

Index

Symbols

- \$ (dollar sign)**, for specifying position in pattern matching, 549
- \$_COOKIE**, superglobal array, 277
- \$_FILES**, superglobal array, 257–258
- \$_GET**, superglobal array, 230
- \$_POST**, superglobal array, 230
- \$_REQUEST**, superglobal array, 230
- \$_SERVER**, superglobal array, 494–496
- \$_SESSION[]**, superglobal array, 283, 289
- % (modulus)**, 35
- & (AND)**
 - PHP bitwise operator, 42
 - setting configuration directive values with, 720
- && (and)**, PHP logical operators, 46
- [] (square brackets)**
 - adding array elements and, 104–105
 - handling multi-value fields in HTML forms, 236
- ^ (caret symbol)**, specifying position in pattern matching, 549
- ^ (XOR)**, PHP bitwise operator, 43
- { } (curly brackets)**
 - coding standards for, 630
 - for including complex expressions in strings, 75–76
 - for writing functions, 145
- | (OR)**, PHP bitwise operator, 42
- | (vertical bar)**, matching alternative patterns and, 548
- || (or)**, PHP logical operator, 46
- ~ (NOT)**
 - PHP bitwise operator, 43
 - setting configuration directive values with, 720
- +** (addition), PHP arithmetic operator, 41
- <** (less than), PHP comparison operator, 44
- <<** (Shift left), PHP bitwise operator, 43
- <=** (less than or equal to), PHP comparison operator, 44
- <=>** (NULL-safe version of equal to), MySQL comparison operator, 383–384
- =** (equals), PHP assignment operator, 42
- !=** (not equal), PHP comparison operator, 44
- ==** (equal), PHP comparison operator, 44
- !==** (not identical), PHP comparison operator, 44
- ===** (identical), PHP comparison operator, 44
- >** (greater than), PHP comparison operator, 44
- >=** (greater than or equal to), PHP comparison operator, 44
- >>** (Shift right), PHP bitwise operator, 43
- \ (back slash)**, for escaping special characters, 542–543
- .** (concatenation operator), 46–47
- /** (division), PHP arithmetic operator, 35
- "** (double quotation marks), in string syntax, 74
- !** (exclamation sign), specifying position in pattern matching, 549
- !** (not), PHP logical operator, 46
- ! (NOT)**, setting configuration directive values with, 720
- '** (single quotation marks), in string syntax, 74–75
- /** (slash), in regular expression syntax, 543
- (subtraction), PHP arithmetic operator, 35

A

- abstract classes, OOP, 200–204
- abstract methods, OOP, 200–204
- Access (Microsoft), 763
- Active Server Pages (ASP), 5
- addition (+), PHP arithmetic operator, 41
- additive color model, 508
- aliases, for simplifying queries, 381–382
- `allow_url_fopen`, configuration directive, 737
- `allow_url_include`, configuration directive, 737
- Alternative PHP Cache (APC), 9
- anchors, for pattern matching at specified positions, 548–550

AND (&)

- PHP bitwise operator, 42
- setting configuration directive values with, 720

and (&), PHP logical operators, 46

AND, **Boolean operators, 384**

and, **PHP logical operator, 46**

anonymous functions

- creating, 154–155
- example for sorting words by length, 155–158

Apache Web Server

- .htaccess file, 719
- installing PHP on Mac OS X, 17–18
- installing PHP on Ubuntu, 12, 14
- installing PHP on Windows, 15–16
- testing, 19–20

APC (Alternative PHP Cache), 9

API (application programming interface), 763

`appendChild()`, **moving elements in XML documents, 606**

application logic, separating from presentation logic, 660–661

application programming interface (API), 763

application quality. See high-quality applications

application security. See also security

- checking input and, 641–643
- encoding output and, 641, 643–644

architectures, database, 338–339

arcs, drawing, 515–516

arguments

- functions accepting, 141
- hints used to check method arguments, 180–182
- parameters compared with, 146
- passing to scripts from command-line, 767–768

arithmetic operators, 41

`array()`, **102–103**

`array_merge()`, **135–136**

`array_multisort()`, **124–128**

`array_pop()`, **129–130**

`array_push()`, **128–129**

`array_shift()`, **128–130**

`array_slice()`, **107–108**

`array_splice()`, **129–134**

`array_unshift()`, **128–130**

arrays

- accessing elements of, 103
- adding/removing elements at start and end, 129–130
- adding/removing elements generally, 128–129

- adding/removing elements in middle, 130–134
- anatomy of, 102
- changing elements of, 104–105
- converting to a list of variables, 137–138
- converting to/from strings, 136–137
- counting elements in, 108–109
- creating, 102–103
- exercise solutions, 678–683
- exercises, 139
- extracting range of elements from, 107–108
- `foreach()` for altering values, 115–116
- `foreach()` for looping through keys and values, 114–115
- `foreach()` for looping through values, 114
- looping through, 113–114
- manipulating, 121
- merging, 134–136
- multidimensional. *See* multidimensional arrays
- multisorting, 124–128
- options for sorting, 121
- outputting entire array with `print_r()`, 105–107
- overview of, 101–102
- searching arrays with `preg_grep()`, 556
- sorting associative array keys with `ksort()` and `krsort()`, 123
- sorting associative arrays with `asort()` and `arsort()`, 122–123
- sorting indexed arrays with `sort()` and `rsort()`, 121–122
- stepping through, 109–113
- summary, 138

`arsort()`, **sorting associative arrays, 122–123**

`asort()`, **sorting associative arrays, 122–123**

ASP (Active Server Pages), 5

ASP.NET, 5

assertions, 548

assignment operators, 41–42

associative arrays

- overview of, 102
- sorting keys with `ksort()` and `krsort()`, 123
- sorting with `asort()` and `arsort()`, 122–123

Atom feeds, 573

attributes, HTML forms, 222

attributes, XML

- changing in XML documents, 603–605
- overview of, 578
- use in XML documents, 574

