

# Index

## A

**abstract classes, 119–120**

**abstract methods, 119–120**

**abstract value types, 35–36**

- dynamic type, 35–36
- garbage collection, 572–573
- Neko and C/C++, 563–565
- null type, 35
- void type, 35

**accessor methods, with dynamic keyword, 113**

**Action Message Format (AMF) server, 426**

- Flash-to-AMF server connection, 434

**ActionScript**

- and Flash, 316
- compared to haXe. *See* ActionScript/haXe comparison
- Screen Weaver HX extensions, 523–524
- synchronous communication in, 523–524
- version 3 and Flex 2, 344

**ActionScript/haXe comparison**

- classes, 357–359
- constant values/special values, 346–347
- data types, 344–346
- functions, top-level, 347–352
- keywords, 354–358
- operators, 352–353
- properties, 356
- statements, 354–358
- structures, 354–358
- trace, 181

**ActionScript Virtual Machine 2, 316**

**ActionStep library, 369**

**add**

- lists, 47
- strings, 71–72

**addChar, 71–72**

**addition (+), binary operator, 55**

**addSub, 71**

**Adobe Flash CS3, benefits to use, 363**

**Adobe Flex library, 369**

**AIR, 510**

**AJAX, 228, 414–422**

- auto-complete control, building, 417–422
- functions of, 414
- HTML panel content, updating, 416–417

**alloc\_root function, 571–572**

**alloc function, 572**

**alternation, regular expressions, 220**

**anchors, regular expressions, 217–218**

**AND (&), bitwise operator, 56, 62**

**AND (&&), logical comparison operator, 59**

**animation, Neko Media Engine (NME), 549**

**anonymous keyword, 28**

**anonymous objects, 126–127**

- declaring, syntax for, 126
- typedef, use with, 128–130

**Apache web server**

- installing on Linux Ubuntu, 233–234
- installing on Windows, 232–233

**application programming interface (API)**

- Flash API classes, 359–361
- Flash drawing API, 337–342
- haXe API, 206–208
- reflection API, 464–469
- Remoting API communication methods, 427–428
- serialization API, 473–474

**arctic project, 146**

**arithmetic operators. *See* binary operators**

**array(s), 40–45**

- adding/removing items from, 42–43
- array access operator, 352
- array fields/descriptions, 41–42
- concatenating, 45
- converted to strings, 68
- converting string to, 67
- copying, 44–45
- data type, changing, 41
- functions of, 41
- hash tables, 48–50
- initializing with literal values, 41
- instantiating, 41
- iterator, creating, 98
- lambda class for modifying, 96–98
- lists, 46–48
- multidimensional, 45–46
- Neko and C/C++, 567–568
- sort method, 96
- typing rules for, 40
- values, conversion methods, 43–44

**array method, lambda class, 98**

**Array string, 66**

- aspell project, 146**
- assignment operators**
  - types of, 56
  - uses of, 59–60
- asterisk, block commenting, 37**
- AsWing library, 369**
- asynchronous communications, 426**
  - AsyncAdapter, 437
  - AsyncConnection, 432–434, 440
  - AsyncDebugConnection, 437
  - AsyncProxy, 438
  - Flash-Neko communication, 518–520
- attr expressions, Templo, 262**
- attributes, HTML, 589–601**
  - attribute descriptions, 590–599
  - attribute groups, 589
  - attribute value types, 600–601
- attributes, XML**
  - accessing, 457–458
  - checking, 461–462
- Author class, SPOD system, 287–288, 296–297, 299**
- AuthorManager class, SPOD system, 300–301**
- auto-complete control, building, AJAX technique, 417–422**

## B

- back-references, regular expressions, 214, 220**
- backward slash, escaped characters, 33–34**
- baseEncode, 69**
- bfailure function, 574**
- BinaryNode class, 123**
- binary operators, 56–58**
  - increment/decrement, 57–58
  - modulo operator, 57
  - order of precedence, 56–57
- Bit Block Transfer, 540**
- bitwise mathematics, utility of, 60**
- bitwise operators, 60–63**
  - filtering bits, 62–63
  - shifting bits, 61–62
- block commenting, 37**
- blog entry, constructors, use of, 108–110**
- bool**
  - standard data type, 28
  - type data conversion, 39
- Boolean(s)**
  - functions of, 33
  - restrictions in haXe, 33
- boot classes, magic, 475**
- break, with loops, 84–85**
- bseDecode, 69**
- buffers, display buffer, flipping, 535–536**
- builtins, Neko, 477**

- buttons**
  - Flash user interface, 376, 377
  - nGui, 488–489
- byte(s)**
  - bytecode, Neko, 156–157
  - extracting from stream, 306
  - readBytes method, 306
  - writeBytes method, 308–309

## C

- CalcProxy class, 438**
- calculation expressions, 253**
- callback**
  - macros in templates, 256–257
  - nGui List control, 491–492
- callStack function, 191–193, 196–199**
- carriage return, escaped characters, 34**
- case keyword, and switch statements, 77**
- casting, 38–40**
  - cast function, operation of, 38–39
  - conversion methods, types of, 39–40
  - type of value, comparing, 40
  - unsafe cast, 38–39
- catchall block, for exceptions, 189**
- caurinaTween project, 146**
- C/C++, with Neko. See Neko and C/C++**
- ceil, float to int conversion, 66**
- Chapter class, SPOD system, 296–297, 299–300**
- ChapterManager class, SPOD system, 300–301**
- character(s)**
  - patterns, regular expressions, 214–215
  - readChar method, 305–306
  - serialization, 473–474
  - writeChar method, 308
- character classes**
  - non-alphanumeric characters, 217
  - range of values, 216
  - regular expressions, 215–217
  - syntax for, 216
- character sets, regular expressions, 217**
- charAT, 66–67**
- charCodeAt, 66–67**
- Check class, XML, 458**
- child controls, Flash user interface, 376–381**
- chr, type data conversion, 39**
- class(es), 102–103, 122–126**
  - abstract classes, 119–120
  - ActionScript/haXe comparison, 357–359
  - advanced, 122
  - exception class, 194–199
  - extensions, use of, 133–134
  - extern classes, 150, 160
  - functions, 102–103
  - functions of, 102

implementing, 123  
 name, elements of, 102  
 public and private, 102  
 syntax for, 102  
 template class, 252  
 type parameters, 123–126  
 variables, 102–103

**class blocks, HaXe program structure, 23**

**class functions, 85–90**  
 functions of, 85  
 structure of, 85–86

**class keyword, 28**

**clear, lists, 47**

**clear function, trace, 184**

**code**  
 documenting, 166–170  
 libraries, 145–162  
 packages, 139–145  
 resources, 163–166  
 unit testing, 170–177

