认的 JPEG 导入品质。

方法摘要

除 [Item 对象](#) 的方法外，[BitmapItem](#) 对象还具有以下方法：

方法	描述
<a href="#">bitmapItem.exportToFile()</a>	将所指定项目导出到 PNG 或 JPG 文件。

### bitmapItem.allowSmoothing

可用性

Flash MX 2004。

用法

```
bitmapItem.allowSmoothing
```

描述

属性；一个布尔值，它指定是 (true) 否 (false) 允许对位图进行平滑处理。

#### 示例

下面的代码将当前文档的库中第一个项目的 `allowSmoothing` 属性设置为 `true`：

```
fl.getDocumentDOM().library.items[0].allowSmoothing = true;  
alert(fl.getDocumentDOM().library.items[0].allowSmoothing);
```

## bitmapItem.compressionType

#### 可用性

Flash MX 2004。

#### 用法

```
bitmapItem.compressionType
```

#### 描述

属性；字符串，用于确定应用于位图的压缩类型。可接受的值为 "photo" 或 "lossless"。如果 `bitmapItem.useImportedJPEGQuality` 的值是 `false`，则 "photo" 使用品质 0 到 100 来对应 JPEG；如果 `bitmapItem.useImportedJPEGQuality` 为 `true`，则 "photo" 使用默认的文档品质值来对应 JPEG。值 "lossless" 对应于 GIF 或 PNG 格式（请参阅 [bitmapItem.useImportedJPEGQuality](#)）。

#### 示例

下面的代码将当前文档的库中第一个项目的 `compressionType` 属性设置为 "photo"：

```
fl.getDocumentDOM().library.items[0].compressionType = "photo";  
alert(fl.getDocumentDOM().library.items[0].compressionType);
```

## bitmapItem.exportToFile()

#### 可用性

Flash CS4 Professional。

#### 用法

```
bitmapItem.exportToFile(fileURI)
```

#### 参数

**fileURI** 一个字符串，表示为 `file:/// URI`，它指定导出的文件的路径和名称。

#### 返回

一个布尔值：如果成功导出文件，则为 `true`；否则为 `false`。

#### 说明

方法；将指定项目导出到 PNG 或 JPG 文件。

#### 示例

假定库中的第一个项目是位图项目，则下面的代码将该项目导出为 JPG 文件：



```
var imageFileURL = "file:///C:/exportTest/out.jpg";  
var libItem = fl.getDocumentDOM().library.items[0];  
libItem.exportToFile(imageFileURL);
```

## bitmapItem.fileLastModifiedDate

可用性

Flash CS4 Professional.

用法

```
bitmapItem.fileLastModifiedDate
```

描述

只读属性；一个字符串，其中包含一个十六进制数字，此数字表示从 1970 年 1 月 1 日至原始文件导入库时的文件修改日期之间的秒数。如果该文件不再存在，则此值为 "00000000"。

示例

假定库中的第一个项目是位图项目，则下面的代码显示一个十六进制数字（如上所述）。

```
var libItem = fl.getDocumentDOM().library.items[0];  
fl.trace("Mod date when imported = " + libItem.fileLastModifiedDate);
```

另请参见

[bitmapItem.sourceFileExists](#)、[bitmapItem.sourceFileIsCurrent](#)、[bitmapItem.sourceFilePath](#)、[FLfile.getModificationDate\(\)](#)

## bitmapItem.originalCompressionType

可用性

Flash CS4 Professional.

用法

```
bitmapItem.originalCompressionType
```

描述

只读属性；一个字符串，它指定所指定项目是否是以 jpeg 文件格式导入的。此属性的可能值包括 "photo"（jpeg 文件）和 "lossless"（未压缩文件类型，如 GIF 和 PNG）。

示例

假定库中第一个项目是位图项目，则如果文件是以 jpeg 文件格式导入库中的，下面的代码显示 "photo"；否则显示 "lossless"：

```
var libItem = fl.getDocumentDOM().library.items[0];  
fl.trace("Imported compression type = " + libItem.originalCompressionType);
```

另请参见

[bitmapItem.compressionType](#)

## bitmapItem.quality

可用性

Flash MX 2004。

用法

`bitmapItem.quality`

描述

属性；整数，用于指定位图的品质。若要使用默认文档品质，则指定为 `-1`；否则指定一个 `0` 到 `100` 范围内的整数。该属性仅对 JPEG 压缩可用。

示例

下面的代码将当前文档的库中第一个项目的 `quality` 属性设置为 `65`：

```
fl.getDocumentDOM().library.items[0].quality = 65;  
alert(fl.getDocumentDOM().library.items[0].quality);
```

## bitmapItem.sourceFileExists

可用性

Flash CS4 Professional.

用法

`bitmapItem.sourceFileExists`

描述

只读属性；一个布尔值，如果之前导入库中的文件现在仍位于其导入时的源位置，则为 `true`；否则为 `false`。

示例

假定库中的第一个项目是位图项目，如果之前导入库中的文件仍然存在，则下面的代码显示 `"true"`。

```
var libItem = fl.getDocumentDOM().library.items[0];  
fl.trace("sourceFileExists = "+ libItem.sourceFileExists);
```

另请参见

[bitmapItem.sourceFilesCurrent](#)、

[bitmapItem.sourceFilePath](#)

## bitmapItem.sourceFilesCurrent

可用性

Flash CS4 Professional.

用法

`bitmapItem.sourceFilesCurrent`

#### 描述

只读属性；一个布尔值，如果库项目的文件修改日期与其导入时在磁盘上的文件修改日期相同，则为 **true**；否则为 **false**。

#### 示例

假定库中的第一个项目为位图项目，如果之前导入的文件自导入以来在磁盘中没有发生修改，则下面的代码显示 **"true"**：

```
var libItem = fl.getDocumentDOM().library.items[0];  
fl.trace("fileIsCurrent = "+ libItem.sourceFileIsCurrent);
```

另请参见

[bitmapItem.fileLastModifiedDate](#)、[bitmapItem.sourceFilePath](#)

## bitmapItem.sourceFilePath

可用性

Flash CS4 Professional.

用法

```
bitmapItem.sourceFilePath
```

#### 描述

只读属性；一个字符串，形式为 **file:/// URI**，表示导入库中的文件的路径和名称。

#### 示例

下面的示例显示库中所有类型为 **"bitmap"** 的项目的名称和源文件路径：

```
for (idx in fl.getDocumentDOM().library.items) {  
  if (fl.getDocumentDOM().library.items[idx].itemType == "bitmap") {  
    var myItem = fl.getDocumentDOM().library.items[idx];  
    fl.trace(myItem.name + " source is " + myItem.sourceFilePath);  
  }  
}
```

另请参见

[bitmapItem.sourceFileExists](#)

## bitmapItem.useDeblocking

可用性

Flash CS4 Professional.

用法

```
bitmapItem.useDeblocking
```

#### 描述

属性；一个布尔值，它指定是 (**true**) 否 (**false**) 启用消除马赛克功能。

#### 示例

假定库中的第一个项目是位图项目，则下面的代码为此项目启用消除马赛克功能：

```
var libItem = fl.getDocumentDOM().library.items[0];  
libItem.useDeblocking = true;
```

## bitmapItem.useImportedJPEGQuality

#### 可用性

Flash MX 2004。

#### 用法

```
bitmapItem.useImportedJPEGQuality
```

#### 描述

属性；一个布尔值，它指定是 (**true**) 否 (**false**) 使用导入 JPEG 的默认品质。该属性仅对 JPEG 压缩可用。

#### 示例

下面的代码将当前文档的库中第一个项目的 **useImportedJPEGQuality** 属性设置为 **true**：

```
fl.getDocumentDOM().library.items[0].useImportedJPEGQuality = true;  
alert(fl.getDocumentDOM().library.items[0].useImportedJPEGQuality);
```

## 第 6 章 : CompiledClipInstance 对象

继承关系 [Element 对象](#) > [Instance 对象](#) > CompiledClipInstance 对象

可用性

Flash MX 2004。

描述

CompiledClipInstance 对象是 Instance 对象的子类。它实质上是一个被转换为经过编译的剪辑库项目的影片剪辑实例（请参阅 [Instance 对象](#)）。

属性摘要

除 [Instance 对象](#) 的属性外，CompiledClipInstance 对象还具有以下属性：

属性	描述
<a href="#">compiledClipInstance.accName</a>	一个字符串，它等效于“辅助功能”面板中的“名称”字段。
<a href="#">compiledClipInstance.actionScript</a>	一个字符串，它表示此实例的 <code>ActionScript</code> ；等效于 <a href="#">symbolInstance.actionScript</a> 。
<a href="#">compiledClipInstance.description</a>	一个字符串，它等效于“辅助功能”面板中的“描述”字段。
<a href="#">compiledClipInstance.forceSimple</a>	一个布尔值，它启用或禁止对子对象的访问。
<a href="#">compiledClipInstance.shortcut</a>	一个字符串，它等效于“辅助功能”面板中的“快捷键”字段。
<a href="#">compiledClipInstance.silent</a>	一个布尔值，允许或禁止访问对象；等效于“辅助功能”面板中的“使对象可访问”设置的反逻辑。
<a href="#">compiledClipInstance.tabIndex</a>	一个整数，等效于“辅助功能”面板中的“Tab 键索引”字段。

### compiledClipInstance.accName

可用性

Flash MX 2004。

用法

```
compiledClipInstance.accName
```

描述

属性；一个字符串，它等效于“辅助功能”面板中的“名称”字段。屏幕读取器通过大声读出该名称来标识对象。

示例

下面的示例获取并设置第一个所选对象的辅助功能名称：

```
// Get the name of the object.
var theName = fl.getDocumentDOM().selection[0].accName;
// Set the name of the object.
fl.getDocumentDOM().selection[0].accName = 'Home Button';
```

## compiledClipInstance.actionScript

可用性

Flash MX 2004。

用法

compiledClipInstance.actionScript

描述

属性；一个字符串，它表示此实例的 `ActionScript`；等效于 `symbolInstance.actionScript`。

示例

以下代码将 `ActionScript` 分配给指定的元素：

```
// Assign some ActionScript to a specified Button compiled clip instance.
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0]
    .actionScript = "on(click) {trace('button is clicked');}";
// Assign some ActionScript to the currently selected Button compiled clip instance.
fl.getDocumentDOM().selection[0].actionScript =
    "on(click) {trace('button is clicked');}";
```

## compiledClipInstance.description

可用性

Flash MX 2004。

用法

compiledClipInstance.description

描述

属性；一个字符串，它等效于“辅助功能”面板中的“描述”字段。该描述由屏幕读取器读出。

示例

下面的示例说明获取和设置 `description` 属性的方法：

```
// Get the description of the current selection.
var theDescription = fl.getDocumentDOM().selection[0].description;
// Set the description of the current selection.
fl.getDocumentDOM().selection[0].description =
    "This is compiled clip number 1";
```

## compiledClipInstance.forceSimple

可用性

Flash MX 2004。

用法

compiledClipInstance.forceSimple

#### 描述

属性；一个布尔值，它允许或禁止对子对象的访问。此属性等效于“辅助功能”面板中的“使子对象可访问”设置的反向逻辑。如果 `forceSimple` 为 `true`，则与不选中“使子对象可访问”选项的效果相同。如果 `forceSimple` 为 `false`，则与选中“使子对象可访问”选项的效果相同。

#### 示例

下面的示例说明获取和设置 `forceSimple` 属性的方法：

```
// Query if the children of the object are accessible.
var areChildrenAccessible = fl.getDocumentDOM().selection[0].forceSimple;
// Allow the children of the object to be accessible.
fl.getDocumentDOM().selection[0].forceSimple = false;
```

## compiledClipInstance.shortcut

#### 可用性

Flash MX 2004。

#### 用法

```
compiledClipInstance.shortcut
```

#### 描述

属性；一个字符串，它等效于“辅助功能”面板中的“快捷键”字段。此快捷键由屏幕读取器读出。此属性对于动态文本字段不可用。

#### 示例

下面的示例说明获取和设置 `shortcut` 属性的方法：

```
// Get the shortcut key of the object.
var theShortcut = fl.getDocumentDOM().selection[0].shortcut;
// Set the shortcut key of the object.
fl.getDocumentDOM().selection[0].shortcut = "Ctrl+I";
```

## compiledClipInstance.silent

#### 可用性

Flash MX 2004。

#### 用法

```
compiledClipInstance.silent
```

#### 描述

属性；一个布尔值，它允许或禁止访问对象；等效于“辅助功能”面板中的“使对象可访问”设置的反向逻辑。也就是说，如果 `silent` 为 `true`，则“使对象可访问”选项未选中。如果 `silent` 为 `false`，则“使对象可访问”选项被选中。

#### 示例

下面的示例说明获取和设置 `silent` 属性的方法：

```
// Query if the object is accessible.  
var isSilent = fl.getDocumentDOM().selection[0].silent;  
// Set the object to be accessible.  
fl.getDocumentDOM().selection[0].silent = false;
```

## compiledClipInstance.tabIndex

可用性

Flash MX 2004。

用法

`compiledClipInstance.tabIndex`

描述

属性；一个整数，它等效于“辅助功能”面板中的“Tab 键索引”字段。创建一个 Tab 键顺序，当用户按 Tab 键时，按此顺序访问对象。

示例

下面的示例说明获取和设置 `tabIndex` 属性的方法：

```
// Get the tabIndex of the object.  
var theTabIndex = fl.getDocumentDOM().selection[0].tabIndex;  
// Set the tabIndex of the object.  
fl.getDocumentDOM().selection[0].tabIndex = 1;
```



## 第 7 章 : compilerErrors 对象

可用性

Flash CS3 Professional。

描述

`compilerErrors` 对象表示“编译器错误”面板，它是 `flash` 对象 (`fl`) 的一个属性，可以通过 `fl.compilerErrors` 访问（请参阅 [flash 对象 \(fl\)](#)）。

方法摘要

以下方法可用于 `compilerErrors` 对象：

方法	描述
<a href="#">compilerErrors.clear()</a>	清除“编译器错误”面板的内容。
<a href="#">compilerErrors.save()</a>	将“编译器错误”面板中的内容保存到一个本地文本文件。

### compilerErrors.clear()

可用性

Flash CS3 Professional。

用法

```
compilerErrors.clear()
```

参数

无。

返回

无。

描述

方法；清除“编译器错误”面板的内容。

示例

下列清除“编译器错误”面板的内容：

```
fl.compilerErrors.clear();
```

另请参见

[compilerErrors.save\(\)](#)

## compilerErrors.save()

可用性

Flash CS3 Professional。

用法

```
compilerErrors.save(fileURI [, bAppendToFile [, bUseSystemEncoding]])
```

参数

**fileURI** 以 `file:/// URI` 形式表示的字符串，指定所保存文件的名称。如果 `fileURI` 已经存在，并且没有为 `bAppendToFile` 指定 `true` 值，则将覆盖 `fileURI` 并且不进行警告。

**bAppendToFile** 一个可选的布尔值，指定是应将“编译器错误”面板的内容追加到 `fileURI` (`true`) 还是不应追加 (`false`)。默认值为 `false`。

**bUseSystemEncoding** 一个可选的布尔值，指定是否使用系统编码来保存“编译器错误”面板中的文本。如果此值为 `false`（默认值），则使用 UTF-8 编码来保存“编译器错误”面板中的文本，并且文本的开头带字节顺序标记字符。默认值为 `false`。

返回

无。

描述

方法；将“编译器错误”面板中的内容保存到一个本地文本文件。

示例

下例将“编译器错误”面板的内容保存到 `C:\tests` 文件夹中名为 `errors.log` 的文件：

```
fl.compilerErrors.save("file:///c:/tests/errors.log");
```

另请参见

[compilerErrors.clear\(\)](#)

## 第 8 章 : ComponentInstance 对象

继承关系 [Element 对象](#) > [Instance 对象](#) > [SymbolInstance 对象](#) > ComponentInstance 对象

可用性

Flash MX 2004。

描述

ComponentInstance 对象是 SymbolInstance 对象的子类，表示帧中的一个组件（请参阅 [SymbolInstance 对象](#)）。

属性摘要

除 [SymbolInstance 对象](#) 的所有属性外，ComponentInstance 对象还具有以下属性：

属性	描述
<a href="#">componentInstance.parameters</a>	只读；一个 ActionScript 2.0 属性的数组，可从组件“属性”检查器中访问。

### componentInstance.parameters

可用性

Flash MX 2004。

用法

```
componentInstance.parameters
```

描述

只读属性；一个 ActionScript 2.0 属性的数组，可从组件“属性”检查器中访问。请参阅 [Parameter 对象](#)。

示例

下面的示例说明获取和设置 parameters 属性的方法：

```
var parms = fl.getDocumentDOM().selection[0].parameters;
parms[0].value = "some value";
```

另请参见

[Parameter 对象](#)

## 第 9 章 : componentsPanel 对象

可用性

Flash MX 2004。

描述

`componentsPanel` 对象表示“组件”面板，它是 `flash` 对象 (`fl`) 的一个属性，可以通过 `fl.componentsPanel` 访问（请参阅 [flash 对象 \(fl\)](#)）。

方法摘要

您可以对 `componentsPanel` 对象使用以下方法：

方法	描述
<a href="#">componentsPanel.addItemToDocument()</a>	将指定的组件添加到文档中指定的位置。
<a href="#">componentsPanel.reload()</a>	刷新“组件”面板的组件列表。

### componentsPanel.addItemToDocument()

可用性

Flash MX 2004。

用法

```
componentsPanel.addItemToDocument(position, categoryName, componentName)
```

参数

**position** 一个点（例如，`{x:0, y:100}`），它指定向文档中添加组件的位置。指定相对于组件中心点而不是组件注册点（即原点或零点）的位置。

**categoryName** 一个字符串，它指定组件类别的名称（例如，“Data”）。“组件”面板中列出了有效的类别名称。

**componentName** 一个字符串，它指示指定类别中的组件的名称（例如，“WebServiceConnector”）。“组件”面板中列出了有效的组件名称。

返回

无。

描述

将指定的组件添加到文档中指定的位置。

示例

下面的示例阐释该方法的某些用法：

```
fl.componentsPanel.addItemToDocument({x:0, y:0}, "User Interface", "CheckBox");
fl.componentsPanel.addItemToDocument({x:0, y:100}, "Data", "WebServiceConnector");
fl.componentsPanel.addItemToDocument({x:0, y:200}, "User Interface", "Button");
```

## componentsPanel.reload()

可用性

Flash 8。

用法

```
componentsPanel.reload()
```

参数

无。

返回

布尔值；如果刷新“组件”面板列表，则为 **true**；否则为 **false**。

说明

方法；刷新“组件”面板的组件列表。

示例

下面的示例可刷新“组件”面板：

```
f1.componentsPanel.reload();
```

## 第 10 章 : Contour 对象

可用性

Flash MX 2004。

描述

**Contour** 对象表示由形状边界上的半边缘组成的封闭路径。

方法摘要

您可以对 **Contour** 对象使用以下方法：

方法	描述
<a href="#">contour.getHalfEdge()</a>	返回所选形状的轮廓上的一个 <a href="#">HalfEdge</a> 对象。

属性摘要

您可以对 **Contour** 对象使用以下属性：

属性	描述
<a href="#">contour.fill</a>	一个 <a href="#">Fill</a> 对象。
<a href="#">contour.interior</a>	只读；如果轮廓围合了一个区域，则值为 <b>true</b> ；否则为 <b>false</b> 。
<a href="#">contour.orientation</a>	只读；整数，用于指示轮廓的方向。

### contour.fill

可用性

Flash CS4 Professional.

用法

`contour.fill`

说明

属性；一个 [Fill](#) 对象。

示例

假定某个轮廓具有选定的填充，则下面的示例在“输出”面板中显示该轮廓的填充颜色：

```
var insideContour = fl.getDocumentDOM().selection[0].contours[1];
var insideFill = insideContour.fill;
fl.trace(insideFill.color);
```

## contour.getHalfEdge()

可用性

Flash MX 2004。

用法

contour.getHalfEdge()

参数

无。

返回

一个 [HalfEdge](#) 对象。

说明

方法；返回所选形状的轮廓上的一个 [HalfEdge](#) 对象。

示例

此示例遍历所选形状的所有轮廓，并在“输出”面板中显示顶点的坐标：

```
// with a shape selected

var elt = fl.getDocumentDOM().selection[0];
elt.beginEdit();

var contourArray = elt.contours;
var contourCount = 0;
for (i=0;i<contourArray.length;i++)
{
    var contour = contourArray[i];
    contourCount++;
    var he = contour.getHalfEdge();

    var iStart = he.id;
    var id = 0;
    while (id != iStart)
    {
        // Get the next vertex.
        var vrt = he.getVertex();

        var x = vrt.x;
        var y = vrt.y;
        fl.trace("vrt: " + x + ", " + y);

        he = he.getNext();
        id = he.id;
    }
}
elt.endEdit();
```

## contour.interior

可用性

Flash MX 2004。

用法

contour.interior

说明

只读属性；如果轮廓封闭了一个区域，则值为 **true**；否则为 **false**。

示例

此示例遍历所选形状的所有轮廓，并在“输出”面板中显示每个轮廓的 **interior** 属性值：

```
var elt = fl.getDocumentDOM().selection[0];
elt.beginEdit();

var contourArray = elt.contours;

var contourCount = 0;
for (i=0;i<contourArray.length;i++) {
    var contour = contourArray[i];
    fl.trace("Next Contour, interior:" + contour.interior );
    contourCount++;
}
elt.endEdit();
```

## contour.orientation

可用性

Flash MX 2004。

用法

contour.orientation

说明

只读属性；一个整数，用于指示轮廓的方向。如果方向为逆时针，则该整数的值为 **-1**；如果为顺时针，则为 **1**；如果轮廓不包含区域，则为 **0**。

示例

下面的示例遍历所选形状的所有轮廓，并在“输出”面板中显示每个轮廓的 **orientation** 属性值：



**Contour 对象**

```
var elt = fl.getDocumentDOM().selection[0];
elt.beginEdit();

var contourArray = elt.contours;

var contourCount = 0;
for (i=0;i<contourArray.length;i++) {
    var contour = contourArray[i];
    fl.trace("Next Contour, orientation:" + contour.orientation);
    contourCount++;
}
elt.endEdit();
```

# 第 11 章 : Document 对象

可用性

Flash MX 2004。

描述

**Document** 对象表示舞台。也就是说，只将 **FLA** 文件视为文档。若要返回当前文档的 **Document** 对象，请使用 [fl.getDocumentDOM\(\)](#)。

方法摘要

可以对 **Document** 对象使用以下方法：

方法	描述
<a href="#">document.addDataToDocument()</a>	随文档存储指定的数据。
<a href="#">document.addDataToSelection()</a>	随所选对象存储指定的数据。
<a href="#">document.addFilter()</a>	向所选对象应用滤镜。
<a href="#">document.addItem()</a>	将任何打开的文档或库中的项目添加到指定的 <b>Document</b> 对象。
<a href="#">document.addNewLine()</a>	在两点之间添加一个新路径。
<a href="#">document.addNewOval()</a>	在指定的边界矩形中添加一个新 <b>Oval</b> 对象。
<a href="#">document.addNewPrimitiveOval()</a>	添加可纳入指定范围的新基本椭圆。
<a href="#">document.addNewPrimitiveRectangle()</a>	添加可纳入指定范围的新基本矩形。
<a href="#">document.addNewPublishProfile()</a>	添加新的发布配置文件，并使其成为当前发布配置文件。
<a href="#">document.addNewRectangle()</a>	添加新的矩形或圆角矩形，使其纳入指定范围。
<a href="#">document.addNewScene()</a>	添加新场景 ( <b>Timeline</b> 对象) 作为当前所选场景之后的下一个场景，并使新场景成为当前所选场景。
<a href="#">document.addNewText()</a>	插入新的空文本字段。
<a href="#">document.align()</a>	对齐所选内容。
<a href="#">document.allowScreens()</a>	此方法在使用第 124 页的 <code>document.screenOutline</code> 属性之前使用。
<a href="#">document.arrange()</a>	排列舞台上的所选内容。
<a href="#">document.breakApart()</a>	对当前所选内容执行分离操作。
<a href="#">document.canEditSymbol()</a>	指示是否已启用“编辑元件”菜单和功能。
<a href="#">document.canRevert()</a>	确定是否可以成功使用 <a href="#">document.revert()</a> 或 <a href="#">fl.revertDocument()</a> 方法。
<a href="#">document.canTestMovie()</a>	确定是否可以成功使用 <a href="#">document.testMovie()</a> 方法。
<a href="#">document.canTestScene()</a>	确定是否可以成功使用 <a href="#">document.testScene()</a> 方法。
<a href="#">document.changeFilterOrder()</a>	更改滤镜在“滤镜”列表中的索引。
<a href="#">document.clipCopy()</a>	将当前所选内容从文档复制到剪贴板。

方法	描述
<code>document.clipCut()</code>	从文档中剪切当前所选内容，并将其写入剪贴板。
<code>document.clipPaste()</code>	将剪贴板的内容粘贴到文档中。
<code>document.close()</code>	关闭指定的文档。
<code>document.convertLinesToFills()</code>	将所选对象上的线条转换为填充。
<code>document.convertToSymbol()</code>	将所选舞台项目转换为新元件。
<code>document.crop()</code>	使用所选最顶层绘制对象裁剪其下所有选择的绘制对象。
<code>document.debugMovie()</code>	启动与该文档的调试会话。
<code>document.deleteEnvelope()</code>	从所选对象中删除封套（包含一个或多个对象的边框）。
<code>document.deletePublishProfile()</code>	如果存在多个配置文件，则删除当前活动配置文件。
<code>document.deleteScene()</code>	删除当前场景（ <b>Timeline</b> 对象）；如果删除的场景不是最后一个场景，则将下一个场景设置为当前 <b>Timeline</b> 对象。
<code>document.deleteSelection()</code>	删除舞台上的当前所选内容。
<code>document.disableAllFilters()</code>	禁用所选对象上的所有滤镜。
<code>document.disableFilter()</code>	禁用“滤镜”列表中指定的滤镜。
<code>document.disableOtherFilters()</code>	禁用所有滤镜，“滤镜”列表中指定位置处的滤镜除外。
<code>document.distribute()</code>	分散所选内容。
<code>document.distributeToLayers()</code>	对当前所选内容执行分散到图层操作，此操作等效于选择“分散到图层”。
<code>document.documentHasData()</code>	检查文档中是否存在具有指定名称的永久数据。
<code>document.duplicatePublishProfile()</code>	直接复制当前活动的配置文件，并提供直接复制版本焦点。
<code>document.duplicateScene()</code>	制作当前所选场景的副本，为新场景提供一个唯一的名称，并使其成为当前场景。
<code>document.duplicateSelection()</code>	直接复制舞台上的所选内容。
<code>document.editScene()</code>	使指定的场景成为当前所选场景，以供编辑。
<code>document.enableAllFilters()</code>	为所选对象启用“滤镜”列表上的所有滤镜。
<code>document.enableFilter()</code>	为所选对象启用指定的滤镜。
<code>document.enterEditMode()</code>	将创作工具切换到由参数指定的编辑模式。
<code>document.exitEditMode()</code>	从元件编辑模式中退出，将焦点由编辑模式返回到上一级。
<code>document.exportPNG()</code>	将文档导出为一个或多个 PNG 文件。
<code>document.exportPublishProfile()</code>	将当前活动配置文件导出到 XML 文件。
<code>document.exportPublishProfileString()</code>	返回一个字符串，它以 XML 格式指明所指定的配置文件。
<code>document.exportSWF()</code>	以 Flash SWF 格式导出文档。
<code>document.getAlignToDocument()</code>	等效于检索“对齐”面板中“相对于舞台”按钮的值。
<code>document.getBlendMode()</code>	返回一个字符串，它为所选对象指定混合模式。

方法	描述
<a href="#">document.getCustomFill()</a>	检索所选形状的填充对象，或者检索“工具”面板和“属性”检查器的填充对象（如果已指定）。
<a href="#">document.getCustomStroke()</a>	返回所选形状的笔触对象，或者返回“工具”面板和“属性”检查器的笔触对象（如果已指定）。
<a href="#">document.getDataFromDocument()</a>	检索指定数据的值。
<a href="#">document.getElementProperty()</a>	获取当前所选内容的指定 Element 属性。
<a href="#">document.getElementTextAttr()</a>	获取所选 Text 对象的指定 TextAttrs 属性。
<a href="#">document.getFilters()</a>	返回一个数组，它包含应用于当前所选对象的滤镜的列表。
<a href="#">document.getMetadata()</a>	返回一个字符串，它包含与文档关联的 XML 元数据。
<a href="#">document.getMobileSettings()</a>	返回传递给 <a href="#">document.setMobileSettings()</a> 的字符串。
<a href="#">document.getPlayerVersion()</a>	返回一个字符串，它表示指定文档的目标播放器版本。
<a href="#">document.getSelectionRect()</a>	获取当前所选内容的边界矩形。
<a href="#">document.getTextString()</a>	获取当前所选文本。
<a href="#">document.getTimeline()</a>	检索文档中的当前 Timeline 对象。
<a href="#">document.getTransformationPoint()</a>	获取当前所选内容的变形点的位置。
<a href="#">document.group()</a>	将当前所选内容转换为组。
<a href="#">document.importFile()</a>	向文档导入文件。
<a href="#">document.importPublishProfile()</a>	从文件中导入配置文件。
<a href="#">document.importPublishProfileString()</a>	导入一个表示发布配置文件的 XML 字符串，并将其设置为当前配置文件。
<a href="#">document.importSWF()</a>	向文档导入 SWF 文件。
<a href="#">document.intersect()</a>	根据所有选择的绘制对象创建交集绘制对象。
<a href="#">document.loadCuepointXML()</a>	加载提示点 XML 文件。
<a href="#">document.match()</a>	使所选对象的大小相同。
<a href="#">document.mouseClick()</a>	从“选择”工具中执行鼠标单击。
<a href="#">document.mouseDbkClk()</a>	从“选择”工具中执行鼠标双击。
<a href="#">document.moveSelectedBezierPointsBy()</a>	如果所选内容至少包含一个路径，且至少选择了一个贝塞尔曲线点，此方法将按照指定的量移动所有选中路径上的所有选中贝塞尔曲线点。
<a href="#">document.moveSelectionBy()</a>	按照指定的距离移动所选对象。
<a href="#">document.optimizeCurves()</a>	优化当前所选内容的平滑，允许多重过渡（如果已指定）以实现最佳平滑；等效于选择“修改”>“形状”>“优化”。
<a href="#">document.publish()</a>	按照活动的发布设置（“文件”>“发布设置”）发布文档；等效于选择“文件”>“发布”。
<a href="#">document.punch()</a>	使用最上层的所选绘制对象来为其下的所有选中绘制对象打孔。
<a href="#">document.removeAllFilters()</a>	从所选对象中删除所有滤镜。
<a href="#">document.removeDataFromDocument()</a>	删除附着到文档的具有指定名称的永久数据。

方法	描述
<code>document.removeDataFromSelection()</code>	删除附着到所选内容的具有指定名称的永久数据。
<code>document.removeFilter()</code>	从所选对象的“滤镜”列表中删除指定的滤镜。
<code>document.renamePublishProfile()</code>	重命名当前配置文件。
<code>document.renameScene()</code>	在“场景”面板中重命名当前所选场景。
<code>document.reorderScene()</code>	将指定的场景移动到另一个指定场景之前。
<code>document.resetOvalObject()</code>	将“属性”检查器中的所有值设置为默认的 <b>Oval</b> 对象设置。
<code>document.resetRectangleObject()</code>	将“属性”检查器中的所有值设置为默认的 <b>Rectangle</b> 对象设置。
<code>document.resetTransformation()</code>	重置变形矩阵；等效于选择“修改”>“变形”>“删除变形”。
<code>document.revert()</code>	将指定文档还原到以前保存的版本；等效于选择“文件”>“还原”。
<code>document.rotate3DSelection()</code>	对所选内容应用 3D 旋转。
<code>document.rotateSelection()</code>	按照指定的度数旋转所选内容。
<code>document.save()</code>	将文档保存在其默认位置；等效于选择“文件”>“保存”。
<code>document.saveAndCompact()</code>	保存并压缩文件；等效于选择“文件”>“保存并压缩”。
<code>document.scaleSelection()</code>	按照指定的量缩放所选内容；等效于使用“任意变形”工具缩放对象。
<code>document.selectAll()</code>	选择舞台上的所有项目；等效于按 <b>Control+A (Windows)</b> 或 <b>Command+A (Macintosh)</b> ，也等效于选择“编辑”>“全选”。
<code>document.selectNone()</code>	取消选择所有选中的项目。
<code>document.setAlignToDocument()</code>	将 <code>document.align()</code> 、 <code>document.distribute()</code> 、 <code>document.match()</code> 和 <code>document.space()</code> 的首选参数设置为针对文档进行操作；等效于在“对齐”面板中启用“相对于舞台”按钮。
<code>document.setBlendMode()</code>	设置所选对象的混合模式。
<code>document.setCustomFill()</code>	设置“工具”面板、“属性”检查器和全部所选形状的填充设置。
<code>document.setCustomStroke()</code>	设置“工具”面板、“属性”检查器和全部所选形状的笔触设置。
<code>document.setElementProperty()</code>	设置文档中所选对象的指定 <b>Element</b> 属性。
<code>document.setElementTextAttr()</code>	将所选文本项目的指定 <b>TextAttrs</b> 属性设置为指定值。
<code>document.setFillColor()</code>	将所选内容的填充颜色更改为指定的颜色。
<code>document.setFilterProperty()</code>	设置当前所选对象的指定滤镜属性。
<code>document.setFilters()</code>	将滤镜应用于所选对象。
<code>document.setInstanceAlpha()</code>	设置实例的不透明度。
<code>document.setInstanceBrightness()</code>	设置实例的亮度。
<code>document.setInstanceTint()</code>	设置实例的色调。
<code>document.setMetadata()</code>	设置指定文档的 XML 元数据，覆盖全部现有元数据。
<code>document.setMobileSettings()</code>	设置移动 FLA 文件中 XML 设置字符串的值。
<code>document.setOvalObjectProperty()</code>	为 <b>Oval</b> 图元对象的指定属性指定值。

方法	描述
<code>document.setPlayerVersion()</code>	设置指定文档的目标 Flash Player 版本。
<code>document.setRectangleObjectProperty()</code>	为 <code>Rectangle</code> 图元对象的指定属性指定值。
<code>document.setSelectionBounds()</code>	在一次操作中移动所选内容并调整其大小。
<code>document.setSelectionRect()</code>	使用指定坐标绘制相对于舞台的矩形选取框。
<code>document.setStageVanishingPoint()</code>	指定用于查看 3D 对象的消失点。
<code>document.setStageViewAngle()</code>	指定用于查看 3D 对象的透视角度。
<code>document.setStroke()</code>	设置所选笔触的颜色、宽度和样式。
<code>document.setStrokeColor()</code>	将所选内容的笔触颜色更改为指定的颜色。
<code>document.setStrokeSize()</code>	将所选内容的笔触大小更改为指定的大小。
<code>document.setStrokeStyle()</code>	将所选内容的笔触样式更改为指定的样式。
<code>document.setTextRectangle()</code>	将所选文本项目的边框矩形更改为指定的大小。
<code>document.setTextSelection()</code>	将当前所选文本字段中的所选文本设置为由 <code>startIndex</code> 和 <code>endIndex</code> 值指定的值。
<code>document.setTextString()</code>	插入文本字符串。
<code>document.setTransformationPoint()</code>	移动当前所选内容的变形点。
<code>document.skewSelection()</code>	按照指定的量倾斜所选内容。
<code>document.smoothSelection()</code>	平滑所选的每条填充轮廓或弯曲线的曲线。
<code>document.space()</code>	均匀间隔所选内容中的对象。
<code>document.straightenSelection()</code>	伸直当前所选笔触；等效于使用“工具”面板中的“伸直”按钮。
<code>document.swapElement()</code>	用指定的选择内容交换当前选择内容。
<code>document.swapStrokeAndFill()</code>	交换笔触颜色和填充颜色。
<code>document.testMovie()</code>	对文档执行“测试影片”操作。
<code>document.testScene()</code>	对文档的当前场景执行“测试场景”操作。
<code>document.traceBitmap()</code>	对当前所选内容执行转换位图为矢量图；等效于选择“修改”>“位图”>“转换位图为矢量图”。
<code>document.transformSelection()</code>	通过应用参数中指定的矩阵，对当前所选内容执行常规变形。
<code>document.translate3DCenter()</code>	设置 XYZ 位置，所选内容围绕此位置进行平移或旋转。
<code>document.translate3DSelection()</code>	对所选内容应用 3D 平移。
<code>document.unGroup()</code>	取消组合当前所选内容。
<code>document.union()</code>	将所有选中的形状合并到绘制对象中。
<code>document.unlockAllElements()</code>	解除锁定当前所选帧上的全部锁定元素。
<code>document.xmlPanel()</code>	张贴 XMLUI 对话框。

#### 属性摘要

可以对 `Document` 对象使用以下属性。

属性	描述
document.accName	一个字符串，它等效于“辅助功能”面板中的“名称”字段。
document.as3AutoDeclare	一个布尔值，它说明是否将放置在舞台上的实例自动添加到用户定义的时间轴类中。
document.as3Dialect	一个字符串，它说明指定文档中正在使用的 ActionScript 3.0“术语”。
document.as3ExportFrame	一个整数，它指定在哪一帧导出 ActionScript 3.0 类。
document.as3StrictMode	一个布尔值，它指定 ActionScript 3.0 编译器在进行编译时是应打开还是应关闭“严格模式”选项。
document.as3WarningsMode	一个布尔值，它指定 ActionScript 3.0 编译器在进行编译时是应打开还是应关闭“警告模式”选项。
document.asVersion	一个整数，它指明所指定文件中正在使用的 ActionScript 版本。
document.autoLabel	一个布尔值，它等效于“辅助功能”面板中的“自动标签”复选框。
document.backgroundColor	一个字符串、十六进制值或整数，它表示背景的颜色。
document.currentPublishProfile	一个字符串，它为所指定文档的活动发布配置文件指定名称。
document.currentTimeline	一个整数，它指定活动时间轴的索引。
document.description	一个字符串，它等效于“辅助功能”面板中的“描述”字段。
document.docClass	指定与文档相关联的顶级 ActionScript 3.0 类。
document.externalLibraryPath	一个字符串，其中包含文档 ActionScript 3.0 外部库路径中项目的列表，该列表指定用作运行时共享库的 SWC 文件的位置。
document.forceSimple	一个布尔值，它指定是否可以访问指定对象的子项。
document.frameRate	一个浮点值，它指定播放 SWF 文件时每秒显示的帧数；默认值为 12。
document.height	一个整数，它以像素为单位指定文档（舞台）的高度。
document.id	一个（自动分配的）唯一整数，用于在 Flash 会话期间标识文档。
document.library	只读；文档的 library 对象。
document.libraryPath	一个字符串，其中包含文档 ActionScript 3.0 库路径中项目的列表，该列表指定 SWC 文件或包含 SWC 文件的文件夹的位置。
document.livePreview	一个布尔值，它指定是否启用“实时预览”。
document.name	只读；一个字符串，它表示文档（FLA 文件）的名称。
document.path	只读；一个字符串，它以平台专用格式表示文档的路径。
document.pathURI	只读；一个字符串，它以 file:/// URI 的形式表示文档的路径。
document.publishProfiles	只读；文档发布配置文件名称的数组。
document.screenOutline	只读；文档的当前 ScreenOutline 对象。从 Flash Professional CS5 开始，已弃用 ScreenOutline 对象。
document.selection	文档中所选对象的数组。
document.silent	一个布尔值，它指定对象是否可访问。
document.sourcePath	一个字符串，其中包含文档 ActionScript 3.0 源路径中项目的列表，该列表指定 ActionScript 类文件的位置。

属性	描述
<a href="#">document.timelines</a>	只读；Timeline 对象的数组（请参阅 <a href="#">Timeline 对象</a> ）。
<a href="#">document.viewMatrix</a>	只读；一个 <a href="#">Matrix 对象</a> 。
<a href="#">document.width</a>	一个整数，它以像素为单位指定文档（舞台）的宽度。
<a href="#">document.zoomFactor</a>	指定创作时舞台的缩放百分比。

## document.accName

可用性

Flash MX 2004。

用法

```
document.accName
```

描述

属性；一个字符串，它等效于“辅助功能”面板中的“名称”字段。屏幕读取器通过大声读出该名称来标识对象。

示例

下面的示例将文档的辅助功能名称设置为“Main Movie”（主影片）：

```
fl.getDocumentDOM().accName = "Main Movie";
```

下面的示例获取文档的辅助功能名称：

```
fl.trace(fl.getDocumentDOM().accName);
```

## document.addDataToDocument()

可用性

Flash MX 2004。

用法

```
document.addDataToDocument(name, type, data)
```

参数

**name** 一个字符串，它指定要添加的数据的名称。

**type** 一个字符串，它定义要添加的数据的类型。可接受值为“integer”、“integerArray”、“double”、“doubleArray”、“string”和“byteArray”。

**data** 要添加的值。有效类型取决于 type 参数。

返回

无。

描述

方法；将指定数据存储存储在文档中。将数据写入 FLA 文件，该文件重新打开时，JavaScript 即可使用这些数据。



## 示例

下面的示例将整数值 12 添加到当前文档：

```
fl.getDocumentDOM().addDataToDocument("myData", "integer", 12);
```

下面的示例返回名为 "myData" 的数据的值，并在“输出”面板中显示结果：

```
fl.trace(fl.getDocumentDOM().getDataFromDocument("myData"));
```

另请参见

[document.getDataFromDocument\(\)](#)、[document.removeDataFromDocument\(\)](#)

## document.addDataToSelection()

## 可用性

Flash MX 2004。

## 用法

```
document.addDataToSelection(name, type, data)
```

## 参数

**name** 一个字符串，它指定永久性数据的名称。

**type** 定义数据的类型。可接受值为 "integer"、"integerArray"、"double"、"doubleArray"、"string" 和 "byteArray"。

**data** 要添加的值。有效类型取决于 type 参数。

## 返回

无。

## 描述

方法；将指定的数据存储在所选对象中。将数据写入 FLA 文件，该文件重新打开时，JavaScript 即可使用这些数据。只有元件和位图才支持永久数据。

## 示例

下面的示例将整数值 12 添加到所选对象：

```
fl.getDocumentDOM().addDataToSelection("myData", "integer", 12);
```

另请参见

[document.removeDataFromSelection\(\)](#)

## document.addFilter()

## 可用性

Flash 8。

## 用法

```
document.addFilter(filterName)
```

## Document 对象

## 参数

**filterName** 一个字符串，它指定要添加到“滤镜”列表并为所选对象启用的滤镜。可接受值为 "adjustColorFilter"、"bevelFilter"、"blurFilter"、"dropShadowFilter"、"glowFilter"、"gradientBevelFilter" 和 "gradientGlowFilter"。

## 返回

无。

## 描述

方法；将滤镜应用于所选对象，并将该滤镜放到“滤镜”列表的末尾。

## 示例

下面的示例将一个发亮滤镜应用于所选对象。

```
fl.getDocumentDOM().addFilter("glowFilter");
```

## 另请参见

[document.changeFilterOrder\(\)](#)、[document.disableFilter\(\)](#)、[document.enableFilter\(\)](#)、[document.getFilters\(\)](#)、[document.removeFilter\(\)](#)、[document.setBlendMode\(\)](#)、[document.setFilterProperty\(\)](#)

## document.addItem()

## 可用性

Flash MX 2004。

## 用法

```
document.addItem(position, item)
```

## 参数

**position** 一个点，它指定项目添加位置的 **x** 和 **y** 坐标。它使用元件的中心，或者使用位图或视频的左上角。

**item** 一个 **Item** 对象，它指定要添加的项目以及项目所在的库（请参阅 [Item 对象](#)）。

## 返回

一个布尔值：如果成功，则为 **true**；否则为 **false**。

## 说明

方法；从任何打开的文档或库将一个项目添加到指定的文档对象。

## 示例

下面的示例将库中第一个项目添加到第一个文档中所选元件、位图或视频的指定位置：

```
var item = fl.documents[0].library.items[0];  
fl.documents[0].addItem({x:0,y:0}, item);
```

下面的示例将元件 **myMovieClip** 从当前文档的库添加到当前文档：

```
var itemIndex = fl.getDocumentDOM().library.findItemIndex("myMovieClip");  
var theItem = fl.getDocumentDOM().library.items[itemIndex];  
fl.getDocumentDOM().addItem({x:0,y:0}, theItem);
```

下面的示例将元件 **myMovieClip** 从文档数组的第二个文档添加到文档数组的第三个文档：

```
var itemIndex = fl.documents[1].library.findItemIndex("myMovieClip");
var theItem = fl.documents[1].library.items[itemIndex];
fl.documents[2].addItem({x:0,y:0}, theItem);
```

## document.addNewLine()

可用性

Flash MX 2004。

用法

```
document.addNewLine(startPoint, endpoint)
```

参数

**startpoint** 一对浮点数，它们指定线条起点的 **x** 和 **y** 坐标。

**endpoint** 一对浮点数，它们指定线条终点的 **x** 和 **y** 坐标。

返回

无。

描述

方法；在两点之间添加一个新路径。此方法使用文档的当前笔触属性，在当前帧和当前图层上添加路径。此方法的工作方式与单击线条工具然后绘制线条相同。

示例

下面的示例在指定的起点和终点之间添加一个线条：

```
fl.getDocumentDOM().addNewLine({x:216.7, y:122.3}, {x:366.8, y:165.8});
```

## document.addNewOval()

可用性

Flash MX 2004。

用法

```
document.addNewOval(boundingRectangle [, bSuppressFill [, bSuppressStroke ]])
```

参数

**boundingRectangle** 一个矩形，它指定要添加的椭圆的范围。有关 **boundingRectangle** 的格式信息，请参阅 [document.addNewRectangle\(\)](#)。

**bSuppressFill** 一个布尔值，如果设置为 **true**，则此方法创建没有填充的形状。默认值为 **false**。此参数是可选的。

**bSuppressStroke** 一个布尔值，如果设置为 **true**，则此方法创建没有笔触的形状。默认值为 **false**。此参数是可选的。

返回

无。

#### 描述

方法；在指定的边界矩形中添加一个新 **Oval** 对象。此方法与“椭圆”工具执行的操作相同。此方法使用文档当前默认的笔触和填充的属性，在当前的帧和图层上添加椭圆。如果 **bSuppressFill** 和 **bSuppressStroke** 都设置为 **true**，则此方法不起作用。

#### 示例

下面的示例在指定的坐标内添加一个新椭圆；它的宽为 164 像素，高为 178 像素：

```
fl.getDocumentDOM().addNewOval({left:72,top:50,right:236,bottom:228});
```

下面的示例绘制一个没有填充的椭圆：

```
fl.getDocumentDOM().addNewOval({left:72,top:50,right:236,bottom:228}, true);
```

下面的示例绘制一个没有笔触的椭圆：

```
fl.getDocumentDOM().addNewOval({left:72,top:50,right:236,bottom:228}, false, true);
```

另请参见

[document.addNewPrimitiveOval\(\)](#)

## document.addNewPrimitiveOval()

#### 可用性

Flash CS4 Professional.

#### 用法

```
document.addNewPrimitiveOval( boundingRectangle [, bSpupressFill [, bSuppressStroke ] ] )
```

#### 参数

**boundingRectangle** 一个矩形，它指定要在其中添加新基本椭圆的范围。有关 **boundingRectangle** 的格式的信息，请参阅 [document.addNewRectangle\(\)](#)。

**bSuppressFill** 一个布尔值，如果设置为 **true**，则此方法创建没有填充的椭圆。默认值为 **false**。此参数是可选的。

**bSuppressStroke** 一个布尔值，如果设置为 **true**，则此方法创建没有笔触的椭圆。默认值为 **false**。此参数是可选的。

#### 返回

无。

#### 描述

方法；添加一个适应指定范围的新基本椭圆。此方法与“基本椭圆”工具执行的操作相同。基本椭圆使用文档当前默认的笔触和填充的属性，并添加在当前的帧和图层上。如果 **bSuppressFill** 和 **bSuppressStroke** 都设置为 **true**，则此方法不起作用。

#### 示例

下面的示例在指定的坐标内添加有填充和笔触以及没有填充和笔触的基本椭圆：

```
// Add an oval primitive with fill and stroke
fl.getDocumentDOM().addNewPrimitiveOval({left:0,top:0,right:100,bottom:100});
// Add an oval primitive without a fill
fl.getDocumentDOM().addNewPrimitiveOval({left:100,top:100,right:200,bottom:200}, true);
// Add an oval primitive without a stroke
fl.getDocumentDOM().addNewPrimitiveOval({left:200,top:200,right:300,bottom:300},false,true);
```

另请参见  
[document.addNewOval\(\)](#)

## document.addNewPrimitiveRectangle()

可用性  
Flash CS4 Professional.

用法  
`document.addNewPrimitiveRectangle( boundingRectangle, roundness, [, bSuppressFill [, bSuppressStroke ] ] )`

参数  
**rect** 一个矩形，它指定要在其中添加新基本矩形的范围。有关 **boundingRectangle** 的格式的信息，请参阅 [document.addNewRectangle\(\)](#)。

**roundness** 一个介于 0 和 999 之间的整数，它表示用于指定圆角圆度的点数。

**bSuppressFill** 一个布尔值，如果设置为 **true**，则此方法创建没有填充的矩形。默认值为 **false**。此参数是可选的。

**bSuppressStroke** 一个布尔值，如果设置为 **true**，则此方法创建没有笔触的矩形。默认值为 **false**。此参数是可选的。

返回  
无。

描述  
方法；添加一个适应指定范围的新基本矩形。此方法与“基本矩形”工具执行的操作相同。基本矩形使用文档当前默认的笔触和填充属性，并添加在当前的帧和图层上。如果 **bSuppressFill** 和 **bSuppressStroke** 都设置为 **true**，则此方法不起作用。

示例  
下面的示例在指定的坐标内添加具有不同圆度、有填充和笔触以及没有填充和笔触的基本矩形：

```
// Add a rectangle primitive with fill and stroke
fl.getDocumentDOM().addNewPrimitiveRectangle({left:0,top:0,right:100,bottom:100}, 0);
// Add a rectangle primitive without a fill
fl.getDocumentDOM().addNewPrimitiveRectangle({left:100,top:100,right:200,bottom:200}, 20, true);
// Add a rectangle primitive without a stroke
fl.getDocumentDOM().addNewPrimitiveRectangle({left:200,top:200,right:300,bottom:300}, 50, false, true);
```

另请参见  
[document.addNewRectangle\(\)](#)

## document.addNewPublishProfile()

可用性  
Flash MX 2004.

用法  
`document.addNewPublishProfile( [profileName] )`

#### 参数

**profileName** 新配置文件的唯一名称。如果不指定名称，则提供一个默认名称。此参数是可选的。

#### 返回

一个整数，它是新配置文件在配置文件列表中的索引。如果无法创建新配置文件，则返回 -1。

#### 说明

方法；添加新的发布配置文件，并用作当前发布配置文件。

#### 示例

下面的示例使用默认名称添加一个新的发布配置文件，然后在“输出”面板中显示该配置文件的名称：

```
fl.getDocumentDOM().addNewPublishProfile();  
fl.outputPanel.trace(fl.getDocumentDOM().currentPublishProfile);
```

下面的示例添加一个名为 "my profile" 的新发布配置文件：

```
fl.getDocumentDOM().addNewPublishProfile("my profile");
```

另请参见

[document.deletePublishProfile\(\)](#)

## document.addNewRectangle()

#### 可用性

Flash MX 2004。

#### 用法

```
document.addNewRectangle(boundingRectangle, roundness  
    [, bSuppressFill [, bSuppressStroke]])
```

#### 参数

**boundingRectangle** 一个矩形，它指定要在其中添加新矩形的范围，其格式为 {left:value1,top:value2,right:value3,bottom:value4}。left 和 top 值指定左上角的位置（如 left:0,top:0 表示舞台的左上角），right 和 bottom 值指定右下角的位置。因此，矩形的宽度为 left 值和 right 值的差；矩形的高度为 top 值和 bottom 值的差。

也就是说，矩形的范围并非全部对应于“属性”检查器中显示的值。left 和 top 值分别对应于“属性”检查器中的 X 值和 Y 值。但是，right 和 bottom 值并不对应于“属性”检查器中的 W 和 H 值。例如，假设有一个矩形具有以下范围：

```
{left:10,top:10,right:50,bottom:100}
```

该矩形将在“属性”检查器中显示以下值：

```
X = 10, Y = 10, W = 40, H = 90
```

**roundness** 一个介于 0 到 999 之间的整数值，它指定角要使用的圆度。该值指定为点的数目。值越大，则圆度越大。

**bSuppressFill** 一个布尔值，如果设置为 true，则此方法创建没有填充的形状。默认值为 false。此参数是可选的。

**bSuppressStroke** 一个布尔值，如果设置为 true，则此方法创建没有笔触的矩形。默认值为 false。此参数是可选的。

#### 返回

无。

#### 描述

方法；添加一个新的矩形或圆角矩形，并使其适应指定的范围。此方法与“矩形”工具执行的操作相同。此方法使用文档当前默认的笔触和填充属性，在当前的帧和图层上添加矩形。如果 `bSuppressFill` 和 `bSuppressStroke` 都设置为 `true`，则此方法不起作用。

#### 示例

下面的示例在指定坐标范围内添加一个不带圆角的新矩形；它的高和宽都是 100 像素：

```
fl.getDocumentDOM().addNewRectangle({left:0,top:0,right:100,bottom:100},0);
```

下面的示例添加一个不带圆角和填充的新矩形；它的宽为 100 像素，高为 200 像素：

```
fl.getDocumentDOM().addNewRectangle({left:10,top:10,right:110,bottom:210},0, true);
```

下面的示例添加一个不带圆角和笔触的新矩形；它的宽为 200 像素，高为 100 像素：

```
fl.getDocumentDOM().addNewRectangle({left:20,top:20,right:220,bottom:120},0, false, true);
```

另请参见

[document.addNewPrimitiveRectangle\(\)](#)

## document.addNewScene()

#### 可用性

Flash MX 2004。

#### 用法

```
document.addNewScene([name])
```

#### 参数

**name** 指定场景的名称。如果不指定名称，则生成一个新的场景名称。

#### 返回

布尔值：如果场景添加成功，则为 `true`；否则为 `false`。

#### 说明

方法；添加一个新场景（[Timeline 对象](#)），作为当前所选场景之后的下一个场景，并使该新场景成为当前所选场景。如果指定的场景名称已经存在，则不添加该场景且方法返回一个错误。

#### 示例

下面的示例在当前文档的当前场景之后添加一个名为 `myScene` 的新场景。如果创建了新场景，则变量 `success` 将为 `true`；否则为 `false`。

```
var success = fl.getDocumentDOM().addNewScene("myScene");
```

下面的示例使用默认命名约定添加一个新场景。如果只存在一个场景，则新建场景命名为 "Scene 2"。

```
fl.getDocumentDOM().addNewScene();
```

## document.addNewText()

可用性

Flash MX 2004 ; 在 Flash CS3 Professional 中添加的可选 **text** 参数。

用法

```
document.addNewText(boundingBox [, text ])
```

参数

**boundingBox** 指定文本字段的大小和位置。有关 **boundingBox** 的格式信息, 请参阅 [document.addNewRectangle\(\)](#)。

**text** 一个可选字符串, 它指定要在字段中放置的文本。如果省略此参数, 则所选的“工具”面板将切换为“文本”工具。因此, 如果不打算更改所选工具, 请为 **text** 传递值。

返回

无。

描述

方法; 插入新文本字段并且可以选择性地将文本放置到字段中。如果省略 **text** 参数, 则可以调用 [document.setTextString\(\)](#) 来填充文本字段。

示例

下面的示例在舞台的左上角创建新的文本字段, 并将文本字符串设置为 **Hello World**:

```
fl.getDocumentDOM().addNewText({left:0, top:0, right:100, bottom:100} , "Hello World!" );  
fl.getDocumentDOM().setTextString('Hello World!');
```

另请参见

[document.setTextString\(\)](#)

## document.align()

可用性

Flash MX 2004。

用法

```
document.align(alignmode [, bUseDocumentBounds])
```

参数

**alignmode** 一个字符串, 它指定所选内容的对齐方式。可接受值为 "left"、"right"、"top"、"bottom"、"vertical center" 和 "horizontal center"。

**bUseDocumentBounds** 一个布尔值, 如果设置为 **true**, 则此方法与文档的范围对齐。否则, 此方法使用所选对象的范围。默认值为 **false**。此参数是可选的。

返回

无。



#### 描述

方法；对齐所选内容。

#### 示例

下面的示例将对象与舞台左对齐。此操作等效于在“对齐”面板中打开“相对于舞台”设置，然后单击“左对齐”按钮：

```
fl.getDocumentDOM().align("left", true);
```

另请参见

[document.distribute\(\)](#)、[document.getAlignToDocument\(\)](#)、[document.setAlignToDocument\(\)](#)

## document.allowScreens()

#### 可用性

Flash MX 2004。

#### 用法

```
document.allowScreens()
```

#### 参数

无。

#### 返回

布尔值：如果可以安全地使用 `document.screenOutline`，则为 `true`；否则为 `false`。

#### 说明

方法；在使用 `document.screenOutline` 属性之前使用。如果此方法返回值 `true`，则可以安全地访问 `document.screenOutline`；如果在文档中访问 `document.screenOutline` 时不使用屏幕，则 **Flash** 显示一个错误。

#### 示例

下面的示例确定是否可在当前文档中使用 `screens` 方法：

```
if(fl.getDocumentDOM().allowScreens()) {  
    fl.trace("screen outline is available.");  
}  
else {  
    fl.trace("whoops, no screens.");  
}
```

另请参见

[document.screenOutline](#)

## document.arrange()

#### 可用性

Flash MX 2004。

#### 用法

```
document.arrange( arrangeMode )
```

#### 参数

**arrangeMode** 指定所选内容的移动方向。可接受值为 "back"、"backward"、"forward" 和 "front"。该参数提供的功能与这些选项在“修改”>“排列”菜单上提供的功能相同。

#### 返回

无。

#### 描述

方法；排列舞台上的所选内容。此方法仅应用于非形状对象。

#### 示例

下面的示例将当前所选内容移到最前：

```
fl.getDocumentDOM().arrange("front");
```

## document.as3AutoDeclare

#### 可用性

Flash CS3 Professional。

#### 用法

```
document.as3AutoDeclare
```

#### 描述

属性；一个布尔值，它说明是否将放置在舞台上的实例自动添加到用户定义的时间轴类中。默认值为 **true**。

#### 示例

下面的示例指定必须将要放置在当前文档的舞台上的实例手动添加到用户定义的时间轴类中。

```
fl.getDocumentDOM().as3AutoDeclare=false;
```

## document.as3Dialect

#### 可用性

Flash CS3 Professional。

#### 用法

```
document.as3Dialect
```

#### 描述

属性；一个字符串，它描述指定文档中所使用的 **ActionScript 3.0**“术语”。默认值为 "AS3"。如果要允许原型类，如早期 **ECMAScript** 规范所允许的那样，请将此值设置为 "ES"。

#### 示例

下面的示例指定当前文档中使用的术语为 ECMAScript:

```
fl.getDocumentDOM().as3Dialect="ES";
```

另请参见

[document.asVersion](#)

## document.as3ExportFrame

#### 可用性

Flash CS3 Professional。

#### 用法

```
document.as3ExportFrame
```

#### 描述

属性; 一个整数, 它指定在哪一帧导出 ActionScript 3.0 类。默认情况下, 类在帧 1 中导出。

#### 示例

下面的示例将导出类的帧从 1 (默认值) 更改为 5。

```
var myDocument = fl.getDocumentDOM();  
fl.outputPanel.trace("Export classes in frame: ' value before modification is " +  
myDocument.as3ExportFrame);  
myDocument.as3ExportFrame = 5;  
fl.outputPanel.trace("Export classes in frame: ' value after modification is " + myDocument.as3ExportFrame);
```

## document.as3StrictMode

#### 可用性

Flash CS3 Professional。

#### 用法

```
document.as3StrictMode
```

#### 描述

属性; 一个布尔值, 它指定 ActionScript 3.0 编译器在进行编译时是打开 (true) 还是关闭 (false) “严格模式” 选项。“严格模式” 将使警告被报告为错误, 意味着如果存在这些错误, 编译将会失败。默认值为 true。

#### 示例

下面的示例关闭 “严格模式” 编译器选项。

```
var myDocument = fl.getDocumentDOM();  
fl.outputPanel.trace("Strict Mode value before modification is " + myDocument.as3StrictMode);  
myDocument.as3StrictMode = false;  
fl.outputPanel.trace("Strict Mode value after modification is " + myDocument.as3StrictMode);
```

另请参见

[document.as3WarningsMode](#)

## document.as3WarningsMode

可用性

Flash CS3 Professional。

用法

`document.as3WarningsMode`

描述

属性；一个布尔值，它指定 ActionScript 3.0 编译器在进行编译时是打开 (`true`) 还是关闭 (`false`)“警告模式”选项。“警告模式”将报告额外警告，在将 ActionScript 2.0 代码更新为 ActionScript 3.0 时，这对于发现不兼容性非常有用。默认值为 `true`。

示例

下面的示例关闭“警告模式”编译器选项。

```
var myDocument = fl.getDocumentDOM();
fl.outputPanel.trace("Warnings Mode value before modification is " + myDocument.as3WarningsMode);
myDocument.as3WarningsMode = false;
fl.outputPanel.trace("Warnings Mode value after modification is " + myDocument.as3WarningsMode);
```

另请参见

[document.as3StrictMode](#)

## document.asVersion

可用性

Flash CS3 Professional。

用法

`document.asVersion`

描述

属性；一个整数，它指定在指定文档中使用 ActionScript 的哪一版本。可接受的值为 1、2 和 3。

若要确定指定文档的目标播放器版本，请使用 [document.getPlayerVersion\(\)](#)；此方法将返回一个字符串，以便能供 Flash® Lite™ 播放器使用。

示例

下面的示例将当前文档中的 ActionScript 版本设置为 ActionScript 2.0（如果当前设置为 ActionScript 1.0）。

```
if(fl.getDocumentDOM().asVersion == 1){
    fl.getDocumentDOM().asVersion = 2;
}
```

另请参见

[document.as3Dialect](#)、[document.getPlayerVersion\(\)](#)

## document.autoLabel

可用性

Flash MX 2004。

用法

```
document.autoLabel
```

描述

属性；一个布尔值，它等效于“辅助功能”面板中的“自动标签”复选框。可以使用此属性指示 **Flash** 使用与舞台上的对象关联的文本自动标记这些对象。

示例

下面的示例获取 `autoLabel` 属性的值，然后在“输出”面板中显示结果：

```
var isAutoLabel = fl.getDocumentDOM().autoLabel;  
fl.trace(isAutoLabel);
```

下面的示例将 `autoLabel` 属性设置为 `true`，以指示 **Flash** 自动标记舞台上的对象：

```
fl.getDocumentDOM().autoLabel = true;
```

## document.backgroundColor

可用性

Flash MX 2004。

用法

```
document.backgroundColor
```

描述

属性；背景的颜色；使用以下格式之一：

- 格式为 "#RRGGBB" 或 "#RRGGBBAA" 的字符串
- 格式为 0xRRGGBB 的十六进制数字
- 表示与十六进制数字等价的十进制整数

示例

下面的示例将背景色设置为黑色：

```
fl.getDocumentDOM().backgroundColor = '#000000';
```

## document.breakApart()

可用性

Flash MX 2004。

用法

```
document.breakApart()
```

参数

无。

返回

无。

描述

方法；对当前所选内容执行分离操作。

示例

下面的示例分离当前所选内容：

```
fl.getDocumentDOM().breakApart();
```

## document.canEditSymbol()

可用性

Flash MX 2004。

用法

```
document.canEditSymbol()
```

参数

无。

返回

布尔值：如果“编辑元件”菜单和功能可以使用，则为 **true**；否则为 **false**。

说明

方法；指示“编辑元件”菜单和功能是否已启用。这与能否编辑所选内容无关。此方法不用来测试是否允许使用 `fl.getDocumentDOM().enterEditMode()`。

示例

下面的示例在“输出”面板中显示“编辑元件”菜单和功能的状态：

```
fl.trace("fl.getDocumentDOM().canEditSymbol() returns: " + fl.getDocumentDOM().canEditSymbol());
```

## document.canRevert()

可用性

Flash MX 2004。

用法

```
document.canRevert()
```

参数

无。

返回

布尔值：如果可以成功地使用 `document.revert()` 或 `fl.revertDocument()` 方法，则为 `true`；否则为 `false`。

说明

方法；确定是否可以成功地使用 `document.revert()` 或 `fl.revertDocument()` 方法。

示例

下面的示例检查当前文档是否可以回复到以前保存的版本。如果可以，则 `fl.getDocumentDOM().revert()` 恢复以前保存的版本。

```
if (fl.getDocumentDOM().canRevert()) {  
    fl.getDocumentDOM().revert();  
}
```

## document.canTestMovie()

可用性

Flash MX 2004。

用法

```
document.canTestMovie()
```

参数

无。

返回

布尔值：如果可以成功使用 `document.testMovie()` 方法，则为 `true`；否则为 `false`。

说明

方法；确定是否可以成功使用 `document.testMovie()` 方法。

示例

下面的示例测试是否可以使用 `fl.getDocumentDOM().testMovie()`。如果可以，则调用此方法。

```
if (fl.getDocumentDOM().canTestMovie()) {  
    fl.getDocumentDOM().testMovie();  
}
```

另请参见

[document.canTestScene\(\)](#)、[document.testScene\(\)](#)

## document.canTestScene()

可用性

Flash MX 2004。

用法

```
document.canTestScene()
```

参数

无。

返回

布尔值：如果可以成功使用 [document.testScene\(\)](#) 方法，则为 `true`；否则为 `false`。

说明

方法；确定是否可以成功使用 [document.testScene\(\)](#) 方法。

示例

下面的示例首先测试是否可以成功地使用 `fl.getDocumentDOM().testScene()`。如果可以，则调用此方法。

```
if (fl.getDocumentDOM().canTestScene()) {  
    fl.getDocumentDOM().testScene();  
}
```

另请参见

[document.canTestMovie\(\)](#)、[document.testMovie\(\)](#)

## document.changeFilterOrder()

可用性

Flash 8。

用法

```
document.changeFilterOrder(oldIndex, newIndex)
```

参数

**oldIndex** 一个整数，它在滤镜列表中表示要改变位置的滤镜的当前索引（从零开始）位置。

**newIndex** 一个整数，它表示滤镜在列表中的新索引位置。

返回

无。



#### 描述

方法；更改滤镜在“滤镜”列表中的索引。newIndex 上方或下方的所有滤镜会相应地向上或向下移位。例如，如果对以下显示的滤镜发出命令 `fl.getDocumentDOM().changeFilterOrder(3, 0)`，则会重新排列滤镜，如下所示：

更改前	截至时限
<code>blurFilterdropShadowFilterglowFiltergradientBevelFilter</code>	<code>gradientBevelFilterblurFilterdropShadowFilterglowFilter</code>

如果接着发出 `fl.getDocumentDOM().changeFilterOrder(0, 2)` 命令，则会重新排列滤镜，如下所示：

更改前	截至时限
<code>gradientBevelFilterblurFilterdropShadowFilterglowFilter</code>	<code>blurFilterdropShadowFiltergradientBevelFilterglowFilter</code>

#### 示例

下面的示例将当前在“滤镜”列表中排在第二的滤镜移动到第一的位置。

```
fl.getDocumentDOM().changeFilterOrder(1, 0);
```

另请参见

[document.addFilter\(\)](#)、[document.disableFilter\(\)](#)、[document.enableFilter\(\)](#)、[document.getFilters\(\)](#)、[document.removeFilter\(\)](#)、[Filter 对象](#)

## document.clipCopy()

#### 可用性

Flash MX 2004。

#### 用法

```
document.clipCopy()
```

#### 参数

无。

#### 返回

无。

#### 描述

方法；将当前所选内容从文档复制到剪贴板。

若要将字符串复制到剪贴板，请使用 [fl.clipCopyString\(\)](#)。

#### 示例

下面的示例将当前所选内容从文档复制到剪贴板：

```
fl.getDocumentDOM().clipCopy();
```

另请参见

[document.clipCut\(\)](#)、[document.clipPaste\(\)](#)

## document.clipCut()

可用性

Flash MX 2004。

用法

```
document.clipCut()
```

参数

无。

返回

无。

描述

方法；从文档中剪切当前所选内容，然后将其写入剪贴板。

示例

下面的示例从文档中剪切当前所选内容，然后将其写入剪贴板：

```
fl.getDocumentDOM().clipCut();
```

另请参见

[document.clipCopy\(\)](#)、[document.clipPaste\(\)](#)、[fl.clipCopyString\(\)](#)

## document.clipPaste()

可用性

Flash MX 2004。

用法

```
document.clipPaste([bInPlace])
```

参数

**bInPlace** 一个布尔值，如果设置为 **true**，则此方法执行“粘贴到当前位置”操作。默认值为 **false**，将使此方法在文档的中心执行粘贴操作。此参数是可选的。

返回

无。

描述

方法；将剪贴板的内容粘贴到文档中。

示例

下面的示例将剪贴板内容粘贴到文档的中心：

```
fl.getDocumentDOM().clipPaste();
```

下面的示例将剪贴板中的内容粘贴到当前文档的当前位置:

```
fl.getDocumentDOM().clipPaste(true);
```

另请参见

[document.clipCopy\(\)](#)、[document.clipCut\(\)](#)、[fl.clipCopyString\(\)](#)

## document.close()

可用性

Flash MX 2004。

用法

```
document.close([bPromptToSaveChanges])
```

参数

**bPromptToSaveChanges** 一个布尔值, 如果设置为 **true**, 且文档中存在未保存的更改, 则此方法使用一个对话框提示用户。如果 **bPromptToSaveChanges** 设置为 **false**, 则不提示用户保存任何更改过的文档。默认值为 **true**。此参数是可选的。

返回

无。

描述

方法; 关闭指定的文档。

示例

下面的示例关闭当前文档, 并使用一个对话框提示用户保存更改:

```
fl.getDocumentDOM().close();
```

下面的示例关闭当前文档, 而不保存更改:

```
fl.getDocumentDOM().close(false);
```

## document.convertLinesToFills()

可用性

Flash MX 2004。

用法

```
document.convertLinesToFills()
```

参数

无。

返回

无。

#### 描述

方法；将所选对象上的线条转换为填充。

#### 示例

下面的示例将当前所选线条转换为填充：

```
fl.getDocumentDOM().convertLinesToFills();
```

## document.convertToSymbol()

#### 可用性

Flash MX 2004。

#### 用法

```
document.convertToSymbol(type, name, registrationPoint)
```

#### 参数

**type** 一个字符串，它指定要创建的元件的类型。可接受值为 "movie clip"、"button" 和 "graphic"。

**name** 一个字符串，它指定新元件的名称，名称必须是唯一的。提交一个空字符串可以使此方法创建一个唯一的元件名称。

**注册点 (registration point)** 指定表示元件的 0,0 位置的点。可接受的值为："top left"、"top center"、"top right"、"center left"、"center"、"center right"、"bottom left"、"bottom center" 和 "bottom right"。

#### 返回

新建元件的一个对象，如果无法创建该元件，则返回 `null`。

#### 说明

方法；将所选舞台项目转换为一个新元件。有关为元件定义链接和共享资源属性的信息，请参阅 [Item 对象](#)。

#### 示例

下面的示例使用指定的名称创建一个影片剪辑元件和一个按钮元件，并使用默认名称创建一个影片剪辑元件：

```
newMc = fl.getDocumentDOM().convertToSymbol("movie clip", "mcSymbolName", "top left");  
newButton = fl.getDocumentDOM().convertToSymbol("button", "btnSymbolName", "bottom right");  
newClipWithDefaultName = fl.getDocumentDOM().convertToSymbol("movie clip", "", "top left");
```

## document.crop()

#### 可用性

Flash 8。

#### 用法

```
document.crop()
```

#### 参数

无。

#### 返回

一个布尔值：如果成功，则为 `true`；否则为 `false`。

#### 说明

方法；使用所选的最上面的绘制对象裁剪在其下面选择的所有绘制对象。如果未选择任何绘制对象，或者所选任何项都不是绘制对象，则此方法返回 `false`。

#### 示例

下面的示例裁剪当前所选对象：

```
fl.getDocumentDOM().crop();
```

#### 另请参见

[document.deleteEnvelope\(\)](#)、[document.intersect\(\)](#)、[document.punch\(\)](#)、[document.union\(\)](#)、[shape.isDrawingObject](#)

## document.currentPublishProfile

#### 可用性

Flash MX 2004。

#### 用法

```
document.currentPublishProfile
```

#### 描述

属性；一个字符串，它为指定文档的活动发布配置文件指定名称。

#### 示例

下面的示例使用默认名称添加一个新的发布配置文件，然后在“输出”面板中显示该配置文件的名称：

```
fl.getDocumentDOM().addNewPublishProfile();  
fl.outputPanel.trace(fl.getDocumentDOM().currentPublishProfile);
```

下面的示例将所选发布配置文件更改为 "Default"：

```
fl.getDocumentDOM().currentPublishProfile = "Default";
```

## document.currentTimeline

#### 可用性

Flash MX 2004。

#### 用法

```
document.currentTimeline
```

#### 描述

属性；一个整数，它指定活动时间轴的索引。通过更改此属性的值可以设置活动时间轴；效果几乎等效于调用 [document.editScene\(\)](#)。唯一的差异在于，如果时间轴的索引无效，将不会收到错误消息；只是由于不设置该属性，因而会导致静默失败。

#### 示例

下面的示例显示当前时间轴的索引：

```
var myCurrentTL = fl.getDocumentDOM().currentTimeline;  
fl.trace("The index of the current timeline is: "+ myCurrentTL);
```

下面的示例将活动时间轴从主时间轴更改为名为 "myScene" 的场景：

```
var i = 0;  
var curTimelines = fl.getDocumentDOM().timelines;  
while(i < fl.getDocumentDOM().timelines.length){  
    if(curTimelines[i].name == "myScene"){  
        fl.getDocumentDOM().currentTimeline = i;  
    }  
    ++i;  
}
```

另请参见

[document.getTimeline\(\)](#)

## document.debugMovie()

可用性

Flash Professional CS5。

用法

```
document.DebugMovie([Boolean abortIfErrorsExist])
```

描述

方法；对文档调用“调试影片”命令。

参数

**abortIfErrorsExist** 布尔值；默认值为 **false**。如果设置为 **true**，将不启动调试会话，如果存在编译器错误，则不会打开 **.swf** 窗口。编译器警告不会中止命令。

示例

以下示例将在调试模式下打开当前文档，但如果存在编译器错误，则中止操作：

```
fl.getDocumentDOM().debugMovie(1);
```

## document.deleteEnvelope()

可用性

Flash 8。

用法

```
document.deleteEnvelope()
```

参数

无。

#### 返回

一个布尔值：如果成功，则为 `true`；否则为 `false`。

#### 说明

方法；从所选对象删除封套（包含一个或多个对象的边框）。

#### 示例

下面的示例从所选对象删除封套：

```
fl.getDocumentDOM().deleteEnvelope();
```

#### 另请参见

[document.crop\(\)](#)、[document.intersect\(\)](#)、[document.punch\(\)](#)、[document.union\(\)](#)、[shape.isDrawingObject](#)

## document.deletePublishProfile()

#### 可用性

Flash MX 2004。

#### 用法

```
document.deletePublishProfile()
```

#### 参数

无。

#### 返回

一个整数，它是新的当前配置文件的索引。如果没有新的配置文件，则此方法不更改当前配置文件并返回其索引。

#### 说明

方法；如果存在多个配置文件，则删除当前活动配置文件。必须至少保留一个配置文件。

#### 示例

下面的示例删除当前活动配置文件（如果有多个），并显示新的当前活动配置文件的索引：

```
alert(fl.getDocumentDOM().deletePublishProfile());
```

#### 另请参见

[document.addNewPublishProfile\(\)](#)

## document.deleteScene()

#### 可用性

Flash MX 2004。

#### 用法

```
document.deleteScene()
```

#### 参数

无。

#### 返回

布尔值：如果场景删除成功，则为 **true**；否则为 **false**。

#### 说明

方法；删除当前场景（**Timeline 对象**）；如果删除的场景不是最后一个场景，则将下一个场景设置为当前 **Timeline 对象**。如果删除的场景是最后一个场景，则将第一个对象设置为当前 **Timeline 对象**。如果只存在一个 **Timeline 对象**（场景），则返回值 **false**。

#### 示例

假设当前文档中有三个场景（Scene0、Scene1 和 Scene2），则下面的示例将 Scene2 作为当前场景，然后将其删除：

```
f1.getDocumentDOM().editScene(2);  
var success = f1.getDocumentDOM().deleteScene();
```

## document.deleteSelection()

#### 可用性

Flash MX 2004。

#### 用法

```
document.deleteSelection()
```

#### 参数

无。

#### 返回

无。

#### 描述

方法；在舞台上删除当前所选内容。如果未选择任何内容，则显示一条错误消息。

#### 示例

下面的示例删除文档中的当前所选内容：

```
f1.getDocumentDOM().deleteSelection();
```

## document.description

#### 可用性

Flash MX 2004。

#### 用法

```
document.description
```



#### 描述

属性；一个字符串，它等效于“辅助功能”面板中的“描述”字段。该描述由屏幕阅读器读出。

#### 示例

下面的示例设置文档的描述：

```
fl.getDocumentDOM().description= "This is the main movie";
```

下面的示例获取文档的描述，然后在“输出”面板中显示：

```
fl.trace(fl.getDocumentDOM().description);
```

## document.disableAllFilters()

#### 可用性

Flash 8。

#### 用法

```
document.disableAllFilters()
```

#### 参数

无。

#### 返回

无。

#### 描述

方法；禁用所选对象上的所有滤镜。

#### 示例

下面的示例禁用所选对象上的所有滤镜：

```
fl.getDocumentDOM().disableAllFilters();
```

#### 另请参见

[document.addFilter\(\)](#)、[document.changeFilterOrder\(\)](#)、[document.disableFilter\(\)](#)、[document.disableOtherFilters\(\)](#)、[document.enableAllFilters\(\)](#)、[document.getFilters\(\)](#)、[document.removeAllFilters\(\)](#)、[Filter 对象](#)

## document.disableFilter()

#### 可用性

Flash 8。

#### 用法

```
document.disableFilter(filterIndex)
```

#### 参数

**filterIndex** 一个整数，它表示滤镜在“滤镜”列表中的零始索引。

**Document** 对象

返回

无。

描述

方法；禁用滤镜列表中的指定滤镜。

示例

下面的示例从所选对象的“滤镜”列表中禁用第一个和第三个滤镜（索引值为 0 和 2）：

```
f1.getDocumentDOM().disableFilter(0);  
f1.getDocumentDOM().disableFilter(2);
```

另请参见

[document.addFilter\(\)](#)、[document.changeFilterOrder\(\)](#)、[document.disableAllFilters\(\)](#)、[document.disableOtherFilters\(\)](#)、[document.enableFilter\(\)](#)、[document.getFilters\(\)](#)、[document.removeFilter\(\)](#)、[Filter](#) 对象

## document.disableOtherFilters()

可用性

Flash 8。

用法

```
document.disableOtherFilters(enabledFilterIndex)
```

参数

**enabledFilterIndex** 一个整数，它表示在禁用其他滤镜后应保持启用的滤镜的索引（从零开始）。

返回

无。

描述

方法；禁用滤镜列表中指定位置之外的所有滤镜。

示例

下面的示例禁用列表中第二个滤镜（索引值为 1）之外的所有滤镜：

```
f1.getDocumentDom().disableOtherFilters(1);
```

另请参见

[document.addFilter\(\)](#)、[document.changeFilterOrder\(\)](#)、[document.disableAllFilters\(\)](#)、[document.disableFilter\(\)](#)、[document.enableFilter\(\)](#)、[document.getFilters\(\)](#)、[document.removeFilter\(\)](#)、[Filter](#) 对象

## document.distribute()

可用性

Flash MX 2004。

#### 用法

```
document.distribute(distributemode [, bUseDocumentBounds])
```

#### 参数

**distributemode** 一个字符串，它指定所选对象的分散位置。可接受值为 "left edge"、"horizontal center"、"right edge"、"top edge"、"vertical center" 和 "bottom edge"。

**bUseDocumentBounds** 一个布尔值，如果设置为 **true**，则使用文档的范围分散所选对象。否则，此方法使用所选对象的范围。默认值为 **false**。

#### 返回

无。

#### 描述

方法；分散所选内容。

#### 示例

下面的示例按所选对象的上边缘来分散这些对象：

```
fl.getDocumentDOM().distribute("top edge");
```

下面的示例按所选对象的上边缘来分散这些对象，并明确设置 **bUseDocumentBounds** 参数：

```
fl.getDocumentDOM().distribute("top edge", false);
```

下面的示例使用文档的范围和所选对象的上边缘分散所选对象：

```
fl.getDocumentDOM().distribute("top edge", true);
```

另请参见

[document.getAlignToDocument\(\)](#)、[document.setAlignToDocument\(\)](#)

## document.distributeToLayers()

#### 可用性

Flash MX 2004。

#### 用法

```
document.distributeToLayers()
```

#### 参数

无。

#### 返回

无。

#### 描述

方法；对当前所选内容执行分散到图层操作，此操作等效于选择“分散到图层”。如果未选择任何内容，则此方法显示一个错误。

#### 示例

下面的示例将当前所选内容分散到图层:

```
f1.getDocumentDOM().distributeToLayers();
```

## document.docClass

#### 可用性

Flash CS3 Professional。

#### 用法

```
document.docClass
```

#### 描述

属性; 一个字符串, 它指定与文档关联的顶级 ActionScript 3.0 类。如果未将文档配置为使用 ActionScript 3.0, 则将忽略此属性。

#### 示例

下面的示例指定与文档关联的 ActionScript 3.0 类为 `com.mycompany.ManagerClass`, 此类是在 `com/mycompany/ManagerClass.as` 中定义的:

```
var myDocument = f1.getDocumentDOM();  
// set the property  
myDocument.docClass = "com.mycompany.ManagerClass";  
// get the property  
f1.outputPanel.trace("document.docClass has been set to " + myDocument.docClass);
```

#### 另请参见

[item.linkageBaseClass](#)

## document.documentHasData()

#### 可用性

Flash MX 2004。

#### 用法

```
document.documentHasData(name)
```

#### 参数

**name** 一个字符串, 它指定要检查的数据的名称。

#### 返回

布尔值: 如果文档具有永久数据, 则为 `true`; 否则为 `false`。

#### 说明

方法; 检查文档中是否存在具有指定名称的永久数据。

#### 示例

下面的示例检查文档中是否存在具有名称 "myData" 的永久性数据:

```
var hasData = fl.getDocumentDOM().documentHasData("myData");
```

另请参见

[document.addDataToDocument\(\)](#)、[document.getDataFromDocument\(\)](#)、[document.removeDataFromDocument\(\)](#)

## document.duplicatePublishProfile()

#### 可用性

Flash MX 2004。

#### 用法

```
document.duplicatePublishProfile([profileName])
```

#### 参数

**profileName** 一个字符串，它指定直接复制的配置文件的唯一名称。如果不指定名称，则此方法使用默认名称。此参数是可选的。

#### 返回

一个整数，它是新配置文件在配置文件列表中的索引。如果该配置文件无法直接复制，则返回 -1。

#### 说明

方法；直接复制当前活动配置文件，并提供直接复制版本焦点。

#### 示例

下面的示例直接复制当前活动的配置文件，并在“输出”面板中显示新配置文件的索引:

```
fl.trace(fl.getDocumentDOM().duplicatePublishProfile("dup profile"));
```

## document.duplicateScene()

#### 可用性

Flash MX 2004。

#### 用法

```
document.duplicateScene()
```

#### 参数

无。

#### 返回

布尔值：如果场景复制成功，则为 true；否则为 false。

#### 说明

方法；制作当前所选场景的副本，为该新场景提供一个唯一的名称，并将它作为当前场景。

#### 示例

下面的示例直接复制当前文档中的第二个场景：

```
fl.getDocumentDOM().editScene(1); //Set the middle scene to current scene.  
var success = fl.getDocumentDOM().duplicateScene();
```

## document.duplicateSelection()

#### 可用性

Flash MX 2004。

#### 用法

```
document.duplicateSelection()
```

#### 参数

无。

#### 返回

无。

#### 描述

方法；直接复制舞台上的所选内容。

#### 示例

下面的示例直接复制当前所选内容，此操作与按住 **Alt** 键单击然后拖动项目类似：

```
fl.getDocumentDOM().duplicateSelection();
```

## document.editScene()

#### 可用性

Flash MX 2004。

#### 用法

```
document.editScene(index)
```

#### 参数

**index** 一个从零开始的整数，它指定要编辑的场景。

#### 返回

无。

#### 描述

方法；将指定场景作为当前所选场景进行编辑。

#### 示例

假设当前文档中有三个场景（Scene0、Scene1 和 Scene2），则下面的示例将 Scene2 作为当前场景，然后将其删除：

```
fl.getDocumentDOM().editScene(2);  
fl.getDocumentDOM().deleteScene();
```

## document.enableAllFilters()

#### 可用性

Flash 8。

#### 用法

```
document.enableAllFilters()
```

#### 参数

无。

#### 返回

无。

#### 描述

方法；为所选对象启用滤镜列表上的所有滤镜。

#### 示例

下面的示例为所选对象启用“滤镜”列表上的所有滤镜：

```
fl.getDocumentDOM().enableAllFilters();
```

#### 另请参见

[document.addFilter\(\)](#)、[document.changeFilterOrder\(\)](#)、[document.disableAllFilters\(\)](#)、[document.enableFilter\(\)](#)、[document.getFilters\(\)](#)、[document.removeAllFilters\(\)](#)、[Filter 对象](#)

## document.enableFilter()

#### 可用性

Flash 8。

#### 用法

```
document.enableFilter(filterIndex)
```

#### 参数

**filterIndex** 一个整数，它指定要启用的滤镜在滤镜列表中的索引（从零开始）。

#### 返回

无。

**Document** 对象

## 描述

方法；为所选对象启用指定的滤镜。

## 示例

下面的示例启用所选对象的第二个滤镜：

```
fl.getDocumentDOM().enableFilter(1);
```

另请参见

[document.addFilter\(\)](#)、[document.changeFilterOrder\(\)](#)、[document.disableFilter\(\)](#)、[document.enableAllFilters\(\)](#)、[document.getFilters\(\)](#)、[document.removeFilter\(\)](#)、[Filter](#) 对象

## document.enterEditMode()

## 可用性

Flash MX 2004。

## 用法

```
document.enterEditMode([editMode])
```

## 参数

**editMode** 一个字符串，它指定编辑模式。可接受值为 "inPlace" 或 "newWindow"。如果未指定任何参数，则默认值为元件编辑模式。此参数是可选的。

## 返回

无。

## 描述

方法；将创作工具切换到由此参数指定的编辑模式。如果未指定任何参数，则此方法默认为元件编辑模式，这与右键单击元件调用上下文菜单并选择“编辑”的结果一样。

## 示例

下面的示例针对当前所选元件使 **Flash** 处于“在当前位置编辑”模式：

```
fl.getDocumentDOM().enterEditMode('inPlace');
```

下面的示例针对当前所选元件使 **Flash** 处于“在新窗口中编辑”模式：

```
fl.getDocumentDOM().enterEditMode('newWindow');
```

另请参见

[document.exitEditMode\(\)](#)

## document.exitEditMode()

## 可用性

Flash MX 2004。



#### 用法

```
document.exitEditMode()
```

#### 参数

无。

#### 返回

无。

#### 描述

方法：退出元件编辑模式，并将焦点返回编辑模式的上一级。例如，如果您正在其他元件中编辑一个元件，则此方法使您从正在编辑的元件向上一级进入到父元件中。

#### 示例

下面的示例退出元件编辑模式：

```
fl.getDocumentDOM().exitEditMode();
```

另请参见

[document.enterEditMode\(\)](#)

## document.exportPNG()

#### 可用性

Flash 8。

#### 用法

```
document.exportPNG([fileURI [, bCurrentPNGSettings [, bCurrentFrame]])
```

#### 参数

**fileURI** 一个字符串，表示为 **file:/// URI**，它指定导出的文件的名称。如果 **fileURI** 为空字符串或未指定，则 **Flash** 显示“导出影片”对话框。

**bCurrentPNGSettings** 一个布尔值，它指定是使用当前的 **PNG** 发布设置 (**true**) 还是显示“导出 **PNG**”对话框 (**false**)。此参数是可选的。默认值为 **false**。

**bCurrentFrame** 一个布尔值，它指定是仅导出当前帧 (**true**)，还是导出所有帧，且每个帧都导出为单独的 **PNG** 文件 (**false**)。此参数是可选的。默认值为 **false**。

#### 返回

一个布尔值；如果文件成功导出为 **PNG** 文件，则为 **true**；否则为 **false**。

#### 说明

方法：将文档导出为一个或多个 **PNG** 文件。如果指定了 **fileURI** 且此文件已经存在，则覆盖此文件并且不进行警告。

#### 示例

下面的示例使用当前的 **PNG** 发布设置将当前文档中的当前帧导出到 **myFile.png**：

```
fl.getDocumentDOM().exportPNG("file:///C:/myProject/myFile.png", true, true);
```

## document.exportPublishProfile()

可用性

Flash MX 2004。

用法

```
document.exportPublishProfile(fileURI)
```

参数

**fileURI** 一个字符串，表示为 **file:/// URI**，它指定将配置文件导出到其中的 XML 文件的路径。

返回

无。

描述

方法；将当前活动的配置文件导出到 XML 文件。

示例

下面的示例将当前活动的配置文件导出到名为 **profile.xml** 的文件，该文件位于 C 驱动器上的文件夹 **/Documents and Settings/username/Desktop** 中：

```
fl.getDocumentDOM().exportPublishProfile('file:///C:/Documents and Settings/username/Desktop/profile.xml');
```

另请参见

[document.exportPublishProfileString\(\)](#)、[document.importPublishProfile\(\)](#)

## document.exportPublishProfileString()

可用性

Flash CS4 Professional.