`authenticate()`, **418**

authentication

- members' area of Web site and, **418**
- not using query strings for, **268**

`auto_detect_line_endings`, **configuration directive, 737**

B

back slash (\), for escaping special characters, 542–543

backreferences, pattern matching and, 547–548

`basename()`, **299**

binary, working with files in binary mode, 301–302

BINARY attribute, collations and, 369

bitwise operators

- overview of, **42–43**
- setting configuration directive values with, **720**

blacklisting, checking input and, 642–643

break statements, for escaping loops, 64

browsers, script for detecting visitor's browser, 448–449

buttons, on HTML forms, 226–227

by reference, passing arguments, 159. See also references

by value, passing arguments, 158

C

C compiler, for compiling PHP, 23–24

`_call()`

- example creating wrapper string class, **188–191**
- overloading method calls with, **187**

callback function, for replacing text, 560

calling functions, 142–144

calling methods, 175

`_callStatic()`, **overloading method, 192**

caret symbol (^), specifying position in pattern matching, 549

case-sensitivity

- string functions and, **88–89**
- variable names and, **34**

casting, changing type by, 38–40

catching exceptions, 654

CHAR data type, MySQL, 346–347

character classes, for pattern matching, 544–545

character data types, 369

characters

- accessing within strings, **78**
- allowable in query strings, **269**
- matching literal characters, **542–543**
- matching multiple characters, **545–546**
- matching using character classes, **544–545**
- reading/writing strings of, **304–305**
- `strbrk()` for finding set of, **81**
- `strtr()` for translating, **87**

chdir(), for changing current directory, 319–320

checkboxes, on HTML forms, 226

`checkdate()`, **for checking date values, 481**

checking input, application security and, 641–643

child classes

- example creating, **193–196**
- overview of, **192**

child elements, XML, 578

`chmod()`, **for changing file permissions, 314–315**

circles, drawing, 515

class constants, 173–174

classes

- abstract, **200–204**
- automatically loading class files, **212–213**
- creating, **168–169**
- creating exception classes, **655**
- for dates and times, **487–488**
- determining class of an object, **215–216**
- encapsulation and, **182–183**
- example creating parent and child classes, **193–196**
- example creating wrapper string class, **188–191**
- final classes blocking inheritance and overriding, **199–200**
- modular code and, **620**
- overriding methods in parent class, **196–198**
- overview of, **167**
- preserving functionality of parent class, **198–199**

classes, member record viewer

- `DataObject` class, **388–390**
- `LogEntry` class, **394–395**
- `Member` class, **390–394**

client-server databases, 339

`closedir()`, **for closing directories, 317**

closing files, 303

code

- automating code testing, 666–667
- avoiding duplication in, 142
- examining editor code, 330
- standards for class names, 168
- standards for consistency, 630–631
- code blocks, indentation conventions, 630**
- code reuse. See also PEAR (PHP Extension and Application Repository)**
 - modular code and, 620
 - using PEAR for, 441
- ColdFusion, 6**
- collations, character data types and, 369**
- colors**
 - allocating when creating images, 510–511
 - color theory, 508
- columns**
 - ENUM data type and, 370–371
 - in relational databases, 341
- combined assignment operators, 42**
- command-line mode, PHP**
 - creating interactive scripts, 768–770
 - options, 772–773
 - overview of, 4, 765–766
 - passing arguments to scripts, 767–768
 - running scripts, 766–767
 - scheduling scripts, 770–772
- command-line tool, MySQL, 349**
- comma-separated-value (CSV) files, 309–310**
- comments**
 - single-line and multi-line, 29–30, 632
 - writing good, 632–633
- `common.inc.php`
 - member record viewer and, 387
 - member registration application and, 408–409
 - members' area of Web site and, 420–421
- comparison operators**
 - MySQL, 381–382
 - PHP, 43–44
- compiling PHP, vs. precompiled, 23–24**
- complex expressions, including within strings, 75–76**
- components, of cookies, 274–275**
- compound data types, 36**
- compressed formats, images, 509**
- computer graphics, 507**
- concatenation operator (.), 46–47**
- `config.php` file, for member record viewer, 386–387
- configuration directives**
 - for data handling, 731
 - for error handling, 647–648, 727
 - for extensions, 737
 - for file upload, 736
 - for `fopen()`, 736–737
 - methods for setting, 719–720
 - for path, 734
 - for sessions, 291
- configuring PHP. See PHP configuration**
- `const` keyword, 173–174
- constants**
 - class constants, 173–174
 - defined, 33
- constructors, for objects, 209–210**
- contact form script, 501–505**
- continue statement, for skipping loop iterations, 64–65**
- control flow structures, 631**
- controls, HTML forms**
 - creating, 225
 - for handling empty fields, 235–236
- conversion specifications, format strings, 89**
- cookies**
 - accessing in scripts, 277
 - components of, 274–275
 - example for remembering user information, 278–282
 - overview of, 274
 - removing, 278
 - setting, 276
- coordinate systems, for drawing shapes, 508–509**
- `copy()`, 316
- copying files, 316**
- `count()`
 - counting elements in an array, 108–109
 - summarizing query data, 376–377
- CPU, setting resource limits, 727**
- CREATE DATABASE command, 353**
- CREATE TABLE command, 354**
- `create_function()`, for anonymous functions, 155, 157
- `createFile()`, 333–334
- cross-platform support**
 - reasons for using PHP, 5
 - SQLite features, 758
- cross-site scripting attacks (XSS), 642**
- CSS, for stylish pages, 25–26**
- CSV (comma-separated-value) files, 309–310**
 - `curdate()`, 384

curly brackets ({}). See **{}** (curly brackets)

`current()`, **110**

D

data

- accessing in query strings, 270–274
- adding to MySQL tables, 355
- arrays for storing, 101
- deciding how to store, 338
- deleting from MySQL tables, 358
- manipulating with MySQL. See MySQL, data manipulation
- PHP configuration for handling, 731–734
- random access to file data, 312
- reading data in MySQL tables, 356–357, 361–364
- reading/writing session data, 283
- retrieving with MySQL. See MySQL, data retrieval
- updating data in MySQL tables, 357

data, form

- capturing, 230
- example writing form handler, 231–234
- `get` and `post` methods for sending to server, 229
- secure handling of, 234

data handling, PHP configuration and, 731–734

data storage, arrays for, 101

data types

- changing type by casting, 38–40
- changing variable type, 38
- defined, 33
- loose typing, 36
- overview of, 35–36
- testing variable type, 36–38

data types, MySQL

- date and time, 345
- numeric, 344–345
- overview of, 344
- string, 346–347

Database Management System (DBMS), 338.