**collection(s)**  
 functions in, 94–96  
 items, accessing directly, 83  
 looping over, 81–82

**collisions, detecting, Neko Media Engine (NME), 543–545**

**color key, Neko Media Engine, 540**

**commenting**  
 block commenting, 37  
 HTML, 588  
 line commenting, 37  
 regular expressions, 220–221

**comparison expressions, templates, 253**

**comparison operators, logical comparison operator, 58–59**

**compiler directives, functions of, 6**

**concatenation, arrays, 41, 45**

**conditional(s), regular expressions, 220–221**

**conditional statements, 74–79**  
 if statement, 74–77  
 switch statement, 77–78  
 values, returning from, 78–79

**config project, 148**

**configStr, 68**

**confirm message box, SysTools, 530**

**Connection class, Neko database support, 277–278**

**constant values, ActionScript/haXe comparison, 346–347**

**constant variables, 31**

**constraint parameters, 125–126**

**constructor(s), 108–110**  
 constructor arguments, 136–138  
 enum arguments, 136–138  
 of instance variables, 108–110  
 overriding in inheritance, 118  
 syntax for, 108

**containers**  
 Flash user interface, 376–377  
 nGui, 487–488

**continue, with loops, 84–85**

**Control class, nGui, 485–487**

**controls**  
 auto-complete control, AJAX technique, 414–422  
 Flash user interface, 375–376  
 HTML, 585–586  
 nGui library, 482

**copying, arrays, 41, 44–45**

**createDirectory method, Neko, 303**

**createdOn, 112, 122**

**cryptographic hashes, 224**

## D

**databases, Neko. See Neko database support**

**data conversion, casting, 38–40**

**data types, ActionScript/haXe comparison, 344–346**

**date/time, 50–54**  
 components of, retrieving, 52–53  
 date method/descriptions, 50–51  
 date objects, creating, 51–52  
 delta( ) function, 53  
 format( ) function, 53–54  
 getMonthDays( ), 54  
 tokens/descriptions, 54