用法

```
document.exportPublishProfileString( [profileName] )
```

参数

**profileName** 一个字符串，它指定要导出到 XML 字符串的配置文件的名称。此参数是可选的。

返回

一个 XML 字符串。

说明

方法：返回一个字符串，它以 XML 格式指示指定的配置文件。如果不为 **profileName** 传递值，则导出当前配置文件。

示例

下面的示例将表示当前配置文件的 XML 字符串存储在名为 **profileXML** 的变量中，然后在“输出”面板中显示该字符串：

```
var profileXML=f1.getDocumentDOM().exportPublishProfileString();  
f1.trace(profileXML);
```

另请参见

[document.exportPublishProfile\(\)](#)、[document.importPublishProfileString\(\)](#)

## document.exportSWF()

可用性

Flash MX 2004。

用法

```
document.exportSWF([fileURI [, bCurrentSettings]])
```

参数

**fileURI** 一个字符串，表示为 `file:/// URI`，它指定导出的文件的名称。如果 `fileURI` 为空或未指定，则 Flash 显示“导出影片”对话框。此参数是可选的。

**bCurrentSettings** 一个布尔值，如果设置为 `true`，则 Flash 使用当前的 SWF 发布设置。否则，Flash 显示“导出 Flash Player”对话框。默认值为 `false`。此参数是可选的。

返回

无。

描述

方法；以 Flash SWF 格式导出文档。

示例

下面的示例使用当前发布设置将文档导出到指定的文件位置：

```
f1.getDocumentDOM().exportSWF("file:///C:/Documents and Settings/joe_user/Desktop/qwerty.swf");
```

下面的示例显示“导出影片”对话框和“导出 Flash Player”对话框，然后按指定设置导出文档：

```
f1.getDocumentDOM().exportSWF("", true);
```

下面的示例显示“导出影片”对话框，然后按指定设置导出文档：

```
f1.getDocumentDOM().exportSWF();
```

## document.externalLibraryPath

可用性

Flash CS4 Professional.

用法

```
document.externalLibraryPath
```

#### 描述

属性；一个字符串，其中包含文档的 **ActionScript 3.0** 外部库路径中项目的列表，用于指定用作运行时共享库的 **SWC** 文件的位置。该字符串中的项目以分号分隔。在创作工具中，通过以下方式指定这些项目：选择“文件”>“发布设置”，然后在“Flash”选项卡上选择“ActionScript 3.0 脚本设置”。

#### 示例

下面的示例将文档的外部库路径设置为“.”和“../mySWCLibrary”：

```
var myDocument = fl.getDocumentDOM();  
myDocument.externalLibraryPath = ".;../mySWCLibrary";  
fl.trace(myDocument.externalLibraryPath);
```

#### 另请参见

[document.libraryPath](#)、[document.sourcePath](#)、[fl.externalLibraryPath](#)

## document.forceSimple

#### 可用性

Flash MX 2004。

#### 用法

```
document.forceSimple
```

#### 描述

属性；一个布尔值，它指定是否可以访问指定对象的子项。此属性等效于“辅助功能”面板中的“使子对象可访问”设置的反向逻辑。即，如果 **forceSimple** 为 **true**，则等效于取消选定“使子对象可访问”选项。如果 **forceSimple** 为 **false**，则与选中“使子对象可访问”选项的效果相同。

#### 示例

下面的示例将 **areChildrenAccessible** 变量设置为 **forceSimple** 属性的值。值为 **false** 表示子对象可访问。

```
var areChildrenAccessible = fl.getDocumentDOM().forceSimple;
```

下面的示例将 **forceSimple** 属性设置为允许访问文档的子项：

```
fl.getDocumentDOM().forceSimple = false;
```

## document.frameRate

#### 可用性

Flash MX 2004。

#### 用法

```
document.frameRate
```

#### 描述

属性；一个浮点值，它指定播放 **SWF** 文件时每秒显示的帧数，默认值为 **12**。此属性的功能与在 **FLA** 文件的“文档属性”对话框（“修改”>“文档”）中设置默认帧频相同。

#### 示例

下面的示例将帧频设置为每秒 25.5 帧：

```
fl.getDocumentDOM().frameRate = 25.5;
```

## document.getAlignToDocument()

#### 可用性

Flash MX 2004。

#### 用法

```
document.getAlignToDocument()
```

#### 参数

无。

#### 返回

布尔值：如果首选参数设置为将对象与舞台对齐，则为 **true**；否则为 **false**。

#### 说明

方法；与在“对齐”面板中检索“相对于舞台”按钮的值相同。获取可用于文档的 [document.align\(\)](#)、[document.distribute\(\)](#)、[document.match\(\)](#) 和 [document.space\(\)](#) 方法的首选参数。

#### 示例

下面的示例在“对齐”面板中检索“相对于舞台”按钮的值。如果返回值为 **true**，则“相对于舞台”按钮是活动的，否则不是活动的。

```
var isAlignToDoc = fl.getDocumentDOM().getAlignToDocument();  
fl.getDocumentDOM().align("left", isAlignToDoc);
```

#### 另请参见

[document.setAlignToDocument\(\)](#)

## document.getBlendMode()

#### 可用性

Flash 8。

#### 用法

```
document.getBlendMode()
```

#### 参数

无。

#### 返回

一个字符串，它指定所选对象的混合模式。如果选中了多个对象，并且它们具有不同的混合模式，则该字符串将反映最深层对象的混合模式。

注：如果所选内容包含有不支持混合模式的对象，或者所包含对象的混合模式值为 "normal"（标准），则返回值不可预知。

说明

方法；返回一个字符串，用于指定所选对象的混合模式。

示例

下面的示例在“输出”面板中显示混合模式的名称：

```
fl.trace(fl.getDocumentDOM().getBlendMode());
```

## document.getCustomFill()

可用性

Flash MX 2004。

用法

```
document.getCustomFill([objectToFill])
```

参数

**objectToFill** 一个字符串，它指定填充对象的位置。下面是有效值：

- "toolbar" 返回“工具”面板和“属性”检查器的填充对象。
- "selection" 返回所选内容的填充对象。

如果省略此参数，则默认值为 "selection"。如果未选择任何内容，则此方法返回 **undefined**。此参数是可选的。

返回

如果成功，则返回由 **objectToFill** 参数指定的 **Fill 对象**；否则返回 **undefined**。

说明

方法；检索所选形状的填充对象，或者（如果指定）检索“工具”面板和“属性”检查器的填充对象。

示例

下面的示例获取所选内容的填充对象，然后将所选内容的颜色更改为白色：

```
var fill = fl.getDocumentDOM().getCustomFill();  
fill.color = '#FFFFFF';  
fill.style = "solid";  
fl.getDocumentDOM().setCustomFill(fill);
```

下面的示例返回“工具”面板和“属性”检查器的填充对象，然后将颜色样本更改为线性渐变：

```
var fill = fl.getDocumentDOM().getCustomFill("toolbar");  
fill.style = "linearGradient";  
fill.colorArray = [ 0x00ff00, 0xff0000, 0x0000ff ];  
fill.posArray = [0, 100, 200];  
fl.getDocumentDOM().setCustomFill( fill );
```

另请参见

[document.setCustomFill\(\)](#)

## document.getCustomStroke()

可用性

Flash MX 2004。

用法

```
document.getCustomStroke([locationOfStroke])
```

参数

**locationOfStroke** 一个字符串，它指定笔触对象的位置。下面是有效值：

- "toolbar", 如果设置为此值，则返回“工具”面板和“属性”检查器的笔触对象。
- "selection", 如果设置为此值，则返回所选内容的笔触对象。

如果省略此参数，则默认为"selection"。如果未选择任何内容，则返回 `undefined`。此参数是可选的。

返回

如果成功，则返回由 `locationOfStroke` 参数指定的 [Stroke 对象](#)；否则返回 `undefined`。

说明

返回所选形状的笔触对象，或者（如果指定）返回“工具”面板和“属性”检查器的笔触对象。

示例

下面的示例返回所选内容的当前笔触设置，并将笔触粗细更改为 2：

```
var stroke = fl.getDocumentDOM().getCustomStroke("selection");  
stroke.thickness = 2;  
fl.getDocumentDOM().setCustomStroke(stroke);
```

下面的示例返回“工具”面板和“属性”检查器的当前笔触设置，并将笔触颜色设置为红色：

```
var stroke = fl.getDocumentDOM().getCustomStroke("toolbar");  
stroke.color = "#FF0000";  
fl.getDocumentDOM().setCustomStroke(stroke);
```

另请参见

[document.setCustomStroke\(\)](#)

## document.getDataFromDocument()

可用性

Flash MX 2004。

用法

```
document.getDataFromDocument(name)
```

参数

**name** 一个字符串，它指定要返回的数据的名称。

返回  
指定的数据。

说明  
方法；检索指定数据的值。返回的类型取决于存储的数据的类型。

示例  
下面的示例将整数 12 添加到当前文档，并使用此方法在“输出”面板中显示该值：

```
fl.getDocumentDOM().addDataToDocument("myData", "integer", 12);  
fl.trace(fl.getDocumentDOM().getDataFromDocument("myData"));
```

另请参见  
[document.addDataToDocument\(\)](#)、[document.documentHasData\(\)](#)、[document.removeDataFromDocument\(\)](#)

## document.getElementProperty()

可用性  
Flash MX 2004。

用法  
`document.getElementProperty(propertyName)`

参数  
**propertyName** 一个字符串，它指定要检索其值的 **Element** 属性的名称。

返回  
指定属性的值。如果属性为不确定的状态（如选择的多个元素具有不同的属性值时），则返回 **null**。如果属性不是所选元素的有效属性，则返回 **undefined**。

说明  
方法；获取当前所选内容的指定 **Element** 属性。有关可接受值的列表，请参阅 [Element 对象](#) 的属性摘要表。

示例  
下面的示例获取当前所选内容的 **Element** 属性的 **name**：

```
// elementName = the instance name of the selected object.  
var elementName = fl.getDocumentDOM().getElementProperty("name");
```

另请参见  
[document.setElementProperty\(\)](#)

## document.getElementTextAttr()

可用性  
Flash MX 2004。



**Document 对象**

## 用法

```
document.getElementByIdTextAttr(attrName [, startIndex [, endIndex]])
```

## 参数

**attrName** 一个字符串，它指定要返回的 **TextAttrs** 属性的名称。有关属性名称和预期值的列表，请参阅 [TextAttrs 对象](#) 的属性摘要表。

**startIndex** 一个整数，它指定第一个字符的索引，用 0（零）指定第一个位置。此参数是可选的。

**endIndex** 一个整数，它指定最后一个字符的索引。此参数是可选的。

## 返回

如果只选择了一个文本字段，且该文本内仅使用了一个值，则返回属性。如果文本字段内使用了多个值，则返回 **undefined**。如果选择了多个文本字段，且所有文本对齐值都相等，则此方法返回此文本对齐值。如果选择了多个文本字段，但并非所有文本对齐值都相等，则此方法返回 **undefined**。如果不传递可选参数，且当前没有编辑文本，则这些规则应用于当前所选范围的文本或整个文本字段。如果只传递了 **startIndex**，且所有所选 **Text** 对象都与值匹配，则返回该索引右侧字符的属性。如果传递了 **startIndex** 和 **endIndex**，则返回的值反映从 **startIndex** 到（但不包括）**endIndex** 整个范围的字符。

## 说明

方法；获取所选 **Text** 对象的某个特定 **TextAttrs** 属性。所选非文本字段对象则被忽略。有关属性名称和预期值的列表，请参阅 [TextAttrs 对象](#) 的属性摘要表。另请参见 [document.setElementTextAttr\(\)](#)。

## 示例

下面的示例获取所选文本字段的大小：

```
fl.getDocumentDOM().getElementTextAttr("size");
```

下面的示例获取所选文本字段中位于索引 3 处的字符的颜色：

```
fl.getDocumentDOM().getElementTextAttr("fillColor", 3);
```

下面的示例获取所选文本字段中从索引 2 到（但不包括）索引 10 的文本的字体名称：

```
fl.getDocumentDOM().getElementTextAttr("face", 2, 10);
```

## document.getFilters()

## 可用性

Flash 8。

## 用法

```
document.getFilters()
```

## 参数

无。

## 返回

一个数组，它包含应用于当前所选对象的滤镜的列表。

## 说明

方法；返回一个数组，它包含应用于当前所选对象的滤镜的列表。如果选中多个对象，并且它们的滤镜不同，则该方法返回应用于第一个所选对象的滤镜的列表。

示例

请参阅 [document.setFilters\(\)](#)。

另请参见

[document.addFilter\(\)](#)、[document.changeFilterOrder\(\)](#)、[document.setFilters\(\)](#)、[Filter](#) 对象

## document.getMetadata()

可用性

Flash 8。

用法

```
document.getMetadata()
```

参数

无。

返回

一个字符串，它包含与文档关联的 XML 元数据；如果没有元数据，则为一个空字符串。

说明

方法；返回一个包含与文档关联的 XML 元数据的字符串，如果没有元数据，则返回一个空字符串。

示例

下面的示例在“输出”面板中显示来自当前文档的 XML 元数据。

```
f1.trace("XML Metadata is : " + f1.getDocumentDOM().getMetadata());
```

另请参见

[document.setMetadata\(\)](#)

## document.getMobileSettings()

可用性

Flash CS3 Professional。

用法

```
document.getMobileSettings()
```

参数

无。

返回

一个字符串，表示文档的 XML 设置。如果尚未设置任何值，则返回空字符串。

说明

方法；返回文档的移动 XML 设置。

示例

下面的示例显示当前文档的 XML 设置字符串：

```
fl.trace(fl.getDocumentDOM().getMobileSettings());  
//traces a string like the following"<? xml version="1.0" encoding="UTF-16" standalone="no"  
?><mobileSettings> <contentType id="standalonePlayer" name="Standalone Player"/> <testDevices> <testDevice  
id="1170" name="Generic Phone" selected="yes"/> </testDevices> <outputMsgFiltering info="no" trace="yes"  
warning="yes"/> <testWindowState height="496" splitterClosed="No" splitterXPos="400" width="907"/>  
</mobileSettings>"
```

另请参见

[document.setMobileSettings\(\)](#)

## document.getPlayerVersion()

可用性

Flash CS3 Professional。

用法

```
document.getPlayerVersion()
```

参数

无。

返回

一个字符串，它表示使用 `document.setPlayerVersion()` 指定的 Flash Player 版本。如果尚未设置任何值，则返回在“发布设置”对话框中指定的值。

说明

方法；返回一个字符串，该字符串表示指定文档的目标播放器版本。有关该方法可能返回值的列表，请参阅 [document.setPlayerVersion\(\)](#)。

若要确定在指定文件中是以哪一版本的 `ActionScript` 作为目标，请使用 [document.asVersion](#)。

示例

下面的示例展示了将指定的播放器版本作为当前文档的目标播放器，然后检索这些值：

```
fl.getDocumentDOM().setPlayerVersion("6");  
var version = fl.getDocumentDOM().getPlayerVersion();  
fl.trace(version) // displays "6"  
fl.getDocumentDOM().setPlayerVersion("FlashPlayer10");  
var version = fl.getDocumentDOM().getPlayerVersion();  
fl.trace(version) // displays "FlashPlayer10"
```

另请参见

[document.setPlayerVersion\(\)](#)

## document.getSelectionRect()

可用性

Flash MX 2004。

用法

```
document.getSelectionRect()
```

参数

无。

返回

当前所选内容的边界矩形，如果未选择任何内容，则为 0。有关返回值的格式的信息，请参阅 [document.addNewRectangle\(\)](#)。

说明

方法；获取当前所选内容的边界矩形。如果所选内容不是矩形，则返回包含所选全部内容的最小矩形。此矩形基于文档空间，如果在编辑模式下，则基于正在编辑的元件的注册点（即原点或零点）。

示例

下面的示例获取当前所选内容的边界矩形，然后显示其属性：

```
var newRect = fl.getDocumentDOM().getSelectionRect();  
var outputStr = "left: " + newRect.left + " top: " + newRect.top + " right: " + newRect.right + " bottom:  
" + newRect.bottom;  
alert(outputStr);
```

另请参见

[document.selection](#)、[document.setSelectionRect\(\)](#)

## document.getTextString()

可用性

Flash MX 2004。

用法

```
document.getTextString([startIndex [, endIndex]])
```

参数

**startIndex** 一个整数，它是要获取的第一个字符的索引。此参数是可选的。

**endIndex** 一个整数，它是要获取的最后一个字符的索引。此参数是可选的。

返回

一个包含所选文本的字符串。

#### 说明

方法；获取当前所选文本。如果不传递可选参数，则使用当前所选文本。如果要编辑的文本当前没有打开，则返回整个文本字符串。如果只传递 `startIndex`，则返回从该索引开始到字段结束的字符串。如果传递 `startIndex` 和 `endIndex`，则返回从 `startIndex` 到（但不包括）`endIndex` 的字符串。

如果选择多个文本字段，则返回所有字符串的连接。

#### 示例

下面的示例获取所选文本字段中的字符串：

```
fl.getDocumentDOM().getTextString();
```

下面的示例获取所选文本字段中位于字符索引 5 处的字符串：

```
fl.getDocumentDOM().getTextString(5);
```

下面的示例获取从字符索引 2 到（但不包括）字符索引 10 的字符串：

```
fl.getDocumentDOM().getTextString(2, 10);
```

另请参见

[document.setTextString\(\)](#)

## document.getTimeline()

#### 可用性

Flash MX 2004。

#### 用法

```
document.getTimeline()
```

#### 参数

无。

#### 返回

当前 **Timeline** 对象。

#### 说明

方法；检索文档中的当前 **Timeline** 对象。当前时间轴可以是当前场景、正在编辑的当前元件或当前屏幕。

#### 示例

下面的示例获取 **Timeline** 对象，并返回最长的图层中的帧数：

```
var longestLayer = fl.getDocumentDOM().getTimeline().frameCount;  
fl.trace("The longest layer has" + longestLayer + "frames");
```

下面的示例进入舞台上所选元件的“在当前位置编辑”模式，并在该元件的时间轴上插入一个帧。

```
fl.getDocumentDOM().enterEditMode("inPlace");  
fl.getDocumentDOM().getTimeline().insertFrames();
```

下面的示例获取 **Timeline** 对象，并显示其名称：

```
var timeline = fl.getDocumentDOM().getTimeline();  
alert(timeline.name);
```

另请参见

[document.currentTimeline](#)、[document.timelines](#)、[symbolItem.timeline](#)

## document.getTransformationPoint()

可用性

Flash MX 2004。

用法

```
document.getTransformationPoint()
```

参数

无。

返回

一个点（例如，{x:10,y:20}，其中 x 和 y 是浮点数），它指定所选元素坐标系内变形点（还可是原点或零点）的位置。

说明

方法；获取当前所选内容的变形点的位置。您可以使用变形点执行变形（如旋转和倾斜）。

注：变形点是相对于不同位置而言的，具体取决于所选项的类型。有关详细信息，请参阅 [document.setTransformationPoint\(\)](#)。

示例

下面的示例获取当前所选内容的变形点。`transPoint.x` 属性提供变形点的 x 坐标。`transPoint.y` 属性提供变形点的 y 坐标。

```
var transPoint = fl.getDocumentDOM().getTransformationPoint();
```

另请参见

[document.setTransformationPoint\(\)](#)、[element.getTransformationPoint\(\)](#)

## document.group()

可用性

Flash MX 2004。

用法

```
document.group()
```

参数

无。

返回

无。

描述

方法；将当前所选内容转换为一个组。

#### 示例

下面的示例将当前所选内容中的对象转换为一个组:

```
fl.getDocumentDOM().group();
```

另请参见

[document.unGroup\(\)](#)

## document.height

#### 可用性

Flash MX 2004。

#### 用法

```
document.height
```

#### 描述

属性; 一个整数, 它以像素为单位指定文档 (舞台) 的高度。

#### 示例

下面的示例将舞台的高度设置为 400 像素:

```
fl.getDocumentDOM().height = 400;
```

另请参见

[document.width](#)

## document.id

#### 可用性

Flash CS3 Professional。

#### 用法

```
document.id
```

#### 描述

只读属性; 一个 (自动分配的) 唯一整数, 用于在 Flash 会话期间标识文档。将此属性与 [fl.findDocumentDOM\(\)](#) 一起使用可为某操作指定特定文档。

#### 示例

下面的示例显示当前文档的文档 ID:

```
fl.trace("Current doc's internal ID is: " + fl.getDocumentDOM().id);
```

另请参见

[fl.findDocumentDOM\(\)](#)

## document.importFile()

可用性

Flash 8。

用法

```
document.importFile(fileURI [, importToLibrary])
```

参数

**fileURI** 一个字符串，表示为 `file:/// URI`，它指定要导入的文件的途径。

**importToLibrary** 一个布尔值，它指定是仅将文件导入文档库 (`true`)，还是同时在舞台上放置一个副本 (`false`)。默认值为 `false`。

返回

无。

描述

方法；将文件导入文档。此方法执行与“导入到库”或“导入到舞台”菜单命令相同的操作。若要导入发布配置文件，请使用 [document.importPublishProfile\(\)](#)。

示例

下面的示例允许用户浏览要导入到舞台的文件：

```
var dom = fl.getDocumentDOM();  
var URI = fl.browseForFileURL("select", "Import File");  
dom.importFile(URI);
```

另请参见

[document.importSWF\(\)](#)、[fl.browseForFileURL\(\)](#)

## document.importPublishProfile()

可用性

Flash MX 2004。

用法

```
document.importPublishProfile( fileURI )
```

参数

**fileURI** 一个字符串，表示为 `file:/// URI`，它指定 XML 文件（该文件定义要导入的配置文件）的途径。

返回

一个整数，它是导入的配置文件在配置文件列表中的索引。如果无法导入配置文件，则返回 `-1`。

说明

方法；从文件导入配置文件。



#### 示例

下面的示例导入 `profile.xml` 文件中包含的配置文件，并在配置文件列表中显示其索引：

```
alert(fl.getDocumentDOM().importPublishProfile('file:///C:/Documents and Settings/janeUser/Desktop/profile.xml'));
```

## document.importPublishProfileString()

#### 可用性

Flash CS4 Professional.

#### 用法

```
document.importPublishProfileString(xmlString)
```

#### 参数

**xmlString** 一个字符串，其中包含要导入为当前配置文件的 XML 数据。

#### 返回

一个布尔值：如果成功导入该字符串，则为 `true`；否则为 `false`。

#### 说明

方法；导入一个表示发布配置文件的 XML 字符串，并将其设置为当前配置文件。若要生成要导入的 XML 字符串，请在使用此方法之前使用 [document.exportPublishProfileString\(\)](#)。

#### 示例

在下面的示例中，默认配置文件导出为 XML 字符串。使用标准 JavaScript `replace` 命令修改该 XML 字符串。然后导入该字符串，并将默认 ActionScript 3 输出设置设为 ActionScript 1。

```
var profileXML=fl.getDocumentDOM().exportPublishProfileString('Default');
fl.trace(profileXML);
var newProfileXML = profileXML.replace("<ActionScriptVersion>3</ActionScriptVersion>",
"<ActionScriptVersion>1</ActionScriptVersion>");
fl.getDocumentDOM().importPublishProfileString(newProfileXML);
```

## document.importSWF()

#### 可用性

Flash MX 2004。

#### 用法

```
document.importSWF(fileURI)
```

#### 参数

**fileURI** 一个字符串，表示为 `file:/// URI`，它指定 SWF 文件要导入的文件。

#### 返回

无。

#### 描述

方法；将 SWF 文件导入文档。此方法执行的操作相当于使用“导入”菜单命令指定 SWF 文件。在 Flash 8 和更高版本中，还可以使用 `document.importFile()` 来导入 SWF 文件（以及其他类型的文件）。

#### 示例

下面的示例从 Flash Configuration 文件夹中导入 "mySwf.swf" 文件：

```
fl.getDocumentDOM().importSWF(fl.configURI+"mySwf.swf");
```

另请参见

[document.importFile\(\)](#)

## document.intersect()

可用性

Flash 8。

用法

```
document.intersect()
```

参数

无。

返回

一个布尔值：如果成功，则为 `true`；否则为 `false`。

说明

方法；从所有所选绘制对象创建一个交集绘制对象。如果未选择任何绘制对象，或者所选项都不是绘制对象，则此方法返回 `false`。

示例

下面的示例从所有所选绘制对象创建一个交集绘制对象：

```
fl.getDocumentDOM().intersect();
```

另请参见

[document.crop\(\)](#)、[document.deleteEnvelope\(\)](#)、[document.punch\(\)](#)、[document.union\(\)](#)、[shape.isDrawingObject](#)

## document.library

可用性

Flash MX 2004。

用法

```
document.library
```

#### 描述

只读属性；文档的 [library](#) 对象。

#### 示例

下面的示例获取当前获得焦点的文档的库：

```
var myCurrentLib = fl.getDocumentDOM().library;
```

假设当前获得焦点的文档不是 `fl.documents[1]`，则下面的示例为当前未获得焦点的库或使用“文件”>“打开”作为外部库打开的库获取库：

```
var externalLib = fl.documents[1].library;
```

## document.libraryPath

#### 可用性

Flash CS4 Professional.

#### 用法

```
document.libraryPath
```

#### 描述

属性；一个字符串，其中包含文档的 **ActionScript 3.0** 库路径中项目的列表，用于指定 SWC 文件或包含 SWC 文件的文件夹的位置。该字符串中的项目以分号分隔。在创作工具中，通过以下方式指定这些项目：选择“文件”>“发布设置”，然后在“Flash”选项卡上选择“ActionScript 3.0 脚本设置”。

#### 示例

下面的示例将 `../Files` 文件夹添加到文档的库路径，然后在“输出”面板中显示库路径：

```
var myDoc = fl.getDocumentDOM();  
fl.trace(myDoc.libraryPath);  
myDoc.libraryPath = "../Files;" + myDoc.libraryPath;  
fl.trace(myDoc.libraryPath);
```

#### 另请参见

[document.externalLibraryPath](#)、[document.sourcePath](#)、[fl.libraryPath](#)

## document.livePreview

#### 可用性

Flash MX 2004。

#### 用法

```
document.livePreview
```

#### 描述

属性；一个布尔值，它指定是否启用“实时预览”。如果设置为 `true`，则组件以在发布的 **Flash** 内容中出现的实际情况（包括其近似大小）显示在舞台上。如果设置为 `false`，则组件仅显示为轮廓。默认值为 `true`。

#### 示例

下面的示例将“实时预览”设置为 `false`：

```
fl.getDocumentDOM().livePreview = false;
```

## document.loadCuepointXML()

#### 可用性

Flash Professional CS5

#### 用法

```
document.loadCuepointXML(String URI)
```

#### 参数

URI 字符串；提示点 XML 文件的绝对路径。

#### 描述

方法；加载提示点 XML 文件。XML 文件的格式和 DTD 与“提示点属性”检查器导入和导出的文件相同。返回值与包含 `FLVPlayback` 组件实例的对象的提示点属性中经过序列化的字符串相同。

#### 示例

在以下示例中，提示点 XML 文件位于 `C:\testCuePoints.xml` 中：

```
var cuePoints = fl.getDocumentDOM().LoadCuepointXML("c:\testCuePoints.xml");
```

## document.match()

#### 可用性

Flash MX 2004。

#### 用法

```
document.match(bWidth, bHeight [, bUseDocumentBounds])
```

#### 参数

**bWidth** 一个布尔值，如果设置为 `true`，则此方法使所选项目的宽度相同。

**bHeight** 一个布尔值，如果设置为 `true`，则此方法使所选项目的高度相同。

**bUseDocumentBounds** 一个布尔值，如果设置为 `true`，则此方法使对象的大小与文档的范围匹配。否则，此方法使用最大对象的范围。默认值为 `false`。此参数是可选的。

#### 返回

无。

#### 描述

方法；使所选对象的大小相同。

#### 示例

下面的示例仅使所选对象的宽度匹配:

```
fl.getDocumentDOM().match(true, false);
```

下面的示例仅使高度匹配:

```
fl.getDocumentDOM().match(false, true);
```

下面的示例仅使宽度与文档的范围匹配:

```
fl.getDocumentDOM().match(true, false, true);
```

另请参见

[document.getAlignToDocument\(\)](#)、[document.setAlignToDocument\(\)](#)

## document.mouseClick()

#### 可用性

Flash MX 2004。

#### 用法

```
document.mouseClick(position, bToggleSel, bShiftSel)
```

#### 参数

**position** 一对浮点值，它们以像素为单位指定鼠标单击的 **x** 和 **y** 坐标。

**bToggleSel** 一个布尔值，它指定 **Shift** 键的状态：若按下，则为 **true**；否则为 **false**。

**bShiftSel** 一个布尔值，它指定应用程序首选的 **Shift** 选择的状态：若打开，则为 **true**；若关闭，则为 **false**。

#### 返回

无。

#### 描述

方法；从“选择”工具执行鼠标单击。

#### 示例

下面的示例在指定位置执行鼠标单击:

```
fl.getDocumentDOM().mouseClick({x:300, y:200}, false, false);
```

另请参见

[document.mouseDbClick\(\)](#)

## document.mouseDbClick()

#### 可用性

Flash MX 2004。

#### 用法

```
document.mouseDblClk(position, bAltDown, bShiftDown, bShiftSelect)
```

#### 参数

**position** 一对浮点值，它们以像素为单位指定鼠标单击的 **x** 和 **y** 坐标。

**bAltdown** 一个布尔值，它记录事件发生时是否按下了 **Alt** 键：若按下，则为 **true**；否则为 **false**。

**bShiftDown** 一个布尔值，它记录事件发生时是否按下了 **Shift** 键：若按下，则为 **true**；否则为 **false**。

**bShiftSelect** 一个布尔值，它指示应用程序首选的 **Shift** 选择的状态：若打开，则为 **true**；若关闭，则为 **false**。

#### 返回

无。

#### 描述

方法；从“选择”工具执行鼠标双击。

#### 示例

下面的示例在指定位置执行鼠标双击：

```
fl.getDocumentDOM().mouseDblClk({x:392.9, y:73}, false, false, true);
```

另请参见

[document.mouseClick\(\)](#)

## document.moveSelectedBezierPointsBy()

#### 可用性

Flash MX 2004。

#### 用法

```
document.moveSelectedBezierPointsBy(delta)
```

#### 参数

**delta** 一对浮点值，它以所选贝塞尔曲线点移动的像素数来指定 **x** 和 **y** 坐标。例如，如果传递的是 **{x:1,y:2}**，则指定当前位置向右 1 像素、向下 2 像素的位置。

#### 返回

无。

#### 描述

方法；如果所选内容至少包含一个路径，并且这些路径都至少选择了一个贝塞尔曲线点，则按照指定的量移动全部所选路径上的全部所选贝塞尔曲线点。

#### 示例

下面的示例将所选贝塞尔曲线点向右移动 10 个像素，向下移动 5 个像素：

```
fl.getDocumentDOM().moveSelectedBezierPointsBy({x:10, y:5});
```

## document.moveSelectionBy()

可用性

Flash MX 2004。

用法

```
document.moveSelectionBy(distanceToMove)
```

参数

**distanceToMove** 一对浮点值，它们以此方法移动所选内容的像素数来指定 x 和 y 坐标值。例如，如果传递的是 ({x:1,y:2})，则指定当前位置向右 1 像素和向下 2 像素的位置。

返回

无。

描述

方法；按照指定的距离移动所选对象。

注：用户使用箭头键移动项目时，“历史记录”面板将箭头键的所有按键操作合并为一个移动步骤。如果用户重复按箭头键而不是在“历史记录”面板中采取多个步骤，则此方法执行一个步骤，参数也会更新，以便反映重复按下的箭头键。

有关进行选择的信息，请参阅 [document.setSelectionRect\(\)](#)、[document.mouseClick\(\)](#)、[document.mouseDbIClk\(\)](#) 和 [Element 对象](#)。

示例

下面的示例将所选项目向右移动 62 像素，向下移动 84 像素：

```
fl.getDocumentDOM().moveSelectionBy({x:62, y:84});
```

## document.name

可用性

Flash MX 2004。

用法

```
document.name
```

描述

只读属性；一个字符串，它表示文档（FLA 文件）名称。

示例

下面的示例将变量 **fileName** 设置为文档数组中第一个文档的文件名：

```
var fileName = flash.documents[0].name;
```

下面的示例在“输出”面板中显示所有打开的文档的名称：

```
var openDocs = fl.documents;  
for(var i=0;i < openDocs.length; i++){  
    fl.trace(i + " " + openDocs[i].name + "\n");  
}
```

## document.optimizeCurves()

可用性

Flash MX 2004。

用法

```
document.optimizeCurves(smoothing, bUseMultiplePasses)
```

参数

**smoothing** 一个整数，范围从 0 到 100，其中 0 指定没有平滑，100 指定最大平滑。

**bUseMultiplePasses** 一个布尔值，如果设置为 true，则指示方法应使用多重过渡，这样速度会较慢，但产生的效果较好。使用此参数与在“优化曲线”对话框中单击“使用多重过渡”按钮效果相同。

返回

无。

描述

方法；优化当前所选内容的平滑，允许多重过渡（如果指定）以实现最佳平滑。此方法等效于选择“修改”>“形状”>“优化”。

示例

下面的示例通过多重过渡将当前所选内容的曲线优化为 50° 平滑：

```
fl.getDocumentDOM().optimizeCurves(50, true);
```

## document.path

可用性

Flash MX 2004。

用法

```
document.path
```

描述

只读属性；一个字符串，它表示文档的路径（采用特定于平台的格式）。如果文档尚未保存过，则此属性为 **undefined**。

示例

下面的示例在“输出”面板中显示文档数组中第一个文档的路径。在运行此脚本之前，必须首先保存该文档。在该示例中，该文件取名 **test.fla** 并保存在 Windows 计算机的“我的文档”文件夹中。

```
var filePath = flash.documents[0].path;  
fl.trace(filePath);  
// displays C:\Documents and Settings\\My Documents\test.fla
```

另请参见

[document.pathURI](#)



## document.pathURI

可用性

Flash CS4 Professional.

用法

```
document.pathURI
```

描述

只读属性；一个字符串，表示文档的路径，表示为 **file:/// URI**。如果文档尚未保存过，则此属性为 **undefined**。

示例

下面的示例在“输出”面板中将文档数组中第一个文档的路径显示为 **file:/// URI** 字符串。在运行此脚本之前，必须首先保存该文档。在该示例中，该文件取名 **test.fla** 并保存在 Windows 计算机的“我的文档”文件夹中。

```
var filePathURI = flash.documents[0].pathURI;  
fl.trace(filePathURI);  
// displays file:///C:/Documents%20and%20Settings/<userName>/My%20Documents/test.fla
```

另请参见

[document.path](#)

## document.publish()

可用性

Flash MX 2004。

用法

```
document.publish()
```

参数

无。

返回

无。

描述

方法；按照活动发布设置（“文件”>“发布设置”）发布文档。此方法等效于选择“文件”>“发布”。

示例

下面的示例发布当前文档：

```
fl.getDocumentDOM().publish();
```

## document.publishProfiles

可用性

Flash MX 2004。

用法

```
document.publishProfiles
```

描述

只读属性；文档的发布配置文件名称的数组。

示例

下面的示例显示文档的发布配置文件的名称：

```
var myPubProfiles = fl.getDocumentDOM().publishProfiles;
for (var i=0; i < myPubProfiles.length; i++){
    fl.trace(myPubProfiles[i]);
}
```

## document.punch()

可用性

Flash 8。

用法

```
document.punch()
```

参数

无。

返回

一个布尔值：如果成功，则为 **true**；否则为 **false**。

说明

方法；使用最上层的所选绘制对象来为其下的所有选中绘制对象打孔。如果未选择任何绘制对象，或者所选任何项都不是绘制对象，则此方法返回 **false**。

示例

下面的示例对所选绘制对象下面的绘制对象打孔：

```
fl.getDocumentDOM().punch();
```

另请参见

[document.crop\(\)](#)、[document.deleteEnvelope\(\)](#)、[document.intersect\(\)](#)、[document.union\(\)](#)、[shape.isDrawingObject](#)

## document.removeAllFilters()

可用性

Flash 8。

用法

```
document.removeAllFilters()
```

参数

无。

返回

无。

描述

方法；从所选对象中删除所有滤镜。

示例

下面的示例从所选对象删除所有滤镜：

```
fl.getDocumentDOM().removeAllFilters();
```

另请参见

[document.addFilter\(\)](#)、[document.changeFilterOrder\(\)](#)、[document.disableAllFilters\(\)](#)、[document.getFilters\(\)](#)、[document.removeFilter\(\)](#)、[Filter 对象](#)

## document.removeDataFromDocument()

可用性

Flash MX 2004。

用法

```
document.removeDataFromDocument (name)
```

参数

**name** 一个字符串，它指定要删除的数据的名称。

返回

无。

描述

方法；删除附加到文档、具有指定名称的永久数据。

示例

下面的示例从文档中删除名为 "myData" 的永久性数据：

```
fl.getDocumentDOM().removeDataFromDocument ("myData");
```

另请参见

[document.addDataToDocument\(\)](#)、[document.documentHasData\(\)](#)、[document.getDataFromDocument\(\)](#)

## document.removeDataFromSelection()

可用性

Flash MX 2004。

用法

```
document.removeDataFromSelection(name)
```

参数

**name** 一个字符串，它指定要删除的永久性数据的名称。

返回

无。

描述

方法；删除附加到所选内容、具有指定名称的永久数据。

示例

下面的示例从所选内容中删除名为 "myData" 的永久性数据：

```
fl.getDocumentDOM().removeDataFromSelection("myData");
```

另请参见

[document.addDataToSelection\(\)](#)

## document.removeFilter()

可用性

Flash 8。

用法

```
document.removeFilter(filterIndex)
```

参数

**filterIndex** 一个整数，它指定要从所选对象删除的滤镜的索引（从零开始）。

返回

无。

描述

方法；从所选对象的滤镜列表中删除指定的滤镜。

## 示例

下面的示例从所选对象的“滤镜”列表中删除第一个滤镜（索引值为 0）：

```
fl.getDocumentDOM().removeFilter(0);
```

## 另请参见

[document.addFilter\(\)](#)、[document.changeFilterOrder\(\)](#)、[document.disableFilter\(\)](#)、[document.getFilters\(\)](#)、[document.removeAllFilters\(\)](#)、[Filter 对象](#)

## document.renamePublishProfile()

## 可用性

Flash MX 2004。

## 用法

```
document.renamePublishProfile([profileNewName])
```

## 参数

**profileNewName** 一个可选参数，它指定配置文件的新名称。新名称必须是唯一的。如果不指定名称，则提供一个默认名称。

## 返回

布尔值：如果名称更改成功，则为 **true**；否则为 **false**。

## 说明

方法；重命名当前配置文件。

## 示例

下面的示例将当前配置文件重命名为一个默认的名称，并显示该名称：

```
alert(fl.getDocumentDOM().renamePublishProfile());
```

## document.renameScene()

## 可用性

Flash MX 2004。

## 用法

```
document.renameScene(name)
```

## 参数

**name** 一个字符串，它指定场景的新名称。

## 返回

布尔值：如果名称更改成功，则为 **true**；否则为 **false**。例如，如果新名称不是唯一的，则此方法返回 **false**。

## 说明

方法；在“场景”面板中重命名当前所选场景。所选场景的新名称必须是唯一的。

#### 示例

下面的示例将当前场景重命名为 "new name":

```
var success = fl.getDocumentDOM().renameScene("new name");
```

## document.reorderScene()

#### 可用性

Flash MX 2004。

#### 用法

```
document.reorderScene(sceneToMove, sceneToPutItBefore)
```

#### 参数

**sceneToMove** 一个整数，它指定要移动的场景，其中 0（零）为第一个场景。

**sceneToPutItBefore** 一个整数，它指定要将由 **sceneToMove** 指定的场景移动到其前面的场景。0（零）为第一个场景。例如，如果将 **sceneToMove** 指定为 1，将 **sceneToPutItBefore** 指定为 0，则表示将第二个场景放在第一个场景前面。指定 -1，场景则移动到最后。

#### 返回

无。

#### 描述

方法；将指定场景移动到另一个指定场景前面。

#### 示例

下面的示例将第二个场景移动到第一个场景前面：

```
fl.getDocumentDOM().reorderScene(1, 0);
```

## document.resetOvalObject()

#### 可用性

Flash CS3 Professional。

#### 用法

```
document.resetOvalObject()
```

#### 参数

无。

#### 返回

无。

#### 描述

方法；将“属性”检查器中的所有值设置为默认的 **Oval** 对象设置。如果选择了任何 **Oval** 对象，则它们的属性也会被重置为默认值。

#### 示例

下面的示例将当前文档中的 **Oval** 对象属性重置为默认值：

```
fl.getDocumentDOM().resetOvalObject();
```

另请参见

[document.resetRectangleObject\(\)](#)

## document.resetRectangleObject()

#### 可用性

Flash CS3 Professional。

#### 用法

```
document.resetRectangleObject()
```

#### 参数

无。

#### 返回

无。

#### 描述

方法；将“属性”检查器中的所有值设置为默认的 **Rectangle** 对象设置。如果选择了任何 **Rectangle** 对象，则它们的属性也会被重置为默认值。

#### 示例

下面的示例将当前文档中的 **Rectangle** 对象属性重置为默认值：

```
fl.getDocumentDOM().resetRectangleObject();
```

另请参见

[document.resetOvalObject\(\)](#)

## document.resetTransformation()

#### 可用性

Flash MX 2004。

#### 用法

```
document.resetTransformation()
```

#### 参数

无。

#### 返回

无。

#### 描述

方法；重置变形矩阵。此方法等效于选择“修改”>“变形”>“删除变形”。

#### 示例

下面的示例重置当前所选内容的变形矩阵：

```
fl.getDocumentDOM().resetTransformation();
```

## document.revert()

#### 可用性

Flash MX 2004。

#### 用法

```
document.revert()
```

#### 参数

无。

#### 返回

无。

#### 描述

方法；使指定文档回复到以前保存的版本。此方法等效于选择“文件”>“回复”。

#### 示例

下面的示例使当前文档回复到以前保存的版本：

```
fl.getDocumentDOM().revert();
```

#### 另请参见

[document.canRevert\(\)](#)、[fl.revertDocument\(\)](#)

## document.rotate3DSelection()

#### 可用性

Flash CS4 Professional.

#### 用法

```
document.rotate3DSelection(xyzCoordinate, bGlobalTransform)
```



#### 参数

**xyzCoordinate** 一个 XYZ 坐标点，它指定 3D 旋转的轴。

**bGlobalTransform** 一个布尔值，它指定变形模式应是全局模式 (**true**) 还是本地模式 (**false**)。

#### 返回

无。

#### 描述

方法：对所选内容应用 3D 旋转。此方法仅适用于影片剪辑。

#### 示例

在下面的示例中，首先相对于舞台旋转所选内容（全局），然后相对于自身旋转所选内容（局部）。

```
var myDocument = fl.getDocumentDOM();  
myDocument.rotate3DSelection({x:52.0, y:0, z:0}, true);  
myDocument.rotate3DSelection({x:52.0, y:0, z:-55.2}, false);
```

## document.rotateSelection()

#### 可用性

Flash MX 2004。

#### 用法

```
document.rotateSelection(angle [, rotationPoint])
```

#### 参数

**angle** 一个浮点值，它指定旋转的角度。

**rotationPoint** 一个字符串，它指定要旋转边框的哪个边。可接受值为 "top right"、"top left"、"bottom right"、"bottom left"、"top center"、"right center"、"bottom center" 和 "left center"。如果未指定，则此方法使用变形点。此参数是可选的。

#### 返回

无。

#### 描述

方法；按照指定度数旋转所选内容。效果与使用“任意变形”工具旋转对象相同。

#### 示例

下面的示例将所选内容绕变形点旋转 45°：

```
fl.getDocumentDOM().rotateSelection(45);
```

下面的示例将所选内容绕左下角旋转 45°：

```
fl.getDocumentDOM().rotateSelection(45, "bottom left");
```

## document.save()

可用性

Flash MX 2004。

用法

```
document.save ([bOkToSaveAs])
```

参数

**bOkToSaveAs** 一个可选参数，如果该参数为 **true** 或被省略，并且文件从未保存过，则打开“另存为”对话框。如果为 **false**，且文件从未保存过，则不保存文件。

返回

布尔值：如果保存操作成功完成，则返回 **true**；否则返回 **false**。

说明

方法；将文档保存在其默认位置。此方法等效于选择“文件”>“保存”。

若要指定文件的名称（而非用同一名称保存），请使用 [fl.saveDocument\(\)](#)。

注：如果文件为新建文件并且尚未修改或保存过，或者自上次保存后再未进行过修改，则此方法无效并返回 **false**。若要允许保存未保存过的文件或未修改过的文件，请使用 [document.saveAndCompact\(\)](#) 或 [fl.saveDocumentAs\(\)](#)。

示例

下面的示例将当前文档保存在其默认位置：

```
fl.getDocumentDOM().save();
```

另请参见

[document.saveAndCompact\(\)](#)、[fl.saveAll\(\)](#)、[fl.saveDocument\(\)](#)、[fl.saveDocumentAs\(\)](#)

## document.saveAndCompact()

可用性

Flash MX 2004。

用法

```
document.saveAndCompact ([bOkToSaveAs])
```

参数

**bOkToSaveAs** 一个可选参数，如果该参数为 **true** 或被省略，并且文件从未保存过，则打开“另存为”对话框。如果为 **false**，且文件从未保存过，则不保存文件。默认值为 **true**。

返回

布尔值：如果“保存并压缩”操作成功完成，则为 **true**；否则为 **false**。

说明

方法；保存并压缩文件。此方法等效于选择“文件”>“保存并压缩”。

注：如果此文件从未保存过，即使用户取消了“另存为”对话框，该方法也会返回 `true`。若要准确地确定文件是否保存过，请使用 `fl.saveDocumentAs()`。

#### 示例

下面的示例保存并压缩当前文档：

```
fl.getDocumentDOM().saveAndCompact();
```

另请参见

[document.save\(\)](#)、[fl.saveDocumentAs\(\)](#)、[fl.saveDocument\(\)](#)、[fl.saveAll\(\)](#)

## document.scaleSelection()

#### 可用性

Flash MX 2004。

#### 用法

```
document.scaleSelection(xScale, yScale [, whichCorner])
```

#### 参数

**xScale** 一个浮点值，它指定 `x` 的缩放量。

**yScale** 一个浮点值，它指定 `y` 的缩放量。

**whichCorner** 一个字符串值，它指定附近要执行变形的边缘。如果省略，则变形点附近进行缩放。可接受值为：“bottom left”、“bottom right”、“top right”、“top left”、“top center”、“right center”、“bottom center”和“left center”。此参数是可选的。

#### 返回

无。

#### 描述

方法；按照指定的量缩放所选内容。此方法等效于使用“任意变形”工具缩放对象。

#### 示例

下面的示例将当前所选内容的宽度扩展为原始宽度的两倍，并将高度缩小一半：

```
fl.getDocumentDOM().scaleSelection(2.0, 0.5);
```

下面的示例垂直翻转所选内容：

```
fl.getDocumentDOM().scaleSelection(1, -1);
```

下面的示例水平翻转所选内容：

```
fl.getDocumentDOM().scaleSelection(-1, 1);
```

下面的示例从顶部中心垂直缩放所选内容 1.9 倍：

```
fl.getDocumentDOM().scaleSelection(1, 1.90, 'top center');
```

## document.screenOutline

可用性

Flash MX 2004。

用法

```
document.screenOutline
```

描述

只读属性；文档的当前 **ScreenOutline** 对象。初次访问对象之前，请务必使用 `document.allowScreens()` 确定此属性是否存在。

示例

下面的示例显示 `screenOutline` 属性中的值组成的数组：

```
var myArray = new Array();
for(var i in fl.getDocumentDOM().screenOutline) {
    myArray.push(" "+i+" : "+fl.getDocumentDOM().screenOutline[i] );
}
fl.trace("Here is the property dump for screenOutline: "+myArray);
```

另请参见

[document.allowScreens\(\)](#)

## document.selectAll()

可用性

Flash MX 2004。

用法

```
document.selectAll()
```

参数

无。

返回

无。

描述

方法；选择舞台上的所有项目。此方法等效于按 **Control+A** (Windows) 或 **Command+A** (Macintosh)，也等效于选择“编辑”>“全选”。

示例

下面的示例选择用户当前可见的所有内容：

```
fl.getDocumentDOM().selectAll();
```

另请参见

[document.selection](#)、[document.selectNone\(\)](#)

# document.selection

可用性

Flash MX 2004。

用法

`document.selection`

描述

属性；文档中的所选对象的数组。如果未选择任何内容，则返回一个长度为零的数组。如果未打开任何文档，则返回 `null`。

要将对象添加到数组，您必须首先使用下列方法之一选择对象：

- 手动选择舞台上的对象。
- 使用选择方法之一，如 `document.setSelectionRect()`、`document.setSelectionBounds()`、`document.mouseClick()`、`document.mouseDbcClk()` 或 `document.selectAll()`。
- 手动选择一个或多个帧。
- 使用 **Timeline 对象**的方法之一选择一个或多个帧，如 `timeline.getSelectedFrames()`、`timeline.setSelectedFrames()` 或 `timeline.selectAllFrames()`。
- 指定特定帧中的所有元素（请参阅 **Element 对象**）。请参阅下面的第一个示例。
- 创建一个或多个元素的数组，然后将该数组赋值给 `document.selection` 数组。请参阅下面的第三个示例。

示例

下面的示例将第 11 帧上的所有元素都指定给当前所选内容（请记住，索引值不同于帧号值）：

```
f1.getDocumentDOM().getTimeline().currentFrame = 10;  
f1.getDocumentDOM().selection = f1.getDocumentDOM().getTimeline().layers[0].frames[10].elements;
```

下面的示例在舞台左上角创建一个矩形，在该矩形下面创建一个文本字符串。然后该示例使用 `document.setSelectionRect()` 选择这两个对象，并将它们添加到 `document.selection` 数组。最后，在“输出”面板中显示 `document.selection` 的内容。

```
f1.getDocumentDOM().addNewRectangle({left:0, top:0, right:99, bottom:99}, 0);  
f1.getDocumentDOM().addNewText({left:-1, top:117.3, right:9.2, bottom:134.6});  
f1.getDocumentDOM().setTextString('Hello World');  
f1.getDocumentDOM().setSelectionRect({left:-28, top:-22, right:156.0, bottom:163});
```

```
var theSelectionArray = f1.getDocumentDOM().selection;
```

```
for (var i=0;i<theSelectionArray.length;i++) {  
f1.trace("f1.getDocumentDOM().selection["+i+"] = " + theSelectionArray[i]);  
}
```

下面的示例是一个高级示例。它显示如何遍历图层数组和元素数组来定位并选择特定元件的实例。您可以扩展此示例，以便包含多帧或多场景的循环。此示例将第一帧中的影片剪辑 `myMovieClip` 的所有实例都指定给当前所选内容：

```
// Assigns the layers array to the variable "theLayers".
var theLayers = fl.getDocumentDOM().getTimeline().layers;
// Creates an array to hold all the elements
// that are instances of "myMovieClip".
var myArray = new Array();
// Counter variable
var x = 0;
// Begin loop through all the layers.
for (var i = 0; i < theLayers.length; i++) {
    // Gets the array of elements in Frame 1
    // and assigns it to the array "theElems".
    var theElems = theLayers[i].frames[0].elements;
    // Begin loop through the elements on a layer.
    for (var c = 0; c < theElems.length; c++) {
        // Checks to see if the element is of type "instance".
        if (theElems[c].elementType == "instance") {
            // If the element is an instance, it checks
            // if it is an instance of "myMovieClip".
            if (theElems[c].libraryItem.name == "myMovieClip") {
                // Assigns elements that are instances of "myMovieClip" to "myArray".
                myArray[x] = theElems[c];
                // Increments counter variable.
                x++;
            }
        }
    }
}
// Now that you have assigned all the instances of "myMovieClip"
// to "myArray", you then set the document.selection array
// equal to myArray. This selects the objects on the Stage.
fl.getDocumentDOM().selection = myArray;
```

## document.selectNone()

可用性

Flash MX 2004。

用法

```
document.selectNone()
```

参数

无。

返回

无。

描述

方法；取消选择全部被选定的项目。

示例

下面的示例取消选择全部被选定的项目：

```
fl.getDocumentDOM().selectNone();
```

另请参见

[document.selectAll\(\)](#)、[document.selection](#)

## document.setAlignToDocument()

可用性

Flash MX 2004。

用法

```
document.setAlignToDocument(bToStage)
```

参数

**bToStage** 一个布尔值，如果设置为 **true**，则使对象与舞台对齐。如果设置为 **false**，则不使对象与舞台对齐。

返回

无。

描述

方法；将 [document.align\(\)](#)、[document.distribute\(\)](#)、[document.match\(\)](#) 和 [document.space\(\)](#) 的首选参数设置为针对文档进行操作。此方法等效于在“对齐”面板中启用“相对于舞台”按钮。

示例

下面的示例在“对齐”面板中启用“相对于舞台”按钮，以便使对象与舞台对齐：

```
fl.getDocumentDOM().setAlignToDocument(true);
```

另请参见

[document.getAlignToDocument\(\)](#)

## document.setBlendMode()

可用性

Flash 8。

用法

```
document.setBlendMode(mode)
```

参数

**mode** 一个字符串，它表示所选对象所需的混合模式。可接受值为 "normal"、"layer"、"multiply"、"screen"、"overlay"、"hardlight"、"lighten"、"darken"、"difference"、"add"、"subtract"、"invert"、"alpha" 和 "erase"。

返回

无。

描述

方法；设置所选对象的混合模式。

#### 示例

下面的示例将所选对象的混合模式设置为 "add"。

```
fl.getDocumentDOM().setBlendMode("add");
```

另请参见

[document.addFilter\(\)](#)、[document.setFilterProperty\(\)](#)、[symbolInstance.blendMode](#)

## document.setCustomFill()

#### 可用性

Flash MX 2004。

#### 用法

```
document.setCustomFill(fill)
```

#### 参数

**填充 (fill)** 一个 Fill 对象，它指定要使用的填充设置。请参阅 [Fill 对象](#)。

#### 返回

无。

#### 描述

方法；设置“工具”面板、“属性”检查器和全部所选形状的填充设置。此方法允许脚本在绘制对象之前设置填充设置，而不是绘制对象、选择对象、然后更改填充设置。它还允许脚本更改“工具”面板和“属性”检查器的填充设置。

#### 示例

下面的示例将“工具”面板、“属性”检查器和全部所选形状中的填充颜色样本的颜色都更改为白色：

```
var fill = fl.getDocumentDOM().getCustomFill();  
fill.color = '#FFFFFF';  
fill.style = "solid";  
fl.getDocumentDOM().setCustomFill(fill);
```

另请参见

[document.getCustomFill\(\)](#)

## document.setCustomStroke()

#### 可用性

Flash MX 2004。

#### 用法

```
document.setCustomStroke(stroke)
```

#### 参数

**笔触 (stroke)** 一个 [Stroke 对象](#)。



返回  
无。

#### 描述

方法；设置“工具”面板、“属性”检查器和全部所选形状的笔触设置。此方法允许脚本在绘制对象之前设置笔触设置，而不是绘制对象、选择对象、然后更改笔触设置。它还允许脚本更改“工具”面板和“属性”检查器的笔触设置。

#### 示例

下面的示例更改“工具”面板、“属性”检查器和全部所选形状中的笔触粗细设置：

```
var stroke = fl.getDocumentDOM().getCustomStroke();  
stroke.thickness += 2;  
fl.getDocumentDOM().setCustomStroke(stroke);
```

另请参见

[document.getCustomStroke\(\)](#)

## document.setElementProperty()

可用性

Flash MX 2004。

#### 用法

```
document.setElementProperty(property, value)
```

#### 参数

**属性** 一个字符串，它指定要设置的 **Element** 属性的名称。有关属性和值的完整列表，请参阅 [Element 对象](#) 的属性摘要表。

此方法不能用于设置只读属性的值，如 [element.elementType](#)、[element.top](#) 或 [element.left](#)。

**value** 一个整数，它指定要在指定的 **Element** 属性中设置的值。

返回  
无。

#### 描述

方法；在文档中对所选对象设置指定的 **Element** 属性。如果未选择任何内容，则此方法不执行任何操作。

#### 示例

下面的示例将所选全部对象的宽度都设置为 100，高度设置为 50：

```
fl.getDocumentDOM().setElementProperty("width", 100);  
fl.getDocumentDOM().setElementProperty("height", 50);
```

## document.setElementTextAttr()

可用性

Flash MX 2004。

#### 用法

```
document.setElementTextAttr(attrName, attrValue [, startIndex [, endIndex]])
```

#### 参数

**attrName** 一个字符串，它指定要更改的 `TextAttrs` 属性的名称。

**attrValue** 要为 `TextAttrs` 属性设置的值。有关属性名称和预期值的列表，请参阅 [TextAttrs 对象](#) 的属性摘要表。

**startIndex** 一个整数值，它指定受影响的第一个字符的索引。此参数是可选的。

**endIndex** 一个整数值，它指定受影响的最后一个字符的索引。此参数是可选的。

#### 返回

布尔值：如果至少更改了一个文本特性属性，则为 `true`；否则为 `false`。

#### 说明

方法；将所选文本项的指定 `textAttrs` 属性设置为指定值。有关属性名称和允许值的列表，请参阅 [TextAttrs 对象](#) 的属性摘要表。如果没有传递可选参数，则此方法设置当前所选文本范围的样式，如果未选择任何文本，则设置整个文本字段的样式。如果只传递 `startIndex`，则此方法设置该字符的属性。如果传递了 `startIndex` 和 `endIndex`，则此方法设置从 `startIndex` 到（但不包括）`endIndex` 的字符的属性。如果指定了段落样式，则此范围内的所有段落都会受影响。

#### 示例

下面的示例设置所选文本项的 `fillColor`、`italic` 和 `bold` 文本属性：

```
var success = fl.getDocumentDOM().setElementTextAttr("fillColor", "#00ff00");  
var pass = fl.getDocumentDOM().setElementTextAttr("italic", true, 10);  
var ok = fl.getDocumentDOM().setElementTextAttr("bold", true, 5, 15);
```

## document.setFillColor()

#### 可用性

Flash MX 2004。

#### 用法

```
document.setFillColor(color)
```

#### 参数

**color** 填充的颜色，使用以下格式之一：

- 格式为 `"#RRGGBB"` 或 `"#RRGGBBAA"` 的字符串
- 格式为 `0xRRGGBB` 的十六进制数字
- 表示与十六进制数字等价的十进制整数

如果设置为 `null`，则不设置任何填充颜色，这与将用户界面中的填充颜色样本设置为无填充相同。

#### 返回

无。

#### 描述

方法；将所选内容的填充颜色更改为指定的颜色。有关更改“工具”面板和“属性”检查器中的填充颜色的信息，请参阅 [document.setCustomFill\(\)](#)。

## 示例

下面的示例中的前三个语句使用指定颜色的每个不同格式设置填充颜色。第四个语句将填充设置为无填充。

```
fl.getDocumentDOM().setFillColor("#cc00cc");
fl.getDocumentDOM().setFillColor(0xcc00cc);
fl.getDocumentDOM().setFillColor(120000);
fl.getDocumentDOM().setFillColor(null);
```

## document.setFilterProperty()

## 可用性

Flash 8。

## 用法

```
document.setFilterProperty(property, filterIndex, value)
```

## 参数

**属性** 一个字符串，它指定要设置的属性。可接受值为 "blurX"、"blurY"、"quality"、"angle"、"distance"、"strength"、"knockout"、"inner"、"bevelType"、"color"、"shadowColor" 和 "highlightColor"。

**filterIndex** 一个整数，它指定滤镜列表中滤镜的索引（从零开始）。

**value** 一个数字或字符串，它指定要为指定的滤镜属性设置的值。可接受的值取决于属性和设置的滤镜。

## 返回

无。

## 描述

方法；为当前所选对象设置指定的滤镜属性（假定对象支持指定的滤镜）。

## 示例

下面的示例将所选对象“滤镜”列表中第二个滤镜（索引值为 1）的 **quality** 属性设置为 2，然后针对所选对象设置“滤镜”列表中第一个滤镜的 **shadowColor** 属性：

```
fl.getDocumentDOM().setFilterProperty("quality", 1, 2);
fl.getDocumentDOM().setFilterProperty("shadowColor", 0, "#FF00FF");
```

## 另请参见

[document.addFilter\(\)](#)、[document.getFilters\(\)](#)、[document.setBlendMode\(\)](#)、[document.setFilters\(\)](#)、[Filter](#) 对象

## document.setFilters()

## 可用性

Flash 8。

## 用法

```
document.setFilters(filterArray)
```

#### 参数

**filterArray** 由当前指定的滤镜组成的数组。

#### 返回

无。

#### 描述

方法；将滤镜应用于所选对象。在调用 `document.getFilters()` 并对滤镜进行任何所需的更改之后，再使用此方法。

#### 示例

下面的示例获取所选对象的滤镜，并将所有“模糊”滤镜的 `blurX` 属性设置为 50：

```
var myFilters = fl.getDocumentDOM().getFilters();
for (i=0; i < myFilters.length; i++) {
    if (myFilters[i].name == "blurFilter"){
        myFilters[i].blurX = 50;
    }
}
fl.getDocumentDOM().setFilters(myFilters);
```

另请参见

[document.addFilter\(\)](#)、[document.getFilters\(\)](#)、[document.setFilterProperty\(\)](#)、[Filter 对象](#)

## document.setInstanceAlpha()

#### 可用性

Flash MX 2004。

#### 用法

```
document.setInstanceAlpha(opacity)
```

#### 参数

**opacity** 一个介于 0（透明）和 100（完全饱和）之间的整数，它调整实例的透明度。

#### 返回

无。

#### 描述

方法；设置实例的不透明度。

#### 示例

下面的示例将色调的不透明度值设置为 50：

```
fl.getDocumentDOM().setInstanceAlpha(50);
```

## document.setInstanceBrightness()

可用性

Flash MX 2004。

用法

```
document.setInstanceBrightness(brightness)
```

参数

**亮度** 一个整数，它将亮度指定为 -100（黑色）到 100（白色）之间的一个值。

返回

无。

描述

方法；设置实例的亮度。

示例

下面的示例将实例的亮度值设置为 50：

```
fl.getDocumentDOM().setInstanceBrightness(50);
```

## document.setInstanceTint()

可用性

Flash MX 2004。

用法

```
document.setInstanceTint( color, strength )
```

参数

**color** 色调的颜色，使用以下格式之一：

- 格式为 "#RRGGBB" 或 "#RRGGBBAA" 的字符串
- 格式为 0xRRGGBB 的十六进制数字
- 表示与十六进制数字等价的十进制整数

**strength** 一个介于 0 到 100 之间的整数，它指定色调的不透明度。

返回

无。

描述

方法；设置实例的色调。

示例

下面的示例将所选实例的色调设置为红色，且不透明度值为 50：

```
fl.getDocumentDOM().setInstanceTint(0xff0000, 50);
```

## document.setMetadata()

可用性

Flash 8。

用法

```
document.setMetadata(strMetadata)
```

参数

**strMetadata** 一个字符串，它包含要与文档关联的 XML 元数据。有关详细信息，请参阅下面的说明。

返回

一个布尔值：如果成功，则为 **true**；否则为 **false**。

说明

方法；设置指定文档的 XML 元数据，覆盖全部现有元数据。传递为 **strMetadata** 的 XML 经过验证，在存储之前都可以进行重写。如果不能验证为合法的 XML 或者违反特定规则，则不设置 XML 元数据，并返回 **false**。（如果返回 **false**，则无法获取详细错误信息。）

注：即使返回 **true**，所设置的 XML 与传入的字符串也有可能并不完全相同。若要获取所设置的 XML 精确值，请使用 [document.getMetadata\(\)](#)。

元数据的格式为 RDF，该格式符合 XMP 规范。有关 RDF 和 XMP 的更多信息，请参阅下列来源：

- 位于 [www.w3.org/TR/rdf-primer/](http://www.w3.org/TR/rdf-primer/) 的 RDF Primer
- 位于 [www.w3.org/TR/1999/REC-rdf-syntax-19990222/](http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/) 的 RDF 规范
- 位于 [www.adobe.com/products/xmp/](http://www.adobe.com/products/xmp/) 的 XMP 主页

示例

下面的示例演示了几种不同的合法方式来表示相同数据。在所有这些方式中（第二种方式除外），如果数据被发送到 `Document.setMetadata()`，则除了删除分行符以外，数据不会进行重写。

在第一个示例中，元数据位于标签内，并且不同的架构被放到不同的 `rdf:Description` 标签中：

```
<rdf:RDF xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
<rdf:Description rdf:about='' xmlns:dc='http://purl.org/dc/1.1/'>
<dc:title>Simple title</dc:title>
<dc:description>Simple description</dc:description>
</rdf:Description>
<rdf:Description rdf:about='' xmlns:xmp='http://ns.adobe.com/xap/1.0/'>
<xmp:CreateDate>2004-10-12T10:29:07:00</xmp:CreateDate>
<xmp:CreatorTool>Flash Authoring WIN 8,0,0,215</xmp:CreatorTool>
</rdf:Description>
</rdf:RDF>
```

在第二个示例中，元数据位于标签内，但不同的架构全部位于同一个 `rdf:Description` 标签中。该示例还包含注释，这些注释将被 `Document.setMetadata()` 忽略和丢弃：

```
<rdf:RDF xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
  <!-- This is before the first rdf:Description tag -->
  <rdf:Description rdf:about='' xmlns:dc='http://purl.org/dc/1.1/'>
  <dc:title>Simple title</dc:title>
  <dc:description>Simple description</dc:description>
  </rdf:Description>
  <!-- This is between the two rdf:Description tags -->
  <rdf:Description rdf:about='' xmlns:xmp='http://ns.adobe.com/xap/1.0/'>
  <xmp:CreateDate>2004-10-12T10:29-07:00</xmp:CreateDate>
  <xmp:CreatorTool>Flash Authoring WIN 8,0,0,215</xmp:CreatorTool>
  </rdf:Description>
  <!-- This is after the second rdf:Description tag -->
</rdf:RDF>
```

在第三个示例中，元数据位于属性中，并且不同的架构全部位于同一个 `rdf:Description` 标签中：

```
<rdf:RDF xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
<rdf:Description rdf:about='' xmlns:dc='http://purl.org/dc/1.1/' dc:title='Simple title'
dc:description='Simple description' />
<rdf:Description rdf:about='' xmlns:xmp='http://ns.adobe.com/xap/1.0/'
xmp:CreateDate='2004-10-12T10:29-07:00' xmp:CreatorTool='Flash Authoring WIN 8,0,0,215' />
</rdf:RDF>
```

另请参见

[document.getMetadata\(\)](#)

## document.setMobileSettings()

可用性

Flash CS3 Professional。

用法

`document.setMobileSettings(xmlString)`

参数

**xmlString** 一个字符串，它说明在移动 FLA 文件中的 XML 设置。

返回

如果该设置已成功设置，则值为 `true`；否则为 `false`。

说明

方法；设置移动 FLA 文件中 XML 设置字符串的值。（大多数移动 FLA 文件都具有一个用于说明文档内设置的 XML 字符串。）

示例

下面的示例设置移动 FLA 文件的 XML 设置字符串。请注意，下面的示例表示单个代码行。

```
fl.getDocumentDOM().setMobileSettings("<? xml version='1.0' encoding='UTF-16' standalone='no' ?>
<mobileSettings> <contentType id='standalonePlayer' name='Standalone Player'/> <testDevice
id='1170' name='Generic Phone' selected='yes'/> </testDevices> <outputMsgFiltering info='no' trace='yes'
warning='yes'/> <testWindowState height='496' splitterClosed='No' splitterXPos='400' width='907'/>
</mobileSettings>");
```

另请参见

[document.getMobileSettings\(\)](#)

## document.setOvalObjectProperty()

可用性

Flash CS3 Professional。

用法

```
document.setOvalObjectProperty(propertyName, value)
```

参数

**propertyName** 一个字符串，它指定要设置的属性。有关可接受值，请参阅 [Oval 对象](#) 的属性摘要表。

**value** 要赋给属性的值。可接受值会根据在 **propertyName** 中指定的属性而有所不同。

返回

无。

描述

方法；为椭圆图元对象的指定属性指定值。

示例

有关示例，请参阅 [Oval 对象](#) 中的各个属性。

另请参见

[Oval 对象](#)、[shape.isOvalObject](#)

## document.setPlayerVersion()

可用性

Flash CS3 Professional。

用法

```
document.setPlayerVersion(version)
```

参数

**version** 一个字符串，表示指定文档的目标 Flash Player 的版本。可接受的值为 FlashLite、FlashLite11、FlashLite20、FlashLite30、1、2、3、4、5、6、7、8、9、FlashPlayer10 和 AdobeAIR1\_1。

返回

如果播放器版本设置成功，则值为 **true**；否则为 **false**。

说明

方法；设置指定文档的目标 Flash Player 的版本。这与“发布设置”对话框中的设置值相同。



#### 示例

下面的示例将 Flash Player 6 设置为当前文档的目标播放器版本：

```
fl.getDocumentDOM().setPlayerVersion("6");
```

另请参见

[document.getPlayerVersion\(\)](#)

## document.setRectangleObjectProperty()

#### 可用性

Flash CS3 Professional。

#### 用法

```
document.setRectangleObjectProperty(propertyName, value)
```

#### 参数

**propertyName** 一个字符串，它指定要设置的属性。有关可接受值，请参阅 [Rectangle 对象](#) 的属性摘要表。

**value** 要赋给属性的值。可接受值会根据在 **propertyName** 中指定的属性而有所不同。

#### 返回

无。

#### 描述

方法；为矩形图元对象的指定属性指定值。

#### 示例

有关示例，请参阅 [Rectangle 对象](#) 中的各个属性。

另请参见

[Rectangle 对象](#)、[shape.isRectangleObject](#)

## document.setSelectionBounds()

#### 可用性

Flash MX 2004；Flash 8 中增加了 **bContactSensitiveSelection** 参数。

#### 用法

```
document.setSelectionBounds(boundingRectangle [, bContactSensitiveSelection])
```

#### 参数

**boundingRectangle** 一个矩形，它指定所选内容的新位置和新大小。有关 **boundingRectangle** 的格式信息，请参阅 [document.addNewRectangle\(\)](#)。

**bContactSensitiveSelection** 一个布尔值，它指定在对象选择过程中是启用 (**true**) 还是禁用 (**false**) 接触感应选择模式。默认值为 **false**。

返回  
无。

描述

方法；在单个操作中移动所选内容并调整其大小。

如果为 `bContactSensitiveSelection` 传递值，则该值仅对此方法有效，并不影响文档的接触感应选择模式（请参阅 [fl.contactSensitiveSelection](#)）。

示例

下面的实例将当前所选内容移动到 10, 20，并将其大小调整为 100, 200：

```
var l = 10;
var t = 20;
fl.getDocumentDOM().setSelectionBounds({left:l, top:t, right:(100+l), bottom:(200+t)});
```

另请参见

[document.selection](#)、[document.setSelectionRect\(\)](#)

## document.setSelectionRect()

可用性

Flash MX 2004；Flash 8 中增加了 `bContactSensitiveSelection` 参数。

用法

```
document.setSelectionRect(rect [, bReplaceCurrentSelection [, bContactSensitiveSelection]])
```

参数

**rect** 一个要设置为选中内容的矩形对象。有关 **rect** 的格式的信息，请参阅 [document.addNewRectangle\(\)](#)。

**bReplaceCurrentSelection** 一个布尔值，它指定方法是替换当前的选择 (`true`)，还是添加到当前的选择中 (`false`)。默认值为 `true`。

**bContactSensitiveSelection** 一个布尔值，它指定在对象选择过程中是启用 (`true`) 还是禁用 (`false`) 接触感应选择模式。默认值为 `false`。

返回  
无。

描述

方法；使用指定坐标绘制相对于舞台的矩形选取框。这不同于 `document.getSelectionRect()`，在后者中，矩形相对于正在编辑的对象。

此方法等效于使用“选择”工具拖动矩形。实例必须完全包含在要选择的矩形内。

如果为 `bContactSensitiveSelection` 传递值，则该值仅对此方法有效，并不影响文档的接触感应选择模式（请参阅 [fl.contactSensitiveSelection](#)）。

注：如果使用“历史记录”面板或菜单项重复执行 `setSelectionRect()`，将重复执行 `setSelectionRect()` 操作之前的步骤。

示例

在下面的示例中，第二个所选内容替换第一个所选内容：

**Document** 对象

```
fl.getDocumentDOM().setSelectionRect({left:1, top:1, right:200, bottom:200});  
fl.getDocumentDOM().setSelectionRect({left:364.0, top:203.0, right:508.0, bottom:434.0}, true);
```

在下面的示例中，将第二个所选内容添加到第一个所选内容中。这与按住 **Shift** 并选择另一个对象的手动操作相同。

```
fl.getDocumentDOM().setSelectionRect({left:1, top:1, right:200, bottom:200});  
fl.getDocumentDOM().setSelectionRect({left:364.0, top:203.0, right:508.0, bottom:434.0}, false);
```

另请参见

[document.getSelectionRect\(\)](#)、[document.selection](#)、[document.setSelectionBounds\(\)](#)

## document.setStageVanishingPoint()

可用性

Flash CS4 Professional.

用法

```
document.setStageVanishingPoint(point)
```

参数

**point** 一个点，它指定设置用于查看 3D 对象的消失点所在位置的 **x** 和 **y** 坐标。

返回

无。

描述

指定用于查看 3D 对象的消失点。

示例

下面的示例设置舞台消失点：

```
fl.getDocumentDOM().setStageVanishingPoint({x:45, y:45});
```

## document.setStageViewAngle()

可用性

Flash CS4 Professional.

用法

```
document.setStageViewAngle(angle)
```

参数

**angle** 0.0 到 179.0 之间的一个浮点值。

返回

无。

#### 描述

指定用于查看 3D 对象的透视角度。

#### 示例

下面的示例将舞台透视角度设置为 70 度：

```
fl.getDocumentDOM().setStageViewAngle(70);
```

## document.setStroke()

#### 可用性

Flash MX 2004。

#### 用法

```
document.setStroke(color, size, strokeType)
```

#### 参数

**color** 笔触的颜色，使用以下格式之一：

- 格式为 "#RRGGBB" 或 "#RRGGBBAA" 的字符串
- 格式为 0xRRGGBB 的十六进制数字
- 表示与十六进制数字等价的十进制整数

**size** 一个浮点值，它指定所选内容的新笔触大小。

**strokeType** 一个字符串，它指定所选内容的新笔触类型。可接受值为 "hairline"、"solid"、"dashed"、"dotted"、"ragged"、"stipple" 和 "hatched"。

#### 返回

无。

#### 描述

方法；设置所选笔触的颜色、宽度和样式。有关更改“工具”面板和“属性”检查器中笔触的信息，请参阅 [document.setCustomStroke\(\)](#)。

#### 示例

下面的示例将笔触的颜色设置为红色，大小设置为 3.25，类型设置为虚线：

```
fl.getDocumentDOM().setStroke("#ff0000", 3.25, "dashed");
```

## document.setStrokeColor()

#### 可用性

Flash MX 2004。

#### 用法

```
document.setStrokeColor(color)
```

#### 参数

**color** 笔触的颜色，使用以下格式之一：

- 格式为 "#RRGGBB" 或 "#RRGGBBAA" 的字符串
- 格式为 0xRRGGBB 的十六进制数字
- 表示与十六进制数字等价的十进制整数

#### 返回

无。

#### 描述

方法；将所选内容的笔触颜色更改为指定的颜色。有关更改“工具”面板和“属性”检查器中笔触的信息，请参阅 [document.setStroke\(\)](#)。

#### 示例

下面的示例中的三个语句使用指定颜色的每个不同格式设置笔触颜色：

```
f1.getDocumentDOM().setStrokeColor("#cc00cc");  
f1.getDocumentDOM().setStrokeColor(0xcc00cc);  
f1.getDocumentDOM().setStrokeColor(120000);
```

## document.setStrokeSize()

#### 可用性

Flash MX 2004。

#### 用法

```
document.setStrokeSize(size)
```

#### 参数

**size** 一个 0.25 到 10 之间的浮点值，它指定笔触的大小。此方法忽略两个小数位以后的精度。

#### 返回

无。

#### 描述

方法；将所选内容的笔触大小更改为指定的大小。有关更改“工具”面板和“属性”检查器中笔触的信息，请参阅 [document.setStroke\(\)](#)。

#### 示例

下面的示例将所选内容的笔触大小更改为 5：

```
f1.getDocumentDOM().setStrokeSize(5);
```

## document.setStrokeStyle()

可用性

Flash MX 2004。

用法

```
document.setStrokeStyle(strokeType)
```

参数

**strokeType** 一个字符串，它指定当前所选内容的笔触样式。可接受值为 "hairline"、"solid"、"dashed"、"dotted"、"ragged"、"stipple" 和 "hatched"。

返回

无。

描述

方法；将所选内容的笔触样式更改为指定的样式。有关更改“工具”面板和“属性”检查器中笔触的信息，请参阅 [document.setCustomStroke\(\)](#)。

示例

下面的示例将所选内容的笔触样式更改为 "dashed"：

```
fl.getDocumentDOM().setStrokeStyle("dashed");
```

## document.setTextRectangle()

可用性

Flash MX 2004。

用法

```
document.setTextRectangle(boundingRectangle)
```

参数

**boundingRectangle** 一个矩形，它指定文本项目所在区域的新尺寸。有关 **boundingRectangle** 的格式的信息，请参阅 [document.addNewRectangle\(\)](#)。

返回

布尔值：如果至少更改了一个文本字段的大小，则为 **true**；否则为 **false**。

说明

方法；将所选文本项目的边界矩形更改为指定的大小。此方法使文本在新矩形内重新填充；文本项目不进行缩放或变形。**boundingRectangle** 中传递的值的用法如下：

- 如果是水平静态文本，则此方法仅考虑 **boundingRectangle** 传递的宽度值；高度是自动计算的，以适合所有文本。
- 如果是垂直文本（因此为静态），则此方法仅考虑 **boundingRectangle** 传递的高度值；宽度是自动计算的，以适合所有文本。

- 如果是动态文本或输入，则此方法会考虑 `boundingBox` 传递的宽度和高度，所产生的矩形可能会大于适合所有文本所需的矩形。但如果参数指定的矩形太小而无法适合所有文本，则此方法仅考虑 `boundingBox` 传递的宽度值（高度是自动计算的，以适合所有文本）。

#### 示例

下面的示例将边界文本矩形的大小更改为指定的尺寸：

```
fl.getDocumentDOM().setTextRectangle({left:0, top:0, right:50, bottom:200})
```

## document.setTextSelection()

#### 可用性

Flash MX 2004。

#### 用法

```
document.setTextSelection(startIndex, endIndex)
```

#### 参数

**startIndex** 一个整数，它指定要选择的第一个字符的位置。第一个字符位置为 0（零）。

**endIndex** 一个整数，它指定所选内容的结束位置一直到（但不包括） **endIndex**。第一个字符位置为 0（零）。

#### 返回

布尔值：如果此方法可以成功地设置所选文本，则为 `true`；否则为 `false`。

#### 说明

方法；将当前所选文本字段中的所选文本设置为由 **startIndex** 和 **endIndex** 值指定的值。如果文本编辑尚未激活，则将其激活。

#### 示例

下面的示例选择从第 6 个字符到第 25 个字符的文本：

```
fl.document.setTextSelection(5, 25);
```

## document.setTextString()

#### 可用性

Flash MX 2004。

#### 用法

```
document.setTextString(text [, startIndex [, endIndex]])
```

#### 参数

**text** 一个字符串，它由要插入文本字段的字符组成。

**startIndex** 一个整数，它指定要替换的第一个字符。第一个字符位置为 0（零）。此参数是可选的。

**endIndex** 一个整数，它指定要替换的最后一个字符。此参数是可选的。

返回

布尔值：如果至少设置了一个文本字符串的文本，则为 **true**；否则为 **false**。

说明

方法；插入文本字符串。如果没有传递可选参数，则替换现有的所选文本；如果 **Text** 对象当前未进行编辑，则替换整个文本字符串。如果只传递 **startIndex**，则在该位置插入传递的字符串。如果传递了 **startIndex** 和 **endIndex**，则传递的字符串替换从 **startIndex** 到（但不包括）**endIndex** 的文本段。

示例

下面的示例将当前所选文本替换为“Hello World”：

```
var success = fl.getDocumentDOM().setTextString("Hello World!");
```

下面的示例将“hello”插入当前所选文本的第 6 个位置：

```
var pass = fl.getDocumentDOM().setTextString("hello", 6);
```

下面的示例在当前所选文本的第 2 个位置开始直到（但不包括）第 7 个位置插入“**Howdy**”：

```
var ok = fl.getDocumentDOM().setTextString("Howdy", 2, 7);
```

另请参见

[document.getTextString\(\)](#)

## document.setTransformationPoint()

可用性

Flash MX 2004。

用法

```
document.setTransformationPoint( transformationPoint )
```

参数

**transformationPoint** 一个点（例如 {x:10,y:20}，其中 **x** 和 **y** 是浮点数），它指定以下每个元素的变形点的值：

- 形状：**transformationPoint** 是相对于文档来设置的（0,0 为舞台的左上角）。
- 元件：**transformationPoint** 是相对于元件的注册点来设置的（0,0 位于注册点）。
- 文本：**transformationPoint** 是相对于文本字段来设置的（0,0 是文本字段的左上角）。
- 位图 / 视频：**transformationPoint** 是相对于位图 / 视频来设置的（0,0 为位图或视频的左上角）。
- 绘制对象、椭圆图元和矩形图元以及组：**transformationPoint** 是相对于文档来设置的（0,0 为舞台的左上角）。若要相对于对象、图元或组的中心点来设置 **transformationPoint**，请使用 [element.setTransformationPoint\(\)](#)。

返回

无。

描述

方法；设置当前所选内容的变形点的位置。



#### 示例

下面的示例将当前所选内容的变形点设置为 100, 200:

```
fl.getDocumentDOM().setTransformationPoint({x:100, y:200});
```

另请参见

[document.getTransformationPoint\(\)](#)、[element.setTransformationPoint\(\)](#)

## document.silent

#### 可用性

Flash MX 2004。

#### 用法

```
document.silent
```

#### 描述

属性；一个布尔值，它指定对象是否可访问。这等效于“辅助功能”面板中的“使影片可访问”设置的反向逻辑。即，如果 `document.silent` 为 `true`，则与不选中“使影片可访问”选项的效果相同。如果为 `false`，则等效于选中“使影片可访问”选项。

#### 示例

下面的示例将 `isSilent` 变量设置为 `silent` 属性的值：

```
var isSilent = fl.getDocumentDOM().silent;
```

下面的示例将 `silent` 属性设置为 `false`，以表明文档可访问：

```
fl.getDocumentDOM().silent = false;
```

## document.skewSelection()

#### 可用性

Flash MX 2004。

#### 用法

```
document.skewSelection(xSkew, ySkew [, whichEdge])
```

#### 参数

**xSkew** 一个浮点数，它指定倾斜的 x 量，以度为单位。

**ySkew** 一个浮点数，它指定倾斜的 y 量，以度为单位。

**whichEdge** 一个字符串，它指定执行变形的边缘；如果省略，则在变形点执行倾斜。可接受值为 "top center"、"right center"、"bottom center" 和 "left center"。此参数是可选的。

#### 返回

无。

#### 描述

方法；按照指定的量倾斜所选内容。效果与使用“任意变形”工具倾斜对象相同。

#### 示例

下面的示例将所选对象垂直倾斜 2.0 度，水平倾斜 1.5 度。第二个示例在顶部中心处使对象变形：

```
fl.getDocumentDOM().skewSelection(2.0, 1.5);  
fl.getDocumentDOM().skewSelection(2.0, 1.5, "top center");
```

## document.smoothSelection()

#### 可用性

Flash MX 2004。

#### 用法

```
document.smoothSelection()
```

#### 参数

无。

#### 返回

无。

#### 描述

方法；平滑选择的每条填充轮廓或弯曲线的曲线。此方法与“工具”面板中的“平滑”按钮执行相同的动作。

#### 示例

下面的示例平滑当前所选内容的曲线：

```
fl.getDocumentDOM().smoothSelection();
```

## document.sourcePath

#### 可用性

Flash CS4 Professional.

