

Index

SYMBOLS AND NUMERICS

& (ampersand) as delimiter, 525. See also ampersand-delimited files

*** (asterisk)**

for comments (`/*` and `*/`), 26–27, 117
as wildcard for domain access, 566, 567
as wildcard with `import` statement, 238

: (colon) separating variable name from data type, 28

{ } (curly braces)

Auto Format preferences, 12, 118
automatic indentation and, 9
for code blocks, 9
after function declarations, 84
separate lines for, 118
surrounding compound statements, 20

> (greater than operator), 55

>= (greater than or equal to operator), 55–56

< (less than operator), 55

<= (less than or equal to operator), 55–56

. (period) as classpath, 11

| (pipe) as delimiter for data, 525

“ (quotation marks)

for `Boolean` values, avoiding, 33
indicating strings, 33
nesting JavaScript variables in strings and, 598

() (round brackets)

for forcing order of operator evaluation, 22
after function names, 84

; (semicolon) terminating statements, 19

' (single quotes), nesting JavaScript variables in strings and, 598

/ (slash) for comments, 26–27, 117

[] (square brackets) for setting or retrieving array elements, 41

_ (underscore), forcing folder order using, 144

9x9 blur matrix kernel, 341

zero

divide-by-zero error with pre-loaders, 200
as starting integer for array indexes, 40, 76

A

absolute addresses for URLs, 187

absolute references, 122, 178

absolute tweens, 404

acceleration. See easing

access functions for variables, 120–121

Accordion component, 233

Actions panel

customizing, 9–12
keyboard shortcuts, 762
overview, 8–9

ActionScript

ActionScript 2.0 settings, 11–12
defined, 1
preferences, 9–12
version for publishing SWF files, 14

activityLevel property

Camera class, 516
Microphone object, 492–494

addCallback() method, 612–613, 615, 616–617

addDropShadow() function, 348–349

addEventListener() method, 248, 262

addListener() method, 206–207

addNewRow() function

`addNewRow()` **function**, 624, 627–628

`addRequestHeader()` **method**

LoadVars class, 525–526

XML class, 538–539

`addRow()` **function**, 623, 626–627

`addToScore()` **function**, 119–120

`addTween()` **function**, 421, 422, 423

Alert component, 233, 286–287

`align` **property for justifying text**, 440

`allowDomain` **event**, 578

`allowDomain()` **method**

localConnection object, 581–582

`System.security.allowDomain()`, 573

`allowInsecureDomain` **event**, 578

`allowInsecureDomain()` **method**, 573

`_alpha` **property**

MovieClip class, 174

transparency and, 196–197

Video class, 507

ampersand (&) as delimiter, 525. *See also* **ampersand-delimited files**

ampersand-delimited files

creating using LoadVars, 528

loading, 529–531

overview, 525

XMLSocket class and, 553

`animateCircle()` **function**, 375, 376, 378

animation. *See also* **frame rate or fps (frames per second); Tween class**

adding random behavior, 382–383

choosing a frame rate or update interval for, 379–380

coordinate system for, 425–426

creating using ActionScript, 372–376

creating using keyframes, 370–371

creating with movie clips, 380–402

drawing API for, 425–432

easing in (acceleration), 391–395

easing in and out, interactive example, 396–402

easing out (deceleration), 387–391

elasticized box example, 411–414

frame rate and speed of, 377

frame-based versus time-based, 376–380

helicopter project, 396–402

masking effect for, 191–193

moving movie clips using `onEnterFrame()`, 372–374

moving movie clips using `setInterval()`, 374–376

of multiple movie clips, 381–382

playing tweens in parallel, 414–417

playing tweens in parallel and in sequence, 418–424

playing tweens in sequence, 417–418

screen saver example, 426–432

scripted versus non-scripted, 369–370

of single snowflake, 380–381

snow effect project, 380–386

speed versus smoothness and, 376–377

`animationFunction()` **function**, 374–375

anonymous functions, 87–88, 94–95

answers to exercises, 699–758

`antiAliasTest fla` **file**, 437–439

`antiAliasType` **text display property**

improving text readability, 438–439

overriding, 436

overview, 436–437

precise control example, 437–438

sharing fonts and, 452

AOL IM socket login negotiator, 554

`appendChild()` **method**, 538, 539

application state

managing with keyframes, 149–151

managing with script, 153–156

`applyFilter()` **method**, 356

arguments of constructors, 666

arguments of functions

defined, 84

delayed action, 88–89

modified by functions, 97–98

passing functions as, 88–92

strong typing for, 129–130

variable names for, 84

Array data type, 34

arrays. *See also* **associative arrays**

adding elements to the end, 42–43

applying multiple filters using, 346–349

assigning value to non-zero element first, avoiding, 41

assigning values to elements, 41

associative arrays versus, 46

combining two arrays, 42–43

`concat()` method, 42–43

converting strings to, 43

converting to strings, 43

data tracking example, 73–76

data types and, 40

defined, 40

deleting elements, 45

`for` loops for managing, 67–68, 76

index, 40

`join()` method, 42, 43

length property, 41

methods (table), 42

off-by-one errors with loops, 76

overview, 40–42

populating, 40–41, 43

`push()` method, 41, 42, 43
 replacing elements, 116
 screen saver example, 429
 setting or retrieving elements, 41
`sort()` method, 42, 43–44
`splice()` method, 42, 45
`split()` method, 43
 string output from, 43
 syntax, 40, 41
ASFunction **method, 452–453**
ASFunctionTest.fla **file, 453**
assignment. See also populating arrays
 assigning data to variables, 28
 calculate-and-assign operators, 21–22
 constants and, 30
 data types and, 35
 of IDs to dynamically created movie clips, 126–129
 passing data between variables, 29, 35–36
 by value versus by reference, 35–36
associative arrays. See also arrays
 for animating multiple movie clips, 381–382
 arrays versus, 46
 data tracking example, 73–76
 defined, 45
 for `..in` loops with, 70
 IDs for rows, 75
 index, 46
 for listener object, 252, 254
 lookup table example, 47–48
 for movie clips, 48–49
 as Object class instances, 46
 overview, 45–46
 photo viewer example, 49–50
 referencing movie clips using, 179, 181
 syntax, 46
 with `this` keyword, 48
associativity of operators
 for common operators (table), 23–24
 defined, 22
asterisk (*)
 for comments (`/*` and `*/`), 26–27, 117
 as wildcard for domain access, 566, 567
 as wildcard with `import` statement, 238
asynchronous communication, 524
`attachBitmap()` **method, 160, 161–162**
`attachMovie()` **method**
 basic form, 162
 `createEmptyMovieClip()` method versus, 182
 creating a simple button, 182–184

 creating movie clip copy with linkage ID, 162
 linkage ID required for, 182
 Object parameter, 182
 overview, 153, 160, 162
 syntax, 153
`attachSound()` **method, 475, 476**
`attachVideo()` **method**
 Camera class `get()` method with, 515
 Video class, 506
attributes **property, 541**
attributes (XML), 536, 552
audio. See sound
authoring time, layers and, 8
authoring tool
 creating animations using, 370–371
 scripted animation versus using, 369–370
Auto Format preferences, 12–13, 118
automated transitions, 404–405. See also easing;
 Tween class
automatic indentation option, 9
available **property, 615**
.avi files
 converting to Flash video, 501–502
 exporting Flash video to, 503
 importing, 503

B

BackButton component, 232
bandwidth **property, 516**
.bat files, fscommand() no longer launching, 656
`beginBitmapFill()` **method, 322–323, 334**
`beginFill()` **method (drawing API)**
 alpha value, 326–327
 Bezier curve example, 328, 329
 filling a square, 326–327
 graph building example, 330, 331, 332
 overview, 322
 parameters (table), 322
`beginFill()` **method (movie clip), 160, 162**
`beginGradientFill()` **method, 323–324, 334**
best practices
 accessing variables in another timeline, 122–129
 commenting, 116–118
 for creating custom functions, 129–133
 defined, 107
 formatting code, 118–119
 variable naming, 107–110
 variable scope, 119–122
 variable typing, 110–116

BevelFilter, 338

Bezier curves, drawing, 325, 327–329

binding. See data binding

bitmap caching, 196

bitmap images. See also Bitmap object; pixels

checking if loaded, 472

ColorMatrixFilter for modifying colors, 344–345

converting movie clips to, 355, 366

declaring images as, 355

dissolving pixels to defined source, 363–364

embedding into movie clips, 145, 147, 148

embedding versus loading dynamically, 144–145

grid structure of, 355

HTML text field support for, 448

importing to library, 146–147

loading into movie clips, 186, 471, 472

manipulating bitmap data, 366–368

merging two images, 361

naming convention for, 145

organizing in the library, 145, 146

pinhole camera look for, 349–351

placing on the timeline, avoiding, 145

pre-loading, 472, 473–474

project folder for, 145

working with, 146–148

Bitmap object. See also bitmap images

attaching to movie clip, 161–162

described, 161

display class package for, 355

linking to movie clip, 360

methods, 356–365

overlapping, detecting, 360

pixel control and, 319

removing all attributes, properties, and data, 357–358

`bitmapBlur fla` file, 366

bitmapData object properties, 365–366

blending modes

applying, 352–353

overview, 319

performance and, 197

`blendMode` property, 175, 352–353

block comments, 26–27, 117

blur effect

BlurFilter for, 337

ConvolutionFilter for, 340–343

BlurFilter

described, 337

pinhole camera look using, 349–351

properties, 337

Boolean data type

described, 34

strong typing example, 31–33

values possible for, 33

variable names, 108

break statements in switch...case statements, 62

`browse()` method

overview, 637, 638

uploading files, 650

browsers

creating communication between movies and, 610–612

`fscommand()` method and, 604

JavaScript capabilities for, 597

opening windows from Flash movies, 618–619, 629–634

targeting HTML wrappers for, 15

BufferingBar component, 232

`bufferLength` event, 505

`bufferTime` event, 505

Button component

for controlling sound objects, 480–482

described, 233

setting styles for, 282

simple audio interface example, 487, 489

skinning, 291–294

video controls example, 509

buttons

background color for clickable area, 189–190

creating a simple button, 6–7, 179–181, 182–184

creating thumbnail buttons, 187–190

event listener for radio buttons, 250–254

event listener for simple button, 249–250

overview, 5

states available, 5

timelines for, 5

`bytesLoaded` event, 505

`bytesTotal` event, 505

C

`cacheAsBitmap` property, 175, 196

calculate-and-assign operators, 21–22

`call()` method, 614, 618, 619, 630

`callExternalInterface()` function, 615, 617

calling functions

examples, 84–85

JavaScript functions using External API, 617–620

JavaScript functions with `call()`, 614

JavaScript functions, with integration kit, 608–609

- JavaScript functions within Flash, 598–599, 600–602
- from JavaScript, using External API, 616–617
- from JavaScript, using integration kit, 609–612
- projector functions using `fscommand()`, 656–658
- registering ActionScript functions as callable from JavaScript, 612–613, 615, 616–617
- syntax, 84
- calling methods**
 - from JavaScript, using External API, 616–617
 - JavaScript, within Flash, 598–599
 - within the same class, 669
 - static methods, 682–683
- `callNextTween()` **function**, 421–422, 424
- Camera class, 515–516**
- Camera object**
 - creating, 516–517
 - displaying video on the stage, 517–521
 - overview, 515
 - security and, 522
- `cancel()` **method**, 637, 638
- Cascading Style Sheets. See CSS**
- case**
 - class names and, 106, 110
 - data type names and, 110
 - function names and, 84, 110
 - sorting arrays and, 44
 - variable names and, 29, 106, 110
- casting types, 112–113, 116**
- CDATA tag (XML), 537**
- CheckBox component, 233**
- `checkBuffer()` **function**, 510–511
- `checkKeys()` **function**, 397, 398–399, 400, 401–402
- `childNodes` **property**, 541
- classes. See also custom classes; specific classes**
 - as basis of OOP, 661
 - built-in easing classes, 404–405
 - component support classes, 237
 - components versus, 215
 - constructors, 662, 665–669
 - for CSS styles, defining, 458
 - for data types, 662
 - defined, 103
 - displaying properties and methods for, 37
 - importing, 238–239
 - inheritance, 106
 - naming conventions, 106, 110
 - objects versus, 103–106
 - reserved names (table), 110
 - search order for, 11
- classpath, 11–12**
- `clear()` **method**
 - drawing API, 325
 - movie clip, 160, 163
 - SharedObject class, 582, 583
 - Video class, 506
- `clearInterval()` **function**, 548
- `clone()` **method**, 356
- `cloneNode()` **method**, 538, 539
- cloning filters, 346**
- `close()` **method**
 - LocalConnection class, 578
 - netStream class, 504
 - XMLSocket class, 553
- code hints**
 - defined, 10
 - delay before displaying, 10
 - for filter constructor parameters, 335
 - strict typing for triggering, 10
 - strong typing and, 111
 - suffixes for triggering manually, 10
- coding. See also OOP (object-oriented programming)**
 - accessing variables in another timeline, 122–129
 - adding code to keyframes, 8, 9
 - adding to symbol instances, avoiding, 8–9
 - Auto Format preferences, 12–13
 - automatic indentation option, 9
 - best practices, defined, 107
 - code hints options, 10
 - color preferences, 11
 - commenting out code, 27, 117–118
 - comments in, 26–27, 116–118
 - custom function recommendations, 129–133
 - duplicating code, avoiding, 133
 - font options, 10
 - formatting code, 118–119
 - hard-coded movie clip references, avoiding, 130
 - keeping code in external files, 151–153
 - managing screen state with script, 153–156
 - placing components on the stage with scripts, 215, 237–241
 - refactoring problematic code, 132, 133–140
 - scripting components, 242–244
 - self-documenting code, 30, 116, 304
 - setting startup parameters, 235
 - tab size option, 10
 - variable naming, 107–110
 - variable scope, 119–122
 - variable typing, 110–116
 - white space in, 25–26, 118–119
- collections of data. See arrays; associative arrays**

colon (:): separating variable name from data type

colon (:): separating variable name from data type, 28

ColorMatrixFilter

- applying multiple filters, 347–349
- `colorTransform()` method versus, 356
- matrices for, 343–344
- modifying bitmap colors using, 344–345
- pinhole camera look using, 349–351

colors. *See also* fills

- background for button clickable area, 189–190
- BevelFilter properties for, 338
- blue variables in editor, 29
- ColorMatrixFilter for modifying, 344–345
- `colorTransform()` method for transforming, 356
- getting RGB color and alpha value for pixel, 360
- getting RGB color for pixel, 359–360
- global component styles for fonts, 282
- GlowFilter for ambient color, 337–338
- GradientBevelFilter properties for, 339–340
- GradientGlowFilter for banded ambient color, 339
- `paletteMap()` method for manipulating channels, 362
- preferences for code, 11
- replacing based on testing pixels, 365
- swatches on Tools panel, 6