**dbName method, 281**

**dcxml project, 146**

**debugging, 179–199**  
 exception class, 194–198  
 exceptions, 185–194  
 trace, 179–185

**declarations, variables, 29–30**

**decrement (-), binary operator, 55, 57–58**

**default**  
 haXe standard library, 145  
 variable value modifier, 111

**default case, 135**

**default keyword, and switch statements, 78**

**delayed actions, timer, 223**

**DelayedConnection, 437**

**deleteDirectory method, Neko, 303**

**deleteFile method, Neko, 303**

**delimiters, string(s), 67**

**delta function, dates, 53**

**dialogs, SysTools, 529–530**

**directory management**  
 create/delete directory, 303  
 Neko, 303–304  
 reading from directory, 303–304

**display buffer, flipping, 535–536**

**division (/)**  
 binary operator, 55  
 divide by zero issue, 64–65  
 modulo operator, 55, 57

## **DNS (Domain Name Server), 227**

### **documentation of code, 166–170**

offline, 167–169

online, 169–170

### **document definition elements, HTML, 577–578**

### **Document Object Model (DOM), 392–396**

definitions in file, 393

typedef declarations, 394–396

### **Document Type Definition (DTD), 229–230**

### **Dojo toolkit, 399**

### **dot expressions, 253–254**

### **double quotes, escaped characters, 34**

### **dowhile loop, 80–81**

### **drag events, 515**

### **drawing**

to display, Neko Media Engine (NME), 538–540

Flash API, 337–342

### **dynamic**

functions, 87–88

keyword, accessor methods with, 113

links, Flash, 154–156

standard data type, 28

variable value modifier, 111

### **dynamic type, 127–133**

parameters, 127–128

typedef, 128–131

using, 35–36

### **dynamic typing, variables, 29**

catchall block, 189

catching exceptions, 187–189

complex, dealing with, 189–190

elements of, 185–186

exception class, 194–199

exceptionStack function, 193–194, 196

manifestation of, 186–187

throwing exceptions, 188

unexpected, catching, 191

### **exception class, 194–199**

code for, 197–199

creating, 194–195

debug functionality, 195–197

generic output function, 197

### **excerpt variable, 112**

### **execute method, Web development, 242**

### **exists method, 303**

hash method, 49

### **explicit import, 142–143**

### **expressions, regular. See regular expressions**

### **extensions, 133–134**

with classes, 133–134

with typedef, 134

### **external libraries, 150–162**

extern classes, 150, 160

Flash, 151–156

JavaScript, 159–162

Neko, 156–158

### **Ext library, 400**

## **E**

### **element**

Flash user interface, 374–375

HTML, 226–227

### **else keyword, and if statement, 75–76**

### **encoding, strings for web, 71**

### **endsWith, 69–70**

### **enum, 134–136**

Condition, 136

constructor arguments, 136–138

keyword, 28

and packages, 143

and switch, 135, 143

syntax for, 134

### **equal (==), 56, 58–59**

### **EReg class, 210–214**

### **error handling, Neko and C/C++, 573–574**

### **escaped characters, string content, 33–34**

### **events**

Flash, 328–331

Flash user interface, 371–373

hxGtk library, 502–503

Neko Media Engine (NME), 549–553

Screen Weaver HX window events, 515–516

### **exception(s), 185–194**

callStack function, 191–193, 196–199

## **F**

### **failure function, 573–574**

### **fash package project, 146**

### **Fast class, XML, 455–456**

### **fmx project, 146**

### **field access operator, 353**

### **FieldLayout, Flash user interface, 381**

### **file(s), database**

appending to existing file, 308

deleting, 303

directory management, 303–304

File class, Neko, 304–311

FileSystem class, Neko, 301–302

path strings, 302–303

read data directly, 307

read data from stream, 305–306

remote files, accessing, 311

traversing stream, 309–310

write data to stream, 307–309

### **FileInput stream, 304–305**

### **FileOutput stream, 304–305**

### **fill expression, Templo, 265**

### **fill List function, 440**

### **filters**

filtering bits, 62–63

- lists, filter method, 95
- XML, rules, 462–463
- Firebug**
- Firebug, haXe trace redirected to, 389–390**
- Firefox**
  - Firebug, 389–390
  - Web Developer extension, 389
- first, lists, 47**
- flags, Screen Weaver HX window flags, 513**
- Flash, 315–385**
  - API classes, 359–361
  - compiling to, 5–6
  - development of, 316
  - drawing API, 337–342
  - dynamic links, 154–156
  - events, 328–331
  - extending methods, for MovieClip class, 342–344
  - external library, 151–156
  - FlashJsConnection, 435
  - Flash-Neko communication, 517–524
  - Flash-to-Action Message Format (AMF) Server connection, 434
  - Flash-to-JavaScript connection, 429–431
  - functions of, 315
  - to haXe. See Action Script/haXe comparison
  - .html file compiled to, 18
  - JavaScript-to-Flash connection, 432
  - LocalConnection, 435
  - magic, 477–478
  - Remoting API communication methods, 427–428
  - and Screen Weaver HX, 517–524
  - slideshow, 331–337
  - SocketConnection, 436
  - static links, 152–154
  - SWFMill, 364–369
  - SWF output, 6
  - switches, 24
  - TicTacToe Game, 443–452
  - trace output, 180
  - Unicode support, 221–222
  - user interface, creating. See Flash user interface
  - version 9 differences, 156
- Flash movies**
  - clip, creating in version 9, 321–322, 324–328
  - clip, creating in versions 6 through 8, 319–321, 324–330
  - duplicating, 321
  - embedding in page, sample code, 318
  - embed technique, problems/solutions, 318–319
  - file extension (.swf), 316
  - images in, 323–324
  - mask, use on movie clip, 342–344
  - MovieClip class, 319–322, 342–344
  - sounds, 325
  - stage and timeline, 316–317
  - text in, 322–323
  - values, loading, 325
  - variables, loading, 326–327
  - XML, loading, 327–328
- flash package, 145**
- Flash Player**
  - Detection Kit, 319
  - express install, 319
  - parameters, determination of, 318
  - purpose of, 315, 317
- Flash user interface, 369–385**
  - containers, 376–377
  - controls, 375–376
  - element fields, 374–375
  - events, 371–373
  - file structure, 370–371
  - Flash 9 implementation, 381
  - layout constructors, 374
  - library, building, 369–371
  - tree editor, example of, 381–384
- Flex 2, 344**
- Flex Builder IDE, compiling to Flash, 6**
- flex project, 146**
- float**
  - elements of, 32
  - to int conversion, 65–66
  - standard data type, 28
- floating-point numbers, double precision, 32**
- floor, float to int conversion, 65–66**
- fmod project, 146**
- fold, lambda class, 98**
- fonts, True Type, embedding in SWFMill movie clips, 368–369**
- fonttools project, 146**
- for loop, 81–82**
  - break and continue with, 84–85
  - IntIter object, 81–82
  - IntIter operator (...), 82
- format( ) function, dates, 53–54**
- forms**
  - HTML, 585–586
  - HTML, data validation, 404–414
- forward slashes, block commenting, 37**
- framework\_delfiweb project, 147**
- free\_root function, 572**
- fromTime, 51**
- front controller, Neko as, 238–239**
- FTP (file transfer protocol), 226**
- fullpath method, 302**
- function(s), 85–98**
  - ActionScript/haXe comparison, 347–352
  - of classes, 102–103
  - class functions, 85–90
  - in collection objects, 94–96
  - dynamic functions, 87–88
  - elements of, 85
  - function type inference, 87
  - identifiers sequence, 108
  - instance functions, 104–107
  - JavaScript, 160
  - lambda class, 96–98
  - local functions, 90–96

## function(s) (continued)

- multi-function recursion, 92–93
- Neko and C/C++, 568–570
- overriding in inheritance, 117–118
- recursive functions, 88–90
- return from, 86–87
- static functions, 114–115
- type parameters, use of, 124–125

**function keyword, 28**

## G

**games, developing. See Neko Media Engine (NME)**

### garbage collection

- abstract values, 572–573
- Neko and C/C++, 571–573

**get, hash method, 49**

**getContent, 123, 307**

**getDate, 50, 51**

**getDay, 50, 51**

**getFullYear, 50, 51**

**getHours, 50, 53**

**GET method, web server, 227–228**

**getMinutes, 50, 53**

**getMonth, 50, 51**

**getMonthDays( ), 54**

**getOnlineInfo, 110**

**getSeconds, 50, 53**

**getStream method, 311–312**

**getter method, variable value modifier, 111**

**getTime, 51, 53**

**Glade, 503–506**

- downloading, 504
- Glade class, 505–506
- with hxGtk library, 503–506
- new project, starting, 504–505

**global cache, SPOD system, 295**

**global constants, ActionScript/haXe comparison, 346–347**

**global functions, ActionScript/haXe comparison, 347–352**

**global properties, ActionScript/haXe comparison, 356**

### graphical user interface (GUI) development

- with Flash. See Flash user interface
- with Glade, 503–506
- hxGtk library, 499–506
- installing libraries, 481
- libraries, 369, 480–481
- with Neko. See Neko desktop applications

**greater than (>), 55, 58–59**

**greater than or equal to (>=), 55**

**GridLayout, Flash user interface, 379–381**

**grouping operators ( ), 55, 57**

**groups, regular expressions, 219**

**GTK. See hxGtk library**

**Gtk class, 501**

**gtranslator project, 147**

## H

### hash tables, 48–50

- adding/removing items from, 49–50
- cryptographic hashes, 224
- functions of, 48
- hash methods/descriptions, 49
- iterating over, 84
- keys, use of, 48–49
- querying keys in, 49–50

**hasNext( ), 81**

### haXe

- abstract value types, 35–36
- arrays, 40–45
- basic operation of, 8
- basic requirements, 9
- benefits to use, 5, 7, 21, 73–74
- casting, data conversion, 38–40
- C/C++ with, 555–575
- commenting, 37
- compiling to JavaScript, 6–7
- compiler directives, 6
- compiling, basic operation, 17–19
- compiling to Flash, 5–6
- compiling to JavaScript, 6–7, 21–22
- compiling to Neko, 7, 19–21
- conditional statements, 74–79
- cross-platform. See regular expressions; Timer class; XML, Message0-Digest 5 (MD5)
- databases, working with. See Neko database support
- dates, 50–54
- debugging, 179–199
- file extension (.hx), 8
- file extension (.hxml), 17–19
- and Flash, 315–385
- functional programming support, 7
- functions, 85–98
- hash tables, 48–50
- installation of, 12–14
- and JavaScript, 387–423
- keywords, 28
- language, capabilities of, 7
- libraries, 8
- lists, 46–48
- loops, 79–85
- magic, 475–478
- main switch, importance of, 19
- math class, 63–66
- as object-oriented language, 27, 74
- operators, 55–63
- program structure, 22–23
- and reflection API, 464–469
- remoting, 425–452
- Runtime Type Information (RTTI), 470–473
- serialization, 473–474
- simple value types, 31–34
- standard data types, 28
- string functions, 66–72