See also **databases**

Database Source Names (DSN), 360

databases

- architectures of, 338–339
- choosing, 340
- dbm-style, 761
- deciding how to store data, 338
- exercise solutions, 698–699
- exercises, 366

- list of other databases supported by PHP, 764
- models, 339–340
- MySQL. See MySQL
- ODBC, 763–764
- Oracle, 762
- overview of, 337
- PostgreSQL, 759–761
- relational. See RDBMS (Relational Database Management Systems)
- setting up sample, 367–369
- SQLite, 757–759
- summary, 365–366

DataObject class, member record viewer, 388–390

date()

- formatting date strings, 478–479
- formatting time characters, 479–480
- member registration application and, 416

date strings, formatting, 478–481

dates and times

- checking date values, 481
- classes for, 487–488
- creating timestamps, 473–475
- example calculating age in days, 483–487
- extracting date and time values from timestamps, 475–477
- formatting date strings, 478–481
- getting current date and time, 472
- microseconds and, 481–483
- MySQL data types for, 345
- overview of, 472
- timestamps and, 472

DateTime class, 487–488

DateTimeZone class, 487–488

DB2 database, 763

DBA abstraction layer, supporting dbm-style databases, 761

DBMS (Database Management System), 338.

See also **databases**

dbm-style, 761

decisions. See also loops

- `else` statements for alternative choices, 54
- example displaying a greeting with, 57–58
- example homing pigeon simulator, 66–70
- exercise solutions, 674–677
- exercises, 72
- `if` statement for simple decision-making, 52–53
- overview of, 51
- statements for, 52

decisions (continued)

- summary, 72
- switch statements for testing one expression many times, 55–56
- ternary operator (?) for compact code, 56–58
- declaring properties, 170**
- declaring variables, 34–35**
- default values, functions, 147–148**
- `default_socket_timeout`, **configuration directive, 737**
- `delete()`, **adding delete methods to Member class, 428**
- DELETE statement**
 - adding deletion method to `LogEntry` class, 430
 - deleting data from MySQL tables, 358
- deleting files, 316**
- deleting records, using PHP scripts, 407**
- delimiters, string, 76–77**
- dependencies, PEAR packages an, 447**
- destructors, object, 210–211**
- directives. See configuration directives**
- directories**
 - changing current, 319–320
 - creating/deleting, 320
 - distinguishing directory files from regular files, 322
 - example of list directory entries, 317–319
 - example traversing directory hierarchy, 323–325
 - functions, 319
 - getting directory path, 321
 - overview of, 317
 - PHP configuration and, 734–736
 - resetting directory pointer, 319
 - understanding, 298
 - working with directory objects, 321–322
- Directory object, 321–322**
- `dirname()`, **for getting directory path, 321**
- `display_errors`, **configuration directive, 647, 728**
- `displayEditForm()`, **displaying file contents for editing, 332**
- `displayFileList()`, **displaying list of files for editing, 331–332**
- `displayForm()`, **displaying registration form, 415**
- `displayPageHeader()`, **text editors, 334**
- DISTINCT keyword, using with SELECT to minimize duplication in queries, 377–378**
- division (/), PHP arithmetic operator, 35**
- `dl()`, **for loading extensions, 737**

DocBlocks, phpDocumentor package, 635

- DOCTYPE declaration, 580–581**
- documenting code**
 - example using phpDocumentor, 636–640
 - overview of, 631–632
 - phpDocumentor for external documentation, 633–635
 - writing good comments, 632–633

documents, XML

- adding elements to existing document, 600–602
- changing attributes and nodes, 603–605
- creating using DOM, 595–599
- creating using SimpleXML, 610–612
- manipulating, 599
- moving elements in, 605–606
- parsing, 584
- reading using DOM, 591–594
- reading using SimpleXML, 608–610
- removing elements from existing document, 602–603
- structure of, 575–576
- valid, 578–579
- well formed, 576–577
- writing/manipulating, 589–590

dollar sign (\$) , for specifying position in pattern matching, 549

DOM (Document Object Model)

- adding elements to XML documents, 600–602
- changing nodes and attributes of XML documents, 603–605
- converting between Simple XML and DOM objects, 612
- creating XML documents, 595–599
- manipulating XML documents, 599
- moving elements in XML documents, 605–606
- overview of, 589–590
- reading XML documents, 591–594
- removing elements from XML documents, 602–603