`colorTransform()` method, 356

ComboBox component, 233, 277

comments

- best practices, 116–118
- block, 26–27, 117
- commenting out code, 27, 117–118, 312–313
- explaining issues with the code, 117
- good coding not replaced by, 116
- inline or single-line, 26, 117
- self-documenting code, 30, 116

compile-time bugs, 297–298

Component Inspector panel

- Bindings tab, 273, 274, 275–277
- opening, 273
- Parameters tab, 273, 274
- Schema tab, 273, 275, 277, 278, 279

`componentReady()` function, 623, 627

components. *See also* event listeners

- adding to the library, 237
- attaching multiple listeners to multiple components, 262–270
- classes versus, 215
- controlling appearance of, 281–294
- data binding, 273, 275–277
- data components (table), 231
- data types representing, 34

- embedded, controlling with JavaScript, 621–629
- file size and, 237
- FLV Playback component, 231–232
- FLV Playback Custom UI components (table), 232
- handling multiple events from same component, 248
- inheritance and, 106
- media components (table), 233
- new features in version 2.0, 229–230
- placing in the library, 215
- placing on the stage manually, 234–237
- placing on the stage with script, 215, 237–241
- registering event listeners to, 248
- reserved names (table), 110
- scripting, 242–244
- setting startup parameters, 235
- skinning, 281
- styles for, 216–217, 281, 282–290
- support classes, 237
- user interface (UI) components (table), 233–234
- XML file as data source for, 277–281

Components panel

- data components, 230–231
- dragging components to the stage, 234–237
- FLV Playback component, 231–232
- FLV Playback Custom UI components, 231–232
- media components, 233
- opening, 89
- placing components in library from, 215
- user interface (UI) components, 233–234

composite data types

- defined, 34
- duplicating data for, 36–37
- primitive data types versus, 34–35

`compositeFLVvideo.flv` file, 511–513

compositing Flash video with movie clips, 510–513

composition

- data queue example, 690–695
- spin effect example, 687–690
- subclassing versus, 686
- uses for, 687

compound statements, 20

compressing SWF files, 15

`concat()` method, 42–43

conditionals. *See also* expressions; loops

- data tracking example, 73–76
- defined, 53
- elements of, 53–54
- `if..then..else` statement, 59–60
- `switch..case` statement, 60–66
- `connect()` function, 327, 328

- `connect()` **method**
drawing API, 328, 329
for local connection between two SWF files, 580
LocalConnection class, 578
XMLSocket class, 553
- constants, 30**
- constructors**
arguments, 666
compiler recognition of, 665
defined, 662
passing startup data, 665–669
private properties in, 666
- `contentType` **property, 541**
- `continueTo()` **method, 405–406**
- Control menu keyboard shortcuts, 762, 767–768**
- converting. See also exporting; importing**
arrays to strings, 43
content to symbols, 4
data types (type casting), 112–113
Flash video to SWF, 500
movie clips to bitmap images, 355, 366
rasterizing vector graphics, 355
strings to arrays, 43
video to Flash video, 501–502
- ConvolutionFilter**
described, 340
Gaussian blur matrix 9x9 kernel, 341
known Gaussian blur matrix example, 342–343
matrices or kernels for, 340–341
properties, 341–342
- cookies**
domain policies and, 563
shared objects as, 582, 585–587
- coordinate systems, 425–426**
- `copyChannel()` **method, 357**
- copying. See also duplicating; passing data between variables**
arrays when sorting, 44
bitmaps using `clone()` method, 356
cloning filters, 346
color channel information, 357
movie clips with `duplicateMovieClip()`, 163–164
pixels using `copyPixels()` method, 357
pixels using `draw()` method, 358
text field formats, 440
- `copyPixels()` **method, 357**
- `countWords` **static method, 685**
- `createButton()` **function**
modifying to provide path to image, 188
passing functions as arguments example, 90–91, 92
refactoring example, 135, 136, 137, 139–140
for simple button, 181, 184
thumbnail buttons example, 187–189
- `createClassObject()` **method**
attaching components to the stage, 215, 238–239, 240, 241
basic form, 238
setting properties on separate lines, 239
- `createElement()` **method, 538, 539**
- `createEmptyMovieClip()` **method**
`attachMovie()` method versus, 182
basic form, 163
graph building example, 330, 332
overview, 160, 163
parent movie clip for, 178
programmatically control and, 178
referencing movie clips, 178–179
- `createSquare()` **function, 327, 329**
- `createTextField()` **method**
basic form, 163
changing defaults if not used, 446
creating text fields on-the-fly, 434–436
overview, 160, 163
- `createTextNode()` **method, 538, 539**
- cross-domain security**
domain policies, 563–570
local versus remote files and, 562–563
proxies, 570–571
shims, 570, 571–573
- `crossdomain.xml` **file**
policy file for remote domain, 565–566
policy file for single domain, 565
secure attribute, 568
wildcard for domain access, 566–567
- CSS (Cascading Style Sheets)**
associating a StyleSheet with a text field, 454–455
cascade mechanism, 461
creating a StyleSheet object, 454
CSS1 properties and Flash equivalents, 456
defining classes, 458
defining custom selectors, 458
defining multiple selectors, 458
defining styles directly on a StyleSheet object, 456–458
defining tags, 458
loading data via LoadVars object, 455
overview, 453
pseudo-classes supported, 459
with XML, 459–461
- CSSStyleDeclaration class, 283**

cssTest.fla file, 454–455

CuePoints for Flash video, 500, 510

curly braces ({ })

- Auto Format preferences, 12, 118
- automatic indentation and, 9
- for code blocks, 9
- after function declarations, 84
- separate lines for, 118
- surrounding compound statements, 20

currentFps **event, 505**

currentFps **property, 516**

_currentframe **property, 174**

curves, drawing, 325, 327–329

curveTo() **method**

- drawing a Bezier curve, 327–329
- overview, 325
- parameters (table), 325
- setting control and anchor points for, 328

custom classes

- composition, 686–695
- constructors, 662, 665–669
- creating, 661–663
- defining methods for, 669–675
- defining properties for, 675–681
- HelloWorldClass example, 662–663
- managing data with, 121–122
- MyClass example, 661–662
- public versus private properties and methods, 663–665
- setting styles using, 283, 287
- SpinEffect example, 687–690
- static methods for, 682–686
- StringLib example, 683–686
- subclassing, 681–682
- ThumbnailButton example, 666–669, 670–675, 677–681

custom functions

- best practices, 129–133
- declaring return values for, 130–131
- duplicating code, avoiding, 133
- entanglement, avoiding, 131–132
- hard-coded movie clip references, avoiding, 130
- key detect example, 131–132
- oversized, avoiding, 131
- refactoring problematic code, 132, 133–140
- strong typing for arguments, 129–130

D

data binding

- bindable properties and, 277
- Component Inspector panel for, 275–277

defined, 273

direction of, 276

using XML file as data source, 277–281

ways of setting up, 273

data latency, 524, 545

data queue example, 690–695

data sources

XML files as, 277–281

XML schema for, 277, 278, 279, 281

data types. See also strong typing; specific data types

arrays and, 40

assignment and, 35

built-in data types, 33–34

case of names, 110

class-based versions, 662

compiler errors for wrong data type, 85

components represented by, 34

composite, defined, 34

for constants, 30

declaring variables and, 28

defined, 30

duplicating data for composite data types, 36–37

primitive, defined, 34

primitive versus composite, 34–35

shared objects and, 584

suffixes suggested for (table), 111–112

type casting, 112–113

dataFile1.xml **file, 692**

dataFile2.xml **file, 692**

dataFile3.xml **file, 692**

DataGrid component, 233, 277, 282

dataHandler() **function, 691–692, 694**

DataHolder component, 231

DataSet component, 231

dataThumbnail.fla **file, 448–450**

Date class, displaying methods of, 37–38

Date data type, 34

DateChooser component, 233, 275–277, 287

DateField component, 233, 287

dates

assigning as labels to text field, 116

comparing, 59

deblocking Flash video, 513–514

deblocking **method, 514**

deblocking **property, 506**

Debugger panel

image loader code example, 310–311

overview, 306–308

practice using, 308–310

debugging

allowing for SWF files, 15

analyzing program execution and output, 313–314

- by commenting out code, 312–313
- compile-time bugs, 297–298
- Debugger panel for, 306–310
- developing a theory, 305–306
- developing for, 303–304
- domain policies and, 564–565
- fence post errors, 301
- global variable challenges for, 95, 119
- infinite loops, 78–79, 302
- listing possible errors, 314
- logic bugs, 298–299
- nested quotes and, 598
- numerical precision errors, 302–303
- off-by-one errors, 76, 301
- readable code as aid to, 303–304
- revising the theory, 305
- running an experiment, 306–313
- side effects and challenges for, 96
- small functions as aid to, 304
- steps in process of, 305
- strong typing and, 31–33
- timeline variable challenges for, 119
- `trace()` function for, 311–312, 313
- uninitialized variables, 299–301
- variable names and, 30
- deceleration. See easing**
- declaring**
 - anonymous functions, 87–88
 - filter properties, 335
 - functions, 84–85
 - images as bitmaps, 355
 - return types for custom functions, 130–131
 - variables, 27–28, 31, 119–120
- `decode()` **method, 525**
- `decrementNumLives()` **function, 119–120**
- `defineCSSwithAS.flc` **file, 457–458**
- delay before displaying code hints, 10**
- `deleteRows()` **function, 624, 628**
- deleting or removing**
 - array elements, 45
 - Bitmap object attributes, properties, and data, 357–358
 - clearing graphics, 160, 163, 325
 - directories from classpath, 11
 - loaded content, 173–174
 - movie clip from stage, 170
 - replacing array elements, 45
 - symbols from Library, 5
 - unloading movie clips, 208–209
- delimiters, 525**
- depths**
 - encapsulated in Movie Clip instances, 8
 - exported layers converted to, 8
 - `getNextHighestDepth()` method, 179
 - for movie clips, 179
 - for programmatically added content, 8
- deringing Flash video, 514**
- detecting**
 - Flash Player version, 16–17
 - motion, 518
 - overlapping Bitmap objects, 360
 - overlapping movie clips, 167–168
- developing for debugging, 303–304**
- development environment. See Flash IDE (Integrated Development Environment)**
- directories. See folders or directories**
- disabling keyboard shortcuts for test environment, 66**
- disk space, managing for shared objects, 587–588**
- DisplacementMapFilter, 345–346**
- displaying. See also text display properties**
 - Camera feed on the stage, 517–521
 - code hints, delay before, 10
 - Date class methods, 37–38
 - HTML in text fields, 446–448
 - methods for ActionScript classes, 37
 - properties for ActionScript classes, 37
 - variable contents, 28–29
 - XML in text fields, 446–448
- `dispose()` **method, 357–358**
- divide-by-zero error, pre-loading movies and, 200**
- `docTypeDecl` **property, 541**
- `dofsccommand.html` **file, 604, 605**
- DOM (Document Object Model) for XML, 549**
- `domain()` **method, 578**
- domain policies**
 - debugging and, 564–565
 - HTTPS and, 567–568
 - knowing when required, 564
 - loading from subfolder, 568–569
 - overview, 563
 - relative file paths versus, 564
 - secure attribute, 568
 - setup for remote domain, 565–566
 - setup for single domain, 565
 - Web services security and, 567–570
 - wildcard to allow domain access, 566, 567
 - with XML Socket, without HTTP, 569–570
- domains**
 - domain policies, 563–566
 - Flash Security Sandbox and, 563

domains (continued)

super domain, defined, 563
Web services security and, 567–570
wildcard to allow access, 566, 567

dot syntax
formats, 37
overview, 37–38, 102
for variables in nested movie clips, 122

double quotes. See quotation marks (“)

do..while loops, 72–73

Down state of buttons, 5, 7

download() method, 637, 639

downloading files
base code for, 641
example, 641–645
previous Flash versions and, 637

draw() method, 358, 366

drawBorder property, 680

drawBox() function
elasticized box example, 412, 414, 415–416
refactoring example, 136–137, 138, 140

drawCircle() function, 194, 195

drawDropShadow property, 681

drawGraph() function, 330, 331–333

drawing. See also specific methods
API for, 320
Bezier curves, 325, 327–329
clearing all graphics, 160, 163, 325
creating filled rectangles, 358
filling last closed shape, 160, 164, 324
filling shapes with bitmap contents, 322–323, 334
filling shapes with gradients, 323–324, 334
filling shapes with solid color, 160, 162, 322
graphs, 329–333
methods, 320–325
moving the drawing position, 161, 170, 324
setting line properties, 160, 168–169, 320–322
simple square, 326–327
straight lines, 160, 169, 325

drawing API. See also drawing
animating with, 425–432
keyboard shortcuts, 763–769
overview, 320

drawScreen() function, 241

drawThumbnail() method, 669, 671–672, 673–674, 678–679

DropShadowFilter
applying multiple filters, 347–349
described, 336
properties, 336–337