#### 用法

```
document.sourcePath
```

#### 描述

属性；一个字符串，其中包含文档的 **ActionScript 3.0** 源路径中项目的列表，用于指定 **ActionScript** 类文件的位置。该字符串中的项目以分号分隔。在创作工具中，通过以下方式指定这些项目：选择“文件”>“发布设置”，然后在“Flash”选项卡上选择“**ActionScript 3.0** 脚本设置”。

#### 示例

下面的示例将 `./Class files` 文件夹添加到文档的源路径：

```
var myDoc = fl.getDocumentDOM();
fl.trace(myDoc.sourcePath);
myDoc.sourcePath = "./Class files;" + myDoc.sourcePath;
fl.trace(myDoc.sourcePath);
```

另请参见

[document.externalLibraryPath](#)、[document.libraryPath](#)、[fl.sourcePath](#)

## document.space()

可用性

Flash MX 2004。

用法

```
document.space(direction [, bUseDocumentBounds])
```

参数

**direction** 一个字符串，它指定在所选内容中间隔对象的方向。可接受值为 "horizontal" 或 "vertical"。

**bUseDocumentBounds** 一个布尔值，如果设置为 **true**，则使用文档范围间隔对象。否则，此方法使用所选对象的范围。默认值为 **false**。此参数是可选的。

返回

无。

描述

方法；均匀间隔所选内容中的对象。

示例

下面的示例相对于舞台水平间隔对象：

```
fl.getDocumentDOM().space("horizontal",true);
```

下面的示例相对于彼此水平间隔对象：

```
fl.getDocumentDOM().space("horizontal");
```

下面的示例使对象相对于彼此水平间隔，并将 **bUseDocumentBounds** 明确设置为 **false**：

```
fl.getDocumentDOM().space("horizontal",false);
```

另请参见

[document.getAlignToDocument\(\)](#)、[document.setAlignToDocument\(\)](#)

## document.straightenSelection()

可用性

Flash MX 2004。

**Document** 对象

## 用法

```
document.straightenSelection()
```

## 参数

无。

## 返回

无。

## 描述

方法；伸直当前所选笔触。此方法等效于使用“工具”面板中的“伸直”按钮。

## 示例

下面的示例伸直当前所选内容的曲线：

```
fl.getDocumentDOM().straightenSelection();
```

## document.swapElement()

## 可用性

Flash MX 2004。

## 用法

```
document.swapElement(name)
```

## 参数

**name** 一个字符串，它指定要使用的库项目的名称。

## 返回

无。

## 描述

方法；用指定的选择内容交换当前选择内容。所选内容必须包含图形、按钮、影片剪辑、视频或位图。如果未选择任何对象或未能找到给定对象，则此方法显示一条错误消息。

## 示例

下面的示例用库中的 Symbol 1 交换当前所选内容：

```
fl.getDocumentDOM().swapElement('Symbol 1');
```

## document.swapStrokeAndFill()

## 可用性

Flash 8。

## 用法

```
document.swapStrokeAndFill()
```

#### 参数

无。

#### 返回

无。

#### 描述

方法；交换笔触颜色和填充颜色。

#### 示例

下面的示例交换当前文档中的笔触颜色和填充颜色：

```
fl.getDocumentDOM().swapStrokeAndFill();
```

## document.testMovie()

#### 可用性

Flash MX 2004。

#### 用法

```
document.testMovie([Boolean abortIfErrorsExist])
```

#### 参数

**abortIfErrorsExist** 布尔值；默认值为 **false**。如果设置为 **true**，将不启动测试影片会话，如果存在编译器错误，则不会打开 .swf 窗口。编译器警告不会中止命令。此参数是在 Flash Professional CS5 中添加的。

#### 返回

无。

#### 描述

方法；对文档执行“测试影片”操作。

#### 示例

以下示例测试当前文档的影片，但如果存在编译器错误，则中止测试影片：

```
fl.getDocumentDOM().testMovie(1);
```

另请参见

[document.canTestMovie\(\)](#)、[document.testScene\(\)](#)

## document.testScene()

#### 可用性

Flash MX 2004。

#### 用法

```
document.testScene()
```

#### 参数

无。

#### 返回

无。

#### 描述

方法；对文档的当前场景执行“测试场景”操作。

#### 示例

下面的示例测试文档中的当前场景：

```
fl.getDocumentDOM().testScene();
```

另请参见

[document.canTestScene\(\)](#)、[document.testMovie\(\)](#)

## document.timelines

#### 可用性

Flash MX 2004。

#### 用法

```
document.timelines
```

#### 描述

只读属性；一个 **Timeline** 对象的数组（请参阅 [Timeline 对象](#)）。

#### 示例

下面的示例获取活动文档中的当前时间轴组成的数组，并在“输出”面板中显示当前时间轴的名称：

```
var i = 0;
var curTimelines = fl.getDocumentDOM().timelines;
while(i < fl.getDocumentDOM().timelines.length){
    alert(curTimelines[i].name);
    ++i;
}
```

另请参见

[document.currentTimeline](#)、[document.getTimeline\(\)](#)

## document.traceBitmap()

可用性

Flash MX 2004。

用法

```
document.traceBitmap(threshold, minimumArea, curveFit, cornerThreshold)
```

参数

**threshold** 一个整数，它控制跟踪的位图中颜色的数目。可接受值为 0 和 500 之间的整数。

**minimumArea** 一个整数，它以像素为单位指定半径。可接受值为 1 和 1000 之间的整数。

**curveFit** 一个字符串，它指定轮廓绘制的平滑程度。可接受值为 "pixels"、"very tight"、"tight"、"normal"、"smooth" 和 "very smooth"。

**cornerThreshold** 一个字符串，与 **curveFit** 类似，但它与位图图像的角有关。可接受值为 "many corners"、"normal" 和 "few corners"。

返回

无。

描述

方法；对当前所选内容执行跟踪位图。此方法等效于选择“修改”>“位图”>“跟踪位图”。

示例

下面的示例使用指定的参数跟踪所选位图：

```
fl.getDocumentDOM().traceBitmap(0, 500, 'normal', 'normal');
```

## document.translate3DCenter()

可用性

Flash CS4 Professional.

用法

```
document.translate3DCenter(xyzCoordinate)
```

参数

**xyzCoordinate** 一个 XYZ 坐标，它指定 3D 旋转或平移的中心点。

返回

无。

描述

方法：设置 XYZ 位置，所选内容围绕此位置进行平移或旋转。此方法仅适用于影片剪辑。

示例

下面的示例指定 3D 平移的 XYZ 轴：

```
fl.getDocumentDOM().translate3DCenter({x:180, y:18, z:-30});
```

## document.translate3DSelection()

可用性

Flash CS4 Professional.

用法

```
document.translate3DSelection(xyzCoordinate, bGlobalTransform)
```

参数

**xyzCoordinate** 一个 XYZ 坐标，它指定 3D 平移的轴。

**bGlobalTransform** 一个布尔值，它指定变形模式应是全局模式 (**true**) 还是本地模式 (**false**)。

返回

无。

描述

方法：对所选内容应用 3D 平移。此方法仅适用于影片剪辑。

示例

在下面的示例中，首先相对于舞台平移所选内容（全局），然后相对于自身平移所选内容（局部）。

```
var myDocument = fl.getDocumentDOM();  
myDocument.translate3DSelection({x:52.0, y:0, z:0}, true);  
myDocument.translate3DSelection({x:52.0, y:0, z:-55.2}, false);
```

另请参见

[document.translate3DCenter\(\)](#)

## document.transformSelection()

可用性

Flash MX 2004。

用法

```
document.transformSelection(a, b, c, d)
```

参数

**a** 一个浮点数，它指定变形矩阵的 (0,0) 元素。

**b** 一个浮点数，它指定变形矩阵的 (0,1) 元素。

**c** 一个浮点数，它指定变形矩阵的 (1,0) 元素。

**d** 一个浮点数，它指定变形矩阵的 (1,1) 元素。



返回  
无。

描述  
方法；通过应用参数指定的矩阵，对当前所选内容执行常规变形。有关详细信息，请参阅 [element.matrix](#) 属性。

示例  
下面的示例在 **x** 方向上使用系数 2 伸展所选内容：  

```
fl.getDocumentDOM().transformSelection(2.0, 0.0, 0.0, 1.0);
```

## document.unGroup()

可用性  
Flash MX 2004。

用法  
`document.unGroup()`

参数  
无。

返回  
无。

描述  
方法；取消组合当前所选内容。

示例  
下面的示例取消组合当前所选内容中的元素：  

```
fl.getDocumentDOM().unGroup();
```

另请参见  
[document.group\(\)](#)

## document.union()

可用性  
Flash 8。

用法  
`document.union()`

参数  
无。

#### 返回

一个布尔值：如果成功，则为 `true`；否则为 `false`。

#### 说明

方法；将所有所选形状合并到一个绘制对象中。

#### 示例

下面的示例将所有所选形状合并到一个绘制对象中：

```
fl.getDocumentDOM().union();
```

#### 另请参见

[document.crop\(\)](#)、[document.deleteEnvelope\(\)](#)、[document.intersect\(\)](#)、[document.punch\(\)](#)、[shape.isDrawingObject](#)

## document.unlockAllElements()

#### 可用性

Flash MX 2004。

#### 用法

```
document.unlockAllElements()
```

#### 参数

无。

#### 返回

无。

#### 描述

方法；解除锁定当前所选帧上的全部锁定元素。

#### 示例

下面的示例解除锁定当前帧中的所有锁定对象：

```
fl.getDocumentDOM().unlockAllElements();
```

#### 另请参见

[element.locked](#)

## document.viewMatrix

#### 可用性

Flash MX 2004。

#### 用法

```
document.viewMatrix
```

#### 描述

只读属性；一个 **Matrix** 对象。文档在编辑模式下时，可使用 **viewMatrix** 从对象空间转换为文档空间。鼠标位置（某个工具接收的位置）是相对于当前正在编辑的对象的。请参阅 [Matrix 对象](#)。

例如，如果创建一个元件，双击它进行编辑，然后使用“多角星形”工具绘制，则点 (0,0) 将位于该元件的注册点。但是，**drawingLayer** 对象需要文档空间中的值，因此如果使用 **drawingLayer** 从 (0,0) 开始绘制直线，该直线将从舞台左上角开始。**viewMatrix** 属性提供一种从正在编辑的对象的的空间转换为文档空间的方法。

#### 示例

下面的示例获取 **viewMatrix** 属性的值：

```
var mat = fl.getDocumentDOM().viewMatrix;
```

## document.width

#### 可用性

Flash MX 2004。

#### 用法

```
document.width
```

#### 描述

属性；一个整数，它以像素为单位指定文档（舞台）的宽度。

#### 示例

下面的示例将舞台宽度设置为 400 像素。

```
fl.getDocumentDOM().width= 400;
```

另请参见

[document.height](#)

## document.xmlPanel()

#### 可用性

Flash MX 2004。

#### 用法

```
document.xmlPanel(fileURI)
```

#### 参数

**fileURI** 一个字符串，表示为 **file:/// URI**，它指定 XML 文件（该文件定义面板中的控件）的路径。必须指定完整路径。

#### 返回

一个对象，它具有为 XML 文件中定义的所有控件定义的属性。所有属性都以字符串的形式返回。返回的对象将有一个名为 "dismiss" 的预定义属性，该属性具有字符串值 "accept" 或 "cancel"。

说明

方法；张贴 XMLUI 对话框。请参阅 [fl.xmlui](#)。

示例

下面的示例加载 **Test.xml** 文件，并显示其中包含的所有属性：

```
var obj = fl.getDocumentDOM().xmlPanel(fl.configURI + "Commands/Test.xml");
for (var prop in obj) {
    fl.trace("property " + prop + " = " + obj[prop]);
}
```

## document.zoomFactor

可用性

Flash 8。

用法

```
document.zoomFactor
```

描述

属性；指定创作时舞台的缩放百分比。值 1 等于 100% 缩放，值 8 等于 800% 缩放，值 .5 等于 50% 缩放，依次类推。

示例

下面的示例将舞台的缩放系数设置为 200%。

```
fl.getDocumentDOM().zoomFactor = 2;
```

## 第 12 章 : drawingLayer 对象

可用性

Flash MX 2004。

描述

`drawingLayer` 对象可以从 JavaScript 作为 `flash` 对象的子对象访问。如果用户希望在拖放时进行临时绘制（例如创建选取框时），请考虑对可扩展工具使用 `drawingLayer` 对象。在调用任何其他 `drawingLayer` 方法之前，应先调用 `drawingLayer.beginFrame()`。

方法摘要

`drawingLayer` 对象具有以下方法：

方法	描述
<code>drawingLayer.beginDraw()</code>	将 <code>Flash</code> 设置为绘制模式。
<code>drawingLayer.beginFrame()</code>	擦除先前使用 <code>drawingLayer</code> 绘制的内容，准备接受更多绘制命令。
<code>drawingLayer.cubicCurveTo()</code>	使用参数作为三次线段的坐标，从当前钢笔位置开始绘制三次曲线。
<code>drawingLayer.curveTo()</code>	绘制一条二次曲线线段，起点为当前绘制位置，终点为指定的点。
<code>drawingLayer.drawPath()</code>	绘制指定的路径。
<code>drawingLayer.endDraw()</code>	退出绘制模式。
<code>drawingLayer.endFrame()</code>	表示一组绘制命令的结束。
<code>drawingLayer.lineTo()</code>	绘制一条直线，起点为当前绘制位置，终点为点 (x,y)。
<code>drawingLayer.moveTo()</code>	设置当前绘制位置。
<code>drawingLayer.newPath()</code>	返回新的 <code>Path</code> 对象。
<code>drawingLayer.setColor()</code>	设置随后所绘制数据的颜色。
<code>drawingLayer.setFill()</code>	此方法不可用。
<code>drawingLayer.setStroke()</code>	此方法不可用。

### drawingLayer.beginDraw()

可用性

Flash MX 2004。

用法

```
drawingLayer.beginDraw([persistentDraw])
```

参数

`persistentDraw` 一个布尔值（可选）。如果设置为 `true`，则表示最后一帧中绘制的内容在调用新的 `beginDraw()` 或 `beginFrame()` 命令之前一直保留在舞台中。（在这种情况下，`frame` 表示开始和结束绘制的位置，而不是指时间轴帧。）例如，

当用户绘制一个矩形时，可以在拖放鼠标时预览此形状的轮廓。如果希望在释放鼠标按键后仍然保持此预览轮廓，可将 `persistentDraw` 设置为 `true`。

返回  
无。

描述

方法；将 `Flash` 设置为绘制模式。绘制模式用于在按下鼠标按钮时进行临时绘制。通常情况下，只有在创建可扩展工具时才使用此方法。

示例

下面的示例将 `Flash` 设置为绘制模式：

```
f1.drawingLayer.beginDraw();
```

## drawingLayer.beginFrame()

可用性

Flash MX 2004。

用法

```
drawingLayer.beginFrame()
```

参数

无。

返回

无。

描述

方法；擦除先前使用 `drawingLayer` 绘制的图像，并准备接受更多绘画命令。应该在 `drawingLayer.beginDraw()` 之后调用。在 `drawingLayer.beginFrame()` 和 `drawingLayer.endFrame()` 之间绘制的图像会保存在舞台中，直到调用下一个 `beginFrame()` 和 `endFrame()`。（在这种情况下，`frame` 表示开始和结束绘制的位置，而不是指时间轴帧。）通常情况下，只有在创建可扩展工具时才使用此方法。请参阅 [drawingLayer.beginDraw\(\)](#)。

## drawingLayer.cubicCurveTo()

可用性

Flash MX 2004。

用法

```
drawingLayer.cubicCurveTo(x1Ctrl, y1Ctrl, x2Ctrl, y2Ctrl, xEnd, yEnd)
```

参数

**x1Ctrl** 一个浮点值，是第一个控制点的 x 位置。

**drawingLayer** 对象

**y1Ctl** 一个浮点值，是第一个控制点的 **y** 位置。

**x2Ctl** 一个浮点值，是中间控制点的 **x** 位置。

**y2Ctl** 一个浮点值，是中间控制点的 **y** 位置。

**xEnd** 一个浮点值，是最后一个控制点的 **x** 位置。

**yEnd** 一个浮点值，是最后一个控制点的 **y** 位置。

返回

无。

描述

方法；从当前钢笔位置，使用参数作为三次线段的坐标绘制三次曲线。通常情况下，只有在创建可扩展工具时才使用此方法。

示例

下面的示例使用指定的控制点绘制一条三次曲线：

```
fl.drawingLayer.cubicCurveTo(0, 0, 1, 1, 2, 0);
```

## drawingLayer.curveTo()

可用性

Flash MX 2004。

用法

```
drawingLayer.curveTo(xCtl, yCtl, xEnd, yEnd)
```

参数

**xCtl** 一个浮点值，是控制点的 **x** 位置。

**yCtl** 一个浮点值，是控制点的 **y** 位置。

**xEnd** 一个浮点值，是最后一个控制点的 **x** 位置。

**yEnd** 一个浮点值，是最后一个控制点的 **y** 位置。

返回

无。

描述

方法；绘制一条从当前绘画位置开始，到指定点结束的二次曲线线段。通常情况下，只有在创建可扩展工具时才使用此方法。

示例

下面的示例使用指定的控制点绘制一条二次曲线：

```
fl.drawingLayer.curveTo(0, 0, 2, 0);
```

## drawingLayer.drawPath()

可用性

Flash MX 2004。

用法

```
drawingLayer.drawPath(path)
```

参数

**path** 一个要绘制的 [Path 对象](#)。

返回

无。

描述

方法；绘制由 **path** 参数指定的路径。通常情况下，只有在创建可扩展工具时才使用此方法。

示例

下面的示例绘制一条由名为 **gamePath** 的 **Path** 对象指定的路径：

```
fl.drawingLayer.drawPath(gamePath);
```

## drawingLayer.endDraw()

可用性

Flash MX 2004。

用法

```
drawingLayer.endDraw()
```

参数

无。

返回

无。

描述

方法；退出绘制模式。绘制模式用于在按下鼠标按钮时进行临时绘制。通常情况下，只有在创建可扩展工具时才使用此方法。

示例

下面的示例退出绘制模式：

```
fl.drawingLayer.endDraw();
```



## drawingLayer.endFrame()

可用性

Flash MX 2004。

用法

```
drawingLayer.endFrame()
```

参数

无。

返回

无。

描述

方法；表示一组绘画命令的结束。一组绘画命令指的是在 `drawingLayer.beginFrame()` 和 `drawingLayer.endFrame()` 之间绘制的所有图像。下一次调用 `drawingLayer.beginFrame()` 时，将擦除这组绘画命令绘制的所有图像。通常情况下，只有在创建可扩展工具时才使用此方法。

## drawingLayer.lineTo()

可用性

Flash MX 2004。

用法

```
drawingLayer.lineTo(x, y)
```

参数

**x** 一个浮点值，是要绘制的直线终点的 x 坐标。

**y** 一个浮点值，是要绘制的直线终点的 y 坐标。

返回

无。

描述

方法；绘制一条从当前绘画位置开始，到点 (x,y) 结束的直线。通常情况下，只有在创建可扩展工具时才使用此方法。

示例

下面的示例绘制一条从当前绘画位置开始，到点 (20,30) 结束的直线：

```
fl.drawingLayer.lineTo(20, 30);
```

## drawingLayer.moveTo()

可用性

Flash MX 2004。

用法

```
drawingLayer.moveTo(x, y)
```

参数

**x** 一个浮点值，它指定要开始绘制的位置的 x 坐标。

**y** 一个浮点值，它指定要开始绘制的位置的 y 坐标。

返回

无。

描述

方法；设置当前绘画位置。通常情况下，只有在创建可扩展工具时才使用此方法。

示例

下面的示例将当前绘画位置设置为点 (10,15):

```
f1.drawingLayer.moveTo(10, 15);
```

## drawingLayer.newPath()

可用性

Flash MX 2004。

用法

```
drawingLayer.newPath()
```

参数

无。

返回

一个 **Path** 对象。

说明

方法；返回一个新的 **Path** 对象。通常情况下，只有在创建可扩展工具时才使用此方法。请参阅 [Path 对象](#)。

示例

下面的示例返回一个新的 **Path** 对象:

```
f1.drawingLayer.newPath();
```

## drawingLayer.setColor()

可用性

Flash MX 2004。

用法

```
drawingLayer.setColor(color)
```

参数

**color** 随后绘制的数据的颜色，使用以下格式之一：

- 格式为 "#RRGGBB" 或 "#RRGGBBAA" 的字符串
- 格式为 0xRRGGBB 的十六进制数字
- 表示与十六进制数字等价的十进制整数

返回

无。

描述

方法；设置随后绘制的数据的颜色。仅适用于永久数据。要使用此方法，传递给 `drawingLayer.beginDraw()` 的参数必须设置为 `true`。通常情况下，只有在创建可扩展工具时才使用此方法。请参阅 [drawingLayer.beginDraw\(\)](#)。

示例

下面的示例在舞台上绘制一条红色直线：

```
f1.drawingLayer.beginDraw( true );  
f1.drawingLayer.beginFrame();  
f1.drawingLayer.setColor( "#ff0000" );  
f1.drawingLayer.moveTo(0,0);  
f1.drawingLayer.lineTo(100,100);  
f1.drawingLayer.endFrame();  
f1.drawingLayer.endDraw();
```

## drawingLayer.setFill()

此方法不可用。

## drawingLayer.setStroke()

此方法不可用。

## 第 13 章 : Edge 对象

### 可用性

Flash MX 2004。

### 描述

Edge 对象表示舞台上一个形状的边缘。

### 方法摘要

Edge 对象具有以下方法：

方法	描述
<a href="#">edge.getControl()</a>	获取一个点对象，该对象设置为所指定边缘控制点的位置。
<a href="#">edge.getHalfEdge()</a>	返回 <a href="#">HalfEdge</a> 对象。
<a href="#">edge.setControl()</a>	设置边缘控制点的位置。
<a href="#">edge.splitEdge()</a>	将边缘拆分为两段。

### 属性摘要

Edge 对象具有以下属性：

属性	描述
<a href="#">edge.cubicSegmentIndex</a>	一个整数，它指定边缘的三次线段的索引值。
<a href="#">edge.id</a>	只读；表示边缘唯一标识符的整数。
<a href="#">edge.isLine</a>	只读；值为 0 或 1 的整数。
<a href="#">edge.stroke</a>	一个 <a href="#">Stroke</a> 对象。

## edge.cubicSegmentIndex

### 可用性

Flash CS4 Professional.

### 用法

`edge.cubicSegmentIndex`

### 描述

只读属性；一个整数，它指定边缘的三次线段的索引值（请参阅 [shape.getCubicSegmentPoints\(\)](#)）。

### 示例

下面的代码显示所指定边缘的所有三次线段的索引值：

```
var theShape = fl.getDocumentDOM().selection[0];
var edgesArray = theShape.edges;
for(var i=0;i<edgesArray.length; i++) {
    fl.trace(edgesArray[i].cubicSegmentIndex);
}
```

## edge.getControl()

可用性

Flash MX 2004。

用法

```
edge.getControl(i)
```

参数

**i** 一个整数，指定返回哪个边缘控制点。指定 **0** 返回第一个控制点，指定 **1** 返回中间的控制点，指定 **2** 返回最后一个控制点。如果 **edge.isLine** 属性为 **true**，那么中间控制点设置为联接起始控制点和终止控制点的线段的中点。

返回

指定的控制点。

说明

方法；返回一个点对象，用于设置指定的边缘控制点的位置。

示例

下面的示例在 **pt** 变量中存储指定形状的第一个控制点：

```
var shape = fl.getDocumentDOM().selection[0];
var pt = shape.edges[0].getControl(0);
```

## edge.getHalfEdge()

可用性

Flash MX 2004。

用法

```
edge.getHalfEdge(index)
```

参数

**index** 一个整数，指定返回哪个半边缘。索引 的值必须为 **0**（返回第一个半边缘）或 **1**（返回第二个半边缘）。

返回

一个 **HalfEdge** 对象。

说明

方法；返回一个 **HalfEdge** 对象。

#### 示例

下面的示例将所指定边缘的半边缘存储在 `hEdge0` 和 `hEdge1` 变量中：

```
var shape = fl.getDocumentDOM().selection[0];  
var edge = shape.edges[0];  
var hEdge0 = edge.getHalfEdge(0);  
var hEdge1 = edge.getHalfEdge(1);
```

## edge.id

#### 可用性

Flash MX 2004。

#### 用法

`edge.id`

#### 描述

只读属性；表示边缘的唯一标识符的整数。

#### 示例

下面的示例将所指定边缘的唯一标识符存储在 `my_shape_id` 变量中：

```
var shape = fl.getDocumentDOM().selection[0];  
var my_shape_id = shape.edges[0].id;
```

## edge.isLine

#### 可用性

Flash MX 2004。

#### 用法

`edge.isLine`

#### 描述

只读属性；值为 0 或 1 的整数。值为 1 表示边缘为直线。在这种情况下，中间的控制点把联接两个端点的直线拆分为两半。

#### 示例

下面的示例确定所指定边缘是否为直线，并在“输出”面板中显示值 1（边缘是直线）或 0（边缘不是直线）：

```
var shape = fl.getDocumentDOM().selection[0];  
fl.trace(shape.edges[0].isLine);
```

## edge.setControl()

#### 可用性

Flash MX 2004。

## Edge 对象

## 用法

```
edge.setControl(index, x, y)
```

## 参数

**index** 一个整数，它指定要设置的控制点。使用值 0、1 或 2 分别指定起始控制点、中间控制点和终止控制点。

**x** 一个浮点值，指定控制点的水平位置。如果舞台处于“编辑”或者“在当前位置编辑”模式，则点坐标相对于被编辑的对象。否则，该点坐标相对于舞台。

**y** 一个浮点值，指定控制点的垂直位置。如果舞台处于“编辑”或者“在当前位置编辑”模式，则点坐标相对于被编辑的对象。否则，该点坐标相对于舞台。

## 返回

无。

## 描述

方法；设置边缘控制点的位置。在使用此方法之前，必须先调用 `shape.beginEdit()`。请参阅 [shape.beginEdit\(\)](#)。

## 示例

下面的示例将指定边缘的起始控制点设置为 (0, 1) 坐标：

```
x = 0; y = 1;
var shape = fl.getDocumentDOM().selection[0];
shape.beginEdit();
shape.edges[0].setControl(0, x, y);
shape.endEdit();
```

## edge.splitEdge()

## 可用性

Flash MX 2004。

## 用法

```
edge.splitEdge(t)
```

## 参数

**t** 0 和 1 之间的浮点值，指定从哪里拆分边缘。值 0 表示一个端点，值 1 表示另一个端点。例如，传递值 0.5 会从中拆分边缘，对直线来说就是从中心拆分。如果边缘为曲线，那么 0.5 表示曲线的参数中点。

## 返回

无。

## 描述

方法；把边缘拆分为两段。在使用此方法之前，必须先调用 `shape.beginEdit()`。

## 示例

下面的示例将指定的边缘平均拆分为两段：

```
var shape = fl.getDocumentDOM().selection[0];  
shape.beginEdit()  
shape.edges[0].splitEdge( 0.5 );  
shape.endEdit()
```

## edge.stroke

可用性

Flash CS4 Professional.

用法

`edge.stroke`

描述

属性; 一个 [Stroke](#) 对象。

示例

下面的示例显示所选对象第一个边缘的笔触颜色:

```
var shape = fl.getDocumentDOM().selection[0];  
fl.trace(shape.edges[0].stroke.color);
```



## 第 14 章 : Element 对象

可用性

Flash MX 2004。

描述

出现在舞台上的所有对象都是 **Element** 类型。下面的代码示例使您可以选择一个元素：

```
var e1 = fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0];
```

方法摘要

**Element** 对象具有以下方法：

方法	描述
<code>element.getPersistentData()</code>	检索 <i>name</i> 参数指定的数据的值。
<code>element.getTransformationPoint()</code>	获取指定元素的变形点的值。
<code>element.hasPersistentData()</code>	确定指定数据是否附加到指定元素。
<code>element.removePersistentData()</code>	删除具有已附加到该对象的指定名称的任何永久数据。
<code>element.setPersistentData()</code>	存储包含元素的数据。
<code>element.setTransformationPoint()</code>	设置元素的变形点的位置。

属性摘要

**Element** 对象具有以下属性：

属性	描述
<code>element.depth</code>	只读；一个大于 0 的整数值，它表示对象在视图中的深度。
<code>element.elementType</code>	只读；一个字符串，它表示指定元素的类型。
<code>element.height</code>	一个浮点值，它指定元素的高度（单位为像素）。
<code>element.layer</code>	只读；它表示元素所在的 <b>Layer</b> 对象。
<code>element.left</code>	只读；一个浮点值，它表示元素的左侧。
<code>element.locked</code>	一个布尔值：如果元素被锁定，则为 <b>true</b> ；否则为 <b>false</b> 。
<code>element.matrix</code>	一个 <b>Matrix</b> 对象。 <b>matrix</b> 的属性包括 <i>a</i> 、 <i>b</i> 、 <i>c</i> 、 <i>d</i> 、 <i>tx</i> 和 <i>ty</i> 。其中 <i>a</i> 、 <i>b</i> 、 <i>c</i> 和 <i>d</i> 是浮点数值； <i>tx</i> 和 <i>ty</i> 是坐标。
<code>element.name</code>	一个字符串，它指定元素的名称，通常称为实例名称。
<code>element.rotation</code>	一个介于 -180 和 180 之间的整数或浮点值，它指定对象的顺时针旋转，以度为单位。
<code>element.scaleX</code>	一个浮点值，它指定元件、绘制对象、矩形图元和椭圆图元的 x 缩放值。
<code>element.scaleY</code>	一个浮点值，它指定了元件、绘制对象、矩形图元和椭圆图元的 y 缩放值。
<code>element.selected</code>	一个布尔值，它指定元素是否处于选中状态。
<code>element.skewX</code>	一个介于 -180 和 180 之间的浮点值，它指定元件、绘制对象、矩形图元和椭圆图元的 x 倾斜值。

属性	描述
<a href="#">element.skewY</a>	一个介于 -180 和 180 之间的浮点值，它指定元件、绘制对象、矩形图元和椭圆图元的 y 倾斜值。
<a href="#">element.top</a>	只读；元素的顶端。
<a href="#">element.transformX</a>	一个浮点数，它指定元素父级坐标系内所选元素变形点的 x 值。
<a href="#">element.transformY</a>	一个浮点数，它指定元素父级坐标系内所选元素变形点的 y 值。
<a href="#">element.width</a>	一个浮点值，它指定元素的宽度（单位为像素）。
<a href="#">element.x</a>	一个浮点值，它指定所选元素注册点的 x 值。
<a href="#">element.y</a>	一个浮点值，它指定所选元素注册点的 y 值。

## element.depth

可用性

Flash MX 2004。

用法

`element.depth`

描述

只读属性；一个大于 0 的整数值，它表示对象在视图中的深度。对象在舞台中的绘制顺序决定了哪个对象在上，哪个对象在下。对象顺序也可通过“修改”>“排列”菜单项进行管理。

示例

下面的示例在“输出”面板中显示指定元素的深度：

```
// Select an object and run this script.  
fl.trace("Depth of selected object: " + fl.getDocumentDOM().selection[0].depth);
```

请参阅 [element.elementType](#) 示例。

## element.elementType

可用性

Flash MX 2004。

用法

`element.elementType`

描述

只读属性；一个字符串，它表示指定元素的类型。其值是下列值之一：“shape”、“text”、“instance”或“shapeObj”。“shapeObj”是通过可扩展工具创建的。

示例

下面的示例将第一个元素的类型保存在 `eType` 变量中：

```
// In a new file, place a movie clip on first frame top layer, and
// then run this line of script.
var eType = fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].elementType; // eType =
instance
```

下面的示例显示当前图层或帧中所有元素的多个属性:

```
var tl = fl.getDocumentDOM().getTimeline()
var elts = tl.layers[tl.currentLayer].frames[tl.currentFrame].elements;
for (var x = 0; x < elts.length; x++) {
    var elt = elts[x];
    fl.trace("Element "+ x +" Name = " + elt.name + " Type = " + elt.elementType + " location = " + elt.left
+ "," + elt.top + " Depth = " + elt.depth);
}
```

## element.getPersistentData()

可用性

Flash MX 2004。

用法

```
element.getPersistentData(name)
```

参数

**name** 一个字符串，它标识要返回的数据。

返回

**name** 参数所指定的数据；如果该数据不存在，则返回 0。

说明

方法；检索 **name** 参数所指定的数据的值。数据的类型取决于此前所存储数据的类型（请参阅 [element.setPersistentData\(\)](#)）。只有元件和位图才支持永久数据。

示例

下面的示例设置并获取指定元素的数据，在“输出”面板中显示其值，然后删除该数据：

```
// At least one symbol or bitmap is selected in the first layer, first frame.
var elt = fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0];
elt.setPersistentData("myData","integer", 12);
if (elt.hasPersistentData("myData")){
fl.trace("myData = "+ elt.getPersistentData("myData"));
elt.removePersistentData( "myData" );
fl.trace("myData = "+ elt.getPersistentData("myData"));
}
```

## element.getTransformationPoint()

可用性

Flash CS3 Professional。

## Element 对象

## 用法

```
element.getTransformationPoint()
```

## 参数

无。

## 返回

一个点（例如 {x:10,y:20}，其中 x 和 y 是浮点数），它指定元素坐标系内变形点（即原点或零点）的位置。

## 说明

方法；获取指定元素的变形点的值。

变形点是相对于不同位置而言的，具体取决于所选项的类型。有关详细信息，请参阅 [element.setTransformationPoint\(\)](#)。

## 示例

下面的示例获取文档中第一图层第九帧中的第三个元素的变形点。`transPoint.x` 属性提供变形点的 x 坐标。`transPoint.y` 属性提供变形点的 y 坐标。

```
var transPoint =  
fl.getDocumentDOM().getTimeline().layers[0].frames[8].elements[2].getTransformationPoint();
```

## 另请参见

[document.getTransformationPoint\(\)](#)、[element.setTransformationPoint\(\)](#)、[element.transformX](#)、[element.transformY](#)

## element.hasPersistentData()

## 可用性

Flash MX 2004。

## 用法

```
element.hasPersistentData(name)
```

## 参数

**name** 一个字符串，它指定待测试数据项的名称。

## 返回

一个布尔值：如果指定的数据附加到该对象，则为 `true`；否则为 `false`。

## 说明

方法；确定指定数据是否附加到指定元素。只有元件和位图才支持永久数据。

## 示例

请参阅 [element.getPersistentData\(\)](#)。

## element.height

可用性

Flash MX 2004。

用法

`element.height`

描述

属性；一个浮点值，它指定元素的高度（单位为像素）。

请不要使用此属性来调整文本字段的大小，而应选择此文本字段并使用 `document.setTextRectangle()`。将此属性用于文本字段时会缩放文本。

示例

下面的示例将指定元素的高度设置为 100：

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].height = 100;
```

## element.layer

可用性

Flash 8。

用法

`element.layer`

描述

只读属性；它表示元素所在的 [Layer 对象](#)。

示例

下面的示例将包含元素的 `Layer` 对象保存在 `theLayer` 变量中：

```
var theLayer = element.layer;
```

## element.left

可用性

Flash MX 2004。

用法

`element.left`

描述

只读属性；一个浮点值，它表示元素的左侧。对于场景中的元素，`element.left` 的值相对于舞台的左上角；如果元素存储在元件中，则该值相对于元件的注册点（即原点或零点）。请使用 `document.setSelectionBounds()` 或 `document.moveSelectionBy()` 设置此属性。

## 示例

下面的示例演示了移动元素时此属性的值如何更改:

```
// Select an element on the Stage and then run this script.
var sel = fl.getDocumentDOM().selection[0];
fl.trace("Left (before) = " + sel.left);
fl.getDocumentDOM().moveSelectionBy({x:100, y:0});
fl.trace("Left (after) = " + sel.left);
```

请参阅 [element.elementType](#) 示例。

## element.locked

## 可用性

Flash MX 2004。

## 用法

```
element.locked
```

## 描述

属性; 一个布尔值: 如果元素被锁定, 则为 `true`; 否则为 `false`。如果 [element.elementType](#) 的值为 "shape", 则忽略此属性。

## 示例

下面的示例锁定顶部图层的第一帧中的第一个元素:

```
// Similar to Modify > Arrange > Lock:
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].locked = true;
```

## element.matrix

## 可用性

Flash MX 2004。

## 用法

```
element.matrix
```

## 描述

属性; 一个 `Matrix` 对象。 `matrix` 的属性包括 `a`、`b`、`c`、`d`、`tx` 和 `ty`。`a`、`b`、`c` 和 `d` 属性是浮点值; `tx` 和 `ty` 属性是坐标。请参阅 [Matrix 对象](#)。

## 示例

下面的示例将指定元素在 `x` 轴方向上移动 10 个像素, 在 `y` 轴方向上移动 20 个像素:

```
var mat = fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].matrix;
mat.tx += 10;
mat.ty += 20;
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].matrix = mat;
```

## element.name

可用性

Flash MX 2004。

用法

`element.name`

描述

属性；一个字符串，它指定元素的名称，通常称为实例名称。如果 `element.elementType` 的值为 "shape"，则忽略此属性。请参阅 [element.elementType](#)。

示例

下面的示例将顶部图层的第 1 帧中的第一个元素的实例名称设置为 "clip\_mc"：

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].name = "clip_mc";
```

请参阅 [element.elementType](#) 示例。

## element.removePersistentData()

可用性

Flash MX 2004。

用法

`element.removePersistentData(name)`

参数

**name** 一个字符串，它指定要删除的数据的名称。

返回

无。

描述

方法；它删除具有已附加到该对象的指定名称的任何永久性数据。只有元件和位图才支持永久数据。

示例

请参阅 [element.getPersistentData\(\)](#)。

## element.rotation

可用性

Flash CS3 Professional。

用法

`element.rotation`

**Element 对象**

## 描述

属性；一个介于 -180 和 180 之间的整数或浮点值，它指定对象的顺时针旋转，以度为单位。

## 示例

下面的示例将当前所选元素设置为旋转 45 度：

```
var element = fl.getDocumentDOM().selection[0];
fl.trace("Element rotation = " + element.rotation);
element.rotation = 45;
fl.trace("After setting rotation to 45: rotation = " + element.rotation);
```

## element.scaleX

## 可用性

Flash CS3 Professional。

## 用法

```
element.scaleX
```

## 描述

属性；一个浮点值，它指定了元件、绘制对象、矩形图元和椭圆图元的 **x** 缩放值。值 1 表示 100% 缩放。

## 示例

下面的示例将当前所选内容的 **x** 缩放值设置为 2（将其值扩大一倍）：

```
var element = fl.getDocumentDOM().selection[0];
element.scaleX = 2;
```

另请参见

[element.scaleY](#)

## element.scaleY

## 可用性

Flash CS3 Professional。

## 用法

```
element.scaleY
```

## 描述

属性；一个浮点值，它指定了元件、绘制对象、矩形图元和椭圆图元的 **y** 缩放值。值 1 表示 100% 缩放。

## 示例

下面的示例将当前所选内容的 **y** 缩放值设置为 2（将其值扩大一倍）：

```
var element = fl.getDocumentDOM().selection[0];
element.scaleY = 2;
```



另请参见  
[element.scaleX](#)

## element.selected

可用性  
Flash 8。

用法  
`element.selected`

描述  
属性；一个布尔值，它指定元素是 (**true**) 否 (**false**) 已选中。

示例  
下面的示例选中元素：

```
element.selected = true;
```

## element.setPersistentData()

可用性  
Flash MX 2004。

用法  
`element.setPersistentData(name, type, value)`

参数  
**name** 一个字符串，它指定与该数据关联的名称。此名称用于检索数据。  
**type** 一个字符串，它定义数据的类型。允许的值为 "integer"、"integerArray"、"double"、"doubleArray"、"string" 和 "byteArray"。  
**value** 指定与该对象关联的值。value 的数据类型取决于 type 参数的值。指定的值应对应于由 type 参数指定的数据类型。

返回  
无。

描述  
方法；存储包含元素的数据。该数据在重新打开包含该元素的 FLA 文件时可用。只有元件和位图才支持永久数据。

示例  
请参阅 [element.getPersistentData\(\)](#)。

## element.setTransformationPoint()

可用性

Flash CS3 Professional。

用法

```
element.setTransformationPoint(transformationPoint)
```

参数

**transformationPoint** 一个点（例如 {x:10,y:20}，其中 x 和 y 是浮点数），它指定元素或组的变形点的值。

- 形状：**transformationPoint** 是相对于文档设置的（0,0 为舞台的左上角）。
- 元件：**transformationPoint** 是相对于元件的注册点设置的（0,0 位于注册点）。
- 文本：**transformationPoint** 是相对于文本字段来设置的（0,0 表示文本字段的左上角）。
- 位图 / 视频：**transformationPoint** 是相对于位图 / 视频设置的（0,0 为位图或视频的左上角）。
- 绘制对象、原始对象和组：**transformationPoint** 是相对于元素或组的中心设置的（0,0 为元素或组的中心点）。

返回

无。

描述

方法；设置元素的变形点的位置。

此方法与 [document.setTransformationPoint\(\)](#) 几乎完全相同。在以下几方面存在不同之处：

- 绘制对象、原始对象和组的变形点是相对于元素或组的中心设置的，而不是相对于舞台设置的。
- 您在设置元素的变形点之前无需选择元素。

此方法将移动变形点，但不会移动元素。相反，[element.transformX](#) 和 [element.transformY](#) 属性移动元素。

示例

下面的示例将舞台上第三个元素的变形点设置为 100, 200：

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[2].setTransformationPoint({x:100, y:200});
```

另请参见

[document.setTransformationPoint\(\)](#)、[element.getTransformationPoint\(\)](#)、[element.transformX](#)、[element.transformY](#)

## element.skewX

可用性

Flash CS3 Professional。

用法

```
element.skewX
```

描述

属性；一个介于 -180 和 180 之间的浮点值，它指定元件、绘制对象及基本矩形和基本椭圆的 x 倾斜值。

## Element 对象

## 示例

下面的示例将当前所选内容的 **x** 倾斜值设置为 10:

```
var element = fl.getDocumentDOM().selection[0];
element.skewX = 10;
```

另请参见

[document.setTransformationPoint\(\)](#)、[element.skewY](#)

## element.skewY

## 可用性

Flash CS3 Professional。

## 用法

```
element.skewY
```

## 描述

属性；一个介于 -180 和 180 之间的浮点值，它指定元件、绘制对象及基本矩形和基本椭圆的 **y** 倾斜值。

## 示例

下面的示例将当前所选内容的 **y** 倾斜值设置为 10:

```
var element = fl.getDocumentDOM().selection[0];
element.skewY = 10;
```

另请参见

[document.setTransformationPoint\(\)](#)、[element.skewX](#)

## element.top

## 可用性

Flash MX 2004。

## 用法

```
element.top
```

## 描述

只读属性；元素的顶端。对于场景中的元素，**element.top** 的值相对于舞台的左上角；如果元素存储在元件中，则该值相对于元件的注册点。请使用 [document.setSelectionBounds\(\)](#) 或 [document.moveSelectionBy\(\)](#) 设置此属性。

## 示例

下面的示例显示了移动元素时此属性的值如何更改：

```
// Select an element on the Stage and then run this script.
var sel = fl.getDocumentDOM().selection[0];
fl.trace("Top (before) = " + sel.top);
fl.getDocumentDOM().moveSelectionBy({x:0, y:100});
fl.trace("Top (after) = " + sel.top);
```

请参阅 [element.elementType](#) 示例。

## element.transformX

可用性

Flash CS3 Professional。

用法

`element.transformX`

描述

属性；一个浮点数，它指定元素父级坐标系内所选元素变形点的 **x** 值。将此属性设置为新值可移动元素。相反，[element.setTransformationPoint\(\)](#) 方法移动变形点，但不移动元素。

示例

另请参见

[element.getTransformationPoint\(\)](#)、[element.setTransformationPoint\(\)](#)、[element.transformY](#)

## element.transformY

可用性

Flash CS3 Professional。

用法

`element.transformY`

描述

属性；一个浮点数，它指定元素父级坐标系内所选元素变形点的 **y** 值。将此属性设置为新值可移动元素。相反，[element.setTransformationPoint\(\)](#) 方法移动变形点，但不移动元素。

另请参见

[element.getTransformationPoint\(\)](#)、[element.setTransformationPoint\(\)](#)、[element.transformX](#)

## element.width

可用性

Flash MX 2004。

用法

`element.width`

**Element 对象**

## 描述

属性；一个浮点值，它指定元素的宽度（单位为像素）。

请不要使用此属性来调整文本字段的大小，而应选择此文本字段并使用 [document.setTextRectangle\(\)](#)。将此属性用于文本字段时会缩放文本。

## 示例

下面的示例将指定元素的宽度设置为 100:

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].width= 100;
```

## element.x

## 可用性

Flash CS3 Professional。

## 用法

`element.x`

## 描述

属性；一个浮点值，它指定所选元素注册点的 x 值。

## 示例

下面的示例将指定元素的注册点的值设置为 100, 200:

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].x= 100;  
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].y= 200;
```

另请参见

[element.y](#)

## element.y

## 可用性

Flash CS3 Professional。

## 用法

`element.y`

## 描述

属性；一个浮点值，它指定所选元素注册点的 y 值。

## 示例

请参阅 [element.x](#)

## 第 15 章 : Fill 对象

可用性

Flash MX 2004。

描述

此对象包含“工具”面板或所选形状的填充颜色设置的所有属性。要检索 Fill 对象，请使用 `document.getCustomFill()`。

属性摘要

Fill 对象具有以下属性：

属性	描述
<code>fill.bitmapIsClipped</code>	一个布尔值，它指定对于大于位图的形状，是裁剪还是重复该形状的位图填充。
<code>fill.bitmapPath</code>	一个字符串，它指定库中位图填充的路径和名称。
<code>fill.color</code>	一个字符串、十六进制值或整数，它表示填充的颜色。
<code>fill.colorArray</code>	渐变中颜色的数组。
<code>fill.focalPoint</code>	一个整数，它指定渐变焦点距离变形点的水平偏移量。
<code>fill.linearRGB</code>	一个布尔值，它指定是将填充呈现为线性 RGB 渐变，还是呈现为放射状 RGB 渐变。
<code>fill.matrix</code>	一个 <code>Matrix</code> 对象，它定义渐变填充的位置、方向和缩放。
<code>fill.overflow</code>	一个字符串，它指定渐变的溢出行为。
<code>fill.posArray</code>	一个整数数组，其中每个整数的值都介于 0 和 255 之间，表示对应颜色的位置。
<code>fill.style</code>	一个字符串，它指定填充样式。

### fill.bitmapIsClipped

可用性

Flash CS4 Professional.

用法

```
fill.bitmapIsClipped
```

描述

属性；一个布尔值，它指定对于大于位图的形状，是裁剪 (`true`) 还是重复 (`false`) 该形状的位图填充。仅在 `fill.style` 属性的值为 "bitmap" 时，此属性才可用。如果形状小于位图，则该值为 `false`。

示例

下面的示例在“输出”面板中显示有关是否根据情况裁剪位图填充的信息：

```
var fill = fl.getDocumentDOM().getCustomFill();
if (fill.style == "bitmap")
    fl.trace("Fill image is clipped: " + fill.bitmapIsClipped);
```

另请参见

[fill.bitmapPath](#)

## fill.bitmapPath

可用性

Flash CS4 Professional.

用法

```
fill.bitmapPath
```

描述

属性；一个字符串，它指定库中位图填充的路径和名称。仅在 [fill.style](#) 属性的值为 "bitmap" 时，此属性才可用。

示例

下面的示例将指定项目的填充样式设置为库中的位图图像：

```
var fill = fl.getDocumentDOM().getCustomFill();
fill.style = "bitmap";
fill.bitmapPath = "myBitmap.jpg";
fl.getDocumentDOM().setCustomFill(fill);
```

另请参见

[fill.bitmapIsClipped](#)

## fill.color

可用性

Flash MX 2004。

用法

```
fill.color
```

描述

属性；填充的颜色，使用以下格式之一：

- 格式为 "#RRGGBB" 或 "#RRGGBBAA" 的字符串
- 格式为 0xRRGGBB 的十六进制数字
- 表示与十六进制数字等价的十进制整数

示例

下面的示例设置当前选定对象的填充颜色：

```
var fill = fl.getDocumentDOM().getCustomFill();
fill.color = "#FFFFFF";
fl.getDocumentDOM().setCustomFill( fill );
```

## fill.colorArray

可用性

Flash MX 2004。

用法

```
fill.colorArray
```

描述

属性；渐变中的颜色数组，表示为整数。仅在 `fill.style` 属性的值为 "radialGradient" 或 "linearGradient" 时，此属性才可用。请参阅 [fill.style](#)

示例

下面的示例在“输出”面板中显示当前选定对象的颜色数组（如果合适）：

```
var fill = fl.getDocumentDOM().getCustomFill();  
if(fill.style == "linearGradient" || fill.style == "radialGradient")  
    alert(fill.colorArray);
```

下面的示例为指定的线性渐变设置填充：

```
var fill = fl.getDocumentDOM().getCustomFill();  
fill.style = "linearGradient";  
fill.colorArray = ["#00ff00", "#ff00ff"];  
fill.posArray = [0, 255];  
fl.getDocumentDOM().setCustomFill(fill);
```

## fill.focalPoint

可用性

Flash 8。

用法

```
fill.focalPoint
```

描述

属性；一个整数，它指定距离变形点的渐变焦点水平偏移量。例如，值为 10 表示将焦点放置在从变形点到渐变边缘距离为 10/255 的地方。值为 -255 表示将焦点位置放置在渐变的左边界。默认值为 0。

仅在 `fill.style` 属性的值为 "radialGradient" 时，此属性才可用。

示例

下面的示例将当前选定对象的放射状渐变的焦点位置设置在距离形状中心右侧 100 像素处：

```
var fill = fl.getDocumentDOM().getCustomFill();  
fill.style = "radialGradient";  
fill.colorArray = ["#00ff00", "#ff00ff"];  
fill.posArray = [0, 255];  
fill.focalPoint = 10100;  
fl.getDocumentDOM().setCustomFill(fill);
```



## fill.linearRGB

可用性

Flash 8。

用法

`fill.linearRGB`

描述

属性；一个布尔值，它指定填充是呈现为线性 RGB 还是放射状 RGB 渐变。将此属性设置为 `true` 时指定渐变为线性插入，设置为 `false` 则指定渐变为放射状插入。默认值为 `false`。

示例

下面的示例指定应使用线性 RGB 来呈现当前选区的渐变：

```
var fill = fl.getDocumentDOM().getCustomFill();
fill.linearRGB style = true"radialGradient";
fill.colorArray = ["#00ff00", "#ff00ff"];
fill.posArray = [0, 255];
fill.focalPoint = 100;
fill.linearRGB = true;
fl.getDocumentDOM().setCustomFill(fill);
```

## fill.matrix

可用性

Flash MX 2004。

用法

`fill.matrix`

描述

属性；[Matrix 对象](#)，它定义渐变填充的位置、方向和缩放。

示例

下面的示例使用 `fill.matrix` 属性为当前选择指定渐变填充：

## Fill 对象

```
var fill = fl.getDocumentDOM().getCustomFill();
fill.style = 'radialGradient';
fill.colorArray = ['#00ff00', '#ff00ff'];
fill.posArray = [0, 255];
fill.focalPoint = 100;
fill.linearRGB = false;
fill.overflow = 'repeat';
var mat = fl.getDocumentDOM().selection[0].matrix;
mat.a = 0.0167083740234375;
mat.b = -0.0096435546875;
mat.c = 0.0312957763671875;
mat.d = 0.05419921875;
mat.tx = 288.65;
mat.ty = 193.05;
for (i in mat) {
    fl.trace(i+' : '+mat[i]);
}
fl.getDocumentDOM().setCustomFill(fill);
```

## fill.overflow

可用性

Flash 8。

用法

```
fill.overflow
```

描述

属性；一个字符串，它指定渐变溢出的行为。可接受值为 "extend"、"repeat" 和 "reflect"；这些字符串不区分大小写。默认值为 "extend"。

示例

下面的示例指定当前选择的溢出行为应该是 "extend"：

```
var fill = fl.getDocumentDOM().getCustomFill();
fill.overflow = "extend";
fl.getDocumentDOM().setCustomFill(fill);
```

## fill.posArray

可用性

Flash MX 2004。

用法

```
fill.posArray
```

描述

属性；一个整数数组，每个数组元素介于 0 和 255 之间，表示对应颜色的位置。仅在 fill.style 属性的值为 "radialGradient" 或 "linearGradient" 时，此属性才可用。

#### 示例

下面的示例指定当前选定对象的线性渐变要使用的颜色：

```
var fill = fl.getDocumentDOM().getCustomFill();  
fill.style = "linearGradient";  
fill.colorArray = [ 0x00ff00, 0xff0000, 0x0000ff ];  
fill.posArray= [0,100, 200];  
fl.getDocumentDOM().setCustomFill( fill );
```

## fill.style

#### 可用性

Flash MX 2004。Flash CS4 Professional 中新增了“bitmap”一值。

#### 用法

`fill.style`

#### 描述

属性；一个字符串，它指定填充样式。可接受的值为 `bitmap`、`solid`、`linearGradient`、`radialGradient` 和 `noFill`。

如果该值为“`linearGradient`”或“`radialGradient`”，则 `fill.colorArray` 和 `fill.posArray` 属性也可用。如果该值为“`bitmap`”，则 `fill.bitmapIsClipped` 和 `fill.bitmapPath` 属性也可用。

#### 示例

下面的示例指定当前选定对象的线性渐变要使用的颜色：

```
var fill = fl.getDocumentDOM().getCustomFill();  
fill.style= "linearGradient";  
fill.colorArray = [ 0x00ff00, 0xff0000, 0x0000ff ];  
fill.posArray= [0,100, 200];  
fl.getDocumentDOM().setCustomFill( fill );
```

## 第 16 章 : Filter 对象

可用性  
Flash 8。

说明

该对象包含所有滤镜的全部属性。`filter.name` 属性指定滤镜的类型，并确定适用于每个滤镜的属性。请参阅 [filter.name](#)。

要为一个或多个对象返回滤镜列表，请使用 `document.getFilters()`。要为一个或多个对象应用滤镜，请使用 `document.setFilters()`。请参阅 [document.getFilters\(\)](#) 和 [document.setFilters\(\)](#)。

属性摘要

以下属性可用于 Filter 对象：

属性	描述
<a href="#">filter.angle</a>	一个浮点值，它指定阴影或加亮颜色的角度（以度为单位）。
<a href="#">filter.blurX</a>	一个浮点值，它指定 <b>x</b> 方向的模糊量（单位为像素）。
<a href="#">filter.blurY</a>	一个浮点值，它指定 <b>y</b> 方向的模糊量。
<a href="#">filter.brightness</a>	一个浮点值，它指定滤镜的亮度。
<a href="#">filter.color</a>	一个字符串、十六进制值或整数，它表示滤镜的颜色。
<a href="#">filter.contrast</a>	一个浮点值，它指定滤镜对比度的值。
<a href="#">filter.distance</a>	一个浮点值，它指定滤镜效果和某个对象之间的距离（以像素为单位）。
<a href="#">filter.enabled</a>	布尔值，指定是否已启用指定的滤镜。
<a href="#">filter.hideObject</a>	布尔值，指定源图像是否处于隐藏状态。
<a href="#">filter.highlightColor</a>	一个字符串、十六进制值或整数，用于表示加亮的颜色。
<a href="#">filter.hue</a>	一个浮点值，它指定滤镜的色相。
<a href="#">filter.inner</a>	一个布尔值，指定阴影是否为内侧阴影。
<a href="#">filter.knockout</a>	布尔值，指定滤镜是否为挖空滤镜。
<a href="#">filter.name</a>	只读；一个指定滤镜类型的字符串。
<a href="#">filter.quality</a>	一个字符串，它指定模糊质量。
<a href="#">filter.saturation</a>	一个浮点值，它指定滤镜饱和度的值。
<a href="#">filter.shadowColor</a>	一个字符串、十六进制值或整数，它表示阴影的颜色。
<a href="#">filter.strength</a>	一个整数，它指定滤镜的百分比强度。
<a href="#">filter.type</a>	一个字符串，它指定斜角或发光的类型。

## filter.angle

可用性

Flash 8。

用法

`filter.angle`

描述

属性；一个浮点值，它指定阴影或加亮颜色的角度（以度为单位）。可接受的值为 0 和 360 之间的值。该属性是为其 `filter.name` 属性值为 "bevelFilter"、"dropShadowFilter"、"gradientBevelFilter" 或 "gradientGlowFilter" 的 Filter 对象定义的。

示例

下面的示例在所选对象上将“斜角”滤镜的角度设置为 120：

```
var myFilters = fl.getDocumentDOM().getFilters();
for(i=0; i < myFilters.length; i++) {
    if(myFilters[i].name == 'bevelFilter'){
        myFilters[i].angle = 120;
    }
}
fl.getDocumentDOM().setFilters(myFilters);
```

另请参见

[document.setFilterProperty\(\)](#)

## filter.blurX

可用性

Flash 8。

用法

`filter.blurX`

描述

属性；一个浮点值，它指定 x 方向的模糊量（单位为像素）。可接受的值为 0 和 255 之间的值。此属性是为 `filter.name` 属性值为 bevelFilter、blurFilter、dropShadowFilter、glowFilter、gradientBevelFilter 或 gradientGlowFilter 的 Filter 对象定义的。

示例

下面的示例在所选对象上将模糊滤镜的 blurX 值设为 30，并将 blurY 值设为 20：

```
var myFilters = fl.getDocumentDOM().getFilters();
for(i=0; i < myFilters.length; i++){
    if(myFilters[i].name == 'blurFilter'){
        myFilters[i].blurX = 30;
        myFilters[i].blurY = 20;
    }
}
fl.getDocumentDOM().setFilters(myFilters);
```

另请参见

[document.setFilterProperty\(\)](#)、[filter.blurY](#)

## filter.blurY

可用性

Flash 8。

用法

`filter.blurY`

描述

属性；一个浮点值，它指定 y 方向的模糊量（单位为像素）。可接受的值为 0 和 255 之间的值。此属性是为 `filter.name` 属性值为 `bevelFilter`、`blurFilter`、`dropShadowFilter`、`glowFilter`、`gradientBevelFilter` 或 `gradientGlowFilter` 的 `Filter` 对象定义的。

示例

请参阅 [filter.blurX](#)。

另请参见

[document.setFilterProperty\(\)](#)、[filter.blurX](#)

## filter.brightness

可用性

Flash 8。

用法

`filter.brightness`

描述

属性；一个浮点值，它指定滤镜的亮度。可接受的值为 -100 和 100 之间的值。此属性是为其 `filter.name` 属性值为 "adjustColorFilter" 的 `Filter` 对象定义的。

示例

下面的示例在选定对象上将“调整颜色”滤镜的亮度设置为 30.5：

```
var myFilters = fl.getDocumentDOM().getFilters();
for(i=0; i < myFilters.length; i++){
    if(myFilters[i].name == 'adjustColorFilter'){
        myFilters[i].brightness = 30.5;
    }
}
fl.getDocumentDOM().setFilters(myFilters);
```

## filter.color

可用性

Flash 8。

用法

`filter.color`

描述

属性；滤镜的颜色；使用以下格式之一：

- 格式为 "#RRGGBB" 或 "#RRGGBBAA" 的字符串
- 格式为 0xRRGGBB 的十六进制数字
- 表示与十六进制数字等价的十进制整数

此属性是为其 [filter.name](#) 属性值为 "dropShadowFilter" 或 "glowFilter" 的 Filter 对象定义的。

示例

下面的示例在所选对象将“投影”滤镜的颜色设为 "#ff00003e"：

```
var myFilters = fl.getDocumentDOM().getFilters();
for(i=0; i < myFilters.length; i++){
    if(myFilters[i].name == 'dropShadowFilter'){
        myFilters[i].color = '#ff00003e';
    }
}
fl.getDocumentDOM().setFilters(myFilters);
```

另请参见

[document.setFilterProperty\(\)](#)

## filter.contrast

可用性

Flash 8。

用法

`filter.contrast`

描述

属性；一个浮点值，它指定滤镜对比度的值。可接受的值为 -100 和 100 之间的值。此属性是为其 [filter.name](#) 属性值为 "adjustColorFilter" 的 Filter 对象定义的。

示例

下面的示例在所选对象上将调整颜色滤镜的对比度值设为 -15.5：

**Filter 对象**

```
var myFilters = fl.getDocumentDOM().getFilters();
for(i=0; i < myFilters.length; i++){
    if(myFilters[i].name == 'adjustColorFilter'){
        myFilters[i].contrast = -15.5;
    }
}
fl.getDocumentDOM().setFilters(myFilters);
```

## filter.distance

可用性

Flash 8。

用法

`filter.distance`

描述

属性；一个浮点值，它指定滤镜效果和某个对象之间的距离（以像素为单位）。可接受的值为 -255 和 255 之间的值。该属性是为其 `filter.name` 属性值为 "bevelFilter"、"dropShadowFilter"、"gradientBevelFilter" 或 "gradientGlowFilter" 的 **Filter** 对象定义的。

示例

下面的示例在所选对象上将“投影”滤镜的距离设为 10 像素：

```
var myFilters = fl.getDocumentDOM().getFilters();
for(i=0; i < myFilters.length; i++){
    if(myFilters[i].name == 'dropShadowFilter'){
        myFilters[i].distance = 10;
    }
}
fl.getDocumentDOM().setFilters(myFilters);
```

另请参见

[document.setFilterProperty\(\)](#)

## filter.enabled

可用性

Flash CS3 Professional。

用法

`filter.enabled`

描述

属性；一个布尔值，它指定所指定的滤镜是否处于启用状态，如果是则为 `true`，否则为 `false`。

示例

下面的示例禁用选定对象的“颜色”滤镜：



**Filter 对象**

```
var myFilters = fl.getDocumentDOM().getFilters();
for(i=0; i < myFilters.length; i++){
    if(myFilters[i].name == 'adjustColorFilter'){
        myFilters[i].enabled = false;
    }
}
fl.getDocumentDOM().setFilters(myFilters);
```

## filter.hideObject

可用性

Flash 8。

用法

`filter.hideObject`

描述

属性；一个布尔值，它指定源图像是隐藏 (**true**)，还是显示 (**false**)。该属性是为其 `filter.name` 属性值为 "dropShadowFilter" 的 **Filter** 对象定义的。

示例

下面的示例在所选对象上将“投影”滤镜的 `hideObject` 值设为 `true`：

```
var myFilters = fl.getDocumentDOM().getFilters();
for(i=0; i < myFilters.length; i++){
    if(myFilters[i].name == 'dropShadowFilter'){
        myFilters[i].hideObject = true;
    }
}
fl.getDocumentDOM().setFilters(myFilters);
```

## filter.highlightColor

可用性

Flash 8。

用法

`filter.highlightColor`

描述

属性；加亮的颜色，使用以下格式之一：

- 格式为 "#RRGGBB" 或 "#RRGGBBAA" 的字符串
- 格式为 0xRRGGBB 的十六进制数字
- 表示与十六进制数字等价的十进制整数

该属性是为其 `filter.name` 属性值为 "bevelFilter" 的 **Filter** 对象定义的。

**Filter 对象**

## 示例

下面的示例在所选对象上将“斜角”滤镜的加亮颜色设为 "#ff00003e":

```
var myFilters = fl.getDocumentDOM().getFilters();
for(i=0; i < myFilters.length; i++){
    if(myFilters[i].name == 'bevelFilter'){
        myFilters[i].highlightColor = '#ff00003e';
    }
}
fl.getDocumentDOM().setFilters(myFilters);
```

## filter.hue

## 可用性

Flash 8。

## 用法

`filter.hue`

## 描述

属性；一个浮点值，它指定滤镜的色相。可接受的值为 -180 和 180 之间的值。此属性是为其 `filter.name` 属性值为 "adjustColorFilter" 的 **Filter** 对象定义的。

## 示例

下面的示例在所选对象上将“调整颜色”滤镜的色相设为 120:

```
var myFilters = fl.getDocumentDOM().getFilters();
for(i=0; i < myFilters.length; i++){
    if(myFilters[i].name == 'adjustColorFilter'){
        myFilters[i].hue = 120;
    }
}
fl.getDocumentDOM().setFilters(myFilters);
```

## filter.inner

## 可用性

Flash 8。

## 用法

`filter.inner`

## 描述

属性；一个布尔值，它指定阴影是否是内阴影，如果是则为 `true`，否则为 `false`。此属性是为其 `filter.name` 属性值为 "dropShadowFilter" 或 "glowFilter" 的 **Filter** 对象定义的。

## 示例

下面的示例在所选对象上将“发光”滤镜的 `inner` 属性值设为 `true`:

```
var myFilters = fl.getDocumentDOM().getFilters();
for(i=0; i < myFilters.length; i++){
    if(myFilters[i].name == 'glowFilter'){
        myFilters[i].inner = true;
    }
}
fl.getDocumentDOM().setFilters(myFilters);
```

另请参见

[document.setFilterProperty\(\)](#)

## filter.knockout

可用性

Flash 8。

用法

`filter.knockout`

描述

属性；一个布尔值，它指定滤镜是否是挖空滤镜，如果是则为 `true`，否则为 `false`。此属性是为其 `filter.name` 属性值为 "bevelFilter"、"dropShadowFilter"、"glowFilter"、"gradientBevelFilter" 或 "gradientGlowFilter" 的 **Filter** 对象定义的。

示例

下面的示例在所选对象上将“发光”滤镜的 `knockout` 属性值设为 `true`：

```
var myFilters = fl.getDocumentDOM().getFilters();
for(i=0; i < myFilters.length; i++){
    if(myFilters[i].name == 'glowFilter'){
        myFilters[i].knockout = true;
    }
}
fl.getDocumentDOM().setFilters(myFilters);
```

另请参见

[document.setFilterProperty\(\)](#)

## filter.name

可用性

Flash 8。

用法

`filter.name`

描述

只读属性；一个字符串，它指定滤镜的类型。该属性的值可确定此 **Filter** 对象还有哪些可用的属性。其值是以下值之一：  
"adjustColorFilter"、"bevelFilter"、"blurFilter"、"dropShadowFilter"、"glowFilter"、"gradientBevelFilter" 或 "gradientGlowFilter"。

示例

下面的示例在“输出”面板中显示滤镜的名称和索引的位置：

```
var myFilters = fl.getDocumentDOM().getFilters();
var traceStr = "";
for(i=0; i < myFilters.length; i++){
    traceStr = traceStr + " At index " + i + ": " + myFilters[i].name;
}
fl.trace(traceStr);
```

另请参见

[document.getFilters\(\)](#)、[document.setFilterProperty\(\)](#)

## filter.quality

可用性

Flash 8。

用法

`filter.quality`

描述

属性；一个字符串，它指定模糊质量。可接受的值为 "low"、"medium" 和 "high" ("high" 与高斯模糊相似)。此属性是为其 `filter.name` 属性值为 "bevelFilter"、"blurFilter"、"dropShadowFilter"、"glowFilter"、"gradientGlowFilter" 或 "gradientBevelFilter" 的 `Filter` 对象定义的。

示例

下面的示例在所选对象上将“发光”滤镜的模糊品质设为 "medium"：

```
var myFilters = fl.getDocumentDOM().getFilters();
for(i=0; i < myFilters.length; i++){
    if(myFilters[i].name == 'glowFilter'){
        myFilters[i].quality = 'medium';
    }
}
fl.getDocumentDOM().setFilters(myFilters);
```

另请参见

[document.setFilterProperty\(\)](#)

## filter.saturation

可用性

Flash 8。

用法

`filter.saturation`

**Filter 对象**

## 描述

属性；一个浮点值，它指定滤镜饱和度的值。可接受的值为 -100 至 100。此属性是为其 [filter.name](#) 属性值为 "adjustColorFilter" 的 **Filter** 对象定义的。

## 示例

下面的示例在所选对象上将调整颜色滤镜的饱和度值设为 -100（灰度）：

```
var myFilters = fl.getDocumentDOM().getFilters();
for(i=0; i < myFilters.length; i++){
    if(myFilters[i].name == 'adjustColorFilter'){
        myFilters[i].saturation = 0-100;
    }
}
fl.getDocumentDOM().setFilters(myFilters);
```

另请参见

[document.setFilterProperty\(\)](#)

## filter.shadowColor

可用性

Flash 8。

用法

`filter.shadowColor`

描述

属性；阴影的颜色，使用以下格式之一：

- 格式为 "#RRGGBB" 或 "#RRGGBBAA" 的字符串
- 格式为 0xRRGGBB 的十六进制数字
- 表示与十六进制数字等价的十进制整数

该属性是为其 [filter.name](#) 属性值为 "bevelFilter" 的 **Filter** 对象定义的。

示例

下面的示例在所选对象上将“斜角”滤镜的阴影颜色设为 "#ff00003e"：

```
var myFilters = fl.getDocumentDOM().getFilters();
for(i=0; i < myFilters.length; i++){
    if(myFilters[i].name == 'bevelFilter'){
        myFilters[i].shadowColor = '#ff00003e';
    }
}
fl.getDocumentDOM().setFilters(myFilters);
```

另请参见

[document.setFilterProperty\(\)](#)

## filter.strength

可用性

Flash 8。

用法

`filter.strength`

描述

属性；一个整数，它指定滤镜的百分比强度。可接受的值为 0 和 25,500 之间的值。此属性是为其 `filter.name` 属性值为 "bevelFilter"、"dropShadowFilter"、"glowFilter"、"gradientGlowFilter" 或 "gradientBevelFilter" 的 Filter 对象定义的。

示例

下面的示例在所选对象上将“发光”滤镜的强度设为 50：

```
var myFilters = fl.getDocumentDOM().getFilters();
for(i=0; i < myFilters.length; i++){
    if(myFilters[i].name == 'glowFilter'){
        myFilters[i].strength = 50;
    }
}
fl.getDocumentDOM().setFilters(myFilters);
```

另请参见

[document.setFilterProperty\(\)](#)

## filter.type

可用性

Flash 8。

用法

`filter.type`

描述

属性；一个字符串，它指定斜角或发光的类型。可接受的值为 "inner"、"outer" 和 "full"。此属性是为其 `filter.name` 属性值为 "bevelFilter"、"gradientGlowFilter" 或 "gradientBevelFilter" 的 Filter 对象定义的。

示例

下面的示例在所选对象上将所有“斜角”滤镜的类型设为 "full"：

**Filter 对象**

```
var myFilters = fl.getDocumentDOM().getFilters();
for(i=0; i < myFilters.length; i++){
    if(myFilters[i].name == 'bevelFilter'){
        myFilters[i].type = 'full';
    }
}
fl.getDocumentDOM().setFilters(myFilters);
```

另请参见

[document.setFilterProperty\(\)](#)

## 第 17 章 : flash 对象 (fl)

可用性

Flash MX 2004。

描述

**flash** 对象表示 Flash 应用程序。您可以使用 **flash** 或 **fl** 来指代此对象。本文档在代码示例中始终采用 **fl**。

方法摘要

可以对 **flash** 对象使用以下方法：

方法	描述
<a href="#">fl.addEventListener()</a>	注册在接收特定事件时要调用的函数。
<a href="#">fl.browseForFileURL()</a>	打开“打开文件”或“保存文件”系统对话框，允许用户指定要打开或保存的文件。
<a href="#">fl.browseForFolderURL()</a>	显示“浏览文件夹”对话框，允许用户选择文件夹。
第 205 页的“ <a href="#">fl.clearPublishCache()</a> ”	清除发布缓存。
<a href="#">fl.clipCopyString()</a>	将指定的字符串复制到剪贴板。
<a href="#">fl.closeAll()</a>	关闭所有打开的文档，为此前没有保存的所有文档显示“另存为”对话框。
<a href="#">fl.closeAllPlayerDocuments()</a>	关闭用“控制”>“测试影片”打开的所有 SWF 文件。
<a href="#">fl.closeDocument()</a>	关闭指定的文档。
<a href="#">fl.createDocument()</a>	打开并选择新文档。
第 212 页的“ <a href="#">fl.exportPublishProfileString()</a> ”	要从中导出发布设置的统一资源标识符 (URI)。
<a href="#">fl.fileExists()</a>	检查文件是否已经存在于磁盘上。
<a href="#">fl.findDocumentDOM()</a>	允许您使用文件的唯一标识符找到特定文件。
<a href="#">fl.findDocumentIndex()</a>	返回整数的数组，这些整数表示文档在 <b>fl.documents</b> 数组中的位置。
<a href="#">fl.findObjectInDocByName()</a>	公开其实例名称与指定文本相匹配的元素。
<a href="#">fl.findObjectInDocByType()</a>	公开文档中具有指定元素类型的元素。
<a href="#">fl.getAppMemoryInfo()</a>	返回一个整数，它表示所指定 <b>Flash.exe</b> 内存区域内正在使用的字节数。
<a href="#">fl.getDocumentDOM()</a>	检索当前活动文档的 DOM ( <b>Document</b> 对象)。
<a href="#">fl.getSwfPanel()</a>	基于面板的本地化名称或其 SWF 文件名返回 <b>SWFPanel</b> 对象。
<a href="#">fl.isFontInstalled()</a>	确定是否已安装指定的字体。
<a href="#">fl.mapPlayerURL()</a>	将转义的 Unicode URL 映射到 UTF-8 或 MBCS URL。
<a href="#">fl.openDocument()</a>	在新的 Flash 文档窗口中打开一个 Flash (FLA) 文档供编辑，并使其获得焦点。
<a href="#">fl.openScript()</a>	在 Flash 文本编辑器中打开脚本 (JSFL、AS 或 ASC) 或其他文件 (XML 或 TXT)。
<a href="#">fl.quit()</a>	退出 Flash，并提示用户保存所有已更改的文档。



## flash 对象 (fl)

方法	描述
<a href="#">fl.reloadTools()</a>	从 <code>toolconfig.xml</code> 文件中重新构建“工具”面板。仅在创建可扩展工具时使用。
<a href="#">fl.removeEventListener()</a>	注销使用 <code>fl.addEventListener()</code> 注册的函数。
<a href="#">fl.resetAS3PackagePaths()</a>	将“ActionScript 3.0 设置”对话框中的全局“类路径”设置重置为默认值。
<a href="#">fl.resetPackagePaths()</a>	将“ActionScript 2.0 设置”对话框中的全局“类路径”设置重置为默认值。
<a href="#">fl.runScript()</a>	执行 JavaScript 文件。
<a href="#">fl.saveAll()</a>	保存所有打开的文档，为此前没有保存的所有文档显示“另存为”对话框。
<a href="#">fl.saveDocument()</a>	将指定的文档保存为 FLA 文档。
<a href="#">fl.saveDocumentAs()</a>	为指定的文档显示“另存为”对话框。
<a href="#">fl.selectElement()</a>	启用元素的选择或编辑。
<a href="#">fl.selectTool()</a>	在“工具”面板中选择指定的工具。
<a href="#">fl.setActiveWindow()</a>	将活动窗口设置为指定文档。
<a href="#">fl.showIdleMessage()</a>	允许您禁用有关脚本运行时间过长的警告。
<a href="#">fl.toggleBreakpoint()</a>	在给定行切换给定 .as 文件的断点。
<a href="#">fl.trace()</a>	将文本字符串发送到“输出”面板。

## 属性摘要

可以对 `flash` 对象使用以下属性。

属性	描述
<a href="#">fl.actionsPanel</a>	只读；一个 <code>actionsPanel</code> 对象。
<a href="#">fl.as3PackagePaths</a>	一个字符串，与“ActionScript 3.0 设置”对话框中的全局“类路径”设置相对应。
<a href="#">fl.compilerErrors</a>	只读；一个 <code>compilerErrors</code> 对象。
<a href="#">fl.componentsPanel</a>	只读；一个 <code>componentsPanel</code> 对象，表示“组件”面板。
<a href="#">fl.configDirectory</a>	只读；一个字符串，它以平台专用路径的形式指定本地用户 <code>Configuration</code> 文件夹的完整路径。
<a href="#">fl.configURI</a>	只读；一个字符串，它以 <code>file:///</code> URI 的形式指定本地用户 <code>Configuration</code> 目录的完整路径。
<a href="#">fl.contactSensitiveSelection</a>	一个布尔值，它指定是否启用接触感应选择模式。
<a href="#">fl.createNewDocList</a>	只读；一个字符串的数组，它表示可创建的各种文档类型。
<a href="#">fl.createNewDocListType</a>	只读；一个字符串的数组，它表示可创建的文档类型的文件扩展名。
<a href="#">fl.createNewTemplateList</a>	只读；一个字符串的数组，它表示可创建的各种模板类型。
<a href="#">fl.documents</a>	只读；一个 <code>Document</code> 对象的数组（请参阅 <code>Document</code> 对象），它表示当前打开供编辑的文档（FLA 文件）。
<a href="#">fl.drawingLayer</a>	只读； <code>drawingLayer</code> 对象，如果用户希望在拖放时进行临时绘制，可扩展工具应使用此对象。
<a href="#">fl.externalLibraryPath</a>	一个字符串，其中包含全局 ActionScript 3.0 外部库路径中项目的列表，该列表指定用作运行时共享库的 SWC 文件的位置。

## flash 对象 (fl)

属性	描述
<a href="#">fl.flexSDKPath</a>	一个字符串，它指定 Flex SDK 文件夹的路径，其中包含 bin、frameworks、lib 和其他文件夹。
<a href="#">fl.installedPlayers</a>	返回与文档属性检查器中已安装的 Flash Player 列表对应的通用对象数组。
<a href="#">fl.languageCode</a>	返回标识应用程序用户界面区域设置的 5 个字符的代码。
<a href="#">fl.libraryPath</a>	一个字符串，其中包含全局 ActionScript 3.0 库路径中项目的列表，该列表指定 SWC 文件或包含 SWC 文件的文件夹的位置。
<a href="#">fl.Math</a>	只读；一个 Math 对象，它提供用于矩阵和点运算的方法。
<a href="#">fl.mruRecentFileList</a>	只读；Flash 创作工具管理的“最近使用的文档”(MRU) 列表内完整文件名的数组。
<a href="#">fl.mruRecentFileListType</a>	只读；Flash 创作工具管理的 MRU 列表内文件类型的数组。
<a href="#">fl.packagePaths</a>	一个字符串，与“ActionScript 2.0 设置”对话框中的全局“类路径”设置相对应。
<a href="#">fl.publishCacheDiskSizeMax</a>	一个整数，用于设置磁盘缓存大小限制首选项。
<a href="#">fl.publishCacheEnabled</a>	一个布尔值，用于设置是否启用发布缓存。
<a href="#">fl.publishCacheMemoryEntrySizeLimit</a>	一个整数属性，用于设置内存缓存项的最大大小首选项。
<a href="#">fl.publishCacheMemorySizeMax</a>	一个整数，用于设置内存缓存大小限制首选项。
<a href="#">fl.objectDrawingMode</a>	一个整数，它表示已启用的对象绘制模式。
<a href="#">fl.outputPanel</a>	只读；对 outputPanel 对象的引用。
<a href="#">fl.presetPanel</a>	只读；一个 presetPanel 对象。
<a href="#">fl.scriptURI</a>	只读；一个字符串，它表示当前运行的 JSFL 脚本的路径，形式为 file:/// URI。
<a href="#">fl.sourcePath</a>	一个字符串，其中包含全局 ActionScript 3.0 源路径中项目的列表，该列表指定 ActionScript 类文件的位置。
<a href="#">fl.swfPanels</a>	已注册 swfPanel 对象的数组（请参阅 swfPanel 对象）。
<a href="#">fl.tools</a>	只读；Tools 对象的数组。
<a href="#">fl.version</a>	只读；Flash 创作工具的长字符串版本（包括平台）。
<a href="#">fl.xmlui</a>	只读；一个 XMLUI 对象。

## fl.actionsPanel

可用性

Flash CS3 Professional。

用法

```
fl.actionsPanel
```

描述

只读属性；一个 actionsPanel 对象，表示当前显示的“动作”面板。有关使用此属性的信息，请参阅 [actionsPanel 对象](#)。

## fl.addEventListener()

可用性

Flash CS3 Professional。

用法

```
fl.addEventListener(eventType, callbackFunction)
```

参数

**eventType** 一个字符串，指定要传递给此回调函数的事件类型。可接受的值为 `documentNew`、`documentOpened`、`documentClosed`、`mouseMove`、`documentChanged`、`layerChanged` 和 `frameChanged`。

`documentChanged` 值并不意味着文档的内容已经更改；它意味着前景中现在有一个不同的文档。即 `fl.getDocumentDOM()` 将返回一个与该事件发生前返回的值不相同的值。

**callbackFunction** 每次发生事件时要执行的函数的名称。

返回

无。

描述

方法；注册在特定事件发生时要调用的函数。

使用此方法时，请注意，如果事件频繁发生（参数值为 `mouseMove` 时可能会存在此种情况）且函数需要很长时间才能运行，则您的应用程序可能已宕机或进入错误状态。

示例

下面的示例在文档关闭时在“输出”面板中显示一条消息：

```
myFunction = function () {  
    fl.trace('document was closed');  
}  
fl.addEventListener("documentClosed", myFunction);
```

另请参见

[fl.removeEventListener\(\)](#)

## fl.as3PackagePaths

可用性

Flash CS3 Professional。

用法

```
fl.as3PackagePaths
```

描述

属性；一个字符串，与“ActionScript 3.0 设置”对话框中的全局类路径设置相对应。该字符串中的项目以分号分隔。若要查看或更改 ActionScript 2.0 类路径设置，请使用 [fl.packagePaths](#)。

**flash 对象 (fl)**

## 示例

下面的示例说明如何更改 **ActionScript 3.0** 类路径设置。

```
fl.trace(fl.as3PackagePaths);
// Output (assuming started with default value)
// .;$(AppConfig)/ActionScript 3.0/Classes
fl.as3PackagePaths="buying;selling";
fl.trace(fl.as3PackagePaths);
// Output
// buying; selling
```

另请参见

[fl.resetAS3PackagePaths\(\)](#)

## fl.browseForFileURL()

可用性

Flash MX 2004。

用法

```
fl.browseForFileURL(browseType [, title [, previewArea]])
```

参数

**browseType** 一个字符串，它指定文件浏览操作的类型。可接受的值为 "open"、"select" 或 "save"。值 "open" 或 "select" 将打开“打开文件”系统对话框。提供这些值是为了与 **Dreamweaver** 兼容。值 "save" 可打开“保存文件”系统对话框。

**title** 一个字符串，它指定“打开文件”或“保存文件”对话框的标题。如果省略了此参数，将使用默认值。此参数是可选的。

**previewArea** 一个被 **Flash** 和 **Fireworks** 忽略的可选参数，仅在要与 **Dreamweaver** 兼容的情况下使用。

返回

文件的 **URL**，表示为 **file:/// URI**；如果用户取消操作并退出对话框，则返回 **null**。

说明

方法；打开“打开文件”或“保存文件”系统对话框，允许用户指定要打开或保存的文件。

示例

下面的示例允许用户选择要打开的 **FLA** 文件，然后再打开该文件。（**fl.browseForFileURL()** 方法可以浏览任何类型的文件，但 **fl.openDocument()** 只能打开 **FLA** 文件。）

```
var fileURL = fl.browseForFileURL("open", "Select file");
var doc = fl.openDocument(fileURL);
```

另请参见

[fl.browseForFolderURL\(\)](#)

## fl.browseForFolderURL()

可用性

Flash 8。

**flash 对象 (fl)**

## 用法

```
fl.browseForFolderURL([description])
```

## 参数

**description** 一个可选字符串，它指定“浏览文件夹”对话框的描述。如果省略此参数，描述区域将不显示任何内容。

## 返回

文件夹的 URL，表示为 **file:/// URI**；如果用户取消操作并退出对话框，则返回 **null**。

## 说明

方法；显示“浏览文件夹”对话框，允许用户选择文件夹。

注：对话框的标题始终为“浏览文件夹”。使用 **description** 参数可以在标题下的说明区域中添加更多详细信息，如“选择文件夹”或“选择包含您要导入的配置文件的的路径”。

## 示例

下面的示例允许用户选择文件夹，然后显示该文件夹中的文件列表：

```
var folderURI = fl.browseForFolderURL("Select a folder.");  
var folderContents = FLfile.listFolder(folderURI);
```

## 另请参见

[fl.browseForFileURL\(\)](#)、[FLfile 对象](#)

## fl.clearPublishCache()

## 可用性

Flash CS5.5 Professional。

## 用法

```
fl.clearPublishCache()
```

## 参数

无。

## 返回

无。

## 描述

方法；清空发布缓存。

## 示例

下面的代码可清空发布缓存：

```
fl.clearPublishCache()
```

## 另请参见

[fl.publishCacheDiskSizeMax](#)、[fl.publishCacheEnabled](#)、第 227 页的“[fl.publishCacheMemoryEntrySizeLimit](#)”、第 228 页的“[fl.publishCacheMemorySizeMax](#)”

## fl.clipCopyString()

可用性

Flash CS3 Professional。

用法

```
fl.clipCopyString(string)
```

参数

**字符串** 要复制到剪贴板的字符串。

返回

无。

描述

方法；将指定字符串复制到剪贴板。

若要将当前所选内容复制到剪贴板，请使用 [document.clipCopy\(\)](#)。

示例

下面的示例将当前文档的路径复制到剪贴板：

```
var documentPath = fl.getDocumentDOM().path;
fl.clipCopyString(documentPath);
```

## fl.closeAll()

可用性

Flash MX 2004。

用法

```
fl.closeAll([bPromptToSave])
```

参数

**bPromptToSave** 一个可选的布尔值，用于指定是为自文件以前保存以来已更改的所有文件显示“保存”对话框，还是为从未保存的文件显示“另存为”对话框。默认值为 **true**。

返回

无。

描述

方法；关闭所有打开的文件（FLA 文件、SWF 文件、JSFL 文件等）。如果要关闭所有打开的文件，但不保存对任何文件的更改，则将 **false** 传递给 **bPromptToSave**。该方法不会终止应用程序。

示例

下面的代码将关闭所有打开的文件，并提示用户保存所有新建或更改的文件。

```
fl.closeAll();
```

另请参见

[fl.closeAllPlayerDocuments\(\)](#)、[fl.closeDocument\(\)](#)

## fl.closeAllPlayerDocuments()

可用性

Flash CS3 Professional。

用法

```
fl.closeAllPlayerDocuments()
```

参数

无。

返回

一个布尔值：如果打开了一个或多个影片窗口，则为 **true**；否则为 **false**。

说明

方法；关闭用“控制”>“测试影片”打开的所有 SWF 文件。

示例

下面的示例关闭用“控制”>“测试影片”打开的所有 SWF 文件。

```
fl.closeAllPlayerDocuments();
```

另请参见

[fl.closeAll\(\)](#)、[fl.closeDocument\(\)](#)

## fl.closeDocument()

可用性

Flash MX 2004。

用法

```
fl.closeDocument (documentObject [, bPromptToSaveChanges])
```

参数

**documentObject** [Document](#) 对象。如果 **documentObject** 引用活动文档，则直到调用此方法的脚本结束运行，“文档”窗口才会关闭。

**bPromptToSaveChanges** 一个布尔值。如果 **bPromptToSaveChanges** 为 **false**，则即使文档中包含未保存的更改也不会提示用户；也就是说，文件会关闭并放弃所做的更改。如果 **bPromptToSaveChanges** 为 **true**，并且文档中包含未保存的更改，则会出现一个标准的对话框，提示用户选择“是”或“否”。默认值为 **true**。此参数是可选的。

返回

无。

#### 描述

方法；关闭指定的文档。

#### 示例

下面的示例说明关闭文档的两种方式。

```
// Closes the specified document and prompts to save changes.  
fl.closeDocument (fl.documents[0]);  
fl.closeDocument (fl.documents[0] , true); // Use of true is optional.  
// Closes the specified document without prompting to save changes.  
fl.closeDocument (fl.documents[0], false);
```

另请参见

[fl.closeAll\(\)](#)

## fl.compilerErrors

#### 可用性

Flash CS3 Professional。

#### 用法

```
fl.compilerErrors
```

#### 描述

只读属性；一个 `compilerErrors` 对象，表示“错误”面板。有关使用此属性的信息，请参阅 [compilerErrors](#) 对象。

## fl.componentsPanel

#### 可用性

Flash MX 2004。

#### 用法

```
fl.componentsPanel
```

#### 描述

只读属性； [componentsPanel](#) 对象，表示“组件”面板。

#### 示例

下面的示例将 `componentsPanel` 对象存储在 `comPanel` 变量中：

```
var comPanel = fl.componentsPanel;
```

## fl.configDirectory

#### 可用性

Flash MX 2004。



**flash 对象 (fl)**

## 用法

```
fl.configDirectory
```

## 描述

只读属性；一个字符串，它以特定于平台的格式指定本地用户的 **Configuration** 目录的完整路径。若要将此路径指定为不特定于平台的 **file:///URI** 格式，请使用 [fl.configURI](#)。

## 示例

下面的示例在“输出”面板中显示 **Configuration** 目录：

```
fl.trace("My local configuration directory is " + fl.configDirectory);
```

## fl.configURI

## 可用性

Flash MX 2004。

## 用法

```
fl.configURI
```

## 描述

只读属性；一个字符串，它指定本地用户的 **Configuration** 目录的完整路径，表示为 **file:/// URI**。请参阅 [fl.configDirectory](#)。

## 示例

下面的示例运行指定的脚本。使用 **fl.configURI** 使您无需了解运行脚本的平台便可指定脚本的位置。

```
// To run a command in your commands menu, change "Test.jsfl"  
// to the command you want to run in the line below.  
fl.runScript( fl.configURI + "Commands/Test.jsfl" );
```

## fl.contactSensitiveSelection

## 可用性

Flash 8。

## 用法

```
fl.contactSensitiveSelection
```

## 描述

一个布尔值，它指定接触感应选择模式是否处于启用状态 (**true**) 或 (**false**)。

## 示例

下面的示例展示如何在选择之前禁用接触感应选择模式，以及如何在选择之后将其重置为原始值：

```
var contact = fl.contactSensitiveSelection;  
fl.contactSensitiveSelection = false;  
// Insert selection code here.  
fl.contactSensitiveSelection = contact;
```

## fl.createDocument()

可用性

Flash MX 2004。

用法

```
fl.createDocument([docType])
```

参数

**docType** 一个字符串，它指定要创建的文档的类型。可接受的值为 "timeline"、"presentation" 和 "application"。默认值为 "timeline"，与选择“文件”>“新建”>“Flash 文件”(ActionScript 3.0) 效果相同。此参数是可选的。

返回

如果方法成功执行，则返回新建文档的 **Document** 对象。如果发生错误，则返回值为 **undefined**。

说明

方法；打开并选择新文档。用于表示大小、分辨率和颜色的值与当前的默认值相同。

示例

下面的示例创建不同类型的文档：

```
// Create two Timeline-based Flash documents.
fl.createDocument();
fl.createDocument("timeline");
// Create a Slide Presentation document.
fl.createDocument("presentation");
// Create a Form Application document.
fl.createDocument("application");
```

## fl.createNewDocList

可用性

Flash MX 2004。

用法

```
fl.createNewDocList
```

描述

只读属性；一个字符串的数组，它表示可创建的各种文档类型。

示例

下面的示例在“输出”面板中显示可创建的文档类型：

```
fl.trace("Number of choices " + fl.createNewDocList.length);
for (i = 0; i < fl.createNewDocList.length; i++)
    fl.trace("choice: " + fl.createNewDocList[i]);
```

## fl.createNewDocListType

可用性

Flash MX 2004。

用法

```
fl.createNewDocListType
```

描述

只读属性；一个字符串的数组，它表示可创建的文档类型的文件扩展名。该数组中的条目直接对应于（按索引）[fl.createNewDocList](#) 数组中的条目。

示例

下面的示例在“输出”面板中显示可创建的文档类型的扩展名：

```
fl.trace("Number of types " + fl.createNewDocListType.length);  
for (i = 0; i < fl.createNewDocListType.length; i++) fl.trace("type: " + fl.createNewDocListType[i]);
```

## fl.createNewTemplateList

可用性

Flash MX 2004。

用法

```
fl.createNewTemplateList
```

描述

只读属性；一个字符串的数组，它表示可创建的各种模板类型。

示例

下面的示例在“输出”面板中显示可创建的模板类型：

```
fl.trace("Number of template types: " + fl.createNewTemplateList.length); for (i = 0; i <  
fl.createNewTemplateList.length; i++) fl.trace("type: " + fl.createNewTemplateList[i]);
```

## fl.documents

可用性

Flash MX 2004。

用法

```
fl.documents
```

描述

只读属性；一个由 **Document** 对象组成的数组（请参阅 [Document 对象](#)），它表示当前打开以进行编辑的文档（FLA 文件）。

**flash 对象 (fl)**

## 示例

下面的示例将打开的文档的数组存储在 docs 变量中：

```
var docs = fl.documents;
```

下面的示例在“输出”面板中显示当前打开文档的名称：

```
for (doc in fl.documents) {  
    fl.trace(fl.documents[doc].name);  
}
```

## fl.drawingLayer

## 可用性

Flash MX 2004。

## 用法

```
fl.drawingLayer
```

## 描述

只读属性； [drawingLayer 对象](#)，当用户在拖动过程中（例如，创建选取框时）要临时绘制时，可扩展工具应使用该对象。

## 示例

请参阅 [drawingLayer.setColor\(\)](#)。

## fl.exportPublishProfileString()

## 可用性

Flash Professional CS5。

## 用法

```
fl.exportPublishProfileString( ucfURI [, profileName] )
```

## 参数

**ucfURI** 一个字符串，用于指定要从中导出发布设置的文件统一资源标识符 (URI)。

**profileName** 一个字符串，用于指定要导出的配置文件名称。此参数是可选的。

## 返回

字符串。

## 描述

返回特定文档的发布配置文件，而无需打开该文件。还可以指定发布配置文件，但这是可选操作。

## 示例

以下示例读取发布配置文件字符串：

**flash** 对象 (fl)

```
var ppXML = "";
var ucfURI = fl.browseForFileURL("open", "select a FLA");
if (ucfURI && ucfURI.length > 0)
ppXML = fl.exportPublishProfileString(ucfURI);
fl.trace(ppXML);
```

## fl.externalLibraryPath

可用性

Flash CS4 Professional.

用法

```
fl.externalLibraryPath
```

描述

属性；一个字符串，其中包含全局 **ActionScript 3.0** 外部库路径中项目的列表，用于指定用作运行时共享库的 SWC 文件的位置。该字符串中的项目以分号分隔。在创作工具中，通过选择“编辑”>“首选参数”>“ActionScript”>“ActionScript 3.0 设置”来指定这些项目。

示例

下面的示例将 /SWC\_runtime 文件夹添加到全局 **ActionScript 3.0** 外部库路径中：

```
fl.trace(fl.externalLibraryPath);
fl.externalLibraryPath = "/SWC_runtime;" + fl.externalLibraryPath;
fl.trace(fl.externalLibraryPath);
```

另请参见

[fl.flexSDKPath](#)、[fl.libraryPath](#)、[fl.sourcePath](#)、[document.externalLibraryPath](#)

## fl.fileExists()

可用性

Flash MX 2004。

用法

```
fl.fileExists(fileURI)
```

参数

**fileURI** 一个字符串，表示为 **file:///** URI，它包含文件的路径。

返回

一个布尔值：如果文件存在于磁盘上，则为 **true**；否则为 **false**。

说明

方法；检查文件是否已经存在于磁盘上。

**flash 对象 (fl)**

## 示例

下面的示例在“输出”面板上为每个指定的文件显示 **true** 或 **false**，具体取决于文件是否存在。

```
alert(fl.fileExists("file:///C:/example fla"));
alert(fl.fileExists("file:///C:/example.jsfl"));
alert(fl.fileExists(""));
```

## fl.findDocumentDOM()

## 可用性

Flash CS3 Professional。

## 用法

```
fl.findDocumentDOM(id)
```

## 参数

**id** 一个整数，表示文档的唯一标识符。

## 返回

一个 **Document** 对象，或者如果不存在具有指定 **id** 的文档，则为 **null**。

## 说明

方法；可让您使用文件的唯一标识符（而不是文件的索引值或其他）找到特定文件。可以将此方法与 [document.id](#) 配合使用。

## 示例

下面的示例说明如何读取文档的 **ID**，然后使用它找到该文档：

```
var originalDocID = fl.getDocumentDOM().id;
// other code here, maybe working in different files
var targetDoc = fl.findDocumentDOM(originalDocID);
// Set the height of the Stage in the original document to 400 pixels.
targetDoc.height = 400;
```

## 另请参见

[fl.findDocumentIndex\(\)](#)

## fl.findDocumentIndex()

## 可用性

Flash MX 2004。

## 用法

```
fl.findDocumentIndex(name)
```

## 参数

**name** 要为其找出索引的文档的名称。该文档必须是打开的。

**flash 对象 (fl)**

## 返回

一个整数数组，这些整数表示 `fl.documents` 数组中文档 `name` 的位置。

## 说明

方法；返回一个整数数组，这些整数表示 `fl.documents` 数组中文档 `name` 的位置。可以打开多个同名的文档（如果这些文档位于不同的文件夹中）。

## 示例

下面的示例在“输出”面板中显示有关任何打开的、名为 `test.fla` 的文件的索引位置：

```
var filename = "test.fla"
var docIndex = fl.findDocumentIndex(filename);
for (var index in docIndex)
    fl.trace(filename + " is open at index " + docIndex[index]);
```

另请参见

[fl.documents](#)、[fl.findDocumentDOM\(\)](#)

## fl.findObjectInDocByName()

## 可用性

Flash CS3 Professional。

## 用法

```
fl.findObjectInDocByName(instanceName, document)
```

## 参数

**instanceName** 一个字符串，它指定项目在指定文档中的实例名称。

**文档 / 注释 (document)** [Document](#) 对象，在其中搜索指定的项目。

## 返回

通用对象数组。使用该数组中每一项的 `.obj` 属性可以获得相应对象。对象具有以下属性：`keyframe`、`layer`、`timeline` 和 `parent`。您可以使用这些属性来访问对象的层次结构。有关这些属性以及如何访问它们的详细信息，请参阅 [fl.findObjectInDocByType\(\)](#)。

您还可以访问 `layer` 和 `timeline` 值的方法和属性；它们分别等效于 [Layer](#) 对象和 [Timeline](#) 对象。

## 说明

方法；公开其实例名称与指定文本相匹配的文档元素。

注：在某些情况下，仅当从 `FLA` 文件中以命令形式运行时，此方法才有效，当前正在查看或编辑 `JSFL` 文件时无效。

## 示例

下面的示例在当前文档中搜索名称为 `instance01` 的元素。

```
var nameToSearchFor = "instance01";
var doc = fl.getDocumentDOM();
var results = fl.findObjectInDocByName(nameToSearchFor, doc);
if (results.length > 0) {
    alert("success, found " + results.length + " objects");
}
else {
    alert("failed, no objects named " + nameToSearchFor + " found");
}
```

另请参见

[fl.findObjectInDocByType\(\)](#)

## fl.findObjectInDocByType()

可用性

Flash CS3 Professional。

用法

```
fl.findObjectInDocByType(elementType, document)
```

参数

**elementType** 一个字符串，表示要搜索的元素的类型。有关可接受的值，请参阅 [element.elementType](#)。

**文档 / 注释 (document)** [Document](#) 对象，在其中搜索指定的项目。

返回

通用对象数组。使用该数组中每一项的 `.obj` 属性可以获得 [Element](#) 对象。每个对象都具有以下属性：`keyframe`、`layer`、`timeline` 和 `parent`。您可以使用这些属性来访问对象的层次结构。

您还可以访问 `layer` 和 `timeline` 值的方法和属性；它们分别等效于 [Layer](#) 对象和 [Timeline](#) 对象。

“示例”一节中的第二个和第三个示例说明如何访问这些属性。

说明

方法；显示文档中指定元素类型的元素。

注：在某些情况下，仅当从 [FLA](#) 文件中以命令形式运行时，此方法才有效，当前正在查看或编辑 [JSFL](#) 文件时无效。

示例

下面的示例在当前文档中搜索文本字段，然后更改其内容：

```
var doc = fl.getDocumentDOM();
var typeToSearchFor = "text";
var results = fl.findObjectInDocByType(typeToSearchFor, doc);
if (results.length > 0) {
    for (var i = 0; i < results.length; i++) {
        results[i].obj.setTextString("new text");
    }
    alert("success, found " + results.length + " objects");
}
else {
    alert("failed, no objects of type " + typeToSearchFor + " found");
}
```



下面的示例说明如何访问此方法返回的对象的特定属性:

```
var doc = fl.getDocumentDOM();
var resultsArray = findObjectInDocByType("text", doc);
if (resultsArray.length > 0)
{
    var firstItem = resultsArray[0];

    // firstItem.obj- This is the element object that was found.