- switches, 23–25
  - templates, 251–268
  - text editor, use with, 17
  - untyped block, 36–37
  - variables, 28–31
  - web development, 230–250
  - Web site, 9, 12
  - and XML, 455–464
  - haxealtdate project, 147**
  - haxedoc**
    - offline documentation, 167–169
    - syntax for, 167
  - haxe.Http class, 415**
  - haxelib libraries, 146–149**
    - haxelib tool, use of, 148–149
    - included projects, listing of, 146–148
    - installing, 148–149
  - haxelib-test project, 147**
  - haXe.Log class, 181–185**
  - haxe.Log trace, 181–182**
  - haxe package, 145**
  - haxe.PosInfos, 184**
  - haxe.Template, compared to mtwin.Templo, 261**
  - haxe.unit package. See unit testing**
  - haxORMap project, 147**
  - Hello World**
    - in C/Neko compatible, 556–559
    - compiling, basic operation, 18–22
    - creating, 17
  - here identifier, trace, 182–184**
  - hex, 69**
  - hook values, custom events with, 516–517**
  - HTML, 228–231**
    - and AJAX, 414–422
    - attribute descriptions, 590–599
    - attribute groups, 589
    - attribute value types, 600–601
    - comments, 588
    - controls, 585–586
    - data validation in forms, 404–414
    - document definition elements, 577–578
    - Document Object Model (DOM), 392–396
    - document structure, 229–230
    - elements in, 226–227
    - files, root location, changing, 21–22
    - forms, 585–586
    - compared to haXe, 230–231
    - hyperlink format, 583
    - image inclusion, 583
    - link elements, 588
    - lists, 581–582
    - maps, 584
    - rules, adding to documents, 405
    - scripts, 587
    - structural elements, 578–581
    - style sheets, 586–587
    - syntax for, 229, 230
    - tables, 582–583
    - table sorting, 401–404
    - tags in, 226–227
    - web server, flow in, 226
  - htmlEscape, 69, 71**
  - htmlUnescape, 69**
  - HTTP protocol, 226**
    - haxe.Http class, 415
    - JavaScript-to-Neko HTTP server, 433
  - HTTPS protocol, 226**
  - hxasm project, 147**
  - hxDev project, 147**
  - hxDiff project, 147**
  - hxGtk library, 499–503**
    - events, 502–503
    - function proxy, 499–501
    - Glade, use with, 503–506
    - Gtk class, 501
    - pros/cons of, 481
  - HxGtk project, 147**
  - HxJSON project, 147**
  - HxLib project, 147**
  - hxScriptlet project, 147**
  - hxServlet project, 147**
  - hxSynth project, 147**
  - hyperlinks, format of, 583**
- |
- identifiers, instance fields, sequence of, 107**
  - if statement, 74–77**
    - nesting, 75–76
  - images**
    - in Flash movie, 323–324
    - HTML, 583
    - in Neko Media Engine, 538
  - implicit import, 141–142**
  - importing**
    - packages, 141–145
    - templates into templates, 256
  - increment (++), binary operator, 55, 57–58**
  - index, database fields, accessing by, 284**
  - indexOf, 66–67**
  - inequality (!=), 56, 58–59**
  - inference, function type, 87**
  - info projectname project, 148**
  - Infos interface, magic, 477**
  - inheritance, 115–121**
    - abstract classes/methods, 119–120
    - constructors, overriding, 118
    - defined, 115
    - extending class, steps in, 115–116
    - functions, overriding, 117–118
    - and instance variables, 120
    - roles, different at same time, 120–121
    - single inheritance, 115
    - and static fields, 120
    - super identifiers, 117
    - toString, 119

## **insert, arrays, 42, 44**

## **Inspect method, RTTI, 471–473**

## **install projectname project, 148**

## **instance fields, 104–114**

- constructors, 108–110
- functions sequence, 108
- identifiers sequence, 107
- inconsistencies, avoiding, 105
- instance functions, 104–105
- optional arguments, 106–107
- variables, 107
- variable value modifiers, 110–114

## **instance functions, 104–107**

- null arguments, 105–106
- optional arguments, 106–107
- syntax for, 104
- XML, 207

## **instance variables, and inheritance, 120**

## **instantiating, arrays, 41**

## **int**

- standard data type, 28
- type data conversion, 39

## **integers, bits allowed, 32–33**

## **interfaces, 121–122**

- body, structure of, 121–122
- declaration, syntax for, 121
- uses of, 122

## **IntHash, 84**

## **IntItr object, 81–82**

## **IntItr operator (...), 82**

## **isEmpty, lists, 47**

## **is method, values, comparing types of, 40**

## **isOnline, blog entries, 110**

## **isSpace, 69**

## **Iterable, typedef, 131–133**

## **iteration**

- arrays, 42, 98
- database, with ResultSet, 285–286
- hash method, 49
- over hash tables, 84
- IntItr object, 81–82
- lists, 47, 98
- and loops, 80
- template loop expressions, 254–256

## **iterator**

- method, 82–83
- typedef, 131–133

## **J**

## **JavaScript, 387–423**

- AJAX, 414–422
- benefits to use, 422
- compiling to, 6–7, 21–22
- cross-browser event system solution, 397–399
- execution errors, redirecting, 392

external libraries, 159–162

Flash content, embedding with, 319

FlashJsConnection, 435

Flash-to-JavaScript connection, 429–431

functions, use of, 160

HTML, traversing with, 404–414

HTML Document Object Model (DOM), 392–396

HTML files, necessity of, 21–22

JavaScript-to-Flash connection, 432

JavaScript-to-Neko HTTP server, 433

libraries, 391–392, 399–400

low-level access to, 396

magic, 478

Mozilla compared to Internet Explorer, 7

Object Notation (JSON), 414

OnLoad problem, 396–399

output size, modifying, 390

Remoting API communication methods, 427–428

SocketConnection, 436

switches, 24

table sorting, 401–404

trace output, 180

Unicode support, 221–222

## **join**

arrays, 42, 68

lists, 47

## **jQuery library, 400**

## **js package, 146**

## **K**

## **keyboard, events, Flash user interface, 372–373**

## **keying, color key, Neko Media Engine, 540**

## **keys, hash tables, 48–49**

## **keywords**

- ActionScript/haXe comparison, 354–358
- of haXe, 28

## **kind method, Neko, 304**

## **kother method, Neko, 304**

## **L**

## **labels**

- Flash user interface, 376
- nGui Label control, 484–485

## **Lambda class, 96–98**

- array, 98
- fold, 98
- map, 97–98
- mapi, 97–98
- methods/description, 97

## **last, lists, 47**

## **lastIndexOf, 67**

**lastInsertID method, 281****layout constructors, Flash user interface, 374****left shift (<<), binary operator, 55****length**

- arrays, 41
- database property, 286
- lists, 47
- string, 66

**less than (<), 55, 58–59****less than or equal to (<=), 55****lhx project, 147****libraries, 145–162**

- external libraries, 150–162
- features of, 8
- GUI libraries, 480–481
- haxelib libraries, 146–149
- hxGtk library, 499–503
- JavaScript, 391–392, 399–400
- Neko, 272–273
- from other projects, use of, 150
- standard haXe, 145–146
- SysTools, 525–530

**line commenting, 37****links**

- dynamic links, 154–156
- HTML hyperlinks, 583
- HTML link elements, 588
- static links, 152–154