DOM class

- methods, 595
- overview of, 590

DOMDocument(), creating DOM documents, 595

double quotation marks ("), in string syntax, 74

do...while loops, 60–61

drawing

- arcs, 515–516
- circles and ellipses, 515

- coordinate systems for drawing shapes, 508–509
 - example drawing rectangle with rounded corners, 517–520
 - individuals pixels, 512–513
 - lines, 513–514
 - polygons, 516–517
 - rectangles, 514
 - driver manager service, ODBC, 763**
 - `DROP DATABASE` **statement, deleting MySQL databases, 358–359**
 - `DROP TABLE` **statement, deleting MySQL tables, 358–359**
 - DSN (Database Source Names), 360**
 - DTDs (document type definitions)**
 - example of, 579
 - major parts of XML documents, 576
 - referencing, 580–581
 - valid XML documents and, 578–579
 - for XHTML, 579–580
 - XML document structure and, 575
 - dynamic extensions, PHP configuration and, 737–739**
 - dynamic includes, 625**
 - dynamic typing, in SQLite, 758**
 - dynamic Web pages, 3**
- ## E
- `each()`, **111–113**
 - `echo()`, **25**
 - elements, array**
 - accessing, 103
 - accessing in multidimensional arrays, 118–119
 - adding/removing, 128–129
 - adding/removing at start and end, 129–130
 - adding/removing in middle, 130–134
 - changing, 104–105
 - counting, 108–109
 - extracting range of, 107–108
 - elements, HTML_QuickForm package, 458**
 - elements, XML**
 - adding to existing document, 600–602
 - moving, 605–606
 - overview of, 578
 - removing, 602–603
 - ellipses, drawing, 515**
 - else statements, decision-making and, 54**
 - elseif statements, decision-making and, 54**
 - email, sending**
 - controlling return path of email address, 499–500
 - example creating contact form script, 501–505
 - HTML email, 500–501
 - overview of, 497–498
 - specifying sender address and adding headers, 498–499
 - embedded databases**
 - dbm, 761
 - SQLite, 757–759
 - types of database architectures, 339
 - embedding PHP, within HTML, 25–27**
 - empty fields, in HTML forms, 234–236**
 - encapsulation**
 - for making classes self-contained, 182–183
 - modular code and, 620
 - encoding output, application security and, 643–644**
 - `end()`, **for manipulating arrays elements, 110**
 - end of file, testing for, 307–308**
 - `ENUM` **data type, 370–371**
 - equal (==), PHP comparison operator, 44**
 - equals (=), PHP assignment operator, 42**
 - error element, in file upload forms, 258**
 - error handling**
 - allowing script to handle errors, 648–651
 - catching exceptions, 654
 - controlling where error messages are sent, 647
 - creating exception classes, 655
 - error levels, 644–645
 - example simulating spaceship flight, 655–660
 - exception objects for, 652–653
 - fine tuning error reporting, 651–652
 - functions for minimizing errors, 142
 - logging errors, 647–648
 - in MySQL, 360–361
 - overview of, 644
 - parameters, 649
 - PHP configuration and, 727–731
 - throwing exceptions, 653–654
 - triggering errors, 646–647
 - error levels, 644–645**
 - `error_log`, **configuration directive, 647–648**
 - `error_reporting()`, **651–652**
 - escape sequences, for strings, 74**
 - event handlers, for parsing XML, 582–583**
 - `Exception` **class, 655**
 - `Exception` **object, 654**

exceptions. See also error handling

- catching exceptions, 654
- creating exception classes, 655
- exception objects for handling errors, 652–653
- throwing exceptions, 653–654

exclamation sign (!), specifying position in pattern matching, 549

`explode()`, **string function, 136–137, 562–563**

expressions. See also regular expressions

- including complex expressions within strings, 75–76
- overview of, 40
- as return value of a function call, 143

eXtensible Markup Language. See XML (eXtensible Markup Language)

Extensible Stylesheet Language (XSL), 613–615

`extension_dir`, **configuration directive, 738**

extensions

- dynamic, 737–739
- module settings, 739–756
- ODBC extension for connecting to ODBC databases, 763

F

`fclose()`, **for closing files, 303**

`feof()`, **for testing end of file, 307–308**

`fgetc()`

- reading one character at a time, 304
- using in conjunction with `feof()`, 308

`fgetcsv()`, **for reading CSV files, 309–310**

`fgets()`, **for reading one line at a time, 308–309**

Fibonacci sequence

- creating with recursion, 161–163
- displaying with `HTML_Table` package, 452–454

fields

- cookie, 275
- in relational databases, 341

fields, HTML forms

- empty, 234–236
- in HTML form example, 226–229
- multi-value, 236

`file()`, **for reading file content into array, 310–311**

file handles, 300

file size, 258–259

file system. See also directories

- accessing information on files, 298
- changing permissions, 314–315
- checking permissions, 315–316
- closing files, 303
- copying, renaming, and deleting files, 316
- example hit counter, 305–307
- exercise, 335
- exercise solutions, 696–697
- opening files, 300–302
- overview of, 297
- permissions, 313–314
- random access to file data, 312
- reading CSV files, 309–310
- reading one line at a time, 308–309
- reading/writing entire files, 310–312
- reading/writing strings of characters, 304–305
- reading/writing to files, 303–304
- retrieving filenames, 299
- summary, 334–335
- testing for end of file, 307–308
- time-related file functions, 299
- understanding files and directories, 298
- working with Unicode files in PHP 6, 302–303

file uploads

- accessing information on uploaded files, 257–258
- creating, 257
- example script for, 260–264
- limiting size of, 258–259
- PHP configuration and, 736
- storing/using uploaded files, 259

`file_exists()`, **for getting information on files, 298**

`file_get_contents()`, **311–312**

`file_put_contents()`, **311–312**

`file_uploads`, **configuration directive, 736**

`fileatime()`, **time-related file functions, 299**

`filectime()`, **time-related file functions, 299**

`filemtime()`, **time-related file functions, 299**

filenames, retrieving, 299

`fileperms()`, **for checking file permissions, 316**

files

- accessing information on, 298
- closing, 303
- copying, renaming, and deleting, 316
- distinguishing directory files from regular files, 322
- opening, 300–302