    // You can access the following properties of this object:
    // firstItem.keyframe- The keyframe that the element is on.
    // firstItem.layer- The layer that the keyframe is on.
    // firstItem.timeline- The timeline that the layer is on.
    // firstItem.parent- The parent of the timeline. For example,
    //     the timeline might be in a symbol instance.
}
```

下面的示例说明如何备份 DOM, 以使用 `resultArray.obj` 对象查找已找到文本字段所在图层的名称:

```
var doc = fl.getDocumentDOM();
var typeToSearchFor = "text";
var resultsArray = fl.findObjectInDocByType(typeToSearchFor, doc);
if (resultsArray.length > 0) {
    for (var i = 0; i < resultsArray.length; i++) {
        resultsArray[i].obj.setTextString("new text");
        var firstItem = resultsArray[0];
        firstItemObj = firstItem.obj;
        fl.trace(firstItemObj.layer.name+"layerName");
    }
} else {
    alert("failed, no objects of type " + typeToSearchFor + " found");
}
```

另请参见

[fl.findObjectInDocByName\(\)](#)

## fl.flexSDKPath

可用性

Flash CS4 Professional.

用法

```
fl.flexSDKPath
```

描述

属性; 一个字符串, 它指定 Flex SDK 文件夹的路径, 其中包含 `bin`、`frameworks`、`lib` 和其他文件夹。在创作工具中, 通过选择“编辑”>“首选参数”>“ActionScript”>“ActionScript 3.0 设置”来指定这些项目。

示例

下面的代码在“输出”面板中显示 Flex SDK 路径:

```
fl.trace(fl.flexSDKPath);
```

**flash** 对象 (fl)

另请参见

[fl.externalLibraryPath](#)、[fl.libraryPath](#)、[fl.sourcePath](#)

## fl.getAppMemoryInfo()

可用性

Flash 8 (仅限 Windows)。

用法

`fl.getAppMemoryInfo(memType)`

参数

**memType** 一个整数，它指定要查询的内存利用区域。若要查看可接受值的列表，请参阅以下说明。

返回

一个整数，它表示在 **Flash.exe** 内存中指定区域内使用的字节数。

说明

方法 (仅适用于 Windows)；返回一个整数，表示在 **Flash.exe** 内存中指定区域内使用的字节数。使用下表确定要作为 **memType** 传递的值：

memType	资源数据
0	PAGEFAULTCOUNT
1	PEAKWORKINGSETSIZE
2	WORKINGSETSIZE
3	QUOTAPEAKPAGEDPOOLUSAGE
4	QUOTAPAGEDPOOLUSAGE
5	QUOTAPEAKNONPAGEDPOOLUSAGE
6	QUOTANONPAGEDPOOLUSAGE
7	PAGEFILEUSAGE
8	PEAKPAGEFILEUSAGE

示例

下面的示例显示当前的工作内存耗用情况：

```
var memsize = fl.getAppMemoryInfo(2);
fl.trace("Flash current memory consumption is " + memsize + " bytes or " + memsize/1024 + " KB");
```

## fl.getDocumentDOM()

可用性

Flash MX 2004。

**flash 对象 (fl)**

## 用法

```
fl.getDocumentDOM()
```

## 参数

无。

## 返回

一个文档对象，如果不存在打开的文档，则返回 **null**。

## 说明

方法；检索当前活动文档（FLA 文件）的 DOM（**Document 对象**）。如果有一个或多个文档是打开的，但有一个文档当前并不具有焦点（例如，如果 JSFL 文件具有焦点），则检索最近的活动文档的 DOM。

## 示例

下面的示例在“输出”面板中显示当前文档或最近的活动文档的名称：

```
var currentDoc = fl.getDocumentDOM();  
fl.trace(currentDoc.name);
```

## fl.getSwfPanel()

## 可用性

Flash CS5.5 Professional。

## 用法

```
fl.getSwfPanel(panelName, [useLocalizedPanelName])
```

## 参数

**panelName** 本地化面板名称或面板的 SWF 文件的根文件名。如果使用后者，则传入 **False** 作为第二个参数。

**useLocalizedPanelName** 可选。默认为 **True**。如果为 **False**，则假定 **panelName** 参数为面板的英语（未本地化）名称，该名称与 SWF 文件名（不包含文件扩展名）相对应。

## 返回

SWFPanel 对象。

## 描述

方法；基于面板的本地化名称或其 SWF 文件名（不包含文件扩展名）返回 SWFPanel 对象。

## 示例

下面的示例在“输出”面板中显示被引用为“Project”的面板的名称：

```
fl.trace('name of panel is: ' + fl.getSwfPanel('Project').name);
```

## fl.installedPlayers

可用性

Flash CS5.5 Professional。

用法

```
fl.installedPlayers()
```

参数

无。

返回

与文档属性检查器中已安装的 Flash Player 列表对应的通用对象数组。

描述

只读属性；与文档属性检查器中已安装的 Flash Player 列表对应的通用对象数组。

数组中的每个对象包含下列属性：

**name** 文档的字符串名称。

**version** 可用于设置文档的当前播放器（使用 Document.setPlayerVersion() 函数）。

**minASVersion** 文档所需的 ActionScript 最低版本。一个介于 minASVersion 和 maxASVersion（包含两者）之间的整数，可用于设置文档的 ActionScript 版本（使用 Document.asVersion 属性）。

**maxASVersion** 文档支持的 ActionScript 最高版本。

**stageWidth** 给定目标设备的默认舞台宽度（以像素为单位）。例如，对于 iPhone，默认大小为 320 x 480 像素。对于 Android，默认大小为 480 x 800。

**stageHeight** 给定目标设备的默认舞台高度（以像素为单位）。例如，对于 iPhone，默认大小为 320 x 480 像素。对于 Android，默认大小为 480 x 800。

示例

下面的示例跟踪 installedPlayers 数组中所有对象的属性并显示到“输出”窗口：

```
var arr = fl.installedPlayers;  
for (var i in arr) fl.trace("name: " + arr[i].name + " version: " + arr[i].version + " minASVersion: " +  
arr[i].minASVersion + " maxASVersion: " + arr[i].maxASVersion + " stageWidth: " + arr[i].stageWidth + "  
stageHeight: " + arr[i].stageHeight + " ");
```

## fl.isFontInstalled()

可用性

Flash CS4 Professional。

用法

```
fl.isFontInstalled(fontName)
```

参数

**fontName** 一个字符串，它指定设备字体的名称。

**flash 对象 (fl)**

## 返回

布尔值：如果已安装指定的字体，则为 `true`；否则为 `false`。

## 说明

方法；确定是否已安装指定的字体。

## 示例

下面的代码在已安装 **Times** 字体的情况下在“输出”面板中显示“true”。

```
fl.trace(fl.isFontInstalled("Times"));
```

## fl.languageCode

## 可用性

Flash CS5 Professional。

## 用法

```
fl.languageCode
```

## 描述

属性；一个字符串，用于返回标识应用程序用户界面区域设置的 5 个字符的代码。

## 示例

以下示例返回由 **Flash** 应用程序的本地化的用户界面指示的 5 个字符语言代码：

```
locConfigURI = fl.applicationURI + fl.languageCode + "/Configuration";
```

## fl.libraryPath

## 可用性

Flash CS4 Professional。

## 用法

```
fl.libraryPath
```

## 描述

属性；一个字符串，其中包含全局 **ActionScript 3.0** 库路径中项目的列表，用于指定 **SWC** 文件或包含 **SWC** 文件的文件夹的位置。该字符串中的项目以分号分隔。在创作工具中，通过选择“编辑”>“首选参数”>“ActionScript”>“ActionScript 3.0 设置”来指定这些项目。

## 示例

下面的示例将 `/SWC` 文件夹添加到全局 **ActionScript 3.0** 库路径中：

```
fl.trace(fl.libraryPath);  
fl.libraryPath = "/SWC;" + fl.libraryPath;  
fl.trace(fl.libraryPath);
```

flash 对象 (fl)

另请参见

[fl.externalLibraryPath](#)、[fl.flexSDKPath](#)、[fl.sourcePath](#)、[document.libraryPath](#)

## fl.mapPlayerURL()

可用性

Flash MX 2004。

用法

```
fl.mapPlayerURL(URI [, returnMBCS])
```

参数

**URI** 一个字符串，其中包含要映射的转义 Unicode URL。

**returnMBCS** 一个布尔值，如果要返回转义的 MBCS 路径，则必须将该值设置为 true。否则，该方法将返回 UTF-8 路径。默认值为 false。此参数是可选的。

返回

一个字符串，它是转换的 URL。

说明

方法；将转义的 Unicode URL 映射到 UTF-8 或 MBCS URL。当要在 ActionScript 中使用字符串来访问外部资源时，可使用此方法。如果需要处理多字节字符，则必须使用此方法。

示例

下面的示例将 URL 转换为 UTF-8，以使播放器可以进行加载：

```
var url = MMExecute( "fl.mapPlayerURL(" + myURL + ", false);" );  
mc.loadMovie( url);
```

## fl.Math

可用性

Flash MX 2004。

用法

```
fl.Math
```

描述

只读属性；[Math](#) 对象提供用于矩阵和点运算的方法。

示例

下面的示例显示所选对象的变形矩阵及其逆矩阵：

**flash 对象 (fl)**

```
// Select an element on the Stage and then run this script.
var mat =fl.getDocumentDOM().selection[0].matrix;
for(var prop in mat){
fl.trace("mat."+prop+" = " + mat[prop]);
}
var invMat = fl.Math.invertMatrix( mat );
for(var prop in invMat) {
fl.trace("invMat."+prop+" = " + invMat[prop]);
}
```

## fl.mruRecentFileList

## 可用性

Flash MX 2004。

## 用法

fl.mruRecentFileList

## 描述

只读属性；Flash 创作工具管理的“最近使用的文档”(MRU) 列表内的完整文件名的数组。

## 示例

下面的示例在“输出”面板中显示最近打开的文件数以及每个文件的名称：

```
fl.trace("Number of recently opened files: " + fl.mruRecentFileList.length);
for (i = 0; i < fl.mruRecentFileList.length; i++) fl.trace("file: " + fl.mruRecentFileList[i]);
```

## fl.mruRecentFileListType

## 可用性

Flash MX 2004。

## 用法

fl.mruRecentFileListType

## 描述

只读属性；Flash 创作工具管理的 MRU 列表内的文件类型的数组。此数组对应于 [fl.mruRecentFileList](#) 属性中的数组。

## 示例

下面的示例在“输出”面板中显示最近打开的文件数以及每个文件的类型：

```
fl.trace("Number of recently opened files: " + fl.mruRecentFileListType.length);
for (i = 0; i < fl.mruRecentFileListType.length; i++) fl.trace("type: " + fl.mruRecentFileListType[i]);
```

## fl.objectDrawingMode

可用性

Flash 8。

用法

```
fl.objectDrawingMode
```

描述

属性；一个布尔值，它指定是启用对象绘制模式 (**true**) 还是启用合并绘制模式 (**false**)。

示例

下面的示例切换对象绘制模式的状态：

```
var toggleMode = fl.objectDrawingMode;
if (toggleMode) {
    fl.objectDrawingMode = false;
} else {
    fl.objectDrawingMode = true;
}
```

## fl.openDocument()

可用性

Flash MX 2004。

用法

```
fl.openDocument (fileURI)
```

参数

**fileURI** 一个字符串，表示为 **file:///** URI，它指定要打开的文件的名称。

返回

如果方法成功执行，则返回最近打开的文档的 **Document** 对象。如果没有找到文件，或该文件不是有效的 **FLA** 文件，将报错并取消脚本。

描述

方法；在新的 **Flash** 文档窗口中打开一个 **Flash** 文档 (**FLA** 文件) 供编辑，并使其获得焦点。对于用户而言，其效果等同于选择“文件”>“打开”并选择一个文件。如果指定的文件已经打开，则包含该文档的窗口将显示在最前面。包含指定文件的窗口将成为当前选择的文档。

示例

下面的示例打开存储在 C 驱动器根目录下名为 **Document fla** 的文件。代码将表示该文档的 **Document** 对象存储在 **doc** 变量中，并将该文档设置为当前所选文档。也就是说，在焦点更改之前，**fl.getDocumentDOM()** 将一直引用此文档。

```
var doc = fl.openDocument("file:///c:/Document.fla");
```



## fl.openScript()

### 可用性

Flash MX 2004。Flash Professional CS5 中添加的可选参数。

### 用法

```
fl.openScript(fileURI [, createExtension, className])
```

### 参数

**fileURI** 一个字符串，表示为 `file:/// URI`，它指定应加载到 Flash 文本编辑器中的 JSFL、AS、ASC、XML、TXT 或其他文件的路径。此参数可能为 `null`。如果为 `null`，此方法将打开由 `createExtension` 参数指定的类型的新脚本。

**createExtension** 一个字符串，用于确定当 `fileURI` 为 `null` 时要创建的文档类型。默认为“AS”；允许的值包括“JSFL”、“AS”、“ASC”、“XML”、“TXT”、“AS3\_CLASS”或“AS3\_INTERFACE”。此参数是在 Flash Professional CS5 中添加的。

**className** 一个字符串，用于在创建类或界面时指定完全限定的类名称（由 `createExtension` 参数确定）。此参数是在 Flash Professional CS5 中添加的。

### 返回

无。

### 描述

方法；在 Flash 文本编辑器中打开一个现有文件或创建一个新脚本（JSFL、AS、ASC）或其他文件（XML、TXT）。

### 示例

下面的示例打开存储在 C 驱动器 `/temp` 目录下名为 `my_test.jsfl` 的文件：

```
fl.openScript("file:///c:/temp/my_test.jsfl");
```

### 示例

以下示例创建一个具有空白 AS3 类定义的新 `.as` 文件：

```
fl.openScript(null, 'AS3_CLASS');
```

## fl.outputPanel

### 可用性

Flash MX 2004。

### 用法

```
fl.outputPanel
```

### 描述

只读属性；对 [outputPanel](#) 对象的引用。

### 示例

请参阅 [outputPanel](#) 对象。

## fl.packagePaths

可用性

Flash CS3 Professional。

用法

```
fl.packagePaths
```

描述

属性；一个字符串，与“ActionScript 2.0 设置”对话框中的全局类路径设置相对应。该字符串内的类路径以分号 (;) 分隔。若要查看或更改 ActionScript 3.0 类路径设置，请使用 [fl.as3PackagePaths](#)。

示例

下面的示例展示如何更改 ActionScript2.0 的“类路径”设置：

```
fl.trace(fl.packagePaths);  
// Output (assuming started with default value)  
// .;$ (LocalData)/Classes  
fl.packagePaths="buying;selling";  
fl.trace(fl.packagePaths);  
// Output  
// buying; selling
```

另请参见

[fl.resetPackagePaths\(\)](#)

## fl.presetPanel

可用性

Flash CS4 Professional。

用法

```
fl.presetPanel
```

描述

只读属性：一个 [presetPanel](#) 对象。

## fl.publishCacheDiskSizeMax

可用性

Flash CS5.5 Professional。

用法

```
fl.publishCacheDiskSizeMax
```

描述

属性：一个整数，用于设置磁盘上发布缓存的最大大小（以 MB 为单位）。

**flash 对象 (fl)**

## 示例

下面的代码将磁盘上发布缓存的最大大小设置为 1 MB:

```
fl.publishCacheDiskSizeMax = 1
```

另请参见

[fl.clearPublishCache\(\)](#)、[fl.publishCacheEnabled](#)、第 227 页的“[fl.publishCacheMemoryEntrySizeLimit](#)”、第 228 页的“[fl.publishCacheMemorySizeMax](#)”

## fl.publishCacheEnabled

## 可用性

Flash CS5.5 Professional。

## 用法

```
fl.publishCacheEnabled
```

## 描述

属性：一个布尔值，用于设置是否启用发布缓存。

## 示例

下面的代码显示是否在“输出”窗口中已启用发布缓存。

```
fl.trace(fl.publishCacheEnabled);
```

另请参见

[fl.publishCacheDiskSizeMax](#)、[fl.clearPublishCache\(\)](#)、第 227 页的“[fl.publishCacheMemoryEntrySizeLimit](#)”、第 228 页的“[fl.publishCacheMemorySizeMax](#)”

## fl.publishCacheMemoryEntrySizeLimit

## 可用性

Flash CS5.5 Professional。

## 用法

```
fl.publishCacheMemoryEntrySizeLimit
```

## 描述

属性：一个整数，用于设置可添加到内存发布缓存的项目的最大大小（以 KB 为单位）。任何不大于此大小的项目都将保留在内存中；任何大于此大小的项目都将写入磁盘。

拥有大量内存的用户可能希望增加此值以提高性能，而拥有很少内存的用户可能希望减小此值，以防止发布缓存消耗太多内存。

## 示例

下面的代码将可以存储在内存中的最大发布缓存项目大小设置为 100 KB:

```
fl.publishCacheMemoryEntrySizeLimit = 100
```

另请参见

[fl.publishCacheDiskSizeMax](#)、[fl.publishCacheEnabled](#)、第 205 页的“[fl.clearPublishCache\(\)](#)”、第 228 页的“[fl.publishCacheMemorySizeMax](#)”

## fl.publishCacheMemorySizeMax

可用性

Flash CS5.5 Professional。

用法

```
fl.publishCacheMemorySizeMax
```

描述

属性：一个整数，用于设置内存中发布缓存的最大大小（以 MB 为单位）。

示例

下面的代码将内存中发布缓存的最大大小设置为 1 MB：

```
fl.publishCacheMemorySizeMax = 1
```

另请参见

[fl.publishCacheDiskSizeMax](#)、[fl.publishCacheEnabled](#)、第 227 页的“[fl.publishCacheMemoryEntrySizeLimit](#)”、第 205 页的“[fl.clearPublishCache\(\)](#)”

## fl.publishDocument()

可用性

Flash CS5 Professional。

用法

```
fl.publishDocument( flaURI [, publishProfile] )
```

参数

**flaURI** 一个字符串，表示为 `file:/// URI`，它指定应以静默方式发布的 FLA 文件的路径。

**publishProfile** 一个字符串，它指定要在发布时使用的发布配置文件。如果省略此参数，将使用默认的发布配置文件。

返回

布尔值

描述

方法；发布 FLA 文件，而无需打开该文件。此 API 在无头模式下打开 FLA 并发布 SWF（或根据对配置文件的设置）。第二个参数 (`publishProfile`) 是可选的。返回值是一个布尔值，指示是否已找到配置文件。如果未提供第二个参数，则返回值始终为 `true`。

**flash 对象 (fl)**

## 示例

以下示例提示用户选择 **FLA** 文件并使用“默认”发布配置文件以静默方式发布该文件：

```
var uri = fl.browseForFileURL("select", "select a FLA file to publish");
var publishProfileName = "Default";
fl.publishDocument(uri, publishProfileName);
```

## fl.quit()

## 可用性

Flash MX 2004。

## 用法

```
fl.quit([bPromptIfNeeded])
```

## 参数

**bPromptIfNeeded** 一个布尔值，如果要提示用户保存任何修改过的文档，则该值应为 **true**（默认值）。如果不想提示用户保存修改过的文档，则应将此参数设置为 **false**。在后一种情形下，将放弃在打开的文档中所做的任何修改并将立即退出应用程序。尽管它对批处理很有用，但使用此方法时要格外小心。此参数是可选的。

## 返回

无。

## 描述

方法；退出 **Flash**，并提示用户保存任何已更改的文档。

## 示例

下面的示例说明了退出时询问和不询问是否保存已修改的文档的情况：

```
// Quit with prompt to save any modified documents.
fl.quit();
fl.quit(true); // True is optional.
// Quit without saving any files.
fl.quit(false);
```

## fl.reloadEffects()

## 可用性

Flash MX 2004。

## 用法

```
fl.reloadEffects()
```

## 参数

无。

## 返回

无。

**flash 对象 (fl)**

## 描述

方法；重新加载在用户的 **Configuration Effects** 文件夹中定义的所有特效描述符。此方法允许您在开发过程中快速更改脚本，并且提供一种无需重新启动应用程序便可改进特效的机制。此方法在用于 **Commands** 文件夹下的命令中时效果最好。

## 示例

下面的示例是可放置在 **Commands** 文件夹下的单行脚本。需要重新加载特效时，请转到“命令”菜单并执行脚本。

```
fl.reloadEffects();
```

## fl.reloadTools()

## 可用性

Flash MX 2004。

## 用法

```
fl.reloadTools()
```

## 参数

无。

## 返回

无。

## 描述

方法；从 **toolconfig.xml** 文件中重新构建“工具”面板。此方法仅在创建可扩展工具时使用。当您需要重新加载“工具”面板时（例如，在修改了一个用于定义面板中已有的工具的 **JSFL** 文件后），应使用此方法。

## 示例

下面的示例是可放置在 **Commands** 文件夹下的单行脚本。需要重新加载“工具”面板时，从“命令”菜单中运行该脚本。

```
fl.reloadTools();
```

## fl.removeEventListener()

## 可用性

Flash CS3 Professional。

## 用法

```
fl.removeEventListener(eventType)
```

## 参数

**eventType** 一个字符串，指定要从回调函数删除的事件类型。可接受的值为 **documentNew**、**documentOpened**、**documentClosed**、**mouseMove**、**documentChanged**、**layerChanged** 和 **frameChanged**。

## 返回

一个布尔值。如果成功删除了事件侦听器，则为 **true**；如果从未使用 **fl.addEventListener()** 方法将函数添加到列表，则为 **false**。

#### 说明

注销使用 [fl.addEventListener\(\)](#) 注册的函数。

#### 示例

下面的示例删除与 `documentClosed` 事件关联的事件侦听器：

```
fl.removeEventListener("documentClosed");
```

另请参见

[fl.addEventListener\(\)](#)

## fl.resetAS3PackagePaths()

#### 可用性

Flash CS3 Professional。

#### 用法

```
fl.resetAS3PackagePaths();
```

#### 参数

无。

#### 说明

方法；在“ActionScript 3.0 设置”对话框中将全局类路径设置重置为默认值。若要重置 ActionScript 2.0 全局类路径，请使用 [fl.resetPackagePaths\(\)](#)。

#### 示例

下面的示例将 ActionScript 3.0 的“类路径”设置重置为默认值。

```
fl.resetAS3PackagePaths();
```

另请参见

[fl.as3PackagePaths](#)

## fl.resetPackagePaths()

#### 可用性

Flash CS3 Professional。

#### 用法

```
fl.resetPackagePaths();
```

#### 参数

无。

**flash 对象 (fl)**

## 说明

方法；在“ActionScript 2.0 设置”对话框中将全局类路径设置重置为默认值。若要重置 ActionScript 3.0 全局类路径，请使用 [fl.resetAS3PackagePaths\(\)](#)。

## 示例

下面的示例将 ActionScript 2.0 的“类路径”设置重置为默认值。

```
fl.resetPackagePaths();
```

另请参见

[fl.packagePaths](#)

## fl.revertDocument()

## 可用性

Flash MX 2004。

## 用法

```
fl.revertDocument (documentObject)
```

## 参数

**documentObject** [Document](#) 对象。如果 documentObject 引用活动文档，则直到调用此方法的脚本结束运行，“文档”窗口才可能还原。

## 返回

一个布尔值：如果“还原”操作成功完成，则为 **true**；否则为 **false**。

## 说明

方法；将指定的 FLA 文档还原为上次保存的版本。与“文件”>“还原”菜单选项不同，此方法并不显示要求用户确认操作的警告窗口。请参阅 [document.revert\(\)](#) 和 [document.canRevert\(\)](#)。

## 示例

下面的示例将当前的 FLA 文档还原为上次保存的版本；上次保存之后所做的任何更改都将丢失。

```
fl.revertDocument (fl.getDocumentDOM());
```

## fl.runScript()

## 可用性

Flash MX 2004。

## 用法

```
fl.runScript (fileURI [, funcName [, arg1, arg2, ...]])
```

## 参数

**fileURI** 一个字符串，表示为 **file:/// URI**，它指定要执行的脚本文件的名称。



**flash 对象 (fl)**

**funcName** 一个字符串，它标识要在 **fileURI** 所指定的 JSFL 文件中执行的函数。此参数是可选的。

**arg** 一个可选的参数，它指定要传递给 **funcname** 的一个或多个参数。

**返回**

如果指定了 **funcName**，则函数的结果为一个字符串；否则，不返回值。

**说明**

方法；执行 **JavaScript** 文件。如果函数被指定为其中的一个参数，它将运行该函数以及不在该函数内的脚本中的任何代码。脚本中的其他代码将在运行函数之前运行。

**示例**

假设驱动器 C 的根目录中存在名为 **testScript.jsfl** 的脚本文件，其内容如下：

```
function testFunc(num, minNum) {
    fl.trace("in testFunc: 1st arg: " + num + " 2nd arg: " + minNum);
}
for (i=0; i<2; i++) {
    fl.trace("in for loop i=" + i);
}
fl.trace("end of for loop");
// End of testScript.jsfl
```

如果您发出以下命令，

```
fl.runScript("file:///C:/testScript.jsfl", "testFunc", 10, 1);
```

将在“输出”面板中显示以下信息：

```
in for loop i=0
in for loop i=1
end of for loop
in testFunc: 1st arg: 10 2nd arg: 1
```

还可以只调用 **testScript.jsfl**，而不执行函数，如下所示：

```
fl.runScript("file:///C:/testScript.jsfl");
```

该命令在“输出”面板中生成以下内容：

```
in for loop i=0
in for loop i=1
end of for loop
```

## fl.saveAll()

**可用性**

Flash MX 2004。

**用法**

```
fl.saveAll()
```

**参数**

无。

**flash 对象 (fl)**

返回

无。

描述

方法；保存所有打开的文档。

如果文件从未保存过，或自上次保存以来从未修改过，则不保存该文件。若要允许保存未保存过的文件或未修改过的文件，请使用 [fl.saveDocumentAs\(\)](#)。

示例

下面的示例保存此前已保存过，且自上次保存以来有过修改的所有打开文档：

```
fl.saveAll();
```

另请参见

[document.save\(\)](#)、[document.saveAndCompact\(\)](#)、[fl.saveDocument\(\)](#)、[fl.saveDocumentAs\(\)](#)

## fl.saveDocument()

可用性

Flash MX 2004。

用法

```
fl.saveDocument(document [, fileURI])
```

参数

**文档 / 注释 (document)** [Document](#) 对象，它指定要保存的文档。如果 **document** 为 **null**，将保存活动文档。

**fileURI** 一个字符串，表示为 **file:/// URI**，它指定保存的文档的名称。如果参数 **fileURI** 为 **null** 或被省略，将以文档当前的名称来保存文档。此参数是可选的。

返回

布尔值：如果保存操作成功完成，则返回 **true**；否则返回 **false**。

如果文件从未保存过，或自上次保存以来从未修改过，则不保存该文件并返回 **false**。若要允许保存未保存过的文件或未修改过的文件，请使用 [fl.saveDocumentAs\(\)](#)。

说明

方法；将指定文档保存为 **FLA** 文档。

示例

下面的示例保存当前的文档和两个指定文档：

```
// Save the current document.
alert(fl.saveDocument(fl.getDocumentDOM()));
// Save the specified documents.
alert(fl.saveDocument(fl.documents[0], "file:///C|/example1 fla"));
alert(fl.saveDocument(fl.documents[1], "file:///C|/example2 fla"));
```

另请参见

[document.save\(\)](#)、[document.saveAndCompact\(\)](#)、[fl.saveAll\(\)](#)、[fl.saveDocumentAs\(\)](#)

## fl.saveDocumentAs()

可用性

Flash MX 2004。

用法

```
fl.saveDocumentAs (document)
```

参数

**文档 / 注释 (document)** [Document](#) 对象，它指定要保存的文档。如果 **document** 为 **null**，将保存活动文档。

返回

一个布尔值：如果“另存为”操作成功完成，则为 **true**；否则为 **false**。

说明

方法；为指定的文档显示“另存为”对话框。

示例

下面的示例提示用户保存指定的文档，然后显示一条警告消息，指明文档是否已保存：

```
alert (fl.saveDocumentAs (fl.documents [1]));
```

另请参见

[document.save\(\)](#)、[document.saveAndCompact\(\)](#)、[fl.saveAll\(\)](#)、[fl.saveDocument\(\)](#)

## fl.scriptURI

可用性

Flash CS3 Professional。

用法

```
fl.scriptURI
```

描述

只读属性；一个字符串，表示当前运行的 JSFL 脚本的路径，表示为 **file:/// URI**。如果脚本是从 [fl.runScript\(\)](#) 调用的，则此属性表示直接父脚本的路径。也就是说，它不会遍历对 [fl.runScript\(\)](#) 的多个调用来查找原始调用脚本的路径。

示例

下面的示例在“输出”面板中显示当前运行的 JSFL 脚本的路径：

```
fl.trace (fl.scriptURI);
```

另请参见

[fl.runScript\(\)](#)

## fl.selectElement()

可用性

Flash CS3 Professional。

用法

```
fl.selectElement(elementObject, editMode)
```

参数

**elementObject** 要选择的 [Element](#) 对象。

**editMode** 一个布尔值，它指定是要编辑该元素 (**true**) 还是只选择该元素 (**false**)。

返回

布尔值：如果成功选择该元素，则为 **true**；否则为 **false**。

说明

方法；启用元素的选择或编辑。通常，您将对 [fl.findObjectInDocByName\(\)](#) 或 [fl.findObjectInDocByType\(\)](#) 返回的对象使用此方法。

示例

下面的示例选择名为 "second text field" 的元素（如果在文档中找到一个这样的元素）：

```
var nameToSearchFor = "second text field";
var doc = fl.getDocumentDOM();

// Start by viewing Scene 1 (index value of 0).
document.editScene(0);

// Search for element by name.
var results = fl.findObjectInDocByName(nameToSearchFor, doc);
if (results.length > 0) {
    // Select the first element found.
    // Pass false, so the symbolInstance you are searching for is selected.
    // If you pass true, the symbol instance will switch to edit mode.
    fl.selectElement(results[0], false);
    alert("success, found " + results.length + " objects")
}
else {
    alert("failed, no objects with name " + nameToSearchFor + " found");
}
```

另请参见

[fl.findObjectInDocByName\(\)](#)、[fl.findObjectInDocByType\(\)](#)

## fl.selectTool()

可用性

Flash CS3 Professional。

**flash 对象 (fl)**

## 用法

```
fl.selectTool(toolName)
```

## 参数

**toolName** 一个字符串，指定要选择的工具的名称。有关此参数可接受的值的信息，请参阅下文的“描述”。

## 说明

方法；在“工具”面板中选择指定的工具。**toolName** 可接受的值为 arrow、bezierSelect、freeXform、fillXform、lasso、pen、penplus、penminus、penmodify、text、line、rect、oval、rectPrimitive、ovalPrimitive、polystar、pencil、brush、inkBottle、bucket、eyeDropper、eraser、hand 和 magnifier。

如果您或用户创建自定义工具，则也可以将这些工具的名称作为 **toolName** 参数进行传递。工具名称列表位于以下文件中：

## • Windows Vista:

引导驱动器 \Users\ 用户名 \Local Settings\Application Data\Adobe\Flash CS3\ 语言  
\Configuration\Tools\toolConfig.xml

## • Windows XP:

引导驱动器 \Documents and Settings\ 用户名 \Local Settings\Application Data\Adobe\Flash CS3\ 语言  
\Configuration\Tools\toolConfig.xml

## • Mac OS X:

Macintosh HD/Users/ 用户名 /Library/Application Support/Adobe/Flash CS3/ 语言  
/Configuration/Tools/toolConfig.xml

## 示例

下面的示例选择钢笔工具。

```
fl.selectTool("pen");
```

另请参见

[Tools 对象](#)、[ToolObj 对象](#)

## fl.setActiveWindow()

## 可用性

Flash MX 2004。

## 用法

```
fl.setActiveWindow(document [, bActivateFrame])
```

## 参数

**文档 / 注释 (document)** [Document 对象](#)，它指定要选作活动窗口的文档。

**bActivateFrame** 一个被 Flash 和 Fireworks 忽略的可选参数，仅在与 Dreamweaver 兼容的情况下使用。

## 返回

无。

#### 描述

方法；将活动窗口设置为指定文档。Dreamweaver 和 Fireworks 也支持此方法。如果文档有多个视图（通过“窗口”>“直接复制窗口”创建），则选择最近的活动视图。

#### 示例

下面的示例显示了两种激活指定文档的方式：

```
fl.setActiveWindow(fl.documents[0]);

var theIndex = fl.findDocumentIndex("myFile fla");
fl.setActiveWindow(fl.documents[theIndex]);
```

## fl.showIdleMessage()

#### 可用性

Flash 8。

#### 用法

```
fl.showIdleMessage(show)
```

#### 参数

**show** 一个布尔值，它指定是启用还是禁用关于脚本运行时间太长的警告。

#### 返回

无。

#### 描述

方法；允许您禁用关于脚本运行时间太长的警告（为 **show** 传递 **false**）。处理需要很长时间才能完成的批处理操作时，可能要使用此方法。若要重新启用该警告，请再次发出该命令，此次将为 **show** 传递 **true**。

#### 示例

下面的示例说明如何禁用和重新启用关于脚本运行时间太长的警告：

```
fl.showIdleMessage(false);
var result = timeConsumingFunction();
fl.showIdleMessage(true); ;
var result = timeConsumingFunction();
```

## fl.sourcePath

#### 可用性

Flash CS4 Professional.

#### 用法

```
fl.sourcePath
```

**flash 对象 (fl)**

## 描述

属性；一个字符串，其中包含全局 **ActionScript 3.0** 源路径中项目的列表，用于指定 **ActionScript** 类文件的位置。该字符串中的项目以分号分隔。在创作工具中，通过选择“编辑”>“首选参数”>“ActionScript”>“ActionScript 3.0 设置”来指定这些项目。

## 示例

下面的示例将 `/Classes` 文件夹添加到全局 **ActionScript 3.0** 源路径中：

```
fl.trace(fl.sourcePath);
fl.sourcePath = "/Classes;" + fl.sourcePath;
fl.trace(fl.sourcePath);
```

## 另请参见

[fl.flexSDKPath](#)、[fl.externalLibraryPath](#)、[fl.libraryPath](#)、[document.sourcePath](#)

## fl.swfPanels

## 可用性

Flash CS4 Professional.

## 用法

```
fl.swfPanels
```

## 描述

只读属性；一个由已注册的 **swfPanel** 对象组成的数组（请参阅 [swfPanel 对象](#)）。只要 **swfPanel** 对象至少已打开一次，就会注册该对象。

面板在数组中的位置表示打开该面板的顺序。如果第一个打开的面板名为 **TraceBitmap**，而第二个打开的面板名为 **AnotherFunction**，则 `fl.swfPanels[0]` 为 **TraceBitmap swfPanel** 对象，`fl.swfPanels[1]` 为 **AnotherFunction swfPanel** 对象，依此类推。

## 示例

下面的代码在“输出”面板中显示任何已注册的“窗口 SWF”面板的名称和路径：

```
if (fl.swfPanels.length > 0) {
    for (x = 0; x < fl.swfPanels.length; x++) {
        fl.trace("Panel: " + fl.swfPanels[x].name + " -- Path: " + fl.swfPanels[x].path);
    }
}
```

## fl.toggleBreakpoint()

## 可用性

Flash Professional CS5。

## 用法

```
fl.toggleBreakPoint(String fileURI, int line, Boolean enable)
```

**flash 对象 (fl)**

## 参数

**fileURI** 一个字符串；要在其中切换断点的 AS 文件的 URI。

**line** 一个整数；切换断点位置的行号。

**enable** 布尔值；如果设置为 **true**，则启用断点。如果设置为 **false**，则禁用断点。

## 描述

在给定行切换给定 .as 文件的断点。如果 **enable** 为 **false**，将擦除当前存储在该行的断点。

## 示例

以下示例启用位于 C:\AS\breakpointTest.as 中的 AS 文件的第 10 行处的断点：

```
fl.toggleBreakPoint("file:///C:/AS/breakpointTest.as", 10, 1);
```

## fl.tools

## 可用性

Flash MX 2004。

## 用法

```
fl.tools
```

## 描述

只读属性；一个由 **Tools** 对象组成的数组（请参阅 [Tools 对象](#)）。此属性仅在创建可扩展工具时使用。

## fl.trace()

## 可用性

Flash MX 2004。

## 用法

```
fl.trace(message)
```

## 参数

**message** 一个出现在“输出”面板中的字符串。

## 返回

无。

## 描述

方法；将一个以新行终止的文本字符串发送到“输出”面板；如果“输出”面板尚不可见，则显示它。此方法与 [outputPanel.trace\(\)](#) 等效，使用方式与 **ActionScript** 中的 `trace()` 语句相同。

要发送一个空行，请使用 `fl.trace("")` 或 `fl.trace("\n")`。可使用后一种内联命令，将 `\n` 作为 **message** 字符串的一部分。

## 示例

下面的示例在“输出”面板中显示几行文本：



**flash 对象 (fl)**

```
fl.outputPanel.clear();
fl.trace("Hello World!!!");
var myPet = "cat";
fl.trace("\nI have a " + myPet);
fl.trace("");
fl.trace("I love my " + myPet);
fl.trace("Do you have a " + myPet + "?");
```

## fl.version

## 可用性

Flash MX 2004。

## 用法

```
fl.version
```

## 描述

只读属性；Flash 创作工具的长字符串版本（包括平台）。

## 示例

下面的示例在“输出”面板中显示 Flash 创作工具的版本：

```
alert(fl.version); // For example, WIN 10,0,0,540
```

## fl.xmlui

## 可用性

Flash MX 2004。

## 用法

```
fl.xmlui
```

## 描述

只读属性；[XMLUI 对象](#)。此属性允许您在“XMLUI”对话框中获取并设置 XMLUI 属性，并允许您以编程方式接受或取消对话框。

## 示例

请参阅 [XMLUI 对象](#)。

## 第 18 章 : FLfile 对象

可用性

Flash MX 2004 7.2。

描述

FLfile 对象允许您编写可对本地文件系统中的文件和文件夹进行访问、修改和删除操作的 Flash 扩展。FLfile API 以 JavaScript API 扩展的形式提供。此扩展称为共享库，它位于以下文件夹内：

- Windows Vista:  
引导驱动器 \Users\ 用户名 \Local Settings\Application Data\Adobe\Flash CS3\ 语言 \Configuration\External Libraries\FLfile.dll
- Windows XP:  
引导驱动器 \Documents and Settings\ 用户名 \Local Settings\Application Data\Adobe\Flash CS3\ 语言 \Configuration\External Libraries\FLfile.dll
- Mac OS X:  
Macintosh HD/Users/ 用户名 /Library/Application Support/Adobe/Flash CS3/ 语言 /Configuration/External Libraries/FLfile.dll

注：不要将包含 Flash 文档元素的共享库与 JavaScript API 共享库混淆。它们是两种不同的共享库。

FLfile 方法可处理磁盘上的文件或文件夹（目录）。因此，每种方法都需要一个或多个参数来指定文件或文件夹的位置。文件或文件夹的位置以字符串形式表示，与网站 URL 非常相似。这称为文件 URI（统一资源标识符），其格式如下所示（包括引号）：

```
"file:///drive|/folder 1/folder 2/.../filename"
```

例如，如果要在驱动器 C 上创建名为 config 的文件夹，并将其放在 Program Files/MyApp 文件夹中，请使用以下命令：

```
FLfile.createFolder("file:///C:/Program Files/MyApp/config");
```

如果还要将名为 config.ini 的文件放入该文件夹中，请使用以下命令：

```
FLfile.write("file:///C:/Program Files/MyApp/config/config.ini", "");
```

若要在 Macintosh 上创建文件夹，可以使用以下命令：

```
FLfile.createFolder("file:///Macintosh/MyApp/config");
```

方法摘要

可以对 FLfile 对象使用以下方法：

方法	描述
<a href="#">FLfile.copy()</a>	复制文件。
<a href="#">FLfile.createFolder()</a>	创建一个或多个文件夹。
<a href="#">FLfile.exists()</a>	确定文件或文件夹是否存在。
<a href="#">FLfile.getAttributes()</a>	查明文件是否为可写入、只读、隐藏、显示，或是否为系统文件夹。
<a href="#">FLfile.getCreationDate()</a>	指定从 1970 年 1 月 1 日至文件或文件夹创建时间之间的秒数。
<a href="#">FLfile.getCreationDateObj()</a>	获取文件或文件夹的创建日期。

方法	描述
<a href="#">FLfile.getModificationDate()</a>	指定从 1970 年 1 月 1 日至文件或文件夹上次修改时间之间的秒数。
<a href="#">FLfile.getModificationDateObj()</a>	获取文件或文件夹上次修改的日期。
<a href="#">FLfile.getSize()</a>	获取文件的大小。
<a href="#">FLfile.listFolder()</a>	列出文件夹的内容。
<a href="#">FLfile.platformPathToURI()</a>	将采用平台特有格式的文件名转换为 file:/// URI。
<a href="#">FLfile.read()</a>	读取文件的内容。
<a href="#">FLfile.remove()</a>	删除文件或文件夹。
<a href="#">FLfile.setAttributes()</a>	使文件或文件夹具有只读、可写入、隐藏或显示属性。
<a href="#">FLfile.uriToPlatformPath()</a>	将形如 file:/// URI 的文件名转换为平台特有的格式。
<a href="#">FLfile.write()</a>	创建、写入或追加到文件。

## FLfile.copy()

可用性

Flash MX 2004 7.2。

用法

```
FLfile.copy(fileURI, copyURI)
```

参数

**fileURI** 一个字符串，表示为 file:/// URI，它指定要复制的文件。

**copyURI** 一个字符串，表示为 file:/// URI，它指定所复制文件的位置和名称。

返回

一个布尔值；如果成功，则为 true；否则为 false。

说明

方法；将文件从一个位置复制到另一个位置。如果 copyURI 已经存在，此方法将返回 false。

示例

下面的示例创建名为 **config.ini** 的配置文件的备份副本，并在重新命名后将其放在原来的文件所在的同一文件夹下：

```
var originalFileURI="file:///C:/Program Files/MyApp/config.ini";
var newFileURI="file:///C:/Program Files/MyApp/config_backup.ini";
FLfile.copy(originalFileURI, newFileURI);
```

如果您愿意，可以只使用一个命令来执行相同的任务：

```
FLfile.copy("file:///C:/Program Files/MyApp/config.ini", file:///C:/Program Files/MyApp/config_backup.ini");
```

## FLfile.createFolder()

可用性

Flash MX 2004 7.2。

用法

```
FLfile.createFolder(folderURI)
```

参数

**folderURI** 文件夹 URI，它指定要创建的文件夹结构。

返回

一个布尔值；如果成功，则为 **true**；如果 **folderURI** 已存在，则为 **false**。

说明

方法；在指定位置创建一个或多个文件夹。

一次可以创建多个文件夹。例如，如果 **MyData** 和 **TempData** 文件夹尚不存在，则以下命令将创建这两个文件夹：

```
FLfile.createFolder("file:///c:/MyData/TempData")
```

示例

下面的示例在配置文件夹 (**fl.configURI**) 下创建一个文件夹和一个子文件夹：

```
fl.trace(FLfile.createFolder(fl.configURI+"folder01/subfolder01"));
```

下面的示例尝试在驱动器 C 的根目录下创建名为 **tempFolder** 的文件夹，并显示一个警告框来指示操作是否成功：

```
var folderURI = "file:///c:/tempFolder";
if (FLfile.createFolder(folderURI)) {
    alert("Created " + folderURI);
}
else {
    alert(folderURI + " already exists");
}
```

另请参见

[FLfile.remove\(\)](#)、[FLfile.write\(\)](#)

## FLfile.exists()

可用性

Flash MX 2004 7.2。

用法

```
FLfile.exists(fileURI)
```

参数

**fileURI** 一个字符串，表示为 **file:/// URI**，它指定要验证的文件。

## FLfile 对象

## 返回

一个布尔值；如果成功，则为 `true`；否则为 `false`。

## 说明

方法；确定指定的文件是否存在。如果指定文件夹和文件名，该文件夹必须已经存在。若要创建文件夹，请参阅 [FLfile.createFolder\(\)](#)。

## 示例

下面的示例在 `temp` 文件夹中检查名为 `mydata.txt` 的文件，并显示一个指示该文件是否存在的警告框：

```
var fileURI = "file:///c:/temp/mydata.txt";
if (FLfile.exists(fileURI)) {
    alert( fileURI + " exists.");
}
else {
    alert( fileURI + " does not exist.");
}
```

下面的示例检查 `MyApplication` 文件夹中是否存在所需的配置文件。如果文件不存在，则创建它。

```
var configFile = "file:///C:/MyApplication/config.ini";
if (!FLfile.exists(configFile)) {
    FLfile.write(configFile, "");
}
```

另请参见

[FLfile.write\(\)](#)

## FLfile.getAttributes()

## 可用性

Flash MX 2004 7.2。

## 用法

```
FLfile.getAttributes(fileOrFolderURI)
```

## 参数

**fileOrFolderURI** 一个字符串，表示为 `file:/// URI`，它指定要检索其属性的文件或文件夹。

## 返回

一个字符串，它表示指定的文件或文件夹的属性。

如果文件或文件夹不存在，结果将无法预料。在使用此方法之前，应该使用 [FLfile.exists\(\)](#)。

## 说明

方法；返回一个字符串，它表示指定文件或文件夹的属性；或者，如果文件不具有特定属性（即该文件不是只读的、隐藏的等等），将返回一个空字符串。在使用此方法前，应总是使用 [FLfile.exists\(\)](#) 来测试文件或文件夹是否存在。

该字符串中的字符表示以下属性：

- R - `fileOrFolderURI` 是只读的
- D - `fileOrFolderURI` 是一个文件夹（目录）

- H - fileOrFolderURI 是隐藏的 (仅适用于 Windows)
- S - fileOrFolderURI 是系统文件或文件夹 (仅适用于 Windows)
- A - fileOrFolderURI 已准备好进行归档 (仅适用于 Windows)

例如, 如果 fileOrFolderURI 是隐藏文件夹, 则返回的字符串为 "DH"。

#### 示例

下面的示例获取文件 mydata.txt 的属性, 如果文件是只读的, 则会显示一个警告框。

```
var URI = "file:///c:/temp/mydata.txt";
if (FLfile.exists(URI)){
    var attr = FLfile.getAttributes(URI);
    if (attr && (attr.indexOf("R") != -1)) { // Returned string contains R.
        alert(URI + " is read only!");
    }
}
```

另请参见

[FLfile.setAttributes\(\)](#)

## FLfile.getCreationDate()

可用性

Flash MX 2004 7.2。

用法

```
FLfile.getCreationDate(fileOrFolderURI)
```

参数

**fileOrFolderURI** 一个字符串, 表示为 file:/// URI, 它指定要将其创建日期和时间作为十六进制字符串检索的文件或文件夹。

返回

一个字符串, 它包含一个十六进制数字, 该数字表示从 1970 年 1 月 1 日至文件或文件夹创建时间之间的秒数; 如果文件或文件夹不存在, 则返回 00000000。

描述

方法; 指定从 1970 年 1 月 1 日至文件或文件夹创建时间之间的秒数。此方法主要用于比较文件或文件夹的创建或修改日期。

示例

下面的示例确定文件自创建以来是否进行过修改:

```
// Make sure the specified file exists
var fileURI = "file:///C:/MyApplication/MyApp fla";
var creationTime = FLfile.getCreationDate(fileURI);
var modificationTime = FLfile.getModificationDate(fileURI);
if ( modificationTime > creationTime ) {
    alert("The file has been modified since it was created.");
}
else {
    alert("The file has not been modified since it was created.");
}
```

另请参见

[FLfile.getCreationDateObj\(\)](#)、[FLfile.getModificationDate\(\)](#)

## FLfile.getCreationDateObj()

可用性

Flash MX 2004 7.2。

用法

```
FLfile.getCreationDateObj(fileOrFolderURI)
```

参数

**fileOrFolderURI** 一个字符串，表示为 **file:///** URI，它指定要将其创建日期和时间作为 **JavaScript Date** 对象检索的文件或文件夹。

返回

一个 **JavaScript Date** 对象，它表示创建指定的文件或文件夹的日期和时间。如果文件不存在，该对象将包含一条信息，指示文件或文件夹是在 1969 年 12 月 31 日午夜 (GMT) 创建的。

说明

方法；返回一个 **JavaScript Date** 对象，它表示创建指定的文件或文件夹的日期和时间。

示例

下面的示例在“输出”面板中以易于阅读的形式显示文件的创建日期：

```
// Make sure the specified file exists.  
var file1Date = FLfile.getCreationDateObj("file:///c:/temp/file1.txt");  
fl.trace(file1Date);
```

另请参见

[FLfile.getCreationDate\(\)](#)、[FLfile.getModificationDateObj\(\)](#)

## FLfile.getModificationDate()

可用性

Flash MX 2004 7.2。

用法

```
FLfile.getModificationDate(fileOrFolderURI)
```

参数

**fileOrFolderURI** 一个字符串，表示为 **file:///** URI，它指定要将其修改日期和时间作为十六进制字符串检索的文件。

返回

一个字符串，它包含一个十六进制数字，该数字表示从 1970 年 1 月 1 日至文件或文件夹上次修改时间之间的秒数；如果文件不存在，则返回 00000000。

#### 描述

方法；指定从 1970 年 1 月 1 日至文件或文件夹上次修改时间之间的秒数。此方法主要用于比较文件或文件夹的创建或修改日期。

#### 示例

下面的示例比较两个文件的修改日期，并确定其中哪个文件的修改时间比较靠后：

```
// Make sure the specified files exist.
file1 = "file:///C:/MyApplication/MyApp fla";
file2 = "file:///C:/MyApplication/MyApp.as";
modificationTime1 = FLfile.getModificationDate(file1);
modificationTime2 = FLfile.getModificationDate(file2) ;
if(modificationTime1 > modificationTime2) {
    alert("File 2 is older than File 1") ;
}
else if(modificationTime1 < modificationTime2) {
    alert("File 1 is older than File 2") ;
}
else {
    alert("File 1 and File 2 were saved at the same time") ;
}
```

另请参见

[FLfile.getCreationDate\(\)](#)、[FLfile.getModificationDateObj\(\)](#)

## FLfile.getModificationDateObj()

可用性

Flash MX 2004 7.2。

用法

```
FLfile.getModificationDateObj(fileOrFolderURI)
```

参数

**fileOrFolderURI** 一个字符串，表示为 **file:///** URI，它指定要将其修改日期和时间作为 **JavaScript Date** 对象检索的文件或文件夹。

返回

一个 **JavaScript Date** 对象，它表示上次修改指定的文件或文件夹的日期和时间。如果文件或文件夹不存在，则该对象将包含一条信息，指示文件或文件夹是在 1969 年 12 月 31 日午夜 (GMT) 创建的。

说明

方法；返回一个 **JavaScript Date** 对象，它表示上次修改指定的文件或文件夹的日期和时间。

示例

下面的示例在“输出”面板中以易于阅读的形式显示文件上次修改的日期：

```
// Make sure the specified file exists.
var file1Date = FLfile.getModificationDateObj("file:///c:/temp/file1.txt");
trace(file1Date);
```



另请参见

[FLfile.getCreationDateObj\(\)](#)、[FLfile.getModificationDate\(\)](#)

## FLfile.getSize()

可用性

Flash MX 2004 7.2。

用法

```
FLfile.getSize(fileURI)
```

参数

**fileURI** 一个字符串，表示为 **file:/// URI**，它指定要检索其大小的文件。

返回

一个表示指定文件大小（以字节为单位）的整数；如果文件不存在，则返回 **0**。

说明

方法；返回一个表示指定文件大小（以字节为单位）的整数；如果文件不存在，则返回 **0**。如果返回值为 **0**，则可以使用 [FLfile.exists\(\)](#) 来确定是否文件为零字节文件，或者文件不存在。

只有文件的大小小于或等于 **2GB** 时，此方法才能返回正确的文件大小值。

示例

下面的示例在 **fileSize** 变量中存储 **mydata.txt** 文件的大小：

```
var URL = "file:///c:/temp/mydata.txt";  
var fileSize = FLfile.getSize(URL);
```

## FLfile.listFolder()

可用性

Flash MX 2004 7.2。

用法

```
FLfile.listFolder(folderURI [, filesOrDirectories])
```

参数

**folderURI** 一个字符串，表示为 **file:/// URI**，它指定要检索其内容的文件夹。您可以将通配符掩码作为 **folderURI** 的一部分提供。有效的通配符是 **\***（匹配一个或多个字符）和 **?**（匹配单个字符）。

**filesOrDirectories** 一个可选字符串，它指定是只返回文件名，还是只返回文件夹（目录）名称。如果省略，则同时返回文件名和文件夹名称。可接受值为 **"files"** 和 **"directories"**。

返回

表示文件夹内容的字符串的数组。如果文件夹不存在，或没有与指定条件相匹配的文件或文件夹，则返回空数组。

## FLfile 对象

## 描述

方法；返回表示文件夹内容的字符串的数组。

## 示例

下面的示例返回三个数组。第一个表示 C:\temp 文件夹中的所有文件，第二个表示 C:\temp 文件夹中的所有文件夹，第三个表示 C:\temp 文件夹中的文件和文件夹：

```
var fileURI = "file:///C:/temp/" ;
var folderURI = "file:///C:/temp" ;
var fileList1 = FLfile.listFolder(fileURI, "files"); // files
var fileList2 = FLfile.listFolder(folderURI, "directories"); //folders
var fileList3 = FLfile.listFolder(folderURI); //files and folders
fl.trace("Files: " + fileList1);
fl.trace("");
fl.trace("Folders: " + fileList2);
fl.trace("");
fl.trace("Files and folders: " + fileList3);
```

下面的示例返回包含 temp 文件夹中的所有文本 (.txt) 文件的数组，并在警告框内显示该列表：

```
var folderURI = "file:///c:/temp";
var fileMask = "*.txt";
var list = FLfile.listFolder(folderURI + "/" + fileMask, "files");
if (list) {
    alert(folderURI + " contains: " + list.join(" "));
}
```

下面的示例在指定的 folderURI 中使用文件掩码，以返回 Windows 应用程序文件夹中所有可执行文件的名称：

```
var executables = FLfile.listFolder("file:///C:/WINDOWS/*.exe", "files");
alert(executables.join("\n"));
```

## FLfile.platformPathToURI()

## 可用性

Flash CS4 Professional.

## 用法

```
FLfile.platformPathToURI(fileName)
```

## 参数

**fileName** 一个字符串，采用特定于平台的格式表示，它指定要转换的文件名。

## 返回

一个表示为 file:/// URI 的字符串。

## 说明

方法；将采用平台特有格式的文件名转换为 file:/// URI。

## 示例

下面的示例将采用平台特有格式的文件名转换为 file:/// URI，然后传递给 outputPanel.save()：

```
var myFilename = "C:\\outputPanel.txt";
var myURI=FLfile.platformPathToURI(myFilename);
fl.outputPanel.save(myURI);
```

另请参见

[FLfile.uriToPlatformPath\(\)](#)

## FLfile.read()

可用性

Flash MX 2004 7.2。

用法

```
FLfile.read()
```

参数

**fileOrFolderURI** 一个字符串，表示为 **file:/// URI**，它指定要检索其属性的文件或文件夹。

返回

以字符串形式表示的指定文件的内容，如果读取失败，则返回 **null**。

说明

方法；将以字符串的形式返回指定文件的内容，如果读取失败，则返回 **null**。

示例

下面的示例读取文件 **mydata.txt**，如果成功，则显示带有该文件内容的警告框。

```
var fileURI = "file:///c:/temp/mydata.txt";
var str = FLfile.read( fileURI);
if (str) {
    alert( fileURL + " contains: " + str);
}
```

下面的示例从类文件中读取 **ActionScript** 代码，并将其存储在 **code** 变量中：

```
var classFileURI = "file:///C:/MyApplication/TextCarousel.as";
var code = FLfile.read(classFileURI);
```

## FLfile.remove()

可用性

Flash MX 2004 7.2。

用法

```
FLfile.remove(fileOrFolderURI)
```

参数

**fileOrFolderURI** 一个字符串，表示为 **file:/// URI**，它指定要删除的文件或文件夹。

返回

一个布尔值；如果成功，则为 **true**；否则为 **false**。

说明

方法；删除指定的文件或文件夹。如果文件夹中包含文件，则这些文件也将被删除。具有 **R**（只读）属性的文件将无法删除。

示例

下面的示例在文件存在时对用户发出警告，如果用户选择删除该文件，则进行删除：

```
var fileURI = prompt ("Enter file/folder to be deleted: ", "file:///c:/temp/delete.txt");
if (FLfile.exists(fileURI)) {
    var confirm = prompt("File exists. Delete it? (y/n)", "y");
    if (confirm == "y" || confirm == "Y") {
        if(FLfile.remove(fileURI)) {
            alert(fileURI + " is deleted.");
        }
        else {
            alert("fail to delete " + fileURI);
        }
    }
}
else {
    alert(fileURI + " does not exist");
}
```

下面的示例删除由某个应用程序创建的配置文件：

```
if(FLfile.remove("file:///C:/MyApplication/config.ini")) {
    alert("Configuration file deleted");
}
```

下面的示例删除 **Configuration** 文件夹及其内容：

```
FLfile.remove("file:///C:/MyApplication/Configuration/");
```

另请参见

[FLfile.createFolder\(\)](#)、[FLfile.getAttributes\(\)](#)

## FLfile.setAttributes()

可用性

Flash MX 2004 7.2。

用法

```
FLfile.setAttributes(fileURI, strAttrs)
```

参数

**fileURI** 以 **file:/// URI** 形式表示的字符串，指定要设置其属性的文件。

**strAttrs** 一个字符串，它指定要设置的属性的值。有关 **strAttrs** 的可接受值，请参阅下面的“描述”部分。

返回

如果成功，则返回布尔值 **true**。

注：如果文件或文件夹不存在，结果将无法预料。在使用此方法之前，应该使用 [FLfile.exists\(\)](#)。

说明

方法；为指定文件指定系统级别的属性。

以下值是 `strAttrs` 的有效值:

- N - 无特定的属性 (非只读、非隐藏等)
- A - 准备好进行归档 (仅适用于 Windows)
- R - 只读 (在 Macintosh 上, 只读意味着“被锁定”)
- W - 可写入 (覆盖 R)
- H - 隐藏 (仅适用于 Windows)
- V - 可见 (覆盖 H, 仅适用于 Windows)

如果 `strAttrs` 中同时包括 R 和 W, 则 R 会被忽略, 文件将设置为可写入。类似地, 如果传递 H 和 V, 则 H 会被忽略, 文件将设置为可见。

如果要确保不设置归档属性, 请在设置属性之前使用此命令及 N 参数。换句话说, 没有与 A 相对的、能够直接禁用归档属性的方法。

示例

下面的示例将文件 `mydata.txt` 设置为只读且隐藏。它对归档属性没有任何影响。

```
var URI = "file:///c:/temp/mydata.txt";
if (FLfile.exists(URI)) {
    FLfile.setAttributes(URI, "RH");
}
```

下面的示例将文件 `mydata.txt` 设置为只读且隐藏。它还确保不设置归档属性。

```
var URI = "file:///c:/temp/mydata.txt";

if (FLfile.exists(URI)) {
    FLfile.setAttributes(URI, "N");
    FLfile.setAttributes(URI, "RH");
}
```

另请参见

[FLfile.getAttributes\(\)](#)

## FLfile.uriToPlatformPath()

可用性

Flash CS4 Professional.

用法

```
FLfile.uriToPlatformPath(fileURI)
```

参数

**fileURI** 一个字符串, 表示为 `file:/// URI`, 它指定要转换的文件名。

返回

一个表示特定于平台的路径的字符串。

说明

方法; 将形如 `file:/// URI` 的文件名转换为平台特有的格式。

## FLfile 对象

## 示例

下面的示例将 `file:/// URI` 转换为平台特有的格式：

```
var dir = (fl.configDirectory);
var URI = FLfile.platformPathToURI(dir);
fl.trace(URI == fl.configURI); // displays "true"
```

另请参见

[FLfile.platformPathToURI\(\)](#)

## FLfile.write()

## 可用性

Flash MX 2004 7.2。

## 用法

```
FLfile.write(fileURI, textToWrite, [ , strAppendMode])
```

## 参数

**fileURI** 一个字符串，表示为 `file:/// URI`，它指定要向其写入内容的文件。

**textToWrite** 一个字符串，它表示要放在文件中的文本。

**strAppendMode** 一个值为 "append" 的可选字符串，它指定要将 `textToWrite` 追加到现有文件中。如果省略，则 `fileURI` 会被 `textToWrite` 覆盖。

## 返回

一个布尔值；如果成功，则为 `true`；否则为 `false`。

## 说明

方法；将指定的字符串写入到指定的文件中（作为 UTF-8）。如果指定的文件不存在，则创建该文件。不过，要放置该文件的文件夹必须存在，才能使用此方法。若要创建文件夹，请使用 [FLfile.createFolder\(\)](#)。

## 示例

下面的示例尝试将字符串 "xxx" 写入到文件 `mydata.txt` 中，如果写入成功，则会显示警告消息。然后，该示例尝试将字符串 "aaa" 追加到文件中，如果写入成功，则会显示第二个警告消息。此脚本执行完毕后，文件 `mydata.txt` 将只包含文本 "xxxaaa"。

```
var URI = "file:///c:/temp/mydata.txt";
if (FLfile.write(URI, "xxx")) {
    alert("Wrote xxx to " + URI);
}
if (FLfile.write(URI, "aaa", "append")) {
    alert("Appended aaa to " + fileURI);
}
```

另请参见

[FLfile.createFolder\(\)](#)、[FLfile.exists\(\)](#)

## 第 19 章 : folderItem 对象

继承关系 [Item 对象](#) > folderItem 对象

可用性

Flash MX 2004。

描述

folderItem 对象是 Item 对象的子类。folderItem 没有专用方法或属性。请参阅 [Item 对象](#)。

## 第 20 章 : fontItem 对象

继承关系 [Item 对象](#) > fontItem 对象

可用性

Flash MX 2004。

描述

fontItem 对象是 Item 对象的子类（请参阅 [Item 对象](#)）。

属性摘要

除 Item 对象的属性外，fontItem 对象还具有以下属性：

属性	描述
<a href="#">fontItem.bitmap</a>	指定字体项目是否为位图字体。
<a href="#">fontItem.bold</a>	指定字体项目是否为粗体字体。
<a href="#">fontItem.embeddedCharacters</a>	指定要嵌入的字符。
<a href="#">fontItem.embedRanges</a>	指定在“字体嵌入”对话框中可以选择的项目。
<a href="#">fontItem.embedVariantGlyphs</a>	指定发布 SWF 文件时，是否应在字体中输出变体字型。
<a href="#">fontItem.font</a>	与字体项目关联的设备字体的名称。
<a href="#">fontItem.isDefineFont4Symbol</a>	指定发布 SWF 文件时，输出字体的格式。
<a href="#">fontItem.italic</a>	指定字体项目是否为斜体字体。
<a href="#">fontItem.size</a>	字体项目的大小（以点为单位）。

### fontItem.bitmap

可用性

Flash CS4 Professional.

用法

```
fontItem.bitmap
```

描述

属性；一个布尔值，它指定字体项目是 (true) 否 (false) 为位图字体。

示例

假定库中的第一个项目是字体项目，如果该项目为位图字体，则下面的代码在“输出”面板中显示 true；否则显示 false：

```
var theItem = fl.getDocumentDOM().library.items[0];
fl.trace("bitmap: "+ theItem.bitmap);
```



## fontItem.bold

可用性

Flash CS4 Professional.

用法

fontItem.bold

描述

属性；一个布尔值，它指定字体项目是 (**true**) 否 (**false**) 为粗体字体。

示例

假定库中的第一个项目是字体项目，如果该项目为粗体字体，则下面的代码在“输出”面板中显示 **true**；否则显示 **false**，然后将该项目设置为粗体字体。

```
var theItem = fl.getDocumentDOM().library.items[0];
fl.outputPanel.clear();
fl.trace("bold: " + theItem.bold);
theItem.bold=true;
fl.trace("bold: " + theItem.bold);
```

## fontItem.embeddedCharacters

可用性

Flash CS5 Professional.

用法

fontItem.embeddedCharacters

描述

属性；一个字符串值，您可以指定要嵌入 SWF 文件的字符，以便最终播放该 SWF 文件的设备上不必存在这些字符。此属性与“字体嵌入”对话框具有相同的功能。

也可以读取此属性，您可以查看使用“字体嵌入”对话框为给定的字体项目指定了哪些字符。

示例

假定库中的第一个项目为字体项目，下面的代码将嵌入字符 **a**、**b** 和 **c**。

```
fl.getDocumentDOM().library.items[0].embeddedCharacters = "abc";
```

## fontItem.embedRanges

可用性

Flash CS5 Professional.

用法

fontItem.embedRanges

#### 描述

属性；一个字符串值，指定一系列分隔的整数，这些整数与在“字体嵌入”对话框中可以选择的项目对应。

也可以读取此属性，您可以查看使用“字体嵌入”对话框为给定的字体项目指定了哪些字符。

注：范围编号与配置文件夹中找到的 FontEmbedding/UnicodeTables.xml 文件相对应。

#### 示例

假定库中第一个项目为字体项目，下面的代码将嵌入由整数 1、3 和 7 标识的范围。

```
fl.getDocumentDOM().library.items[0].embedRanges = "1|3|7";
```

假定库中第一个项目为字体项目，下面的代码将重置要嵌入的范围。

```
fl.getDocumentDOM().library.items[0].embedRanges = "";
```

## fontItem.embedVariantGlyphs

#### 可用性

Flash CS4 Professional.

#### 用法

```
fontItem.embedVariantGlyphs
```

#### 描述

注：虽然此属性在 Flash CS5 Professional 中可用，但在应用于 Text Layout Framework (TLF) 文本时不起作用。从 Flash Professional CS5 开始，始终在用于 TLF 文本的字体中嵌入变体字型。下面引用的 flash.text.engine (FTE) 仅在 Flash Professional CS4 中可用。

属性；一个布尔值，它指定发布 SWF 文件时，是 (true) 否 (false) 应在字体中输出变体字型。将此值设置为 true 会增加 SWF 文件的大小。默认值为 false。

某些语言会在您键入时动态替换字符字型，例如泰语、阿拉伯语、希伯来语和希腊语。如果正在布局或输入这类语言的文本，请将此属性设置为 true。

#### 示例

与 flash.text API 兼容的字体元件会显示在库中，用户可直接管理它们。但库中不显示与 flash.text.engine (FTE) API 兼容的字体元件，因此必须手动管理。下面的函数向库中添加可配合 FTE API 使用的新字体。

**fontItem** 对象

```
function embedFontSymbol(symbolName, fontName, includeVariants) {
    var doc = fl.getDocumentDOM();
    if (doc) {
        // look up the item. if it exists, delete it.
        var index = doc.library.findItemIndex(symbolName);
        if (index > -1)
            doc.library.deleteItem(symbolName);

        // make a new font symbol in the library
        doc.library.addNewItem('font', symbolName);

        // look up the symbol by its name
        var index = doc.library.findItemIndex(symbolName);
        if (index > -1) {
            // get the item from the library and set the attributes of interest
            var fontObj = doc.library.items[index];
            fontObj.isDefineFont4Symbol = true;
            fontObj.font = fontName;
            fontObj.bold = false;
            fontObj.italic = false;
            fontObj.embedVariantGlyphs = includeVariants;
            // this is what forces the font into the SWF stream
            fontObj.linkageExportForAS = true;
            fontObj.linkageExportInFirstFrame = true;
        }
    }
}
```

下面的函数在“输出”面板中显示所有字体元件。

```
function dumpFontSymbols()
{
    var doc = fl.getDocumentDOM();
    if (doc) {
        var items = doc.library.items;
        fl.trace("items length = " + items.length);
        var i;
        for(i=0; i<items.length; i++) {
            var item = items[i];
            fl.trace("itemType = " + item.itemType);
            if (item.itemType == 'font') {
                fl.trace("name = " + item.name);
                fl.trace("DF4 symbol = " + item.isDefineFont4Symbol);
                fl.trace("font = " + item.font);
            }
        }
    }
}
```

另请参见

[fontItem.isDefineFont4Symbol](#)、[text.embedVariantGlyphs](#)

## fontItem.font

可用性

Flash CS4 Professional.

**fontItem** 对象

## 用法

`fontItem.font`

## 描述

属性；一个字符串，它指定与字体项目关联的设备字体的名称。如果输入的字符串与已安装的设备字体不对应，则会显示一条错误消息。若要确定系统上是否存在某种字体，请使用 [fl.isFontInstalled\(\)](#)。

注：设置此值时，所得的属性值可能不同于所输入的字符串。请参阅下面的示例。

## 示例

假定库中的第一个项目是字体项目，下面的代码显示当前与该字体项目关联的设备字体的名称，然后将其更改为 **Times**：

```
fl.outputPanel.clear();
var theItem = fl.getDocumentDOM().library.items[0];
fl.trace(theItem.font);
theItem.font = "Times";
// depending on your system, the following may display something like "Times-Roman"
fl.trace(theItem.font);
```

## fontItem.isDefineFont4Symbol

## 可用性

Flash CS4 Professional.

## 用法

`fontItem.isDefineFont4Symbol`

## 描述

属性；一个布尔值，它指定发布 SWF 文件时，输出字体的格式。如果此值为 **true**，Flash 将输出可配合 `flash.text.engine` (FTE) API 使用的字体。如果此值为 **false**，字体将可配合 `flash.text` API 使用，包括文本字段。默认值为 **false**。

## 示例

请参阅 [fontItem.embedVariantGlyphs](#)。

## fontItem.italic

## 可用性

Flash CS4 Professional.

## 用法

`fontItem.italic`

## 描述

属性；一个布尔值，它指定字体项目是 (**true**) 否 (**false**) 为斜体字体。

## 示例

假定库中的第一个项目是字体项目，如果该项目为斜体字体，则下面的代码在“输出”面板中显示 **true**；否则显示 **false**，然后将该项目设置为斜体字体：

```
var theItem = fl.getDocumentDOM().library.items[0];  
fl.outputPanel.clear();  
fl.trace("italic: " + theItem.italic);  
theItem.italic=true;  
fl.trace("italic: " + theItem.italic);
```

## fontItem.size

可用性

Flash CS4 Professional.

用法

fontItem.size

描述

属性；一个整数，它表示字体项目的大小（以点为单位）。

示例

假定库中的第一个项目是字体项目，下面的代码在“输出”面板中显示该项目的点值，然后将其设置为 24。

```
var theItem = fl.getDocumentDOM().library.items[0];  
fl.outputPanel.clear();  
fl.trace("font size: " + theItem.size);  
theItem.size=24;  
fl.trace("font size: " + theItem.size);
```

## 第 21 章 : Frame 对象

### 可用性

Flash MX 2004。

### 描述

**Frame** 对象表示图层中的帧。

### 方法摘要

以下方法可用于 **Frame** 对象：

方法	描述
<a href="#">frame.convertMotionObjectTo2D()</a>	将选定的动画对象转换为 2D 动画对象。
<a href="#">frame.convertMotionObjectTo3D()</a>	将选定的动画对象转换为 3D 动画对象。
<a href="#">frame.getCustomEase()</a>	返回一个 JavaScript 对象数组，数组中的每个对象都具有 x 和 y 属性。
<a href="#">frame.getMotionObjectXML()</a>	从选定的动画对象返回动画 XML。
<a href="#">frame.hasMotionPath()</a>	告知当前选定内容是否具有补间动画。
<a href="#">frame.is3DMotionObject()</a>	告知当前选定内容是否为 3D 动画对象。
<a href="#">frame.isMotionObject()</a>	告知当前选定内容是否为动画对象。
<a href="#">frame.selectMotionPath()</a>	选择或取消选择当前动画对象的运动路径。
<a href="#">frame.setCustomEase()</a>	指定要用作自定义缓动曲线的三次贝塞尔曲线。
<a href="#">frame.setMotionObjectDuration()</a>	指定当前选定动画对象的持续时间（补间范围长度）。
<a href="#">frame.setMotionObjectXML()</a>	将指定动画 XML 应用于选定的动画对象。

### 属性摘要

以下属性可用于 **Frame** 对象：

属性	描述
<a href="#">frame.actionScript</a>	一个字符串，它表示 ActionScript 代码。
<a href="#">frame.duration</a>	只读；一个整数，它表示帧序列中帧的数量。
<a href="#">frame.elements</a>	只读；Element 对象的数组（请参阅 <a href="#">Element</a> 对象）。
<a href="#">frame.hasCustomEase</a>	一个布尔值，它指定帧是否从自定义缓入缓出曲线中获取缓动信息。
<a href="#">frame.labelType</a>	一个字符串，它指定 Frame 名称的类型。
<a href="#">frame.motionTweenOrientToPath</a>	一个布尔值，它指定补间元素在沿着路径移动时是否旋转，以便与路径上的每个点保持一定的角度。
<a href="#">frame.motionTweenRotate</a>	一个字符串，它指定补间元素如何旋转。
<a href="#">frame.motionTweenRotateTimes</a>	一个整数，它指定补间元素在起始关键帧和下一关键帧之间旋转的次数。
<a href="#">frame.motionTweenScale</a>	一个布尔值，它指定补间元素是缩放为下一关键帧中对象的大小，即随补间中的每一帧放大 (true)，还是不进行缩放 (false)。

属性	描述
<code>frame.motionTweenSnap</code>	一个布尔值，它指定补间元素是自动贴紧至与此帧所处图层关联的运动引导层上的最近点 ( <code>true</code> )，还是不贴紧 ( <code>false</code> )。
<code>frame.motionTweenSync</code>	一个布尔值，如果设置为 <code>true</code> ，则补间对象的动画和主时间轴同步。
<code>frame.name</code>	一个字符串，它指定帧的名称。
<code>frame.shapeTweenBlend</code>	一个字符串，它指定补间形状在补间开始的关键帧中的形状和下一个关键帧中的形状之间如何混合。
<code>frame.soundEffect</code>	一个字符串，它指定直接附加在帧上的声音 ( <code>frame.soundLibraryItem</code> ) 的效果。
<code>frame.soundLibraryItem</code>	用于创建声音的库项目 (请参阅 <a href="#">SoundItem</a> 对象)。
<code>frame.soundLoop</code>	一个整数值，它指定直接附加在帧上的声音 ( <code>frame.soundLibraryItem</code> ) 的播放次数。
<code>frame.soundLoopMode</code>	一个字符串，它指定直接附加在帧上的声音 ( <code>frame.soundLibraryItem</code> ) 是播放特定的次数，还是无限循环播放。
<code>frame.soundName</code>	一个字符串，它指定直接附加在帧上的声音 ( <code>frame.soundLibraryItem</code> ) 的名称 (存储在库中的名称)。
<code>frame.soundSync</code>	一个字符串，它指定直接附加在帧上的声音 ( <code>frame.soundLibraryItem</code> ) 的同步行为。
<code>frame.startFrame</code>	只读；序列中第一个帧的索引。
<code>frame.tweenEasing</code>	一个整数，它指定应用于补间对象的缓动数量。
<code>frame.tweenInstanceName</code>	为指定的动画对象分配一个实例名称。
<code>frame.tweenType</code>	一个字符串，它指定补间的类型。
<code>frame.useSingleEaseCurve</code>	一个布尔值，它指定是否将单个自定义缓入缓出曲线用作所有属性的缓动信息。

## frame.convertMotionObjectTo2D()

可用性

Flash Professional CS5。

用法

```
frame.convertMotionObjectTo2D()
```

描述

方法；将选定的动画对象转换为 2D 动画对象。

示例

下面的示例将选定的动画对象转换为 2D 动画对象：

## Frame 对象

```
var doc = fl.getDocumentDOM();
var my_tl = doc.getTimeline();
this.getCurrentFrame = function(){
var layer = my_tl.layers[my_tl.currentLayer];
var frame = layer.frames[my_tl.currentFrame];
return frame;
}
var theFrame = getCurrentFrame();
if(theFrame.isMotionObject() && the()){
theFrame.convertMotionObjectTo2D();
}else{
fl.trace("It isn't motion or it's already a 2D motion");
}
```

## frame.convertMotionObjectTo3D()

可用性

Flash Professional CS5。

用法

```
frame.convertMotionObjectTo3D()
```

描述

方法；将选定的动画对象转换为 3D 动画对象。

示例

下面的示例将选定的动画对象转换为 3D 动画对象：

```
var doc = fl.getDocumentDOM();
var my_tl = doc.getTimeline();
this.getCurrentFrame = function(){
var layer = my_tl.layers[my_tl.currentLayer];
var frame = layer.frames[my_tl.currentFrame];
return frame;}
var theFrame = getCurrentFrame();
if(theFrame.isMotionObject() && !theFrame.is3DMotionObject()){
theFrame.convertMotionObjectTo3D();
}else{
fl.trace("It isn't motion or it's already a 3D motion");
}
```

## frame.actionScript

可用性

Flash MX 2004。

用法

```
frame.actionScript
```

描述

属性；一个字符串，它表示 ActionScript 代码。若要插入一个换行符，请使用 "\n"。



### 示例

下面的示例将 `stop()` 分配给顶部图层中第一帧的动作：

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].actionScript = 'stop()';
```