**Linux**

- HaXe installation on, 13–14
- Neko installation on, 16–17
- Ubuntu, Apache installation on, 233–234

**list(s), 46–48**

- adding/removing items from, 48
- filter method, 95
- functions of, 46–47
- HTML, 581–582
- iterator, creating, 98
- list fields/descriptions, 47
- map method, 94–95
- querying values in, 48

**List control, nGui, 491–495****list method, lambda class, 98****list project, 148****LocalConnection, 435****local functions, 90–96**

- multi-function recursion, 92–93
- passing functions to, 92
- structure of, 90
- type structure, 91
- variable scope, 93–94

**Logger class**

- data requirements, 311–312
- Neko, 311–314
- stream for output, accepting, 311–312

**logical comparison operators, 58–59****logical expressions, 254**

- Templo, 263

**look-around constructs**

- regular expressions, 220–221
- syntax for, 221

**loops, 79–85**

- break and continue, 84–85
- for loop, 81–82
- looping over collections, 82–84
- recursive functions, 88–90
- ResultSet class iteration, 285–286
- template loop expressions, 254–256
- Templo loop expressions, 263–264
- while loops, 80–81

**Loops, output string data to stream, 313****lpad, 69–70****Lsys project, 147****ltrim, 69–70****M****macros**

- calling inside template, 256–257
- with Templo, 260, 266–268

**magic, 475–478**

- boot classes, 475
- Flash, 477–478
- Infos interface, 477
- JavaScript, 478
- Neko, 477
- PosInfo, 477
- Public interface, 477
- resolve method, 475–477
- setfield method, 475–477
- static initialization, 475

**main function**

- execution at startup, 85, 104
- HaXe program structure, 23
- search, storing location of, 68

**-main switch**

- functions of, 19, 24
- importance of, 19

**make method, Manager class, extending, 295****Manager class**

- extending class, steps in, 294–295
- methods/description, 292–293
- Neko, 288, 291–295
- Neko Media Engine (NME), 534–537
- nGui library, 482–484

**map<X>, lists, 47****mapi method, lambda class, 97–98****map method**

- Lambda class, 97–98
- lists, 94–95

**maps, HTML, 584****mask**

- Flash movie, 342–344
- Neko Media Engine (NME), 540

**matched method, regular expressions, 211–213, 214–215, 218–219**

**math, 63–66**

- divide by zero issue, 64–65
- float to int conversion functions, 65–66
- math fields/descriptions, 63–64
- numerical validity, testing for, 65
- template numeric/calculation expressions, 253

**menus**

- creating, nGui, 496–499
- creating, SysTools, 526–529
- submenus, 497, 528

**message boxes**

- confirm box, 530
- SysTools, 529–530

**message hooks, custom events with, 516–517**

**methods, abstract methods, 119–120**

**MochiKit library, 400**

**mod\_neko**

- installing on Linux Ubuntu, 233–234
- installing on Windows, 232–233

**modulo operator (%), 55, 57**

- in Templo, 261

**mootools library, 400**

**Motion-Twin ActionScript Compiler (MTASC), 6**

**mouse events, Flash user interface, 371–372**

**movie clips. See Flash movie; SWFMILL**

**mtwin project, 147**

**multidimensional arrays, 45–46**

- Arrays of Arrays, 46

**multimedia**

- Flash movies, 323–324
- with Neko. See Neko Media Engine (NME)

**multiplication (\*), binary operator, 55**

**MySQL database, connecting to, 274**

## N

**name qualifier, 353**

**negation operator (!), 55**

- if statement, 75

**Neko**

- applications, running, 20
- basic requirements, 9
- builtins, 477
- bytecode, 156–157
- and C/C++. See Neko and C/C++
- compiling to, 7, 19–21
- databases, working with. See Neko database support
- desktop applications. See Neko desktop applications
- downloading, 15
- executable files, creating, 20
- external libraries, 156–158
- Flash-Neko communication, 517–524
- front controller, web development, 238–239

functions of, 271–272

installation of, 14–17

integers, bits allowed, 32–33

JavaScript-to-Neko HTTP server, 433

multimedia with. See Neko Media Engine (NME)