_droptarget property, 174

duplicateMovieClip() method, 160, 163–164

duplicating. See also copying
avoiding for code, 133
data for composite data types, 36–37
refactoring problematic code and, 133–140

duration property
Sound class, 480
Tween class, 410

dynamic text fields, creating, 435–436. See also text fields

E

easing
built-in easing classes and methods, 404–405
calculating movie clip positions, 388, 392
defined, 387
easing in (acceleration), 391–395
easing out (deceleration), 387–391
interactive example easing out and in, 396–402

Edit menu
keyboard shortcuts, 761, 763–764
Preferences command, 1, 9

elasticized box example
bouncing into position, 411–414
multiple tweens with, 414–417

Embed tag
adding FlashVars to, 590–591
allowScriptAccess parameter, 603, 619, 628
nesting within Object tag, 589
parameters, 589–590

enabled property, 175

endFill() method (drawing API)
Bezier curve example, 327
graph building example, 330, 331, 333
overview, 324

endFill() method (movie clip), 160, 164

entanglement
avoiding, 131–132
refactoring problematic code, 133–140

erasing. See deleting or removing

errors. See also debugging
compiler errors for wrong data type, 85
compile-time bugs, 297–298
fence post errors, 301
infinite loops, 77–79, 302
listing possible errors, 314
logic bugs, 298–299
numerical precision errors, 302–303

- off-by-one errors, 76, 301
 - onLoad() event handler codes for, 544–545
 - uninitialized variables, 299–301
 - escape() **function, URL encoding and, 528**
 - event handlers. See also event listeners; events**
 - animated masking effect example, 193
 - game keystroke-handling example, 64–66
 - LoadVars class, 529
 - mouse-driven masking effect example, 194
 - onRelease event for simple button, 181, 183–184
 - pre-loader, 193
 - XML class, 541–542
 - event listeners. See also event handlers; events**
 - adding to image viewer, 257–262
 - attaching multiple listeners to multiple components, 262–270
 - creating listener functions, 255–257
 - creating listener objects, overview, 247–249
 - creating listener objects, radio button and check boxes example, 250–254
 - creating listener objects, simple button example, 249–250
 - creating listener objects, using an existing object, 254–255
 - deciding which technique to use, 257
 - defined, 66, 247
 - documentation for accessing event data, 249
 - game keystroke-handling example, 64–66
 - handling events from multiple sources, 264–265
 - handling multiple events from same component, 248
 - for Loader and ProgressBar components, 215–216
 - in monolithic movie pre-loading strategy, 218–219, 220
 - properties of event objects and, 248–249
 - registering to a component, 248
 - structuring code for multiple listeners, 263–264
 - using the stage as, 254–255
 - events. See also event handlers; event listeners**
 - Camera class, 516
 - documentation for accessing event data, 249
 - event sounds, 483
 - FileReference class (table), 640–641
 - handling multiple events from same component, 248
 - inheritance and, 106
 - JavaScript (table), 607
 - Loader component (table), 213
 - LocalConnection class (table), 578
 - manually triggering, 270–271
 - Microphone class (table), 491
 - MovieClip class (table), 177
 - MovieClipLoader class (table), 206
 - netStream class, 505–506
 - ProgressBar component (table), 214
 - SharedObject class, 583
 - Sound class (table), 476
 - Tween class (table), 411
 - XMLSocket class (table), 554
 - exercise answers, 699–758**
 - exporting. See also publishing projects**
 - Flash video to other formats, 503–504
 - Flash video wrapped in SWF file, 502–503
 - expressions**
 - building, 55–59
 - comparing dates, 59
 - comparing values, 55–56
 - in conditionals, 53, 54
 - defined, 53
 - modifying program behavior, 56–57
 - order of operator evaluation in, 57–58
 - simple examples, 54–55
 - values possible for, 54
 - extends **keyword, 682**
 - External API. See also ExternalInterface class**
 - calling ActionScript functions from JavaScript, 615–616
 - calling ActionScript methods from JavaScript, 616–617
 - calling JavaScript functions from ActionScript, 617–620
 - overview, 612
 - external files. See also importing; loading; pre-loading movies**
 - #include directive for, 151–152, 153
 - keeping code in, 151–153
 - refreshing data, usability concerns and, 523–524
 - ExternalInterface class**
 - available property, 615
 - creating pop-up windows, 630
 - defining window parameters, 632–633
 - designating SWF files as trusted content, 595, 596–597
 - methods, 612–614
 - pop-up window launcher example, 633–634
 - security alert window for, 595–596
- ## F
- fence post errors, 301**
 - fforward() **method, 405, 407**
 - Fibonacci sequence, 68**
 - File menu**
 - Import submenu, 146
 - keyboard shortcuts, 760, 763
 - Publish command, 13

file size

- components and, 237
- compressing SWF files, 15
- JPEG Quality setting and, 15

FileReference class

- events (table), 640–641
- methods, 637–640
- properties (table), 640

`fillColor` **property, 366**

`fillRect()` **method, 358**

fills

- `beginBitmapFill()` method for, 322–323, 334
- `beginFill()` method for, 160, 162, 322, 326–327
- `beginGradientFill()` method for, 323–324, 334
- `endFill()` method for, 160, 164, 324
- `fillRect()` method for, 358
- `floodFill()` method or, 358–359

filters

- applying multiple filters, 346–351
- applying to Bitmap object, 356
- applying to movie clip, 335
- BevelFilter, 338
- BlurFilter, 337
- cloning, 346
- ColorMatrixFilter, 343–345
- constructor parameters for, 335
- ConvolutionFilter, 340–343
- declaring properties for, 335
- DisplacementMapFilter, 345–346
- DropShadowFilter, 336–337
- ease of using, 335
- GlowFilter, 337–338
- GradientBevelFilter, 339–340
- GradientGlowFilter, 339
- overview, 335
- performance and, 197
- pinhole camera look using, 349–351

`filters` **property, 176, 335**

`findAHrefLinks()` **function, 532**

`findPixel()` **function, 520, 521**

`finish` **property, 410**

`firstChild` **property, 541**

Flash authoring tool. See Flash IDE (Integrated Development Environment)

Flash Communication Server

- attaching audio to netStream object, 494
- Camera object and, 517
- streaming video deployment, 498

Flash Detection Kit, 16–17

Flash 8 Video Encoder

- converting video to FLV, 501–502
- creating Flash video, 500

Flash IDE (Integrated Development Environment)

- Actions panel, 8–9
- ActionScript preferences, 9–12
- Auto Format preferences, 12–13, 118
- drawing API, 320
- Flash Detection Kit, 16–17
- Flash movies versus, 1
- keyframes and animation, 3–4
- Library, 4–5
- panel system, 1
- Properties panel, 2, 3
- publishing projects, 13–16
- start page, 1, 2
- symbols, 4–8
- timeline, 3
- Tools panel, 2, 3

Flash JavaScript Integration Kit

- calling ActionScript functions from JavaScript, 609–612
- calling JavaScript functions from ActionScript, 608–609
- obtaining, 607
- overview, 607
- setting up, 608

Flash movies. See Flash video; SWF files

Flash Only HTML template, 16

Flash Player

- defined, 1
- detection options, 16–17
- System object limitations and versions of, 655
- targeting versions of, 14

Flash video

- Camera object for, 515–522
- compositing with movie clips, 511–513
- controlling video position, 508–510
- controlling video quality, 513–514
- converting other formats to, 501–502
- converting to SWF, 500
- creating, 500
- CuePoints, 500, 510
- data rate, 497–498
- deblocking, 513–514
- deringing, 514
- exporting to other formats, 503–504
- Flash 8 Video Encoder for, 500, 501–502
- frame rate, 499–500
- interlacing, 499
- keyframes for, 498–499

- loading external files, 504–508
- On2 codec, 502
- playing in a movie clip, 507–508
- post-processing, 513
- progressive, 497, 498
- rotoscoping, 511
- scaling, 514
- streaming, 498
- SWF video versus, 497
- transparency in, 510–511, 514
- variable bit rate compression, 499
- wrapping in SWF file, 502–503
- Flash wrappers**
 - defined, 653
 - Flash browser plug-in as, 653
 - for Flash video over HTTP, 498, 502–503
 - HTML wrapper settings, 15, 16
 - launching programs using `fscommand()`, 656–658
 - Projector wrapper, 653
 - SDK for, 653, 658
 - third-party, 653, 658–659
- `FlashJSKit.html` **file**, 610–611
- `FlashtoJS fla` **file**, 600–602
- FlashType (Saffron)**, 433
- FlashVars**
 - adding to Object and Embed tag, 590–591
 - creating with JavaScript, 591–592
 - defined, 588
 - Object and Embed tag parameters, 589–590
 - passing via a servlet page, 593
- `flashVarsTest fla` **file**, 591
- `flashVarsTest.html` **file**, 591–592
- `floodFill()` **method**, 358–359
- `flush()` **method**
 - managing disk space for shared objects, 587–588
 - overview, 582, 583
 - user settings and, 587
- FLV. See Flash video**
- FLV Playback component**, 231–232
- focused code**, 106
- `focusEnabled` **property**, 176
- `_focusrect` **property**, 174
- folders or directories**
 - for bitmap images in project directory, 145
 - deleting from classpath, 11
 - library, recommended setup, 143–144
- fonts**
 - embedding, 434
 - global component styles for, 282
 - preferences for code, 10
 - recommendations for code, 10
 - sharing at runtime, 450–452
 - system versus embedded, 434
- `fontSource fla` **file**, 450–452
- `fontSource.swf` **file**, 451, 452
- for **loops**
 - animation speed and, 377
 - assigning IDs to dynamically created movie clips, 126–129
 - data tracking example, 73–76
 - Fibonacci sequence example, 68
 - graph building example, 330, 333
 - init portion, 67
 - iterator variables, 67, 108
 - for managing arrays, 67–68, 76
 - for movie clip creation and initialization, 68–69
 - naming variables for indices, 67
 - overview, 66–69
 - syntax, 66–67
 - while loop duplicating, 71
 - while loops versus, 71, 72
- `for..in` **loops**, 69–70, 347
- `formatEmailAddress()` **function**, 84–85
- formatting code**
 - Auto Format preferences, 12–13, 118
 - automatic indentation option, 9
 - indentation tab size option, 10
 - overview, 118–119
 - white space for, 25–26, 118–119
- ForwardButton component**, 232
- fps (frames per second). See frame rate or fps (frames per second)**
- `fps` **property of Camera class**, 516
- `FPS` **property of Tween class**, 410
- frame placeholders, inserting**, 3–4
- frame rate or fps (frames per second)**
 - choosing for animations, 379–380
 - common refresh rates, 379
 - deciding between `onEnterFrame()` and `setInterval()`, 379
 - default rate, 376
 - defined, 370
 - effect on animation, 376–377
 - experimenting with, 378–379
 - Flash video and, 499–500
 - `onEnterFrame()` function and, 376
 - `setInterval()` function and, 377
 - upper limit for, 380
- frames. See also frame rate or fps (frames per second); keyframes**
 - defined, 370
 - frame-based versus time-based animation, 376–380

frames (continued)

- going to and playing, 167
- inserting, 3–4
- interlaced video frames, 499
- single frame application, 154–156
- white versus gray on timeline, 4

`_framesloaded` **property**, 174

`fscommand` **event (JavaScript)**, 607

`fscommand()` **method**

- calling JavaScript functions using, 602–604
- calling projector functions using, 656–658
- command and `parm` values, 656–657
- Flash JavaScript Integration Kit versus, 612
- `getURL()` method versus, 603, 604
- launching programs using, 657–658
- overview, 656