## frame.duration

### 可用性

Flash MX 2004。

### 用法

```
frame.duration
```

### 描述

只读属性；一个整数，它表示帧序列中帧的数量。

### 示例

下面的示例将从顶部图层的第一帧开始的帧序列中的帧数存储在 `frameSpan` 变量中：

```
var frameSpan = fl.getDocumentDOM().getTimeline().layers[0].frames[0].duration;
```

## frame.elements

### 可用性

Flash MX 2004。

### 用法

```
frame.elements
```

### 描述

只读属性；**Element** 对象的数组（请参阅 [Element 对象](#)）。数组中的元素顺序就是它们在 **FLA** 文件中的存储顺序。如果在舞台上存在多个形状，并且每个形状都未组合，则 **Flash** 将它们作为单个元素处理。如果每个形状都经过组合，舞台上便存在多个组，**Flash** 将这些组作为单独的元素处理。换句话说，**Flash** 将所有原始的未组合形状作为单个元素处理，无论舞台上存在多少单独的形状。举例来说，如果一个帧包含三个原始的未组合形状，则该帧中的 `elements.length` 返回值 **1**。若要解决此问题，请分别选择每个形状，然后将其组合在一起。

### 示例

下面的示例将顶部图层第一帧中的一组当前元素存储在 `myElements` 变量中：

```
var myElements = fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements;
```

## frame.getCustomEase()

### 可用性

Flash 8。

**Frame 对象**

## 用法

```
Frame.getCustomEase([property])
```

## 参数

**属性** 一个可选字符串，它指定要为其返回自定义缓动值的属性。可接受值为 "all"、"position"、"rotation"、"scale"、"color" 和 "filters"。默认值为 "all"。

## 返回

返回一个 JavaScript 对象数组，数组中的每个对象均具有 x 和 y 属性。

## 说明

方法；返回对象的数组，用于表示对缓动曲线进行定义的三次贝塞尔曲线的控制点。

## 示例

下面的示例返回顶部图层中第一帧的 **position** 属性的自定义缓动值：

```
var theFrame = fl.getDocumentDOM().getTimeline().layers[0].frames[0]
var easeArray = theFrame.getCustomEase("position");
```

## 另请参见

[frame.hasCustomEase](#)、[frame.setCustomEase\(\)](#)、[frame.useSingleEaseCurve](#)

## frame.getMotionObjectXML()

## 可用性

Flash Professional CS5。

## 用法

```
Frame.getMotionObjectXML()
```

## 描述

从选定的动画对象返回动画 XML 的字符串。

## 示例

下面的示例从选定的动画对象返回动画 XML。

```
var doc = fl.getDocumentDOM();
var my_tl = doc.getTimeline();
this.getCurrentFrame = function(){
var layer = my_tl.layers[my_tl.currentLayer];
var frame = layer.frames[my_tl.currentFrame];
return frame;
}
var theFrame = getCurrentFrame();
if(theFrame.isMotionObject()) {
//fl.trace(theFrame.getMotionObjectXML());
}else{
fl.trace("It is not motion.");
}
}
```

## frame.hasCustomEase

可用性

Flash 8。

用法

```
frame.hasCustomEase
```

描述

属性；一个布尔值。如果为 **true**，帧将从自定义缓动曲线中获取缓动信息。如果为 **false**，帧将从其缓动值中获取缓动信息。

示例

下面的示例指定顶部图层中的第一帧从缓动值而不是从自定义缓入缓出曲线中获取缓动信息：

```
var theFrame = fl.getDocumentDOM().getTimeline().layers[0].frames[0]
theFrame.hasCustomEase = false;
```

另请参见

[frame.getCustomEase\(\)](#)、[frame.setCustomEase\(\)](#)、[frame.useSingleEaseCurve](#)

## frame.hasMotionPath()

可用性

Flash Professional CS5。

用法

```
Frame.hasMotionPath()
```

描述

方法；一个布尔型值。让您了解当前选定内容中是否包含运动路径。

示例

下面的示例返回一个跟踪语句，告知当前选定内容是否具有运动路径。

```
var doc = fl.getDocumentDOM();
var my_tl = doc.getTimeline();
this.getCurrentFrame = function(){
var layer = my_tl.layers[my_tl.currentLayer];
var frame = layer.frames[my_tl.currentFrame];
return frame;
}
var theFrame = getCurrentFrame();
if (theFrame.isMotionObject()){
if (theFrame.hasMotionPath()){
fl.trace("There is a motion path");
}else{
fl.trace("There is no motion path");
}
}
```

## frame.is3DMotionObject()

可用性

Flash Professional CS5。

用法

Frame.is3DMotionObject()

描述

方法；一个布尔型值。让您了解当前选定内容是否为 3D 动画对象。

示例

下面的示例返回一个跟踪语句，告知当前选定内容是否为 3D 动画对象。

```
var doc = fl.getDocumentDOM();
var my_tl = doc.getTimeline();
this.getCurrentFrame = function() {
var layer = my_tl.layers[my_tl.currentLayer];
var frame = layer.frames[my_tl.currentFrame];
return frame;
}
var theFrame = getCurrentFrame();
if(theFrame.isMotionObject() && theFrame.is3DMotionObject()){
fl.trace("This selection is 3D Motion");
}else{
fl.trace("This selection is not 3D motion");
}
```

## frame.isMotionObject()

可用性

Flash Professional CS5。

用法

Frame.isMotionObject()

描述

方法；一个布尔型值。让您了解当前选定内容是否为动画对象。

示例

下面的示例返回一个跟踪语句，告知当前选定内容是否为动画对象。

**Frame 对象**

```
var my_tl = doc.getTimeline()      ;
this.getCurrentFrame = function(){
var layer = my_tl.layers[my_tl.currentLayer];<
var frame = layer.frames[my_tl.currentFrame];
return frame;
}
var theFrame = getCurrentFrame();
if(theFrame.isMotionObject()) {
fl.trace("This selection is motion.");
}else{
fl.trace("This selection is not motion.");
}
```

## frame.labelType

可用性

Flash MX 2004。

用法

frame.labelType

描述

属性；一个字符串，它指定 **Frame** 名称的类型。可接受值为 "none"、"name"、"comment" 和 "anchor"。将标签设置为 "none" 可清除 [frame.name](#) 属性。

示例

下面的示例将顶部图层中第一帧的名称设置为 "First Frame"，然后将其标签设置为 "comment"：

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].name = 'First Frame';
fl.getDocumentDOM().getTimeline().layers[0].frames[0].labelType = 'comment';
```

## frame.motionTweenOrientToPath

可用性

Flash MX 2004。

用法

frame.motionTweenOrientToPath

描述

属性；一个布尔值，它指定补间元素在沿着路径移动时是 (true) 否 (false) 旋转，以便与路径上途经的每个点都保持同样的角度。

如果您希望为此属性指定一个值，应将 [frame.motionTweenRotate](#) 设置为 "none"。

## frame.motionTweenRotate

可用性

Flash MX 2004。

用法

```
frame.motionTweenRotate
```

描述

属性；一个字符串，它指定补间元素如何旋转。可接受值为 "none"、"auto"、"clockwise" 和 "counter-clockwise"。"auto" 值表示对象以某个方向旋转，使得为匹配下一关键帧中对象的旋转所需的动作最少。

如果您希望为 [frame.motionTweenOrientToPath](#) 指定一个值，请将此属性设置为 "none"。

示例

请参阅 [frame.motionTweenRotateTimes](#)。

## frame.motionTweenRotateTimes

可用性

Flash MX 2004。

用法

```
frame.motionTweenRotateTimes
```

描述

属性；一个整数，它指定补间元素在起始关键帧和下一关键帧之间旋转的次数。

示例

下面的示例将帧中的元素在到达下一关键帧前按逆时针方向旋转三次：

```
f1.getDocumentDOM().getTimeline().layers[0].frames[0].motionTweenRotate = "counter-clockwise";  
f1.getDocumentDOM().getTimeline().layers[0].frames[0].motionTweenRotateTimes = 3;
```

## frame.motionTweenScale

可用性

Flash MX 2004。

用法

```
frame.motionTweenScale
```

描述

属性；一个布尔值，它指定补间元素是缩放为下一关键帧中对象的大小，即随补间中的每一帧放大 (**true**)，还是不进行缩放 (**false**)。

### 示例

下面的示例指定补间元素是否缩放以适应以下关键帧中对象的大小，即随着补间中的每一帧放大。

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].motionTweenScale = true;
```

## frame.motionTweenSnap

### 可用性

Flash MX 2004。

### 用法

```
frame.motionTweenSnap
```

### 描述

属性；一个布尔值，它指定补间元素是自动贴紧至与此帧所处图层关联的运动引导层上的最近点 (**true**)，还是不贴紧 (**false**)。

## frame.motionTweenSync

### 可用性

Flash MX 2004。

### 用法

```
frame.motionTweenSync
```

### 描述

属性；一个布尔值，如果设置为 **true**，则补间对象的动画和主时间轴同步。

### 示例

下面的示例指定补间对象应与时间轴同步：

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].motionTweenSync = true;
```

## frame.name

### 可用性

Flash MX 2004。

### 用法

```
frame.name
```

### 描述

属性；一个字符串，它指定帧的名称。

### 示例

下面的示例将顶部图层中第一帧的名称设置为 "First Frame"，然后将 **name** 属性的值存储在 **frameLabel** 变量中：

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].name = 'First Frame';  
var frameLabel = fl.getDocumentDOM().getTimeline().layers[0].frames[0].name;
```

## frame.selectMotionPath()

可用性

Flash Professional CS5。

用法

```
Frame.selectMotionPath()
```

描述

方法；一个布尔型值。选择 (**true**) 或取消选择 (**false**) 当前动画对象的运动路径。

示例

该示例选择或取消选择当前动画对象的运动路径。

```
var doc = fl.getDocumentDOM();  
var my_tl = doc.getTimeline();  
t    his.getCurrentFrame = function(){  
var layer = my_tl.layers[my_tl.c u rrentLayer];  
var frame = layer.frames[my_tl.currentFrame];  
return frame;  
}  
var theFrame = getCurrentFrame();  
if(theFrame.isMotionObject()){  
if (theFrame.hasMotionPath()){  
theFrame.selectMotionPath(true);  
}  
else{  
fl.trace("There is no motion path");  
}  
}else{  
fl.trace("It is no motion");  
}  
}
```

## frame.setCustomEase()

可用性

Flash 8。

用法

```
frame.setCustomEase(property, easeCurve)
```

参数

**属性** 一个字符串，它指定缓动曲线用于的属性。可接受值为 "all"、"position"、"rotation"、"scale"、"color" 和 "filters"。

**easeCurve** 定义缓动曲线的对象数组。组数中的每一个元素都必须为 JavaScript 对象，且具有 x 属性和 y 属性。

返回

无。



## Frame 对象

## 描述

方法；指定控制点和切线端点坐标的数组，以描述用作自定义缓动曲线的三次贝塞尔曲线。该数组由控制点的水平（按从左向右的顺序）位置和切线端点构成。

## 示例

下面的示例将第一图层第一帧中所有属性的缓动曲线设置成由 `easeCurve` 数组指定的贝塞尔曲线：

```
var theFrame = fl.getDocumentDOM().getTimeline().layers[0].frames[0];
var easeCurve = [ {x:0,y:0}, {x:.3,y:.3}, {x:.7,y:.7}, {x:1,y:1} ];
theFrame.setCustomEase( "all", easeCurve );
```

## 另请参见

[frame.getCustomEase\(\)](#)、[frame.hasCustomEase](#)、[frame.useSingleEaseCurve](#)

## frame.setMotionObjectDuration()

## 可用性

Flash Professional CS5。

## 用法

```
Frame.setMotionObjectDuration( duration [, stretchExistingKeyframes] )
```

## 参数

**duration** 为选定动画对象的补间范围指定帧数。

**stretchExistingKeyframes** 一个布尔型值，它确定补间范围是否拉伸至最后一帧的末尾，或者在最后一帧的末尾是否添加了帧。

## 描述

方法；设置当前选定动画对象的持续时间（补间范围长度）。

## 示例

下面的示例指定选定动画对象的 11 个帧的持续时间。

```
var doc = fl.getDocumentDOM();
var my_tl = doc.getTimeline();
this.getCurrentFrame = function(){
var layer = my_tl.layers[my_tl.currentLayer];
var frame = layer.frames[my_tl.currentFrame];
return frame;
}
var theFrame = getCurrentFrame();
if(theFrame.isMotionObject()){
theFrame.setMotionObjectDuration(11);
}else{
fl.trace("It isn't motion");
}
```

## frame.setMotionObjectXML()

可用性

Flash Professional CS5。

用法

```
Frame.setMotionObjectXML( xmlstr [, endAtCurrentLocation] )
```

参数

**xmlstr** 一个字符串值，它指定 XML 字符串。

**endAtCurrentLocation** 一个布尔型值，它确定补间在当前位置是开始还是结束。

描述

方法；将指定的动画 XML 应用于选定的动画对象。

示例

此示例指定将标识为 myMotionXML 的动画 XML 应用于选定的动画对象。

```
var doc = fl.getDocumentDOM();
var my_tl = doc.getTimeline();
this.getCurrentFrame = function(){
var layer = my_tl.layers[my_tl.currentLayer];
var frame = layer.frames[my_tl.currentFrame];
return frame;
}
var theFrame = getCurrentFrame();
theFrame.setMotionObjectXML(myMotionXML.toString(), false);
```

## frame.shapeTweenBlend

可用性

Flash MX 2004。

用法

```
frame.shapeTweenBlend
```

描述

属性；一个字符串，它指定补间形状在补间开始的关键帧中的形状和下一个关键帧中的形状之间如何混合。可接受值为 "distributive" 和 "angular"。

## frame.soundEffect

可用性

Flash MX 2004。

用法

```
frame.soundEffect
```

**Frame 对象****描述**

属性；一个字符串，它指定直接附加在帧上的声音 ([frame.soundLibraryItem](#)) 的效果。可接受值为 "none"、"left channel"、"right channel"、"fade left to right"、"fade right to left"、"fade in"、"fade out" 和 "custom"。

**示例**

下面的示例指定附加于第一帧的声音应该淡入：

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].soundEffect = "fade in";
```

## frame.soundLibraryItem

**可用性**

Flash MX 2004。

**用法**

```
frame.soundLibraryItem
```

**描述**

属性；用于创建声音的库项目（请参阅 [SoundItem 对象](#)）。声音直接附加于帧中。

**示例**

下面的示例将库中的第一个项目分配给第一帧的 `soundLibraryItem` 属性：

```
// The first item in the library must be a sound object.  
fl.getDocumentDOM().getTimeline().layers[0].frames[0].soundLibraryItem  
=fl.getDocumentDOM().library.items[0];
```

## frame.soundLoop

**可用性**

Flash MX 2004。

**用法**

```
frame.soundLoop
```

**描述**

属性；一个整数值，它指定直接附加在帧上的声音 ([frame.soundLibraryItem](#)) 的播放次数。如果您希望为此属性指定一个值，请将 [frame.soundLoopMode](#) 设置为 "repeat"。

**示例**

请参阅 [frame.soundLoopMode](#)。

## frame.soundLoopMode

可用性

Flash MX 2004。

用法

`frame.soundLoopMode`

描述

属性；一个字符串，它指定直接附加在帧上的声音 ([frame.soundLibraryItem](#)) 是播放特定的次数还是无限循环播放。可接受值为 "repeat" 和 "loop"。若要指定声音的播放次数，请为 [frame.soundLoop](#) 设置一个值。

示例

下面的示例指定声音应播放两次：

```
f1.getDocumentDOM().getTimeline().layers[0].frames[0].soundLoopMode = "repeat";  
f1.getDocumentDOM().getTimeline().layers[0].frames[0].soundLoop = 2;
```

## frame.soundName

可用性

Flash MX 2004。

用法

`frame.soundName`

描述

属性；一个字符串，它指定直接附加在帧上的声音 ([frame.soundLibraryItem](#)) 的名称（存储在库中的名称）。

示例

下面的示例将第一帧的 `soundName` 属性更改为 "song1.mp3"；`song1.mp3` 必须存在于库中：

```
f1.getDocumentDOM().getTimeline().layers[0].frames[0].soundName = "song1.mp3";
```

## frame.soundSync

可用性

Flash MX 2004。

用法

`frame.soundSync`

描述

属性；一个字符串，它指定直接附加在帧上的声音 ([frame.soundLibraryItem](#)) 的同步行为。可接受值为 "event"、"stop"、"start" 和 "stream"。

## 示例

下面的示例指定声音应为声音流:

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].soundSync = 'stream';
```

## frame.startFrame

## 可用性

Flash MX 2004。

## 用法

```
frame.startFrame
```

## 描述

只读属性; 序列中第一帧的索引。

## 示例

在下面的示例中, `stFrame` 是帧序列中第一帧的索引。在此示例中, 帧序列跨越从第 5 帧到第 10 帧的 6 个帧。因此, 第 5 帧到第 10 帧中任何一帧的 `stFrame` 值均为 4 (请注意帧的索引值和帧的编号值是不同的)。

```
var stFrame = fl.getDocumentDOM().getTimeline().layers[0].frames[4].startFrame;
fl.trace(stFrame); // 4
var stFrame = fl.getDocumentDOM().getTimeline().layers[0].frames[9].startFrame;
fl.trace(stFrame); // 4
```

## frame.tweenEasing

## 可用性

Flash MX 2004。

## 用法

```
frame.tweenEasing
```

## 描述

属性; 一个整数, 它指定应用于补间对象的缓动数量。可接受值介于 -100 和 100 之间。要使补间动画缓慢开始并不断加快补间的速度直到动画结束, 请使用介于 -1 和 -100 之间的值。要使补间动画快速开始并不断降低补间的速度直到动画结束, 请使用介于 1 和 100 之间的某个正值。

## 示例

下面的示例指定补间对象的动作以相当快的速度开始, 并不断减速直到动画结束:

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].tweenEasing = 50;
```

## frame.tweenInstanceName

可用性

Flash Professional CS5。

用法

```
Frame.tweenInstanceName()
```

描述

属性；一个字符串，它为选定的动画对象分配实例名称。

示例

下面的示例将实例名称 `MyMotionTween` 分配给指定的动画对象。

```
theFrame.tweenInstanceName = "MyMotionTween";
```

## frame.tweenType

可用性

Flash MX 2004。

用法

```
frame.tweenType
```

描述

属性；一个字符串，它指定补间的类型；可接受值为 `"motion"`、`"shape"` 或 `"none"`。指定 `"none"` 值将删除补间动画。使用 [timeline.createMotionTween\(\)](#) 方法创建一个补间动画。

如果指定 `"motion"` 值，帧中的对象必须为元件、文本字段或组合对象。该对象将从它在当前关键帧中的位置补间至下一关键帧中的位置。

如果指定 `"shape"`，帧中的对象必须为形状对象。该对象将从当前关键帧中的形状开始，混合成下一关键帧中的形状。

示例

下面的示例指定一个对象为补间动画，所以它应从当前关键帧中的位置补间至后续关键帧中的位置：

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].tweenType = "motion";
```

## frame.useSingleEaseCurve

可用性

Flash 8。

用法

```
frame.useSingleEaseCurve
```

**Frame 对象****描述**

属性；一个布尔值。如果为 **true**，则将同一个自定义缓动曲线用作所有属性的缓动信息。如果为 **false**，则每个属性都将拥有自己的缓动曲线。

如果帧没有应用自定义缓动，则忽略此属性。

**示例**

下面的示例指定将同一个自定义缓入缓出曲线用于第一图层中第一帧的所有属性：

```
var theFrame = fl.getDocumentDOM().getTimeline().layers[0].frames[0]
theFrame.useSingleEaseCurve = true;
```

另请参见

[frame.getCustomEase\(\)](#)、[frame.hasCustomEase](#)、[frame.setCustomEase\(\)](#)

## 第 22 章 : HalfEdge 对象

可用性

Flash MX 2004。

描述

**HalfEdge** 对象是 **Shape** 对象的边缘的有向侧。一个边缘有两个半边缘。围绕这些半边缘“行走”可以横跨形状的轮廓。例如，从一个半边缘开始，可以跟踪围绕形状轮廓的所有两个半边缘，然后返回最初的半边缘。

半边缘是有序的。一个半边缘表示边缘的一侧，另一个半边缘表示另一侧。

方法摘要

**HalfEdge** 对象具有以下方法：

方法	描述
<a href="#">halfEdge.getEdge()</a>	获取 <b>HalfEdge</b> 对象的 <b>Edge</b> 对象。
<a href="#">halfEdge.getNext()</a>	获取当前轮廓上的下一个半边缘。
<a href="#">halfEdge.getOppositeHalfEdge()</a>	获取边缘另一侧的 <b>HalfEdge</b> 对象。
<a href="#">halfEdge.getPrev()</a>	获取当前轮廓上的上一个 <b>HalfEdge</b> 对象。
<a href="#">halfEdge.getVertex()</a>	在 <b>HalfEdge</b> 对象的顶部获取 <b>Vertex</b> 对象。

属性摘要

**HalfEdge** 对象具有以下属性：

属性	描述
<a href="#">halfEdge.id</a>	只读； <b>HalfEdge</b> 对象的唯一整数标识符。
<a href="#">halfEdge.index</a>	值为 0 或 1 的整数，指定此 <b>HalfEdge</b> 对象在父边缘中的索引。

### halfEdge.getEdge()

可用性

Flash MX 2004。

用法

```
halfEdge.getEdge()
```

参数

无。

返回

**Edge** 对象。



说明

方法；获取 HalfEdge 对象的 Edge 对象。请参阅 [Edge 对象](#)。

示例

下面的示例展示如何获取指定形状的边缘和半边缘：

```
var shape = fl.getDocumentDOM().selection[0];  
var hEdge = shape.edges[0].getHalfEdge(0);  
var edge = hEdge.getEdge();
```

## halfEdge.getNext()

可用性

Flash MX 2004。

用法

```
halfEdge.getNext()
```

参数

无。

返回

一个 HalfEdge 对象。

说明

方法；获取当前轮廓的下一个半边缘。

注：虽然半边缘具有方向和序列顺序，边缘却没有。

示例

下面的示例在 nextHalfEdge 变量中存储指定轮廓的下一个半边缘：

```
var shape = fl.getDocumentDOM().selection[0];  
var hEdge = shape.edges[0].getHalfEdge( 0 );  
var nextHalfEdge = hEdge.getNext();
```

## halfEdge.getOppositeHalfEdge()

可用性

Flash MX 2004。

用法

```
halfEdge.getOppositeHalfEdge()
```

参数

无。

返回

一个 HalfEdge 对象。

说明

方法；获取边缘另一侧的 HalfEdge 对象。

示例

下面的示例将与 hEdge 相对的半边缘存储在 otherHalfEdge 变量中：

```
var shape = fl.getDocumentDOM().selection[0];  
var hEdge = shape.edges[0].getHalfEdge(0);  
var otherHalfEdge = hEdge.getOppositeHalfEdge();
```

## halfEdge.getPrev()

可用性

Flash MX 2004。

用法

```
halfEdge.getPrev()
```

参数

无。

返回

一个 HalfEdge 对象。

说明

方法；获取当前轮廓上的上一个 HalfEdge 对象。

注：虽然半边缘具有方向和序列顺序，边缘却没有。

示例

下面的示例在 prevHalfEdge 变量中存储指定轮廓的上一个半边缘：

```
var shape = fl.getDocumentDOM().selection[0];  
var hEdge = shape.edges[0].getHalfEdge( 0 );  
var prevHalfEdge = hEdge.getPrev();
```

## halfEdge.getVertex()

可用性

Flash MX 2004。

用法

```
halfEdge.getVertex()
```

参数  
无。

返回  
[Vertex 对象](#)

说明  
方法；获取 HalfEdge 对象顶部的 Vertex 对象。请参阅 [Vertex 对象](#)

示例  
下面的示例在 vertex 变量中存储 hEdge 顶部的 Vertex 对象：

```
var shape = fl.getDocumentDOM().selection[0];  
var edge = shape.edges[0];  
var hEdge = edge.getHalfEdge(0);  
var vertex = hEdge.getVertex();
```

## halfEdge.id

可用性  
Flash MX 2004。

用法  
halfEdge.id

描述  
只读属性； HalfEdge 对象的唯一整数标识符。

示例  
下面的示例在“输出”面板中显示指定半边缘的唯一标识符：

```
var shape = fl.getDocumentDOM().selection[0];  
alert(shape.contours[0].getHalfEdge().id);
```

## halfEdge.index

可用性  
Flash MX 2004。

用法  
halfEdge.index

描述  
只读属性；值为 0 或 1 的整数，指定此 HalfEdge 对象在父边缘中的索引。

#### 示例

下面的示例显示“输出”面板中指定的半边缘的索引值：

```
var shape = fl.getDocumentDOM().selection[0];  
var hEdge = shape.edges[0].getHalfEdge(0);  
var heIndex = hEdge.index;
```

## 第 23 章 : Instance 对象

继承关系 [Element 对象](#) > Instance 对象

可用性

Flash MX 2004。

描述

Instance 是 [Element 对象](#) 的子类。

属性摘要

除 [Element](#) 对象的所有属性外，Instance 还具有下列属性：

属性	描述
<a href="#">instance.instanceType</a>	只读；一个字符串，它表示实例的类型。
<a href="#">instance.libraryItem</a>	用于实例化此实例的库项目。

### instance.instanceType

可用性

Flash MX 2004; Flash 8 中新增的 video 的可能值。

用法

```
instance.instanceType
```

描述

只读属性；一个字符串，它表示实例的类型。可能值为 symbol、bitmap、embedded video、linked video、video 和 compiled clip。

在 Flash MX 2004 中，使用 `library.addItem("video")` 添加到库中的项的 `instance.instanceType` 值为 `embedded_video`。在 Flash 8 及更高版本中，该值为 `video`。请参阅 [library.addItem\(\)](#)。

示例

下面的示例显示影片剪辑的实例类型为 `symbol`：

```
// Select a movie clip and then run this script.
var type = fl.getDocumentDOM().selection[0].instanceType;
fl.trace("This instance type is " + type);
```

### instance.libraryItem

可用性

Flash MX 2004。

**Instance 对象**

## 用法

```
instance.libraryItem
```

## 描述

属性；一个库项目，它用于实例化此实例。此属性只能更改为相同类型的另一个库项目（即不能将 **symbol** 实例设置为对位图的引用）。请参阅 [library 对象](#)。

## 示例

下面的示例将所选元件更改为引用库中的第一个项目：

```
fl.getDocumentDOM().selection[0].libraryItem = fl.getDocumentDOM().library.items[0];
```

## 第 24 章 : Item 对象

可用性

Flash MX 2004。

描述

**Item** 对象是一种抽象基类。库中的所有内容都派生自 **Item**。请参阅 [library 对象](#)。

方法摘要

**Item** 对象具有以下方法：

方法	描述
<a href="#">item.addData()</a>	将指定数据添加到一个库项目中。
<a href="#">item.getData()</a>	检索指定数据的值。
<a href="#">item.hasData()</a>	确定库项目是否具有命名数据。
<a href="#">item.removeData()</a>	从库项目中删除永久数据。

属性摘要

**Item** 对象具有以下属性：

属性	描述
<a href="#">item.itemType</a>	只读；一个字符串，它指定元素的类型。
<a href="#">item.linkageBaseClass</a>	指定将与元件相关联的 <b>ActionScript 3.0</b> 类的字符串。
<a href="#">item.linkageClassName</a>	指定将与元件相关联的 <b>ActionScript 2.0</b> 类的字符串。
<a href="#">item.linkageExportForAS</a>	一个布尔值。如果为 <b>true</b> ，则为 <b>ActionScript</b> 导出该项目。
<a href="#">item.linkageExportForRS</a>	一个布尔值。如果为 <b>true</b> ，则为运行时共享导出该项目。
<a href="#">item.linkageExportInFirstFrame</a>	一个布尔值。如果为 <b>true</b> ，则在第一帧导出该项目。
<a href="#">item.linkageIdentifier</a>	一个字符串，它指定在链接到目标 SWF 文件时 <b>Flash</b> 用于标识资源的名称。
<a href="#">item.linkageImportForRS</a>	一个布尔值。如果为 <b>true</b> ，则为运行时共享导入该项目。
<a href="#">item.linkageURL</a>	一个字符串，它指定包含共享资源的 SWF 文件所在的 URL。
<a href="#">item.name</a>	一个字符串，它指定包含文件夹结构的库项目的名称。

### item.addData()

可用性

Flash MX 2004。

用法

```
item.addData(name, type, data)
```

**Item 对象****参数**

**name** 一个字符串，它指定数据的名称。

**type** 一个字符串，它指定数据的类型。有效值为 "integer"、"integerArray"、"double"、"doubleArray"、"string" 和 "byteArray"。

**data** 要添加到指定库项目的数据。数据的类型取决于类型参数的值。例如，如果类型为 "integer"，则数据的值必须是一个整数，依此类推。

**返回**

无。

**描述**

方法；将指定数据添加一个库项目中。

**示例**

下面的示例将具有整数值 12 的名为 **myData** 的数据添加到库中的第一个项目：

```
fl.getDocumentDOM().library.items[0].addData("myData", "integer", 12);
```

## item.getData()

**可用性**

Flash MX 2004。

**用法**

```
item.getData(name)
```

**参数**

**name** 一个字符串，它指定要检索的数据的名称。

**返回**

由 **name** 参数指定的数据。返回的数据的类型取决于存储的数据的类型。

**说明**

方法；检索指定数据的值。

**示例**

下面的示例从库中第一个项目获取名为 **myData** 的数据的值，并将其存储在变量 **libData** 中：

```
var libData = fl.getDocumentDOM().library.items[0].getData("myData");
```

## item.hasData()

**可用性**

Flash MX 2004。

**用法**

```
item.hasData(name)
```



**Item 对象**

## 参数

**name** 一个字符串，它指定要在库项目中检查的数据的名称。

## 返回

布尔值：如果指定的数据存在，则为 **true**；否则为 **false**。

## 说明

方法；确定库项目是否具有命名数据。

## 示例

如果库中第一个项目包含名为 **myData** 的数据，则下面的示例会在“输出”面板中显示一条消息：

```
if (fl.getDocumentDOM().library.items[0].hasData("myData")){  
    fl.trace("Yep, it's there!");  
}
```

## item.itemType

## 可用性

Flash MX 2004。

## 用法

`item.itemType`

## 描述

只读属性；一个字符串，它指定元素的类型。其值是下列值之一：“undefined”、“component”、“movie clip”、“graphic”、“button”、“folder”、“font”、“sound”、“bitmap”、“compiled clip”、“screen”或“video”。如果此属性为“video”，则可以确定视频类型；请参阅 [videoItem.videoType](#)。

## 示例

下面的示例在“输出”面板中显示指定库项目的类型：

```
fl.trace(fl.getDocumentDOM().library.items[0].itemType);
```

## item.linkageBaseClass

## 可用性

Flash CS3 Professional。

## 用法

`item.linkageBaseClass`

## 描述

属性；一个字符串，它指定将与元件关联的 **ActionScript 3.0** 类。此处指定的值将显示在创作环境的“链接”对话框以及其他包括“链接”对话框控件的对话框（例如元件的“属性”对话框）中。（若要为 **ActionScript 2.0** 类指定此值，请使用 [item.linkageClassName](#)。）

如果基类是元件类型的默认值（例如，对于影片剪辑为“flash.display.MovieClip”，对于按钮为“flash.display.SimpleButton”等），则此属性为空字符串(“”)。同样，若要使某个项成为默认的基类，请将该值设置为空字符串。

设置该值时，“链接”对话框不会执行任何既定的检查，因此如果 Flash 无法将基类设置为指定值，将不会引发任何错误。例如，在“链接”对话框中设置此值会强制进行检查，以确保可以在 FLA 文件的类路径中找到基类。这确保了在“发布设置”对话框的“Flash”选项卡中可以选择 ActionScript 3.0，依此类推。在脚本中设置该属性时，不会执行这些检查。

#### 示例

下面几行代码展示了使用该属性的几种方式：

```
// sets the library item base class to "Sprite"
fl.getDocumentDOM().library.items[0].linkageBaseClass = "flash.display.Sprite";
// sets the library item base class to the default for that item type
fl.getDocumentDOM().library.items[0].linkageBaseClass = "";
// finds and displays the library item's base class
fl.trace(fl.getDocumentDOM().library.items[0].linkageBaseClass);
```

另请参见

[document.docClass](#)

## item.linkageClassName

#### 可用性

Flash MX 2004。

#### 用法

```
item.linkageClassName
```

#### 描述

属性；一个字符串，它指定将与元件关联的 ActionScript 2.0 类。（若要为 ActionScript 3.0 类指定此值，请使用 [item.linkageBaseClass](#)。）

若要定义此属性，[item.linkageExportForAS](#) 和 / 或 [item.linkageExportForRS](#) 属性必须设置为 true，并且 [item.linkageImportForRS](#) 属性必须设置为 false。

#### 示例

下面的示例指定与库中第一个项目关联的 ActionScript 2.0 类的名称为 myClass：

```
fl.getDocumentDOM().library.items[0].linkageClassName = "myClass";
```

## item.linkageExportForAS

#### 可用性

Flash MX 2004。

#### 用法

```
item.linkageExportForAS
```

## Item 对象

## 描述

属性；一个布尔值。如果此属性为 **true**，则为 **ActionScript** 导出该项。也可以将 [item.linkageExportForRS](#) 和 [item.linkageExportInFirstFrame](#) 属性设置为 **true**。

如果将此属性设置为 **true**，则必须将 [item.linkageImportForRS](#) 属性设置为 **false**。此外，还必须指定标识符 ([item.linkageIdentifier](#)) 和 URL ([item.linkageURL](#))。

## 示例

下面的示例将为指定的库项目设置此属性：

```
fl.getDocumentDOM().library.items[0].linkageExportForAS = true;
```

## item.linkageExportForRS

## 可用性

Flash MX 2004。

## 用法

```
item.linkageExportForRS
```

## 描述

属性；一个布尔值。如果此属性为 **true**，则为运行时共享导出该项目。也可以将 [item.linkageExportForAS](#) 和 [item.linkageExportInFirstFrame](#) 属性设置为 **true**。

如果将此属性设置为 **true**，则必须将 [item.linkageImportForRS](#) 属性设置为 **false**。此外，还必须指定标识符 ([item.linkageIdentifier](#)) 和 URL ([item.linkageURL](#))。

## 示例

下面的示例将为指定的库项目设置此属性：

```
fl.getDocumentDOM().library.items[0].linkageExportForRS = true;
```

## item.linkageExportInFirstFrame

## 可用性

Flash MX 2004。

## 用法

```
item.linkageExportInFirstFrame
```

## 描述

属性；一个布尔值。如果为 **true**，则项目在第一帧中导出；如果为 **false**，则项目在第一个实例的帧中导出。如果该项目没有出现在舞台上，则表明其未被导出。

仅当 [item.linkageExportForAS](#) 和 / 或 [item.linkageExportForRS](#) 设置为 **true** 时，此属性才能设置为 **true**。

## 示例

下面的示例指定在第一帧导出指定的库项目：

```
fl.getDocumentDOM().library.items[0].linkageExportInFirstFrame = true;
```

## item.linkageIdentifier

可用性

Flash MX 2004。

用法

```
item.linkageIdentifier
```

描述

属性；一个字符串，它指定 Flash 链接目标 SWF 文件时用于标识资源的名称。如果 [item.linkageImportForRS](#)、[item.linkageExportForAS](#) 和 [item.linkageExportForRS](#) 设置为 `false`，则 Flash 忽略此属性。相反，当上述任何属性设置为 `true` 时，都必须设置此属性。

示例

下面的示例指定当字符串 `my_mc` 链接其导出目标 SWF 文件时，将用于标识库项目：

```
fl.getDocumentDOM().library.items[0].linkageIdentifier = "my_mc";
```

另请参见

[item.linkageURL](#)

## item.linkageImportForRS

可用性

Flash MX 2004。

用法

```
item.linkageImportForRS
```

描述

属性；一个布尔值：如果为 `true`，则该项目是为运行时共享而导入的。如果此属性设置为 `true`，则 [item.linkageExportForAS](#) 和 [item.linkageExportForRS](#) 都必须设置为 `false`。此外，还必须指定标识符 ([item.linkageIdentifier](#)) 和 URL ([item.linkageURL](#))。

示例

下面的示例将库项目的这一属性设置为 `true`：

```
fl.getDocumentDOM().library.items[0].linkageImportForRS = true;
```

## item.linkageURL

可用性

Flash MX 2004。

**Item 对象**

## 用法

`item.linkageURL`

## 描述

属性；一个字符串，它指定包含共享资源的 SWF 文件所在的 URL。如果 `item.linkageImportForRS`、`item.linkageExportForAS` 和 `item.linkageExportForRS` 设置为 `false`，则 Flash 忽略此属性。相反，当上述任何属性设置为 `true` 时，都必须设置此属性。可采用与平台相关的格式（即根据平台选择正斜杠 [/] 或反斜杠 [\]）来指定 Web URL 或文件名。

## 示例

下面的示例指定了指定库项目的链接 URL：

```
fl.getDocumentDOM().library.items[0].linkageURL = "theShareSWF.swf";
```

## 另请参见

[item.linkageIdentifier](#)

## item.name

## 可用性

Flash MX 2004。

## 用法

`item.name`

## 描述

方法；一个字符串，它指定提供文件夹结构的库项目的名称。例如，如果 `Symbol_1` 在名为 `Folder_1` 的文件夹内，`Symbol_1` 的 `name` 属性则为 `"Folder_1/Symbol_1"`。

## 示例

下面的示例在“输出”面板中显示指定库项目的名称：

```
fl.trace(fl.getDocumentDOM().library.items[0].name);
```

## item.removeData()

## 可用性

Flash MX 2004。

## 用法

`item.removeData(name)`

## 参数

**name** 指定要从库项目中删除的数据的名称。

## 返回

无。

#### 描述

属性；从库项目中删除永久数据。

#### 示例

下面的示例从库中第一个项目删除名为 **myData** 的数据：

```
fl.getDocumentDOM().library.items[0].removeData("myData");
```

## 第 25 章 : Layer 对象

可用性

Flash MX 2004。

描述

Layer 对象表示时间轴中的图层。 `timeline.layers` 属性包含 Layer 对象的数组， `fl.getDocumentDOM().getTimeline().layers` 可以访问这些对象。

属性摘要

Layer 对象具有以下属性：

属性	描述
<code>layer.color</code>	一个字符串、十六进制值或整数，它指定用于显示图层轮廓的颜色。
<code>layer.frameCount</code>	只读；一个整数，它指定图层中的帧数。
<code>layer.frames</code>	只读；Frame 对象的数组。
<code>layer.height</code>	一个整数，它指定百分比图层高度；等效于“图层属性”对话框中的“图层”高度值。
<code>layer.layerType</code>	一个字符串，它指定当前使用的图层；等效于“图层属性”对话框中的“类型”设置。
<code>layer.locked</code>	一个布尔值，它指定图层的锁定状态。
<code>layer.name</code>	一个字符串，它指定图层的名称。
<code>layer.outline</code>	一个布尔值，它指定图层中所有对象的轮廓的状态。
<code>layer.parentLayer</code>	一个 Layer 对象，它表示包含此图层的文件夹、引导层或遮罩层。
<code>layer.visible</code>	一个布尔值，它指定舞台上的图层的对象是显示的还是隐藏的。

### layer.color

可用性

Flash MX 2004。

用法

`layer.color`

描述

属性；所分配的用于显示图层轮廓的颜色，使用以下格式之一：

- 格式为 "#RRGGBB" 或 "#RRGGBBAA" 的字符串
- 格式为 0xRRGGBB 的十六进制数字
- 表示与十六进制数字等价的十进制整数

该属性等效于“图层属性”对话框中的“轮廓”颜色设置。

**Layer 对象**

## 示例

下面的示例在 `colorValue` 变量中存储第一图层的值：

```
var colorValue = fl.getDocumentDOM().getTimeline().layers[0].color;
```

下面的示例显示将第一图层的颜色设置为红色的三种方法：

```
fl.getDocumentDOM().getTimeline().layers[0].color=16711680;  
fl.getDocumentDOM().getTimeline().layers[0].color="#ff0000";  
fl.getDocumentDOM().getTimeline().layers[0].color=0xFF0000;
```

## layer.frameCount

## 可用性

Flash MX 2004。

## 用法

```
layer.frameCount
```

## 描述

只读属性；一个整数，它指定图层中的帧数。

## 示例

下面的示例在 `fcNum` 变量中存储第一图层的帧数：

```
var fcNum = fl.getDocumentDOM().getTimeline().layers[0].frameCount;
```

## layer.frames

## 可用性

Flash MX 2004。

## 用法

```
layer.frames
```

## 描述

只读属性；**Frame** 对象的数组（请参阅 [Frame 对象](#)）。

## 示例

下面的示例将变量 `frameArray` 设置为当前文档中帧的 **Frame** 对象的数组：

```
var frameArray = fl.getDocumentDOM().getTimeline().layers[0].frames;
```

若要确定帧是否为关键帧，可以检查 `frame.startFrame` 属性是否与数组索引匹配，如下面的示例所示：

```
var frameArray = fl.getDocumentDOM().getTimeline().layers[0].frames;  
var n = frameArray.length;  
for (i=0; i<n; i++) {  
    if (i==frameArray[i].startFrame) {  
        alert("Keyframe at: " + i);  
    }  
}
```



## layer.height

可用性

Flash MX 2004。

用法

`layer.height`

描述

属性；一个整数，它指定百分比图层高度；等效于“图层属性”对话框中的“图层”高度值。可接受的值表示默认高度的百分比：100、200 或 300。

示例

下面的示例存储第一图层的高度设置的百分比值：

```
var layerHeight = fl.getDocumentDOM().getTimeline().layers[0].height;
```

下面的示例将第一图层的高度设置为 300%：

```
fl.getDocumentDOM().getTimeline().layers[0].height = 300;
```

## layer.layerType

可用性

Flash MX 2004。

用法

`layer.layerType`

描述

属性；一个字符串，它指定当前使用的图层；等效于“图层属性”对话框中的“类型”设置。可接受值为“normal”、“guide”、“guided”、“mask”、“masked”和“folder”。

示例

下面的示例将时间轴中的第一个图层设置为 **folder** 类型：

```
fl.getDocumentDOM().getTimeline().layers[0].layerType = "folder";
```

## layer.locked

可用性

Flash MX 2004。

用法

`layer.locked`

描述

属性；一个布尔值，它指定图层的锁定状态。如果设置为 **true**，则图层被锁定。默认值为 **false**。

**Layer 对象**

## 示例

下面的示例在 `lockStatus` 变量中存储第一图层的状态的布尔值:

```
var lockStatus = fl.getDocumentDOM().getTimeline().layers[0].locked;
```

下面的示例将第一图层的状态设置为未锁定:

```
fl.getDocumentDOM().getTimeline().layers[0].locked = false;
```

## layer.name

## 可用性

Flash MX 2004。

## 用法

```
layer.name
```

## 描述

属性; 一个字符串, 它指定图层的名称。

## 示例

下面的示例将当前文档中第一个图层的名称设置为 `foreground`:

```
fl.getDocumentDOM().getTimeline().layers[0].name = "foreground";
```

## layer.outline

## 可用性

Flash MX 2004。

## 用法

```
layer.outline
```

## 描述

属性; 一个布尔值, 它指定图层中所有对象的轮廓的状态。如果设置为 `true`, 则图层中的所有对象仅显示轮廓。如果为 `false`, 则对象在创建后会即刻显示。

## 示例

下面的示例使第一图层上的所有对象仅显示轮廓:

```
fl.getDocumentDOM().getTimeline().layers[0].outline = true;
```

## layer.parentLayer

## 可用性

Flash MX 2004。

**Layer 对象**

## 用法

```
layer.parentLayer
```

## 描述

属性；一个 **Layer** 对象，它表示包含此图层的文件夹、引导层或遮罩层。父图层必须在图层前面的文件夹、引导层或遮罩层，或者是前面或后面的图层的 **parentLayer**。设置图层的 **parentLayer** 不会移动图层在列表中的位置；如果尝试将图层的 **parentLayer** 设置为一个需要移动该图层的另一个图层，则不会有任何效果。顶级图层使用 **null**。

## 示例

下面的示例使用在同一时间轴上同一级的两个图层。将第一个图层 (**layers[0]**) 转换为文件夹，然后将其设置为第二个图层 (**layers[1]**) 的父文件夹。此动作将第二图层移动到第一图层内。

```
var parLayer = fl.getDocumentDOM().getTimeline().layers[0];
parLayer.layerType = "folder";
fl.getDocumentDOM().getTimeline().layers[1].parentLayer = parLayer;
```

## layer.visible

## 可用性

Flash MX 2004。

## 用法

```
layer.visible
```

## 描述

属性；一个布尔值，它指定舞台上的图层的对象是显示的还是隐藏的。如果设置为 **true**，则图层的所有对象都可见；如果为 **false**，则它们都是隐藏的。默认值为 **true**。

## 示例

下面的示例使第一图层中的所有对象都不可见：

```
fl.getDocumentDOM().getTimeline().layers[0].visible = false;
```

## 第 26 章 : library 对象

可用性

Flash MX 2004。

描述

**library** 对象表示“库”面板。它是 **Document** 对象的属性（请参阅 [document.library](#)），可以使用 `fl.getDocumentDOM().library` 进行访问。

**library** 对象包含不同类型（包括元件、位图、声音和视频）的项目的数组。

方法摘要

**library** 对象具有以下方法：

方法	描述
<a href="#">library.addItemToDocument()</a>	将当前项目或指定的项目添加到舞台的指定位置。
<a href="#">library.addNewItem()</a>	创建“库”面板中指定类型的新项目并将新项目设置为当前选择的项目。
<a href="#">library.deleteItem()</a>	从“库”面板中删除当前项目或指定项目。
<a href="#">library.duplicateItem()</a>	复制当前选择或指定的项目。
<a href="#">library.editItem()</a>	在“编辑”模式下打开当前选择或指定的项目。
<a href="#">library.expandFolder()</a>	展开或折叠库中当前选择或指定的文件夹。
<a href="#">library.findItemIndex()</a>	返回库项目的索引值（从零开始）。
<a href="#">library.getItemProperty()</a>	获取所选项目的属性。
<a href="#">library.getItemType()</a>	获取当前由库路径选择或指定的对象的类型。
<a href="#">library.getSelectedItems()</a>	获取库中所有当前选择项目的数组。
<a href="#">library.importEmbeddedSWF()</a>	将 SWF 文件作为经过编译的剪辑导入库中。
<a href="#">library.itemExists()</a>	检查指定的项目是否存在于库中。
<a href="#">library.moveToFolder()</a>	将当前选择或指定的库项目移动到指定的文件夹。
<a href="#">library.newFolder()</a>	在当前选择的文件夹中，使用指定的名称创建一个新文件夹，如果没有提供 <code>folderName</code> 参数则使用默认名称（“untitled folder #”）。
<a href="#">library.renameItem()</a>	在“库”面板中重命名当前选择的库项目。
<a href="#">library.selectAll()</a>	选择或取消选择库中的所有项目。
<a href="#">library.selectItem()</a>	选择指定的库项目。
<a href="#">library.selectNone()</a>	取消选择所有库项目。
<a href="#">library.setItemProperty()</a>	设置选择的所有库项目（忽略文件夹）的属性。
<a href="#">library.updateItem()</a>	更新指定的项目。

**library** 对象的属性摘要

**library** 对象具有以下属性：

属性	描述
<a href="#">library.items</a>	库中项目对象的数组

## library.addItemToDocument()

可用性

Flash MX 2004。

用法

```
library.addItemToDocument(position [, namePath])
```

参数

**position** 一个点，它指定舞台上项目的中心的 x,y 位置。

**namePath** 一个字符串，它指定项的名称。如果项目在文件夹内，可以使用斜杠记号指定其名称和路径。如果未指定 **namePath**，则使用当前库选择。此参数是可选的。

返回

布尔值：如果项目成功添加到文档，则为 **true**；否则为 **false**。

说明

方法；将当前项目或指定的项目添加到舞台的指定位置。

示例

下面的示例将当前选择的项目添加到舞台的 (3, 60) 位置：

```
fl.getDocumentDOM().library.addItemToDocument({x:3, y:60});
```

下面的示例将位于库中 **folder1** 内的项目 **Symbol1** 添加到舞台的 (550, 485) 位置：

```
fl.getDocumentDOM().library.addItemToDocument({x:550.0, y:485.0}, "folder1/Symbol1");
```

## library.addItemToDocument()

可用性

Flash MX 2004。

用法

```
library.addItemToDocument(type [, namePath])
```

参数

**type** 一个字符串，它指定要创建的项目的类型。**type** 可接受的值只能是 "video"、"movie clip"、"button"、"graphic"、"bitmap"、"screen" 和 "folder"（所以，有些操作，比如向库添加声音，不能用此方法来完成）。指定文件夹路径与在调用此方法之前使用 [library.newFolder\(\)](#) 具有相同效果。

**namePath** 一个字符串，它指定要添加的项目的名称。如果项目在文件夹内，可以使用斜杠记号指定其名称和路径。此参数是可选的。

**library** 对象

## 返回

布尔值：如果成功创建项目，则为 **true**；否则为 **false**。

## 说明

方法；创建“库”面板中指定类型的新项目并将新项目设置为当前选择的项目。有关向库中导入项目（包括声音等项目）的详细信息，请参阅 [document.importFile\(\)](#)。

## 示例

下面的示例在名为 **folderTwo** 的新文件夹中创建名为 **start** 的新按钮项目：

```
fl.getDocumentDOM().library.addNewItem("button", "folderTwo/start");
```

## library.deleteItem()

## 可用性

Flash MX 2004。

## 用法

```
library.deleteItem([namePath])
```

## 参数

**namePath** 一个字符串，它指定要删除的项目的名称。如果项目在文件夹内，可以使用斜杠记号指定其名称和路径。如果传递一个文件夹名称，则删除该文件夹及其所有项目。如果未指定任何名称，**Flash** 则删除当前选择的一个或多个项目。要删除“库”面板中的所有项目，请在使用此方法前选择所有项目。此参数是可选的。

## 返回

布尔值：如果成功删除项目，则为 **true**；否则为 **false**。

## 说明

方法；从“库”面板中删除当前项目或指定的项目。如果选择了多个项目，此方法可以影响多个项目。

## 示例

下面的示例删除当前选择的项目：

```
fl.getDocumentDOM().library.deleteItem();
```

下面的示例从库文件夹 **Folder\_1** 中删除项 **Symbol\_1**：

```
fl.getDocumentDOM().library.deleteItem("Folder_1/Symbol_1");
```

## library.duplicateItem()

## 可用性

Flash MX 2004。

## 用法

```
library.duplicateItem([ namePath ] )
```

**library** 对象

## 参数

**namePath** 一个字符串，它指定要直接重制的项目的名称。如果项目在文件夹内，可以使用斜杠记号指定其名称和路径。此参数是可选的。

## 返回

布尔值：如果成功重制项目，则为 **true**；否则为 **false**。如果选择了多个项目，则 **Flash** 返回 **false**。

## 说明

方法；复制当前选择或指定的项目。新项目具有默认名称（如 **item copy**）且设置为当前选择的项目。如果选择了多个项目，则该命令会失败。

## 示例

下面的示例在库文件夹 **test** 中创建项目 **square** 的一个副本：

```
fl.getDocumentDOM().library.duplicateItem("test/square");
```

## library.editItem()

## 可用性

Flash MX 2004。

## 用法

```
library.editItem([namePath])
```

## 参数

**namePath** 一个字符串，它指定项的名称。如果项目在文件夹内，可以使用斜杠记号指定其名称和路径。如果未指定 **namePath**，则在“编辑”模式下打开单个所选库。如果当前在库中没有选择项或者选择了多个项，则会显示主时间轴中的第一个场景以进行编辑。此参数是可选的。

## 返回

布尔值：如果指定的项目存在并且可以编辑，则为 **true**；否则为 **false**。

## 说明

方法；在“编辑”模式下打开当前选择或指定的项目。

## 示例

下面的示例打开库中 **test** 文件夹内的项目 **circle** 供编辑：

```
fl.getDocumentDOM().library.editItem("test/circle");
```

## library.expandFolder()

## 可用性

Flash MX 2004。

## 用法

```
library.expandFolder(bExpand [, bRecurseNestedParents [, namePath]])
```

**library** 对象

## 参数

**bExpand** 一个布尔值：如果为 **true**，则文件夹是展开的；如果为 **false**（默认值），则文件夹是折叠的。

**bRecurseNestedParents** 一个布尔值：如果为 **true**，则根据 **bExpand** 的值展开或折叠指定文件夹内的所有文件夹。默认值为 **false**。此参数是可选的。

**namePath** 一个字符串，它指定要展开或折叠的文件夹的名称，以及路径（可选）。如果未指定此参数，此方法则应用于当前选择的文件夹。此参数是可选的。

## 返回

布尔值：如果成功展开或折叠项目，则为 **true**；如果不成功或指定的项目不是文件夹，则为 **false**。

## 说明

方法；展开或折叠库中当前选择或指定的文件夹。

## 示例

下面的示例折叠库中的 **test** 文件夹以及 **test** 文件夹内的所有文件夹（如果有）：

```
fl.getDocumentDOM().library.expandFolder(false, true, "test");
```

## library.findItemIndex()

## 可用性

Flash MX 2004。

## 用法

```
library.findItemIndex(namePath)
```

## 参数

**namePath** 一个字符串，它指定项的名称。如果项目在文件夹内，可以使用斜杠记号指定其名称和路径。

## 返回

一个整数值，它表示项目从零开始的索引值。

## 说明

方法；返回库项目的索引值（从零开始）。库索引是平构的，因此文件夹被视为主索引的一部分。文件夹路径可以用来指定嵌套项目。

## 示例

下面的示例在变量 **sqIndex** 中存储库的 **test** 文件夹中的项目 **square** 的从零开始的索引值，然后在对话框中显示该索引值：

```
var sqIndex = fl.getDocumentDOM().library.findItemIndex("test/square");  
alert(sqIndex);
```

## library.getItemProperty()

## 可用性

Flash MX 2004。



**library** 对象

## 用法

```
library.getItemProperty(property)
```

## 参数

**属性** 字符串。有关可用作 **property** 参数的值的列表，请参阅 [Item 对象](#) 的属性摘要表及其子类的属性摘要。

## 返回

属性的字符串值。

## 说明

方法；获取所选项目的属性。

## 示例

下面的示例在使用 **ActionScript** 引用元件或为运行时共享引用元件时，显示一个包含该元件的“链接标识符”值的对话框：

```
alert(fl.getDocumentDOM().library.getItemProperty("linkageIdentifier"));
```

## library.getItemType()

## 可用性

Flash MX 2004。

## 用法

```
library.getItemType([namePath])
```

## 参数

**namePath** 一个字符串，它指定项的名称。如果项目在文件夹内，可以使用斜杠记号指定其名称和路径。如果未指定 **namePath**，Flash 则提供当前选择的类型。如果当前选择了多个项目且没有提供 **namePath**，Flash 则忽略该命令。此参数是可选的。

## 返回

一个指定对象的类型的字符串。有关可能的返回值，请参阅 [item.itemType](#)。

## 说明

方法；获取当前由库路径选择或指定的对象的类型。

## 示例

下面的示例显示一个对话框，其中包含 **Folder\_1/Folder\_2** 文件夹中的 **Symbol\_1** 项目类型：

```
alert(fl.getDocumentDOM().library.getItemType("Folder_1/Folder_2/Symbol_1"));
```

## library.getSelectedItems()

## 可用性

Flash MX 2004。

**library** 对象

## 参数

无。

## 返回

库中所有当前选择项目的值的数组。

## 说明

方法；获取库中所有当前选择项目的数组。

## 示例

下面的示例将当前所选库项目（在本例中是多个音频文件）的数组存储在 `selItems` 变量中，然后将数组中第一个音频文件的 `sampleRate` 属性更改为 11 kHz：

```
var selItems = fl.getDocumentDOM().library.getSelectedItems();
selItems[0].sampleRate = "11 kHz";
```

## library.importEmbeddedSWF()

## 可用性

Flash MX 2004。

## 用法

```
library.importEmbeddedSWF(linkageName, swfData [, libName])
```

## 参数

**linkageName** 一个字符串，它提供根影片剪辑的 SWF 链接的名称。

**swfData** 一个二进制 SWF 数据的数组，它来自外部库或 DLL。

**libName** 一个字符串，它指定已创建的项目的库名称。如果该名称已被使用，此方法则创建另外一个名称。此参数是可选的。

## 返回

无。

## 描述

方法；将 SWF 文件导入库中作为编译剪辑。与“文件”>“导入”>“SWF”不同，此方法允许将编译的 SWF 文件嵌入库中。没有对应的用户界面功能，此方法必须与外部库或 DLL 一起使用（请参阅第 488 页的“C 级可扩展性”）。

要导入的 SWF 文件必须有一个包含全部内容的顶级影片剪辑。该影片剪辑的链接标识符应该设置为与传递给此方法的 **linkageName** 参数相同的值。

## 示例

下面的示例将 **linkageName** 值为 MyMovie 的 SWF 文件添加到库中，作为名为 Intro 的编译剪辑：

```
fl.getDocumentDOM().library.importEmbeddedSWF("MyMovie", swfData, "Intro");
```

## library.itemExists()

可用性

Flash MX 2004。

用法

```
library.itemExists(namePath)
```

参数

**namePath** 一个字符串，它指定项的名称。如果项目在文件夹内，可以使用斜杠记号指定其名称和路径。

返回

布尔值：如果指定的项目存在于库中，则为 **true**；否则为 **false**。

说明

方法；检查指定的项目是否存在于库中。

示例

下面的示例根据 **Folder\_1** 库文件夹中是否存在项目 **Symbol\_1**，在对话框中显示 **true** 或 **false**：

```
alert(fl.getDocumentDOM().library.itemExists('Folder_1/Symbol_1'));
```

## library.items

可用性

Flash MX 2004。

用法

```
library.items
```

描述

属性；库中项目对象的数组。

示例

下面的示例在 **itemArray** 变量中存储所有库项目的数组：

```
var itemArray = fl.getDocumentDOM().library.items;
```

## library.moveToFolder()

可用性

Flash MX 2004。

用法

```
library.moveToFolder(folderPath [, itemToMove [, bReplace]])
```

**library** 对象

## 参数

**folderPath** 一个字符串，它以 "FolderName" 或 "FolderName/FolderName" 的形式指定文件夹的路径。若要将项目移到顶级，可以将 **folderPath** 指定为空字符串 ("")。

**itemToMove** 一个字符串，它指定要移动的项目的名称。如果未指定 **itemToMove**，则移动当前选择的项目。此参数是可选的。

**bReplace** 一个布尔值。如果具有相同名称的项目已经存在，将 **bReplace** 参数指定为 **true** 则会使用移动的项目替换现有的项目。如果为 **false**，则删除的项目的名称会更改为一个唯一的名称。默认值为 **false**。此参数是可选的。

## 返回

布尔值：如果成功移动项目，则为 **true**；否则为 **false**。

## 说明

方法；将当前选择或指定的库项目移动到指定的文件夹。如果 **folderPath** 参数为空，则将项目移动到顶级。

## 示例

下面的示例将项目 **Symbol\_1** 移动到库文件夹 **new**，然后替换该文件夹中的同名项目：

```
fl.getDocumentDOM().library.moveToFolder("new", "Symbol_1", true);
```

## library.newFolder()

## 可用性

Flash MX 2004。

## 用法

```
library.newFolder([folderPath])
```

## 参数

**folderPath** 一个字符串，它指定要创建的文件夹的名称。如果将该参数指定为不存在的路径，则会创建该路径。此参数是可选的。

## 返回

布尔值：如果成功创建文件夹，则为 **true**；否则为 **false**。

## 说明

方法；在当前选择的文件夹中，使用指定的名称创建一个新文件夹；如果没有提供 **folderName** 参数，则使用默认名称 ("untitled folder #")。

## 示例

下面的示例创建两个新的库文件夹。第二个文件夹是第一个文件夹的子文件夹：

```
fl.getDocumentDOM().library.newFolder("first/second");
```

## library.renameItem()

可用性

Flash MX 2004。

用法

```
library.renameItem(name)
```

参数

**name** 一个字符串，它指定库项目的新名称。

返回

一个布尔值，如果项目的名称更改成功则为 **true**，否则为 **false**。如果选择了多个项，则不会更改任何名称，并且返回值为 **false**（为了与用户界面的行为相匹配）。

说明

方法；在“库”面板中重命名当前选择的库项目。

示例

下面的示例将当前所选库项目重命名为 **new name**：

```
fl.getDocumentDOM().library.renameItem("new name");
```

## library.selectAll()

可用性

Flash MX 2004。

用法

```
library.selectAll([bSelectAll])
```

参数

**bSelectAll** 一个布尔值，它指定是选择还是取消选择库中的所有项目。省略此参数或使用默认值 **true** 来选择库中的所有项目；如果为 **false**，则取消选择所有库项目。此参数是可选的。

返回

无。

描述

方法；选择或取消选择库中的所有项目。

示例

下面的示例选择库中的所有项：

```
fl.getDocumentDOM().library.selectAll();  
fl.getDocumentDOM().library.selectAll(true);
```

下面的示例取消选择库中的所有项：

library 对象

```
f1.getDocumentDOM().library.selectAll(false);  
f1.getDocumentDOM().library.selectNone();
```

## library.selectItem()

可用性

Flash MX 2004。

用法

```
library.selectItem(namePath [, bReplaceCurrentSelection [, bSelect]])
```

参数

**namePath** 一个字符串，它指定项的名称。如果项目在文件夹内，可以使用斜杠记号指定其名称和路径。

**bReplaceCurrentSelection** 一个布尔值，它指定是替换当前选择还是将项目添加到当前选择。默认值为 **true**（替换当前选择）。此参数是可选的。

**bSelect** 一个布尔值，它指定是选择还是取消选择某个项目。默认值为 **true**（选择）。此参数是可选的。

返回

布尔值：如果指定的项目存在，则为 **true**；否则为 **false**。

说明

方法；选择指定的库项目。

示例

下面的示例将库中当前所选项目更改为 **untitled Folder\_1** 内的 **Symbol\_1**：

```
f1.getDocumentDOM().library.selectItem("untitled Folder_1/Symbol_1");
```

下面的示例扩展库中当前所选项，以包含 **untitled Folder\_1** 内的 **Symbol\_1**：

```
f1.getDocumentDOM().library.selectItem("untitled Folder_1/Symbol_1", false);
```

下面的示例取消选择 **untitled Folder\_1** 内的 **Symbol\_1**，但不更改其他已选的项目：

```
f1.getDocumentDOM().library.selectItem("untitled Folder_1/Symbol_1", true, false);
```

## library.selectNone()

可用性

Flash MX 2004。

用法

```
library.selectNone()
```

参数

无。

返回  
无。

描述  
方法；取消选择所有库项目。

示例  
下面的示例取消选择库中的所有项：

```
fl.getDocumentDOM().library.selectNone();  
fl.getDocumentDOM().library.selectAll(false);
```

## library.setItemProperty()

可用性  
Flash MX 2004。

用法  
`library.setItemProperty(property, value)`

参数  
**属性** 一个字符串，它是要设置的属性的名称。有关属性的列表，请参阅 [Item 对象](#) 的属性摘要表及其子类的属性摘要。若要查看哪些对象是 **Item** 对象的子类，请参阅第 10 页的“[DOM 结构摘要](#)”。  
**value** 赋给指定属性的值。

返回  
无。

描述  
方法；设置选择的所有库项目（忽略文件夹）的属性。

示例  
下面的示例将值按钮赋给选择的一个或多个库项目的 `symbolType` 属性。在这种情况下，项目必须是 [SymbolItem 对象](#)；`symbolType` 是 [SymbolItem](#) 对象的有效属性。

```
fl.getDocumentDOM().library.setItemProperty("symbolType", "button");
```

## library.updateItem()

可用性  
Flash MX 2004。

用法  
`library.updateItem([namePath])`

#### 参数

**namePath** 一个字符串，它指定项的名称。如果项目在文件夹内，可以使用斜杠记号指定其名称和路径。这等效于在一项上右键单击并从用户界面中的菜单内选择“更新”。如果未提供任何名称，则更新当前选择。此参数是可选的。

#### 返回

布尔值：如果 **Flash** 成功更新项目，则为 **true**；否则为 **false**。

#### 说明

方法；更新指定的项目。

#### 示例

下面的示例显示一个对话框，它显示当前选择的项目是 (**true**) 否 (**false**) 已更新：

```
alert (fl.getDocumentDOM().library.updateItem());
```



## 第 27 章 : Math 对象

可用性

Flash MX 2004。

描述

**Math** 对象可用作 **flash** 对象的只读属性；请参阅 [fl.Math](#)。此对象提供执行常见数学运算的方法。

方法摘要

**Math** 对象具有以下方法：

方法	描述
<a href="#">Math.concatMatrix()</a>	执行矩阵级联并返回结果。
<a href="#">Math.invertMatrix()</a>	返回指定矩阵的逆矩阵。
<a href="#">Math.pointDistance()</a>	计算两点之间的距离。

### Math.concatMatrix()

可用性

Flash MX 2004。

用法

```
Math.concatMatrix(mat1, mat2)
```

参数

**mat1, mat2** 指定要连接的 **Matrix** 对象（请参阅 [Matrix 对象](#)）。每个参数都必须是具有 **a**、**b**、**c**、**d**、**tx** 和 **ty** 字段的对象。

返回

一个级联的对象矩阵。

说明

方法；执行矩阵级联并返回结果。

示例

下面的示例将当前选择的对象存储在 **elt** 变量中，用视图矩阵乘以该对象矩阵，然后将该值存储在变量 **mat** 中：

```
var elt = fl.getDocumentDOM().selection[0];
var mat = fl.Math.concatMatrix( elt.matrix , fl.getDocumentDOM().viewMatrix );
```

### Math.invertMatrix()

可用性

Flash MX 2004。

**Math** 对象

## 用法

```
Math.invertMatrix(mat)
```

## 参数

**mat** 指示要反转的 **Matrix** 对象（请参阅 [Matrix 对象](#)）。它必须具有下列字段：a、b、c、d、tx 和 ty。

## 返回

一个 **Matrix** 对象，即原始矩阵的逆矩阵。

## 说明

方法；返回指定矩阵的逆矩阵。

## 示例

下面的示例将当前选择的对象存储在 **elt** 变量中，将该矩阵赋给 **mat** 变量，然后将该矩阵的逆矩阵存储在变量 **inv** 中：

```
var elt = fl.getDocumentDOM().selection[0];  
var mat = elt.matrix;  
var inv = fl.Math.invertMatrix( mat );
```

## Math.pointDistance()

## 可用性

Flash MX 2004。

## 用法

```
Math.pointDistance(pt1, pt2)
```

## 参数

**pt1, pt2** 指定要测量其之间的距离的两个点。

## 返回

一个浮点数值，它表示点之间的距离。

## 说明

方法；计算两点之间的距离。

## 示例

下面的示例在 **dist** 变量中存储 **pt1** 和 **pt2** 之间的距离值：

```
var pt1 = {x:10, y:20}  
var pt2 = {x:100, y:200}  
var dist = fl.Math.pointDistance(pt1, pt2);
```

## 第 28 章 : Matrix 对象

可用性

Flash MX 2004。

描述

**Matrix** 对象表示一个变形矩阵。

属性摘要

**Matrix** 对象具有以下属性：

属性	描述
<a href="#">matrix.a</a>	一个浮点值，它指定变形矩阵中的 (0,0) 元素。
<a href="#">matrix.b</a>	一个浮点值，它指定矩阵中的 (0,1) 元素。
<a href="#">matrix.c</a>	一个浮点值，它指定矩阵中的 (1,0) 元素。
<a href="#">matrix.d</a>	一个浮点值，它指定矩阵中的 (1,1) 元素。
<a href="#">matrix.tx</a>	一个浮点值，它指定元件的注册点或形状中心点的 x 轴的位置。
<a href="#">matrix.ty</a>	一个浮点值，它指定元件的注册点或形状中心点的 y 轴的位置。

### matrix.a

可用性

Flash MX 2004。

用法

`matrix.a`

描述

属性；一个浮点值，它指定变形矩阵中的 (0,0) 元素。此值表示对象 x 轴的缩放系数。

示例

矩阵中的 **a** 和 **d** 属性表示缩放。在下面的示例中，这些值分别设置为 2 和 3，将所选对象的宽度放大两倍，高度放大三倍：

```
var mat = fl.getDocumentDOM().selection[0].matrix;
mat.a = 2;
mat.d = 3;
fl.getDocumentDOM().selection[0].matrix = mat;
```

通过设置彼此相关的矩阵属性 **a**、**b**、**c** 和 **d**（其中 **a** = **d** 且 **b** = -**c**），可以旋转对象。例如，值 0.5、0.8、-0.8 和 0.5 将对象旋转 60°：

**Matrix 对象**

```
var mat = fl.getDocumentDOM().selection[0].matrix;  
mat.a = 0.5;  
mat.b = 0.8;  
mat.c = 0.8*(-1);  
mat.d = 0.5;  
fl.getDocumentDOM().selection[0].matrix = mat;
```

设置  $a = d = 1$  且  $c = b = 0$  可以将对象重置为其原始形状。

## matrix.b

可用性

Flash MX 2004。

用法

`matrix.b`

描述

属性；一个浮点值，它指定矩阵中的 (0,1) 元素。此值表示形状的垂直倾斜；它使 Flash 沿垂直轴移动形状的右边缘。

矩阵中的 `matrix.b` 和 `matrix.c` 属性表示倾斜（请参阅 [matrix.c](#)）。

示例

在下面的示例中，可分别将 `b` 和 `c` 设置为 -1 和 0；这些设置将对象沿垂直方向倾斜 45°：

```
var mat = fl.getDocumentDOM().selection[0].matrix;  
mat.b = -1;  
mat.c = 0;  
fl.getDocumentDOM().selection[0].matrix = mat;
```

若要将对象倾斜为其原始形状，可以将 `b` 和 `c` 设置为 0。

请参阅 [matrix.a](#) 示例。

## matrix.c

可用性

Flash MX 2004。

用法

`matrix.c`

描述

属性；一个浮点值，它指定矩阵中的 (1,0) 元素。此值使 Flash 通过沿水平轴移动对象底边缘来倾斜对象。

矩阵中的 `matrix.b` 和 `matrix.c` 属性表示倾斜。

示例

请参阅 [matrix.b](#) 示例。

## matrix.d

可用性

Flash MX 2004。

用法

`matrix.d`

描述

属性；一个浮点值，它指定矩阵中的 (1,1) 元素。此值表示对象 y 轴的缩放系数。

示例

请参阅 [matrix.a](#) 示例。

## matrix.tx

可用性

Flash MX 2004。

用法

`matrix.tx`

描述

属性；一个浮点值，它指定元件的注册点（即“原点”或“零点”）或形状中心点的 x 轴的位置。它定义变形的 x 转换。

通过设置 `matrix.tx` 和 `matrix.ty` 属性，可以移动对象（请参阅 [matrix.ty](#)）。

示例

在下面的示例中，将 `tx` 和 `ty` 设置为 0 可以在文档中将对象的注册点移动到点 0,0。

```
var mat = fl.getDocumentDOM().selection[0].matrix;  
mat.tx = 0;  
mat.ty = 0;  
fl.getDocumentDOM().selection[0].matrix = mat;
```

## matrix.ty

可用性

Flash MX 2004。

**Matrix 对象**

## 用法

`matrix.ty`

## 描述

属性；一个浮点值，它指定元件的注册点或形状中心点的 **y** 轴的位置。它定义变形的 **y** 转换。

通过设置 `matrix.tx` 和 `matrix.ty` 属性，可以移动对象。

## 示例

请参阅 [matrix.tx](#) 示例。

## 第 29 章 : outputPanel 对象

### 可用性

Flash MX 2004。

### 描述

此对象表示“输出”面板，它显示疑难解答信息，如语法错误。使用 `fl.outputPanel`（或 `flash.outputPanel`）来访问此对象。请参阅 [fl.outputPanel](#)。

### 方法摘要

`outputPanel` 对象具有以下方法：

方法	描述
<a href="#">outputPanel.clear()</a>	清除“输出”面板的内容。
<a href="#">outputPanel.save()</a>	将“输出”面板中的内容保存到一个本地文本文件。
<a href="#">outputPanel.trace()</a>	在“输出”面板的内容中添加一行，以新行终止。

## outputPanel.clear()

### 可用性

Flash MX 2004。

### 用法

```
outputPanel.clear();
```

### 参数

无。

### 返回

无。

### 描述

方法；清除“输出”面板的内容。可以在批处理应用程序中使用此方法来清除错误列表，或通过将此方法与 [outputPanel.save\(\)](#) 一起使用来逐步保存这些错误。

### 示例

下面的示例清除“输出”面板的当前内容：

```
fl.outputPanel.clear();
```

## outputPanel.save()

可用性

Flash MX 2004；Flash 8 增加的 bUseSystemEncoding 参数。

用法

```
outputPanel.save(fileURI [, bAppendToFile [, bUseSystemEncoding]])
```

参数

**fileURI** 一个表示为 file:/// URI 的字符串，它指定包含“输出”面板内容的本地文件。

**bAppendToFile** 一个可选的布尔值。如果为 true，则它会将“输出”面板中的内容追加到输出文件中；如果为 false，则该方法会在输出文件存在时将其覆盖。默认值为 false。

**bUseSystemEncoding** 一个可选的布尔值。如果为 true，则使用系统编码来保存“输出”面板文本；如果为 false，则使用 UTF-8 编码来保存“输出”面板文本，并且文本的开头带字节顺序标记字符。默认值为 false。

返回

无。

描述

方法；将“输出”面板的内容保存到本地文本文件中，保存方式是覆盖该文件或追加到该文件中。

如果 fileURI 无效或未指定，则报告一个错误。

此方法可用于批处理。例如，您可以创建一个 JSFL 文件，其中编译了几个组件。所有编译错误都出现在“输出”面板中，而且您可以使用此方法将出现的错误保存到文本文件中，该文件可以由使用的生成系统自动分析。

示例

下面的示例将“输出”面板中的内容保存到 /tests 文件夹下的 batch.log 文件中，如果 batch.log 文件已存在则覆盖该文件：

```
fl.outputPanel.save("file:///c:/tests/batch.log");
```

## outputPanel.trace()

可用性

Flash MX 2004。

用法

```
outputPanel.trace(message)
```

参数

**message** 一个字符串，它包含要添加到“输出”面板的文本。

返回

无。

描述

方法；将一个以新行终止的文本字符串发送到“输出”面板；如果“输出”面板尚不可见，则显示它。此方法与 fl.trace() 相同，并与 ActionScript 中的 trace() 语句的工作方式相同。



要发送一个空行，请使用 `outputPanel.trace("")` 或 `outputPanel.trace("\n")`。可使用后一种内联命令，将 `\n` 作为 `message` 字符串的一部分。

#### 示例

下面的示例在“输出”面板中显示几行文本：

```
fl.outputPanel.clear();
fl.outputPanel.trace("Hello World!!!");
var myPet = "cat";
fl.outputPanel.trace("\nI have a " + myPet);
fl.outputPanel.trace("");
fl.outputPanel.trace("I love my " + myPet);
fl.outputPanel.trace("Do you have a " + myPet + "?");
```

# 第 30 章 : Oval 对象

继承关系 [Element 对象](#) > [Shape 对象](#) > [Oval 对象](#)

可用性

Flash CS3 Professional。

描述

[Oval](#) 对象是使用“基本椭圆”工具绘制的形状。若要确定一个项目是否为 [Oval](#) 对象，请使用 [shape.isOvalObject](#)。

属性摘要

除 [Shape 对象](#) 的属性外，还可以对 [Oval](#) 对象使用下列属性。若要设置 [Oval](#) 对象的属性，请使用 [document.setOvalObjectProperty\(\)](#)。

属性	描述
<a href="#">OvalObject.closePath</a>	只读；一个布尔值，它指定是否已选中“属性”检查器中的“闭合路径”复选框。
<a href="#">OvalObject.endAngle</a>	只读；一个浮点值，它指定 <a href="#">Oval</a> 对象的结束角度。
<a href="#">OvalObject.innerRadius</a>	只读；一个浮点值，它以百分比的形式指定 <a href="#">Oval</a> 对象的内径。
<a href="#">OvalObject.startAngle</a>	只读；一个浮点值，它指定 <a href="#">Oval</a> 对象的开始角度。

## OvalObject.closePath

可用性

Flash CS3 Professional。

用法

`OvalObject.closePath`

描述

只读属性；一个布尔值，它指定是否已选中“属性”检查器中的“闭合路径”复选框。如果对象的开始角度和结束角度值相同，则在将两个值设置为不同之前设置此属性没有效果。

若要设置此值，请使用 [document.setOvalObjectProperty\(\)](#)。

示例

下面的示例将取消选中 `OvalObject.closePath` 属性：

```
fl.getDocumentDOM().setOvalObjectProperty("closePath", false);
```

另请参见

[document.setOvalObjectProperty\(\)](#)、[shape.isOvalObject](#)

## OvalObject.endAngle

可用性

Flash CS3 Professional。

用法

`OvalObject.endAngle`

描述

只读属性；一个浮点值，它指定 **Oval** 对象的结束角度。可接受值为从 0 到 360 的值。

若要设置此值，请使用 [document.setOvalObjectProperty\(\)](#)。

示例

下面的示例将所选 **Oval** 对象的结束角度设置为 270。

```
fl.getDocumentDOM().setOvalObjectProperty("endAngle",270);
```

另请参见

[document.setOvalObjectProperty\(\)](#)、[OvalObject.startAngle](#)、[shape.isOvalObject](#)

## OvalObject.innerRadius

可用性

Flash CS3 Professional。

用法

`OvalObject.innerRadius`

描述

只读属性；一个浮点值，它以百分比的形式指定 **Oval** 对象的内径。可接受值为从 0 到 99 的值。

若要设置此值，请使用 [document.setOvalObjectProperty\(\)](#)。

示例

下面的示例将所选 **Oval** 对象的内径设置为 50%：

```
fl.getDocumentDOM().setOvalObjectProperty("innerRadius",50);
```

另请参见

[document.setOvalObjectProperty\(\)](#)、[shape.isOvalObject](#)

## OvalObject.startAngle

可用性

Flash CS3 Professional。

**Oval 对象**

## 用法

`OvalObject.startAngle`

## 描述

只读属性；一个浮点值，它指定 **Oval** 对象的开始角度。可接受值为从 0 到 360 的值。

若要设置此值，请使用 [document.setOvalObjectProperty\(\)](#)。

## 示例

下面的示例将所选 **Oval** 对象的开始角度设置为 270：

```
fl.getDocumentDOM().setOvalObjectProperty("startAngle",270);
```

另请参见

[document.setOvalObjectProperty\(\)](#)、[OvalObject.endAngle](#)、[shape.isOvalObject](#)

## 第 31 章 : Parameter 对象

可用性

Flash MX 2004。

描述

Parameter 对象类型是从 `componentInstance.parameters` 数组（对应于创作工具中的组件“属性”检查器）访问的。

方法摘要

Parameter 对象具有以下方法：

方法	描述
<code>parameter.insertItem()</code>	将一个项目插入列表、对象或数组。
<code>parameter.removeItem()</code>	删除屏幕或组件参数的列表、对象或数组类型的一个元素。

属性摘要

Parameter 对象具有以下属性：

属性	描述
<code>parameter.category</code>	一个字符串，它为 <code>screen</code> 参数或 <code>componentInstance</code> 参数指定 <code>category</code> 属性。
<code>parameter.listIndex</code>	一个整数，它指定选择的列表项目的值。
<code>parameter.name</code>	只读；一个字符串，它指定参数的名称。
<code>parameter.value</code>	与“组件”检查器的“参数”选项卡、“属性”检查器的“参数”选项卡或屏幕“属性”检查器中的“值”字段相对应。
<code>parameter.valueType</code>	只读；一个字符串，它指示屏幕或组件参数的类型。
<code>parameter.verbose</code>	指定参数的显示位置。

### parameter.category

可用性

Flash MX 2004。

用法

`parameter.category`

描述

属性；一个字符串，它指定 `screen` 参数或 `componentInstance` 参数的 `category` 属性。此属性提供一种显示参数列表的替代方法。Flash 用户界面不提供此功能。

## parameter.insertItem()

可用性

Flash MX 2004。

用法

```
parameter.insertItem(index, name, value, type)
```

参数

**index** 一个从零开始的整数索引，它指示项目插入列表、对象或数组的位置。如果索引为 0，项目则插入列表起始处。如果索引大于列表的大小，新项目则插入数组末尾。

**name** 一个字符串，它指定要插入的项目的名称。对于对象参数，这是必需的参数。

**value** 一个字符串，它指定要插入的项目的值。

**type** 一个字符串，它指定要插入的项目的类型。

返回

无。

描述

方法；将一个项目插入列表、对象或数组。如果参数是一个列表、对象或数组，则 **value** 属性为一个数组。

示例

下面的示例在 **labelPlacement** 参数中插入 **New Value** 的值：

```
// Select an instance of a Button component on the Stage.  
var parms = fl.getDocumentDOM().selection[0].parameters;  
parms[2].insertItem(0, "name", "New Value", "String");  
var values = parms[2].value;  
for(var prop in values){  
    fl.trace("labelPlacement parameter value = " + values[prop].value);  
}
```

## parameter.listIndex

可用性

Flash MX 2004。

用法

```
parameter.listIndex
```

描述

属性；选择的列表项目的值。仅当 **parameter.valueType** 为 **List** 时，此属性才有效。

示例

下面的示例设置幻灯片的第一个参数，即 **autoKeyNav** 参数。为了将该参数设置为可接受的值之一（**true**、**false** 或 **inherit**），**parameter.listIndex** 被设置为列表中的项目的索引（0 对应于 **true**，1 对应于 **false**，2 对应于 **inherit**）。

**Parameter** 对象

```
var parms = fl.getDocumentDOM().screenOutline.screens[1].parameters;  
parms[0].listIndex = 1;
```

## parameter.name

可用性

Flash MX 2004。

用法

```
parameter.name
```

描述

只读属性；一个字符串，它指定参数的名称。

示例

下面的示例显示所选组件的第五个参数的名称：

```
var parms = fl.getDocumentDOM().selection[0].parameters;  
fl.trace("name: " + parms[4].name);
```

下面的示例为指定屏幕显示的第五个参数的名称：

```
var parms = fl.getDocumentDOM().screenOutline.screens[1].parameters; fl.trace("name: " + parms[4].name);
```

## parameter.removeItem()

可用性

Flash MX 2004。

用法

```
parameter.removeItem(index)
```

参数

**index** 要从屏幕或组件属性中删除的项目的零始整数索引。

返回

无。

描述

方法；删除屏幕或组件参数的列表、对象或数组类型的一个元素。

示例

下面的示例从组件的 `labelPlacement` 参数中删除索引 1 处的元素：

```
// Select an instance of a Button component on the Stage.
var parms = fl.getDocumentDOM().selection[0].parameters;
var values = parms[2].value;
fl.trace("--Original--");
for(var prop in values){
fl.trace("labelPlacement value = " + values[prop].value);
}
parms[2].removeItem(1);

var newValues = parms[2].value;
fl.trace("--After Removing Item--");
for(var prop in newValues){
fl.trace("labelPlacement value = " + newValues[prop].value);
}
```

下面的示例从屏幕的 `autoKeyNav` 参数中删除索引 1 处的元素：

```
// Open a presentation document.
var parms = fl.getDocumentDOM().screenOutline.screens[1].parameters;
var values = parms[0].value;
fl.trace("--Original--");
for(var prop in values){
fl.trace("autoKeyNav value = " + values[prop].value);
}
parms[0].removeItem(1);

var newValues = parms[0].value;
fl.trace("--After Removing Item--");
for(var prop in newValues){
fl.trace("autoKeyNav value = " + newValues[prop].value);
}
```

## parameter.value

可用性

Flash MX 2004。

用法

```
parameter.value
```

描述

属性；与“组件”检查器的“参数”选项卡、“属性”检查器的“参数”选项卡或屏幕“属性”检查器中的“值”字段相对应。  
`value` 属性的类型由参数的 `valueType` 属性确定（请参阅 [parameter.valueType](#)）。

## parameter.valueType

可用性

Flash MX 2004。

用法

```
parameter.valueType
```



#### 描述

只读属性；一个字符串，它指示屏幕或组件参数的类型。类型可以为下列值之一："Default"、"Array"、"Object"、"List"、"String"、"Number"、"Boolean"、"Font Name"、"Color"、"Collection"、"Web Service URL" 或 "Web Service Operation"。

另请参见

[parameter.value](#)

## parameter.verbose

#### 可用性

Flash MX 2004。

#### 用法

```
parameter.verbose
```

#### 描述

属性；指定参数的显示位置。如果此属性的值为 0 (**nonverbose**)，则参数只显示在“组件”检查器中。如果值为 1 (**verbose**)，则参数显示在组件检查器和“属性”检查器的“参数”选项卡中。

## 第 32 章 : Path 对象

可用性

Flash MX 2004。

描述

**Path** 对象定义线段（直线、曲线或两者）的序列，通常在创建可扩展工具时使用。下面的示例显示从 **flash** 对象返回的 **Path** 对象的一个实例：

```
path = fl.drawingLayer.newPath();
```

请参阅 [drawingLayer](#) 对象。

方法摘要

**Path** 对象具有以下方法：

方法	描述
<a href="#">path.addCubicCurve()</a>	向路径追加一条三次贝塞尔曲线线段。
<a href="#">path.addCurve()</a>	向路径追加一条二次贝塞尔曲线线段。
<a href="#">path.addPoint()</a>	向路径添加一个点。
<a href="#">path.clear()</a>	删除路径中的所有点。
<a href="#">path.close()</a>	在路径第一个点的位置追加一个点，然后将路径扩展到该点，使路径闭合。
<a href="#">path.makeShape()</a>	使用当前的笔触和填充设置在舞台上创建一个形状。
<a href="#">path.newContour()</a>	在路径中开始一个新轮廓。

属性摘要

**Path** 对象具有以下属性：

属性	描述
<a href="#">path.nPts</a>	只读；一个整数，它表示路径中的点数。

### path.addCubicCurve()

可用性

Flash MX 2004。

用法

```
path.addCubicCurve(xAnchor, yAnchor, x2, y2, x3, y3, x4, y4)
```

参数

**xAnchor** 一个浮点数，它指定第一个控制点的 **x** 位置。

**yAnchor** 一个浮点数，它指定第一个控制点的 **y** 位置。

**x2** 一个浮点数，它指定第二个控制点的 **x** 位置。

**Path 对象**

**y2** 一个浮点数，它指定第二个控制点的 **y** 位置。

**x3** 一个浮点数，它指定第三个控制点的 **x** 位置。

**y3** 一个浮点数，它指定第三个控制点的 **y** 位置。

**x4** 一个浮点数，它指定第四个控制点的 **x** 位置。

**y4** 一个浮点数，它指定第四个控制点的 **y** 位置。

返回

无。

描述

方法；向路径追加一条三次贝塞尔曲线线段。

示例

下面的示例创建一个新路径，并将其存储在 **myPath** 变量中，然后将曲线赋给该路径：

```
var myPath = fl.drawingLayer.newPath();  
myPath.addCubicCurve(0, 0, 10, 20, 20, 20, 30, 0);
```

## path.addCurve()

可用性

Flash MX 2004。

用法

```
path.addCurve(xAnchor, yAnchor, x2, y2, x3, y3)
```

参数

**xAnchor** 一个浮点数，它指定第一个控制点的 **x** 位置。

**yAnchor** 一个浮点数，它指定第一个控制点的 **y** 位置。

**x2** 一个浮点数，它指定第二个控制点的 **x** 位置。

**y2** 一个浮点数，它指定第二个控制点的 **y** 位置。

**x3** 一个浮点数，它指定第三个控制点的 **x** 位置。

**y3** 一个浮点数，它指定第三个控制点的 **y** 位置。

返回

无。

描述

方法；向路径追加一条二次贝塞尔曲线线段。

示例

下面的示例创建一个新路径，并将其存储在 **myPath** 变量中，然后将曲线赋给该路径：

```
var myPath = fl.drawingLayer.newPath();  
myPath.addCurve(0, 0, 10, 20, 20, 0);
```

## path.addPoint()

可用性

Flash MX 2004。

用法

```
path.addPoint(x, y)
```

参数

**x** 一个浮点数，它指定该点的 **x** 位置。

**y** 一个浮点数，它指定该点的 **y** 位置。

返回

无。

描述

方法；向路径添加一个点。

示例

下面的示例创建一个新路径，并将其存储在 **myPath** 变量中，然后将新的点分配给该路径：

```
var myPath = fl.drawingLayer.newPath();  
myPath.addPoint(10, 100);
```

## path.clear()

可用性

Flash MX 2004。

用法

```
path.clear()
```

参数

无。

返回

无。

描述

方法；删除路径中的所有点。

示例

下面的示例删除存储在 **myPath** 变量中的路径中的所有点：

```
var myPath = fl.drawingLayer.newPath();  
myPath.clear();
```

## path.close()

可用性

Flash MX 2004。

用法

```
path.close()
```

参数

无。

返回

无。

描述

方法；在路径第一个点的位置追加一个点，然后将路径扩展到该点，使路径闭合。如果路径没有点，则不添加点。

示例

下面的示例创建一个闭合路径：

```
var myPath = fl.drawingLayer.newPath();  
myPath.close();
```

## path.makeShape()

可用性

Flash MX 2004。

用法

```
path.makeShape([bSupressFill [, bSupressStroke]])
```

参数

**bSupressFill** 一个布尔值，如果设置为 **true**，则禁止将应用到形状的填充。默认值为 **false**。此参数是可选的。

**bSupressStroke** 一个布尔值，如果设置为 **true**，则禁止将应用到形状的笔触。默认值为 **false**。此参数是可选的。

返回

无。

描述

方法；使用当前的笔触和填充设置在舞台上创建一个形状。形状创建后路径即被清除。此方法有两个可选参数可禁止结果形状对象的填充和笔触。如果省略这些参数或将它们设置为 **false**，则使用填充和笔触的当前值。

示例

下面的示例使用当前的填充但不使用笔触创建一个形状：

**Path 对象**

```
var myPath = fl.drawingLayer.newPath();  
myPath.makeShape(false, true);
```

## path.newContour()

**可用性**

Flash MX 2004。

**用法**

```
path.newContour()
```

**参数**

无。

**返回**

无。

**描述**

方法；在路径中开始一个新轮廓。

**示例**

下面的示例创建了一个空心方形：

```
var myPath = fl.drawingLayer.newPath();  
myPath.addPoint(0, 0);  
myPath.addPoint(0, 30);  
myPath.addPoint(30, 30);  
myPath.addPoint(30, 0);  
myPath.addPoint(0, 0);  
  
myPath.newContour();  
myPath.addPoint(10, 10);  
myPath.addPoint(10, 20);  
myPath.addPoint(20, 20);  
myPath.addPoint(20, 10);  
myPath.addPoint(10, 10);  
  
myPath.makeShape();
```

## path.nPts

**可用性**

Flash MX 2004。

**用法**

```
path.nPts
```

**Path 对象**

## 描述

只读属性；一个整数，它表示路径中的点数。新路径有 0 个点。

## 示例

下面的示例使用“输出”面板显示 `myPath` 变量所引用路径中的点数：

```
var myPath = fl.drawingLayer.newPath();
var numOfPoints = myPath.nPts;
fl.trace("Number of points in the path: " + numOfPoints);
// Displays: Number of points in the path: 0
```

## 第 33 章 : presetItem 对象

可用性

Flash CS4 Professional.