- random access to file data, 312
- reading CSV files, 309–310
- reading/writing entire files, 310–312
- reading/writing to, 303–304
- retrieving filenames, 299
- testing for end of file, 307–308
- time-related functions, 299
- understanding, 298
- working with Unicode, 302–303
- filtering, checking input with, 643**
- final classes and methods, for blocking inheritance and overriding, 199–200**
- floating-point numbers**
 - casting values to, 39
 - printf() precision specifier, 93
- folders. See directories**
- fonts**
 - displaying system fonts, 532–533
 - TrueType fonts used with images, 533–534
- fopen()
 - configuration directives, 736–737
 - locking open files, 312
 - opening files, 300–302
- for **loops, 61–63**
- foreach()
 - altering array values, 115–116
 - example displaying an array of books, 119–121
 - looping through array keys and values, 114–115
 - looping through array values, 114
 - overview of, 113–114
- foreign keys, in relational databases, 343**
- form controls. See controls, HTML forms**
- <form>...</form> tags, 222**
- format strings**
 - date strings, 478–481
 - general purpose formatting, 89–90
 - overview of, 89
 - printf() for swapping arguments, 93–94
 - printf() padding specifier, 92
 - printf() precision specifier, 93
 - printf() sign specifier, 91
 - printf() type specifier, 90–91
 - sprintf() for storing results instead of printing, 94
- formatting numbers, 98–99**
- formatting strings, 89**
- forms**
 - creating contact form script, 501–505
 - example validating form input, 564–570
 - HTML. See HTML forms
 - Web. See Web forms
- fpassthru(), **312**
- fread()
 - reading strings of characters, 304
 - using in conjunction with feof(), 308
- FreeType2 library, 533**
- fseek(), **for random access to file data, 312–313**
- ftell(), **for random access to file data, 312–313**
- function calls, 631**
- functions**
 - for adding/removing array elements, 128
 - calling, 142–144
 - for case conversion, 88–89
 - creating anonymous, 154–155
 - defining parameters of, 145–146
 - directory, 319
 - example creating Fibonacci sequence with recursion, 161–163
 - example for sorting words by length, 155–158
 - exercise solutions, 683–685
 - exercises, 164
 - global variables, 152–153
 - for manipulating array elements, 110
 - methods compared with, 174
 - modular code and, 620
 - MySQL, 382–385
 - optional parameters and default values, 147–148
 - overview of, 141
 - passing references to, 159
 - recursive, 160–161
 - references and, 158–159
 - returning references from, 160
 - returning values from, 148–150
 - SimpleXML, 607, 612
 - static variables for preserving values, 153–154
 - string explode and implode functions, 136–137
 - for summarizing query data, 376–377
 - summary, 163–164
 - type testing, 37
 - usefulness of, 142
 - variable scope and, 150–152
 - what they are, 141–142
 - working with variable functions, 144–145
 - writing own, 145
- fwrite(), **for writing data to a file, 304–305**

G

`_get()`, **overloading property access with, 184–187**

`get` **method**

- creating HTML forms with, 225
- sending form data to server, 229

`get_class()`, **for determining class of an object, 215–216**

`getdate()`

- extracting date and time values from timestamps, 475–476
- working with UNIX timestamps, 299

`gettype()`, **for testing variable type, 36–37**

GIF

- image formats, 509
- outputting, 511–512

global variables, 152–153

`gmtime()`, **473–474**

GMT (Greenwich Mean Time)

- creating timestamps from GMT date and time values, 473–474
- UTC compared with, 472

GNU, coding standard, 631

greater than (>), PHP comparison operator, 44

greater than or equal to (>=), PHP comparison operator, 44

greedy matching, 546

GROUP BY clause, for query results, 378–379

H

handling errors. See error handling

hard drives, as persistent storage mechanism, 297

`header()`, **for modifying HTTP responses, 493–494**

headers

- email and, 498–499
- HTTP request headers, 489–490
- HTTP response headers, 492

heredoc syntax, for delimiters, 76–77

hidden fields

- in HTML forms, 227, 249
- in multi-step HTML form, 250

high-quality applications

- allowing script to handle errors, 648–651
- application logic separated from presentation logic, 660–661

automating code testing, 666–667

catching exceptions, 654

checking input, 641–643

coding standards for consistency, 630–631

comments, 632–633

controlling where error messages are sent, 647

creating exception classes, 655

documenting code, 631–632

dynamic includes, 625

encoding output, 643–644

error handling and, 644

error levels, 644–645

example separating application logic from presentation logic, 662–666

example simulating spaceship flight, 655–660

example using phpDocumentor, 636–640

example using PHPUnit tests, 667–670

exception objects for handling errors, 652–653

exercise solutions, 716–718

exercises, 672

fine tuning error reporting, 651–652

include paths, 623–625

including/requiring files, 621–622

including/requiring files once only, 622–623

logging errors, 647–648

modular code for, 620

namespaces for avoiding clashes, 625–630

overview of, 619

phpDocumentor for external documentation, 633–635

summary, 671

throwing exceptions, 653–654

triggering errors, 646–647

hints, checking method arguments, 180–182

hit counter, 305–307

.htaccess file, Apache configuration directive, 719

HTML (Hypertext Markup Language)

- embedding PHP within, 25–27
- mixing decisions and loops with, 70–72
- sending HTML email, 500–501

HTML forms, 250–256

accessing information on uploaded files, 257–258

capturing form data, 230

creating, 223–225

creating file upload forms, 257

creating Web forms, 242

empty fields on, 234–236

example creating file upload script, 260–264

- example creating interactive Web form, 242–249
- example creating multi-step form, 250–256
- example registration form with multi-value fields, 237–242
- example writing form handler, 231–234
- exercise solutions, 689–691
- exercises, 265
- form fields, 226–229
- how they work, 222
- limiting size of file uploads, 258–259
- multi-value fields, 236
- overview of, 221–222
- redirecting after form submission, 264–265
- secure handling of form data, 234
- storing variables in, 249
- storing/using uploaded files, 259
- summary, 265
- HTML tables, creating with `HTML_Table` package, 450–452**
- `HTML_Common` package, 450
- `HTML_QuickForm` package
 - checking input with, 643
 - creating Web forms with, 455
 - example calculating age in days, 483–487
 - example registration form using, 462–469
 - installing, 455
 - validation rules, 460–462
 - working with, 455–460
- `HTML_QuickForm_Renderer_Tableless` package, 483–487
- `HTML_Table` package
 - creating HTML tables with, 450–452
 - example displaying Fibonacci numbers with, 452–454
- `htmlspecialchars()`, for encoding output, 643
- HTTP (Hypertext Transfer Protocol)**
 - modifying responses, 493–494
 - overview of, 488–489
 - requests, 489–490
 - responses, 490–493
- HTTP headers, cookies sent as part of, 274**
- `http_build_query()`, for creating query strings, 270
- `httpd.conf`, Apache configuration directive, 719
- Hypertext Markup Language. See HTML (Hypertext Markup Language)**
- Hypertext Transfer Protocol. See HTTP (Hypertext Transfer Protocol)**
- I**
- `idate()`, extracting date/time values from timestamps, 476–477
- identical (===), PHP comparison operator, 44**
- identifiers, coding standards for, 631**
- if statements, for simple decision-making, 52–53**
- `ignore_repeated_errors`, configuration directive, 727
- IIS (Internet Information Server), 23**
- image fields, in HTML forms, 227**
- `imagearc()`, for drawing arcs, 515–516
- `imagecolorallocate()`, for allocating image colors, 510–511
- `imagecolorat()`, for retrieving palette index of colors, 526
- `imagecopy()`, for copying images, 524–525
- `imagecopymerge()`, for adding transparency to images, 527–528
- `imagecreate()`, for creating images, 510
- `imagecreatefrom...()`, for creating images based on existing images, 521
- `imagedestroy()`, for deleting images, 512
- `imageellipse()`, for drawing circles and ellipses, 515
- `imagefttext()`, for TrueType fonts, 533–534
- `imagegif()`, for outputting GIF images, 511–512
- `imagejpeg()`, for outputting JPEG images, 511–512
- `imageline()`, for drawing lines, 513–514
- `imagepng()`, for outputting PNG images, 511–512
- `imagepolygon()`, for drawing polygons, 516–517
- `imagerectangle()`, for drawing rectangles, 514
- images**
 - adding standard text to, 531
 - allocating colors when creating, 510–511
 - applying watermarks to, 523–525
 - color theory and, 508
 - computer graphics and, 507
 - coordinate systems for drawing shapes, 508–509
 - creating, 510