ndll library files, 555–556

NekoML, 9

Neko script, standard, 9

NekoSocketConnection, 436–437

NekoTools, 20, 480

neko.vm.Loader, 157

neko.Web class, 236–238

nGui library, 481–498

NXML, 9

Page Controller, web development, 235–236

Remoting API communication methods, 428

Screen Weaver HX application, 511–513

source files, requirements for compiling, 16

standard libraries, 272–273

switches, 24

Templo system, 258–268

toolkit, 20

trace output, 180

Unicode support, 221–222

virtual machine, 272

web server. See NekoTools web server

Web site, 9, 15

**neko\_error function, 574**

**Neko and C/C++**

abstract values, 563–565

arrays, 567–568

C/C++ values, passing to Neko, 562–563

C foreign function interface (C FFI), 555

data restructuring, 560

error handling, 573–574

functions, Neko, executing in C/C++, 569–570

functions, passing from C/C++, 569

functions, passing from haXe, 568–569

garbage collection, 571–573

haXe, data transit from, 559–560

Hello World, 556–559

Neko library files, 556

Neko primitives, 570–571

objects, creating/modifying, 566–567

objects, passing from Neko, 565–566

type checking, 560–561

value struct, 558–559

**Neko database support, 273–314**

choosing database, 273–275

Connection class, 277–278

database type, determining, 281

data from executed query, container for, 283–286

data integrity, insuring, 281–282

dbName method, 281

deleting records, 280–281

directory management, 303–304

fields, access by index, 284

- fields, access by name, 283–284
- File class, 304–311
- FileSystem class, 301–302
- inserting records, 281
- iteration, 285–286
- lastInsertID method, 281
- length property, 286
- Logger class, 311–314
- Manager class, 291–295
- of MySQL database, 274
- nfields property, 284–285
- object to relational mapping, 286
- ODBC driver, 277
- path strings, 302–303
- populating database, 279–281
- of PostgreSQL database, 274
- read data directly, 307
- read data from stream, 305–307
- remote files, accessing, 311
- request method, 278
- ResultSet class, 283–286
- SPOD system. *See* SPOD (Simple Persistent Objects Database) system
- SQLite database, 274–275
- tables, creating, 278–279
- transactions, 281–282
- traversing stream, 309–310
- updating records, 280
- write data to stream, 307–309
- Neko desktop applications, 479–506**
  - benefits to use, 479–480
  - Button controls, 488–489
  - container classes, 487–488
  - Control class, 485–487
  - GUI libraries, 480–481
  - GUI library installation, 481
  - Label control, 484–485
  - List control, 491–495
  - Manager class, 482–484
  - menus, 496–499
  - nGui library controls, 482
  - static controls, 484–485
  - submenus, 497
  - Tab control, 495–496
  - Text control, 489–491
- Neko Media Engine (NME), 532–553**
  - animation, 549
  - collisions, detecting, 543–545
  - color key, setting, 540
  - display buffer, flipping, 535–536
  - events, capturing, 549–553
  - images, file formats for, 538
  - Manager class, 534–537
  - refresh rate, slowing, 536–537
  - and SDL library, 532–533
  - sprites, 546–549
  - surface, drawing to display, 538–540
  - Surface class, 537–545
  - surface transforms, 540–543
  - text, adding, 537
  - timers, 546
- Neko.ndlls library, 158**
- neko package, 146**
- NekoTools web server, 231–234**
  - deploying website to, 20–21
  - mod\_neko, installation of, 232–234
  - parameters, listing of, 231
  - root location, changing, 21, 232
  - TicTacToe Game, 443–452
- Neko Virtual Machine, 480, 509**
- neko.Web class, 236–238**
  - static methods, 237–238
  - static variables, 238
- nesting, if statement, 75–76**
- newlines, escaped characters, 33–34**
- next( ), 81**
- nfields property, database, 284–285**
- nGame project, 147**
- nGui library. *See* Neko desktop applications**
- nMagick project, 147**
- nme project, 147**
- nodes, XML**
  - checking, 458–461
  - creating, 206
  - traversing, 456–457
- no-traces compiler directive, 185**
- nPostgres project, 147**
- null, variable value modifier, 111**
- Null<T>, 345**
- null arguments, instance functions, 105–106**
- null type, functions of, 35**
- numbers, floating-point, 32**
- numeric expressions, 253**

## O

- object(s), anonymous objects, 126–127**
- object initializer, 353**
- Objective Caml, downloading, 13**
- object-oriented programming**
  - anonymous objects, 126–127
  - classes, 102–103, 122–126
  - constructor arguments, 136–138
  - dynamic type, 127–133
  - enum, 134–136
  - extensions, 133–134
  - with haXe, 27, 74
  - inheritance, 115–121
  - instance fields, 104–114
  - interfaces, 121–122
  - polymorphism, 118
  - static fields, 114–115

- object types, Neko and C/C++, 565–567**
- ODBC driver, Neko database connection, 277**
- onClose events, 515**
- onFilesDropped events, 516**
- online documentation, 169–170**
- onMinimize/onMaximize events, 515**
- onRightClick events, 516**
- opengl project, 147**
- operators, 55–63**
  - ActionScript/haXe comparison, 352–353
  - assignment operators, 59–60
  - binary operators, 56–58
  - bitwise operators, 60–63
  - listing of, 55–56
  - logical comparison operators, 58–59
- optional arguments, 106–107**
  - placement of, 106
- OR ( | ), bitwise operator, 62**
- OR ( || ), logical comparison operator, 56, 59**

## P

- packages, 139–145**
  - declaring package, 140–141
  - and enum, 143
  - explicit import, 142–143
  - implicit import, 141–142
  - importing package, 144–145
  - type lookup sequence, 144
- Page Controller, Neko as, 235–236**
- PanelLayout, Flash user interface, 376–378**
- parseFloat, type data conversion, 39**
- parseInt, type data conversion, 39**
- pascal4neko project, 147**
- passwords, and cryptographic hashes, 224**
- path projectname project, 148**
- path strings, Neko, 302–303**
- patterns, regular expressions, 214–222**
- Person class, 438**
- PersonDatabase class, 438–440**
- polymorphism, 118**
- pop, arrays, 42, 43**
- PosInfo, magic, 477**
- PostgreSQL database, connecting to, 274**
- POST method, web server, 227–228**
- primitives, Neko and C/C++, 570–571**
- PRIVATE\_FIELDS, 290**
- private classes, 102**
- properties, ActionScript/haXe comparison, 356**
- Prototype library, 400**
- proxy**
  - class, remotng, 437–442
  - hxGtk function proxy, 499–501
  - XML Proxy, 463–464
- public classes, 102**
- Public interface, magic, 477**

- publish, blog entry, 109–110**
- pull technology, remote communication, 426**
- push**
  - arrays, 42, 43
  - lists, 47
- push technology, remote communication, 426–427**

## Q

- quantifiers**
  - regular expressions, 218–219
  - syntax for, 218
- queued actions, timer, 223**

## R

- raw expression, Templo, 262–263**
- RChoice, 460**
- RData, 460–461**
- readAll method, 306–307**
- readBytes method, 306**
- readChar method, 305–306**
- readDirectory method, Neko, 303–304**
- read-only access, variable value modifier, 111–112**
- recursive functions, 88–90**
  - functions of, 88
  - multifunction recursion, 92–93
  - operation of, 89–90
- Reflect class, 122–123**
- reflection API, 464–469**
  - Reflect class, 464–466
  - Type class, 467–469
- refresh rate, slowing, 536–537**
- register project, 149**
- regular expressions, 210–222**
  - alternation, 220
  - anchors, 217–218
  - back-references, 214, 220
  - character classes, 215–217
  - characters, 214–215
  - character sets, 217
  - conditionals and comments, 220–221
  - defined, 210
  - EReg class, 210–214
  - groups, 219
  - look-around constructs, 220–221
  - matched method, 211–213, 214–215, 218–219
  - optional modifiers, 211–214
  - patterns, 214–222
  - quantifiers, 218–219
  - replace method, 213–214
  - split method, 213
  - Unicode support, 221–222
- RELATIONS method, SPOD system, 297–298**

**remoting, 425–452**

- AsyncAdapter, 437
- asyncConnection, creating, 432–434
- AsyncDebugConnection, 437
- connection, creating, 428–429
- DelayedConnection, 437
- FlashJsConnection, 435
- Flash-Neko communication, 517–524
- Flash-to-Action Message Format (AMF) Server connection, 434
- Flash-to-JavaScript connection, 429–431
- functions of, 426
- JavaScript-to-Flash connection, 432
- JavaScript-to-Neko HTTP server, 433
- LocalConnection, 435
- NekoSocketConnection, 436–437
- proxy class, 437–442
- push or pull type communication, 426–427
- remote files, accessing, Neko, 311
- Remoting API communication methods, 427–428
- server side requirements, 426
- SocketConnection, 436
- synchronous versus asynchronous communication, 426
- TicTacToe Game, 443–452

**remove**

- arrays, 42, 44
- hash method, 49
- lists, 47

**remove projectname project, 149****replace**

- characters in string, 71
- regular expressions method, 213–214
- StringTools method, 69

**request method, Neko database support, 278****reserved characters, regular expressions, 215****—resolve method, magic, 475–477****resources, 163–166**