描述

`presetItem` 对象表示“动画预设”面板（“窗口”>“动画预设”）中的项目（预设或文件夹）。`presetItem` 对象的数组为 `presetPanel` 对象 (`presetPanel.items`) 的属性。

`presetItem` 对象的所有属性都是只读的。若要执行删除、重命名或移动项目等任务，请使用 `presetPanel` 对象的方法。

属性摘要

您可以使用 `presetItem` 对象的以下属性：

属性	描述
<code>presetItem.isDefault</code>	指定该项目是随 Flash 一起安装的项目，还是由您或他人创建的自定义项目。
<code>presetItem.isFolder</code>	指定“动画预设”面板中的项目是文件夹还是预设。
<code>presetItem.level</code>	“动画预设”面板文件夹结构中的项目的层级。
<code>presetItem.name</code>	预设或文件夹的名称，不含路径信息。
<code>presetItem.open</code>	指定“动画预设”面板中的文件夹当前是否展开。
<code>presetItem.path</code>	“动画预设”面板文件夹树中项目的路径和项目名称。

### presetItem.isDefault

可用性

Flash CS4 Professional.

用法

```
presetItem.isDefault
```

描述

只读属性；一个布尔值，它指定项目是安装 Flash 时附带的项目 (`true`) 还是由您或其他人创建的自定义项目 (`false`)。如果此值为 `true`，则可以将该项目视为“只读”项目；不能移动、删除该项目，也不能对其应用任何类似的操作。

示例

下面的示例显示“动画预设”面板的内容，并指示某项目是否是安装 Flash 时附带的项目：

```
fl.outputPanel.clear();
var presetItemArray=fl.presetPanel.items;
for (i=0;i<presetItemArray.length; i++){
    var presetItem = presetItemArray[i];
    fl.trace(presetItem.name +", default =" + presetItem.isDefault);
}
```



## presetItem.isFolder

可用性

Flash CS4 Professional.

用法

```
presetItem.isFolder
```

描述

只读属性：一个布尔值，它指定“动画预设”面板中的项目是文件夹 (**true**) 还是预设 (**false**)。

示例

下面的示例显示“动画预设”面板中的第一个项目是文件夹，第二个项目是预设：

```
var presetItemArray=fl.presetPanel.items;  
fl.trace(presetItemArray[0].isFolder);  
fl.trace(presetItemArray[1].isFolder);
```

## presetItem.level

可用性

Flash CS4 Professional.

用法

```
presetItem.level
```

描述

只读属性：一个整数，它指定“动画预设”面板的文件夹结构中项目的层级。“默认文件夹”和“自定义预设”文件夹为 **0** 级。

示例

下面的示例显示“动画预设”面板中的第一个项目为 **0** 级，第二个项目为 **1** 级：

```
var presetItemArray=fl.presetPanel.items;  
fl.trace(presetItemArray[0].level);  
fl.trace(presetItemArray[1].level);
```

## presetItem.name

可用性

Flash CS4 Professional.

用法

```
presetItem.name
```

描述

只读属性：一个字符串，它表示预设或文件夹的名称，不包含路径信息。

示例

请参阅 [presetItem.path](#)。

## presetItem.open

可用性

Flash CS4 Professional.

用法

presetItem.open

描述

只读属性：它指定“动画预设”面板中的文件夹当前是 (**true**) 否 (**false**) 是展开的。

如果项目不是文件夹，则此属性为 **true**。若要确定项目是文件夹还是预设，请使用 [presetItem.isFolder](#)。

示例

下面的示例显示有关“动画预设”面板中的文件夹是展开还是折叠的信息：

```
fl.outputPanel.clear();
var presetItemArray=fl.presetPanel.items;
for (i=0;i<presetItemArray.length; i++){
    var presetItem = presetItemArray[i];
    if (presetItem.isFolder) {
        var status = presetItem.open ? "Open" : "Closed"
        fl.trace(presetItem.level + "-" + presetItem.name +" folder is " + status);
    }
}
```

## presetItem.path

可用性

Flash CS4 Professional.

用法

presetItem.path

描述

只读属性：一个字符串，它表示“动画预设”面板文件夹树中项目的路径和项目名称。

示例

下面的示例说明 [presetItem.name](#) 和 [presetItem.path](#) 中的值之间的差别。

```
fl.outputPanel.clear();
var presetItemArray=fl.presetPanel.items;
for (i=0;i<presetItemArray.length; i++){
    var presetItem = presetItemArray[i];
    fl.trace("Name: " + presetItem.name + "\n" + "Path: " + presetItem.path);
    fl.trace("");
}
```

## 第 34 章 : presetPanel 对象

可用性

Flash CS4 Professional.

描述

`presetPanel` 对象表示“动画预设”面板（“窗口”>“动画预设”）。它是 `flash` 对象的属性 (`fl.presetPanel`)。

方法摘要

可以对 `presetPanel` 对象使用以下方法：

方法	描述
<code>presetPanel.addNewItem()</code>	如果当前在舞台上选择了单个补间动画，则将该动画添加到“动画预设”面板。
<code>presetPanel.applyPreset()</code>	将指定的或当前所选的预设应用于舞台上当前选择的项目。
<code>presetPanel.deleteFolder()</code>	从“动画预设”面板的文件夹树中删除指定的文件夹及其所有子文件夹。
<code>presetPanel.deleteItem()</code>	从“动画预设”面板中删除指定的预设。
<code>presetPanel.expandFolder()</code>	展开或折叠“动画预设”面板中当前选择的一个或多个文件夹。
<code>presetPanel.exportItem()</code>	将当前所选的或指定的预设导出到 XML 文件。
<code>presetPanel.findItemIndex()</code>	返回一个整数，它表示项目在“动画预设”面板中的索引位置。
<code>presetPanel.getSelectedItems()</code>	返回 <code>presetItem</code> 对象的数组，这些对象对应于“动画预设”面板中当前选择的项目。
<code>presetPanel.importItem()</code>	将指定的 XML 文件中的预设添加到“动画预设”面板。
<code>presetPanel.moveToFolder()</code>	将指定的项目移动到指定的文件夹。
<code>presetPanel.newFolder()</code>	在“动画预设”面板的文件夹树中创建文件夹。
<code>presetPanel.renameItem()</code>	将当前选择的预设或文件夹重命名为指定的名称。
<code>presetPanel.selectItem()</code>	在“动画预设”面板中选择或取消选择项目。

属性摘要

可以对 `presetPanel` 对象使用以下属性：

属性	描述
<code>presetPanel.items</code>	“动画预设”面板中的 <code>presetItem</code> 对象的数组。

### `presetPanel.addNewItem()`

可用性

Flash CS4 Professional.

用法

```
fl.presetPanel.addNewItem( [namePath] );
```

#### 参数

**namePath** 一个字符串，它指定要添加到“动画预设”面板中的项目的路径和名称。此参数是可选的。

#### 返回

一个布尔值：如果成功添加项目，则为 **true**；否则为 **false**。

#### 说明

方法；如果当前在舞台上选择了单个补间动画，则将该动画添加到“动画预设”面板中具有指定名称的指定文件夹中。

**namePath** 中指定的路径必须存在于该面板中。

如果存在与 **namePath** 匹配的预设，则此方法无效，并返回 **false**。

如果没有为 **namePath** 传递值，则项目将添加到名为“Custom preset n”的 Custom Presets 文件夹中，每次以此方式添加项目时，**n** 都会递增。

#### 示例

假定在舞台上选择了单个补间动画，则下面的代码将名为 Bouncing Ball 的预设添加到 Custom Presets 文件夹：

```
fl.presetPanel.addNewItem("Custom Presets/Bouncing Ball");
```

另请参见

[presetPanel.newFolder\(\)](#)

## presetPanel.applyPreset()

#### 可用性

Flash CS4 Professional.

#### 用法

```
presetPanel.applyPreset( [presetPath] )
```

#### 参数

**presetPath** 一个字符串，它指定要应用的预设的完整路径和名称，显示在“动画预设”面板中。此参数是可选的；如果没有传递值，则应用当前选择的预设。

#### 返回

一个布尔值：如果成功应用预设，则为 **true**；否则为 **false**。

#### 说明

方法；将指定的或当前所选的预设应用于舞台上当前选择的项目。该项目必须是补间动画、元件或可转换为元件的项目。如果该项目是补间动画，则会将其当前动画替换为所选预设，而不会请求用户确认。

在下列情况下，此方法将失败：

- 指定为 **presetPath** 的路径不存在。
- 没有为 **presetPath** 传递值并且未选择预设。
- 没有为 **presetPath** 传递值并且选择了多个预设。
- 在舞台上选择的项目不是元件并且不能转换为元件。

#### 示例

下面的示例将 aDribble 预设应用于舞台上当前选择的项目：

```
var result = fl.presetPanel.applyPreset("Custom Presets/Bounces/aDribble");  
fl.trace(result);
```

## presetPanel.deleteFolder()

#### 可用性

Flash CS4 Professional.

#### 用法

```
presetPanel.deleteFolder( [folderPath]
```

#### 参数

**folderPath** 一个字符串，它指定要从“动画预设”面板中删除的文件夹。此参数是可选的。

#### 返回

一个布尔值：如果成功删除文件夹，则为 **true**；否则为 **false**。

#### 说明

方法；从“动画预设”面板的文件夹树中删除指定的文件夹及其所有子文件夹。这些文件夹中的所有预设也将被删除。不能删除 **Default Presets** 文件夹中的文件夹。

如果没有为 **folderPath** 传递值，则将删除当前选择的所有文件夹。

注：不请求用户确认就会删除文件夹，并且无法撤消此操作。

#### 示例

下面的代码删除 **Custom Presets** 文件夹下名为 **Bouncing** 的文件夹；同时一并删除 **Bouncing** 的所有子文件夹：

```
fl.presetPanel.deleteFolder("Custom Presets/Bouncing");
```

#### 另请参见

[presetPanel.deleteItem\(\)](#)

## presetPanel.deleteItem()

#### 可用性

Flash CS4 Professional.

#### 用法

```
presetPanel.deleteItem( [namePath] )
```

#### 参数

**namePath** 一个字符串，它指定要从“动画预设”面板中删除的项目的路径和名称。此参数是可选的。

#### 返回

一个布尔值：如果成功删除项目，则为 **true**；否则为 **false**。

#### 说明

方法；从“动画预设”面板中删除指定的预设。如果没有为 **namePath** 传递值，则将删除当前选择的所有预设。不能删除 **Default Presets** 文件夹中的项目。

注：不请求用户确认就会删除项目，并且无法撤消此操作。

#### 示例

下面的代码从 **Custom Presets** 文件夹中删除名为 **aDribble** 的预设：

```
fl.presetPanel.deleteItem("Custom Presets/aDribble");
```

另请参见

[presetPanel.deleteFolder\(\)](#)

## presetPanel.expandFolder()

#### 可用性

Flash CS4 Professional.

#### 用法

```
presetPanel.expandFolder( [bExpand [, bRecurse [, folderPath] ] ] )
```

#### 参数

**bExpand** 一个布尔值，它指定是展开 (**true**) 还是折叠 (**false**) 文件夹。此参数是可选的；默认值为 **true**。

**bRecurse** 一个布尔值，它指定是展开 (**true**) 还是折叠 (**false**) 文件夹的子文件夹。此参数是可选的；默认值为 **false**。

**folderPath** 一个字符串，它指定要展开或折叠的文件夹的路径。此参数是可选的。

#### 返回

一个布尔值：如果成功展开或折叠文件夹，则为 **true**；否则为 **false**。

#### 说明

方法；展开或折叠“动画预设”面板中当前选择的一个或多个文件夹。若要展开或折叠当前选择的文件夹以外的文件夹，请为 **folderPath** 传递值。

#### 示例

下面的示例展开 **Custom Presets** 文件夹，但不展开其子文件夹：

```
fl.presetPanel.expandFolder(true, false, "Custom Presets");
```

下面的示例展开 **Custom Presets** 文件夹及其所有子文件夹：

```
fl.presetPanel.expandFolder(true, true, "Custom Presets");
```

## presetPanel.exportItem()

可用性

Flash CS4 Professional.

用法

```
presetPanel.exportItem(fileURI [, namePath] )
```

参数

**fileURI** 一个字符串，表示为 **file:/// URI**，它指定导出的文件的路径和可选文件名。有关详细信息，请参阅下面的“描述”。

**namePath** 一个字符串，它指定要从“动画预设”面板中选择的项目的路径和名称。此参数是可选的。

返回

一个布尔值：如果成功导出预设，则为 **true**；否则为 **false**。

说明

方法；将当前所选的或指定的预设导出到 XML 文件。只能导出预设；如果试图导出文件夹，则此方法将失败。如果试图覆盖磁盘上的文件，此方法也将失败。

如果没有在 **fileURI** 中指定文件名（即 **fileURI** 的最后一个字符是斜杠 (/）），则使用预设导出时的名称保存导出的预设。如果没有为 **namePath** 指定值，则将导出当前选择的预设。请参见下例。

示例

下面的示例演示向此方法传递不同的参数时将创建哪些文件，并通知您是否成功创建指定的文件。运行此示例之前，请选择 **Default Presets** 文件夹中的 **fly-in-left** 预设，并在磁盘上创建 **My Presets** 文件夹。

```
//Exports fly-in-left to C:\My Presets\fly-in-left.xml  
fl.presetPanel.exportItem("file:///C:/My Presets/");  
//Exports fly-in-left to C:\My Presets\myFavoritePreset.xml  
fl.presetPanel.exportItem("file:///C:/My Presets/myFavoritePreset.xml");  
// Exports the "pulse" preset to C:\My Presets\pulse.xml  
fl.presetPanel.exportItem("file:///C:/My Presets/", "Default Presets/pulse");  
// Exports the "pulse" preset to C:\My Presets\thePulsePreset.xml  
fl.presetPanel.exportItem("file:///C:/My Presets/thePulsePreset.xml", "Default Presets/pulse");
```

另请参见

[presetPanel.importItem\(\)](#)

## presetPanel.findItemIndex()

可用性

Flash CS4 Professional.

用法

```
presetPanel.findItemIndex([presetName])
```

参数

**presetName** 一个字符串，它指定返回其索引值的预设的名称。此参数是可选的。

返回

一个整数，它表示 `presetPanel.items` 数组中的指定预设的索引。如果没有为 `presetName` 传递值，则返回当前指定的预设的索引。在下列情况下，此方法返回 -1：

- 没有为 `presetName` 传递值并且未选择预设。
- 没有为 `presetName` 传递值并且选择了多个预设。
- 为其传递值的 `presetName` 与面板中的项目不对应。

说明

方法；返回一个整数，该整数表示项目在“动画预设”面板中的索引位置。

示例

下面的代码显示当前选择的预设的索引值和完整路径名：

```
// Select one preset in the Motions Preset panel before running this code
var selectedPreset = fl.presetPanel.findItemIndex();
fl.trace(selectedPreset);
fl.trace(fl.presetPanel.items[selectedPreset].path);
```

## presetPanel.getSelectedItems()

可用性

Flash CS4 Professional.

用法

```
presetPanel.getSelectedItems()
```

参数

无。

返回

`presetItem` 对象的数组。

说明

方法；返回 `presetItem` 对象的数组，这些对象对应于“动画预设”面板中当前选择的项目（请参阅 [presetItem 对象](#)）。该数组中的每个项目都表示文件夹或预设。

示例

下面的代码显示“动画预设”面板中当前所选项目的完整路径名：

```
var itemArray = fl.presetPanel.getSelectedItems();
var length = itemArray.length
for (x=0; x<length; x++) {
    fl.trace(itemArray[x].path);
}
```

另请参见

[presetPanel.items](#)



## presetPanel.importItem()

可用性

Flash CS4 Professional.

用法

```
presetPanel.importItem(fileURI [,namePath ])
```

参数

**fileURI** 一个字符串，表示为 **file:/// URI**，它指定要导入为“动画预设”面板中的预设的 XML 文件。

**namePath** 一个字符串，它指定要放置导入文件的文件夹以及要为该文件指定的名称。此参数是可选的。

返回

一个布尔值：如果成功导入文件，则为 **true**；否则为 **false**。

说明

方法；将指定的 XML 文件中的预设添加到“动画预设”面板。**namePath** 中指定的路径必须存在于该面板中。

若要创建可导入的 XML 文件，请使用 [presetPanel.exportItem\(\)](#)。

如果没有为 **namePath** 传递值，则将导入的预设放入 **Custom Presets** 文件夹中，并为其指定与所导入文件相同的名称（无 XML 扩展名）。

示例

下面的示例将预设导入 **Custom Presets/Pulse** 文件夹，并命名为 **fastPulse**。

```
fl.presetPanel.importItem("file:///C:/My Presets/thePulsePreset.xml", "Custom Presets/Pulse/fastPulse");
```

另请参见

[presetPanel.exportItem\(\)](#)

## presetPanel.items

可用性

Flash CS4 Professional.

用法

```
presetPanel.items
```

描述

属性；“动画预设”面板中 **presetItem** 对象的数组（请参阅 [presetItem](#) 对象）。该数组中的每个项目都表示文件夹或预设。

示例

下面的代码显示“动画预设”面板中的项目的完整路径名：

**presetPanel** 对象

```
var itemArray = fl.presetPanel.items;
var length = itemArray.length
for (x=0; x<length; x++) {
    fl.trace(itemArray[x].path);
}
```

另请参见

[presetPanel.getSelectedItems\(\)](#)

## presetPanel.moveToFolder()

可用性

Flash CS4 Professional.

用法

```
presetPanel.moveToFolder(folderPath [, namePath] )
```

参数

**folderPath** 一个字符串，它指定项目要移至的“动画预设”面板中的文件夹的路径。

**namePath** 一个字符串，它指定要移动的项目的路径和名称。此参数是可选的。

返回

一个布尔值：如果成功移动项目，则为 **true**；否则为 **false**。

说明

方法；将指定的项目移动到指定的文件夹。

如果为 **folderPath** 传递空字符串 ("")，则将项目移动到 **Custom Presets** 文件夹。如果没有为 **namePath** 传递值，则移动当前选择的项目。

不能在对 **Default Presets** 文件夹进行项目移动操作。

示例

在下面的示例中，将当前所选项目移动到 **Custom Presets/Bouncing** 文件夹，然后将 **Fast Bounce** 预设移动到同一文件夹：

```
fl.presetPanel.moveToFolder("Custom Presets/Bouncing");
fl.presetPanel.moveToFolder("Custom Presets/Bouncing" , "Custom Presets/Fast Bounce");
```

## presetPanel.newFolder()

可用性

Flash CS4 Professional.

用法

```
presetPanel.newFolder( [folderPath] )
```

参数

**folderPath** 一个字符串，它指定在“动画预设”面板中添加新文件夹的位置以及新文件夹的名称。此参数是可选的。

#### 返回

一个布尔值：如果成功添加文件夹，则为 **true**；否则为 **false**。

#### 说明

方法；在“动画预设”面板的文件夹树中创建文件夹。用此命令只能创建一个新文件夹层级。也就是说，如果为 **folderPath** 传递“Custom Presets/My First Folder/My Second Folder”，则“Custom Presets/My First Folder”必须存在于文件夹树中。

如果没有为 **folderPath** 传递值，则将在 **Custom Presets** 下的第一层级创建名为“Untitled folder n”的文件夹，每次以此方式添加文件夹时，**n** 都会递增。

注：不能将文件夹添加到 **Default Presets** 文件夹中。

#### 示例

下面的示例在 **Custom Presets** 文件夹下添加名为 **Bouncing** 的文件夹：

```
fl.presetPanel.newFolder("Custom Presets/Bouncing");
```

另请参见

[presetPanel.addItem\(\)](#)

## presetPanel.renameItem()

#### 可用性

Flash CS4 Professional.

#### 用法

```
presetPanel.renameItem(newName)
```

#### 参数

**newName** 一个字符串，它为预设或文件夹指定新名称。

#### 返回

一个布尔值：如果成功重命名预设或文件夹，则为 **true**；否则为 **false**。

#### 说明

方法；将当前选择的预设或文件夹重命名为指定的名称。仅当选择 **Custom Presets** 文件夹中的单个预设或文件夹时，此方法才会成功。在下列情况下，此方法将失败：

- 未选择项目。
- 选择多个项目。
- 所选项目位于 **Default Presets** 文件夹中。
- 名为 **newName** 的项目与所选项目位于同一位置。

#### 示例

下面的示例将 **Custom Presets** 文件夹中当前选择的预设重命名为 **Bounce Faster**。

```
var renamed = fl.presetPanel.renameItem("Bounce Faster");  
fl.trace(renamed);
```

## presetPanel.selectItem()

可用性

Flash CS4 Professional.

用法

```
presetPanel.selectItem(namePath [, bReplaceCurrentSelection [, bSelect] ])
```

参数

**namePath** 一个字符串，它指定要从“动画预设”面板中选择的项目的路径和名称。

**bReplaceCurrentSelection** 一个布尔值，它指示指定项目是替换当前选择 (**true**)，还是添加到当前选择中 (**false**)。此参数是可选的；默认值为 **true**。

**bSelect** 一个布尔值，它指定是选择 (**true**) 还是取消选择项目 (**false**)。此参数是可选的；默认值为 **true**。如果为 **bSelect** 传递 **false**，则忽略 **bReplaceCurrentSelection** 的值。

返回

一个布尔值：如果成功选择或取消选择项目，则为 **true**；否则为 **false**。

说明

方法；在“动画预设”面板中选择或取消选择项目，可选择替换当前所选的任何项目。

示例

下面的代码将 **fly-in-blur-right** 预设添加到“动画预设”面板中的当前所选预设（如果有）：

```
fl.presetPanel.selectItem("Default Presets/fly-in-blur-right", false);
```

## 第 35 章 : Rectangle 对象

继承关系 [Element 对象](#) > [Shape 对象](#) > Rectangle 对象

可用性

Flash CS3 Professional。

描述

Rectangle 对象是使用“基本矩形”工具绘制的形状。若要确定一个项目是否为 Rectangle 对象，请使用 [shape.isRectangleObject](#)。

属性摘要

除 [Shape 对象](#) 的属性外，还可以对 Rectangle 对象使用下列属性。若要设置 Rectangle 对象的属性，请使用 [document.setRectangleObjectProperty\(\)](#)。

属性	描述
<a href="#">RectangleObject.bottomLeftRadius</a>	只读；一个浮点值，它设置 Rectangle 对象的左下角的半径。
<a href="#">RectangleObject.bottomRightRadius</a>	只读；一个浮点值，它设置 Rectangle 对象的右下角的半径。
<a href="#">RectangleObject.lockFlag</a>	只读；一个布尔值，它确定 Rectangle 对象的不同角是否可以具有不同的半径值。
<a href="#">RectangleObject.topLeftRadius</a>	只读；一个浮点值，它设置 Rectangle 对象所有角的半径，或只设置 Rectangle 对象左上角的半径。
<a href="#">RectangleObject.topRightRadius</a>	只读；一个浮点值，它设置 Rectangle 对象的右上角的半径。

### RectangleObject.bottomLeftRadius

可用性

Flash CS3 Professional。

用法

```
RectangleObject.bottomLeftRadius
```

描述

只读属性；一个浮点值，它设置 Rectangle 对象的左下角的半径。如果 [RectangleObject.lockFlag](#) 为 true，则尝试设置该值不会有任何效果。

若要设置该值，请使用 [document.setRectangleObjectProperty\(\)](#)。

另请参见

[document.setRectangleObjectProperty\(\)](#)、[RectangleObject.bottomRightRadius](#)、[RectangleObject.lockFlag](#)、[RectangleObject.topLeftRadius](#)、[RectangleObject.topRightRadius](#)

## RectangleObject.bottomRightRadius

可用性

Flash CS3 Professional。

用法

`RectangleObject.bottomRightRadius`

描述

只读属性；一个浮点值，它设置 **Rectangle** 对象的右下角的半径。如果 `RectangleObject.lockFlag` 为 `true`，则尝试设置该值不会有任何效果。

若要设置该值，请使用 `document.setRectangleObjectProperty()`。

另请参见

[document.setRectangleObjectProperty\(\)](#)、[RectangleObject.bottomLeftRadius](#)、[RectangleObject.lockFlag](#)、[RectangleObject.topLeftRadius](#)、[RectangleObject.topRightRadius](#)

## RectangleObject.lockFlag

可用性

Flash CS3 Professional。

用法

`RectangleObject.lockFlag`

描述

只读属性；一个布尔值，它确定 **Rectangle** 对象的不同角是否可以具有不同的半径值。如果该值为 `true`，则所有角的值均被指定为 `RectangleObject.topLeftRadius`。如果为 `false`，则每个角的半径可以单独设置。

若要设置该值，请使用 `document.setRectangleObjectProperty()`。

另请参见

[document.setRectangleObjectProperty\(\)](#)、[RectangleObject.bottomLeftRadius](#)、[RectangleObject.bottomRightRadius](#)、[RectangleObject.topLeftRadius](#)、[RectangleObject.topRightRadius](#)

## RectangleObject.topLeftRadius

可用性

Flash CS3 Professional。

用法

`RectangleObject.topLeftRadius`

#### 描述

只读属性；一个浮点值，它设置 **Rectangle** 的所有角的半径（如果 **RectangleObject.lockFlag** 为 **true**）或只设置左上角的半径（如果 **RectangleObject.lockFlag** 为 **false**）。

若要设置该值，请使用 [document.setRectangleObjectProperty\(\)](#)。

#### 另请参见

[document.setRectangleObjectProperty\(\)](#)、[RectangleObject.bottomLeftRadius](#)、[RectangleObject.bottomRightRadius](#)、[RectangleObject.lockFlag](#)、[RectangleObject.topRightRadius](#)

## RectangleObject.topRightRadius

#### 可用性

Flash CS3 Professional。

#### 用法

```
RectangleObject.topRightRadius
```

#### 描述

只读属性；一个浮点值，它设置 **Rectangle** 对象的右上角的半径。如果 **RectangleObject.lockFlag** 为 **true**，则尝试设置该值不会有任何效果。

若要设置该值，请使用 [document.setRectangleObjectProperty\(\)](#)。

#### 另请参见

[document.setRectangleObjectProperty\(\)](#)、[RectangleObject.bottomLeftRadius](#)、[RectangleObject.bottomRightRadius](#)、[RectangleObject.lockFlag](#)、[RectangleObject.topLeftRadius](#)

## 第 36 章 : Shape 对象

继承关系 [Element 对象](#) > Shape 对象

可用性

Flash MX 2004。

描述

Shape 对象是 Element 对象的子类。在舞台上操作或创建几何形状时，Shape 对象提供的控制比绘图 API 更精确。这种控制是必要的，这使得脚本可以创建有用的效果及其他绘制命令（请参阅 [Element 对象](#)）。

对于所有更改形状及其任何从属部分的 Shape 方法和属性，它们必须放置在 [shape.beginEdit\(\)](#) 调用和 [shape.endEdit\(\)](#) 调用之间才能正确工作。

方法摘要

除 Element 对象的方法外，还可以对 Shape 对象使用以下方法：

方法	描述
<a href="#">shape.getCubicSegmentPoints()</a>	返回定义三次曲线的点数组。
<a href="#">shape.beginEdit()</a>	定义编辑会话的开始。
<a href="#">shape.deleteEdge()</a>	删除指定的边缘。
<a href="#">shape.endEdit()</a>	定义形状编辑会话的结束。

属性摘要

除 Element 对象的属性外，Shape 对象还具有以下属性：

属性	描述
<a href="#">shape.contours</a>	只读；形状的 Contour 对象的数组（请参阅 <a href="#">Contour 对象</a> ）。
<a href="#">shape.edges</a>	只读；Edge 对象的数组（请参阅 <a href="#">Edge 对象</a> ）。
<a href="#">shape.isDrawingObject</a>	只读；如果为 true，该形状为绘制对象。
<a href="#">shape.isGroup</a>	只读；如果为 true，该形状是一个组合。
<a href="#">shape.isOvalObject</a>	只读；如果为 true，该形状是一个基本 Oval 对象（使用“椭圆”工具创建）。
<a href="#">shape.isRectangleObject</a>	只读；如果为 true，该形状是一个基本 Rectangle 对象（使用“矩形”工具创建）。
<a href="#">shape.members</a>	当前所选组中对象的数组。
<a href="#">shape.numCubicSegments</a>	只读；形状中三次线段的数量。
<a href="#">shape.vertices</a>	只读；Vertex 对象的数组（请参阅 <a href="#">Vertex 对象</a> ）。



## shape.beginEdit()

可用性

Flash MX 2004。

用法

```
shape.beginEdit()
```

参数

无。

返回

无。

描述

方法；定义编辑会话的开始。在发出任何更改 **Shape** 对象或其任何从属部分的命令前，必须先使用此方法。

示例

下面的示例使用了当前选定的形状，并删除了其边缘数组中的第一条边缘：

```
var shape = fl.getDocumentDOM().selection[0];  
shape.beginEdit();  
shape.deleteEdge(0);  
shape.endEdit();
```

## shape.contours

可用性

Flash MX 2004。

用法

```
shape.contours
```

描述

只读属性；形状的 **Contour** 对象的数组（请参阅 [Contour 对象](#)）。

示例

下面的示例将轮廓数组中的第一个轮廓存储在 **c** 变量中，然后将该轮廓的 [HalfEdge 对象](#) 存储在 **he** 变量中：

```
var c = fl.getDocumentDOM().selection[0].contours[0];  
var he = c.getHalfEdge();
```

## shape.deleteEdge()

可用性

Flash MX 2004。

**Shape 对象**

## 用法

```
shape.deleteEdge(index)
```

## 参数

**index** 从零开始的索引，它指定要从 [shape.edges](#) 数组中删除的边缘。此方法可更改 [shape.edges](#) 数组的长度。

## 返回

无。

## 描述

方法；删除指定的边缘。在使用此方法之前，必须先调用 [shape.beginEdit\(\)](#)。

## 示例

下面的示例使用了当前选择的形状，并删除了其边缘数组中的第一条边缘：

```
var shape = fl.getDocumentDOM().selection[0];
shape.beginEdit();
shape.deleteEdge(0);
shape.endEdit();
```

## shape.edges

## 可用性

Flash MX 2004。

## 用法

```
shape.edges
```

## 描述

只读属性；Edge 对象的数组（请参阅 [Edge 对象](#)）。

## shape.endEdit()

## 可用性

Flash MX 2004。

## 用法

```
shape.endEdit()
```

## 参数

无。

## 返回

无。

#### 描述

方法；定义形状编辑会话的结束。对 Shape 对象或其任何从属部分的所有更改都将应用于该形状。在发出任何更改 Shape 对象或其任何从属部分的命令之后，必须使用此方法。

#### 示例

下面的示例使用了当前选定的形状，并删除了其边缘数组中的第一条边缘：

```
var shape = fl.getDocumentDOM().selection[0];
shape.beginEdit();
shape.deleteEdge(0);
shape.endEdit();
```

## shape.getCubicSegmentPoints()

#### 可用性

Flash CS4 Professional.

#### 用法

```
shape.getCubicSegmentPoints(cubicSegmentIndex)
```

#### 参数

**cubicSegmentIndex** 一个整数，指定返回其点的三次线段。

#### 返回

定义 Edge 对象的三次曲线的点数组，该数组对应于指定的 cubicSegmentIndex（请参阅 [edge.cubicSegmentIndex](#)）。

#### 说明

方法；返回定义三次曲线的点数组。

#### 示例

下面的示例显示选定项第一个边缘的三次曲线上每个点的 x 和 y 值：

```
var elem = fl.getDocumentDOM().selection[0];
var index = elem.edges[0].cubicSegmentIndex;
var cubicPoints = elem.getCubicSegmentPoints(index);
for (i=0; i<cubicPoints.length; i++) {
    fl.trace("index " + i + " x: " + cubicPoints[i].x + " y: " + cubicPoints[i].y);
}
```

## shape.isDrawingObject

#### 可用性

Flash 8.

#### 用法

```
shape.isDrawingObject
```

#### 描述

只读属性；如果为 `true`，则该形状为绘制对象。

#### 示例

下面的示例将所选第一个对象存储在 `sel` 变量中，然后使用 `element.elementType` 和 `shape.isDrawingObject` 属性来确定所选项目是否为绘制对象：

```
var sel = fl.getDocumentDOM().selection[0];
var shapeDrawingObject = (sel.elementType == "shape") && sel.isDrawingObject;
fl.trace(shapeDrawingObject);
```

#### 另请参见

[document.crop\(\)](#)、[document.deleteEnvelope\(\)](#)、[document.intersect\(\)](#)、[document.punch\(\)](#)、[document.union\(\)](#)、[shape.isGroup](#)

## shape.isGroup

#### 可用性

Flash MX 2004。

#### 用法

`shape.isGroup`

#### 描述

只读属性；如果为 `true`，则该形状是一个组合。组可以包含不同类型的元素（如文本元素）和元件。但是，组本身被视为一个形状，无论组包含什么类型的元素，都可以使用 `shape.isGroup` 属性。

#### 示例

下面的示例将所选第一个对象存储在 `sel` 变量中，然后使用 `element.elementType` 和 `shape.isGroup` 属性来确定所选项目是否为组：

```
var sel = fl.getDocumentDOM().selection[0];
var shapeGroup = (sel.elementType == "shape") && sel.isGroup;
fl.trace(shapeGroup);
```

#### 另请参见

[shape.isDrawingObject](#)

## shape.isOvalObject

#### 可用性

Flash CS3 Professional。

#### 用法

`shape.isOvalObject`

#### 描述

只读属性；如果为 `true`，则该形状是一个基本椭圆对象（使用基本椭圆工具创建）。

**Shape 对象**

## 示例

如果第一个选定项目是基本椭圆对象，则下面的示例显示 "true"，否则显示 "false"：

```
var sel = fl.getDocumentDOM().selection[0];
fl.trace(sel.isOvalObject);
```

另请参见

[shape.isRectangleObject](#)

## shape.isRectangleObject

## 可用性

Flash CS3 Professional。

## 用法

```
shape.isRectangleObject
```

## 描述

只读属性；如果为 **true**，则该形状是一个基本矩形对象（使用基本矩形工具创建）。

## 示例

如果第一个选定项目是基本矩形对象，则下面的示例显示 "true"，否则显示 "false"：

```
var sel = fl.getDocumentDOM().selection[0];
fl.trace(sel.isRectangleObject);
```

另请参见

[shape.isOvalObject](#)

## shape.members

## 可用性

Flash CS4 Professional.

## 用法

```
shape.members
```

## 描述

只读属性；当前所选组中对象的数组。仅当 **shape.isGroup** 的值为 **true** 时，此属性才可用。**shape.members** 数组不包含组中的原始形状。

例如，如果组包含三个绘制对象和三个原始形状，则 **shape.members** 数组包含三个条目，每个条目分别对应于每个绘制对象。如果组仅包含原始形状，则该数组为空。

## 示例

下面的代码显示当前选定组中每个绘制对象的三次线段数：

```
var shapesArray = fl.getDocumentDOM().selection[0].members;
for (i=0; i<shapesArray.length; i++) {
    fl.trace(shapesArray[i].numCubicSegments);
}
```

另请参见  
[shape.isGroup](#)

## shape.numCubicSegments

可用性  
Flash CS4 Professional.

用法  
`shape.numCubicSegments`

描述  
只读属性；形状的三次线段数。

示例  
假定选择了正方形或矩形形状，则下面的示例在“输出”面板中显示“4”：

```
var theShape = fl.getDocumentDOM().selection[0];
fl.trace(theShape.numCubicSegments);
```

## shape.vertices

可用性  
Flash MX 2004.

用法  
`shape.vertices`

描述  
只读属性；Vertex 对象的数组（请参阅 [Vertex 对象](#)）。

示例  
下面的示例将所选第一个对象存储在 `someShape` 变量中，然后在“输出”面板中显示该对象的顶点个数：

```
var someShape = fl.getDocumentDOM().selection[0];
fl.trace("The shape has " + someShape.vertices.length + " vertices.");
```

## 第 37 章 : SoundItem 对象

继承关系 [Item 对象](#) > SoundItem 对象

可用性

Flash MX 2004。

描述

SoundItem 对象是 Item 对象的子类。它表示一个用于创建声音的库项目。另请参见 [frame.soundLibraryItem](#) 和 [Item](#) 对象。

方法摘要

除 [Item](#) 对象的方法外，SoundItem 对象还具有以下方法：

属性	描述
<a href="#">soundItem.exportToFile()</a>	在 Macintosh 上将指定项目导出为 QuickTime 文件，或在 Windows 上导出为 WAV 或 QT 文件。

属性摘要

除 [Item](#) 对象的属性外，SoundItem 对象还具有以下属性：

属性	描述
<a href="#">soundItem.bitRate</a>	一个字符串，它指定库中声音的比特率。仅适用于 MP3 压缩类型。
<a href="#">soundItem.bits</a>	一个字符串，它指定库中采用 ADPCM 压缩类型的声音的比特值。
<a href="#">soundItem.compressionType</a>	一个字符串，它指定库中声音的压缩类型。
<a href="#">soundItem.convertStereoToMono</a>	一个布尔值，仅适用于 MP3 和 Raw 压缩类型。
<a href="#">soundItem.fileLastModifiedDate</a>	只读；一个字符串，其中包含一个十六进制数字，此数字表示从 1970 年 1 月 1 日至原始文件导入库时的文件修改日期（在磁盘上）之间的秒数。
<a href="#">soundItem.originalCompressionType</a>	只读；一个字符串，它指定所指定项目是否是以 MP3 文件格式导入的。
<a href="#">soundItem.quality</a>	一个字符串，它指定库中声音的播放品质。仅适用于 MP3 压缩类型。
<a href="#">soundItem.sampleRate</a>	一个字符串，它指定音频剪辑的采样率。
<a href="#">soundItem.sourceFileExists</a>	只读；一个布尔值，它指定之前导入库中的文件现在是否仍位于其导入时的源位置。
<a href="#">soundItem.sourceFileIsCurrent</a>	只读；一个布尔值，它指定库项目的文件修改日期是否与其导入时在磁盘上的文件修改日期相同。
<a href="#">soundItem.sourceFilePath</a>	只读；一个字符串，形式为 file:/// URI，表示导入库中的文件的路径和名称。
<a href="#">soundItem.useImportedMP3Quality</a>	一个布尔值；如果为 true，则忽略所有其他属性，使用导入的 MP3 品质。

### soundItem.bitRate

可用性

Flash MX 2004。

#### 用法

```
soundItem.bitRate
```

#### 描述

属性；一个字符串，它指定库中声音的比特率。此属性仅适用于 MP3 压缩类型。可接受的值为 "8 kbps"、"16 kbps"、"20 kbps"、"24 kbps"、"32 kbps"、"48 kbps"、"56 kbps"、"64 kbps"、"80 kbps"、"112 kbps"、"128 kbps" 和 "160 kbps"。以 8 Kbps 或 16 Kbps 导出的立体声音将转换为单声道声音。对于其他压缩类型，该属性为 `undefined`。

如果要为此属性指定值，请将 [soundItem.useImportedMP3Quality](#) 设置为 `false`。

#### 示例

如果指定的库项目采用 MP3 压缩类型，则下面的示例在“输出”面板中显示 `bitRate` 值：

```
alert(fl.getDocumentDOM().library.items[0].bitRate);
```

另请参见

[soundItem.compressionType](#)、[soundItem.convertStereoToMono](#)

## soundItem.bits

#### 可用性

Flash MX 2004。

#### 用法

```
soundItem.bits
```

#### 描述

属性；一个字符串，它指定库中采用 ADPCM 压缩类型的声音的比特值。可接受的值为 "2 bit"、"3 bit"、"4 bit" 和 "5 bit"。

如果要为此属性指定值，请将 [soundItem.useImportedMP3Quality](#) 设置为 `false`。

#### 示例

如果当前所选库项目采用 ADPCM 压缩类型，则下面的示例在“输出”面板中显示比特值：

```
alert(fl.getDocumentDOM().library.items[0].bits);
```

另请参见

[soundItem.compressionType](#)

## soundItem.compressionType

#### 可用性

Flash MX 2004。

#### 用法

```
soundItem.compressionType
```



**SoundItem** 对象

## 描述

属性；一个字符串，它指定库中声音的压缩类型。可接受的值有 "Default"、"ADPCM"、"MP3"、"Raw" 和 "Speech"。

如果要为此属性指定值，请将 [soundItem.useImportedMP3Quality](#) 设置为 `false`。

## 示例

下面的示例将库项目的压缩类型更改为 `Raw`：

```
fl.getDocumentDOM().library.items[0].compressionType = "Raw";
```

下面的示例将所选库项目的压缩类型更改为 `Speech`：

```
fl.getDocumentDOM().library.getSelectedItems().compressionType = "Speech";
```

另请参见

[soundItem.originalCompressionType](#)

## soundItem.convertStereoToMono

可用性

Flash MX 2004。

用法

```
soundItem.convertStereoToMono
```

描述

属性；一个布尔值，它只能用于 `MP3` 和 `Raw` 压缩类型。将此值设置为 `true` 可以将立体声转换为单声道；设置为 `false` 时则保持为立体声。对于 `MP3` 压缩类型，如果 `soundItem.bitRate` 小于 20 Kbps，将忽略此属性，并强制设置为 `true`（请参阅 [soundItem.bitRate](#)）。

如果要为此属性指定值，请将 [soundItem.useImportedMP3Quality](#) 设置为 `false`。

示例

下面的示例将库项目转换为单声道（仅当该项目采用 `MP3` 或 `Raw` 压缩类型时）：

```
fl.getDocumentDOM().library.items[0].convertStereoToMono = true;
```

另请参见

[soundItem.compressionType](#)

## soundItem.exportToFile()

可用性

Flash CS4 Professional.

用法

```
soundItem.exportToFile(fileURI)
```

#### 参数

**fileURI** 一个字符串，表示为 `file:/// URI`，它指定导出的文件的路径和名称。

#### 返回

一个布尔值：如果成功导出文件，则为 `true`；否则为 `false`。

#### 说明

方法；在 **Macintosh** 上将指定项目导出为 **QuickTime** 文件，或在 **Windows** 上导出为 **WAV** 或 **QT** 文件。导出的 **QuickTime** 或 **QT** 文件只包含音频；视频未导出。导出设置基于要导出的项目。

#### 示例

假定库中的第一个项目是声音项目，则下面的代码将该项目导出为 **WAV** 文件：

```
var soundFileURL = "file:///C:/out.wav";
var libItem = fl.getDocumentDOM().library.items[0];
libItem.exportToFile(soundFileURL);
```

## soundItem.fileLastModifiedDate

#### 可用性

Flash CS4 Professional.

#### 用法

```
soundItem.fileLastModifiedDate
```

#### 描述

只读属性：一个字符串，其中包含一个十六进制数字，此数字表示从 1970 年 1 月 1 日至原始文件导入库时的文件修改日期之间的秒数。如果该文件不再存在，则此值为 "00000000"。

#### 示例

假定库中的第一个项目是声音项目，则下面的代码显示一个十六进制数字（如上所述）。

```
var libItem = fl.getDocumentDOM().library.items[0];
fl.trace("Mod date when imported = " + libItem.fileLastModifiedDate);
```

#### 另请参见

[soundItem.sourceFileExists](#)、[soundItem.sourceFileIsCurrent](#)、[soundItem.sourceFilePath](#)、[FLfile.getModificationDate\(\)](#)

## soundItem.originalCompressionType

#### 可用性

Flash CS4 Professional.

#### 用法

```
soundItem.originalCompressionType
```

#### 描述

只读属性：一个字符串，指定是否将指定项目导入为 mp3 文件。此属性的可能值为“RAW”和“MP3”。

#### 示例

假定库中第一个项目是声音项目，如果文件是以 MP3 文件格式导入库中的，则下面的代码显示“MP3”，否则显示“RAW”：

```
var libItem = fl.getDocumentDOM().library.items[0];  
fl.trace("Imported compression type = "+ libItem.originalCompressionType);
```

另请参见

[soundItem.compressionType](#)

## soundItem.quality

#### 可用性

Flash MX 2004。

#### 用法

`soundItem.quality`

#### 描述

属性；一个字符串，它指定库中声音的播放品质。此属性仅适用于 MP3 压缩类型。可接受值为 "Fast"、"Medium" 和 "Best"。

如果要为此属性指定值，请将 [soundItem.useImportedMP3Quality](#) 设置为 `false`。

#### 示例

如果库项目采用 MP3 压缩类型，下面的示例将该项目的播放品质设置为 Best：

```
fl.getDocumentDOM().library.items[0].quality = "Best";
```

另请参见

[soundItem.compressionType](#)

## soundItem.sampleRate

#### 可用性

Flash MX 2004。

#### 用法

`soundItem.sampleRate`

#### 描述

属性；一个字符串，它指定音频剪辑的采样率。此属性仅适用于 ADPCM、Raw 和 Speech 压缩类型。可接受的值为 "5 kHz"、"11 kHz"、"22 kHz" 和 "44 kHz"。

如果要为此属性指定值，请将 [soundItem.useImportedMP3Quality](#) 设置为 `false`。

#### 示例

如果库项目采用 ADPCM、Raw 或 Speech 压缩类型，下面的示例将该项目的采样率设置为 5 kHz:

```
fl.getDocumentDOM().library.items[0].sampleRate = "5 kHz";
```

另请参见

[soundItem.compressionType](#)

## soundItem.sourceFileExists

可用性

Flash CS4 Professional.

用法

```
soundItem.sourceFileExists
```

描述

只读属性：一个布尔值，如果在导入库中的文件的导入之前所在位置仍然存在此文件，则为 **true**；否则为 **false**。

示例

假定库中的第一个项目是声音项目，如果导入库中的文件仍然存在，则下面的代码显示“**true**”。

```
var libItem = fl.getDocumentDOM().library.items[0];  
fl.trace("sourceFileExists = "+ libItem.sourceFileExists);
```

另请参见

[soundItem.sourceFileIsCurrent](#)、[soundItem.sourceFilePath](#)

## soundItem.sourceFileIsCurrent

可用性

Flash CS4 Professional.

用法

```
soundItem.sourceFileIsCurrent
```

描述

只读属性：一个布尔值，如果库项目的文件修改日期与之前导入的文件在磁盘中的修改日期相同，则为 **true**；否则为 **false**。

示例

假定库中的第一个项目是声音项目，如果导入的文件自导入以来未在磁盘中进行过修改，则下面的代码显示“**true**”：

```
var libItem = fl.getDocumentDOM().library.items[0];  
fl.trace("fileIsCurrent = "+ libItem.sourceFileIsCurrent);
```

另请参见

[soundItem.fileLastModifiedDate](#)、[soundItem.sourceFilePath](#)

## soundItem.sourceFilePath

可用性

Flash CS4 Professional.

用法

soundItem.sourceFilePath

描述

只读属性：一个字符串，表示为 **file:///** URI，它表示导入库中的文件的路径和名称。

示例

下面的示例显示库中所有类型为 "sound" 的项目的名称和源文件路径：

```
for (idx in fl.getDocumentDOM().library.items) {
  if (fl.getDocumentDOM().library.items[idx].itemType == "sound") {
    var myItem = fl.getDocumentDOM().library.items[idx];
    fl.trace(myItem.name + " source is " + myItem.sourceFilePath);
  }
}
```

另请参见

[soundItem.sourceFileExists](#)

## soundItem.useImportedMP3Quality

可用性

Flash MX 2004。

用法

soundItem.useImportedMP3Quality

描述

属性；一个布尔值。如果为 **true**，则忽略所有其他属性，而使用导入的 MP3 品质。

示例

下面的示例将库中的一个声音项设置为使用导入的 MP3 品质：

```
fl.getDocumentDOM().library.items[0].useImportedMP3Quality = true;
```

另请参见

[soundItem.compressionType](#)

## 第 38 章 : Stroke 对象

可用性

Flash MX 2004。

描述

**Stroke** 对象包含笔触的所有设置（包括自定义设置）。此对象表示“属性”检查器中包含的信息。将 **Stroke** 对象用于 `document.setCustomStroke()` 方法，可以更改“工具”面板、“属性”检查器和当前所选项目的笔触设置。此外，还可以使用 `document.getCustomStroke()` 方法获取“工具”面板、“属性”检查器或者当前所选项目的笔触设置。

该对象始终具有以下四个属性：`style`、`thickness`、`color` 和 `breakAtCorners`。（在 Flash CS3 中，不推荐使用 `breakAtCorners` 属性，而推荐使用 `stroke.joinType`。）其他属性可以根据 `stroke.style` 属性的值来设置。

属性摘要

**Stroke** 对象具有以下属性：

属性	描述
<code>stroke.breakAtCorners</code>	布尔值，与自定义的“笔触样式”对话框中的“锐化转角”设置相同。
<code>stroke.capType</code>	指定笔触端类型的字符串。
<code>stroke.color</code>	一个字符串、十六进制值或整数，它表示笔触的颜色。
<code>stroke.curve</code>	指定笔触的斑马线类型的字符串。
<code>stroke.dash1</code>	指定虚线的实心部分长度的整数。
<code>stroke.dash2</code>	指定虚线的空心部分长度的整数。
<code>stroke.density</code>	指定斑点线密度的字符串。
<code>stroke.dotSize</code>	指定斑点线的点大小的字符串。
<code>stroke.dotSpace</code>	一个整数，指定点线中点之间的距离。
<code>stroke.hatchThickness</code>	指定斑马线粗细的字符串。
<code>stroke.jiggle</code>	指定斑马线的微动属性的字符串。
<code>stroke.joinType</code>	一个字符串，它指定笔触的联接类型。
<code>stroke.length</code>	指定斑马线长度的字符串。
<code>stroke.miterLimit</code>	一个浮点值，用于指定一个角度，在该角度以上，尖角的尖端将被一个线段截断。
<code>stroke.pattern</code>	指定锯齿状线的模式的字符串。
<code>stroke.rotate</code>	指定斑马线旋转的字符串。
<code>stroke.scaleType</code>	一个字符串，它指定要对笔触应用的缩放类型。
<code>stroke.shapeFill</code>	一个 <b>Fill</b> 对象，它表示笔触的填充设置。
<code>stroke.space</code>	指定斑马线间距的字符串。
<code>stroke.strokeHinting</code>	一个布尔值，它指定是否对笔触设置笔触提示。
<code>stroke.style</code>	描述笔触样式的字符串。

属性	描述
<code>stroke.thickness</code>	指定笔触大小的整数。
<code>stroke.variation</code>	指定斑点线变体的字符串。
<code>stroke.waveHeight</code>	指定锯齿状线的波高的字符串。
<code>stroke.waveLength</code>	指定锯齿状线的波长的字符串。

## stroke.breakAtCorners

可用性

Flash MX 2004。在 Flash CS3 中不推荐使用，而推荐使用 `stroke.joinType`。

用法

```
stroke.breakAtCorners
```

描述

属性；一个布尔值。此属性与自定义的“笔触样式”对话框中的“锐化转角”设置相同。

示例

下面的示例将 `breakAtCorners` 属性设置为 `true`：

```
var myStroke = fl.getDocumentDOM().getCustomStroke();  
myStroke.breakAtCorners = true;  
fl.getDocumentDOM().setCustomStroke(myStroke);
```

## stroke.capType

可用性

Flash 8。

用法

```
stroke.capType
```

描述

属性；指定笔触端类型的字符串。可接受的值为 `"none"`、`"round"` 和 `"square"`。

示例

下面的示例将笔触端类型设置为 `round`：

```
var myStroke = fl.getDocumentDOM().getCustomStroke();  
myStroke.capType = "round";  
fl.getDocumentDOM().setCustomStroke(myStroke);
```

## stroke.color

### 可用性

Flash MX 2004。在 Flash 8 及更高版本中，不推荐使用此属性，而推荐使用 `stroke.shapeFill.color`。

### 用法

`stroke.color`

### 描述

属性；笔触的颜色，使用以下格式之一：

- 格式为 "#RRGGBB" 或 "#RRGGBBAA" 的字符串
- 格式为 0xRRGGBB 的十六进制数字
- 表示与十六进制数字等价的十进制整数

### 示例

下面的示例设置笔触颜色：

```
var myStroke = fl.getDocumentDOM().getCustomStroke();  
myStroke.color = "#000000";  
fl.getDocumentDOM().setCustomStroke(myStroke);
```

另请参见

[stroke.shapeFill](#)

## stroke.curve

### 可用性

Flash MX 2004。

### 用法

`stroke.curve`

### 描述

属性；一个字符串，指定笔触的斑马线类型。只有在 `stroke.style` 属性设置为 "hatched" 时，才能设置此属性（请参阅 [stroke.style](#)）。可接受的值为 "straight"、"slight curve"、"medium curve" 和 "very curved"。

### 示例

下面的示例为具有 `hatched` 笔触样式设置曲线属性以及其他属性：

```
var myStroke = fl.getDocumentDOM().getCustomStroke();  
myStroke.style = "hatched";  
myStroke.curve = "straight";  
myStroke.space = "close";  
myStroke.jiggle = "wild";  
myStroke.rotate = "free";  
myStroke.length = "slight";  
myStroke.hatchThickness = "thin";  
fl.getDocumentDOM().setCustomStroke(myStroke);
```



## stroke.dash1

可用性

Flash MX 2004。

用法

```
stroke.dash1
```

描述

属性；指定虚线的实心部分长度的整数。仅当 `stroke.style` 属性设置为 `dashed` 时，此属性才可用（请参阅 [stroke.style](#)）。

示例

下面的示例为 `dashed` 样式的笔触设置 `dash1` 和 `dash2` 属性：

```
var myStroke = fl.getDocumentDOM().getCustomStroke();  
myStroke.style = "dashed";  
myStroke.dash1 = 1;  
myStroke.dash2 = 2;  
fl.getDocumentDOM().setCustomStroke(myStroke);
```

## stroke.dash2

可用性

Flash MX 2004。

用法

```
stroke.dash2
```

描述

属性；指定虚线的空白部分长度的整数。仅当 `stroke.style` 属性设置为 `dashed` 时，此属性才可用（请参阅 [stroke.style](#)）。

示例

请参阅 [stroke.dash1](#)。

## stroke.density

可用性

Flash MX 2004。

用法

```
stroke.density
```

描述

属性；指定斑点线密度的字符串。仅当 `stroke.style` 属性设置为 `stipple` 时，此属性才可用（请参阅 [stroke.style](#)）。可接受的值为 `"very dense"`、`"dense"`、`"sparse"` 和 `"very sparse"`。

## 示例

下面的示例将 `stipple` 笔触样式的 `density` 属性设置为 `sparse`：

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.style = "stipple";
myStroke.dotSpace= 3;
myStroke.variation = "random sizes";
myStroke.density = "sparse";
fl.getDocumentDOM().setCustomStroke(myStroke);
```

## stroke.dotSize

## 可用性

Flash MX 2004。

## 用法

`stroke.dotSize`

## 描述

属性；指定斑点线的点大小的字符串。仅当 `stroke.style` 属性设置为 `stipple` 时，此属性才可用（请参阅 [stroke.style](#)）。可接受的值为 `tiny`、`small`、`medium` 和 `large`。

下面的示例将 `stipple` 笔触样式的 `dotSize` 属性设置为 `tiny`：

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.style = "stipple";
myStroke.dotSpace= 3;
myStroke.dotsize = "tiny";
myStroke.variation = "random sizes";
myStroke.density = "sparse";
fl.getDocumentDOM().setCustomStroke(myStroke);
```

## stroke.dotSpace

## 可用性

Flash MX 2004。

## 用法

`stroke.dotSpace`

## 描述

属性；指定斑点线中点与点之间的距离的整数。仅当 `stroke.style` 属性设置为 `dotted` 时，此属性才可用。请参阅 [stroke.style](#)。

## 示例

下面的示例将 `dotted` 的笔触样式的 `dotSpace` 属性设置为 3：

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.style = "dotted";
myStroke.dotSpace= 3;
fl.getDocumentDOM().setCustomStroke(myStroke);
```

## stroke.hatchThickness

可用性

Flash MX 2004。

用法

```
stroke.hatchThickness
```

描述

属性；指定斑马线粗细的字符串。仅当 `stroke.style` 属性设置为 `hatched` 时，此属性才可用（请参阅 [stroke.style](#)）。可接受的值为 `"hairline"`、`"thin"`、`"medium"` 和 `"thick"`。

示例

下面的示例将 `hatched` 笔触样式的 `hatchThickness` 属性设置为 `thin`：

```
var myStroke = fl.getDocumentDOM().getCustomStroke();  
myStroke.style = "hatched";  
myStroke.curve = "straight";  
myStroke.space = "close";  
myStroke.jiggle = "wild";  
myStroke.rotate = "free";  
myStroke.length = "slight";  
myStroke.hatchThickness = "thin";  
fl.getDocumentDOM().setCustomStroke(myStroke);
```

## stroke.jiggle

可用性

Flash MX 2004。

用法

```
stroke.jiggle
```

描述

属性；指定斑马线的微动属性的字符串。仅当 `stroke.style` 属性设置为 `hatched` 时，此属性才可用（请参阅 [stroke.style](#)）。可接受的值为 `"none"`、`"bounce"`、`"loose"` 和 `"wild"`。

示例

下面的示例将 `hatched` 笔触样式的 `jiggle` 属性设置为 `wild`：

```
var myStroke = fl.getDocumentDOM().getCustomStroke();  
myStroke.style = "hatched";  
myStroke.curve = "straight";  
myStroke.space = "close";  
myStroke.jiggle = "wild";  
myStroke.rotate = "free";  
myStroke.length = "slight";  
myStroke.hatchThickness = "thin";  
fl.getDocumentDOM().setCustomStroke(myStroke);
```

## stroke.joinType

可用性

Flash 8。

用法

```
stroke.joinType
```

描述

属性；一个字符串，它指定笔触联接的类型。可接受值为 "miter"、"round" 和 "bevel"。

另请参见

[stroke.capType](#)

## stroke.length

可用性

Flash MX 2004。

用法

```
stroke.length
```

描述

属性；指定斑马线长度的字符串。仅当 `stroke.style` 属性设置为 `hatched` 时，此属性才可用（请参阅 [stroke.style](#)）。可接受的值为 "equal"、"slight variation"、"medium variation" 和 "random"。（值 "random" 实际映射到 "medium variation"。）

示例

下面的示例将 `hatched` 笔触样式的 `length` 属性设置为 `slight`：

```
var myStroke = fl.getDocumentDOM().getCustomStroke();  
myStroke.style = "hatched";  
myStroke.curve = "straight";  
myStroke.space = "close";  
myStroke.jiggle = "wild";  
myStroke.rotate = "free";  
myStroke.length = "slight variation";  
myStroke.hatchThickness = "thin";  
fl.getDocumentDOM().setCustomStroke(myStroke);
```

## stroke.miterLimit

可用性

Flash 8。

用法

```
stroke.miterLimit
```

#### 描述

属性；指定角度的浮点值，在该角度以上，尖角的尖端将被一个线段截断。这意味着只有在尖角大于 `miterLimit` 的值的条件下，才会截断尖角。

#### 示例

下面的示例将笔触设置的尖角限制更改为 3。如果尖角大于 3，则会截断尖角。

```
var myStroke = fl.getDocumentDOM().getCustomStroke();  
myStroke.miterLimit = 3;  
var myStroke = fl.getDocumentDOM().setCustomStroke();
```

## stroke.pattern

#### 可用性

Flash MX 2004。

#### 用法

`stroke.pattern`

#### 描述

属性；指定锯齿状线的模式的字符串。仅当 `stroke.style` 属性设置为 `ragged` 时，此属性才可用（请参阅 [stroke.style](#)）。可接受的值为 `"solid"`、`"simple"`、`"random"`、`"dotted"`、`"random dotted"`、`"triple dotted"` 和 `"random triple dotted"`。

#### 示例

下面的示例将 `ragged` 笔触样式的 `pattern` 属性设置为 `random`：

```
var myStroke = fl.getDocumentDOM().getCustomStroke();  
myStroke.style = "ragged";  
myStroke.pattern = "random";  
fl.getDocumentDOM().setCustomStroke(myStroke);
```

## stroke.rotate

#### 可用性

Flash MX 2004。

#### 用法

`stroke.rotate`

#### 描述

属性；指定斑马线旋转的字符串。仅当 `stroke.style` 属性设置为 `hatched` 时，此属性才可用（请参阅 [stroke.style](#)）。可接受的值为 `"none"`、`"slight"`、`"medium"` 和 `"free"`。

#### 示例

下面的示例将 `hatched` 笔触样式的 `rotate` 属性设置为 `free`：

**Stroke 对象**

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.style = "hatched";
myStroke.curve = "straight";
myStroke.space = "close";
myStroke.jiggle = "wild";
myStroke.rotate = "free";
myStroke.length = "slight";
myStroke.hatchThickness = "thin";
```

## stroke.scaleType

可用性

Flash 8。

用法

```
stroke.scaleType
```

描述

属性；一个字符串，它指定要对笔触应用的缩放类型。可接受值为 "normal"、"horizontal"、"vertical" 和 "none"。

示例

下面的示例将笔触的缩放类型设置为 **horizontal**：

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.scaleType = "horizontal";
fl.getDocumentDOM().setCustomStroke(myStroke);
```

## stroke.shapeFill

可用性

Flash 8。

用法

```
stroke.shapeFill
```

描述

属性；表示笔触的填充设置的 [Fill 对象](#)。

示例

下面的示例指定填充设置，然后将这些设置应用到笔触：

```
var fill = fl.getDocumentDOM().getCustomFill();
fill.linearGradient = true;
fill.colorArray = [ 00ff00, ff0000, fffff ];
var stroke = fl.getDocumentDOM().getCustomStroke();
stroke.shapeFill = fill;
fl.getDocumentDOM().setCustomStroke(stroke);
```

## stroke.space

可用性

Flash MX 2004。

用法

`stroke.space`

描述

属性；指定斑马线间距的字符串。仅当 `stroke.style` 属性设置为 `hatched` 时，此属性才可用（请参阅 [stroke.style](#)）。可接受的值为 "very close"、"close"、"distant" 和 "very distant"。

示例

下面的示例将 `hatched` 笔触样式的 `space` 属性设置为 `close`：

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.style = "hatched";
myStroke.curve = "straight";
myStroke.space = "close";
myStroke.jiggle = "wild";
myStroke.rotate = "free";
myStroke.length = "slight";
myStroke.hatchThickness = "thin";
fl.getDocumentDOM().setCustomStroke(myStroke);
```

## stroke.strokeHinting

可用性

Flash 8。

用法

`stroke.strokeHinting`

描述

属性；指定是否对笔触设置了笔触提示的布尔值。

示例

下面的示例为笔触启用笔触提示：

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.strokeHinting = true;
fl.getDocumentDOM().setCustomStroke(myStroke);
```

## stroke.style

可用性

Flash MX 2004。

**Stroke 对象**

## 用法

`stroke.style`

## 描述

属性；描述笔触样式的字符串。可接受的值为 "noStroke"、"solid"、"dashed"、"dotted"、"ragged"、"stipple" 和 "hatched"。其中一些值需要设置 **Stroke** 对象的附加属性，如以下列表所示：

- 如果值为 "solid" 或 "noStroke"，则没有其他属性。
- 如果值为 `dashed`，则有两个附加属性：`dash1` 和 `dash2`。
- 如果值为 `dotted`，则有一个附加属性：`dotSpace`。
- 如果值为 `ragged`，则有三个附加属性：`pattern`、`waveHeight` 和 `waveLength`。
- 如果值为 `stipple`，则有三个附加属性：`dotSize`、`variation` 和 `density`。
- 如果值为 `hatched`，则有六个附加属性：`hatchThickness`、`space`、`jiggle`、`rotate`、`curve` 和 `length`。

## 示例

下面的示例将笔触样式设置为 `ragged`：

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.style = "ragged";
fl.getDocumentDOM().setCustomStroke(myStroke);
```

## stroke.thickness

## 可用性

Flash MX 2004。

## 用法

`stroke.thickness`

## 描述

属性；指定笔触大小的整数。

## 示例

下面的示例将笔触的 `thickness` 属性设置为值 2：

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.thickness = 2;
fl.getDocumentDOM().setCustomStroke(myStroke);
```

## stroke.variation

## 可用性

Flash MX 2004。

## 用法

`stroke.variation`



### 描述

属性；指定斑点线变体的字符串。仅当 `stroke.style` 属性设置为 `stipple` 时，此属性才可用（请参阅 [stroke.style](#)）。可接受的值为 `"one size"`、`"small variation"`、`"varied sizes"` 和 `"random sizes"`。

### 示例

下面的示例将 `stipple` 笔触样式的 `variation` 属性设置为 `random sizes`：

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.style = "stipple";
myStroke.dotSpace = 3;
myStroke.variation = "random sizes";
myStroke.density = "sparse";
fl.getDocumentDOM().setCustomStroke(myStroke);
```

## stroke.waveHeight

### 可用性

Flash MX 2004。

### 用法

```
stroke.waveHeight
```

### 描述

属性；指定锯齿状线的波高的字符串。仅当 `stroke.style` 属性设置为 `ragged` 时，此属性才可用（请参阅 [stroke.style](#)）。可接受的值为 `"flat"`、`"wavy"`、`"very wavy"` 和 `"wild"`。

### 示例

下面的示例将 `ragged` 笔触样式的 `waveHeight` 属性设置为 `flat`：

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.style = "ragged";
myStroke.pattern = "random";
myStroke.waveHeight = "flat";
myStroke.waveLength = "short";
fl.getDocumentDOM().setCustomStroke(myStroke);
```

## stroke.waveLength

### 可用性

Flash MX 2004。

### 用法

```
stroke.waveLength
```

### 描述

属性；一个字符串，它指定锯齿状线的 `wavelength`。仅当 `stroke.style` 属性设置为 `ragged` 时，此属性才可用（请参阅 [stroke.style](#)）。可接受的值为 `"very short"`、`"short"`、`"medium"` 和 `"long"`。

#### 示例

下面的示例将 ragged 笔触样式的 waveLength 属性设置为 short:

```
var myStroke = fl.getDocumentDOM().getCustomStroke();  
myStroke.style = "ragged";  
myStroke.pattern = "random";  
myStroke.waveHeight = "flat";  
myStroke.waveLength = "short";  
fl.getDocumentDOM().setCustomStroke(myStroke);
```

## 第 39 章 : swfPanel 对象

可用性

Flash CS4 Professional.

描述

`swfPanel` 对象表示窗口 SWF 面板。窗口 SWF 面板是一些 SWF 文件，它们实现的应用程序可从 Flash 创作环境中运行；这些面板可从“窗口”>“其他面板”菜单中访问。默认情况下，窗口 SWF 面板存储在 `Configuration` 文件夹的子文件夹中（请参阅第 2 页的“保存 JSFL 文件”）。例如，在 Windows XP 中，该文件夹位于引导驱动器 \Documents and Settings\用户\Local Settings\Application Data\Adobe\Flash CS4\语言\Configuration\WindowSWF 中。我们提供了窗口 SWF 面板的范例；请参阅第 12 页的““转换位图为矢量图”面板范例”。已注册窗口 SWF 面板的数组存储在 `fl.swfPanels` 属性中。

方法摘要

可以对 `swfPanel` 对象使用以下方法：

方法	描述
<code>swfPanel.call()</code>	与 <code>ActionScript ExternalInterface.addCallback()</code> 和 <code>MMExecute()</code> 方法结合使用，供从创作环境中与 SWF 面板通信。
第 382 页的“ <code>swf-Panel.setFocus()</code> ”	将键盘焦点设置到指定的 SWF 面板。

属性摘要

可以对 `swfPanel` 对象使用以下属性：

属性	描述
<code>swfPanel.name</code>	只读；一个字符串，它表示所指定窗口 SWF 面板的名称。
<code>swfPanel.path</code>	只读；一个字符串，它表示所指定窗口 SWF 面板中使用的 SWF 文件的路径。

### `swfPanel.call()`

可用性

Flash CS4 Professional.

用法

```
swfPanel.call(request)
```

参数

`request` 要传递给函数的参数（请参阅下面的“描述”和“示例”）。

返回

`null` 或由函数调用返回的字符串。函数结果可能是空字符串。

说明

方法；与 `ActionScript ExternalInterface.addCallback()` 和 `MMExecute()` 方法结合使用，供从创作环境中与 SWF 面板通信。

### 示例

下面的示例演示如何使用 **ActionScript** 和 **JavaScript** 代码来创建窗口 **SWF** 面板，并从创作环境中与其通信。

- 1 创建 **ActionScript 3.0 FLA** 文件，将其颜色设置为中度灰，大小设置为宽 400 像素、高 250 像素。
- 2 在舞台中间放置一个动态文本框，将其实例名称设置为 **myTextField**，并在文本框中键入单词“**Status**”。
- 3 如下设置其他文本框属性：
  - 居中对齐
  - 宽 355 像素，高 46 像素
  - **Times New Roman** 字体，28 点，红色

- 4 添加以下 **ActionScript** 代码：

```
// Here's the callback function to be called from JSAPI
function callMeFromJavascript(arg:String):void
{
    try {
        var name:String = String(arg);
        myTextField.text = name;
    } catch (e:Error) {
    }
}

// Expose the callback function as "callMySWF"
ExternalInterface.addCallback("callMySWF", callMeFromJavascript);

// run the JSAPI to wire up the callback
MMExecute("fl.runScript( fl.configURI + \"WindowSWF/fileOp.jsfl\" );");

MMExecute("fl.trace(\"AS3 File Status Panel Initialized\");");
```

- 5 将文件另存为 **fileStatus fla**，利用默认发布设置发布 **SWF** 文件。
- 6 关闭 **Flash**。
- 7 将 **fileStatus.swf** 文件复制到 **WindowSWF** 文件夹，这是 **Configuration** 文件夹的子文件夹（请参阅第 2 页的“[保存 JSFL 文件](#)”）。例如，在 **Windows XP** 中，该文件夹位于引导驱动器 **\Documents and Settings\ 用户 \Local Settings\Application Data\Adobe\Flash CS4\ 语言 \Configuration\WindowSWF** 中。
- 8 启动 **Flash**。
- 9 使用以下代码创建 **JSFL** 文件：

```
function callMyPanel(panelName, arg)
{
    if(fl.swfPanels.length > 0){
        for(x = 0; x < fl.swfPanels.length; x++){
            // look for a SWF panel of the specified name, then call the specified AS3 function
            // in this example, the panel is named "test" and the AS3 callback is "callMySWF"
            if(fl.swfPanels[x].name == panelName) // name busted?
            {
                fl.swfPanels[x].call("callMySWF",arg);
                break;
            }
        }
    }
    else
        fl.trace("no panels");
}

// define the various handlers for events
documentClosedHandler = function () { callMyPanel("fileStatus", "Document Closed");};
fl.addEventListener("documentClosed", documentClosedHandler );

var dater = "New Document";
documentNewHandler = function () { callMyPanel("fileStatus", dater );};
fl.addEventListener("documentNew", documentNewHandler );

documentOpenedHandler = function () { callMyPanel("fileStatus", "Document Opened");};
fl.addEventListener("documentOpened", documentOpenedHandler );
```

10 将 JSFL 文件保存到 SWF 文件所在目录中，命名为 fileOp.jsfl。

11 选择“窗口”>“其他面板”>“fileStatus”。

现在，当创建、打开和关闭 FLA 文件时，fileStatus 面板会显示相应信息，指示已执行的操作。

## swfPanel.name

可用性

Flash CS4 Professional.

用法

swfPanel.name

描述

只读属性：一个字符串，它表示指定的窗口 SWF 面板的名称。

示例

下面的代码在“输出”面板中显示第一个注册窗口 SWF 面板的名称：

```
fl.trace(fl.swfPanels[0].name);
```

另请参见

[swfPanel.path](#)、[fl.swfPanels](#)

## swfPanel.path

可用性

Flash CS4 Professional.

用法

```
swfPanel.path
```

描述

只读属性：一个字符串，它表示在指定的窗口 SWF 面板中使用的 SWF 文件的路径。

示例

下面的代码在“输出”面板中显示在第一个注册窗口 SWF 面板使用的 SWF 文件的路径：

```
fl.trace(fl.swfPanels[0].path);
```

另请参见

[swfPanel.name](#)、[fl.swfPanels](#)

## swfPanel.setFocus()

可用性

Flash CS5.5 Professional。

用法

```
swfPanel.setFocus()
```

描述

方法：将键盘焦点设置到指定的 SWF 面板。

示例

下面的代码将焦点设置到名为“Project”的 SWF 面板：

请先执行以下步骤，然后再运行此命令：

- 1 取消停放“Project”面板，它即变成浮动面板。
- 2 打开“Project”面板中的“Create File”对话框，然后单击“Stage”。
- 3 按几次 Tab 键，以确保“Project”面板不具有焦点。
- 4 从“命令”菜单运行下面的脚本（将一个包含下列代码的 JSFL 文件置于“user/config/Commands”目录下）：
- 5 按 Tab 键。在“Create File”对话框中的其中一个文本字段中，应该可以看到一个插入光标。

```
flash.getSwfPanel("Project").setFocus();
```

另请参见

[swfPanel.name](#)、[fl.swfPanels](#)

## 第 40 章 : SymbolInstance 对象

继承关系 [Element 对象](#) > [Instance 对象](#) > SymbolInstance 对象

可用性

Flash MX 2004。

描述

SymbolInstance 是 Instance 对象的子类，它表示帧中的元件（请参阅 [Instance 对象](#)）。

属性摘要

除 Instance 对象的属性外，SymbolInstance 对象还具有以下属性：

属性	描述
<a href="#">symbolInstance.accName</a>	一个字符串，它等效于“辅助功能”面板中的“名称”字段。
<a href="#">symbolInstance.actionScript</a>	一个字符串，它指定分配给元件的动作。
<a href="#">symbolInstance.backgroundColor</a>	一个字符串，用于指定选择“不透明”模式时的杂边颜色。
<a href="#">symbolInstance.bitmapRenderMode</a>	一个字符串，用于指定元件实例的显示类型。
<a href="#">symbolInstance.blendMode</a>	一个字符串，它指定要应用到影片剪辑元件上的混合模式。
<a href="#">symbolInstance.buttonTracking</a>	一个字符串，仅用于按钮元件，它与“属性”检查器中“音轨作为按钮”或“音轨作为菜单项”的弹出菜单设置的属性相同。
<a href="#">symbolInstance.cacheAsBitmap</a>	一个布尔值，它指定是否要启用运行时位图缓存。
<a href="#">symbolInstance.colorAlphaAmount</a>	一个整数，它是实例的颜色变形的一部分，指定“高级效果”Alpha 设置；等效于在“属性”检查器中使用“颜色”>“高级”设置并调整对话框右侧的控件。
<a href="#">symbolInstance.colorAlphaPercent</a>	一个整数，它指定实例的颜色转换的一部分；等效于在实例“属性”检查器中使用“颜色”>“高级”设置（对话框左侧的百分比控件）。
<a href="#">symbolInstance.colorBlueAmount</a>	一个整数，它是实例的颜色转换的一部分；等效于在实例“属性”检查器中使用“颜色”>“高级”设置。
<a href="#">symbolInstance.colorBluePercent</a>	一个整数，它是实例的颜色转换的一部分；等效于在实例“属性”检查器中使用“颜色”>“高级”设置（对话框左侧的百分比控件）。
<a href="#">symbolInstance.colorGreenAmount</a>	一个整数，它是实例的颜色转换的一部分；等效于在实例“属性”检查器中使用“颜色”>“高级”设置。允许的值为 -255 到 255。
<a href="#">symbolInstance.colorGreenPercent</a>	实例的颜色转换的一部分；等效于在实例“属性”检查器中使用“颜色”>“高级”设置（对话框左侧的百分比控件）。
<a href="#">symbolInstance.colorMode</a>	一个字符串，它按照元件“属性”检查器“颜色”弹出菜单中的标识指定颜色模式。
<a href="#">symbolInstance.colorRedAmount</a>	一个整数，它是实例的颜色转换的一部分；等效于在实例“属性”检查器中使用“颜色”>“高级”设置。
<a href="#">symbolInstance.colorRedPercent</a>	实例的颜色转换的一部分；等效于在实例“属性”检查器中使用“颜色”>“高级”设置（对话框左侧的百分比控件）。
<a href="#">symbolInstance.description</a>	一个字符串，它等效于“辅助功能”面板中的“描述”字段。

属性	描述
<code>symbolInstance.filters</code>	Filter 对象的数组（请参阅 <a href="#">Filter 对象</a> ）。
<code>symbolInstance.firstFrame</code>	一个从零开始的整数，它指定要出现在图形的时间轴中的第一帧。
<code>symbolInstance.forceSimple</code>	一个布尔值，它允许或禁止访问对象的子对象；等效于“辅助功能”面板中的“使子对象可访问”设置的反逻辑。
<code>symbolInstance.loop</code>	一个字符串，对于图形元件，它与“属性”检查器中“循环”弹出菜单设置相同的属性。
<code>symbolInstance.shortcut</code>	一个字符串，它等效于与元件关联的快捷键；等效于“辅助功能”面板中的“快捷键”字段。
<code>symbolInstance.silent</code>	一个布尔值，允许或禁止访问对象；等效于“辅助功能”面板中的“使对象可访问”设置的反逻辑。
<code>symbolInstance.symbolType</code>	一个字符串，它指定元件的类型；等效于“创建新元件”和“转换为元件”对话框中“行为”的值。
<code>symbolInstance.tabIndex</code>	一个整数，等效于“辅助功能”面板中的“Tab 键索引”字段。
第 394 页的“ <code>symbolInstance.usesBackgroundColor</code> ”	一个布尔值，用于指定显示格式。
第 394 页的“ <code>symbolInstance.visible</code> ”	一个布尔值，用于指定该实例是否可见。

## symbolInstance.accName

可用性

Flash MX 2004。

用法

```
symbolInstance.accName
```

描述

属性；一个字符串，它等效于“辅助功能”面板中的“名称”字段。屏幕读取器通过大声读出该名称来标识对象。此属性不适用于图形元件。

示例

下面的示例将“辅助功能”面板中对象名称的值存储在 `theName` 变量中：

```
var theName = fl.getDocumentDOM().selection[0].accName;
```

下面的示例将对象在“辅助功能”面板中名称的值设置为 `Home Button`：

```
fl.getDocumentDOM().selection[0].accName = "Home Button";
```

## symbolInstance.actionScript

可用性

Flash MX 2004。



#### 用法

```
symbolInstance.actionScript
```

#### 描述

属性；一个字符串，它指定分配给元件的动作。此属性仅适用于影片剪辑和按钮实例。对于图形元件实例，该值返回 undefined。

#### 示例

下面的示例将一个 onClipEvent 动作分配给时间轴第一个图层第一帧中的第一项：

```
f1.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].actionScript  
= "onClipEvent(enterFrame) {trace('movie clip enterFrame')}";
```

## symbolInstance.backgroundColor

#### 可用性

Flash CS5.5 Professional。

#### 用法

```
symbolInstance.backgroundColor
```

#### 描述

属性；一个字符串，用于指定为实例选择 24 位模式时的杂边颜色。这是一个十六进制 #rrggbb 格式的字符串或者包含相应值的整数。

#### 示例

下面的示例为元件实例分配黑色的背景颜色：

```
var bitmapInstance = f1.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0];  
bitmapInstance.backgroundColor = "#000000";
```

## symbolInstance.bitmapRenderMode

#### 可用性

Flash CS5.5 Professional。

#### 用法

```
symbolInstance.bitmapRenderMode
```

#### 描述

属性；一个字符串，用于设置元件的显示类型。

可接受的值包括：

- “none”
- “cache”- 设置 Flash Player 在运行时将该元件缓存为位图。
- “export”- 设置在编译 SWF 时将该元件导出为位图。

以前使用的第 387 页的 `symbolInstance.cacheAsBitmap` 属性与此属性类似，但提供的选项较少，因为该属性是一个布尔类型的属性。将来可能会弃用 `cacheAsBitmap` 属性，因此用户应改为使用这一新属性。布尔类型的 `cacheAsBitmap` 属性中的 `true/false` 选项等效于这一新属性的“`cache`”/“`none`”值。

#### 示例

下面的示例将元件的 `bitmapRenderMode` 指定为“`export`”：

```
var symbol = fl.getDocumentDOM().selection[0];
fl.trace(symbol.bitmapRenderMode);
symbol.bitmapRenderMode = "export";
```

## symbolInstance.blendMode

#### 可用性

Flash 8。

#### 用法

```
symbolInstance.blendMode
```

#### 描述

属性；一个字符串，它指定要应用到影片剪辑元件的混合模式。可接受值为“`normal`”、“`layer`”、“`multiply`”、“`screen`”、“`overlay`”、“`hardlight`”、“`lighten`”、“`darken`”、“`difference`”、“`add`”、“`subtract`”、“`invert`”、“`alpha`”和“`erase`”。

#### 示例

下面的示例将第一层第一帧中第一个影片剪辑元件的混合模式设置为 `add`：

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].blendMode = "add";
```

#### 另请参见

[document.setBlendMode\(\)](#)

## symbolInstance.buttonTracking

#### 可用性

Flash MX 2004。

#### 用法

```
symbolInstance.buttonTracking
```

#### 描述

属性；一个字符串，仅用于按钮元件，它与“属性”检查器中“音轨作为按钮”或“音轨作为菜单项”的弹出菜单设置相同的属性。对于其他类型的元件，则忽略此属性。可接受值为“`button`”或“`menu`”。

#### 示例

下面的示例将时间轴中第一个图层第一帧的第一个元件设置为“当作菜单项”（只要该元件是按钮）：

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].buttonTracking = "menu";
```

## symbolInstance.cacheAsBitmap

可用性

Flash 8。

用法

```
symbolInstance.cacheAsBitmap
```

描述

属性；一个布尔值，它指定运行时位图缓存是否已启用。

注：从 Flash Professional CS5.5 开始，用户应改为使用第 385 页的“[symbolInstance.bitmapRenderMode](#)”属性，而不使用此属性。

示例

下面的示例为第一图层第一帧中的第一个元素启用运行时位图缓存：

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].cacheAsBitmap = true;
```

## symbolInstance.colorAlphaAmount

可用性

Flash MX 2004。

用法

```
symbolInstance.colorAlphaAmount
```

描述

属性；一个整数，它是实例的颜色变形的一部分，指定“高级效果”Alpha 设置。此属性等效于在“属性”检查器中使用“颜色”>“高级”设置并调整对话框右侧的控件。此值将色调和 Alpha 值减小或增加一个常量。此值就添加到当前值中。用于 [symbolInstance.colorAlphaPercent](#) 时，此属性最有用。允许的值为 -255 到 255。

示例

下面的示例将选定元件实例的 alpha 设置减小 100：

```
fl.getDocumentDOM().selection[0].colorAlphaAmount = -100;
```

## symbolInstance.colorAlphaPercent

可用性

Flash MX 2004。

用法

```
symbolInstance.colorAlphaPercent
```

**SymbolInstance** 对象

## 描述

属性，一个整数，它指定实例的颜色变形的一部分。此属性等效于在实例“属性”检查器中使用“颜色”>“高级”设置（对话框左侧的百分比控件）。此值将色调和 alpha 值更改为指定的百分比。允许的值为 -100 到 100。请参阅 [symbolInstance.colorAlphaAmount](#)。

## 示例

下面的示例将选定元素实例的 `colorAlphaPercent` 设置为 80：

```
fl.getDocumentDOM().selection[0].colorAlphaPercent = 80;
```

## symbolInstance.colorBlueAmount

## 可用性

Flash MX 2004。

## 用法

```
symbolInstance.colorBlueAmount
```

## 描述

属性；一个整数，它是实例的颜色转换的一部分。此属性等效于在实例“属性”检查器中使用“颜色”>“高级”设置。允许的值为 -255 到 255。

## symbolInstance.colorBluePercent

## 可用性

Flash MX 2004。

## 用法

```
symbolInstance.colorBluePercent
```

## 描述

属性；一个整数，它是实例的颜色转换的一部分。此属性等效于在实例“属性”检查器中使用“颜色”>“高级”设置（对话框左侧的百分比控件）。此值将蓝色值设置为指定的百分比。允许的值为 -100 到 100。

## 示例

下面的示例将选定元素实例的 `colorBluePercent` 设置为 80：

```
fl.getDocumentDOM().selection[0].colorBluePercent = 80;
```

## symbolInstance.colorGreenAmount

## 可用性

Flash MX 2004。

#### 用法

```
symbolInstance.colorGreenAmount
```

#### 描述

属性；一个整数，它是实例的颜色转换的一部分。此属性等效于在实例“属性”检查器中使用“颜色”>“高级”设置。允许的值为 -255 到 255。

## symbolInstance.colorGreenPercent

#### 可用性

Flash MX 2004。

#### 用法

```
symbolInstance.colorGreenPercent
```

#### 描述

属性，实例的颜色转换的一部分。此属性等效于在实例“属性”检查器中使用“颜色”>“高级”设置（对话框左侧的百分比控件）。此值将绿色值设置为指定的百分比。允许的值为 -100 到 100。