images (continued)

- drawing arcs, 515–516
 - drawing circles and ellipses, 515
 - drawing individual pixels, 512–513
 - drawing lines, 513–514
 - drawing polygons, 516–517
 - drawing rectangles, 514
 - example displaying system fonts, 532–533
 - example displaying text with TrueType font, 534–535
 - example drawing rectangle with rounded corners, 517–520
 - exercise solutions, 708–712
 - exercises, 537
 - manipulating, 520
 - opacity of, 527–528
 - opening existing, 521–523
 - outputting, 511–512
 - overview of, 507
 - summary, 536
 - thumbnails of, 528–531
 - transparency of, 525–527
 - TrueType fonts used with, 533–534
 - types of, 509
- imagejpeg(), for drawing individual pixels, 512–513**
- imagestring(), for adding text to images, 531**
- implode(), string function, 137**
- include(), for including files, 621–622**
- include paths, 623–625**
- include_once(), for including files only once, 622–623**
- include_path, configuration directive, 734**
- including files**
- dynamic includes, 625
 - include paths, 623–625
 - once only, 622–623
 - overview of, 621–622
 - writing of modular code and, 620
- increment/decrement operators, 44–45**
- indexed arrays**
- overview of, 102
 - retrieving last element of, 108–109
 - sorting, 121–122
- indexes**
- accessing characters within strings, 78
 - relational databases, 347–348
- inheritance**
- example creating parent and child classes, 193–196
 - final classes and methods for blocking, 199–200
 - overriding methods in the parent class, 196–198
 - overview of, 192–193
 - preserving functionality of parent class, 198–199
- ini_set, for setting configuration directives, 720**
- initializing variables**
- array variables, 104–105
 - overview of, 35
- INSERT command, MySQL, 356**
- insertBefore(), for moving elements in XML documents, 605–606**
- inserting records, using PHP scripts, 403–405**
- install command, PEAR packages, 446–447**
- installing PHP. See PHP installation/set up**
- integers, casting values to, 39**
- integration testing, 666**
- interaction with outside world**
- dates and times. *See* dates and times
 - email, 497–501
 - example creating contact form script, 497–501
 - exercise solutions, 705–707
 - exercises, 506
 - getting information from Web server, 494–497
 - overview of, 471
 - summary, 505–506
 - working with HTTP. *See* HTTP (Hypertext Transfer Protocol)
- interactive scripts, creating from command-line, 768–770**
- interactive Web forms, 242–249**
- interactive Web sites, 3**
- interface keyword, 204**
- interfaces**
- example creating, 205–209
 - overview of, 204–205
- Internet Information Server (IIS), 23**
- is_dir(), for determining directory files, 322**
- is_executable(), for checking file permissions, 315**
- is_file(), for determining regular files, 322**
- is_readable(), for checking file permissions, 315**
- is_writable(), for checking file permissions, 315**
- _isset(), overloading method, 191–192**
- iteration, 160. See also loops**

J**Java**

- PHP compared with, 6
- as strongly-typed language, 36

joins

- pulling query data from multiple tables, 379–381
- in relational databases, 343

JPEG

- example displaying, 521–523
- image formats, 509
- outputting JPEG images, 511–512

justifying text, 83–87**K**

`key()`, **for manipulating arrays elements, 110**

keys, arrays and, 102

keys, in RDBMS

- foreign, 343
- primary, 342, 347
- UNIQUE keys compared with primary keys, 370
- using, 347–348

`krsort()`, **for sorting associative array keys, 123**

`ksort()`, **for sorting associative array keys, 123**

L

LAMP (Linux, Apache, MySQL, and PHP), 12

Language Options section, of `php.ini`, 722–726

`lcfirst()`, **for case conversion, 88**

length, of strings, 77–78

less than (<), PHP comparison operator, 44

less than or equal to (<=), PHP comparison operator, 44

LIKE operator, for flexible queries, 374–376

lines, drawing, 513–514

links, example finding all links in a Web page, 553–555

Linux. See Ubuntu

Linux, Apache, MySQL, and PHP (LAMP), 12

`list()`, **for converting array to a list of variables, 137–138**

list boxes, on HTML forms, 228

local variables, 151–152

`log_errors`, **configuration directive, 647**

LogEntry class

- adding deletion method to, 430
- enhancing for members' area of Web site, 419
- member record viewer and, 394–395

logging

- errors, 647–648
- PHP configuration and, 727–731

logical operators, 45–46

login script, for members' area of Web site, 421–423

login system, creating, 292–295

logout script, for members' area of Web site, 424

looping through arrays

- `foreach()` for altering values, 115–116
- `foreach()` for looping through keys and values, 114–115
- `foreach()` for looping through values, 114
- multidimensional arrays and, 119–121
- overview of, 113–114

loops. See also decisions

- `break` statements for escaping, 64
- `continue` statement for skipping iterations, 64–65
- defined, 51
- `do...while` loops, 60–61
- example homing pigeon simulator, 66–70
- exercise solutions, 674–677
- exercises, 72
- mixing with HTML, 70–72
- nested, 65–66
- overview of, 59
- `for` statements, 61–63
- summary, 72
- `while` loops, 59–60

loosely-typed languages, 36

lossless compression, 509

lossy compression, 509

lowercase, converting strings to/from, 87–89

`ltrim()`, **for trimming strings, 95**

M

Mac, Apache, MySQL, and PHP. See MAMP (Mac, Apache, MySQL, and PHP)