`fscommandtoJS fla` **file**, 604, 605, 606

full screen sites, performance and, 197

fully qualified addresses for URLs, 186, 187

`Function` **data type**, 87–88

`function` **keyword**, 84

functions. See also custom functions; specific functions

- access functions for variables, 120–121
- anonymous, 87–88, 94–95
- applying filters using, 348–349
- arguments (input parameters), 83, 84
- assigning to `onRelease` event, 176–177
- calling, 84–85
- calling from a hyperlink, 452–453
- declared within another function, 94–95
- declaring functions, 84–85
- declaring variables in, avoiding, 119
- defined, 83
- delayed action and, 88–89
- general form, 84
- JavaScript, calling using `fscommand()`, 602–604
- JavaScript, calling within Flash, 598–599, 600–602
- modifying arguments passed in, 97–98
- naming, 84, 110
- overview, 83–86, 98–99
- passing as arguments, 88–92
- returns, 83
- reusability and, 87
- side effects of, 96–98
- small, debugging aided by, 304
- straight code versus, 86–87
- as variables, 88

G

Gaussian blur matrix

- ConvolutionFilter example using, 342–343
- 9x9 kernel, 341

`gaussianBlur fla` **file**, 342–343

`generateFilterRect()` **method**, 359

`get()` **method**

- Camera class, 515, 517
- Microphone class, 490

GET method (HTTP)

- LoadVars `send()` method and, 526
- LoadVars `sendAndLoad()` method and, 527
- overview, 558
- POST method versus, 558–559
- using with XML, 557–559

`getBytesLoaded()` **method**

- load listener function and, 472
- LoadVars class, 525, 526
- movie clip, 160, 164–165
- pre-loader examples, 165, 199–200
- pre-loading XML, 546, 547, 548
- Sound class, 474
- XML class, 538, 539

`getBytesLoaded` **property (XML object)**, 545

`getBytesTotal()` **method**

- load listener function and, 472
- LoadVars class, 525, 526
- movie clip, 160, 165–166
- pre-loader examples, 165–166, 199–200
- pre-loading XML, 546, 547, 548
- Sound class, 474
- XML class, 538, 539

`getBytesTotal` **property (XML object)**, 545, 548

`getColorBoundsRect()` **method**, 359

`getCurrentTime()` **function**, 85

`getFieldValue()` **function (JavaScript)**, 620

`getFormFields()` **function side effect**, 98

`getLocal()` **method**

- overview, 582–583
- shared objects as cookies example, 585
- sharing shared objects, 588

`getNextHighestDepth()` **method**, 179, 385

`getPan()` **method**, 475, 483

`getPercentLoaded()` **function**, 546, 547

`getPixel()` **method**, 359–360, 366–368

`getPixel32()` **method**, 360

`getProgress()` **method**, 206, 207

`getRGB()` **function, 10, 521**

`getSize()` **method, 582**

getters

defined, 676

photo thumbnail class example, 680

retrieving property values, 676

uses for, 676–677

`getTextFormat()` **method, 440**

`getTime()` **function, 59**

`getTransform()` **method, 475**

`getURL()` **function**

basic form, 166

creating pop-up windows, 629–630

defining window parameters, 632

`getURL()` method versus, 166–167

overview, 166

pop-up window launcher example, 633–634

`getURL()` **method**

basic form, 166

calling JavaScript using, 598–599, 600–602

Flash JavaScript Integration Kit versus, 612

`fscommand()` method versus, 603, 604

`getURL()` function versus, 166–167

limitations for JavaScript, 602

overview, 160, 166, 598

`getVariable` **method (JavaScript), 606**

`getVolume()` **method, 475, 483**

`_global` **keyword**

declaring global variables using, 92

minimizing use of, 39, 95

global variables

access functions for, 120–121

declaring in one place, 119–120

defined, 92

memory use by, 95, 119

minimizing use of, 39, 95

`_global.style` **variable, 282**

`_global.styles` **variable, 283**

GlowFilter

described, 337

pinhole camera look using, 349–351

properties, 337–338

`gotoAndPlay()` **function, 167**

`gotoAndPlay()` **method, 160, 167**

`gotoAndStop()` **method**

described, 149

in `gotoScreen()` function, 152

labels with, 149, 150

managing screen state with frames, 150

`gotoFrame` **method (JavaScript), 606**

`gotoScreen()` **function**

in split-up movie pre-loading strategy, 222–223, 224

switching screens using frames, 152–153

switching screens using movie clips, 155, 156

GradientBevelFilter, 339–340

GradientGlowFilter, 339

Graphic symbols, 5

graphs, building, 329–333

greater than operator (>), 55

greater than or equal to operator (>=), 55–56

H

`handleButtonClick()` **function, 127, 128–129**

`handleData()` **method, 691, 694**

`handleLoad()` **function, 306, 312, 313**

`handleStart()` **function, 90, 91–92**

`handleStop()` **function, 90, 92**

hard-coded movie clip references

avoiding in functions, 130

refactoring problematic code, 133–140

`hasChildNodes()` **method, 538, 539**

`HeaderDateText` **class, 287**

`height` **property**

bitmapData object, 365

Camera class, 516

Video class, 506

`_height` **property**

MovieClip class, 174

Video class, 507

helicopter animation project, 396–402

`HelloWorldClass.as` **file, 662–663**

Help menu keyboard shortcuts, 769

Help panel, 284

Hit state of buttons, 5

`hitArea` **property, 176**

`hitTest()` **method**

Bitmap object, 360

movie clip, 160, 167–168

HTML

calling functions from hyperlinks, 452–453

displaying, 446–448

enabling in a text field, 447

FlashVars, 588–593

image and SWF support in HTML text fields, 448–450

options for SWF file wrappers, 15

tags supported in Flash, 446

in XML elements, CDATA wrapper for, 537

`htmlInFlash.fla` **file, 447–448**

HTTP

- data rate for video and, 498
- POST and GET methods, 526, 527, 557–559
- progressive video with, 497, 498
- wrapping video in SWF file for, 498, 502–503

HTTPS, 567–568

hyperlinks, calling functions from, 452–453

I

IDE. See Flash IDE (Integrated Development Environment)

IDs

- assigning to dynamically created movie clips, 126–129
- for associative array rows, 75
- linkage, for movie clips, 153, 182

if...then...else statements

- logic errors, 298–299
- nesting, 60
- overview, 59–60
- shorthand, avoiding, 119
- switch...case statement compared to, 60–61
- trace() function for debugging, 311

ignoreWhite property, 541, 542, 547

image viewer project. See tryItOut_scriptingComponents project

imageClipLoaderTest.fla file, 473–474

imageLoadTest.fla file, 472

images. See bitmap images; drawing; vector graphics

implicit instantiation, 104, 105

import statement, 238

importing

- bitmap images to library, 146–147
- classes, 238–239
- LoadVars class for, 524
- packages, 238, 240–241, 411
- sound object, 477
- SWF file Protect From Import attribute and, 15
- video files, 503

#include directive, 151–152, 153

increment operator, 21

incrementNumber() function

- with side effect, 96–97
- without side effect, 97

indentation

- automatic, 9
- tab size option, 10

index property, 516

indexOf() method, 533

indices

- of arrays, 40

- of associative arrays, 46
- zero as starting integer for arrays, 40, 76

infinite loops

- debugging, 78–79, 302
- defined, 77
- while loop examples, 77–78

inheritance

- defined, 106
- overview, 106
- of setStyle() method for components, 282
- subclassing and, 681–682

init() function

- attaching movie clips with, 155
- controlling embedded component with JavaScript example, 622, 625
- debugging examples, 305–306, 308–309, 312, 313
- declaring variables in a container object, 120
- declaring variables in one place, 119–120
- image viewer example, setting properties using code, 243–244
- image viewer example, setting styles, 288–290
- image viewer example, with event listener, 258–259, 260–261
- image viewer example, with multiple listeners, 267, 269
- image viewer example, with XMLConnector, 280, 281
- loading XML files and extracting values, 550
- screen saver example, 426–427, 430
- in split-up movie pre-loading strategy, 223, 224–225

initTraces() function, 427, 430

inline comments, 26, 117

input text fields, creating, 435–436. See also text fields

Insert menu

- keyboard shortcuts, 765
- New Symbol command, 4, 6
- Timeline submenu, 3, 4, 151

inserting

- frame placeholders, 3–4
- frames, 3–4
- keyframes, 4, 151
- symbols, 4

instances, 104. See also objects

instantiation

- of classes, constructors and, 662, 665–669
- defined, 104
- implicit, 104, 105
- new operator for, 104–105

Integrated Development Environment. See Flash IDE (Integrated Development Environment)

interframes, 498–499

interlaced video, 499

Internet resources

Flash JavaScript Integration Kit, 607
 free socket server, 554, 555
 Northcode Flash wrappers, 653

`invokePlay()` **function**, **134**, **135**, **137**, **138**

`invokeStop()` **function**, **134**, **135–136**, **137**, **138**

`IsPlaying` **method (JavaScript)**, **606**

`isValid()` **function**, **87**

iterator variables, **67**, **69**, **108**

J**JavaScript. See also External API; Flash JavaScript Integration Kit**

calling ActionScript functions using External API, 615–616

calling ActionScript functions using integration kit, 609–612

calling ActionScript methods using External API, 616–617

calling Flash from, 604–606

calling functions using External API, 617–620

calling functions using `fscommand()`, 602–604

calling functions using integration kit, 608–609

calling functions with `call()`, 614

calling functions within Flash, 597–602

controlling an embedded component, 621–629

creating FlashVars with, 591–592

events (table), 607

External API for, 612–629

Flash Detection Kit and, 17

Flash JavaScript Integration Kit for, 607–612

`getUrl()` method for calling, 598–599, 600–602

methods (table), 606–607

nesting variables within a String, 598

opening browser windows, 618–619, 629–634

registering ActionScript functions as callable from, 612–613, 615, 616–617

security settings, 595–597

setting and getting data using ActionScript, 619–620

`window.open()` method, 629

wrapper function for `window.open()` method, 630–631

javascript **keyword**, **599**

`join()` **method**

loading and parsing raw text, 532, 533

overview, 42, 43

`split()` method versus, 43

JPEG Quality setting, **15**

`JStoFlash.html` **file**, **605**, **606**

justifying text, **440**

K

Kerning property for text, **441**

key detect function, **131–132**

keyboard shortcuts

Actions panel, 762

Control menu, 762, 767–768

disabling for test environment, 66

drawing IDE, 763–769

Edit menu, 761, 763–764

File menu, 760, 763

Help menu, 769

Insert menu, 765

integrated script editor, 760–762

Modify menu, 765–767

Text menu, 767

Tools menu, 761

Tools panel, 759–760

View menu, 761, 764–765

Window menu, 768–769

keyframes

adding script to, 8, 9

circles on timeline for, 4

creating animations using, 370–371

defined, 4, 370

for Flash video, 498–499

inserting, 4, 151

interframes and, 498–499

managing screen state with, 149–151

for navigating between screens, 149–151

temporal compression and, 498

keywords. See also specific keywords

not allowed for names, 29, 108–109

reserved words (tables), 109

L**Label component**

for application title, 241

data binding with DateChooser component, 275–277

described, 234

placing on the stage manually, 237

placing on the stage with script, 241

labels. See also Label component; names

assigning to text fields, 116, 181

with `gotoAndStop()` method, 149, 150

lastChild property, **541**

latency, **524**, **545**

launching programs using fscommand() method, **657–658**

layers, **8**, **149**

lcmovie1.fla file

- allowDomain() method example, 581–582
- local connection between two SWF files, 579–580

lcmovie2.fla file

- allowDomain() method example, 581–582
- local connection between two SWF files, 580–581

left-to-right operator associativity, 22

length property of arrays, 41

less than operator (<), 55

less than or equal to operator (<=), 55–56

letterSpacing property for text, 440–441

levels, 8, 184–185

_level10 reference, avoiding, 178, 220

_level10 timeline, 4

_level0 keyword, avoiding, 39

Library panel or library

- adding components to, 237
- adding symbols from, 5, 6, 162
- attaching movie clips from, 181–184
- dragging movie clips from, 178
- folder setup recommended for, 143–144
- importing bitmap images, 146–147
- loading sounds, 477–478
- New Symbol command, 4
- opening, 4, 89
- organizing bitmap images in, 145, 146
- placing components in, 215
- symbol names in, 5

lines (vector)

- drawing Bezier curves, 325, 327–329
- drawing straight lines, 160, 169, 325
- lineStyle() method for setting properties, 160, 168–169, 320–322

lineStyle() method (drawing API)

- Bezier curve example, 327, 328, 329
- graph building example, 330, 331, 332
- overview, 320
- parameters, 321–322

lineStyle() method (movie clip), 160, 168–169

lineTo() method (drawing API)