#### 示例

下面的示例将选定元件实例的 `colorGreenPercent` 设置为 70：

```
fl.getDocumentDOM().selection[0].colorGreenPercent = 70;
```

## symbolInstance.colorMode

#### 可用性

Flash MX 2004。

#### 用法

```
symbolInstance.colorMode
```

#### 描述

属性，一个字符串，它按照元件“属性”检查器“颜色”弹出菜单中的标识指定颜色模式。可接受值为 "none"、"brightness"、"tint"、"alpha" 和 "advanced"。

#### 示例

下面的示例将时间轴中第一个图层第一帧第一个元素的 `colorMode` 属性更改为 `alpha`：

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].colorMode = "alpha";
```

## symbolInstance.colorRedAmount

#### 可用性

Flash MX 2004。

#### 用法

```
symbolInstance.colorRedAmount
```

#### 描述

属性；一个整数，它是实例的颜色转换的一部分。此属性等效于在实例“属性”检查器中使用“颜色”>“高级”设置。允许的值为 -255 到 255。

#### 示例

下面的示例将选定元件实例的 `colorRedAmount` 设置为 255:

```
fl.getDocumentDOM().selection[0].colorRedAmount = 255;
```

## symbolInstance.colorRedPercent

#### 可用性

Flash MX 2004。

#### 用法

```
symbolInstance.colorRedPercent
```

#### 描述

属性，实例的颜色转换的一部分。此属性等效于在实例“属性”检查器中使用“颜色”>“高级”设置（对话框左侧的百分比控件）。此值将红色值设置为指定的百分比。允许的值为 -100 到 100。

#### 示例

下面的示例将选定元件实例的 `colorRedPercent` 设置为 10:

```
fl.getDocumentDOM().selection[0].colorRedPercent = 10;
```

## symbolInstance.description

#### 可用性

Flash MX 2004。

#### 用法

```
symbolInstance.description
```

#### 描述

属性；一个字符串，它等效于“辅助功能”面板中的“描述”字段。该描述由屏幕读取器读出。此属性不适用于图形元件。

#### 示例

下面的示例将“辅助功能”面板中对象描述的值存储在 `theDescription` 变量中:

```
var theDescription = fl.getDocumentDOM().selection[0].description;
```

下面的示例将“辅助功能”面板描述的值设置为 `Click the home button to go to home:`

```
fl.getDocumentDOM().selection[0].description= "Click the home button to go to home";
```

## symbolInstance.filters

可用性

Flash 8。

用法

`symbolInstance.filters`

描述

属性；Filter 对象的数组（请参阅 [Filter 对象](#)）。不能直接写入此数组来修改滤镜属性。而应检索此数组，设置单个属性，然后再设置数组来反映新的属性。

示例

下面的示例将跟踪索引为 0 的滤镜的名称。假设它是一个“发光”滤镜，它的 blurX 属性设置为 100，并且新值被写入到滤镜数组中。

```
var filterName = fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].filters[0].name;
fl.trace(filterName);
var filterArray = fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].filters;
if (filterName == 'glowFilter'){
    filterArray[0].blurX = 100;
}
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].filters = filterArray;
```

## symbolInstance.firstFrame

可用性

Flash MX 2004。

用法

`symbolInstance.firstFrame`

描述

属性；一个从零开始的整数，它指定要出现在图形的时间轴中的第一帧。此属性仅适用于图形元件，它与“属性”检查器中“第一个”字段设置的属性相同。对于其他类型的元件，此属性为 `undefined`。

示例

下面的示例指定“帧 10”应是出现在指定元素的时间轴中的第一帧：

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].firstFrame = 10;
```

## symbolInstance.forceSimple

可用性

Flash MX 2004。

#### 用法

```
symbolInstance.forceSimple
```

#### 描述

属性；一个布尔值，它允许和禁止访问对象的子对象。此属性等效于“辅助功能”面板中的“使子对象可访问”设置的反逻辑。例如，如果 `forceSimple` 为 `true`，则与不选中“使子对象可访问”选项的效果相同。如果 `forceSimple` 为 `false`，则与选中“使子对象可访问”选项的效果相同。

此属性仅可用于 `MovieClip` 对象。

#### 示例

下面的示例检查对象的子对象是否可访问，返回值为 `false` 表示子对象可访问：

```
var areChildrenAccessible = fl.getDocumentDOM().selection[0].forceSimple;
```

下面的示例允许访问对象的子对象：

```
fl.getDocumentDOM().selection[0].forceSimple = false;
```

## symbolInstance.loop

#### 可用性

Flash MX 2004。

#### 用法

```
symbolInstance.loop
```

#### 描述

属性；一个字符串，对于图形元件，它与“属性”检查器中“循环”弹出菜单设置的属性相同。对于其他类型的元件，此属性为 `undefined`。可接受的值为 `"loop"`、`"play once"` 和 `"single frame"`，它们相应地设置图形的动画。

#### 示例

只要时间轴中第一个图层第一帧的第一个元件是图形，下面的示例就将其设置为单帧（显示图形时间轴中的一个指定帧）：

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].loop = 'single frame';
```

## symbolInstance.shortcut

#### 可用性

Flash MX 2004。

#### 用法

```
symbolInstance.shortcut
```

#### 描述

属性；一个字符串，等效于与元件关联的快捷键。此属性等效于“辅助功能”面板中的“快捷键”字段。此键由屏幕阅读器读出。此属性不适用于图形元件。



#### 示例

下面的示例将对象快捷键的值存储在 `theShortcut` 变量中:

```
var theShortcut = fl.getDocumentDOM().selection[0].shortcut;
```

下面的示例将对象的快捷键设置为 `Ctrl+i`:

```
fl.getDocumentDOM().selection[0].shortcut = "Ctrl+i";
```

## symbolInstance.silent

#### 可用性

Flash MX 2004。

#### 用法

```
symbolInstance.silent
```

#### 描述

属性; 一个布尔值, 允许或禁止访问对象。此属性等效于“辅助功能”面板中的“使对象可访问”设置的反逻辑。例如, 如果 `silent` 为 `true`, 则与不选中“使对象可访问”选项的效果相同。如果 `silent` 为 `false`, 则与选中“使对象可访问”选项的效果相同。

此属性不适用于图形对象。

#### 示例

下面的示例检查对象是否可访问, 返回值为 `false`, 表示对象可访问:

```
var isSilent = fl.getDocumentDOM().selection[0].silent;
```

下面的示例将对象设置为可访问:

```
fl.getDocumentDOM().selection[0].silent = false;
```

## symbolInstance.symbolType

#### 可用性

Flash MX 2004。

#### 用法

```
symbolInstance.symbolType
```

#### 描述

属性; 一个字符串, 指定元件的类型。此属性等效于“创建新元件”和“转换为元件”对话框中“行为”的值。可接受值为 `"button"`、`"movie clip"` 和 `"graphic"`。

#### 示例

下面的示例将当前文档的时间轴中第一个图层第一帧的第一个元件设置为具有图形元件的行为:

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].symbolType = "graphic";
```

## symbolInstance.tabIndex

### 可用性

Flash MX 2004。

### 用法

```
symbolInstance.tabIndex
```

### 描述

属性；一个整数，它等效于“辅助功能”面板中的“Tab 键索引”字段。创建一个 Tab 键顺序，当用户按 Tab 键时，按此顺序访问对象。此属性不适用于图形元件。

### 示例

下面的示例将 mySymbol 对象的 tabIndex 属性设置为 3，并将该值显示在“输出”面板中：

```
var mySymbol = fl.getDocumentDOM().selection[0];  
mySymbol.tabIndex = 3;  
fl.trace(mySymbol.tabIndex);
```

## symbolInstance.usesBackgroundColor

### 可用性

Flash CS5.5 Professional。

### 用法

```
symbolInstance.usesBackgroundColor
```

### 描述

属性；一个布尔值，指示是为实例使用 24 位模式 (True) 还是使用 32 位 alpha 模式 (False)。如果为 True，则使用为实例指定的 backgroundColor。

### 示例

下面的示例将实例的 usesBackgroundColor 属性设置为 True：

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].useTransparentBackground = true;
```

## symbolInstance.visible

### 可用性

Flash CS5.5 Professional。

### 用法

```
symbolInstance.visible
```

#### 描述

属性；一个布尔值，用于将对象的 **Visible** 属性设置为 On (True) 或 Off (False)。

#### 示例

下面的示例将第一个图层第一帧的第一个项目的可见性设置为 **False**：

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].visible = false;
```

# 第 41 章 : SymbolItem 对象

继承关系 [Item 对象](#) > [SymbolItem 对象](#)

可用性

Flash MX 2004。

描述

[SymbolItem](#) 对象是 [Item 对象](#) 的子类。

方法摘要

除 [Item](#) 对象的方法外，还可以对 [SymbolItem](#) 对象使用以下方法：

方法	描述
<a href="#">symbolItem.convertToCompiledClip()</a>	将库中的元件项目转换为编译影片剪辑。
<a href="#">symbolItem.exportSWC()</a>	将元件项目导出到 SWC 文件。
<a href="#">symbolItem.exportSWF()</a>	将元件项目导出到 SWF 文件。

属性摘要

除 [Item](#) 对象的属性外，[SymbolItem](#) 对象还具有以下属性：

属性	描述
<a href="#">symbolItem.scalingGrid</a>	一个布尔值，它指定是否为项启用 9 段式缩放。
<a href="#">symbolItem.scalingGridRect</a>	指定四条 9 切片辅助线的位置的 <a href="#">Rectangle</a> 对象。
<a href="#">symbolItem.sourceAutoUpdate</a>	一个布尔值，它指定在发布 FLA 文件时是否更新项目。
<a href="#">symbolItem.sourceFilePath</a>	一个字符串，它将源 FLA 文件的路径指定为 file:/// URI。
<a href="#">symbolItem.sourceLibraryName</a>	一个字符串，它指定源文件库中的项目的名称。
<a href="#">symbolItem.symbolType</a>	一个字符串，指定元件的类型。
<a href="#">symbolItem.timeline</a>	只读；一个 <a href="#">Timeline</a> 对象。

## [symbolItem.convertToCompiledClip\(\)](#)

可用性

Flash MX 2004。

用法

```
symbolItem.convertToCompiledClip()
```

参数

无。

返回  
无。

描述  
方法；将库中的元件项目转换为编译影片剪辑。

示例  
下面的示例将库中的一个项目转换为编译影片剪辑：

```
fl.getDocumentDOM().library.items[3].convertToCompiledClip();
```

## symbolItem.exportSWC()

可用性  
Flash MX 2004。

用法  
symbolItem.exportSWC(outputURI)

参数  
**outputURI** 一个表示为 **file:///** URI 的字符串，它指定该方法要将元件导出到的 SWC 文件。**outputURI** 必须引用本地文件。如果 **outputURI** 不存在，Flash 将不会创建该文件夹。

返回  
无。

描述  
方法；将元件项目导出到 SWC 文件。

示例  
下面的示例将库中的项导出至 **tests** 文件夹中名为 **mySymbol.swc** 的 SWC 文件：

```
fl.getDocumentDOM.library.selectItem("mySymbol");  
var currentSelection = fl.getDocumentDOM().library.getSelectedItems();  
currentSelection[0].exportSWC("file:///Macintosh HD/SWCDirectory/mySymbol.swc");
```

## symbolItem.exportSWF()

可用性  
Flash MX 2004。

用法  
symbolItem.exportSWF(outputURI)

参数  
**outputURI** 一个表示为 **file:///** URI 的字符串，它指定该方法要将元件导出到的 SWF 文件。**outputURI** 必须引用本地文件。如果 **outputURI** 不存在，Flash 不会创建文件夹。

返回  
无。

描述  
方法；将元件项目导出到 SWF 文件。

示例  
下面的示例将库中的项导出至 tests 文件夹中的 my.swf 文件：

```
fl.getDocumentDOM().library.items[0].exportSWF("file:///c:/tests/my.swf");
```

## symbolItem.scalingGrid

可用性  
Flash 8。

用法  
symbolItem.scalingGrid

描述  
属性；一个布尔值，它指定是否为项目启用 9 切片缩放。

示例  
下面的示例为库中的项目启用 9 切片缩放：

```
fl.getDocumentDOM().library.items[0].scalingGrid = true;
```

另请参见  
[symbolItem.scalingGridRect](#)

## symbolItem.scalingGridRect

可用性  
Flash 8。

用法  
symbolItem.scalingGridRect

描述  
属性；一个 Rectangle 对象，它指定四条 9 切片辅助线的位置。有关矩形格式的信息，请参阅 [document.addNewRectangle\(\)](#)。

示例  
下面的示例指定 9 切片辅助线的位置：

```
fl.getDocumentDOM().library.items[0].scalingGridRect = {left:338, top:237, right:3859, bottom:713};
```

另请参见  
[symbolItem.scalingGrid](#)

## symbolItem.sourceAutoUpdate

可用性  
Flash MX 2004。

用法  
`symbolItem.sourceAutoUpdate`

描述  
属性；一个布尔值，它指定在发布 FLA 文件时是否更新项目。默认值为 `false`。用于共享库元件。

示例  
下面的示例设置库项目的 `sourceAutoUpdate` 属性：

```
fl.getDocumentDOM().library.items[0].sourceAutoUpdate = true;
```

## symbolItem.sourceFilePath

可用性  
Flash MX 2004。

用法  
`symbolItem.sourceFilePath`

描述  
属性；一个字符串，它将源 FLA 文件的路径指定为 `file:/// URI`。路径必须是绝对路径而不是相对路径。此属性用于共享库元件。

示例  
下面的示例在“输出”面板中显示 `sourceFilePath` 属性的值：

```
fl.trace(fl.getDocumentDOM().library.items[0].sourceFilePath);
```

## symbolItem.sourceLibraryName

可用性  
Flash MX 2004。

用法  
`symbolItem.sourceLibraryName`

#### 描述

属性；一个字符串，它指定源文件库中的项目的名称。它用于共享库元件。

#### 示例

下面的示例在“输出”面板中显示 `sourceLibraryName` 属性的值：

```
fl.trace(fl.getDocumentDOM().library.items[0].sourceLibraryName);
```

## symbolItem.symbolType

#### 可用性

Flash MX 2004。

#### 用法

```
symbolItem.symbolType
```

#### 描述

属性；一个字符串，指定元件的类型。可接受值为 "movie clip"、"button" 和 "graphic"。

#### 示例

下面的示例显示 `symbolType` 属性的当前值，将其更改为 `button`，然后再次显示：

```
alert(fl.getDocumentDOM().library.items[0].symbolType);  
fl.getDocumentDOM().library.items[0].symbolType = "button";  
alert(fl.getDocumentDOM().library.items[0].symbolType);
```

## symbolItem.timeline

#### 可用性

Flash MX 2004。

#### 用法

```
symbolItem.timeline
```

#### 描述

只读属性；一个 [Timeline](#) 对象。

#### 示例

下面的示例获取并显示库中所选影片剪辑所包含的图层数：

```
var tl = fl.getDocumentDOM().library.getSelectedItems()[0].timeline;  
alert(tl.layerCount);
```



## 第 42 章 : Text 对象

继承关系 [Element 对象](#) > Text 对象

可用性

Flash MX 2004。

描述

**Text** 对象表示文档中单独的文本项。文本的所有属性都与整个文本块有关。

若要设置文本字段中某个文本串的属性，请参阅 [TextAttrs 对象](#) 的“属性”摘要。若要更改文本字段中的所选文本的属性，可以使用 [document.setTextAttr\(\)](#) 并指定文本范围，或使用当前所选文本。

若要设置所选文本字段的一般属性，请使用 [document.setProperty\(\)](#)。下面的示例将所选文本字段的注册点的 x 值设置为 50:

```
fl.getDocumentDOM().setProperty("x", 50);
```

方法摘要

除 **Element** 对象的方法外，**Text** 对象还具有以下方法：

方法	描述
<a href="#">text.getTextAttr()</a>	检索由可选的 <i>startIndex</i> 和 <i>endIndex</i> 参数标识的文本的指定属性。
<a href="#">text.getTextString()</a>	检索指定范围的文本。
<a href="#">text.setTextAttr()</a>	设置与由 <i>startIndex</i> 和 <i>endIndex</i> 标识的文本相关联的指定属性。
<a href="#">text.setTextString()</a>	更改此 <b>Text</b> 对象中的文本字符串。

属性摘要

除 **Element** 对象的属性外，**Text** 对象还具有以下属性：

属性	描述
<a href="#">text.accName</a>	一个字符串，它等效于“辅助功能”面板中的“名称”字段。
<a href="#">text.antiAliasSharpness</a>	一个浮点值，它指定文本消除锯齿的清晰度。
<a href="#">text.antiAliasThickness</a>	一个浮点值，它指定文本消除锯齿的粗细。
<a href="#">text.autoExpand</a>	一个布尔值，它控制静态文本字段的边框宽度的扩展，或者控制动态或输入文本的边框宽度或高度的扩展。
<a href="#">text.border</a>	一个布尔值，它控制 <b>Flash</b> 是显示 ( <b>true</b> ) 还是隐藏 ( <b>false</b> ) 动态或输入文本的边框。
<a href="#">text.description</a>	一个字符串，它等效于“辅助功能”面板中的“描述”字段。
<a href="#">text.embeddedCharacters</a>	一个字符串，它指定要嵌入的字符。它等效于在“字符嵌入”对话框中输入文本。
<a href="#">text.embedRanges</a>	一个字符串，由分隔的整数组成，这些整数与“字符嵌入”对话框中可以选择的项目对应。
<a href="#">text.embedVariantGlyphs</a>	一个布尔值，它指定是否支持嵌入变体字型。
<a href="#">text.fontRenderingMode</a>	一个字符串，它指定文本的呈现模式。
<a href="#">text.length</a>	只读；一个整数，它表示 <b>Text</b> 对象中的字符数。

## Text 对象

属性	描述
<code>text.lineType</code>	一个字符串，它将线条类型设置为 "single line"、"multiline"、"multiline no wrap" 或 "password"。
<code>text.maxCharacters</code>	一个整数，它指定用户可在此 Text 对象中输入的最大字符数。
<code>text.orientation</code>	一个字符串，它指定文本字段的方向。
<code>text.renderAsHTML</code>	一个布尔值，它控制 Flash 是否将文本绘制为 HTML 并解释嵌入的 HTML 标签。
<code>text.scrollable</code>	一个布尔值，它控制文本是 (true) 否 (false) 可以滚动。
<code>text.selectable</code>	一个布尔值，它控制文本是 (true) 否 (false) 可选择。输入文本始终是可选择的。
<code>text.selectionEnd</code>	一个从零开始的整数，它指定文本部分选定的末尾的偏移量。
<code>text.selectionStart</code>	一个从零开始的整数，它指定文本部分选定的开头的偏移量。
<code>text.shortcut</code>	一个字符串，它等效于“辅助功能”面板中的“快捷键”字段。
<code>text.silent</code>	一个布尔值，它指定对象是否可访问。
<code>text.tabIndex</code>	一个整数，等效于“辅助功能”面板中的“Tab 键索引”字段。
<code>text.textRuns</code>	只读；TextRun 对象数组。
<code>text.textType</code>	一个字符串，它指示文本字段的类型。可接受的值为 "static"、"dynamic" 和 "input"。
<code>text.useDeviceFonts</code>	一个布尔值。值 true 会使 Flash 使用设备字体绘制文本。
<code>text.variableName</code>	一个字符串，它包含 Text 对象的内容。

## text.accName

可用性

Flash MX 2004。

用法

```
text.accName
```

描述

属性；一个字符串，它等效于“辅助功能”面板中的“名称”字段。屏幕读取器通过大声读出该名称来标识对象。此属性不能用于动态文本。

示例

下面的示例检索对象的名称：

```
var doc = fl.getDocumentDOM();
var theName = doc.selection[0].accName;
```

下面的示例设置当前选定对象的名称：

```
fl.getDocumentDOM().selection[0].accName = "Home Button";
```

## text.antiAliasSharpness

可用性

Flash 8。

用法

`text.antiAliasSharpness`

描述

属性；一个浮点值，它指定文本消除锯齿的清晰度。此属性控制绘制文本的清晰程度；此值越高，其指定的文本就越清晰。如果值为 0，将指定正常清晰度。仅当 `text.fontRenderingMode` 设置为 `customThicknessSharpness` 时，此属性才可用。

示例

请参阅 [text.fontRenderingMode](#)。

另请参见

[text.antiAliasThickness](#), [text.fontRenderingMode](#)

## text.antiAliasThickness

可用性

Flash 8。

用法

`text.antiAliasThickness`

描述

属性；一个浮点值，它指定文本消除锯齿的粗细。此属性控制绘制文本的粗细，此值越高，其指定的文本就越粗。如果值为 0，将指定正常粗细。仅当 `text.fontRenderingMode` 设置为 `customThicknessSharpness` 时，此属性才可用。

示例

请参阅 [text.fontRenderingMode](#)。

另请参见

[text.antiAliasSharpness](#), [text.fontRenderingMode](#)

## text.autoExpand

可用性

Flash MX 2004。

用法

`text.autoExpand`

**Text 对象****描述**

属性；一个布尔值。对于静态文本字段，值 **true** 会扩展边框宽度以显示所有文本。对于动态或输入文本字段，值 **true** 会扩展边框宽度和高度以显示所有文本。

**示例**

下面的示例将 `autoExpand` 属性设置为值 **true**：

```
fl.getDocumentDOM().selection[0].autoExpand = true;
```

## text.border

**可用性**

Flash MX 2004。

**用法**

```
text.border
```

**描述**

属性；一个布尔值。值为 **true** 会使 **Flash** 在文本周围显示边框。

**示例**

下面的示例将 `border` 属性设置为值 **true**：

```
fl.getDocumentDOM().selection[0].border = true;
```

## text.description

**可用性**

Flash MX 2004。

**用法**

```
text.description
```

**描述**

属性；一个字符串，它等效于“辅助功能”面板中的“描述”字段。该描述由屏幕读取器读出。

**示例**

下面的示例检索对象的说明：

```
var doc = fl.getDocumentDOM();  
var desc = doc.selection[0].description;
```

下面的示例设置对象的描述：

```
var doc = fl.getDocumentDOM();  
doc.selection[0].description= "Enter your name here";
```

## text.embeddedCharacters

可用性

Flash MX 2004。

用法

```
text.embeddedCharacters
```

描述

属性；一个字符串，它指定要嵌入的字符。它等效于在“字符嵌入”对话框中输入文本。

此属性仅对动态或输入文本有效；用于其他文本类型会生成警告。

注：从 Flash Professional CS5 开始，将在文档级别（而不是文本对象级别）控制字体嵌入。使用第 257 页的“[fontItem.embeddedCharacters](#)”属性代替 `text.embeddedCharacters` 属性。

示例

以下示例假定当前所选内容中的第一项或唯一一项是传统文本对象并将 `embeddedCharacters` 属性设置为 `abc`：

```
fl.getDocumentDOM().selection[0].embeddedCharacters = "abc";
```

## text.embedRanges

可用性

Flash MX 2004。

用法

```
text.embedRanges
```

描述

属性；一个字符串，由分隔的整数组成，这些整数与“字符嵌入”对话框中可以选择的项目对应。此属性仅对动态或输入文本有效；用于静态文本时，将忽略此属性。

此属性对应于 `Configuration/Font Embedding` 文件夹中的 XML 文件。

注：从 Flash Professional CS5 开始，将在文档级别（而不是文本对象级别）控制字体嵌入。使用第 257 页的“[fontItem.embedRanges](#)”属性代替 `text.embedRanges` 属性。

示例

以下示例假定当前所选内容中的第一项或唯一一项是传统文本对象并将 `embedRanges` 属性设置为 `"1|3|7"`：

```
var doc = fl.getDocumentDOM();  
doc.selection[0].embedRanges = "1|3|7";
```

下面的示例重置该属性：

```
var doc = fl.getDocumentDOM();  
doc.selection[0].embedRanges = "";
```

## text.embedVariantGlyphs

可用性

Flash CS4 Professional.

用法

```
text.embedVariantGlyphs
```

描述

属性；一个布尔值，它指定是 (**true**) 否 (**false**) 启用变体字型的嵌入。此属性仅对动态或输入文本有效；用于静态文本时，将忽略此属性。默认值为 **false**。

注：从 Flash Professional CS5 开始，将在文档级别（而不是文本对象级别）控制字体嵌入。使用第 258 页的“[fontItem.embedVariantGlyphs](#)”属性代替 `text.embedVariantGlyphs` 属性。在 Flash Professional CS5 中，`text.embedVariantGlyphs` 属性不再起作用，因为 Flash 总是为 TLF 文本嵌入变体字型，而从不为传统文本嵌入这些字型。

示例

下面的示例支持在所选 **Text** 对象中嵌入变体字型：

```
fl.getDocumentDOM().selection[0].embedVariantGlyphs = true;
```

另请参见

[fontItem.embedVariantGlyphs](#)

## text.fontRenderingMode

可用性

Flash 8。

用法

```
text.fontRenderingMode
```

描述

属性；一个字符串，它指定文本的呈现模式。此属性影响文本在舞台上和 **Flash Player** 中的显示方式。下表中描述了可接受的值：

属性值	如何呈现文本
设备	使用设备字体呈现文本。
位图	将带有锯齿的文本呈现为位图或像素字体的样子。
standard	用 Flash MX 2004 使用的标准消除锯齿的方法来呈现文本。这是用于动画文本、大型文本或倾斜文本的最佳设置。
advanced	使用在 Flash 8 中实现的高级消除锯齿字体呈现技术来呈现文本，这样可以获得更好的消除锯齿效果并提高可读性，尤其是对小型文本更是如此。
customThicknessSharpness	用于在使用 Flash 8 中实现的高级消除锯齿字体呈现技术时，指定文本清晰度和粗细的自定义设置。

示例

下面的示例显示如何使用 `customThicknessSharpness` 值来指定文本的清晰度和粗细：

## Text 对象

```
fl.getDocumentDOM().setElementProperty("fontRenderingMode", "customThicknessSharpness");
fl.getDocumentDOM().setElementProperty("antiAliasSharpness", 400);
fl.getDocumentDOM().setElementProperty("antiAliasThickness", -200);
```

另请参见

[text.antiAliasSharpness](#)、[text.antiAliasThickness](#)

## text.getTextAttr()

可用性

Flash MX 2004。

用法

```
text.getTextAttr(attrName [, startIndex [, endIndex]])
```

参数

**attrName** 一个字符串，指定要返回的 **TextAttrs** 对象属性的名称。有关 **attrName** 的可能值的列表，请参阅 **TextAttrs** 对象的“属性”摘要。

**startIndex** 一个整数，为第一个字符的索引。此参数是可选的。

**endIndex** 一个整数，指定文本范围的结尾，文本范围从 **startIndex** 开始，直到（但不包括）**endIndex**。此参数是可选的。

返回

在 **attrName** 参数中指定的属性的值。

说明

方法；检索由文本的 **attrName** 参数指定的属性，该文本由可选的 **startIndex** 和 **endIndex** 参数标识。如果属性与指定范围不相符，**Flash** 将返回 **undefined**。如果省略可选参数 **startIndex** 和 **endIndex**，该方法将使用整个文本范围。如果仅指定 **startIndex**，则使用的范围将是该位置上的单个字符。如果指定 **startIndex** 和 **endIndex**，则范围将从 **startIndex** 开始直到（但不包括）**endIndex**。

示例

下面的示例获取当前所选文本字段的字体大小并显示：

```
var TheTextSize = fl.getDocumentDOM().selection[0].getTextAttr("size");
fl.trace(TheTextSize);
```

下面的示例获取所选文本字段的文本填充颜色：

```
var TheFill = fl.getDocumentDOM().selection[0].getTextAttr("fillColor");
fl.trace(TheFill);
```

下面的示例获取第三个字符的大小：

```
var Char3 = fl.getDocumentDOM().selection[0].getTextAttr("size", 2);
fl.trace(Char3);
```

下面的示例获取所选文本字段中从第三个字符到第八个字符的颜色：

```
fl.getDocumentDOM().selection[0].getTextAttr("fillColor", 2, 8);
```

## text.getTextString()

可用性

Flash MX 2004。

用法

```
text.getTextString([startIndex [, endIndex]])
```

参数

**startIndex** 一个整数，指定第一个字符的索引（从零开始）。此参数是可选的。

**endIndex** 一个整数，指定文本范围的结尾，文本范围从 **startIndex** 开始，直到（但不包括）**endIndex**。此参数是可选的。

返回

指定范围中的文本的字符串。

说明

方法；检索指定范围的文本。如果省略了可选的参数 **startIndex** 和 **endIndex**，将返回整个文本字符串。如果仅指定 **startIndex**，该方法将返回从索引位置开始到字段末尾结束的字符串。如果同时指定 **startIndex** 和 **endIndex**，方法将返回从 **startIndex** 开始，直到（但不包括）**endIndex** 的字符串。

示例

下面的示例获取从所选文本字段的第五个字符到末尾的字符：

```
var myText = fl.getDocumentDOM().selection[0].getTextString(4);  
fl.trace(myText);
```

下面的示例获取所选文本字段的第四个字符到第九个字符：

```
var myText = fl.getDocumentDOM().selection[0].getTextString(3, 9);  
fl.trace(myText);
```

## text.length

可用性

Flash MX 2004。

用法

```
text.length
```

描述

只读属性；一个整数，它表示 **Text** 对象中的字符数。

示例

下面的示例返回所选文本中的字符数：

```
var textLength = fl.getDocumentDOM().selection[0].length;
```



## text.lineType

可用性

Flash MX 2004。

用法

`text.lineType`

描述

属性；一个字符串，它设置线条类型。可接受的值为 "single line"、"multiline"、"multiline no wrap" 和 "password"。

此属性仅对动态或输入文本有效，用于静态文本会生成警告。"password" 值仅用于输入文本。

示例

下面的示例将 `lineType` 属性设置为值 `multiline no wrap`：

```
fl.getDocumentDOM().selection[0].lineType = "multiline no wrap";
```

## text.maxCharacters

可用性

Flash MX 2004。

用法

`text.maxCharacters`

描述

属性；一个整数，它指定用户可在此 `Text` 对象中输入的最大字符数。

此属性仅对输入文本起作用；用于其他文本类型会生成警告。

示例

下面的示例将 `maxCharacters` 属性的值设置为 30：

```
fl.getDocumentDOM().selection[0].maxCharacters = 30;
```

## text.orientation

可用性

Flash MX 2004。

用法

`text.orientation`

描述

属性；一个字符串，它指定文本字段的方向。可接受的值为 "horizontal"、"vertical left to right" 和 "vertical right to left"。

此属性仅对静态文本起作用；用于其他文本类型会生成警告。

#### 示例

下面的示例将 `orientation` 属性设置为 `vertical right to left`:

```
fl.getDocumentDOM().selection[0].orientation = "vertical right to left";
```

## text.renderAsHTML

#### 可用性

Flash MX 2004。

#### 用法

```
text.renderAsHTML
```

#### 描述

属性；一个布尔值。如果值为 `true`，Flash 将文本绘制为 HTML 并解释嵌入的 HTML 标签。

此属性仅对动态或输入文本有效；用于其他文本类型会生成警告。

#### 示例

下面的示例将 `renderAsHTML` 属性设置为 `true`:

```
fl.getDocumentDOM().selection[0].renderAsHTML = true;
```

## text.scrollable

#### 可用性

Flash MX 2004。

#### 用法

```
text.scrollable
```

#### 描述

属性；一个布尔值。如果值为 `true`，则文本可以滚动。

此属性仅对动态文本或输入文本起作用；用于静态文本时会出现警告。

#### 示例

下面的示例将 `scrollable` 属性设置为 `false`:

```
fl.getDocumentDOM().selection[0].scrollable = false;
```

## text.selectable

#### 可用性

Flash MX 2004。

**Text 对象**

## 用法

```
text.selectable
```

## 描述

属性；一个布尔值。如果值为 **true**，则文本可选择。

输入文本始终是可选择的。当此属性设置为 **false** 并用于输入文本时，**Flash** 会生成警告。

## 示例

下面的示例将 **selectable** 属性设置为 **true**：

```
fl.getDocumentDOM().selection[0].selectable = true;
```

## text.selectionEnd

## 可用性

Flash MX 2004。

## 用法

```
text.selectionEnd
```

## 描述

属性；一个从零开始的整数，它指定文本部分选定的末尾。有关详细信息，请参阅 [text.selectionStart](#)。

## text.selectionStart

## 可用性

Flash MX 2004。

## 用法

```
text.selectionStart
```

## 描述

属性；一个从零开始的整数，它指定文本部分选定的开头。可以使用此属性和 **text.selectionEnd** 来选择一系列字符。会选中直到（但不包括）**text.selectionEnd** 的字符。请参阅 [text.selectionEnd](#)。

- 如果有一个插入点或没有所选内容，则 **text.selectionEnd** 等于 **text.selectionStart**。
- 如果将 **text.selectionStart** 设置为大于 **text.selectionEnd** 的值，则会将 **text.selectionEnd** 设置为 **text.selectionStart**，并且不选择任何文本。

## 示例

下面的示例将文本部分选定的开始设置为第六个字符：

```
fl.getDocumentDOM().selection[0].selectionStart = 5;
```

下面的示例从包含文本 **My name is Barbara** 的文本字段中选择字符 **Barbara**，并将其格式设置为粗体和绿色：

```
fl.getDocumentDOM().selection[0].selectionStart = 11;
fl.getDocumentDOM().selection[0].selectionEnd = 18;
var s = fl.getDocumentDOM().selection[0].selectionStart;
var e = fl.getDocumentDOM().selection[0].selectionEnd;
fl.getDocumentDOM().setElementTextAttr('bold', true, s, e);
fl.getDocumentDOM().setElementTextAttr("fillColor", "#00ff00", s, e);
```

## text.setTextAttr()

可用性

Flash MX 2004。

用法

```
text.setTextAttr(attrName, attrValue [, startIndex [, endIndex]])
```

参数

**attrName** 一个字符串，指定要更改的 **TextAttrs** 对象属性的名称。

**attrValue** **TextAttrs** 对象属性的值。

有关 **attrName** 和 **attrValue** 的可能值的列表，请参阅 [TextAttrs 对象](#) 的“属性”摘要。

**startIndex** 一个整数，为数组中第一个字符的索引（从零开始）。此参数是可选的。

**endIndex** 一个整数，指定选定文本字符串终点的索引，文本范围从 **startIndex** 开始，直到（但不包括）**endIndex**。此参数是可选的。

返回

无。

描述

方法；将由 **attrName** 参数（该参数与由 **startIndex** 和 **endIndex** 标识的文本相关联）指定的属性设置为由 **attrValue** 指定的值。此方法可用于更改可能跨 **TextRun** 元素（请参阅 [TextRun 对象](#)）的文本的属性，或那些是现有的 **TextRun** 元素的一部分的文本的属性。使用此方法可能会更改此对象 **text.textRuns** 数组中 **TextRun** 元素的位置和数量（请参阅 [text.textRuns](#)）。

如果省略了可选参数，该方法将使用整个 **Text** 对象的字符范围。如果仅指定 **startIndex**，则范围将是该位置上的单个字符。如果指定 **startIndex** 和 **endIndex**，则范围将从 **startIndex** 开始直到（但不包括）**endIndex** 处的字符。

示例

下面的示例将所选文本字段设置为斜体：

```
fl.getDocumentDOM().selection[0].setTextAttr("italic", true);
```

下面的示例将第三个字符的大小设置为 10：

```
fl.getDocumentDOM().selection[0].setTextAttr("size", 10, 2);
```

下面的示例将所选文本中从第三个字符到第八个字符的颜色设置为红色：

```
fl.getDocumentDOM().selection[0].setTextAttr("fillColor", 0xff0000, 2, 8);
```

## text.setTextString()

可用性

Flash MX 2004。

用法

```
text.setTextString(text [, startIndex [, endIndex]])
```

参数

**text** 一个字符串，包含要插入此 **Text** 对象的字符。

**startIndex** 一个整数，指定要插入文本的字符串中的字符的索引（从零开始）。此参数是可选的。

**endIndex** 一个整数，指定所选文本字符串终点的索引。新文本将覆盖从 **startIndex** 开始直到（但不包括）**endIndex** 的文本。此参数是可选的。

返回

无。

描述

属性；更改此 **Text** 对象中的文本字符串。如果省略了可选参数，将替换整个 **Text** 对象。如果仅指定 **startIndex**，指定字符串将插入 **startIndex** 位置。如果指定 **startIndex** 和 **endIndex**，则指定字符串将替换从 **startIndex** 开始直到（但不包括）**endIndex** 的文本段。

示例

下面的示例将字符串 **this is a string** 赋值给所选文本字段：

```
fl.getDocumentDOM().selection[0].setTextString("this is a string");
```

下面的示例从所选文本字段的第五个字符处开始，插入字符串 **abc**：

```
fl.getDocumentDOM().selection[0].setTextString("01234567890");
fl.getDocumentDOM().selection[0].setTextString("abc", 4);
// text field is now "0123abc4567890"
```

下面的示例将所选文本字符串从第三个字符到第八个字符的文本替换为字符串 **abcdefghij**。**startIndex** 和 **endIndex** 之间的字符将被覆盖。从 **endIndex** 开始的字符跟在插入的字符串之后。

```
fl.getDocumentDOM().selection[0].setTextString("01234567890");
fl.getDocumentDOM().selection[0].setTextString("abcdefghij", 2, 8);
// text field is now "01abcdefghij890"
```

## text.shortcut

可用性

Flash MX 2004。

用法

```
text.shortcut
```

**Text 对象****描述**

属性；一个字符串，它等效于“辅助功能”面板中的“快捷键”字段。此快捷键由屏幕读取器读出。此属性不能用于动态文本。

**示例**

下面的示例获取所选对象的快捷键并显示其值：

```
var theShortcut = fl.getDocumentDOM().selection[0].shortcut;  
fl.trace(theShortcut);
```

下面的示例设置所选对象的快捷键：

```
fl.getDocumentDOM().selection[0].shortcut = "Ctrl+i";
```

## text.silent

**可用性**

Flash MX 2004。

**用法**

```
text.silent
```

**描述**

属性；一个布尔值，它指定对象是否可访问。这等效于“辅助功能”面板中的“使对象可访问”设置的反逻辑。也就是说，如果 `silent` 为 `true`，则“使对象可访问”选项未选中。如果为 `false`，则“使对象可访问”处于选中状态。

**示例**

下面的示例确定对象是否可访问（值 `false` 表示对象可访问）：

```
var isSilent = fl.getDocumentDOM().selection[0].silent;
```

下面的示例将对象设置为可访问：

```
fl.getDocumentDOM().selection[0].silent = false;
```

## text.tabIndex

**可用性**

Flash MX 2004。

**用法**

```
text.tabIndex
```

**描述**

属性；一个整数，它等效于“辅助功能”面板中的“Tab 键索引”字段。此值使您可以确定用户按 **Tab** 键时访问对象的顺序。

**示例**

下面的示例获取当前所选对象的 `tabIndex`：

```
var theTabIndex = fl.getDocumentDOM().selection[0].tabIndex;
```

下面的示例设置当前所选对象的 `tabIndex`:

```
fl.getDocumentDOM().selection[0].tabIndex = 1;
```

## text.textRuns

可用性

Flash MX 2004。

用法

```
text.textRuns
```

描述

只读属性； `TextRun` 对象的数组（请参阅 [TextRun 对象](#)）。

示例

下面的示例在 `myTextRuns` 变量中存储 `textRuns` 属性的值：

```
var myTextRuns = fl.getDocumentDOM().selection[0].textRuns;
```

## text.textType

可用性

Flash MX 2004。

用法

```
text.textType
```

描述

属性；一个字符串，它指示文本字段的类型。可接受的值为 "static"、"dynamic" 和 "input"。

示例

下面的示例将 `textType` 属性设置为 `input`：

```
fl.getDocumentDOM().selection[0].textType = "input";
```

## text.useDeviceFonts

可用性

Flash MX 2004。

用法

```
text.useDeviceFonts
```

描述

属性；一个布尔值。值 `true` 会使 `Flash` 使用设备字体绘制文本。

#### 示例

下面的示例让 Flash 在绘制文本时使用设备字体:

```
fl.getDocumentDOM().selection[0].useDeviceFonts = true;
```

## text.variableName

#### 可用性

Flash MX 2004。

#### 用法

```
text.variableName
```

#### 描述

属性; 一个字符串, 它包含与 Text 对象关联的变量的名称。此属性仅对动态或输入文本有效; 用于其他文本类型会生成警告。

仅支持在 ActionScript 1.0 和 ActionScript 2.0 中使用此属性。

#### 示例

下面的示例将所选文本框的变量名称设置为 firstName:

```
fl.getDocumentDOM().selection[0].variableName = "firstName";
```



## 第 43 章 : TextAttrs 对象

可用性

Flash MX 2004。

描述

TextAttrs 对象包含能应用于部分选定的文本的所有属性。此对象为 TextRun 对象的属性 ([textRun.textAttrs](#))。

属性摘要

下列属性可用于 TextAttrs 对象：

属性	描述
<a href="#">textAttrs.aliasText</a>	一个布尔值，它指定 Flash 是否应使用为增加较小文本的清晰度而经优化的方法来绘制文本。
<a href="#">textAttrs.alignment</a>	一个字符串，它指定段落的对齐方式。可接受值为 "left"、"center"、"right" 和 "justify"。
<a href="#">textAttrs.autoKern</a>	一个布尔值，它确定 Flash 是使用 (true) 还是忽略 (false) 字体的字符对间距微调信息来调整文本的字距。
<a href="#">textAttrs.bold</a>	一个布尔值。值为 true 会导致文本以粗体显示。
<a href="#">textAttrs.characterPosition</a>	一个字符串，用于确定文本的基线。
<a href="#">textAttrs.characterSpacing</a>	不推荐使用，推荐使用 <a href="#">textAttrs.letterSpacing</a> 。一个整数，它表示字符的间距。
<a href="#">textAttrs.face</a>	一个字符串，它表示字体的名称，如 "Arial"。
<a href="#">textAttrs.fillColor</a>	一个字符串、十六进制值或整数，它表示填充的颜色。
<a href="#">textAttrs.indent</a>	一个整数，它指定段落的缩进。
<a href="#">textAttrs.italic</a>	一个布尔值。值为 true 将导致文本以斜体显示。
<a href="#">textAttrs.leftMargin</a>	一个整数，它指定段落的左边距。
<a href="#">textAttrs.letterSpacing</a>	一个整数，它表示字符的间距。
<a href="#">textAttrs.lineSpacing</a>	一个整数，它指定段落的行距（前导）。
<a href="#">textAttrs.rightMargin</a>	一个整数，它指定段落的右边距。
<a href="#">textAttrs.rotation</a>	一个布尔值。值为 true 会导致 Flash 将文本字符旋转 90°。默认值为 false。
<a href="#">textAttrs.size</a>	一个整数，它指定字体的大小。
<a href="#">textAttrs.target</a>	一个字符串，表示文本字段的 target 属性。
<a href="#">textAttrs.url</a>	一个字符串，它表示文本字段的 URL 属性。

### textAttrs.aliasText

可用性

Flash MX 2004。

**TextAttrs 对象**

## 用法

```
textAttrs.aliasText
```

## 描述

属性；一个布尔值，它指定 **Flash** 是否应使用为增加较小文本的清晰度而经优化的方法来绘制文本。

## 示例

下面的示例将把当前选定文本字段中的所有文本的 `aliasText` 属性设置为 `true`：

```
fl.getDocumentDOM().setElementTextAttr('aliasText', true);
```

## textAttrs.alignment

## 可用性

Flash MX 2004。

## 用法

```
textAttrs.alignment
```

## 描述

属性；指定段落对齐方式的字符串。可接受值为 "left"、"center"、"right" 和 "justify"。

## 示例

下面的示例将包含索引 0 至索引 3（但不包括索引 3）之间的字符的段落设置为两端对齐。如果同一段落中存在指定范围以外的其他字符，则此设置还会影响这些字符。

```
fl.getDocumentDOM().setTextSelection(0, 3);  
fl.getDocumentDOM().setElementTextAttr("alignment", "justify");
```

## textAttrs.autoKern

## 可用性

Flash MX 2004。

## 用法

```
textAttrs.autoKern
```

## 描述

属性；一个布尔值，它确定 **Flash** 在调整文本的字距时是使用 (`true`) 还是忽略 (`false`) 字体的相对距离调整信息。

## 示例

下面的示例将选择索引 2 至索引 6（但不包括索引 6）中的字符，并将 `autoKern` 属性设置为 `true`：

```
fl.getDocumentDOM().setTextSelection(2, 6);  
fl.getDocumentDOM().setElementTextAttr('autoKern', true);
```

## textAttrs.bold

可用性

Flash MX 2004。

用法

```
textAttrs.bold
```

描述

属性；一个布尔值。值为 `true` 会导致文本以粗体显示。

示例

下面的示例将选择所选文本对象的第一个字符，并将 `bold` 属性设置为 `true`：

```
fl.getDocumentDOM().setTextSelection(0, 1);  
fl.getDocumentDOM().setElementTextAttr('bold', true);
```

## textAttrs.characterPosition

可用性

Flash MX 2004。

用法

```
textAttrs.characterPosition
```

描述

属性；确定文本基线的字符串。可接受值为 `"normal"`、`"subscript"` 和 `"superscript"`。此属性仅适用于静态文本。

示例

下面的示例选择所选文本字段在索引 2 至索引 6（但不包括索引 6）中的字符，并将 `characterPosition` 属性设置为 `subscript`：

```
fl.getDocumentDOM().setTextSelection(2, 6);  
fl.getDocumentDOM().setElementTextAttr("characterPosition", "subscript");
```

## textAttrs.characterSpacing

可用性

Flash MX 2004。在 Flash 8 中，不推荐使用，而推荐使用 [textAttrs.letterSpacing](#)。

用法

```
textAttrs.characterSpacing
```

描述

属性；一个整数，它表示字符间距。可接受值为 `-60` 至 `60`。

此属性仅适用于静态文本；如果用于其他文本类型，将生成警告。

#### 示例

下面的示例将选定文本字段的字符间距设置为 10:

```
fl.getDocumentDOM().setElementTextAttr("characterSpacing", 10);
```

## textAttrs.face

#### 可用性

Flash MX 2004。

#### 用法

```
textAttrs.face
```

#### 描述

属性; 表示字体名称的字符串, 如 "Arial"。

#### 示例

下面的示例将所选文本字段在索引 2 至索引 8 (但不包括索引 8) 之间的字符设置为使用 Arial 字体:

```
fl.getDocumentDOM().selection[0].setTextAttr("face", "Arial", 2, 8);
```

## textAttrs.fillColor

#### 可用性

Flash MX 2004。

#### 用法

```
textAttrs.fillColor
```

#### 描述

属性; 填充的颜色, 使用以下格式之一:

- 格式为 "#RRGGBB" 或 "#RRGGBBAA" 的字符串
- 格式为 0xRRGGBB 的十六进制数字
- 表示与十六进制数字等价的十进制整数

#### 示例

下面的示例将选定文本字段中位于索引 2 至索引 8 (但不包括索引 8) 之间的字符的颜色设置为红色:

```
fl.getDocumentDOM().selection[0].setTextAttr("fillColor", 0xff0000, 2, 8);
```

## textAttrs.indent

#### 可用性

Flash MX 2004。

**TextAttrs 对象**

## 用法

```
textAttrs.indent
```

## 描述

属性；一个整数，它指定段落的缩进。可接受值为 -720 至 720。

## 示例

下面的示例将位于索引 2 至索引 8（但不包括索引 8）之间的字符的选定文本字段的缩进设置为 100。如果同一段落中存在指定范围以外的其他字符，则此设置还会影响这些字符。

```
fl.getDocumentDOM().selection[0].setTextAttr("indent", 100, 2, 8);
```

## textAttrs.italic

## 可用性

Flash MX 2004。

## 用法

```
textAttrs.italic
```

## 描述

属性；一个布尔值。值为 **true** 将导致文本以斜体显示。

## 示例

下面的示例将所选文本字段设置为斜体：

```
fl.getDocumentDOM().selection[0].setTextAttr("italic", true);
```

## textAttrs.leftMargin

## 可用性

Flash MX 2004。

## 用法

```
textAttrs.leftMargin
```

## 描述

属性；一个整数，它指定段落的左边距。可接受值为 0 至 720。

## 示例

下面的示例将位于索引 2 至索引 8（但不包括索引 8）之间的字符的选定文本字段的 **leftMargin** 属性设置为 100。如果同一段落中存在指定范围以外的其他字符，则此设置还会影响这些字符。

```
fl.getDocumentDOM().selection[0].setTextAttr("leftMargin", 100, 2, 8);
```

## textAttrs.letterSpacing

可用性

Flash 8。

用法

```
textAttrs.letterSpacing
```

描述

属性；一个整数，它表示字符间距。可接受值为 -60 至 60。

此属性仅适用于静态文本；如果用于其他文本类型，将生成警告。

示例

以下代码将选择从索引 0 到索引 10（不含）的所有字符，并将字符间距设置为 60：

```
fl.getDocumentDOM().setTextSelection(0, 10);  
fl.getDocumentDOM().setElementTextAttr("letterSpacing", 60);
```

## textAttrs.lineSpacing

可用性

Flash MX 2004。

用法

```
textAttrs.lineSpacing
```

描述

属性；一个整数，它指定段落的行距（前导）。可接受值为 -360 至 720。

示例

下面的示例将所选文本字段的 `lineSpacing` 属性设置为 100：

```
fl.getDocumentDOM().selection[0].setTextAttr("lineSpacing", 100);
```

## textAttrs.rightMargin

可用性

Flash MX 2004。

用法

```
textAttrs.rightMargin
```

描述

属性；一个整数，它指定段落的右边距。可接受值为 0 至 720。

#### 示例

下面的示例将位于索引 2 至索引 8（但不包括索引 8）之间的字符的选定文本字段的 `rightMargin` 属性设置为 100。如果同一段落中存在指定范围以外的其他字符，则此设置还会影响这些字符。

```
f1.getDocumentDOM().selection[0].setTextAttr("rightMargin", 100, 2, 8);
```

## textAttrs.rotation

#### 可用性

Flash MX 2004。

#### 用法

```
textAttrs.rotation
```

#### 描述

属性；一个布尔值。值为 `true` 会导致 Flash 将文本字符旋转 90°。默认值为 `false`。此属性仅适用于垂直方向的静态文本；如果用于其他文本类型，则系统将发出警告。

#### 示例

下面的示例将选定文本字段的旋转设置为 `true`：

```
f1.getDocumentDOM().setElementTextAttr("rotation", true);
```

## textAttrs.size

#### 可用性

Flash MX 2004。

#### 用法

```
textAttrs.size
```

#### 描述

属性；一个整数，它指定字体的大小。

#### 示例

下面的示例将检索位于索引 2 处的字符大小，并在“输出”面板中显示结果：

```
f1.outputPanel.trace(f1.getDocumentDOM().selection[0].getTextAttr("size", 2));
```

## textAttrs.target

#### 可用性

Flash MX 2004。

#### 用法

```
textAttrs.target
```

#### 描述

属性；一个字符串，它表示文本字段的 **target** 属性。此属性仅适用于静态文本。

#### 示例

下面的示例获取当前场景顶层第一帧中的文本字段的 **target** 属性，并在“输出”面板中显示该属性：

```
f1.outputPanel.trace(f1.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].getTextAttr("target"));
```

## textAttrs.url

#### 可用性

Flash MX 2004。

#### 用法

```
textAttrs.url
```

#### 描述

属性；一个字符串，它表示文本字段的 **URL** 属性。此属性仅适用于静态文本。

#### 示例

下面的示例将所选文本字段的 **URL** 设置为 <http://www.adobe.com>：

```
f1.getDocumentDOM().setElementTextAttr("url", "http://www.adobe.com");
```



## 第 44 章 : TextRun 对象

### 可用性

Flash MX 2004。

### 描述

**TextRun** 对象表示一串字符，其属性与 **TextAttrs** 对象中的所有属性相匹配。此对象为 **Text** 对象的一个属性 (**text.textRuns**)。

### 属性摘要

除了可与 **Text** 对象一起使用的属性以外，**TextRun** 对象还提供以下属性：

属性	描述
<a href="#">textRun.characters</a>	一个字符串，它表示包含在 <b>TextRun</b> 对象中的文本。
<a href="#">textRun.textAttrs</a>	<b>TextAttrs</b> 对象，包含一串文本的属性。

## textRun.textAttrs

### 可用性

Flash MX 2004。

### 用法

```
textRun.textAttrs
```

### 描述

属性；**TextAttrs** 对象包含文本串的属性。

### 示例

下面的示例将在“输出”面板中显示选定文本字段中的第一串字符的属性。

```
var curTextAttrs = fl.getDocumentDOM().selection[0].textRuns[0].textAttrs;
for (var prop in curTextAttrs) {
    fl.trace(prop + " = " + curTextAttrs[prop]);
}
```

## textRun.characters

### 可用性

Flash MX 2004。

### 用法

```
textRun.characters
```

**TextRun** 对象

描述

属性；包含在 **TextRun** 对象中的文本。

示例

下面的示例将在“输出”面板中显示组成选定文本字段中的第一串字符的字符：

```
fl.trace(fl.getDocumentDOM().selection[0].textRuns[0].characters);
```

## 第 45 章 : Timeline 对象

可用性

Flash MX 2004。

描述

**Timeline** 对象表示 Flash 时间轴，可使用 `fl.getDocumentDOM().getTimeline()` 访问当前文档的时间轴。此方法返回当前正在编辑的场景或元件的时间轴。

处理场景时，每个场景的时间轴都有一个索引值，可通过 `fl.getDocumentDOM().timelines[i]` 访问当前文档的时间轴。（在本示例中，`i` 是时间轴的索引值。）

使用 **Timeline** 对象的方法和属性处理帧时，请记住帧值是一个零始索引，而非帧序列在时间轴中的实际帧编号。也就是说，第 1 帧的帧索引为 0。

方法摘要

**Timeline** 对象有以下方法：

方法	描述
<code>timeline.addMotionGuide()</code>	在当前图层上面添加一个运动引导层，并将当前图层附着到新添加的引导层。
<code>timeline.addNewLayer()</code>	在文档中添加一个新图层，并使其成为当前图层。
<code>timeline.clearFrames()</code>	删除当前图层上一个帧或一系列帧的所有内容。
<code>timeline.clearKeyframes()</code>	将当前图层上的关键帧转换为普通帧，并删除其内容。
<code>timeline.convertToBlankKeyframes()</code>	将当前图层上的帧转换为空白关键帧。
<code>timeline.convertToKeyframes()</code>	将当前图层上的一系列帧转换成关键帧；如果没有指定帧，则转换所选范围内的帧。
<code>timeline.copyFrames()</code>	将当前图层上的一系列帧复制到剪贴板。
<code>timeline.copyLayers()</code>	将一系列时间轴图层复制到剪贴板。
<code>timeline.copyMotion()</code>	从补间动画或从逐帧动画中复制选定帧上的动画，以便可以将其应用于其他帧。
<code>timeline.copyMotionAsAS3()</code>	从补间动画或从逐帧动画中，将所选帧上的动画以 <b>ActionScript 3.0</b> 代码的形式复制到剪贴板上。
<code>timeline.createMotionObject()</code>	在指定的起始帧和结束帧处创建一个新动画对象。
<code>timeline.createMotionTween()</code>	将当前图层上每个所选关键帧的 <code>frame.tweenType</code> 属性设置为 <code>motion</code> 。如果需要，还可以将每个帧的内容转换为单个元件实例。
<code>timeline.cutFrames()</code>	从时间轴中剪切当前图层上的一系列帧，并将它们保存到剪贴板。
<code>timeline.cutLayers()</code>	剪切一系列时间轴图层并将其保存到剪贴板。
<code>timeline.deleteLayer()</code>	删除图层。
<code>timeline.duplicateLayers()</code>	直接复制所选图层或指定的图层。
<code>timeline.expandFolder()</code>	展开或折叠指定的一个或多个文件夹。
<code>timeline.findLayerIndex()</code>	在索引数组中查找具有指定名称的图层。

方法	描述
<code>timeline.getFrameProperty()</code>	为所选帧检索指定属性的值。
<code>timeline.getGuidelines()</code>	返回一个 XML 字符串，它表示时间轴水平和垂直辅助线的当前位置（“视图”>“辅助线”>“显示辅助线”）。
<code>timeline.getLayerProperty()</code>	为所选图层检索指定属性的值。
<code>timeline.getSelectedFrames()</code>	在数组中检索当前选择的帧。
<code>timeline.getSelectedLayers()</code>	检索当前所选图层的零始索引值。
<code>timeline.insertBlankKeyframe()</code>	在指定的帧索引处插入空白关键帧；如果没有指定索引，则使用“播放头/选择”插入空白关键帧。
<code>timeline.insertFrames()</code>	在给定的帧编号处插入指定个数的帧。
<code>timeline.insertKeyframe()</code>	在指定的帧处插入关键帧。
<code>timeline.pasteFrames()</code>	将剪贴板中的一系列帧粘贴到指定的帧中。
<code>timeline.pasteLayers()</code>	将复制的图层粘贴到指定图层索引上方的时间轴上。
<code>timeline.pasteMotion()</code>	将 <code>timeline.copyMotion()</code> 检索到的一系列动画帧粘贴到时间轴。
<code>timeline.removeFrames()</code>	删除帧。
<code>timeline.removeMotionObject()</code>	删除使用 <code>timeline.createMotionObject()</code> 创建的动画对象并将帧转换为静态帧。
<code>timeline.reorderLayer()</code>	将指定的第一个图层移动到指定的第二个图层之前或之后。
<code>timeline.reverseFrames()</code>	翻转一系列帧。
<code>timeline.selectAllFrames()</code>	选择当前时间轴中的所有帧。
<code>timeline setFrameProperty()</code>	为所选帧设置 <b>Frame</b> 对象的属性。
<code>timeline.setGuidelines()</code>	将时间轴的辅助线替换为指定的信息。
<code>timeline.setLayerProperty()</code>	将所选全部图层上的指定属性设置为指定的值。
<code>timeline.setSelectedFrames()</code>	选择当前图层中的一系列帧，或者将所选帧设置为传递到此方法中的选择数组。
<code>timeline.setSelectedLayers()</code>	将图层设置为选中；使所选图层成为当前图层。
<code>timeline.showLayerMasking()</code>	通过锁定遮罩和被遮罩的图层，在创作过程中显示图层遮罩。
<code>timeline.startPlayback()</code>	如果时间轴当前没有播放，则启动时间轴的自动播放。
<code>timeline.stopPlayback()</code>	如果时间轴当前正在播放，则停止时间轴的自动播放。

#### 属性摘要

Timeline 对象具有以下属性：

属性	说明
<code>timeline.currentFrame</code>	当前播放头位置处的帧的零始索引。
<code>timeline.currentLayer</code>	当前活动图层的零始索引。
<code>timeline.frameCount</code>	只读；一个整数，它表示此时间轴最长图层中的帧数。
<code>timeline.layerCount</code>	只读；一个整数，它表示指定时间轴中的图层数。

属性	说明
<a href="#">timeline.layers</a>	只读；图层对象的数组。
<a href="#">timeline.libraryItem</a>	只读属性；指示时间轴是否属于某个场景。
<a href="#">timeline.name</a>	一个字符串，它表示当前时间轴的名称。

## timeline.addMotionGuide()

可用性

Flash MX 2004。

用法

```
timeline.addMotionGuide()
```

参数

无。

返回

一个整数，它表示新添加的引导层的从零开始的索引。如果当前图层的类型不是 "Normal"，则 Flash 返回 -1。

说明

方法；在当前图层上方添加一个运动引导层并将当前图层附加到新添加的引导层上，同时将当前图层转换成类型为 "Guided" 的图层。

此方法只对类型为 "Normal" 的图层起作用。它对类型为 "Folder"、"Mask"、"Masked"、"Guide" 或 "Guided" 的层没有任何作用。

示例

下面的示例在当前图层上方添加一个运动引导层，并将当前图层转换为 Guided：

```
fl.getDocumentDOM().getTimeline().addMotionGuide();
```

## timeline.addNewLayer()

可用性

Flash MX 2004。

用法

```
timeline.addNewLayer([name] [, layerType [, bAddAbove]])
```

参数

**name** 一个字符串，它指定新图层的名称。如果省略此参数，则新图层会被赋予一个默认的新图层名 ("Layer n"，其中 n 是总图层数)。此参数是可选的。

**layerType** 一个字符串，它指定要添加的图层的类型。如果省略此参数，则创建 "Normal" 类型的图层。此参数是可选的。可接受值为 "normal"、"guide"、"guided"、"mask"、"masked" 和 "folder"。

**bAddAbove** 一个布尔值，如果设置为 true (默认值)，则 Flash 会在当前图层上方添加新图层；如果设置为 false，则 Flash 会在当前图层下方添加图层。此参数是可选的。

**Timeline 对象**

## 返回

新添加图层的从零开始的整数索引值。

## 说明

方法；在文档中添加新图层并将它作为当前图层。

## 示例

下面的示例在时间轴上添加一个新图层，新图层的名称是 Flash 生成的默认名称：

```
fl.getDocumentDOM().getTimeline().addNewLayer();
```

下面的示例在当前图层上方添加一个新的文件夹图层，并命名为 **Folder1**：

```
fl.getDocumentDOM().getTimeline().addNewLayer("Folder1", "folder", true);
```

## timeline.clearFrames()

## 可用性

Flash MX 2004。

## 用法

```
timeline.clearFrames([startFrameIndex [, endFrameIndex]])
```

## 参数

**startFrameIndex** 一个从零开始的索引，它定义要清除的帧范围的起点。如果省略 **startFrameIndex**，则该方法使用当前的选择。此参数是可选的。

**endFrameIndex** 一个从零开始的索引，它定义要清除的帧范围的终点。该范围的终点为 **endFrameIndex**（但不包括此值）。如果您只指定 **startFrameIndex**，则 **endFrameIndex** 默认为 **startFrameIndex** 的值。此参数是可选的。

## 返回

无。

## 描述

方法；删除当前图层中某个帧或某个范围内的帧的所有内容。

## 示例

下面的示例清除第 6 帧到（但不包括）第 11 帧（记住：索引值不同于帧编号值）：

```
fl.getDocumentDOM().getTimeline().clearFrames(5, 10);
```

下面的示例清除第 15 帧：

```
fl.getDocumentDOM().getTimeline().clearFrames(14);
```

## timeline.clearKeyframes()

## 可用性

Flash MX 2004。

#### 用法

```
timeline.clearKeyframes([startFrameIndex [, endFrameIndex]])
```

#### 参数

**startFrameIndex** 一个从零开始的索引，它定义要清除的帧范围的起点。如果省略 **startFrameIndex**，则该方法使用当前的选择。此参数是可选的。

**endFrameIndex** 一个从零开始的索引，它定义要清除的帧范围的终点。该范围的终点为 **endFrameIndex**（但不包括此值）。如果您只指定 **startFrameIndex**，则 **endFrameIndex** 默认为 **startFrameIndex** 的值。此参数是可选的。

#### 返回

无。

#### 描述

方法；将关键帧转换为普通帧，并删除它在当前图层中的内容。

#### 示例

下面的示例清除从第 5 帧到（但不包括）第 10 帧的关键帧（记住：索引值不同于帧编号值）：

```
fl.getDocumentDOM().getTimeline().clearKeyframes(4, 9);
```

下面的示例清除第 15 帧处的关键帧并将它转换成普通帧：

```
fl.getDocumentDOM().getTimeline().clearKeyframes(14);
```

## timeline.convertToBlankKeyframes()

#### 可用性

Flash MX 2004。

#### 用法

```
timeline.convertToBlankKeyframes([startFrameIndex [, endFrameIndex]])
```

#### 参数

**startFrameIndex** 一个从零开始的索引，它指定要转换成关键帧的起始帧。如果省略 **startFrameIndex**，则该方法转换当前选定的帧。此参数是可选的。

**endFrameIndex** 一个从零开始的索引，它指定将停止转换成关键帧时的帧位置。要转换的帧范围的终点为 **endFrameIndex**（但不包括此值）。如果您只指定 **startFrameIndex**，则 **endFrameIndex** 默认为 **startFrameIndex** 的值。此参数是可选的。

#### 返回

无。

#### 描述

方法；将当前图层的帧转换为空白关键帧。

#### 示例

下面的示例将第 2 帧到（但不包括）第 10 帧转换成空白关键帧（记住：索引值不同于帧编号值）：

```
fl.getDocumentDOM().getTimeline().convertToBlankKeyframes(1, 9);
```

下面的示例将第 5 帧转换成空白关键帧：

```
fl.getDocumentDOM().getTimeline().convertToBlankKeyframes(4);
```

## timeline.convertToKeyframes()

可用性

Flash MX 2004。

用法

```
timeline.convertToKeyframes([startFrameIndex [, endFrameIndex]])
```

参数

**startFrameIndex** 一个从零开始的索引，它指定要转换成关键帧的起始帧。如果省略 **startFrameIndex**，则该方法转换当前选定的帧。此参数是可选的。

**endFrameIndex** 一个从零开始的索引，它指定将停止转换成关键帧时的帧位置。要转换的帧范围的终点为 **endFrameIndex**（但不包括此值）。如果您只指定 **startFrameIndex**，则 **endFrameIndex** 默认为 **startFrameIndex** 的值。此参数是可选的。

返回

无。

描述

方法；将当前图层中的某个范围内的帧转换成关键帧（如果没有指定帧，则转换所选范围内的帧）。

示例

下面的示例将选定的帧转换成关键帧：

```
fl.getDocumentDOM().getTimeline().convertToKeyframes();
```

下面的示例将第 2 帧到（但不包括）第 10 帧转换成关键帧（记住：索引值不同于帧编号值）：

```
fl.getDocumentDOM().getTimeline().convertToKeyframes(1, 9);
```

下面的示例将第 5 帧转换成关键帧：

```
fl.getDocumentDOM().getTimeline().convertToKeyframes(4);
```

## timeline.copyFrames()

可用性

Flash MX 2004。

用法

```
timeline.copyFrames([startFrameIndex [, endFrameIndex]])
```

参数

**startFrameIndex** 一个从零开始的索引，它指定要复制的帧范围的起点。如果省略 **startFrameIndex**，则该方法使用当前的选择。此参数是可选的。

**endFrameIndex** 一个从零开始的索引，它指定将停止复制时的帧位置。要复制的帧范围的终点为 **endFrameIndex**（但不包括此值）。如果您只指定 **startFrameIndex**，则 **endFrameIndex** 默认为 **startFrameIndex** 的值。此参数是可选的。



**Timeline** 对象

## 返回

无。

## 描述

方法；将当前图层的某个范围内的帧复制到剪贴板。

## 示例

下面的示例将选定的帧复制到剪贴板：

```
fl.getDocumentDOM().getTimeline().copyFrames();
```

下面的示例将第 2 帧到（但不包括）第 10 帧复制到剪贴板（记住：索引值不同于帧编号值）：

```
fl.getDocumentDOM().getTimeline().copyFrames(1, 9);
```

下面的示例将第 5 帧复制到剪贴板：

```
fl.getDocumentDOM().getTimeline().copyFrames(4);
```

## timeline.copyLayers()

## 可用性

Flash CS5.5 Professional。

## 用法

```
timeline.copyLayers([startLayerIndex [, endLayerIndex]])
```

## 参数

**startLayerIndex** 可选。一个从零开始的索引，它指定要复制的图层范围的起点。如果省略 **startLayerIndex**，则该方法使用当前的选择。

**endLayerIndex** 可选。一个从零开始的索引，它指定将停止复制时的图层位置。要复制的图层范围的终点为 **endLayerIndex**（包括此值）。如果您只指定 **startLayerIndex**，则 **endLayerIndex** 默认为 **startLayerIndex** 的值。

## 返回

无。

## 描述

方法；复制时间轴中当前选定的图层，或复制指定范围内的图层。可提供可选参数，以便指定要复制的图层或图层范围。

## 示例

下面的示例复制时间轴中从索引 2 至索引 7 的图层：

```
fl.getDocumentDOM().getTimeline().copyLayers(2, 7);
```

## 另请参见

[timeline.cutLayers\(\)](#)、[timeline.pasteLayers\(\)](#)、[timeline.duplicateLayers\(\)](#)

## timeline.copyMotion()

可用性

Flash CS3 Professional。

用法

```
timeline.copyMotion()
```

参数

无。

返回

无。

描述

方法；从补间动画或从逐帧动画复制选定帧上的动画。然后，您可以使用 [timeline.pasteMotion\(\)](#) 将动画应用于其他帧。

要将动画复制为可以粘贴到脚本中的文本（代码），请参阅 [timeline.copyMotionAsAS3\(\)](#)。

示例

下面的示例从选定的一个或多个帧中复制动画：

```
fl.getDocumentDOM().getTimeline().copyMotion();
```

另请参见

[timeline.copyMotionAsAS3\(\)](#), [timeline.pasteMotion\(\)](#)

## timeline.copyMotionAsAS3()

可用性

Flash CS3 Professional。

用法

```
timeline.copyMotionAsAS3()
```

参数

无。

返回

无。

描述

方法；从补间动画或从逐帧动画中将所选帧上的动画以 ActionScript 3.0 代码的形式复制到剪贴板上。然后，可以将此代码粘贴到脚本中。

要以可应用于其他帧的格式复制动画，请参阅 [timeline.copyMotion\(\)](#)。

## 示例

下面的示例从选定的一个或多个帧中将动画以 **ActionScript 3.0** 代码的形式复制到剪贴板上：

```
fl.getDocumentDOM().getTimeline().copyMotionAsAS3();
```

另请参见

[timeline.copyMotion\(\)](#)

## timeline.createMotionObject()

## 可用性

Flash Professional CS5。

## 用法

```
timeline.createMotionObject([startFrame [,endFrame])
```

## 参数

**startFrame** 指定要创建动画对象的第一个帧。如果省略 **startFrame**，则该方法使用当前的选择；如果没有选择，则删除所有图层中当前播放头位置的所有帧。此参数是可选的。

**endFrame** 指定要停止创建动画对象的帧位置；帧范围的终点为 **endFrame**（但不包括此值）。如果您只指定 **startFrame**，则 **endFrame** 默认为 **startFrame** 值。此参数是可选的。

## 返回

无。

## 描述

方法；创建一个新动画对象。这些参数是可选的，指定这些参数后，会在创建动画对象之前将时间轴选项设置为指示的帧。

## 示例

以下示例在顶层的当前播放头位置创建动画对象：

```
fl.getDocumentDOM().getTimeline().currentLayer = 0;  
fl.getDocumentDOM().getTimeline().createMotionObject();
```

以下示例从第 5 帧开始创建动画对象，直到当前场景中顶层的第 15 帧（但不包括该帧）：

```
fl.getDocumentDOM().getTimeline().currentLayer = 0;  
fl.getDocumentDOM().getTimeline().createMotionObject(5, 15);
```

## timeline.createMotionTween()

## 可用性

Flash MX 2004。

## 用法

```
timeline.createMotionTween([startFrameIndex [, endFrameIndex]])
```

## Timeline 对象

## 参数

**startFrameIndex** 一个从零开始的索引，它指定要创建补间动画的起始帧位置。如果省略 **startFrameIndex**，则该方法使用当前的选择。此参数是可选的。

**endFrameIndex** 一个从零开始的索引，它指定要停止创建补间动画时的帧位置。帧范围的终点为 **endFrameIndex**（但不包括此值）。如果您只指定 **startFrameIndex**，则 **endFrameIndex** 默认为 **startFrameIndex** 的值。此参数是可选的。

## 返回

无。

## 描述

方法；将当前图层中每个选定的关键帧的 **frame.tweenType** 属性设置为 **motion**，如果需要，还可以将每个帧的内容转换为单个元件实例。此属性等同于 Flash 创作工具中的“创建补间动画”菜单项。

## 示例

下面的示例将从第一帧到（但不包括）第 10 帧中的形状转换成图形元件实例，并将 **frame.tweenType** 设置为 **motion**（记住：索引值不同于帧编号值）：

```
fl.getDocumentDOM().getTimeline().createMotionTween(0, 9);
```

## timeline.currentFrame

## 可用性

Flash MX 2004。

## 用法

```
timeline.currentFrame
```

## 描述

属性；当前播放头位置的帧的从零开始的索引。

## 示例

下面的示例将当前时间轴的播放头设置为第 10 帧（请记住，索引值不同于帧编号值）：

```
fl.getDocumentDOM().getTimeline().currentFrame = 9;
```

下面的示例将当前播放头位置值存储在 **curFrame** 变量中：

```
var curFrame = fl.getDocumentDOM().getTimeline().currentFrame;
```

## timeline.currentLayer

## 可用性

Flash MX 2004。

## 用法

```
timeline.currentLayer
```

## Timeline 对象

## 描述

属性；当前活动图层的从零开始的索引。值 **0** 指的是最上面的图层，值 **1** 指的是它的下一图层，依此类推。

## 示例

下面的示例将顶层设为活动图层：

```
fl.getDocumentDOM().getTimeline().currentLayer = 0;
```

下面的示例将当前活动图层的索引存储在 `curLayer` 变量中：

```
var curLayer = fl.getDocumentDOM().getTimeline().currentLayer;
```

## timeline.cutFrames()

## 可用性

Flash MX 2004。

## 用法

```
timeline.cutFrames([startFrameIndex [, endFrameIndex]])
```

## 参数

**startFrameIndex** 一个从零开始的索引，它指定要剪切的帧范围的起点。如果省略 **startFrameIndex**，则该方法使用当前的选择。此参数是可选的。

**endFrameIndex** 一个从零开始的索引，它指定要停止剪切时的帧位置。帧范围的终点为 **endFrameIndex**（但不包括此值）。如果您只指定 **startFrameIndex**，则 **endFrameIndex** 默认为 **startFrameIndex** 的值。此参数是可选的。

## 返回

无。

## 描述

方法；从时间轴中剪切当前图层上的某个范围内的帧并将它们保存到剪贴板。

## 示例

下面的示例从时间轴剪切选定的帧并将它们保存到剪贴板：

```
fl.getDocumentDOM().getTimeline().cutFrames();
```

下面的示例从时间轴剪切第 **2** 帧到（但不包括）第 **10** 帧，并将它们保存到剪贴板（请记住，索引值不同于帧编号值）：

```
fl.getDocumentDOM().getTimeline().cutFrames(1, 9);
```

下面的示例从时间轴剪切第 **5** 帧并将它保存到剪贴板：

```
fl.getDocumentDOM().getTimeline().cutFrames(4);
```

## timeline.cutLayers()

## 可用性

Flash CS5.5 Professional。

## Timeline 对象

## 用法

```
timeline.cutLayers([startLayerIndex [, endLayerIndex]])
```

## 参数

**startLayerIndex** 可选。一个从零开始的索引，它指定要剪切的图层范围的起点。如果省略 **startLayerIndex**，则该方法使用当前的选择。

**endLayerIndex** 可选。一个从零开始的索引，它指定将停止剪切时的图层位置。要剪切的图层范围的终点为 **endLayerIndex**（包括此值）。如果您只指定 **startLayerIndex**，则 **endLayerIndex** 默认为 **startLayerIndex** 的值。

## 返回

无。

## 描述

方法；剪切时间轴中当前选定的图层，或剪切指定范围内的图层。可提供可选参数，以便指定要剪切的图层或图层范围。

## 示例

下面的示例剪切时间轴中从索引 2 至索引 7 的图层：

```
fl.getDocumentDOM().getTimeline().cutLayers(2, 7);
```

## 另请参见

[timeline.copyLayers\(\)](#)、[timeline.pasteLayers\(\)](#)、[timeline.duplicateLayers\(\)](#)

## timeline.deleteLayer()

## 可用性

Flash MX 2004。

## 用法

```
timeline.deleteLayer([index])
```

## 参数

**index** 一个从零开始的索引，它指定要删除的图层。如果时间轴中只有一个图层，则此方法无效。此参数是可选的。

## 返回

无。

## 描述

方法；删除一个图层。如果该图层是文件夹，则该文件夹中的所有图层都将被删除。如果您没有指定图层索引，则 **Flash** 会删除当前选定的图层。

## 示例

下面的示例删除从顶层起的第二个图层：

```
fl.getDocumentDOM().getTimeline().deleteLayer(1);
```

下面的示例删除当前选定的图层：

```
fl.getDocumentDOM().getTimeline().deleteLayer();
```

## timeline.duplicateLayers()

可用性

Flash CS5.5 Professional。

用法

```
timeline.duplicateLayers([startLayerIndex [, endLayerIndex]])
```

参数

**startLayerIndex** 可选。一个从零开始的索引，它指定要复制的图层范围的起点。它还指定将剪贴板上的图层粘贴到其上方的图层。如果省略 **startLayerIndex**，则该方法使用当前的图层选择。

**endLayerIndex** 可选。一个从零开始的索引，它指定将停止复制时的图层位置。要复制的图层范围的终点为 **endLayerIndex**（包括此值）。如果您只指定 **startLayerIndex**，则 **endLayerIndex** 默认为 **startLayerIndex** 的值。

返回

无。

描述

方法；直接复制时间轴中当前选定的图层，或直接复制指定范围内的图层。可提供可选参数，以便指定要复制的图层或图层范围。

示例

下面的示例直接复制时间轴中当前选定的图层：

```
fl.getDocumentDOM().getTimeline().duplicateLayers();
```

下面的示例复制从索引 2 到索引 7 位于图层索引 2 上方的图层：

```
fl.getDocumentDOM().getTimeline().duplicateLayers(2,7);
```

另请参见

[timeline.copyLayers\(\)](#)、[timeline.cutLayers\(\)](#)、[timeline.pasteLayers\(\)](#)

## timeline.expandFolder()

可用性

Flash MX 2004。

用法

```
timeline.expandFolder(bExpand [, bRecurseNestedParents [, index]])
```

参数

**bExpand** 一个布尔值，如果设置为 **true**，则该方法会展开文件夹；如果设置为 **false**，则该方法会折叠文件夹。

**bRecurseNestedParents** 一个布尔值，如果设置为 **true**，则会根据 **bExpand** 参数打开或关闭指定文件夹中的所有图层。此参数是可选的。

**index** 要展开或折叠的文件夹的从零开始的索引。使用 **-1** 可应用到所有图层（您还必须将 **bRecurseNestedParents** 设置为 **true**）。此属性等同于 Flash 创作工具中的“展开所有 / 折叠所有”菜单项。此参数是可选的。

**Timeline 对象**

返回

无。

描述

方法；展开或折叠指定的文件夹。如果您没有指定图层，则此方法在当前图层上操作。

示例

下面的示例使用此文件夹结构：

```
Folder 1 ***
--layer 7
--Folder 2 ****
---Layer 5
```

下面的示例只展开文件夹 1：

```
fl.getDocumentDOM().getTimeline().currentLayer = 1;
fl.getDocumentDOM().getTimeline().expandFolder(true);
```

下面的示例只展开文件夹 1（假设上次文件夹 1 折叠时文件夹 2 也折叠；否则，文件夹 2 会显示为展开）：

```
fl.getDocumentDOM().getTimeline().expandFolder(true, false, 0);
```

下面的示例折叠当前时间轴中的所有文件夹：

```
fl.getDocumentDOM().getTimeline().expandFolder(false, true, -1);
```

## timeline.findLayerIndex()

可用性

Flash MX 2004。

用法

```
timeline.findLayerIndex(name)
```

参数

**name** 一个字符串，它指定要查找的图层的名称。

返回

所指定图层的索引值数组。如果没有找到指定图层，则 Flash 返回 `undefined`。

说明

方法；查找具有指定名称的图层的索引数组。图层索引是一维的，所以文件夹被认为是主索引的一部分。

示例

下面的示例在“输出”面板中显示所有图层名为 Layer 7 的索引值：

```
var layerIndex = fl.getDocumentDOM().getTimeline().findLayerIndex("Layer 7");
fl.trace(layerIndex);
```

下面的示例演示如何将此方法返回的值传回 `timeline.setSelectedLayers()`：

```
var layerIndex = fl.getDocumentDOM().getTimeline().findLayerIndex("Layer 1");
fl.getDocumentDOM().getTimeline().setSelectedLayers(layerIndex[0], true);
```



## timeline.frameCount

可用性

Flash MX 2004。

用法

```
timeline.frameCount
```

描述

只读属性；一个整数，它表示此时间轴的最长图层中的帧数。

示例

下面的示例使用 `countNum` 变量来存储当前文档的最长图层中的帧数：

```
var countNum = fl.getDocumentDOM().getTimeline().frameCount;
```

## timeline.getFrameProperty()

可用性

Flash MX 2004。

用法

```
timeline.getFrameProperty(property [, startFrameIndex [, endFrameIndex]])
```

参数

**属性** 一个字符串，它指定要获得其值的属性的名称。有关属性的完整列表，请参阅 [Frame 对象](#) 的“属性”摘要。

**startFrameIndex** 一个从零开始的索引，它指定要获得其值的起始帧编号。如果省略 `startFrameIndex`，则该方法使用当前的选择。此参数是可选的。

**endFrameIndex** 一个从零开始的索引，它指定要选择的帧范围的终点。该范围的终点为 `endFrameIndex`（但不包括此值）。如果您只指定 `startFrameIndex`，则 `endFrameIndex` 默认为 `startFrameIndex` 的值。此参数是可选的。

返回

指定属性的值，如果所有选定的帧都没有相同的属性值，则返回 `undefined`。

说明

方法；检索选定帧的指定属性的值。

示例

下面的示例检索当前文档最上面的图层中起始帧的名称，并将该名称显示在“输出”面板中：

```
fl.getDocumentDOM().getTimeline().currentLayer = 0;  
fl.getDocumentDOM().getTimeline().setSelectedFrames(0, 0, true);  
var frameName = fl.getDocumentDOM().getTimeline().getFrameProperty("name");  
fl.trace(frameName);
```

## timeline.getGuidelines()

可用性

Flash CS4 Professional.