Mac OS X

- PHP installation on, 17–19
- starting MySQL server on, 349–350
- testing PEAR package manager on, 443–444

Magic Quotes feature, removed in PHP 6, 8

`mail()`

- controlling return path of email address, 499–500
- sending email, 497–498
- specifying sender address and adding headers, 498–499

main logic, of text editor code, 330–331

MAMP (Mac, Apache, MySQL, and PHP)

- PHP installation on Mac OS X, 17–19
- starting MySQL server on Mac OS X, 349–350
- testing PEAR package manager on Mac OS X, 443–444

manipulating

- arrays, 121
- images, 520

Mastering Regular Expressions (Friedl), 542

mathematic functions, MySQL, 385

`max()`, for summarizing query data, 377

Member class

- member management application, 428–430
- member record viewer application, 390–394
- member registration application, 409–411

member manager application

- adding deletion method to `LogEntry` class, 430
- adding update and delete methods to `Member` class, 428–430
- creating `view_member.php` script, 431–437
- overview of, 428
- testing, 437–438
- tweaking `view_members.php` script, 431

member record viewer application

- `common.inc.php`, 387
- `config.php` file, 386–387
- `DataObject` class, 388–390
- `LogEntry` class, 394–395
- `Member` class, 390–394
- overview of, 385–386
- testing application, 400–401
- `view_member.php` script, 399–400
- `view_members.php` script, 395–398

member registration application

- adding common code to `common.inc.php`, 408–409
- creating registration script, 411–416
- enhancing `Member` class, 409–411
- overview of, 408
- testing, 416

members' area, of Web site

- adding authentication method, 418
- adding common code to `common.inc.php`, 420–421
- creating pages for, 424–426
- enhancing `LogEntry` class to record page views, 419
- login script for, 421–423
- logout script for, 424
- overview of, 417–418
- testing, 426–428

menus, on HTML forms, 227–228

merging arrays, 134–136

method calls

- overloading, 187
- overview of, 175

methods

- abstract, 200–204
- accessing object properties from, 175–176
- adding parameters to, 175
- calling, 175
- constructors, 209–210
- creating, 174
- destructors, 210–211
- DOM class, 595
- `Exception` object, 654
- final methods for blocking inheritance and overriding, 199–200
- hints used to check method arguments, 180–182
- `HTML_QuickForm` package, 456–457
- `HTML_Table` package, 450–452
- overloading method calls with `_call()`, 187
- overview of, 168
- SimpleXML, 607
- static methods, 179–180
- visibility of, 174

microseconds, 481–483

Microsoft Access database, 763

`microtime()`, for working with microseconds, 481–482

MIME (Multipurpose Internet Mail Extensions)

- file upload forms and, 257
- sending HTML email and, 500–501

`min()`, for summarizing query data, 377

`mkdir()`, for creating directories, 320

`mktime()`, for creating timestamps from date/time values, 473

mode (permissions), 314

models, database, 339–340

modular code, for high-quality applications, 620

multidimensional arrays

- accessing elements of, 118–119
- creating, 117–118
- looping through, 119–121
- overview of, 116–117

multi-line comments, 29–30, 632**Multipurpose Internet Mail Extensions (MIME)**

- file upload forms and, 257
- sending HTML email and, 500–501

multi-step form example, HTML forms, 250–256**multi-value fields, HTML forms**

- example registration form with, 237–242
- overview of, 236

MySQL

- adding data to tables, 355
- choosing a database engine, 340
- connecting with from PHP, 359
- creating databases, 353–354
- creating tables, 354–355
- data types, 344
- date and time data types, 345
- deleting data from tables, 358
- deleting tables and databases, 358–359
- error handling, 360–361
- example reading database table with PHP, 362–364
- exercise solutions, 698–699
- exercises, 366
- installing PHP on Mac OS X, 17–18
- installing PHP on Ubuntu, 14
- installing PHP on Windows, 16
- making connection from PHP, 360
- numeric data types, 344–345
- PHP 6 and Unicode and, 365
- PostgreSQL compared with, 759–760
- programs in, 349
- reading data from PHP, 361–374
- reading data in tables, 356–357
- setting up root password, 350–353
- starting MySQL server, 349–350
- string data types, 346–347
- summary, 365–366
- updating data in tables, 357

MySQL, data manipulation

- creating members' area. *See* members' area, of Web site
- deleting records, 407
- exercise solutions, 701–702
- exercises, 439

- inserting records, 403–405

- member management. *See* member manager application
- member registration. *See* member registration application
- overview of, 403
- summary, 438–439
- updating records, 406

MySQL, data retrieval

- aliases for simplifying queries, 381–382
- BINARY attribute and collations and, 369
- eliminating duplicate results from queries, 377–378
- ENUM data type and, 370–371
- exercise solutions, 700–701
- exercises, 402
- grouping query results, 378–379
- member record viewer application. *See* member record viewer application
- operators and functions, 382–385
- overview of, 367
- pattern matching for flexible queries, 374–376
- querying multiple tables, 379–381
- SELECT statements for, 371–372
- setting up sample database, 367–369
- sorting query results, 373–374
- summarizing query data, 376–377
- summary, 401–402
- TIMESTAMP data type and, 371
- UNIQUE constraint and, 370
- WHERE clause for limiting number of rows returned, 372–373

MySQL improved (mysqli), 359**mysqli (MySQL improved), 359****N****names/naming**