- in createSquare() function, 327
- drawing a square, 326
- graph building example, 330, 331, 332–333
- overview, 325

lineTo() method (movie clip), 160, 169

List component

- described, 234
- placing on the stage with script, 238, 241
- properties for event information, 248–249
- XML file as data source for, 277

listener function, 255–257

listener objects. See also event listeners

- adding to image viewer, 257–262
- attaching multiple listeners to multiple components, 262–270
- overview, 247–249
- radio button example, 250–251
- simple button example, 249–250
- using an existing object, 254–255

listeners. See event listeners; listener objects

load() method

- Loader component, 212
- LoadVars class, 525, 526, 531
- ThumbnailButton class, 670, 679
- XML class, 538, 540

loadBitmap() method, 360

loadCamera.fla file, 517–518

loadClip() method, 206, 207–208

loaded property, 541

Loader component

- customizing appearance of, 216–217
- described, 234
- events (table), 213
- implementing, 215–216
- load() method, 212
- properties (table), 213
- styles (table), 217

loader shim movie, 217

loaderShim files, 218, 220, 223

loadFLVVideo.fla file

- playing FLV video in a movie clip, 507–508
- simple controls, 508–510

loadImage() function, 259, 261–262

loading. See also pre-loading movies

- absolute addresses for, 187
- ampersand-delimited files, 529–531
- bitmap images dynamically, embedding versus, 144–145
- bitmap images dynamically, formats supported, 145
- bitmap images into movie clips, 145, 186, 471, 472
- checking if image is loaded, 472
- CSS data via LoadVars object, 455
- debugging loader code, 305–306, 310, 311–312
- external media, statement processing and, 20
- external media with loadMovie(), 169–170
- Flash video files, 504–508
- fully qualified addresses for, 186, 187
- library sounds, 477–478
- load listener function for, 472
- media into existing movie clips, 185–186
- movies into levels, 184–185
- policy file from subfolder, 568–569

pre-loading images, 472, 473–474
pre-loading XML files, 545–548
raw text and parsing, 531–534
relative addresses for, 187
removing loaded content, 173–174
thumbnail images, 189
unloading movie clips, 208–209
video files, 503
XML file and extracting values, 549–551
XML file (external), 542–545
XML file using XMLConnector, 559–561

`loadListener` **event listener**, 218–219, 220

`loadMedia()` **function**, 267–268, 269

`loadMediaError()` **function**, 268, 269

`loadMovie()` **method**
basic form, 169
loading images into movie clips, 473
loading media into movie clips, 186
loading movies into levels, 185
loading thumbnail images, 189
overview, 161, 169–170

`loadMovieNum()` **function**, 185

`loadNextData()` **method**, 691, 693

`loadPolicyFile()` **method**, 568–569

`loadProgress()` **function**, 486

`loadSound()` **method**
loading external MP3 files, 478
overview, 474, 475
for sound objects, 480, 482

`loadSoundFile()` **function**, 486, 489

LoadVars class
ampersand-delimited data and, 525, 528
creating an object, 527
event handlers, 529
methods, 525–527
overview, 524
properties, 527

LoadVars object
creating, 527
creating ampersand-delimited data using, 528
loading CSS data via, 455

local connections
LocalConnection class for, 578
between two SWF files, 579–581
uses for, 577

local variables, defined, 92

LocalConnection class, 578. *See also* local connections

LocalConnection object
allowDomain() method for security, 581–582
creating, 579

`_lockroot` **property**, 174

loops. See also specific kinds
data tracking example, 73–76
debugging, 76–79
defined, 66
do..while loops, 72–73
for loops, 66–69
for..in loops, 69–70
infinite, 77–79, 302
off-by-one errors for arrays, 76
while loops, 70–72

lowercase. See case

M

main or root timeline, 4

masking
animated effect using, 191–193
basic form for applying masks, 171, 191
defined, 190
mouse-driven effect, 194–195
uses for, 191

mathematical operators, 21

Matrix object, 334

`maxScroll` **property**, 469

measuring microphone activity, 492–494

Media component, 508

MediaController component, 233

MediaDisplay component, 233

`mediaList.xml` **file**, 279

MediaPlayback component, 233

memory, variables and use of, 95, 119

Menu component
described, 234
properties for event information, 249
simple audio interface example, 484, 487, 488, 489

`menu` **property**, 176

MenuBar component, 234

`merge()` **method**, 361

merging bitmap images, 361

methods. See also constructors; specific methods
adding using subclassing, 681–682
of arrays (table), 42
Bitmap object, 356–365
built-in easing methods, 404, 405
Camera class (table), 515
defining for custom classes, 669–675
displaying for ActionScript classes, 37
dot syntax, 37–38, 102
for drawing, 320–325
ExternalInterface class, 612–614
FileReference class, 637–640

methods (continued)

- getters and setters, 676–677
- inheritance and, 106
- JavaScript, calling within Flash, 598–599
- JavaScript (table), 606–607
- LoadVars class (table), 525
- LoadVars event handlers, 529
- LocalConnection class, 578
- Matrix object, 334
- Microphone class (table), 490
- movie clip (table), 160–161
- MovieClipLoader class (table), 206
- netStream class (table), 504–505
- photo thumbnail class example, 670–675
- private versus public, 663–664, 669
- SharedObject class, 582–583
- Sound class (table), 474–475
- static, 682–686
- Tween class (table), 405–406
- Video class, 506
- XML class (table), 538
- XMLSocket class (table), 553

micLevel **function, 493, 494**

Microphone class

- creating an object, 491–492
- events, 491
- methods, 489–490
- properties, 490–491

Microphone object

- activityLevel property, 492–494
- creating, 491–492
- security and, 522

microphone fla file, 491–492

Modify menu

- Convert to Symbol command, 4
- keyboard shortcuts, 765–767

monitorProgress() **function**

- checking for zero value, 200
- halting playback until movie is loaded, 201
- passing parameters to, 202–203
- progress bar function call from, 201–202
- removing custom event handler, 204, 205
- simple pre-loader example, 200

monolithic movie pre-loading strategy, 217–220

motionLevel **property, 516**

motionTimeOut **property, 516**

mouse-driven masking effect, 194–195

.mov files

- converting to Flash video, 501–502
- exporting Flash video to, 503
- importing, 503

- moveShipDown()** **function, 398, 401**
- moveShipLeft()** **function, 398, 401**
- moveShipRight()** **function, 398, 401**
- moveShipUp()** **function, 398, 401**
- moveSpheres()** **function, 427–428, 431**
- moveTo()** **method (drawing API)**
 - Bezier curve example, 327
 - graph building example, 330, 331, 332
 - overview, 324
- moveTo()** **method (movie clip), 161, 170**

movie clips. See also specific methods

- accessing variables in another movie clip, 122–129
- assigning IDs to dynamically created movie clips, 126–129
- associative arrays for, 48–49
- attaching Bitmap objects, 161–162
- attaching from the library, 181–184
- attaching to stage, 153–154
- BlendMode property, 352–353
- changing stacking order, 173
- compositing Flash video with, 510–513
 - as containers, 148, 186
- converting to bitmap images, 355, 366
- creating a simple button, 179–181, 182–184
- creating by dragging from the library, 178
- depth numbers for, 179
- depths encapsulated in, 8
- dragging by user, methods for, 171–173
- fading using Tween class, 404
- filters property, 335
- for loop for creating and initializing, 68–69
- hard-coded references to, avoiding, 130
- _level10 reference, avoiding, 178, 220
- linkage IDs, 153
- linking Bitmap objects to, 360
- loading external movies into levels, 184–185
- loading media into, 185–186
- masking, 190–195
- methods (table), 160–161
- moving using **onEnterFrame()**, 372–374
- moving using **setInterval()**, 374–376
- moving using Tween class, 404
- overlapping, detecting, 167–168
- overview, 5, 7
- parent movie clip for, 178
- performance improvements, 196–197
- placing in buttons, 147
- playing, 170
- playing FLV video in, 507–508
- playing SWF video in, 503–504
- positioning, 180

referencing absolutely, avoiding, 178
referencing by variable, 178, 180
referencing via associative array syntax, 179, 181
referencing via return from creation method, 178
removing from stage, 170
scrolling text fields using, 465–467
timelines for, 7
using as a base, 148–149
`_visible` property, 153

MovieClip class. See also specific methods
events (table), 177
methods (table), 160–161
properties (tables), 174–176
`scrollRect()` method, 465–467

MovieClip data type, 34

MovieClipLoader class
`addListener()` method, 206–207
events, 205–206
`getProgress()` method, 206, 207
implementing, 209–212
`loadClip()` method, 206, 207–208
pre-loading images using, 473–474
`removeListener()` method, 208
`unloadClip()` method, 206, 208–209

.mpg or .mpeg files
converting to Flash video, 501–502
importing, 503

MP3 files
ID3 tags, 478
loading external files, 478–479
playing and controlling sound objects, 480–482
simple audio interface for, 484–489
streaming MP3 audio, 480
V2 tag properties, 478–479

`multiFilters.fla` file, 347–349
`musicPlayer.fla` file, 484–489

MuteButton component, 232

`mx.transitions.easing` classes, 405
`myChangeHandler()` function, 255–256
`MyClass.as` file, 661–662
`myData.xml` file, 542–543, 546
`myFlashtoJavaScript.html` file
for calling Flash from JavaScript, 605
for calling JavaScript from Flash, 600, 601, 603–604
`myVideo.fla` file, 502–503

N

`_name` property
MovieClip class, 174
Video class, 507

`name` property of Camera class, 516

names. See also labels
of bitmap images, 145
of Boolean variables, 108
camelBack style for variables, 107–108
of classes, 106
of functions, 84
of iterator variables, 67, 108
main timeline default, 4
of objects, 106
reserved words not allowed for, 108–110
of symbols, 4–5
text field instance names versus variable names, 434
understandable code example, 113–116
variable naming conventions, 29–30, 106
of variables, avoiding duplicates, 96
of variables, best practices, 107–110

`names` property of Camera class, 516

`NaN` return value, 200

nesting. See also loading
`if...then...else` statements, 60
JavaScript variables in strings, 598
movie clips, accessing variables and, 122–129
performance and, 60

netConnection object, 510

netStream class, 504–506

netStream object
attaching audio to, 494
Camera object and, 517
`seek()` method, 510

new keyword
creating an Object instance using, 46
instantiation using, 104–105
for XML class access, 537

New Symbol dialog, 6

`nextFrame()` method, 406, 407–408
`nextSibling` property, 541
`nextTrack()` function, 484–485

9x9 blur matrix kernel, 341

`nodeName` property, 541
`nodeType` property, 541
`nodeValue` property, 541

noise
`noise()` method for, 361–362
`perlinNoise()` method for, 362–363
`noise()` method, 361–362

Northcode third-party Flash wrappers, 653, 659

null bytes, 553

null keyword, 29, 42

Number data type, 34

numerical precision errors, 302–303

NumericStepper component

- described, 234
- event listener for, 259–260
- placing on the stage manually, 237
- placing on the stage with script, 241
- scripting initial parameters, 244
- simple audio interface example, 485, 488

O

Object class

- creating an instance using `new` keyword, 46
- creating an object and assigning properties, 47

Object data type, 34

Object tag

- adding FlashVars to, 590–591
- `allowScriptAccess` parameter, 603, 619, 628
- parameters, 589–590

object-oriented programming. See OOP

objects

- classes versus, 103–106
- defined, 101
- grouping variables in a container object, 120
- naming conventions, 106
- package of code and data in, 102
- properties and methods derived from parent class, 104

off-by-one errors, 76, 301

onCancel **event, 640**

onClap **object, 493**

onClose **event, 554**

onComplete **event**

- described, 640
- downloading files, 643, 645
- uploading files, 648, 650

onConnect **event, 554**

onData **event, 554**

onData () **event handler**

- loading ampersand-delimited files, 530, 531
- loading and parsing raw text, 532, 533
- loading external XML files, 543, 544
- LoadVars class, 529
- XML class, 541–542

onDragOut **event, 177**

onDragOver **event, 177**

onEnterFrame **event, 177**

onEnterFrame () **function**

- adding random behavior, 383
- animating multiple movie clips, 382
- animating single snowflake, 380
- experimenting with, 378–379
- frame rate and, 376

- moving movie clips using, 372–374
- `setInterval ()` function versus, 378–379
- snowstorm animation, 384
- timing mechanism for pre-loader, 204–205

onHTTPError **event**

- described, 640
- uploading files, 648, 650

onID3 **event, 485, 488**

onIOError **event**

- described, 640
- downloading files, 643, 645

onKeyDown **event, 177**

onKeyUp **event, 177**

onKillFocus **event, 177**

onLoad **event, 177**

onLoad () **event handler**

- error codes, 544–545
- loading ampersand-delimited files, 530, 531
- loading CSS data, 455
- loading external XML files, 544–545
- loading XML files and extracting values, 549–550
- LoadVars class, 529
- pre-loading XML, 546, 547
- XML class, 542

onLoadComplete **event, 206**

onLoadError **event, 206**

onLoadError **event handler, 212**

onLoadInit **event, 206**

onLoadInit () **function, 473–474**

onLoadInit **handler, 193**

onLoadInit () **method, 670–672, 673**

onLoadProgress **event, 206**

onLoadStart **event, 206**

onLoadStart **handler, 193**

onMetaData **event**

- overview, 505–506
- video controls example, 508, 510

onMotionChanged **event, 411**

onMotionChanged () **function**

- bouncing elasticized box into position, 411–412, 413
- playing tweens in parallel, 415, 417

onMotionFinished **event, 411**

onMotionFinished () **function**

- bouncing elasticized box into position, 412, 413
- playing tweens in parallel, 415, 417
- playing tweens in sequence, 417, 418

onMotionResumed **event, 411**

onMotionStarted **event, 411**

onMotionStopped **event, 411**

onMouseDown **event, 177**

onMouseMove **event, 177**

onMouseUp **event, 177**

onOpen **event**
described, 640
downloading files, 643, 644
uploading files, 647, 649

onPress **event, 177, 348**

onProgress **event (FileReference class)**
described, 640
downloading files, 643, 644
uploading files, 647–648, 650

onProgress **event (JavaScript), 607**

onReadyStateChange **event (JavaScript), 607**

onRelease **event**
assigning function to, 176–177
for controlling sound objects, 481, 482
described, 177
handler for simple button, 181, 183–184

onReleaseOutside **event, 177**

onRollOut **event, 177**

onRollOver **event, 177**

onSecurityError **event**
described, 641
uploading files, 648, 650

onSelect **event**
described, 641
uploading files, 647, 649

onSetFocus **event, 177**

onSoundComplete **event, 481, 482, 485–486, 488**

onStatus **event**
LocalConnection class, 578
netStream class, 505
SharedObject class, 583