用法

```
timeline.getGuidelines()
```

参数

无。

返回

一个 XML 字符串。

说明

方法：返回一个 XML 字符串，它表示时间轴的水平辅助线的当前位置（“视图”>“辅助线”>“显示辅助线”）。若要将这些辅助线应用于时间轴，请使用 [timeline.setGuidelines\(\)](#)。

示例

假定第一个时间轴上有一些辅助线，下面的示例在“输出”面板中将它们显示为一个 XML 字符串：

```
var currentTimeline = fl.getDocumentDOM().timelines[0];  
fl.trace(currentTimeline.getGuidelines());
```

## timeline.getLayerProperty()

可用性

Flash MX 2004。

用法

```
timeline.getLayerProperty(property)
```

参数

**属性** 一个字符串，它指定要检索其值的属性的名称。有关属性的列表，请参阅 [Frame 对象](#) 的“摘要”属性。

返回

指定属性的值。Flash 查看该图层的属性以确定类型。如果所有指定图层都没有相同的属性值，则 Flash 返回 `undefined`。

说明

方法；检索选定图层的指定属性的值。

示例

下面的示例检索当前文档中最上面图层的名称，并将它显示在“输出”面板中：

```
fl.getDocumentDOM().getTimeline().currentLayer = 0;  
var layerName = fl.getDocumentDOM().getTimeline().getLayerProperty("name");  
fl.trace(layerName);
```

## timeline.getSelectedFrames()

可用性

Flash MX 2004。

参数

无。

返回

一个包含  $3n$  个整数的数组，其中  $n$  是选定区域的数目。每个组中的第一个整数是图层索引，第二个整数是选择范围开始位置的起始帧，而第三个整数指定选择范围的结束帧。结束帧不包含在选择范围内。

说明

方法；检索一个数组中当前选定的帧。

示例

当最上面的图层为当前图层时，下面的示例在“输出”面板中显示 0,5,10,0,20,25:

```
var timeline = fl.getDocumentDOM().getTimeline();
timeline.setSelectedFrames(5,10);
timeline.setSelectedFrames(20,25,false);
var theSelectedFrames = timeline.getSelectedFrames();
fl.trace(theSelectedFrames);
```

另请参见

[timeline.setSelectedFrames\(\)](#)

## timeline.getSelectedLayers()

可用性

Flash MX 2004。

参数

无。

返回

选定图层的从零开始的索引值数组。

说明

方法；获得当前选定图层从零开始的索引值。

示例

下面的示例在“输出”面板中显示 1,0:

```
fl.getDocumentDOM().getTimeline().setSelectedLayers(0);
fl.getDocumentDOM().getTimeline().setSelectedLayers(1, false);
var layerArray = fl.getDocumentDOM().getTimeline().getSelectedLayers();
fl.trace(layerArray);
```

另请参见

[timeline.setSelectedLayers\(\)](#)

## timeline.insertBlankKeyframe()

可用性

Flash MX 2004。

用法

```
timeline.insertBlankKeyframe([frameNumIndex])
```

参数

**frameNumIndex** 一个从零开始的索引，它指定要插入关键帧的帧位置。如果省略 **frameNumIndex**，则该方法使用当前播放头帧编号。此参数是可选的。

如果指定或选定的帧是一个普通帧，则关键帧会插在该帧上。例如，如果您有编号为从 1 到 10 的 10 个帧，并且您选择第 5 帧，则此方法会使第 5 帧成为空白关键帧，而帧范围的长度仍然是 10 帧。如果选择了第 5 帧，而它是关键帧且旁边是一个普通帧，则此方法会在第 6 帧处插入一个空白关键帧。如果第 5 帧是关键帧且它旁边的帧已经是关键帧，则不会插入关键帧，但播放头会移到第 6 帧处。

返回

无。

描述

方法；在指定的帧索引处插入一个空白关键帧；如果没有指定索引，则该方法会使用“播放头 / 选择”来插入空白关键帧。请参阅 [timeline.insertKeyframe\(\)](#)。

示例

下面的示例在第 20 帧处插入一个空白关键帧（记住：索引值不同于帧编号值）：

```
fl.getDocumentDOM().getTimeline().insertBlankKeyframe(19);
```

下面的示例在当前选定的帧处插入一个空白关键帧（如果没有选择帧，则在播放头位置插入）：

```
fl.getDocumentDOM().getTimeline().insertBlankKeyframe();
```

## timeline.insertFrames()

可用性

Flash MX 2004。

用法

```
timeline.insertFrames([numFrames [, bAllLayers [, frameNumIndex]])
```

参数

**numFrames** 一个整数，它指定要插入的帧的数目。如果省略此参数，则该方法会在当前图层的当前选择位置插入帧。此参数是可选的。

**bAllLayers** 一个布尔值，如果设置为 **true**（默认值），则该方法会将 **numFrames** 参数中指定数目的帧插入所有图层中；如果设置为 **false**，则该方法会将帧插入当前图层中。此参数是可选的。

**frameNumIndex** 一个从零开始的索引，它指定要插入新帧的帧位置。此参数是可选的。

返回

无。

描述

方法；在指定索引处插入指定数目的帧。

如果没有指定参数，则此方法会按以下方式工作：

- 如果没有选择一个或多个帧，则该方法会在当前图层的第一个选定帧的位置上插入选定数目的帧。也就是说，如果选择范围是从第 6 帧到第 10 帧（总共五帧），则该方法会在包含选定帧的图层上的第 6 帧处添加五个帧。
- 如果没有选择帧，则该方法会在所有图层的当前帧处插入一个帧。

如果指定参数，则该方法按以下方式工作：

- 如果只指定 **numFrames**，则在当前图层的当前帧处插入指定数目的帧。
- 如果指定了 **numFrames**，并且 **bAllLayers** 为 **true**，则在所有图层的当前帧处插入指定数目的帧。
- 如果三个参数都指定，则在指定索引 (**frameIndex**) 处插入指定数目的帧；传递给 **bAllLayers** 的值确定这些帧是只添加到当前图层中，还是添加到所有图层中。

如果指定或选定的帧是一个普通帧，则会在该帧处插入帧。例如，如果您有编号为从 1 到 10 的 10 个帧，并且您选择第 5 帧（或者向 **frameIndex** 传递值 4），则此方法会在第 5 帧处添加一个帧，帧范围的长度变为 11 帧。如果选择了第 5 帧并且它是一个关键帧，则无论第 6 帧旁的帧是否是关键帧，此方法都会在第 6 帧处插入一个帧。

示例

下面的示例会在当前图层的当前选择位置上插入一个帧（或者多个帧，取决于选择范围）：

```
fl.getDocumentDOM().getTimeline().insertFrames();
```

下面的示例在所有图层的当前帧处插入五个帧：

```
fl.getDocumentDOM().getTimeline().insertFrames(5);
```

注：如果在多个图层中都包含帧，则当您使用上一个命令选择一个图层中的某一帧时，**Flash** 会只在选定图层中插入帧。如果您有多个图层，而在其中没有选择帧，则 **Flash** 会在所有图层中插入帧。

下面的示例只在当前图层中插入三个帧：

```
fl.getDocumentDOM().getTimeline().insertFrames(3, false);
```

下面的示例会在所有图层中插入四个帧，从第一个帧开始：

```
fl.getDocumentDOM().getTimeline().insertFrames(4, true, 0);
```

## timeline.insertKeyframe()

可用性

Flash MX 2004。

用法

```
timeline.insertKeyframe([frameNumIndex])
```

### 参数

**frameNumIndex** 一个从零开始的索引，它指定当前图层中要插入关键帧的帧索引。如果省略 **frameNumIndex**，则该方法使用当前播放头或选定帧的帧编号。此参数是可选的。

### 返回

无。

### 描述

方法；在指定帧处插入一个关键帧。如果省略该参数，则该方法会使用播放头或选择位置插入关键帧。

此方法的作用与 [timeline.insertBlankKeyframe\(\)](#) 相同，不同之处在于插入的关键帧包含它转换的帧的内容（也就是说，它不是空白帧）。

### 示例

下面的示例在播放头或选择位置上插入一个关键帧：

```
fl.getDocumentDOM().getTimeline().insertKeyframe();
```

下面的示例在第二层的第 10 帧上插入一个关键帧（记住：索引值不同于帧或图层编号值）：

```
fl.getDocumentDOM().getTimeline().currentLayer = 1;  
fl.getDocumentDOM().getTimeline().insertKeyframe(9);
```

## timeline.layerCount

### 可用性

Flash MX 2004。

### 用法

```
timeline.layerCount
```

### 描述

只读属性；一个整数，它表示指定时间轴中的图层数。

### 示例

下面的示例使用 **NumLayer** 变量存储当前场景中的图层数：

```
var NumLayer = fl.getDocumentDOM().getTimeline().layerCount;
```

## timeline.layers

### 可用性

Flash MX 2004。

### 用法

```
timeline.layers
```

### 描述

只读属性；图层对象数组。

**Timeline 对象**

## 示例

下面的示例使用 `currentLayers` 变量存储当前文档中的图层对象数组：

```
var currentLayers = fl.getDocumentDOM().getTimeline().layers;
```

## timeline.libraryItem

## 可用性

Flash Professional CS5。

## 用法

```
timeline.libraryItem
```

## 描述

只读属性；如果时间轴的 `libraryItem` 属性为空，则该时间轴属于某个场景。如果不为空，则可以将其视为 **LibraryItem** 对象。

## 示例

如果 `libraryItem` 的值不是 `null`，则以下示例输出 `libraryItem` 的名称，如果 `libraryItem` 为 `null`，则输出该场景的名称：

```
var item = fl.getDocumentDOM().getTimeline().libraryItem;
if (item)
    fl.trace("libraryItem name: " + item.name);
else
    fl.trace("scene name: " + fl.getDocumentDOM().getTimeline().name);
```

## timeline.name

## 可用性

Flash MX 2004。

## 用法

```
timeline.name
```

## 描述

属性；一个字符串，它指定当前时间轴的名称。此名称是正在编辑的当前场景、屏幕（幻灯片或表单）或元件的名称。

## 示例

下面的示例检索第一个场景名称：

```
var sceneName = fl.getDocumentDOM().timelines[0].name;
```

下面的示例将第一个场景名称设置为 `FirstScene`：

```
fl.getDocumentDOM().timelines[0].name = "FirstScene";
```

## timeline.pasteFrames()

可用性

Flash MX 2004。

用法

```
timeline.pasteFrames([startFrameIndex [, endFrameIndex]])
```

参数

**startFrameIndex** 一个从零开始的索引，它指定要粘贴的帧范围的起点。如果省略 **startFrameIndex**，则该方法使用当前的选择。此参数是可选的。

**endFrameIndex** 一个从零开始的索引，它指定要停止粘贴帧时的帧位置。该方法粘贴范围的终点为 **endFrameIndex**（但不包括此值）。如果您只指定 **startFrameIndex**，则 **endFrameIndex** 默认为 **startFrameIndex** 的值。此参数是可选的。

返回

无。

描述

方法；将剪贴板中的某个范围内的帧粘贴到指定帧处。

示例

下面的示例将剪贴板中的帧粘贴到当前选定的帧位置或播放头位置：

```
fl.getDocumentDOM().getTimeline().pasteFrames();
```

下面的示例粘贴剪贴板中的第 2 帧到（但不包括）第 10 帧（记住：索引值不同于帧编号值）：

```
fl.getDocumentDOM().getTimeline().pasteFrames(1, 9);
```

下面的示例粘贴剪贴板中从第 5 帧开始的帧：

```
fl.getDocumentDOM().getTimeline().pasteFrames(4);
```

## timeline.pasteLayers()

可用性

Flash CS5.5 Professional。

用法

```
timeline.pasteLayers([layerIndex])
```

参数

**layerIndex** 可选。一个从零开始的索引，它指定将剪贴板上的图层粘贴到其上方的图层。如果省略 **layerIndex**，则该方法使用当前的选择。

返回

一个整数，指示已粘贴的图层的最小图层索引编号。



#### 描述

方法；将以前剪切或复制的图层粘贴到当前所选图层上方，或粘贴到指定的图层索引位置的上方。如果指定的图层是一个文件夹图层，则将图层粘贴到该文件夹中。返回已粘贴的图层的最小图层索引编号。此操作不会影响系统剪贴板。

#### 示例

下面的示例将图层剪贴板中的图层粘贴到时间轴中当前选定的图层上方：

```
fl.getDocumentDOM().getTimeline().pasteLayers();
```

下面的示例将图层剪贴板上的图层粘贴到图层索引 2 的上方：

```
fl.getDocumentDOM().getTimeline().pasteLayers(2);
```

#### 另请参见

[timeline.cutLayers\(\)](#)、[timeline.copyLayers\(\)](#)、[timeline.duplicateLayers\(\)](#)

## timeline.pasteMotion()

#### 可用性

Flash CS3 Professional。

#### 用法

```
timeline.pasteMotion()
```

#### 参数

无。

#### 返回

无。

#### 描述

方法；将由 [timeline.copyMotion\(\)](#) 检索的某个范围内的动画帧粘贴至时间轴。如果有必要，会移动现有帧的位置（向右移动）以便为要粘贴的帧留出空间。

#### 示例

下面的示例将剪贴板上的动画粘贴至当前所选的帧或播放头位置，并将该帧移动到所粘贴帧的右边。

```
fl.getDocumentDOM().getTimeline().pasteMotion();
```

#### 另请参见

[timeline.copyMotion\(\)](#)

## timeline.removeFrames()

#### 可用性

Flash MX 2004。

## Timeline 对象

## 用法

```
timeline.removeFrames([startFrameIndex [,endFrameIndex]])
```

## 参数

**startFrameIndex** 一个从零开始的索引，它指定要开始删除的第一个帧。如果省略 **startFrameIndex**，则该方法使用当前的选择；如果没有选择，则删除所有图层中当前播放头位置的所有帧。此参数是可选的。

**endFrameIndex** 一个从零开始的索引，它指定要停止删除帧时的帧位置；帧范围的终点为 **endFrameIndex**（但不包括此值）。如果您只指定 **startFrameIndex**，则 **endFrameIndex** 默认为 **startFrameIndex** 的值。此参数是可选的。

## 返回

无。

## 描述

方法；删除帧。

## 示例

下面的示例删除当前场景的顶层中的第 5 帧到（但不包括）第 10 帧（记住：索引值不同于帧编号值）：

```
fl.getDocumentDOM().getTimeline().currentLayer = 0;  
fl.getDocumentDOM().getTimeline().removeFrames(4, 9);
```

下面的示例删除当前场景的顶层中的第 8 帧：

```
fl.getDocumentDOM().getTimeline().currentLayer = 0;  
fl.getDocumentDOM().getTimeline().removeFrames(7);
```

## timeline.removeMotionObject()

## 可用性

Flash Professional CS5。

## 用法

```
timeline.removeMotionObject([startFrame [,endFrame])
```

## 参数

**startFrame** 指定要开始删除动画对象的第一个帧位置。如果省略 **startFrame**，则该方法使用当前的选择；如果没有选择，则删除所有图层中当前播放头位置的所有帧。此参数是可选的。

**endFrame** 指定要停止删除动画对象的帧位置；帧范围的终点为 **endFrame**（但不包括此值）。如果您只指定 **startFrame**，则 **endFrame** 默认为 **startFrame** 值。此参数是可选的。

## 返回

无。

## 描述

方法；删除动画对象并将帧转换回静态帧。这些参数是可选的，指定这些参数后，会在删除动画对象之前将时间轴选项设置为指示的帧。

## 示例

以下示例删除顶层中当前播放头位置处的所有动画对象并将帧转换回静态帧：

## Timeline 对象

```
fl.getDocumentDOM().getTimeline().currentLayer = 0;  
fl.getDocumentDOM().getTimeline().removeMotionObject();
```

以下示例删除从当前场景中顶层的第 5 帧到第 15 帧（但不包括该帧）的动画对象：

```
fl.getDocumentDOM().getTimeline().currentLayer = 0;  
fl.getDocumentDOM().getTimeline().removeMotionObject(5, 15);
```

另请参见

第 435 页的“[timeline.createMotionObject\(\)](#)”

## timeline.reorderLayer()

可用性

Flash MX 2004。

用法

```
timeline.reorderLayer(layerToMove, layerToPutItBy [, bAddBefore])
```

参数

**layerToMove** 一个从零开始的索引，它指定要移动的图层。

**layerToPutItBy** 一个从零开始的索引，它指定要将该图层移到哪一个图层旁边。例如，如果您为 **layerToMove** 指定值 1，为 **layerToPutItBy** 指定值 0，则第二图层会放在第一图层旁边。

**bAddBefore** 指定是将图层移到 **layerToPutItBy** 的前面还是后面。如果您指定 **false**，则该图层会移到 **layerToPutItBy** 后面。默认值为 **true**。此参数是可选的。

返回

无。

描述

方法；将第一个指定图层移到第二个指定图层的前面或后面。

示例

下面的示例将索引 2 处的图层移到顶部（索引 0 处的图层的顶部）：

```
fl.getDocumentDOM().getTimeline().reorderLayer(2, 0);
```

下面的示例将索引 3 处的图层放在索引 5 处的图层的后面：

```
fl.getDocumentDOM().getTimeline().reorderLayer(3, 5, false);
```

## timeline.reverseFrames()

可用性

Flash MX 2004。

用法

```
timeline.reverseFrames([startFrameIndex [, endFrameIndex]])
```

### 参数

**startFrameIndex** 一个从零开始的索引，它指定要开始翻转的第一个帧。如果省略 **startFrameIndex**，则该方法使用当前的选择。此参数是可选的。

**endFrameIndex** 一个从零开始的索引，它指定要停止翻转的第一帧；帧范围的终点为 **endFrameIndex**（但不包括此值）。如果您只指定 **startFrameIndex**，则 **endFrameIndex** 默认为 **startFrameIndex** 的值。此参数是可选的。

### 返回

无。

### 描述

方法；翻转某个范围内的帧。

### 示例

下面的示例翻转当前选定帧的位置：

```
fl.getDocumentDOM().getTimeline().reverseFrames();
```

下面的示例翻转第 10 帧到（但不包括）第 15 帧（记住：索引值不同于帧编号值）：

```
fl.getDocumentDOM().getTimeline().reverseFrames(9, 14);
```

## timeline.selectAllFrames()

### 可用性

Flash MX 2004。

### 用法

```
timeline.selectAllFrames()
```

### 参数

无。

### 返回

无。

### 描述

方法；选择当前时间轴中的所有帧。

### 示例

下面的示例选择当前时间轴中的所有帧。

```
fl.getDocumentDOM().getTimeline().selectAllFrames();
```

## timeline setFrameProperty()

### 可用性

Flash MX 2004。

#### 用法

```
timeline.setFrameProperty(property, value [, startFrameIndex [, endFrameIndex]])
```

#### 参数

**属性** 一个字符串，它指定要修改的属性的名称。有关属性和值的完整列表，请参阅 [Frame 对象](#) 的“摘要”属性。

此方法不能用来设置只读属性（如 [frame.duration](#) 和 [frame.elements](#)）的值。

**value** 指定要为该属性设置的值。若要确定正确的值和类型，请参阅 [Frame 对象](#) 的“属性”摘要。

**startFrameIndex** 一个从零开始的索引，它指定要修改的起始帧编号。如果省略 **startFrameIndex**，则该方法使用当前的选择。此参数是可选的。

**endFrameIndex** 一个从零开始的索引，它指定要停止的第一帧。帧范围的终点为 **endFrameIndex**（但不包括此值）。如果您指定了 **startFrameIndex** 但省略 **endFrameIndex**，则 **endFrameIndex** 默认为 **startFrameIndex** 的值。此参数是可选的。

#### 返回

无。

#### 描述

方法；设置选定帧的 [Frame](#) 对象的属性。

#### 示例

下面的示例为当前文档中顶层的第一帧指定了 `stop()` [ActionScript](#) 命令：

```
f1.getDocumentDOM().getTimeline().currentLayer = 0;  
f1.getDocumentDOM().getTimeline().setSelectedFrames(0,0,true);  
f1.getDocumentDOM().getTimeline().setFrameProperty("actionScript", "stop();");
```

下面的示例为当前图层的第 2 帧到（但不包括）第 5 帧设置补间动画（记住：索引值不同于帧编号值）：

```
var doc = f1.getDocumentDOM();  
doc.getTimeline().setFrameProperty("tweenType", "motion", 1, 4);
```

## timeline.setGuidelines()

#### 可用性

Flash CS4 Professional.

#### 用法

```
timeline.setGuidelines(xmlString)
```

#### 参数

**xmlString** 一个 XML 字符串，包含有关要应用的辅助线的信息。

#### 返回

一个布尔值，如果成功应用了辅助线，则为 `true`；否则为 `false`。

#### 说明

方法：将时间轴的辅助线（“视图”>“辅助线”>“显示辅助线”）替换为 **xmlString** 中指定的信息。若要检索可传给此方法的 XML 字符串，请使用 [timeline.getGuidelines\(\)](#)。

若要查看新设置的辅助线，必须将其隐藏，然后再查看。

#### 示例

下面的示例将一个 FLA 文件中的辅助线应用于另一个 FLA 文件：

```
var doc0 = fl.documents[0];
var guides0 = doc0.timelines[0].getGuidelines();
var doc1 = fl.documents[1];
doc1.timelines[0].setGuidelines(guides0);
```

## timeline.setLayerProperty()

#### 可用性

Flash MX 2004。

#### 用法

```
timeline.setLayerProperty(property, value [, layersToChange])
```

#### 参数

**属性** 一个字符串，它指定要设置的属性。若要查看属性列表，请参阅第 295 页的“[Layer 对象](#)”。

**value** 要为该属性设置的值。请使用与您您在设置 Layer 对象属性时将使用的同样类型的值。

**layersToChange** 一个字符串，它指出应该修改哪些图层。可接受的值为 "selected"、"all" 和 "others"。如果省略此参数，则默认值为 "selected"。此参数是可选的。

#### 返回

无。

#### 描述

方法；将所有选择的图层的指定属性设置为一个指定的值。

#### 示例

下面的示例让选定的图层不可见：

```
fl.getDocumentDOM().getTimeline().setLayerProperty("visible", false);
```

下面的示例将所选图层的名称设置为 selLayer：

```
fl.getDocumentDOM().getTimeline().setLayerProperty("name", "selLayer");
```

## timeline.setSelectedFrames()

#### 可用性

Flash MX 2004。

#### 用法

```
timeline.setSelectedFrames(startFrameIndex, endFrameIndex [, bReplaceCurrentSelection])
timeline.setSelectedFrames(selectionList [, bReplaceCurrentSelection])
```

#### 参数

**startFrameIndex** 一个从零开始的索引，它指定要设置的起始帧。

**endFrameIndex** 一个从零开始的索引，它指定选择范围的终点；**endFrameIndex** 是选择范围最后一帧后面的帧。

**bReplaceCurrentSelection** 一个布尔值，如果设置为 **true**，则在选择指定帧之前取消选择当前选定的帧。默认值为 **true**。

**selectionList** 三个整数组成的数组，由 `timeline.getSelectedFrames()` 返回。

#### 返回

无。

#### 描述

方法；选择当前图层中的某个范围内的帧，或者将选择的帧设置为传递到此方法的选定数组。

#### 示例

下面的示例展示了两种方法，它们用于选择顶层图层的第 1 帧到（但不包括）第 10 帧，然后将该图层的第 12 帧到（但不包括）第 15 帧添加到当前选择（记住，索引值不同于帧编号值）：

```
fl.getDocumentDOM().getTimeline().setSelectedFrames(0, 9);  
fl.getDocumentDOM().getTimeline().setSelectedFrames(11, 14, false);  
fl.getDocumentDOM().getTimeline().setSelectedFrames([0, 0, 9]);  
fl.getDocumentDOM().getTimeline().setSelectedFrames([0, 11, 14], false);
```

下面的示例首先将所选帧的数组存储在 `savedSelectionList` 变量中，然后在有命令或用户交互更改了选择之后，在代码中使用该数组来重新选择这些帧：

```
var savedSelectionList = fl.getDocumentDOM().getTimeline().getSelectedFrames();  
// Do something that changes the selection.  
fl.getDocumentDOM().getTimeline().setSelectedFrames(savedSelectionList);
```

另请参见

[timeline.getSelectedFrames\(\)](#)

## timeline.setSelectedLayers()

#### 可用性

Flash MX 2004。

#### 用法

```
timeline.setSelectedLayers(index [, bReplaceCurrentSelection])
```

#### 参数

**index** 一个从零开始的索引，它指定要选择的图层。

**bReplaceCurrentSelection** 一个布尔值，如果设置为 **true**，则该方法替代当前选定的内容；如果设置为 **false**，则该方法扩展当前选定的内容。默认值为 **true**。此参数是可选的。

#### 返回

无。

#### 描述

方法；将图层设置为选定，同时让指定的图层成为当前图层。选择一个图层也意味着选择了该图层中的所有帧。

## 示例

下面的示例选择顶层:

```
fl.getDocumentDOM().getTimeline().setSelectedLayers(0);
```

下面的示例将下一图层添加到选择中:

```
fl.getDocumentDOM().getTimeline().setSelectedLayers(1, false);
```

另请参见

[timeline.getSelectedLayers\(\)](#)

## timeline.showLayerMasking()

## 可用性

Flash MX 2004。

## 用法

```
timeline.showLayerMasking([layer])
```

## 参数

**图层** 一个从零开始的索引，它指定要在创作时显示遮罩的遮罩或被遮罩的图层。此参数是可选的。

## 返回

无。

## 描述

方法；在创作时通过锁定遮罩和被遮罩的图层来显示图层遮罩。如果没有指定图层，则此方法使用当前图层。如果您在类型不是“Mask”或“Masked”的图层上使用此方法，Flash 会在“输出”面板中显示错误。

## 示例

下面的示例指定在创作时应该显示的顶层图层遮罩。

```
fl.getDocumentDOM().getTimeline().showLayerMasking(0);
```

## timeline.startPlayback()

## 可用性

Flash Professional CS5。

## 用法

```
timeline.startPlayback()
```

## 返回

无。



#### 描述

方法；如果时间轴当前正在播放，则启动时间轴的自动播放。此方法可以与 SWF 面板一起使用，以便在创作环境中控制时间轴播放。

#### 示例

下面的示例启动时间轴的播放。

```
fl.getDocumentDOM().getTimeline().startPlayback();
```

## timeline.stopPlayback()

#### 可用性

Flash Professional CS5。

#### 用法

```
timeline.stopPlayback()
```

#### 返回

无。

#### 描述

方法；如果时间轴当前正在播放，则停止时间轴的自动播放。此方法可以与 SWF 面板一起使用，以便在创作环境中控制时间轴播放。

#### 示例

下面的示例停止时间轴的播放。

```
fl.getDocumentDOM().getTimeline().stopPlayback();
```

## 第 46 章 : ToolObj 对象

可用性

Flash MX 2004。

描述

一个 ToolObj 对象表示“工具”面板中的单个工具。若要访问 ToolObj 对象，请使用 Tools 对象的属性：`tools.toolObjs` 数组或 `tools.activeTool`。

方法摘要

ToolObj 对象有以下可用方法。

注：下列方法仅在创建可扩展工具时使用。

方法	描述
<code>toolObj.enablePIControl()</code>	启用或禁用“属性”检查器中的指定控件。仅在创建可扩展工具时使用。
<code>toolObj.setIcon()</code>	标识将在 Flash“工具”面板中用作工具图标的 PNG 文件。
<code>toolObj.setMenuString()</code>	将显示在弹出菜单中的字符串设置为工具名称。
<code>toolObj.setOptionsFile()</code>	将 XML 文件关联到工具。
<code>toolObj.setPI()</code>	设置一个当工具处于激活状态时使用的特殊“属性”检查器。
<code>toolObj.setToolName()</code>	为用于“工具”面板配置的工具指定一个名称。
<code>toolObj.setToolTip()</code>	设置鼠标停留在工具图标上将显示的工具提示。
<code>toolObj.showPIControl()</code>	显示或隐藏“属性”检查器中的控件。
<code>toolObj.showTransformHandles()</code>	在可扩展工具的 JavaScript 文件的 <code>configureTool()</code> 方法中调用，以提示当工具处于活动状态时应显示任意变形手柄。

属性摘要

以下属性可用于 ToolObj 对象：

属性	描述
<code>toolObj.depth</code>	一个整数，它指定工具在“工具”面板的弹出菜单中的深度。
<code>toolObj.iconID</code>	一个整数，它指定工具的资源 ID。
<code>toolObj.position</code>	只读；一个整数，它指定“工具”面板中工具的位置。

### toolObj.depth

可用性

Flash MX 2004。

用法

`toolObj.depth`

## ToolObj 对象

## 描述

只读整数；一个整数，它指定工具在“工具”面板的弹出菜单中的深度。此属性仅在创建可扩展工具时使用。

## 示例

下面的示例指定工具的深度为 1，也就是处于“工具”面板中某个工具下面一个层级：

```
fl.tools.activeTool.depth = 1;
```

## toolObj.enablePIControl()

## 可用性

Flash MX 2004。

## 用法

```
toolObj.enablePIControl(control, bEnable)
```

## 参数

**control** 一个字符串，它指定要启用或禁用的控件的名称。有效值取决于由该工具调用的“属性”检查器（请参阅 [toolObj.setPI\(\)](#)）。

形状“属性”检查器具有以下控件：

笔触 (stroke)	填充 (fill)
-------------	-----------

文本“属性”检查器具有以下控件：

type	字体 (font)	pointsize
color	bold	italic
direction	alignLeft	alignCenter
alignRight	alignJustify	spacing
position	autoKern	small
rotation	format	lineType
selectable	html	border
deviceFonts	varEdit	options
link	maxChars	target

影片“属性”检查器具有以下控件：

size	发布 (publish)	background
framerate	player	profile

**bEnable** 一个布尔值，它确定是启用 (true) 还是禁用 (false) 控件。

## 返回

无。

**ToolObj 对象**

## 描述

方法；启用或禁用“属性”检查器中的指定控件。仅在创建可扩展工具时使用。

## 示例

在可扩展工具的 JavaScript 文件中使用以下命令，可将 **Flash** 设置为在该工具的“属性”检查器中不显示笔触选项：

```
theTool.enablePControl("stroke", false);
```

## toolObj.iconID

## 可用性

Flash MX 2004。

## 用法

```
toolObj.iconID
```

## 描述

只读属性；一个整数，其值为 -1。此属性仅在创建可扩展工具时使用。iconID 值为 -1 表示 **Flash** 不尝试为该工具查找图标。相反，应由该工具脚本指定要在“工具”面板中显示的图标；请参阅 [toolObj.setIcon\(\)](#)。

## 示例

下面的示例将值 -1（当前工具的图标 ID）指定给 toolIconID 变量：

```
var toolIconID = fl.tools.activeTool.iconID
```

## toolObj.position

## 可用性

Flash MX 2004。

## 用法

```
toolObj.position
```

## 描述

只读属性；一个整数，它指定“工具”面板中工具的位置。此属性仅在创建可扩展工具时使用。

## 示例

使用工具的 JavaScript 文件的 mouseDown() 方法中的以下命令可将该工具在“工具”面板中的位置以整数形式显示在“输出”面板中：

```
myToolPos = fl.tools.activeTool.position;  
fl.trace(myToolPos);
```

## toolObj.setIcon()

可用性

Flash MX 2004。

用法

```
toolObj.setIcon(file)
```

参数

**file** 一个字符串，它指定要用作图标的 PNG 文件的名称。PNG 文件必须与 JSFL 文件放置在同一个文件夹中。

返回

无。

描述

方法；标识将在“工具”面板中用作工具图标的 PNG 文件。此方法仅在创建可扩展工具时使用。

示例

下面的示例指定 **PolyStar.png** 文件中的图像应该用作名为 **PolyStar** 的工具的图标。这段代码摘自 **PolyStar.jsfl** 范例文件（请参阅第 11 页的“[“多角星形”工具范例](#)”）：

```
theTool = fl.tools.activeTool;  
theTool.setIcon("PolyStar.png");
```

## toolObj.setMenuString()

可用性

Flash MX 2004。

用法

```
toolObj.setMenuString(menuStr)
```

参数

**menuStr** 一个字符串，它指定出现在弹出菜单中的名称作为工具名称。

返回

无。

描述

方法；将显示在弹出菜单中的字符串设置为工具名称。此方法仅在创建可扩展工具时使用。

示例

下面的示例指定名为 **theTool** 的工具应该在其弹出菜单中显示名称“**PolyStar Tool**”。这段代码摘自 **PolyStar.jsfl** 范例文件（请参阅第 11 页的“[“多角星形”工具范例](#)”）：

```
theTool = fl.tools.activeTool;  
theTool.setMenuString("PolyStar Tool");
```

## toolObj.setOptionsFile()

可用性

Flash MX 2004。

用法

```
toolObj.setOptionsFile(xmlFile)
```

参数

**xmlFile** 一个字符串，它指定具有工具选项描述的 XML 文件的名称。该 XML 文件必须与 JSFL 文件放在同一个文件夹中。

返回

无。

描述

方法；将 XML 文件关联到工具。此文件指定模式面板（由“属性”检查器中的“选项”按钮调用）中显示的选项。通常可在 JSFL 文件中的 `configureTool()` 函数内使用此方法。请参阅 [configureTool\(\)](#)。

例如，`PolyStar.xml` 文件指定了与 `Polygon` 工具关联的 3 个选项：

```
<properties>
  <property name="Style"
    variable="style"
    list="polygon,star"
    defaultValue="0"
    type="Strings"/>

  <property name="Number of Sides"
    variable="nsides"
    min="3"
    max="32"
    defaultValue="5"
    type="Number" />

  <property name="Star point size"
    variable="pointParam"
    min="0"
    max="1"
    defaultValue=".5"
    type="Double" />
</properties>
```

示例

下面的示例指定名为 `PolyStar.xml` 的文件与当前活动工具相关联。这段代码摘自 `PolyStar.jsfl` 范例文件（请参阅第 11 页的““多角星形”工具范例”）：

```
theTool = fl.tools.activeTool;
theTool.setOptionsFile("PolyStar.xml");
```

## toolObj.setPI()

可用性

Flash MX 2004。

用法

```
toolObj.setPI(pi)
```

参数

**pi** 一个字符串，它指定将为该工具调用的“属性”检查器。

返回

无。

描述

方法；指定当工具处于活动状态时应使用的“属性”检查器。此方法仅在创建可扩展工具时使用。可接受值为“shape”（默认值）、“text”和“movie”。

示例

下面的示例指定当工具处于活动状态时应使用形状“属性”检查器。这段代码摘自 PolyStar.jsfl 范例文件（请参阅第 11 页的““多角星形”工具范例”）：

```
theTool = fl.tools.activeTool;  
theTool.setPI("shape");
```

## toolObj.setToolName()

可用性

Flash MX 2004。

用法

```
toolObj.setToolName(name)
```

参数

**name** 一个字符串，它指定工具的名称。

返回

无。

描述

方法；为用于“工具”面板配置的工具指定一个名称。此方法仅在创建可扩展工具时使用。名称仅用于 XML 布局文件，Flash 读取该文件来构造“工具”面板。名称不在 Flash 用户界面中显示。

示例

下面的示例为名为 theTool 的工具指定名称 polystar。这段代码摘自 PolyStar.jsfl 范例文件（请参阅第 11 页的““多角星形”工具范例”）：

**ToolObj** 对象

```
theTool = fl.tools.activeTool;  
theTool.setToolName("polystar");
```

## toolObj.setToolTip()

可用性

Flash MX 2004。

用法

```
toolObj.setToolTip(toolTip)
```

参数

**toolTip** 一个字符串，它指定用于某工具的工具提示。

返回

无。

描述

方法；设置鼠标停留在工具图标上将显示的工具提示。此方法仅在创建可扩展工具时使用。

示例

下面的示例指定该工具的工具提示应为 PolyStar Tool。这段代码摘自 PolyStar.jsfl 范例文件（请参阅第 11 页的“[“多角星形”工具范例](#)”）：

```
theTool = fl.tools.activeTool;  
theTool.setToolTip("PolyStar Tool");
```

## toolObj.showPIControl()

可用性

Flash MX 2004。

用法

```
toolObj.showPIControl(control, bShow)
```

参数

**control** 一个字符串，它指定要显示或隐藏的控件名称。此方法仅在创建可扩展工具时使用。有效值取决于由该工具调用的属性检查器（请参阅 [toolObj.setPI\(\)](#)）。

形状“属性”检查器具有以下控件：

笔触 (stroke)	填充 (fill)
-------------	-----------

文本“属性”检查器具有以下控件：



## ToolObj 对象

type	字体 (font)	pointsize
color	bold	italic
direction	alignLeft	alignCenter
alignRight	alignJustify	spacing
position	autoKern	small
rotation	format	lineType
selectable	html	border
deviceFonts	varEdit	options
link	maxChars	target

影片“属性”检查器具有以下控件：

size	发布 (publish)	background
framerate	player	profile

**bShow** 一个布尔值，它确定显示或隐藏指定的控件（**true** 显示控件；**false** 隐藏控件）。

返回

无。

描述

方法；显示或隐藏“属性”检查器中的控件。此方法仅在创建可扩展工具时使用。

示例

使用可扩展工具的 JavaScript 文件中的以下命令，可将 Flash 设置为在该工具的“属性”检查器中不显示填充选项：

```
fl.tools.activeTool.showPIControl("fill", false);
```

## toolObj.showTransformHandles()

可用性

Flash MX 2004。

用法

```
toolObj.showTransformHandles (bShow)
```

参数

**bShow** 一个布尔值，它确定显示或隐藏当前工具的任意变形手柄（**true** 显示手柄；**false** 隐藏手柄）。

返回

无。

#### 描述

方法；在可扩展工具的 JavaScript 文件的 `configureTool()` 方法中调用，以指示当工具处于活动状态时应显示任意变形手柄。此方法仅在创建可扩展工具时使用。

#### 示例

请参阅 [configureTool\(\)](#)。

## 第 47 章 : Tools 对象

可用性

Flash MX 2004。

描述

可从 `flash` 对象 (`fl.tools`) 访问 `Tools` 对象。`tools.toolObjs` 属性包含 `ToolObj` 对象数组, `tools.activeTool` 属性返回当前活动工具的 `ToolObj` 对象。请参阅 [ToolObj 对象](#) 和第 13 页的“[顶级函数和方法](#)”中的可扩展工具列表。)

注: 下列方法和属性仅在创建可扩展工具时使用。

方法摘要

`Tools` 对象具有以下方法:

方法	描述
<code>tools.constrainPoint()</code>	以两个点作为输入参数, 并返回一个不同的经调整或受到限制 的点。
<code>tools.getKeyDown()</code>	返回最近按下的键。
第 471 页的“ <code>tools.setCreatingBbox()</code> ”	PLACEHOLDER
<code>tools.setCursor()</code>	将指针设置为指定外观。
<code>tools.snapPoint()</code>	以一个点作为输入参数, 并返回一个新的点, 该点可能已针对最接近的几何对象被调整过或者可能已将该点贴紧 最接近的几何对象。

属性摘要

`Tools` 对象具有以下属性:

属性	描述
<code>tools.activeTool</code>	只读; 返回当前活动工具的 <a href="#">ToolObj 对象</a> 。
<code>tools.altIsDown</code>	只读; 一个布尔值, 它标识 <code>Alt</code> 键是否正被按下。
<code>tools.ctrlIsDown</code>	只读; 一个布尔值, 它标识 <code>Ctrl</code> 键是否正被按下。
<code>tools.mouselsDown</code>	只读; 一个布尔值, 它标识鼠标左按钮是否正被按下。
<code>tools.penDownLoc</code>	只读; 一个点, 它表示舞台上最后一次鼠标按下事件的位置。
<code>tools.penLoc</code>	只读; 一个点, 它表示鼠标的当前位置。
<code>tools.shiftIsDown</code>	只读; 一个布尔值, 它标识 <code>Shift</code> 键是否正被按下。
<code>tools.toolObjs</code>	只读; <code>ToolObj</code> 对象数组。

## tools.activeTool

可用性

Flash MX 2004。

**Tools 对象**

## 用法

```
tools.activeTool
```

## 描述

只读属性；返回当前活动工具的 **ToolObj** 对象。

## 示例

下面的示例在 **theTool** 变量中保存表示当前活动工具的对象：

```
var theTool = fl.tools.activeTool;
```

## tools.altIsDown

## 可用性

Flash MX 2004。

## 用法

```
tools.altIsDown
```

## 描述

只读属性；一个布尔值，它标识 **Alt** 键是否正被按下。如果 **Alt** 键被按下，则值为 **true**；否则为 **false**。

## 示例

下面的示例可确定 **Alt** 键是否正被按下：

```
var isAltDown = fl.tools.altIsDown;
```

## tools.constrainPoint()

## 可用性

Flash MX 2004。

## 用法

```
tools.constrainPoint(pt1, pt2)
```

## 参数

**pt1, pt2** 点，用于指定开始单击的点和拖动到的点。

## 返回

一个新的经调整或受到限制的点。

## 说明

方法；以两个点作为输入参数并返回一个新的经调整或受到限制的点。在运行该命令时，如果按下 **Shift** 键，则返回的点会限制为 **45°** 约束（对诸如带有箭头的线有用）或者限制对象保持其高宽比（例如用“矩形”工具拉出正方形）。

## 示例

下面的示例返回一个受限制的点：

```
pt2 = fl.tools.constrainPoint(pt1, tempPt);
```

## tools.ctrlIsDown

可用性

Flash MX 2004。

用法

```
tools.ctrlIsDown
```

描述

只读属性；一个布尔值，如果 **Ctrl** 键被按下，则为 **true**；否则为 **false**。

示例

下面的示例可确定 **Ctrl** 键是否正被按下：

```
var isCtrlDown = fl.tools.ctrlIsDown;
```

## tools.getKeyDown()

可用性

Flash MX 2004。

用法

```
tools.getKeyDown()
```

参数

无。

返回

键的整数值。

说明

方法；返回最近按下的键。

示例

下面的示例显示最近按下的键的整数值：

```
var theKey = fl.tools.getKeyDown();  
fl.trace(theKey);
```

## tools.mouselsDown

可用性

Flash MX 2004。

**Tools 对象**

## 用法

```
tools.mouseIsDown
```

## 描述

只读属性；一个布尔值，如果鼠标左按钮正被按下，则为 **true**；否则为 **false**。

## 示例

下面的示例可确定鼠标左按钮是否被按下。

```
var isMouseDown = fl.tools.mouseIsDown;
```

## tools.penDownLoc

## 可用性

Flash MX 2004。

## 用法

```
tools.penDownLoc
```

## 描述

只读属性；一个点，它表示舞台上最后一次鼠标按下事件的位置。**tools.penDownLoc** 属性由两个属性组成：**x** 和 **y**，它们分别与鼠标指针的 **x,y** 位置相对应。

## 示例

下面的示例可确定舞台上上一次鼠标按下事件的位置，并在“输出”面板中显示 **x** 和 **y** 值：

```
var pt1 = fl.tools.penDownLoc;  
fl.trace("x,y location of last mouseDown event was " + pt1.x + ", " + pt1.y)
```

## 另请参见

[tools.penLoc](#)

## tools.penLoc

## 可用性

Flash MX 2004。

## 用法

```
tools.penLoc
```

## 描述

只读属性；一个点，它表示鼠标指针的当前位置。**tools.penLoc** 属性由两个属性组成：**x** 和 **y**，它们分别与鼠标指针的 **x,y** 位置相对应。

## 示例

下面的示例可确定鼠标的当前位置：

```
var tempPt = fl.tools.penLoc;
```

另请参见

[tools.penDownLoc](#)

## tools.setCreatingBbox()

可用性

Flash 11。

用法

```
tools.setCreatingBbox()
```

参数

**Placeholder** 一个整数，它定义指针外观，如下面的列表所示：

返回

无。

描述

方法；将指针设置为指定外观。

示例

下面的示例将指针外观设置为黑箭头。

```
fl.tools.setCursor(1);
```

## tools.setCursor()

可用性

Flash MX 2004。

用法

```
tools.setCursor(cursor)
```

参数

**cursor** 一个整数，它定义指针外观，如下面的列表所示：

- 0 = 加号光标 (+)
- 1 = 黑色箭头
- 2 = 白色箭头
- 3 = 四向箭头
- 4 = 双向水平箭头
- 5 = 双向垂直箭头
- 6 = X
- 7 = 手型光标

**Tools 对象**

返回

无。

描述

方法；将指针设置为指定外观。

示例

下面的示例将指针外观设置为黑箭头。

```
fl.tools.setCursor(1);
```

## tools.shiftIsDown

可用性

Flash MX 2004。

用法

```
tools.shiftIsDown
```

描述

只读属性；一个布尔值，如果 **Shift** 键被按下，则为 **true**；否则为 **false**。

示例

下面的示例可确定 **Shift** 键是否正被按下。

```
var isShiftDown = fl.tools.shiftIsDown;
```

## tools.snapPoint()

可用性

Flash MX 2004。

用法

```
tools.snapPoint(pt)
```

参数

**pt** 指定要为其返回贴紧点的点的位置。

返回

一个可能经过调整或者已贴紧到最近的几何对象的新点。

说明

方法；以一个点作为输入参数，并返回一个新的点，该点可能已针对最近的几何对象被调整过或者可能已将该点贴紧最近的几何对象。如果 **Flash** 用户界面的“视图”菜单中贴紧选项已禁用，则返回点是原点。



**Tools 对象**

## 示例

下面的示例返回一个新的点，它可能已贴紧到最接近的几何对象。

```
var theSnapPoint = fl.tools.snapPoint(pt1);
```

## tools.toolObjs

## 可用性

Flash MX 2004。

## 用法

```
tools.toolObjs
```

## 描述

只读属性； ToolObj 对象的数组（请参阅 [ToolObj 对象](#)）。

## 第 48 章 : Vertex 对象

可用性

Flash MX 2004。

描述

**Vertex** 对象是形状数据结构中保存坐标数据的部分。

方法摘要

可以使用 **Vertex** 对象的以下方法：

方法	描述
<a href="#">vertex.getHalfEdge()</a>	获取共享该 <b>vertex</b> 的 <b>HalfEdge</b> 对象。
<a href="#">vertex.setLocation()</a>	设置 <b>vertex</b> 的位置。

属性摘要

**Vertex** 对象有以下可用属性：

属性	描述
<a href="#">vertex.x</a>	只读； <b>vertex</b> 的 <b>x</b> 坐标值（以像素为单位）。
<a href="#">vertex.y</a>	只读； <b>vertex</b> 的 <b>y</b> 坐标值（以像素为单位）。

### vertex.getHalfEdge()

可用性

Flash MX 2004。

用法

```
vertex.getHalfEdge()
```

参数

无。

返回

一个 **HalfEdge** 对象。

说明

方法；获取共享该 **vertex** 的 **HalfEdge** 对象。

示例

下面的示例说明如何获取共享同一顶点的其他半边缘：

## Vertex 对象

```
var shape = fl.getDocumentDOM().selection[0];
var hEdge = shape.edges[0].getHalfEdge(0);
var theVertex = hEdge.getVertex();
var someHEdge = theVertex.getHalfEdge(); // Not necessarily the same half edge
var theSameVertex = someHEdge.getVertex();
fl.trace('the same vertex: ' + theSameVertex);
```

## vertex.setLocation()

可用性

Flash MX 2004。

用法

```
vertex.setLocation(x, y)
```

参数

**x** 一个浮点值，它指定 **vertex** 放置位置的 **x** 坐标值（以像素为单位）。

**y** 一个浮点值，它指定 **vertex** 放置位置的 **y** 坐标值（以像素为单位）。

返回

无。

描述

方法；设置 **vertex** 的位置。在使用此方法之前，必须先调用 [shape.beginEdit\(\)](#)。

示例

下面的示例将顶点设置为原点：

```
var shape = fl.getDocumentDOM().selection[0];
shape.beginEdit();
var hEdge = shape.edges[0].getHalfEdge(0);
var vertex = hEdge.getVertex();
var someHEdge = vertex.getHalfEdge();
var vertex = someHEdge.getVertex();
// Move the vertex to the origin.
vertex.setLocation(0.0, 0.0);
shape.endEdit();
```

## vertex.x

可用性

Flash MX 2004。

用法

```
vertex.x
```

描述

只读属性；顶点的 **x** 坐标值（以像素为单位）。

**Vertex 对象**

## 示例

下面的示例在“输出”面板中显示顶点的 **x** 和 **y** 值。

```
var shape = fl.getDocumentDOM().selection[0];
var hEdge = shape.edges[0].getHalfEdge(0);
var vertex = hEdge.getVertex();

fl.trace('x location of vertex is: ' + vertex.x);
fl.trace('y location of vertex is: ' + vertex.y);
```

## vertex.y

## 可用性

Flash MX 2004。

## 用法

`vertex.y`

## 描述

只读属性；`vertex` 的 **y** 坐标值（以像素为单位）。

## 示例

请参阅 [vertex.x](#)。

## 第 49 章 : VideoItem 对象

继承关系 [Item 对象](#) > VideoItem 对象

可用性

Flash MX 2004。

描述

VideoItem 对象是 [Item 对象](#) 的子类。

方法摘要

除 [Item 对象](#) 的方法外，VideoItem 对象还具有以下方法：

属性	描述
<a href="#">videoItem.exportToFLV()</a>	将指定的项目导出到 FLV 文件。

属性摘要

除 [Item 对象](#) 的属性外，还可以对 VideoItem 对象使用以下属性：

属性	描述
<a href="#">videoItem.fileLastModifiedDate</a>	只读；一个字符串，其中包含一个十六进制数字，此数字表示从 1970 年 1 月 1 日至原始文件导入库时的文件修改日期（在磁盘上）之间的秒数。
<a href="#">videoItem.sourceFileExists</a>	只读；一个布尔值，它指定之前导入库中的文件现在是否仍位于其导入时的源位置。
<a href="#">videoItem.sourceFileIsCurrent</a>	只读；一个布尔值，它指定库项目的文件修改日期是否与其导入时在磁盘上的文件修改日期相同。
<a href="#">videoItem.sourceFilePath</a>	只读；一个字符串，它指定视频项目的路径。
<a href="#">videoItem.videoType</a>	只读；一个字符串，它指定项目所代表视频的类型。

### videoItem.exportToFLV()

可用性

Flash CS4 Professional.

用法

```
videoItem.exportToFLV(fileURI)
```

参数

**fileURI** 一个字符串，表示为 file:/// URI，它指定导出的文件的路径和名称。

返回

一个布尔值，如果成功导出文件，则为 true；否则为 false。

**VideoItem** 对象

## 描述

方法；将指定的项目导出到 FLV 文件。

## 示例

假定库中的第一个项目是视频项目，则下面的代码将该项目导出为 FLV 文件：

```
var videoFileURL = "file:///C:/out.flv";
var libItem = fl.getDocumentDOM().library.items[0];
libItem.exportToFLV(videoFileURL);
```

## videoItem.fileLastModifiedDate

## 可用性

Flash CS4 Professional.

## 用法

```
videoItem.fileLastModifiedDate
```

## 描述

只读属性：一个字符串，其中包含一个十六进制数字，此数字表示从 1970 年 1 月 1 日至原始文件导入库时的文件修改日期之间的秒数。如果该文件不再存在，则此值为 "00000000"。

## 示例

假定库中的第一个项目是视频项目，则下面的代码显示一个十六进制数字（如上所述）。

```
var libItem = fl.getDocumentDOM().library.items[0];
fl.trace("Mod date when imported = " + libItem.fileLastModifiedDate);
```

## 另请参见

[videoItem.sourceFileExists](#)、[videoItem.sourceFileIsCurrent](#)、[videoItem.sourceFilePath](#)、[FLfile.getModificationDate\(\)](#)

## videoItem.sourceFileExists

## 可用性

Flash CS4 Professional.

## 用法

```
videoItem.sourceFileExists
```

## 描述

只读属性：一个布尔值，如果在导入库中的文件的导入之前所在位置仍然存在此文件，则为 **true**；否则为 **false**。

## 示例

假定库中的第一个项目是视频项目，如果导入库中的文件仍然存在，则下面的代码显示 **"true"**。

```
var libItem = fl.getDocumentDOM().library.items[0];
fl.trace("sourceFileExists = " + libItem.sourceFileExists);
```

另请参见

[videoItem.sourceFileIsCurrent](#)、[videoItem.sourceFilePath](#)

## videoItem.sourceFileIsCurrent

可用性

Flash CS4 Professional.

用法

```
videoItem.sourceFileIsCurrent
```

描述

只读属性：一个布尔值，如果库项目的文件修改日期与磁盘中导入文件的修改日期相同，则为 **true**；否则为 **false**。

示例

假定库中的第一个项目是视频项目，如果导入的文件自导入以来未在磁盘中进行过修改，则下面的代码显示 **“true”**：

```
var libItem = fl.getDocumentDOM().library.items[0];  
fl.trace("fileIsCurrent = " + libItem.sourceFileIsCurrent);
```

另请参见

[videoItem.fileLastModifiedDate](#)、[videoItem.sourceFilePath](#)

## videoItem.sourceFilePath

可用性

Flash 8。

用法

```
videoItem.sourceFilePath
```

描述

只读属性；一个以 **file:///** URI 形式表示的字符串，它指定视频项的路径。

示例

下面的示例显示库中所有类型为 **video** 的项目的名称和源文件路径：

```
for (idx in fl.getDocumentDOM().library.items) {  
  if (fl.getDocumentDOM().library.items[idx].itemType == "video") {  
    var myItem = fl.getDocumentDOM().library.items[idx];  
    fl.trace(myItem.name + " source is " + myItem.sourceFilePath);  
  }  
}
```

另请参见

[videoItem.sourceFileExists](#)

## videoItem.videoType

可用性

Flash 8。

用法

videoItem.videoType

描述

只读属性；一个字符串，它指定项所代表的视频类型。可能的值为 "embedded video"、"linked video" 和 "video"。

示例

下面的示例显示库中所有类型为 **video** 的项目的名称和类型：

```
for (idx in fl.getDocumentDOM().library.items) {  
  if (fl.getDocumentDOM().library.items[idx].itemType == "video") {  
    var myItem = fl.getDocumentDOM().library.items[idx];  
    fl.trace(myItem.name + " is " + myItem.videoType);  
  }  
}
```



## 第 50 章 : XMLUI 对象

可用性

Flash MX 2004。

描述

Flash 8 支持以 XML 用户界面语言 (XUL) 的子集编写的自定义对话框。一个 XML 用户界面 (XMLUI) 对话框可由多个 Flash 功能 (如“命令”和“行为”) 使用, 这样就可以为采用可扩展性构建的功能提供用户界面。XMLUI 对象能够获取和设置 XMLUI 对话框的属性, 并能接受或取消其中的某个功能。XMLUI 方法可用于回调中, 例如, 按钮中的 `oncommand` 处理函数。

可以编写 `dialog.xml` 文件, 然后使用 `document.xmlPanel()` 方法从 JavaScript API 调用该文件。若要检索表示当前 XMLUI 对话框的对象, 请使用 `fl.xmlui`。

方法摘要

下列方法可用于 XMLUI 对象:

方法	描述
<code>xmlui.accept()</code>	以接受状态关闭当前 XMLUI 对话框。
<code>xmlui.cancel()</code>	以取消状态关闭当前 XMLUI 对话框。
<code>xmlui.get()</code>	检索当前 XMLUI 对话框的指定属性的值。
<code>xmlui.getControlItemElement()</code>	为指定控件返回当前控件项目。
<code>xmlui.setEnabled()</code>	返回一个布尔值, 它指定控件是处于启用状态, 还是禁用状态 (变暗)。
<code>xmlui.setVisible()</code>	返回一个布尔值, 它指定控件是可见还是隐藏。
<code>xmlui.set()</code>	修改当前 XMLUI 对话框的指定属性的值。
<code>xmlui.setControlItemElement()</code>	为当前项目设置标签和值。
<code>xmlui.setControlItemElements()</code>	设置当前项目的标签和值对。
<code>xmlui.setEnabled()</code>	启用或禁用 (变暗) 控件。
<code>xmlui.setVisible()</code>	显示或隐藏控件。

### xmlui.accept()

可用性

Flash MX 2004。

用法

```
xmlui.accept()
```

参数

无。

返回  
无。

描述  
方法；以接受状态关闭当前 XMLUI 对话框，这等效于用户单击“确定”按钮。

另请参见  
[fl.xmlui](#)、[document.xmlPanel\(\)](#)、[xmlui.cancel\(\)](#)

## xmlui.cancel()

可用性  
Flash MX 2004。

用法  
`xmlui.cancel()`

参数  
无。

返回  
无。

描述  
方法；以取消状态关闭当前 XMLUI 对话框，这等效于用户单击“取消”按钮。

另请参见  
[fl.xmlui](#)、[document.xmlPanel\(\)](#)、[xmlui.accept\(\)](#)

## xmlui.get()

可用性  
Flash MX 2004。

用法  
`xmlui.get(controlPropertyName)`

参数  
**controlPropertyName** 一个字符串，它指定要检索其值的 XMLUI 属性的名称。

返回  
一个字符串，它表示指定属性的值。如果需要返回布尔值 **true** 或 **false**，则返回字符串 **"true"** 或 **"false"**。

说明  
方法；检索当前 XMLUI 对话框的指定属性的值。

#### 示例

下面的示例返回名为 URL 的属性的值:

```
fl.xmlui.get("URL");
```

#### 另请参见

[fl.xmlui](#)、[document.xmlPanel\(\)](#)、[xmlui.getControlItemElement\(\)](#)、[xmlui.set\(\)](#)

## xmlui.getControlItemElement()

#### 可用性

Flash 8。

#### 用法

```
xmlui.getControlItemElement(controlPropertyName)
```

#### 参数

**controlPropertyName** 一个字符串，它指定要检索的控件项目元素的属性。

#### 返回

一个对象，它表示 **controlPropertyName** 指定的控件的当前控件项目。

#### 说明

方法；返回由 **controlPropertyName** 指定的 **ListBox** 或 **ComboBox** 控件中选定行的标签和值。

#### 示例

下面的示例返回 **myListBox** 控件中当前选定行的标签和值:

```
var elem = new Object();  
elem = fl.xmlui.getControlItemElement("myListBox");  
fl.trace("label = " + elem.label + " value = " + elem.value);
```

#### 另请参见

[fl.xmlui](#)、[document.xmlPanel\(\)](#)、[xmlui.get\(\)](#)、[xmlui.setControlItemElement\(\)](#)、[xmlui.setControlItemElements\(\)](#)

## xmlui.getEnabled()

#### 可用性

Flash 8。

#### 用法

```
xmlui.getEnabled(controlID)
```

#### 参数

**controlID** 一个字符串，它指定要检索其状态的控件的 ID 属性。

#### 返回

一个布尔值；如果控件处于启动状态，则为 `true`，否则为 `false`。

#### 说明

方法；返回一个布尔值，用于指定控件是处于启用状态，还是处于禁用状态（变暗）。

#### 示例

下面的示例返回一个值，该值指示 ID 属性为 `myListBox` 的控件是否处于启用状态：

```
var isEnabled = fl.xmlui.setEnabled("myListBox");  
fl.trace(isEnabled);
```

#### 另请参见

[fl.xmlui](#)、[document.xmlPanel\(\)](#)、[xmlui.setEnabled\(\)](#)

## xmlui.getVisible()

#### 可用性

Flash 8。

#### 用法

```
xmlui.getVisible(controlID)
```

#### 参数

**controlID** 一个字符串，它指定要检索其可见性状态的控件的 ID 属性。

#### 返回

如果控件可见，则返回布尔值 `true`；或者，如果控件不可见（隐藏），则返回布尔值 `false`。

#### 说明

方法；返回一个布尔值，它指定控件是可见还是隐藏。

#### 示例

下面的示例返回一个值，该值指示 ID 属性为 `myListBox` 的控件是否可见：

```
var isVisible = fl.xmlui.getVisible("myListBox");  
fl.trace(isVisible);
```

#### 另请参见

[xmlui.setVisible\(\)](#)

## xmlui.set()

#### 可用性

Flash MX 2004。

#### 用法

```
xmlui.set(controlPropertyName, value)
```

#### 参数

**controlPropertyName** 一个字符串，它指定待修改的 XMLUI 属性的名称。

**value** 一个字符串，它指定要为 XMLUI 属性设置的值。

#### 返回

无。

#### 描述

方法；修改当前 XMLUI 对话框的指定属性的值。

#### 示例

下面的示例将名为 URL 的属性的值设置为 `www.adobe.com`：

```
fl.xmlui.set("URL", "www.adobe.com");
```

#### 另请参见

[fl.xmlui](#)、[document.xmlPanel\(\)](#)、[xmlui.get\(\)](#)、[xmlui.setControlItemElement\(\)](#)、[xmlui.setControlItemElements\(\)](#)

## xmlui.setControlItemElement()

#### 可用性

Flash 8。

#### 用法

```
xmlui.setControlItemElement(controlPropertyName, elementItem)
```

#### 参数

**controlPropertyName** 一个字符串，它指定待设置的控件项目元素。

**elementItem** 具有名为 `label` 的字符串属性和名为 `value` 的可选字符串属性的 JavaScript 对象。如果 `value` 属性不存在，则系统将创建此属性，并为它分配与 `label` 相同的值。

#### 返回

无。

#### 描述

方法；为 `controlPropertyName` 指定的 `ListBox` 或 `ComboBox` 控件中当前选定行设置标签和值。

#### 示例

下面的示例为名为 `PhoneNumber` 的控件属性的当前项设置标签和值：

```
var elem = new Object();  
elem.label = "Fax";  
elem.value = "707-555-5555";  
fl.xmlui.setControlItemElement("PhoneNumber", elem);
```

另请参见

[fl.xmlui](#)、[document.xmlPanel\(\)](#)、[xmlui.getControlItemElement\(\)](#)、[xmlui.set\(\)](#)、[xmlui.setControlItemElements\(\)](#)

## xmlui.setControlItemElements()

可用性

Flash 8。

用法

```
xmlui.setControlItemElements(controlID, elementItemArray)
```

参数

**controlID** 一个字符串，它指定要设置控件的 ID 属性。

**elementItemArray** JavaScript 对象的数组，其中每个对象都具有名为 **label** 的字符串属性和名为 **value** 的可选字符串属性。如果 **value** 属性不存在，则系统将创建此属性，并为它分配与 **label** 相同的值。

返回

无。

描述

方法；清除由 **controlID** 指定的 **ListBox** 或 **ComboBox** 控件的值，并且使用由 **elementItemArray** 指定的 **label,value** 对来替换列表或菜单项。

示例

下面的示例将 ID 属性为 **myControlID** 的控件中项的标签和值设置为指定的 **label,value** 对：

```
var nameArray = new Array("January", "February", "March");
var monthArray = new Array();
for (i=0;i<nameArray.length;i++){
    elem = new Object();
    elem.label = nameArray[i];
    elem.value = i;
    monthArray[i] = elem;
}
fl.xmlui.setControlItemElements("myControlID", monthArray);
```

另请参见

[xmlui.getControlItemElement\(\)](#)、[xmlui.set\(\)](#)、[xmlui.setControlItemElement\(\)](#)

## xmlui.setEnabled()

可用性

Flash 8。

用法

```
xmlui.setEnabled(controlID, enable)
```

#### 参数

**controlID** 一个字符串，它指定要启用或禁用的控件的 ID 属性。

**enable** 如果要启用控件，则为布尔值 **true**；如果要禁用（变暗）控件，则为布尔值 **false**。

#### 返回

无。

#### 描述

方法；启用或禁用（变暗）控件。

#### 示例

下面的示例将使 ID 属性为 **myControl** 的控件变暗：

```
f1.xmlui.setEnabled("myControl", false);
```

另请参见

[xmlui.setEnabled\(\)](#)

## xmlui.setVisible()

#### 可用性

Flash 8。

#### 用法

```
xmlui.setVisible(controlID, visible)
```

#### 参数

**controlID** 一个字符串，它指定要显示或隐藏的控件的 ID 属性。

**visible** 如果要显示控件，则为布尔值 **true**；如果要隐藏控件，则为布尔值 **false**。

#### 返回

无。

#### 描述

方法；显示或隐藏控件。

#### 示例

下面的示例将隐藏 ID 属性为 **myControl** 的控件：

```
f1.xmlui.setVisible("myControl", false);
```

另请参见

[xmlui.setVisible\(\)](#)

## 第 51 章：C 级可扩展性

本章介绍 C 级可扩展性机制，该机制使您可以组合使用 JavaScript 和自定义的 C 代码来实现 Adobe Flash CS4 Professional 可扩展性文件。在 Flash 的此发行版中，该机制没有发生任何更改。

### 关于可扩展性

若要实现可扩展性，您可以使用 C 语言定义一些函数，将这些函数捆绑在一个动态链接库 (DLL) 或共享库中，将该库保存到适当的目录，然后使用 Adobe Flash JavaScript API 从 JavaScript 调用这些函数。

例如，您可能想要定义一个比 JavaScript 更有效地执行密集计算的函数以提高性能，或想要创建更高级的工具或效果。

此可扩展性机制是 Adobe Dreamweaver CS3 API 的一个子集。如果您熟悉该 API，则可能会认识此 C 级可扩展性机制 API 中的函数。但是，此 API 与 Dreamweaver API 存在以下几方面差异：

- 此 API 不包含 Dreamweaver API 中的所有命令。
- 在此 API 中，Dreamweaver API 中类型为 wchar\_t 和 char 的所有声明都作为 unsigned short 声明实现，以在传递字符串时支持 Unicode。
- 此 API 中的 JSVal JS\_BytesToValue() 函数未包含在 Dreamweaver API 中。
- DLL 或共享库文件必须存储在不同位置（请参阅第 488 页的“集成 C 函数”）。

### 集成 C 函数

C 级可扩展性机制使您可以组合使用 JavaScript 和 C 代码来实现 Flash 可扩展性文件。实现此功能的过程可概括为以下步骤：

- 1 使用 C 或 C++ 语言定义函数。
- 2 将这些函数捆绑在某一 DLL 文件 (Windows) 或共享库 (Macintosh) 中。
- 3 将该 DLL 文件或库保存到适当的位置：
  - Windows 7:  
引导驱动器 \Users\ 用户名 \AppData\Adobe\Flash CS5 或 Flash CS5.5\ 语言 \Configuration\External Libraries
  - Windows Vista:  
引导驱动器 \Users\ 用户名 \Local Settings\Application Data\Adobe\Flash CS5 或 Flash CS5.5\ 语言 \Configuration\External Libraries
  - Windows XP:  
引导驱动器 \Documents and Settings\ 用户名 \Local Settings\Application Data\Adobe\Flash CS5 或 Flash CS5.5\ 语言 \Configuration\External Libraries
  - Mac OS X:  
Macintosh HD/Users/ 用户名 /Library/Application Support/Adobe/Flash CS5 或 Flash CS5.5/ 语言 /Configuration/External Libraries
- 4 创建一个调用这些函数的 JSFL 文件。
- 5 在 Flash 创作环境中从“命令”菜单运行该 JSFL 文件。



有关详细信息，请参阅第 492 页的“[DLL 实现范例](#)”。

## C 级可扩展性和 JavaScript 解释程序

DLL 或共享库中的 C 代码在三个不同阶段与 Flash JavaScript API 交互：

- 启动时，注册库的函数
- 调用 C 函数时，解开从 JavaScript 传递给 C 的参数包
- 在 C 函数返回之前，打包返回值

为完成这些任务，解释程序定义了几种数据类型并公开了一个 API。本节列出的数据类型和函数定义显示在 `mm_jsapi.h` 文件中。要使库正常工作，必须使用下面一行内容使得库中每个文件的最上方包含 `mm_jsapi.h` 文件：

```
#include "mm_jsapi.h"
```

包含 `mm_jsapi.h` 文件就会包含 `mm_jsapi_environment.h` 文件，后者定义了 `MM_Environment` 结构。

若要获取 `mm_jsapi.h` 文件的副本，请从范例 ZIP 或 SIT 文件（请参阅第 492 页的“[DLL 实现范例](#)”）提取该副本，或者将下面的代码复制到命名为 `mm_jsapi.h` 的文件中：

```
#ifndef _MM_JSAPI_H_
#define _MM_JSAPI_H_

/*****
 * Public data types
 *****/

typedef struct JSContext JSContext;
typedef struct JSObject JSObject;
typedef long jsval;
#ifndef JSBool
typedef long JSBool;
#endif

typedef JSBool (*JSNative)(JSContext *cx, JSObject *obj, unsigned int argc,
jsval *argv, jsval *rval);

/* Possible values for JSBool */
#define JS_TRUE 1
#define JS_FALSE 0

/*****
 * Public functions
 *****/

/* JSBool JS_DefineFunction(unsigned short *name, JSNative call, unsigned int nargs) */
#define JS_DefineFunction(n, c, a) \
(mmEnv.defineFunction ? (*(mmEnv.defineFunction))(mmEnv.libObj, n, c, a) \
: JS_FALSE)

/* unsigned short *JS_ValueToString(JSContext *cx, jsval v, unsigned int *pLength) */
#define JS_ValueToString(c, v, l) \
(mmEnv.valueToString? (*(mmEnv.valueToString))(c, v, l) : (char *)0)

/* unsigned char *JS_ValueToBytes(JSContext *cx, jsval v, unsigned int *pLength) */
#define JS_ValueToBytes(c, v, l) \
(mmEnv.valueToBytes? (*(mmEnv.valueToBytes))(c, v, l) : (unsigned char *)0)

/* JSBool JS_ValueToInteger(JSContext *cx, jsval v, long *lp); */
```

```
#define JS_ValueToInteger(c, v, l) \
(mmEnv.valueToInteger ? (*(mmEnv.valueToInteger))(c, v, l) : JS_FALSE)

/* JSBool JS_ValueToDouble(JSContext *cx, jsval v, double *dp); */
#define JS_ValueToDouble(c, v, d) \
(mmEnv.valueToDouble? (*(mmEnv.valueToDouble))(c, v, d) : JS_FALSE)

/* JSBool JS_ValueToBoolean(JSContext *cx, jsval v, JSBool *bp); */
#define JS_ValueToBoolean(c, v, b) \
(mmEnv.valueToBoolean ? (*(mmEnv.valueToBoolean))(c, v, b) : JS_FALSE)

/* JSBool JS_ValueToObject(JSContext *cx, jsval v, JSObject **op); */
#define JS_ValueToObject(c, v, o) \
(mmEnv.valueToObject? (*(mmEnv.valueToObject))(c, v, o) : JS_FALSE)

/* JSBool JS_StringToValue(JSContext *cx, unsigned short *bytes, uint sz, jsval *vp); */
#define JS_StringToValue(c, b, s, v) \
(mmEnv.stringToValue? (*(mmEnv.stringToValue))(c, b, s, v) : JS_FALSE)

/* JSBool JS_BytesToValue(JSContext *cx, unsigned char *bytes, uint sz, jsval *vp); */
#define JS_BytesToValue(c, b, s, v) \
(mmEnv.bytesToValue? (*(mmEnv.bytesToValue))(c, b, s, v) : JS_FALSE)

/* JSBool JS_DoubleToValue(JSContext *cx, double dv, jsval *vp); */
#define JS_DoubleToValue(c, d, v) \
(mmEnv.doubleToValue? (*(mmEnv.doubleToValue))(c, d, v) : JS_FALSE)

/* jsval JS_IntegerToValue(long lv); */
#define JS_IntegerToValue(lv) (((jsval)(lv) << 1) | 0x1)

/* jsval JS_BooleanToValue(JSBool bv); */
#define JS_BooleanToValue(bv) (((jsval)(bv) << 3) | 0x6)

/* jsval JS_ObjectToValue(JSObject *obj); */
#define JS_ObjectToValue(ov) ((jsval)(ov))

/* unsigned short *JS_ObjectType(JSObject *obj); */
#define JS_ObjectType(o) \
(mmEnv.objectType ? (*(mmEnv.objectType))(o) : (char *)0)

/* JSObject *JS_NewArrayObject(JSContext *cx, unsigned int length, jsval *v) */
#define JS_NewArrayObject(c, l, v) \
(mmEnv.newArrayObject ? (*(mmEnv.newArrayObject))(c, l, v) : (JSObject *)0)

/* long JS_GetArrayLength(JSContext *cx, JSObject *obj) */
#define JS_GetArrayLength(c, o) \
(mmEnv.getArrayLength ? (*(mmEnv.getArrayLength))(c, o) : -1)

/* JSBool JS_GetElement(JSContext *cx, JSObject *obj, jsint idx, jsval *vp) */
#define JS_GetElement(c, o, i, v) \
(mmEnv.getElement ? (*(mmEnv.getElement))(c, o, i, v) : JS_FALSE)

/* JSBool JS_SetElement(JSContext *cx, JSObject *obj, jsint idx, jsval *vp) */
#define JS_SetElement(c, o, i, v) \
(mmEnv.setElement ? (*(mmEnv.setElement))(c, o, i, v) : JS_FALSE)

/* JSBool JS_ExecuteScript(JSContext *cx, JSObject *obj, unsigned short *script,
 * unsigned int sz, jsval *rval) */
#define JS_ExecuteScript(c, o, s, z, r) \
(mmEnv.executeScript? (*(mmEnv.executeScript))(c, o, s, z, (LPCTSTR)__FILE__, \
__LINE__, r) : JS_FALSE)
```

```

/* JSBool JS_ReportError(JSContext *cx, unsigned short *error, unsigned int sz) */
#define JS_ReportError(c, e, s) \
(mmEnv.reportError? (*(mmEnv.reportError))(c, e, s) : JS_FALSE)

/*****
 * Private data types, macros, and globals
 *****/

typedef struct {
JSObject *libObj;
JSBool (*defineFunction)(JSObject *libObj, unsigned short *name, JSNative call,
unsigned int nargs);
unsigned short *(*valueToString)(JSContext *cx, jsval v, unsigned int *pLength);
unsigned char *(*valueToBytes)(JSContext *cx, jsval v, unsigned int *pLength);
JSBool (*valueToInteger)(JSContext *cx, jsval v, long *lp);
JSBool (*valueToDouble)(JSContext *cx, jsval v, double *dp);
JSBool (*valueToBoolean)(JSContext *cx, jsval v, JSBool *bp);
JSBool (*valueToObject)(JSContext *cx, jsval v, JSObject **op);
JSBool (*stringToValue)(JSContext *cx, unsigned short *b, unsigned int sz, jsval *vp);
JSBool (*bytesToValue)(JSContext *cx, unsigned char *b, unsigned int sz, jsval *vp);
JSBool (*doubleToValue)(JSContext *cx, double dv, jsval *vp);
unsigned short *(*objectType)(JSObject *obj);
JSObject *(*newArrayObject)(JSContext *cx, unsigned int length, jsval *vp);
long (*getArrayLength)(JSContext *cx, JSObject *obj);
JSBool (*getElement)(JSContext *cx, JSObject *obj, unsigned int idx,
jsval *vp);
JSBool (*setElement)(JSContext *cx, JSObject *obj, unsigned int idx,
jsval *vp);
JSBool (*executeScript)(JSContext *cx, JSObject *obj, unsigned short *script,
unsigned int sz, unsigned short *file, unsigned int lineNum, jsval *rval);
JSBool (*reportError)(JSContext *cx, unsigned short *error, unsigned int sz);
} MM_Environment;

extern MM_Environment mmEnv;

// Declare the external entry point and linkage
#ifdef _WIN32
# ifndef _MAC
// Windows
__declspec( dllexport ) void MM_InitWrapper( MM_Environment *env, unsigned int envSize );
# endif
#else
extern void MM_InitWrapper( MM_Environment *env, unsigned int envSize );
#endif

#define MM_STATE\
/* Definitions of global variables */ \
MM_Environment mmEnv; \
\
void\
MM_InitWrapper(MM_Environment *env, unsigned int envSize) \
{ \

```

```
extern void MM_Init();\
\
char **envPtr = (char **)env; \
char **mmPtr = (char **)(&mmEnv);\
char **envEnd = (char **)((char *)envPtr + envSize);\
char **mmEnd = (char **)((char *)mmPtr + sizeof(MM_Environment)); \
\
/* Copy fields from env to mmEnv, one pointer at a time */\
while (mmPtr < mmEnd && envPtr < envEnd)\
*mmPtr++ = *envPtr++; \
\
/* If env doesn't define all of mmEnv's fields, set extras to NULL */ \
while (mmPtr < mmEnd) \
*mmPtr++ = (char *)0; \
\
/* Call user's MM_Init function */\
MM_Init();\
} \

#endif /* _MM_JSAPI_H_ */
```

## DLL 实现范例

本节演示如何生成一个简单的 DLL 实现。如果要了解该过程的工作原理而不必实际生成 DLL，您可以安装 Samples.zip 文件中提供的 DLL 范例文件；这些文件位于 ExtendingFlash/dllSampleComputeSum 文件夹中。（有关下载 Samples.zip 文件的信息，请参阅第 11 页的“实现范例”。）从 dllSampleComputeSum.dmg 或 dllSampleComputeSum.zip 文件中提取范例文件，然后执行下列操作：

- 将 Sample.jsfl 文件存储在 Configuration/Commands 目录中（请参阅第 2 页的“保存 JSFL 文件”）。
- 将 Sample.dll 文件存储在 Configuration/External Libraries 目录中（请参阅第 488 页的“集成 C 函数”）。
- 在 Flash 创作环境中，选择“命令”>“范例”。JSFL 文件中的 trace 语句会将 Sample.dll 中定义的函数的结果发送到“输出”面板中。

本节的其余部分讨论开发范例。在本例中，DLL 只包含一个将两个数字相加的函数。下面的示例显示了 C 代码：

```
// Source code in C
// Save the DLL or shared library with the name "Sample".
#include <windows.h>
#include <stdlib.h>

#include "mm_jsapi.h"

// A sample function
// Every implementation of a JavaScript function must have this signature.
JSBool computeSum(JSContext *cx, JSObject *obj, unsigned int argc, jsval *argv, jsval *rval)
{
    long a, b, sum;

    // Make sure the right number of arguments were passed in.
    if (argc != 2)
        return JS_FALSE;

    // Convert the two arguments from jsvals to longs.
    if (JS_ValueToInteger(cx, argv[0], &a) == JS_FALSE ||
        JS_ValueToInteger(cx, argv[1], &b) == JS_FALSE)
        return JS_FALSE;

    /* Perform the actual work. */
    sum = a + b;

    /* Package the return value as a jsval. */
    *rval = JS_IntegerToValue(sum);

    /* Indicate success. */
    return JS_TRUE;
}
```

编写这段代码后，生成 DLL 文件或共享库，然后将其存储在适当的 Configuration/External Libraries 目录中（请参阅第 488 页的“集成 C 函数”）。然后用以下代码创建一个 JSFL 文件，并将其存储在 Configuration/Commands 目录中（请参阅第 2 页的“保存 JSFL 文件”）。

```
// JSFL file to run C function defined above.
var a = 5;
var b = 10;
var sum = Sample.computeSum(a, b);
fl.trace("The sum of " + a + " and " + b + " is " + sum );
```

若要运行 DLL 中定义的函数，请在 Flash 创作环境中选择“命令”>“范例”。

## 数据类型

JavaScript 解释程序定义本节中描述的数据类型。

### typedef struct JSContext JSContext

一个指向此不透明数据类型的指针将传递给 C 级函数。该 API 中的某些函数接受此指针以作为它们的一个参数。

### typedef struct JSObject JSObject

一个指向此不透明数据类型的指针将传递给 C 级函数。此数据类型表示一个对象，该对象可能是数组对象或某一其他对象类型。

## typedef struct jsval jsval

一个不透明数据结构，可以包含一个整数，或包含一个指向浮点数、字符串或对象的指针。该 API 中的某些函数可以通过读取 jsval 结构的内容来读取函数参数值，某些函数可以用来通过写入 jsval 结构来写入函数的返回值。

## typedef enum { JS\_FALSE = 0, JS\_TRUE = 1 } JSBool

一个存储布尔值的简单数据类型。

## C 级 API

C 级可扩展性 API 由 JSBool (\*JSNative) 函数签名和以下函数组成：

- JSBool JS\_DefineFunction()
- unsigned short \*JS\_ValueToString()
- JSBool JS\_ValueToInteger()
- JSBool JS\_ValueToDouble()
- JSBool JS\_ValueToBoolean()
- JSBool JS\_ValueToObject()
- JSBool JS\_StringToValue()
- JSBool JS\_DoubleToValue()
- JSVal JS\_BooleanToValue()
- JSVal JS\_BytesToValue()
- JSVal JS\_IntegerToValue()
- JSVal JS\_ObjectToValue()
- unsigned short \*JS\_ObjectType()
- JSObject \*JS\_NewArrayObject()
- long JS\_GetArrayLength()
- JSBool JS\_GetElement()
- JSBool JS\_SetElement()
- JSBool JS\_ExecuteScript()

## typedef JSBool (\*JSNative)(JSContext \*cx, JSObject \*obj, unsigned int argc, jsval \*argv, jsval \*rval)

描述

方法；描述 JavaScript 函数在下列情况中的 C 级实现：

- cx 指针指向不透明的 JSContext 结构，该结构必须传递给 JavaScript API 中的某些函数。此变量包含解释程序的执行上下文。
- obj 指针指向在其上下文中执行脚本的对象。在运行脚本时，this 关键字等同于此对象。
- argc 整数是传递给函数的参数的数目。

- `argv` 指针指向 `jsval` 结构的数组。该数组的长度为 `argc` 个元素。
- `rval` 指针指向单个 `jsval` 结构。函数的返回值应写入 `*rval`。

如果成功，则函数返回 `JS_TRUE`；否则返回 `JS_FALSE`。如果函数返回 `JS_FALSE`，则停止执行当前脚本并显示一条错误信息。

## JSBool JS\_DefineFunction()

用法

```
JSBool JS_DefineFunction(unsigned short *name, JSNative call, unsigned int nargs)
```

描述

方法；在 Flash 中向 JavaScript 解释程序注册 C 级函数。在 `JS_DefineFunction()` 函数注册您在 `call` 参数中指定的 C 级函数；您可以在 JavaScript 脚本中通过使用在 `name` 参数中指定的名称引用该函数来调用它。`name` 参数区分大小写。

通常，此函数从 `MM_Init()` 函数调用，后者在 Flash 启动时调用。

参数

`unsigned short *name`、`JSNativecall`、`unsigned int nargs`

- `name` 参数是函数向 JavaScript 公开时的名称。
- `call` 参数是指向 C 级函数的指针。该函数必须返回一个表示成功或失败的 `JSBool`。
- `nargs` 参数是该函数预计要接收的参数的数目。

返回

一个布尔值：`JS_TRUE` 表示成功；`JS_FALSE` 表示失败。

## unsigned short \*JS\_ValueToString()

用法

```
unsigned short *JS_ValueToString(JSContext *cx, jsval v, unsigned int *pLength)
```

描述

方法；从 `jsval` 结构提取一个函数参数，将该参数转换为字符串（如果可能），然后将转换后的值传递回调用方。

注：请勿修改返回的缓冲区指针，否则可能会破坏 JavaScript 解释程序的数据结构。要更改字符串，必须将字符复制到其他缓冲区中，然后创建一个新的 JavaScript 字符串。

参数

`JSContext *cx`、`jsval v`、`unsigned int *pLength`

- `cx` 参数是传递给 JavaScript 函数的不透明 `JSContext` 指针。
- `v` 参数是要从中提取字符串的 `jsval` 结构。
- `pLength` 参数是一个指向无符号整数的指针。此函数将 `*pLength` 设置为与字符串的长度相等（以字节为单位）。

返回

一个指针，该指针在成功时指向一个以 `null` 结尾的字符串，失败时指向一个 `null` 值。在调用完成时，调用例程不得释放此字符串。

## JSBool JS\_ValueToInteger()

### 用法

```
JSBool JS_ValueToInteger(JSContext *cx, jsval v, long *lp);
```

### 描述

方法；从 jsval 结构提取一个函数参数，将该参数转换为整数（如果可能），然后将转换后的值传递回调用方。

### 参数

JSContext \*cx、jsval v、long \*lp

- cx 参数是传递给 JavaScript 函数的不透明 JSContext 指针。
- v 参数是要从中提取整数的 jsval 结构。
- lp 参数是一个指向 4 字节整数的指针。此函数将转换后的值存储在 \*lp 中。

### 返回

一个布尔值：JS\_TRUE 表示成功；JS\_FALSE 表示失败。

## JSBool JS\_ValueToDouble()

### 用法

```
JSBool JS_ValueToDouble(JSContext *cx, jsval v, double *dp);
```

### 描述

方法；从 jsval 结构提取一个函数参数，将该参数转换为双精度值（如果可能），然后将转换后的值传递回调用方。

### 参数

JSContext \*cx、jsval v、double \*dp

- cx 参数是传递给 JavaScript 函数的不透明 JSContext 指针。
- v 参数是要从中提取双精度值的 jsval 结构。
- dp 参数是一个指向 8 字节双精度值的指针。此函数将转换后的值存储在 \*dp 中。

### 返回

一个布尔值：JS\_TRUE 表示成功；JS\_FALSE 表示失败。

## JSBool JS\_ValueToBoolean()

### 用法

```
JSBool JS_ValueToBoolean(JSContext *cx, jsval v, JSBool *bp);
```

### 描述

方法；从 jsval 结构提取一个函数参数，将该参数转换为布尔值（如果可能），然后将转换后的值传递回调用方。



#### 参数

`JSContext *cx`、`jsval v`、`JSBool *bp`

- `cx` 参数是传递给 JavaScript 函数的不透明 `JSContext` 指针。
- `v` 参数是要从中提取布尔值的 `jsval` 结构。
- `bp` 参数是一个指向 `JSBool` 布尔值的指针。此函数将转换后的值存储在 `*bp` 中。

#### 返回

一个布尔值：`JS_TRUE` 表示成功；`JS_FALSE` 表示失败。

## JSBool JS\_ValueToObject()

#### 用法

```
JSBool JS_ValueToObject(JSContext *cx, jsval v, JSObject **op);
```

#### 描述

方法；从 `jsval` 结构提取一个函数参数，将该参数转换为对象（如果可能），然后将转换后的值传递回调用方。如果该对象是一个数组，请使用 `JS_GetArrayLength()` 和 `JS_GetElement()` 读取其内容。

#### 参数

`JSContext *cx`、`jsval v`、`JSObject **op`

- `cx` 参数是传递给 JavaScript 函数的不透明 `JSContext` 指针。
- `v` 参数是要从中提取对象的 `jsval` 结构。
- `op` 参数是一个指向 `JSObject` 指针的指针。此函数将转换后的值存储在 `*op` 中。

#### 返回

一个布尔值：`JS_TRUE` 表示成功；`JS_FALSE` 表示失败。

## JSBool JS\_StringToValue()

#### 用法

```
JSBool JS_StringToValue(JSContext *cx, unsigned short *bytes, uint sz, jsval *vp);
```

#### 描述

方法；将一个字符串返回值存储在 `jsval` 结构中。它分配一个新的 JavaScript 字符串对象。

#### 参数

`JSContext *cx`、`unsigned short *bytes`、`size_t sz`、`jsval *vp`

- `cx` 参数是传递给 JavaScript 函数的不透明 `JSContext` 指针。
- `bytes` 参数是要存储在 `jsval` 结构中的字符串。该字符串数据将被复制，因此调用方在不需要该字符串时应释放它。如果未指定该字符串的大小（请参阅 `sz` 参数），则该字符串必须以 `null` 结尾。
- `sz` 参数是该字符串的大小（以字节为单位）。如果 `sz` 为 0，则自动计算以 `null` 结尾的字符串的长度。
- `vp` 参数是指向 `jsval` 结构的指针，字符串的内容应复制到该结构中。

返回

一个布尔值: JS\_TRUE 表示成功; JS\_FALSE 表示失败。

## JSBool JS\_DoubleToValue()

用法

```
JSBool JS_DoubleToValue(JSContext *cx, double dv, jsval *vp);
```

描述

方法; 将一个浮点数返回值存储在 jsval 结构中。

参数

JSContext \*cx、double dv、jsval \*vp

- cx 参数是传递给 JavaScript 函数的不透明 JSContext 指针。
- dv 参数是一个 8 字节浮点数。
- vp 参数是指向 jsval 结构的指针, 该双精度值的内容应复制到该结构中。

返回

一个布尔值: JS\_TRUE 表示成功; JS\_FALSE 表示失败。

## JSVal JS\_BooleanToValue()

用法

```
jsval JS_BooleanToValue(JSBool bv);
```

描述

方法; 将一个布尔返回值存储在 jsval 结构中。

参数

JSBool bv

- bv 是一个布尔值: JS\_TRUE 表示成功; JS\_FALSE 表示失败。

返回

一个 JSVal 结构, 其中包含作为参数传递给函数的布尔值。

## JSVal JS\_BytesToValue()

用法

```
JSBool JS_BytesToValue(JSContext *cx, unsigned short *bytes, uint sz, jsval *vp);
```

描述

方法; 将字节转换为 JavaScript 值。

#### 参数

`JSText *cx`、`unsignedshort*bytes`、`uintsz`、`jval *vp`

- `cx` 参数是 JavaScript 上下文。
- `bytes` 参数是要转换为 JavaScript 对象的字节字符串。
- `sz` 参数是要转换的字节数目。
- `vp` 参数是 JavaScript 值。

#### 返回

一个布尔值：`JS_TRUE` 表示成功；`JS_FALSE` 表示失败。

## JSVal JS\_IntegerToValue()

#### 用法

```
jval JS_IntegerToValue(long lv);
```

#### 描述

方法；将一个长整型值转换为 JSVal 结构。

#### 参数

`lv`

`lv` 参数是要将其转换为 `jval` 结构的长整型值。

#### 返回

一个 JSVal 结构，其中包含作为参数传递给函数的整数。

## JSVal JS\_ObjectToValue()

#### 用法

```
jval JS_ObjectToValue(JSObject *obj);
```

#### 描述

方法；将一个对象返回值存储在 JSVal 中。使用 `JS_NewArrayObject()` 创建一个数组对象；使用 `JS_SetElement()` 定义其内容。

#### 参数

`JSObject *obj`

`obj` 参数是指向要转换为 JSVal 结构的 JSObject 对象的指针。

#### 返回

一个 JSVal 结构，其中包含作为参数传递给函数的对象。

## unsigned short \*JS\_ObjectType()

#### 用法

```
unsigned short *JS_ObjectType(JSObject *obj);
```

#### 描述

方法；如果给定一个对象引用，则返回该对象的类名称。例如，如果对象是 DOM 对象，该函数返回“Document”。如果对象是文档中的一个节点，该函数返回“Element”。对于数组对象，该函数返回“Array”。

注：请勿修改返回的缓冲区指针，否则可能会破坏 JavaScript 解释程序的数据结构。

#### 参数

JSObject \*obj

通常，使用 JS\_ValueToObject() 函数传递和转换该参数。

#### 返回

一个指向以 null 结尾的字符串的指针。在调用完成时，调用方不应释放此字符串。

## JSObject \*JS\_NewArrayObject()

#### 用法

```
JSObject *JS_NewArrayObject(JSContext *cx, unsigned int length [, jsval *v])
```

#### 描述

方法；创建一个包含 JSVals 数组的新对象。

#### 参数

JSContext \*cx、 unsigned int length、 jsval \*v

- cx 参数是传递给 JavaScript 函数的不透明 JSContext 指针。
- length 参数是数组可包含的元素数量。
- v 参数是指向要存储在数组中的 jsvals 的可选指针。如果返回值不是 null，则 v 是一个包含 length 个元素的数组。如果返回值是 null，则数组对象的初始内容未定义，并可使用 JS\_SetElement() 函数进行设置。

#### 返回

一个指针，指向新数组对象；如果调用失败，则指向值 null。

## long JS\_GetArrayLength()

#### 用法

```
long JS_GetArrayLength(JSContext *cx, JSObject *obj)
```

#### 描述

方法；如果给定一个指向数组对象的指针，则获取数组中元素的数量。

#### 参数

JSContext \*cx、 JSObject\*obj

- cx 参数是传递给 JavaScript 函数的不透明 JSContext 指针。
- obj 参数是一个指向数组对象的指针。

返回

返回数组中元素的数量；如果失败，则返回 -1。

## JSBool JS\_GetElement()

用法

```
JSBool JS_GetElement(JSContext *cx, JSObject *obj, jsint idx, jsval *vp)
```

描述

方法；读取数组对象的单个元素。

参数

JSContext \*cx、JSObject \*obj、jsint idx、jsval \*vp

- cx 参数是传递给 JavaScript 函数的不透明 JSContext 指针。
- obj 参数是一个指向数组对象的指针。
- idx 参数是数组的整数索引。第一个元素是索引 0，最后一个元素是索引 (length - 1)。
- vp 参数是指向一个 jsval 的指针，其中应复制数组内 jsval 结构的内容。

返回

一个布尔值：JS\_TRUE 表示成功；JS\_FALSE 表示失败。

## JSBool JS\_SetElement()

用法

```
JSBool JS_SetElement(JSContext *cx, JSObject *obj, jsint idx, jsval *vp)
```

描述

方法；写入数组对象的单个元素。

参数

JSContext \*cx、JSObject \*obj、jsint idx、jsval \*vp

- cx 参数是传递给 JavaScript 函数的不透明 JSContext 指针。
- obj 参数是一个指向数组对象的指针。
- idx 参数是数组的整数索引。第一个元素是索引 0，最后一个元素是索引 (length - 1)。
- vp 参数是指向一个 jsval 结构的指针，该结构的内容将复制到数组内的 jsval 中。

返回

一个布尔值：JS\_TRUE 表示成功；JS\_FALSE 表示失败。

## JSBool JS\_ExecuteScript()

用法

```
JS_ExecuteScript(JSContext *cx, JSObject *obj, unsigned short *script, unsigned int sz, jsval *rval)
```

#### 描述

方法；编译并执行 JavaScript 字符串。如果脚本生成返回值，则在 \*rval 中返回。

#### 参数

JSContext \*cx、JSObject \*obj、unsigned short \*script、unsigned intsz、jsval \*rval

- cx 参数是传递给 JavaScript 函数的不透明 JSContext 指针。
- obj 参数是一个指向在其上下文中执行脚本的对象的指针。在运行脚本时，this 关键字等同于此对象。通常，此对象是传递给 JavaScript 函数的 JSObject 指针。
- script 参数是包含 JavaScript 代码的字符串。如果未指定该字符串的大小（请参阅 sz 参数），则该字符串必须以 null 结尾。
- sz 参数是该字符串的大小（以字节为单位）。如果 sz 为 0，则自动计算以 null 结尾的字符串的长度。
- rval 参数是指向单个 jsval 结构的指针。该函数的返回值存储在 \*rval 中。

#### 返回

一个布尔值：JS\_TRUE 表示成功；JS\_FALSE 表示失败。