- embedded XML files, use of, 163
- resourcefile path, 163
- templates, 257–258

**ResultSet class, 283–286****return keyword, end of function, 86–87****right shift (>>), binary operator, 55****RList, 459–460****RMulti, 459****RNode, 459****ROptional, 460****round, float to int conversion, 66****rpad, 69–70****rtrim, 69–70****rules, HTML forms, data validation, 404–414****run projectname project, 149****Runtime Type Information (RTTI), 470–473**

- accessing with haXe, 471
- function of, 470
- Inspect method, 471–473
- XML RTTI documents processing, 470–473

**S****Screen Weaver HX, 510–524**

- ActionScript extensions, 523–524
- benefits to use, 510, 513
- downloading, 511
- with Flash. See Flash and Screen Weaver HX
- Flash-Neko communication, 517–524
- installing, 511
- library, 480
- menus, creating, 526–529
- message boxes, creating, 529–530
- Neko application, example of, 511–513
- SWF2Exe applications, 510
- synchronous communication system, 520–524
- system tray icon, creating, 526
- SysTools library, 525–530
- window events, 515–516
- window flags, 513
- window properties, 514

**script.aculo.us library, 400****scripts, HTML, 587****search**

- storing location of, 68
- strings, 67

**search keyword project, 149****SeekBegin method, 310****SeekCur method, 310****SeekEnd method, 310****seek method, 309–310****serialization, 473–474**

- defined, 473
- serialization API, 473–474
- serialize/deserialize process, 474

**set, hash method, 49–50****setColor function, trace, 184****set expression, Templo, 264–265****—setfield method, magic, 475–477****setParser function, 401–403****set projectname version project, 149****setter method, variable value modifier, 111****setup project, 149****shifting bits, 61–62****simple value types, 31–34**

- Booleans, 33
- floating-point numbers, 32
- strings, 33–34

**single inheritance, 115****single quotes, escaped characters, 34****slice, arrays, 42, 44–45****slideshow, Flash, 331–337****socket connections**

- NekoSocketConnection, 436–437
- SocketConnection, 436

**sort method, arrays, 96****sounds, Flash movie, 325****spaces, in strings, removing, 70**

**special values, ActionScript/haXe comparison, 346–347**

**splice, arrays, 42, 44**

**split, 66–67**

**split method, regular expressions, 213**

**SPOD (Simple Persistent Objects Database) system, 286–301**

Author class, 287–288, 296–297, 299

AuthorManager class, 300–301

Chapter class, 296–297, 299–300

ChapterManager class, 300–301

deleting records, 291

functions of, 286

global cache, 295

inserting data, 291

RELATIONS method, 297–298

synchronize data, 291

table mapping, 289–291

table relations, 295–301

updating records, 291

**sprites**

animating, 548

Neko Media Engine (NME), 546–549

sprite sheet, 547–548

**SQLite database**

connecting to, 274–275

hxGtk library, use of, 499–500

pros/cons of, 275

**StackLayout, Flash user interface, 378–379**

**stage, Flash movie, 316–317**

**standard data types**

haXe types, 28

simple value types, 31–34

**startsWith, 70**

**statements, ActionScript/haXe comparison, 354–358**

**static controls, nGui, 484–485**

**static fields, 114–115**

functions of, 114

and inheritance, 120

static functions, 114–115

static variables, 114–115

**static fromCharCode, 66**

**static fromString, date object, 51**

**static functions, XML, 206**

**static initialization, magic, 475**

**static links, Flash, 152–154**

**static methods, neko.Web class, 237–238**

**static now, date object, 51**

**static variables, 103, 114–115**

declaring, syntax for, 114

neko.Web class, 238

values, assigning, 115

**statis fromTime, date object, 51**

**streams**

output string data to, 313

read data from, 305–307

stream for output, accepting, 311–312

traversing, 309–310

write data to, 307–309

**string(s), 66–72**

add characters to end of, 70

arrays converted to, 68

characters, extracting from, 67

constructing from smaller strings/characters, 71–72

converting to arrays, 67

delimiters, 67

encoding for web, 71

escaped characters, 33–34

fields/descriptions, 66

functions of, 33

length, determining, 67

replace characters, 71

searching strings, 67

starting/ending characters, verifying, 70

StringBuf method, 71–72

string literals, 33–34

StringTools methods, 69–70

unwanted spaces in, 70

verify start and end of, 70

**string, standard data type, 28**

**string, type data conversion, 39**

**structural elements, HTML, 578–581**

**structures, ActionScript/haXe comparison, 354–358**

**style sheets, HTML, 586–587**

**submenus**

nGui, 497

Screen Weaver HX, 528

**submit file.zip project, 149**

**substr, 66–67**

**subtraction (-), binary operator, 55**

**super identifiers, 117–118**

**Surface class, Neko Media Engine (NME), 537–545**

**SWFMill, 364–369**

file formats supported, 365

functions of, 364

movie clip, creating, 365

movie clip use with haXe, 365–368

parameters/description, 364

transition between frames, 367–368

True Type Fonts (TTF), embedding, 368–369

**SWF Studio, 510**

**swhx project, 147**

**switch(es)**

all platforms, 24–25

Flash, 24

JavaScript, 24

-main switch, importance of, 19

Neko, 24

**switch statement, 77–78**

and enum, 135, 143

values, returning from conditional statements, 78–79

**synchronous communications, 426**

- in ActionScript, 523–524
- in haXe, 520–523
- Screen Weaver HX system, 520–524

**SysTools**

- features, listing of, 525
- menus, creating, 526–529
- message boxes, 529–530
- Screen Weaver HX, 525–530
- tray icon, creating, 526

**systools project, 147****T****Tab control, nGui, 495–496****table(s)**

- database, creating, 278–279
- HTML, 582–583
- many-to-many relations, creating, 298–301
- mapping, SPOD system, 289–291, 296
- relations, adding, SPOD system, 295–301
- sorting, HTML, 401–404

**TABLE\_IDS, 290****TABLE\_NAME, 290****tabs, escaped characters, 33–34****tags, HTML, 226–227****TDD (Test-Driven Development), 172–177****tell method, 309****templates, 251–268**

- class methods/description, 252
- code, situations for use, 258
- comparison expressions, 253
- dot expressions, 253–254
- file extensions, 260
- functions of, 251–252
- importing into templates, 256
- logical expressions, 254
- loop expressions, 254–256
- macros, use of, 256–257
- manual compilation of, 268
- numeric/calculation expressions, 253
- resources, 257–258
- syntax for, 253–256
- Templo, template engine, 258–268

**Templo, 258–268**

- attr expressions, 262
- features of, 258–259
- file extensions, 260
- fill expression, 265
- haXe.Template compared to mtwin.Templo, 261
- installing, 259
- logical expressions, 263
- loop expressions, 263–264
- macros, 260, 266–268
- raw expression, 262–263