OO2 video codec, 502

onUnload **event, 177**

onXML **event, 554**

OOP (object-oriented programming). See also coding
class inheritance, 106
classes as basis of, 661
classes versus objects, 103–106
defined, 101, 661
goals of, 106–107
overview, 101–102
procedural programming versus, 102

opaqueBackground **property, 176**

opening
Actions panel, 8
browser windows from Flash movies, 618–619,
629–634
Component Inspector panel, 273
Components panel, 89

Library panel, 4, 89
Preferences dialog, 9

openWindow() **function, 618**

operators
associativity, 22
common operators (table), 23–24
in conditionals, 53, 54
defined, 20
forcing order of evaluation, 22
multiple per statement, 21
order of evaluation in expressions, 57–58
overview, 20–22
precedence, 22
using spaces between operands and, 119

Over state of buttons, 5, 7

overlapping objects, detecting
Bitmap objects, 360
movie clips, 167–168

oversized functions, avoiding, 131

P

packages
for Bitmap object, 355
defined, 238
importing, 238, 240–241, 411
referencing for components, 238

paletteMap() **method, 362**

pan **method (JavaScript), 606**

panels, 1. See also specific panels

_parent property
MovieClip class, 122, 125–126, 174
Video class, 507

parent-child node relationship (XML)
attributes versus nodes, 552
navigating a node tree, 549–551
overview, 549

parentNode **property, 541**

parseCSS() **method, 455**

parseStation() **function, 550–551**

parseXML() **method, 538, 540**

passing data between variables
duplicating data for composite data types, 36–37
overview, 29
by value versus by reference, 35–36

passing functions as arguments
example, 89–92
overview, 88–89

pause() **method, 504, 505**

PauseButton component, 232

.pdf files, fscommand() no longer opening, 656

percentLoaded method (JavaScript)

percentLoaded method (JavaScript), 606

performance

- bitmap caching for improving playback, 196
- blending modes and, 197
- filters and, 197
- full screen sites and, 197
- movie clip, improving, 196–197
- nesting and, 60
- transparency and, 196–197
- `_visible` property for improving, 197

period (.) as classpath, 11

`perlinNoise()` **method, 362–363**

persistent Flash applications

- refreshing data, usability concerns and, 523–524
- runtime delays and, 524

photo thumbnail class. See `tryItOut_thumbnailButton` project

photo viewer project. See `tryItOut_scriptingComponents` project

PHP

- passing FlashVars via servlet, 593
- `proxy.php` file, 570–571

`pinHoleCamera.fla` **file, 349–351**

pipe (|) as delimiter for data, 525

`pixelDissolve()` **method, 363–364**

pixels

- Bitmap object and control of, 319
- copying using `copyPixels()` method, 357
- copying using `draw()` method, 358
- creating on specified coordinates, 364–365
- dissolving from original to defined source, 363–364
- getting RGB color and alpha value for, 360
- getting RGB color for, 359–360
- replacing colors based on testing, 365
- scrolling, 364
- tracking in webcam in real time, 519–521

`play()` **function, 170**

`play()` **method**

- JavaScript, 606
- movie clip, 161, 170
- `netStream` class, 505, 508–509
- `play()` function versus, 170

PlayButton component, 232

playhead, 370

playing

- Flash video in a movie clip, 507–508
- movie clips, 170
- simple audio interface for, 484–489
- sound objects, 477–478, 480–482
- SWF video in movie clips, 503–504
- tweens in parallel, 414–417

- tweens in parallel and in sequence, 418–424

- tweens in sequence, 417–418

PlayPauseButton component, 232

`playSound.fla` **file, 480–482**

`playTweenSequence()` **function, 421, 423–424**

`plotLine()` **function, 428, 431**

polar coordinate system, 425–426

policy files. See domain policies

populating arrays

- by assigning values to elements, 41
- `push()` method for, 41, 43
- when creating, 40–41

pop-up windows

- creating using `ExternalInterface`, 630
- creating using `getURL()`, 629–630
- defining parameters for, 631–633
- JavaScript wrapper function for, 630–631
- launcher for, 633–634

`portfolio.as` **file, 218, 222**

`portfolio.fla` **file**

- attaching code to a project, 152–153
- managing screen state with frames, 150–151
- monolithic movie pre-loading strategy, 218–221
- single frame application, 154–156
- split-up movie pre-loading strategy, 222–225
- using movie clip as base, 148–149
- working with bitmap images, 146–148

`position` **property**

- Sound class, 480, 482
- Tween class, 410

POST method (HTTP)

- GET method versus, 558–559
- `LoadVars send()` method and, 526
- `LoadVars sendAndLoad()` method and, 527
- overview, 558
- using with XML, 557–559

post-processing for Flash video

- deblocking, 513–514
- defined, 513
- deringing, 514
- scaling, 514

`powerUpShield()` **function, 119–120**

precedence of operators, 22

Preferences dialog

- ActionScript preferences, 9–12
- Auto Format preferences, 12–13, 118
- opening, 9

pre-loading images, 472, 473–474

pre-loading movies

- animated masking effect example, 193
- binding loader to object holding event handlers, 206–207

- breaking the binding with event handlers, 208
- checking for zero value, 200
- creating a custom pre-loader, 199–203
- `getBytesLoaded()` method for, 164–165, 199–200
- `getBytesTotal()` method for, 165–166, 199–200
- getting progress manually, 207
- halting playback until movie is loaded, 201
- Loader component for, 212–213, 215–217
- loader shim movie for, 217
- monolithic movie approach, 217–221
- `MovieClipLoader` class for, 205–212
- overview, 199
- passing parameters for references, 202–203
- polling for number of bytes loaded, 199–200, 204–205
- progress bar function for, 201–202
- ProgressBar component for, 213–217
- simple example, 199–200
- split-up movie approach, 221–225
- timing mechanism for pre-loader, 200, 204–205
- triggering the loading, 207–208
- unloading movie clips, 208–209
- pre-loading XML files, 545–548**
- `prevFrame()` method, 406, 408
- `previousSibling` property, 541
- primitive data types, 34–35**
- `private` keyword, 664–665
- private properties and methods**
 - in constructors, 666
 - making code private, 664–665
 - public versus, 663–664, 669
- profiles for publishing, saving, 13**
- progress bar. See also ProgressBar component**
 - function for pre-loading movies, 201–202
 - for scripting components, 243–244
 - updating from event listener, 220
- ProgressBar component**
 - described, 234
 - events (table), 214
 - implementing, 215–216
 - placing on the stage manually, 236
 - properties (table), 214
 - `setProgress()` method, 213
 - simple audio interface example, 486, 488
 - video controls example, 509
- progressive video, 497, 498**
- Projector**
 - calling functions using `fscommand()`, 656–658
 - as Flash wrapper, 653
- projects**
 - attaching code to, 152–153
 - folder for bitmap images, 145
 - publishing, 13–16
- properties**
 - bitmapData object, 365–366
 - Camera class (table), 516
 - in constructors, private, 666
 - CSS1 and Flash equivalents, 456
 - data binding, 273, 275–277
 - defining for custom classes, 675–681
 - displaying for ActionScript classes, 37
 - dot syntax, 37–38, 102
 - of event objects, 248–249
 - ExternalInterface class, 615
 - FileReference class (table), 640
 - getters and setters for, 676–677
 - inheritance and, 106
 - Loader component (table), 213
 - LoadVars class, 527
 - Microphone class (table), 490–491
 - MovieClip class (tables), 174–176
 - netStream class (table), 505
 - photo thumbnail class example, 677–681
 - private versus public, 663–664
 - ProgressBar component (table), 214
 - SharedObject class, 583
 - Sound class (table), 476
 - text display properties, 436–439
 - Tween class (table), 410
 - Video class (table), 506–507
 - XML class (table), 541
- Properties panel**
 - overview, 2, 3
 - setting component startup parameters, 235
- prototype property, not recommended, 681**
- proxies, 570–571**
- pseudo-classes for CSS styles, 459**
- public keyword**
 - declaring methods with, 669
 - as default, 669
 - `private` keyword versus, 664–665
- public properties and methods**
 - getters and setters, 676–677
 - private versus, 663–664
 - problems with public properties, 675
- Publish Settings dialog**
 - Flash tab, 14–15
 - Formats tab, 13, 14
 - HTML tab, 15
 - Image tabs, 15–16, 17
 - overview, 13–14
 - saving profiles, 13

publishing projects

- components and file size, 237
 - Flash Detection Kit and, 16–17
 - formats for, 13
 - HTML wrapper settings, 15, 16
 - image attribute options, 15, 17
 - overview, 13
 - paths for, 13
 - saving profiles, 13
 - SWF file settings, 14–15
- `push()` **method**
- data queue example, 690, 693
 - described, 42
 - overview, 43
 - populating arrays using, 41, 43

Q

quality control for Flash video

- deblocking, 513–514
 - deringing, 514
 - as post-processing, 513
 - scaling, 514
- `_quality` **property, 174**

QuickTime, required for Flash 8 Video Encoder, 500

quotation marks (“)

- for Boolean values, avoiding, 33
- indicating strings, 33
- nesting JavaScript variables in strings and, 598

R

RadioButton component, 234

- `random()` **function, 382**
- `randomRange()` **function**
- overview, 382
 - for screen saver animation, 429, 432
 - for snow effect animation, 383, 384, 385–386

rasterizing vector graphics, 355

RDBMSResolver component, 231

readability. *See* understandability

Real-time Messaging Protocol (RTMP), 498

rectangle **property, 366**

rectangles

- `fillRect()` method for, 358
- flash.geom package for, 465
- `generateFilterRect()` method for, 359
- `getColorBoundsRect()` method for, 359
- `noise()` method for, 361–362

refactoring problematic code, 132, 133–140

relative addresses for URLs, 187

relative file paths, domain policies versus, 564

relative references, *this* keyword for, 178

relative tweens, 404

- `removeListener()` **method, 208**
- `removeMovieClip()` **method, 161, 170**
- `removeNode()` **method, 538, 540**

removing. *See* deleting or removing

- `replace()` **method**
- static method example, 683–684
 - subclassing example, 681–682

replacing

- array elements, 45
- colors based on testing pixels, 365
- Web pages using `getURL()`, 166–167

`repositionMedia()` **function, 185**

reserved words. *See also* keywords

- class and component names (table), 110
- defined, 108
- keywords (tables), 109
- not allowed for names, 108–110

`reset()` **function, 623, 627**

`resume()` **method, 406, 408–409**

`resumeTweenSequence()` **function, 421, 424**

return **statements**

- declaring return types, 130–131
- side effects from not using, 96

reusability

- as critical aspect of programming, 83
- functions for, 87
- as goal of OOP, 106
- inheritance and, 106
- prototype property and, 681
- reduced by global variables, 95

`rewind()` **method**

- JavaScript, 606
- Tween class, 406, 409

rich text formatting

- `align` property, 440
- deciding which technique to use, 439
- displaying HTML, 446–448
- example using, 441–443
- `getTextFormat()` method for, 440
- `kerning` property, 441
- `letterSpacing` property, 440–441
- new options for Flash 8, 440–441
- `setTextFormat()` method for, 440, 444, 445
- `TextFormat` class for, 439–440

right-to-left operator associativity, 22

`_root` **keyword, avoiding, 38–39**

root or `_root` **timeline, 4**

`ROOT_PATH` **constant**, 190

`_rotation` **property**, 507

rotoscoping, 511

round brackets [()]

for forcing order of operator evaluation, 22
after function names, 84

`rowClicked()` **function**, 623–624, 627

RTMP (Real-time Messaging Protocol), 498

runtime

delays, persistent Flash applications and, 524
depths and, 8
sharing fonts at, 450–452

S

Saffron (FlashType), 433

`SampleTheme.fla` **file**, 292

saving profiles for publishing, 13

`scale9Grid` **property**, 176

scaling Flash video, 514

scope of variables

access functions and, 120–121
accessing variables and, 95–96
best practices, 119–122
custom classes for managing data, 121–122
declaring variables in one place, 119–120
defined, 92
example, 93–94
function defined within another function and, 94–95
global, 39, 92, 95
grouping variables in a container object, 120
local, 92
managing, 95
scope chain, 92–93, 96
side effects and, 96–98

screen persistence, 523

screen saver example

coding, 426–429
`init()` function, 430
`initTraces()` function, 427, 430
`moveSpheres()` function, 427–428, 431
overview, 426
`plotLine()` function, 428, 431
`randomRange()` function, 429, 432
storing data for, 426, 429
`updateSeeds()` function, 427, 428, 430, 431–432
variables, 429–430

screen state

managing with keyframes, 149–151
managing with script, 153–156

scripting. **See coding; OOP (object-oriented programming)**

scripting environment setup

Actions panel overview, 8–9
ActionScript preferences, 9–12
Auto Format preferences, 12–13
Flash Player detection options, 16–17
publishing a project, 13–16

`scroll()` **method**, 364

`scroll` **property**, 467, 469

ScrollBar component

scrolling text fields using, 463–465
`setScrollTarget()` method, 463, 464–465
`scrollComponentTest.fla` **file**, 464–465

scrolling

custom scroll bar for, 461
`MovieClip scrollRect()` method for, 465–467
pixels, 364
`scroll` and `maxScroll` properties for, 467–469
ScrollBar component for, 463–465
text fields, 461–469
TextArea component for, 462–463

ScrollPane component

described, 234
placing on the stage manually, 236
placing on the stage with script, 241
`scrollDrag` property, 244

`scrollRect()` **method**, 465–467

`scrollRect` **property**, 176

`scrollRectTest.fla` **file**, 465–467

SDK (Software Development Kit), 653, 658

search order for styles, 283–284

security

Camera object and, 522
cross-domain, 562–573
designating SWF files as trusted content, 595, 596–597
domain policies, 563–570
HTTP GET method versus POST method and, 559
HTTPS, 567–568
loading XML and, 545
local versus remote files and, 562–563
localConnection object, `allowDomain()` method for, 581–582
Microphone object and, 522
proxies, 570–571
sandbox, 563
settings for JavaScript, 595–597
shims, 570, 571–573
for SWF files in HTML image tags, 448

`seek()` **method**, 505, 510

SeekBar component, 232

selectors (CSS), defining, 458

self-documenting code, 30, 116, 304

semicolon (;) terminating statements, 19

`send()` **method**

LoadVars class, 525, 526

LocalConnection class, 578

XML class, 538, 540

XMLSocket class, 553

`sendAndLoad()` **method**

LoadVars class, 525, 527

XML class, 538, 540, 556–557

`sendMessage()` **function**, 599

`sepiaTone.fla` **file**, 344–345

servlets, passing FlashVars via, 593

`setBufferTime()` **method**, 505, 510

`setClickHandler()` **function**, 623, 627

`setClipboard()` **method**, 655

`setColumnLabels()` **function**, 622, 626

`setColumnWidths()` **function**, 622–623, 626

`setDraggable()` **function**, 327–328, 329

`setFieldValue()` **function (JavaScript)**, 620

`setFormatProperties.fla` **file**, 441–443

`setFormatSpan.fla` **file**, 443–445

`setGain()` **method**, 490

`setHighlights()` **method**, 444, 445

`setInterval()` **function**

animation smoothness and, 377

calling `updateAfterEvent()` function from, 377, 378–379, 391

checking handle availability for levels, 185

choosing an update interval for animations, 379–380

delayed action and, 88–89

easing in example, 393, 395

easing out example, 389, 390

experimenting with, 378–379

frame rate and, 377

function as argument for, 88–89

helicopter animation example, 397, 399–400

measuring microphone activity and, 493, 494

moving movie clips using, 372–374

`onEnterFrame()` function versus, 378–379

pre-loading images and, 472

pre-loading XML, 548

timing mechanism for pre-loader, 200

`setKeyFrameInterval()` **method**, 515

`setLabel()` **method**, 669

`setLoopBack()` **method**, 515

`setMask()` **method**, 161, 171, 191. *See also* masking

`setMode()` **method**, 515, 516

`setMotionLevel()` **method**, 515

`setMusicLevel()` **function**, 617

`setNewTextFormat()` **method**, 445–446

`setPan()` **method**, 475, 476, 483

`setPixel()` **method**, 364, 366–368

`setPixel32()` **method**, 364–365

`setProgress()` **method**, 213

`setQuality()` **method**, 515

`setRate()` **method**, 490

`setScrollTarget()` **method**, 463, 464–465

`setSilenceLevel()` **method**, 490

`setStyle()` **method**

for Alert component, 287

available styles, 284–286

basic form, 282

for Button component, 282

for DataGrid component, 282

for DateChooser component, 287

for DateField component, 287

image viewer example, 287–290

inheritance for all components, 282

for Loader component, 216–217

setting styles for component type, 283

setting styles for text fields, 456–458

setting styles globally, 282, 289, 290

setting styles using custom classes, 283

skinning components versus, 290–291

style search order, 283–284

setters

defined, 676

photo thumbnail class example, 680

setting property values, 676

uses for, 676–677

`setTextExtent()` **method**, 442–443

`setTextFormat()` **method**, 440, 444, 445

`setTransform()` **method**, 475–476, 483–484

`setupInterface()` **function**

downloading files, 642–643, 644

uploading files, 647, 649

`setupMovie()` **function**, 134–135, 136, 137, 139

`setUseEchoSuppress()` **method**, 490

`setVariable` **method (JavaScript)**, 607

`setVolume()` **method**, 475, 483, 617

`setZoom()` **function**, 259, 262

`setZoomRect` **method (JavaScript)**, 606

shadows

BevelFilter and, 338

DropShadowFilter for, 336–337

shapes, filling. *See* fills

shared objects

as cookies, 582, 585–587

creating and saving data in, 583–584

data types acceptable for, 584

managing disk space, 587–588

- retrieving data objects, 584–585
- SharedObject class for, 582–583
- sharing, 588
- user settings and, 587
- sharedFontText.swf **file**, **452**
- SharedObject class, 582–583. See also shared objects**
- sharedObjectSample.fla **file**
 - creating shared object and saving data, 583–584
 - retrieving data objects, 584–585
 - using shared objects as cookies, 585–587
- shareFontTest.fla **file**, **451–452**
- sharing fonts at runtime, 450–452**
- sharpness **text display property, 437, 438–439**
- shim.fla **file**, **572–573**
- shims, 570, 571–573**
- showProgress() **function**
 - in custom pre-loader, 202, 203
 - in monolithic movie pre-loading strategy, 219, 221
 - in MovieClipLoader class implementation, 210
- side effects of functions**
 - defined, 96
 - example changing a variable, 96–97
 - example modifying and argument, 97–98
 - uses for, 98
- simpleFlashToJS.fla **file**, **598–599**
- single quotes ('), nesting JavaScript variables in strings and, 598**
- single-line comments, 26, 117**
- skinning components**
 - defined, 281
 - image viewer example, 291–294
 - setStyle() method versus, 290–291
 - switching themes, 291
 - themes for, 291
- slash (/) for comments, 26–27, 117**
- smoothing property, 506**
- snow effect project**
 - adding random behavior, 382–383
 - animating multiple movie clips, 381–382
 - animating single snowflake, 380–381
 - creating a snowstorm, 383–386
- Socket transactions, policy file for, 569–570**
- Software Development Kit (SDK), 653, 658**
- Sorenson Squeeze, 504**
- sort() **method**, **42, 43–44**
- sound. See also Sound class**
 - attaching audio to netStream object, 494
 - creating a sound object, 477
 - event sounds, 483
 - loading external MP3 files, 478–479
 - loading library sounds, 477–478
 - Microphone class for, 489–494
 - overview, 474
 - playing sound objects, 477–478, 480–482
 - simple audio interface, 484–489
 - streaming MP3 audio, 480
 - SWF file settings for, 15
 - system volume settings and, 477
- sound channels, setTransform() method for, 475–476, 483–484**
- Sound class. See also specific methods**
 - creating an object, 477
 - duration property, 480
 - events (table), 476
 - instantiating, 104–105
 - loading a sound file and starting playback, 105
 - methods, 474–476
 - position property, 480, 482
 - properties (table), 476
- soundIndex component, 486**
- SoundTest.fla **file**, **477–478**
- spaces as delimiter for data, 525**
- SpinEffect.as **file**, **687–690**
- splice() **method**, **42, 45**
- split() **method**
 - join() method versus, 43
 - parsing raw text, 532, 533
- split-up movie pre-loading strategy, 220–225**
- square brackets ([]) for setting or retrieving array elements, 41**
- square, drawing, 326–327**
- stacking order of movie clips, changing, 173**
- start() **method**
 - Sound class, 475, 480, 482, 483
 - Tween class, 406, 409
- start page, 1, 2**
- startDrag() **method**, **161, 171–172**
- startTweenSequence() **function**, **421, 422, 423**
- statements. See also specific statements**
 - compound, 20
 - in conditionals, 54
 - described, 19
 - simple, 19–20
- static methods**
 - calling, 682–683
 - creating classes having, 683
 - overview, 682–683
 - StringLib class example, 683–686
- status **property**, **541**
- stop() **function**, **172**

stop() method

stop() method

- movie clip, 161, 172
- Sound class, 475
- stop() function versus, 172
- Tween class, 406, 409

StopButton component, 232

stopDrag() method, 161, 172–173

stopPlay method (JavaScript), 607

stopTweenSequence() function, 421, 424

strict typing for triggering code hints, 10

String class, 103, 104

String data type, 33, 34. *See also* strings

strings. *See also* ampersand-delimited files; String data type; text fields

- applying TextFormat property to substrings, 443–445
- array output as, 43
- converting arrays to, 43
- converting to arrays, 43
- loading and parsing raw text, 531–534
- nesting JavaScript variables in, 598
- quotation marks for indicating, 33
- String class properties and methods and, 103

strong typing

- code hints and, 111
- debugging and, 31–33
- defined, 31
- example, 31–33, 113–116
- for function arguments, 129–130
- need for, 30–31, 110–111
- suffixes suggested for data types (table), 111–112
- type casting, 112–113

styles for components

- Alert component style names, 286–287
- available styles, 284–286
- DateChooser component, 287
- DateField component, 287
- defined, 281
- Loader component styles (table), 217
- search order for, 283–284
- setStyle() method, 216–217, 282–284, 287–290
- setting for component type, 283
- setting globally, 282, 287–290
- setting using custom classes, 283, 287

styles variable for custom classes, 283

StyleSheet class

- creating an object, 454
- parseCSS() method, 455
- power of, 453
- setStyle() method, 456–458

StyleSheet objects

- associating with a text field, 454–455
- creating, 454
- defining styles directly on, 456–458

styleSheet property, 454

subclassing, 681–682, 686

suffixes for triggering code hints manually, 10

super domain, 563

swapDepths() method, 161, 173

SWF files

- compressing, 15
- converting Flash video to, 500
- designating as trusted content, 595, 596–597
- Flash authoring tool versus Flash movies, 1
- FLV files versus, 497
- HTML text field support for, 448–450
- local connection between two files, 579–581
- playing in a movie clip, 503–504
- progressive video with, 497
- Protect From Import attribute, 15
- security for files in HTML image tags, 448
- specifying options for publishing, 14–15
- version setting for, 14
- wrapping Flash video in, 498, 502–503

swfInTextTest.fla file, 449–450

switch..case statements

- break statements in, 62
- game keystroke-handling example, 64–66
- if..then..else statement compared to, 60–61
- overview, 60–64
- plastic pop bottle example, 62–64
- syntax, 61

symbolExample.fla file, 6–7

symbols. *See also* buttons; movie clips

- adding from Library, 5
- adding script to instances, avoiding, 8–9
- converting content to, 4
- deleting from Library, 5
- Graphic symbols, 5
- naming, 4–5
- overview, 4–5
- timelines for, 4
- ways of creating, 4

System object

- limitations among player versions, 655
- overview, 653
- querying, 654–655
- setClipboard() method, 655

T**tab size for indenting code, 10**tabChildren **property, 176**tabEnabled **property, 176**tabIndex **property, 176**target **property of event objects, 248–249**_target **property of MovieClip class, 175****test environment, disabling keyboard shortcuts for, 66****text display properties**

align, 440

antiAliasType, 436–439, 452

applying a TextFormat property to substrings, 443–445

bullet, 442

example using all properties available, 441–443

getTextFormat() method for, 440

improving text readability using, 438–439

kerning, 441

letterSpacing, 440–441

precise control over text display using, 437–438

for rich text formatting, 440–441

setNewTextFormat() method for, 445–446

setTextExtent() method for, 442–443

setTextFormat() method for, 440, 444, 445

sharpness, 437

thickness, 437

text fields. See also CSS (Cascading Style Sheets)

applying TextFormat property to substrings, 443–445

associating a StyleSheet object, 454–455

associating TextFormat object to a text field, 441–443

copying formats, 440

createTextField() method for, 160, 163,
434–436, 446

creating on-the-fly, 434–436

CSS with XML and, 459–461

displaying HTML, 446–448

displaying XML, 459–461

getTextFormat() method for, 440

HTML, image and SWF support in, 448–450

improving text readability, 438–439

instance names versus variable names, 434

precise control over text display, 437–438

rich text formatting, 439–446

scrolling using custom scroll bar, 461

scrolling using MovieClip scrollRect() method,
465–467scrolling using scroll and maxScroll properties,
467–469

scrolling using ScrollBar component, 463–465

scrolling using TextArea component, 462–463

setNewTextFormat() method for, 445–446

setStyle() method for, 456–458

setTextExtent() method for, 442–443

setTextFormat() method for, 440, 444, 445

text display properties, 436–439

types of, 433

ways of scrolling, 461

Text menu keyboard shortcuts, 767text **property, 434****TextArea component**

described, 234

scrolling text fields using, 462–463

simple audio interface example, 486–487, 489

transparent instance of, 463

textAreaTest.fla **file, 462–463****TextField class**

displaying properties of, 38

maxScroll property, 469

scroll property, 467, 469

stylesheet property, 454

text versus variable property, 434

TextField data type, 34**TextFormat objects**

applying a property to substrings, 443–445

associating to a text field, 441–443

creating, 440

getTextFormat() method, 440

overview, 339–340

setNewTextFormat() method, 445–446

setTextExtent() method, 442–443

setTextFormat() method, 440, 444, 445

textTest.fla **file, 435–436****themes, 291**thickness **text display property, 437****third-party Flash wrappers, 653, 658–659****this keyword**

associative arrays with, 48

implied, 40

_level10 reference versus, 178, 220

with listener function, 256

overview, 39–40

as relative reference, 178

threshold() **method, 365****thumbnail buttons, creating, 187–190****ThumbnailButton class. See tryItOut_**thumbnailButton **project**thumbnail.jpg **image, 305–306**thumbnailName **property, 681**Time **event, 505**time **property, 410**