- set expression, 264–265
- Template system, 259–260
- use expression, 265–266

**TestCase, unit testing, 171–172****test file.zip project, 149****text**

- box, Flash user interface, 376
- editor, use with haXe, 17
- in Flash movie, 322–323
- Flash user interface, 375
- nGui Text control, 489–491
- True Type Fonts, embedding in SWFMill movie clips, 368–369
- writing, Neko Media Engine (NME), 537

**TextField class, Flash, 322–323****this keyword, current object, calling, 104****throwing exceptions, 188****TicTacToe Game, 443–452****time. See date/time; timer****timeline, Flash movie, 316–317****timer, 222–223**

- action, delayed, 223
- functions of, 222
- Neko Media Engine (NME), 546
- query time, 546
- queued actions, 223
- timer instance, creating, 222

**token, date formats, 54****toLowerCase, 66****tools package, 146****top-level domain (TLD), 227****toString**

- arrays, 42
- date object, 51
- hash method, 49
- inheritance, 119
- lists, 47
- string, 66, 71–72

**toUpperCase, 66****trace, 179–185**

- clear function, 184
- Flash output, 180
- haXe compared to ActionScript, 181
- haXe.Log class, 181–185
- haXe.Log.trace, 181–182
- haXe.PosInfos, 184
- here identifier, 182–184
- JavaScript output, 180
- Neko output, 180
- no-traces compiler directive, 185
- redirect to Firebug, 389–390
- setColor function, 184

**transactions, database, 281–282****transforms, surface, Neko Media Engine, 540–543****tray icon, creating with SysTools, 526****trim, 70**

**true, and if statement, 75–76**

**True Type Fonts (TTF), embedding in SWFMill**

movie clips, 368–369

**try...catch block, 304–305**

**twister project, 148**

**type(s), lookup sequence, 144**

**type checking, Neko and C/C++, 560–561**

**Type class, 122–123**

reflection API, 467–469

**typedef, 128–131**

with anonymous objects, 128–130

benefits to use, 129–130

declaring, syntax for, 128

extensions, use of, 134

external structure mapping, 150

HTML DOM declarations, 394–396

Iterator and Iterable, 131–133

keyword, 28

**type operator, 353**

**type parameters, 123–126**

constraints on, 125–126

at function level, 124–125

situations for use, 123–124

**typing**

dynamic typing, 29

type inference, 30–31

untyped block, 36–37

## U

**Unicode support**

characters, embedding in source code, 222

scope of, 221–222

**unit testing, 170–177**

haxe.unit package, 171–177

purpose of, 170–171

TDD (Test-Driven Development), 172–177

TestCase method, 171–172

**unknown keyword, 28**

**unmake method, Manager class, extending, 295**

**Unobtrusive Flash Objects (UFO), Flash content,**

embedding with, 319

**unpublish, blog entry, 109–110**

**unsafe cast, 38–39**

**Unserializer class, 474**

**unshift, arrays, 42, 43**

**untyped block, 36–37**

**upgrade project, 149**

**URI (Unified Resource Identifier)**

URL rewritten to, 231

web server, 226, 228

**urlDecode, 70**

**urlEncode, 70, 71**

**use expression, Templo, 265–266**

**user interface. See graphical user interface (GUI) development**

**user username project, 149**

## V

**val\_print function, 574**

**val\_throw and val\_rethrow function, 573**

**value(s)**

arrays, adding/removing, 43–44

C/C++ values, passing to Neko, 562–563

comparing types of, 40

Flash movie, loading, 325

lists, adding/removing, 47

querying in lists, 48

returned by functions, 87

returning from conditional statements, 78–79

variable modifiers of, 110–114

**value struct, 558–559**

**variables, 28–31**

of classes, 102–103

constant variables, 31

declarations, 29–30

dynamic typing, 29

Flash movie, loading, 326–327

identifiers sequence, 107

initializing, 29

instance fields, 107

local functions, 90–91, 93–94

modified, syntax for, 110

scope, local functions, 93–94

static variables, 103, 114–115, 238

type inference, 30–31

value modifiers, 110–114

var keyword, 29

Window class, 514

XML, 208

**var keyword, 29**

**vector graphics, Flash drawing API, 337–342**

**VectorSpace project, 148**

**virtual machine, Neko, 272**

**void**

functions of, 35

standard data type, 28, 87

## W

**Web Developer extension, FireFox, 389**

**web development, 225–250**

Apache web server, 232–234

behavior layer problems, 388–389

blog entries, 108–110

Firefox extensions, 389

GUI. See graphical user interface (GUI) development

haXe compared to HTML, 230–231

haxe.Http class, 415

HTML, 228–231

movie clips. See Flash; Flash movies

remote communication. See remoting

Screen Weaver HX, 510–524

site domains, levels of, 227  
 strings, encoding for web, 71  
 web page, layers of, 387–388  
 web server, 226–228  
 Wiki site development example, 238–250  
 See also Java Script; Neko

#### **web server, 226–228**

accessing, addresses for, 227  
 content distribution with, 226–228  
 functions of, 226  
 GET method, 227–228  
 information flow in, 226  
 NekoTools. See NekoTools web server  
 POST method, 227–228  
 Remoting API communication methods, 427–428  
 templates, Templo, 258–268  
 URI addresses, 226, 228

#### **websites, development of. See web development**

#### **while loops, 80–81**

break and continue with, 84–85  
 while and do while, 80–81

#### **Wiki, site development, example of, 238–239**

#### **Windows**

Apache, installation of, 233–234  
 HaXe installation on, 12–13  
 Neko installation on, 14–16  
 Screen Weaver HX functionality, 513–517  
 Unicode support, 221–222

#### **writeBytes method, 308–309**

## **X**

#### **xcross project, 148**

#### **Xinf, 480**

#### **xinf project, 148**

#### **XML, 455–464**

attributes, accessing, 457–458  
 attributes, checking, 461–462

class instance functions, 207  
 class static functions, 206  
 class variables, 208  
 config information, storing, 163–165  
 document creation, 209–210  
 documents, structure of, 203–206  
 Fast class, 455–456  
 filter rules, 462–463  
 Flash movie, loading, 327–328  
 haXe API, 206–208  
 nodes, checking, 458–461  
 nodes, creating, 206  
 nodes, traversing, 455–457  
 online documentation, 169–170  
 Proxy class, 463–464  
 RChoice, 460  
 RData, 460–461  
 RList, 459–460  
 RMulti, 459  
 RNode, 459  
 ROptional, 460  
 SWFMill movie clips, 364–368  
 traversing documents, 208–209  
 validation rule, 458  
 XML RTTI documents processing, 470–473

#### **XOR (^), bitwise operator, 56, 62**

#### **xpath project, 148**

## **Y**

#### **Yahoo! User Interface Library, 400**

## **Z**

#### **zero, divide by zero issue, 64–65**

#### **Zinc, 510**