Timeline submenu (Insert menu)

Timeline submenu (Insert menu)

- Frame command, 3
 - Keyframe command, 4, 151
- timeline variables**
- access functions for, 120–121
 - accessing in another timeline, 122–129
 - container object for, 120
 - declaring in one place, 119–120
- timelines. See also timeline variables**
- animating in, 370–371
 - base, as parent movie clip, 178
 - for Button symbols, 5
 - circles for keyframes on, 4
 - defined, 370
 - determining which you are editing, 4
 - frame-based versus time-based animation, 376–380
 - main or root, 4
 - for Movie Clip symbols, 7
 - overview, 3
 - placing images on, avoiding, 145
 - for symbols, 4
 - white versus gray frames on, 4
- TodayStyle class, 287**
- Tools menu keyboard shortcuts, 761**
- Tools panel**
- color swatches on, 6
 - keyboard shortcuts, 759–760
 - overview, 2, 3
- toString() method**
- unescape() function with, 528
 - XML class, 538, 540
- totalFrames method (JavaScript), 607**
- _totalframes property, 175**
- toUpperCase() method, 102, 682**
- trace() function**
- debugging with, 311–312, 313
 - displaying variable contents, 28–29
 - this keyword with, 39–40
- trackAsMenu property, 176**
- tracking**
- data tracking example, 49–50, 73–76
 - pixel in webcam in real time, 519–521
- trackPixel.fla file, 519–521**
- transform property, 176**
- transitions. See easing**
- transparency**
- in Flash video, 510–511, 514
 - performance and, 196–197
 - for TextArea component instant, 463
- transparent property, 365**

Tree component, 234, 277

- trigger() method, 281
- trimWhiteSpace static method, 684–685
- tryItOut_accessTimelineVars project, 123–126
- tryItOut_assignID project, 126–129
- tryItOut_attachMovie project, 182–184
- tryItOut_attachWithScript project, 239–241
- tryItOut_bindingComponents project, 275–276
- tryItOut_createButton project, 179–181
- tryItOut_debug project
 - commenting out code, 313
 - developing a theory, 305–306
 - using Debugger panel, 310
 - using trace() function, 311–312
- tryItOut_debuggerPractice project, 308–310
- tryItOut_decisions project, 65–66
- tryItOut_downloadFiles project, 641–645
- tryItOut_easingIn project, 393–395
- tryItOut_easingOut project, 389–391
- tryItOut_externalInterface project, 621–629
- tryItOut_firstAnimation project, 370–371
- tryItOut_frameRate project, 378–379
- tryItOut_helicopterAnimation project, 396–402
- tryItOut_helloWorld project
 - creating a class, 662–663
 - making code private, 664–665
- tryItOut_listenerFunction project, 256–257
- tryItOut_listenerObject project
 - radio button and check boxes example, 250–254
 - simple button example, 249–250
- tryItOut_maskEffect project
 - animated masking effect example, 191–193
 - mouse-driven masking effect example, 194–195
- tryItOut_onEnterFrame project, 372–374
- tryItOut_openWindow project, 633–634
- tryItOut_passingFunctions project, 89–92
- tryItOut_refactor project, 133–140
- tryItOut_screenSaver project, 426–432
- tryItOut_scriptingComponents project
 - adding an event listener, 257–262
 - adding an XMLConnector, 278–281
 - basic image viewer, 236–237
 - handling events from multiple sources, 266–270
 - setting styles for components, 287–290
 - skinning components, 291–294
 - using code instead of Parameters panel, 243–244
- tryItOut_setInterval project, 374–376
- tryItOut_shutterMask project, 191
- tryItOut_snowStorm project, 383–386
- tryItOut_stringLib project, 683–686

`tryItOut_thumbnailButton` **project**

adding methods, 670–675

adding properties, 677–681

creating the class, 666–669

`tryItOut_thumbnailButtons` **project, 187–190**

`tryItOut_trackData` **project, 49–50, 73–76**

`tryItOut_tween` **project**

bouncing elasticized box into position, 411–414

multiple tweens with, 414–417

`tryItOut_tweenPlayback` **project, 418–419**

`tryItOut_tweenPlaybackLib` **project, 419–424**

`tryItOut_understandableCode` **project, 113–116**

`tryItOut_uploadFiles` **project, 646–650**

`tryItOut_variableScope` **project, 95–96**

`tryItOut_xmlQueue` **project, 690–695**

Tween class. See also tweens; specific methods

`clipHandle` parameter, 403

`duration` parameter, 403

`easingFunction` parameter, 403

events (table), 411

fading a movie clip, 404

general form, 403

methods (table), 405–406

moving a movie clip, 404

properties (table), 410

`propertyName` parameter, 403

`startValue` parameter, 403

`stopValue` parameter, 403

`useSeconds` parameter, 403

tweens. See also Tween class

absolute versus relative, 404

defined, 370

elasticized box example, 411–414

playing in parallel, 414–417

playing in parallel and in sequence, 418–424

playing in sequence, 417–418

type casting, 112–113, 116

type property of event objects, 248

U

UIScrollBar component, 234

undefined keyword

for array elements, avoiding, 41

for undeclared variable value, 29, 42

underscore (_), forcing folder order using, 144

understandability

creating understandable code, 113–116

as critical aspect of programming, 83

debugging and, 303–304

as goal of OOP, 107

self-documenting code for, 30, 116, 304

`unescape()` **function, URL encoding and, 528**

uninitialized variables, debugging, 299–301

`unloadClip()` **method, 206, 208–209**

unloading

external content, 173–174

movie clips, 208–209

`unloadMovie()` **method, 161, 173–174**

Up state of buttons, 5, 6

`updateAfterEvent()` **function**

animation smoothness and, 377, 391, 395

easing in example, 395

easing out example, 391

experimenting with, 378–379

helicopter animation example, 398, 500

`updateAnimation()` **function**

easing in example, 393–394, 395

easing out, 389, 390–391

helicopter animation example, 397–398, 399–400

`updateSeeds()` **function, 427, 428, 430, 431–432**

`upload()` **method, 637, 639**

uploading files

base code for, 645–646

example, 646–650

previous Flash versions and, 637

uppercase. See case

`_url` **property, 175**

URLs

absolute addresses for, 187

`escape()` function for encoding, 528

fully qualified addresses for, 186, 187

loading and parsing raw text, 532–534

relative addresses for, 187

super domain of, 563

`unescape()` function for encoding, 528

usability, refreshing data and, 523–524

`useHandCursor` **property, 176**

user interaction. See event handlers; event listeners;

events

V

validation, straight code versus functions for, 86–87

var keyword, 27

variable bit rate compression, 499

variable property, 434

variables. See also data types; scope of variables;

strong typing

accessing in another timeline, 122–129

assigning data to, 28

variables (continued)

- blue in editor, 29
- Boolean, naming, 108
- camelBack style names, 107–108
- constants versus, 30
- declaring, 27–28, 31, 119–120
- defined, 27
- dot syntax, 37–38, 102
- duplicating names, avoiding, 96
- FlashVars, 588–593
- functions as, 88
- global, defined, 92
- global, minimizing use of, 39, 95
- grouping in a container object, 120
- iterator variables, 67, 69, 108
- `_level0` keyword, avoiding, 39
- local, 92
- naming, 29–30, 106, 107–110, 113–116
- in nested movie clips, accessing, 122–129
- null value, 29, 42
- passing data between, 29, 35–36
- referencing movie clips using, 178, 180
- `_root` keyword, avoiding, 38–39
- side effects of functions and, 96, 97
- suffixes suggested for data types (table), 111–112
- text field instance names versus variable names, 434
- `this` keyword, 39
- undefined value, 29, 42
- uninitialized, debugging, 299–301
- viewing contents of, 28–29

vector graphics. See also drawing; specific methods

- creating a simple square, 326–327
- drawing API for, 320
- drawing Bezier curves, 327–329
- graphs, 329–333
- Matrix object, 334
- methods, 320–325
- rasterizing, 355

video. See Flash video; SWF files; specific formats

Video class

- displaying Camera feed on stage, 517–518
- methods, 506
- properties (table), 506–507

View menu keyboard shortcuts, 761, 764–765

viewing. See displaying

visibility of variables. See scope of variables

_visible property

- MovieClip class, 153, 175, 197
- performance improvement and, 197
- Video class, 507

volume

- `getVolume()` method for, 475, 483
- `setVolume()` method for, 475, 483

VolumeBar component, 232

W

`watchMyXMLLoad()` function

- loading files and extracting values, 549
- pre-loading XML, 546, 547

`watchThumbLoad()` function, 449–450

waving flag effect, 345

weak typing, 30–31

Web browsers. See browsers

Web pages, replacing using `getURL()`, 166–167

webcams. See also Camera object

- detecting motion, 518
- displaying feed on stage, 517–518
- tracking in real time, 519–521

WebServiceConnector component, 231

WeekDayStyle class, 287

while loops

- for complex termination conditions, 71–72
- `do..while` loops versus, 72
- duplicating a `for` loop, 71
- `for` loops versus, 71, 72
- infinite loops, 77–78, 302
- order of execution, 71
- overview, 70–72
- syntax, 71

white space in code

- `ignoreWhite` property (XML class) and, 541, 542, 547
- overview, 25–26, 118–119

width property

- bitmapData object, 365
- Camera class, 516
- Video class, 506

_width property

- MovieClip class, 175
- Video class, 507

wildcards

- for domain access, 566, 567
- with `import` statement, 238

Window component, 234

Window menu

- Actions command, 8
- Components command, 89
- keyboard shortcuts (drawing IDE), 768–769
- Library command, 4, 89
- Workspace Layout submenu, 1

`window.open()` **method (JavaScript)**, 629, 630, 631–632

X

`_x` property

MovieClip class, 175
Video class, 507

XML

attributes, 536, 552
CDATA wrapper for HTML strings, 537
cross-domain security, 562–573
CSS with, 459–461
defined, 535
displaying in text fields, 459–461
DOM interface, 549
elements, 536–537
loading external files, 542–545
loading files and extracting values, 549–551
measuring bytes loaded, 545
navigating a node tree, 549–551
overview, 535–537
parent-child node relationship, 549–552
security, testing for, 545
simple file example, 536
socket connection, 552–557
tags, 535–536
text-based pre-loader, 545–548
using HTTP GET and POST, 557–559
white space in, 542

XML class

accessing with `new` keyword, 537
event handlers, 541–542
`ignoreWhite` property, 541, 542, 547
methods, 538–539
properties (table), 541

XML files

as data source for components, 277–281
examples, 277–278, 279
XML schema for, 277, 278, 279, 281

XML schema

defined, 277
for XMLConnector example, 278, 279, 281

XMLConnector component

adding to image viewer, 278–281
binding data to display, 561–562
Component Inspector panel and, 273
described, 231
GET and POST parameters, 561
loading an XML file using, 559–561

`XMLConnectorSample.fla` **file**, 561–562

`XMLConnectSample.fla` **file**

binding XMLConnector data to display, 561–562
loading an XML file using XMLConnector, 559–561

`XMLDataQueue` **method**, 690

`xmlDecl` **property**, 541

`xmlLoadSample.fla` **file**

loading external XML files, 543–545
loading files and extracting values, 549–551
pre-loading XML, 546–548

`XMLQueueClass.as` **file**, 690–695

XMLSocket class

creating a socket request and receiving an answer, 555
creating an initial socket connection, 554
events (table), 554
free socket server for testing, 554, 555
methods (table), 553
null bytes and, 553
requirements for using, 552
XML not required for, 553

`xmlWithCSS.fla` **file**, 459–461

`_xmouse` property

MovieClip class, 175
Video class, 506

`_xscale` property

MovieClip class, 175
Video class, 507

XUpdateResolver component, 231

Y

`_y` property

MovieClip class, 175
Video class, 507

`_ymouse` property

MovieClip class, 175
Video class, 507

`yoyo()` **method**, 406, 410

`_yscale` property

MovieClip class, 175
Video class, 507

Z

zero

divide-by-zero error with pre-loaders, 200
as starting integer for array indexes, 40, 76

`zoom` **method (JavaScript)**, 607

`zoomIn()` **method**, 688, 689

`zoomOut()` **method**, 687–688, 689