- classes, 168
- filenames, 299
- identifiers, 631
- renaming files, 316
- usernames, 418
- variables, 34

namespaces

- for avoiding clashes, 625–630
- modular code and, 620
- XML, 581

nested arrays. *See* multidimensional arrays

nested includes, 622

nested loops, 65–66

`Net_UserAgent_Detect` package, 448–449

`next()`, for manipulating arrays elements, 110

nodes, changing in XML documents, 603–605

non-greedy matching, 546

normal forms, 342

normalization, relational databases and, 341–342

not (!), PHP logical operator, 46

NOT (!), setting configuration directive values with, 720

NOT (~)

PHP bitwise operator, 43

setting configuration directive values with, 720

NOT, Boolean operator, 384

not equal (!=), PHP comparison operator, 44

not identical (!==), PHP comparison operator, 44

NOTLIKE operator, for flexible queries, 376

`now()`, MySQL functions, 384

nowdoc syntax, for delimiters, 76–77

NULL values, SQL statements and, 349

NULL-safe version of equal to (<=>), MySQL comparison operator, 383–384

`number_format()`, for formatting numbers, 98–99

numbers

formatting string numbers, 98–99

MySQL numeric types, 344–345

O

object oriented programming. See OOP (object oriented programming)

objects

accessing object properties from methods, 175–176

constructors, 209–210

creating, 168–169

destructors, 210–211

determining class of, 215–216

exception objects for handling errors, 652–653

overloading. See overloading

overview of, 167–168

storing as strings, 213–214

working with directory objects, 321–322

ODBC (Open Database Connectivity), 763–764

ODF (OpenDocument Format), 573

Office Open XML (OOXML), 573

one-to-many relationships, in relational databases, 343

OOP (object oriented programming)

abstract classes and methods, 200–204

accessing properties, 170–172

accessing properties from methods, 175–176

adding parameters to methods, 175

advantages of, 166–167

automatically loading class files, 212–213

basic concepts, 167

calling methods, 175

class constants, 173–174

classes, 167

classes following active record of design pattern, 388

constructors for creating objects, 209–210

creating classes and objects, 168–169

creating methods, 174

declaring properties, 170

destructors for cleaning up objects, 210–211

determining class of an object, 215–216

encapsulation, 182–183

evolution of PHP and, 7, 9

example creating interface, 205–209

example creating parent and child classes, 193–196

example creating wrapper string class, 188–191

example of a car moving, 177–179

exercise solutions, 686–688

exercises, 218

final classes and methods for blocking inheritance and overriding, 199–200

hints used to check method arguments, 180–182

inheritance, 192–193

interfaces, 204–205

method visibility and, 174

methods, 168

objects, 167–168

overloading, 183–184

overloading methods, 187, 191–192

overloading property access, 184–187

overriding methods in the parent class, 196–198

overview of, 165

preserving functionality of parent class, 198–199

properties, 168

property visibility and, 169–170

static methods, 179–180

- static properties, 172–173
- storing objects as strings, 213–214
- summary, 217
- what it is, 166
- working with directory objects, 321–322
- OOXML (Office Open XML), 573**
- opacity, of images, 527–528**
- Open Database Connectivity (ODBC), 763–764**
- open source databases**
 - MySQL, 759
 - PostgreSQL, 759
 - SQLite, 759–761
- `opendir()`, for opening directories, 317
- OpenDocument Format (ODF), 573**
- opening files, 300–302**
- opening images, 521–523**
- operands, 40**
- operators**
 - arithmetic operators, 41
 - assignment operators, 41–42
 - bitwise operators, 42–43
 - comparison operators, 43–44
 - defined, 33
 - increment/decrement operators, 44–45
 - logical operators, 45–46
 - MySQL, 382–385
 - overview of, 40–41
 - precedence, 47–48
 - string operators, 46–47
- Options section, of `php.ini`, 722**
- OR (!), PHP bitwise operator, 42**
- or (||), PHP logical operator, 46**
- OR, Boolean operators, 384**
- or, PHP logical operator, 46**
- Oracle, 762**
- Oracle Berkeley DB, 761**
- ORDER BY clause, for sorting query results, 373–374**
- outputting images, 511–512**
- overloading**
 - example creating wrapper string class, 188–191
 - method calls, 187, 191–192
 - overview of, 183–184
 - property access, 184–187
- overriding**
 - blocking with final classes and methods, 199–200
 - methods in parent classes, 196–198

P

- padding specifier, `printf()`, 92**
- padding strings, with `str_pad()`, 96**
- page headers, displaying, 334**
- page views, recording visitors to member's area of Web site, 419**
- pagination, using square numbers in, 270–274**
- parameters**
 - adding to methods, 175
 - defining function, 145–146
 - error handlers, 649
 - optional, 147–148
- parent classes**
 - example creating, 193–196
 - overriding methods in, 196–198
 - overview of, 192
 - preserving functionality of, 198–199
- parse errors, XML, 584–585**
- parsers, XML. See XML Parser**
- passwords**
 - adding authentication methods to `Member` class, 418
 - setting up MySQL root password, 350–353
- paths, PHP configuration and, 734–736**
- pattern matching, 541–542**
 - altering matching behavior with pattern modifiers, 560–562
 - anchors for matching at specified positions, 548–550
 - backreferences, 547–548
 - example finding all links in a Web page, 553–555
 - finding multiple matches, 550–552
 - for flexible queries, 374–376
 - greedy and non-greedy, 546
 - matching alternative patterns, 548
 - matching characters using character classes, 544–545
 - matching literal characters, 542–543
 - matching multiple characters, 545–546
 - subpatterns as means of grouping patterns, 546–547
- pattern modifiers, 560–562**
- PDO (PHP Data Objects)**
 - connecting to ODBC database with, 763
 - database systems supported by, 764
 - error handling and, 360–361
 - example using PDO to create Oracle database, 762

PDO (PHP Data Objects) (continued)

- example using PDO to create SQLite database, 758–759
- making database connections and, 360
- passing `DELETE` statement to MySQL from PHP, 407
- passing `INSERT` statement to MySQL from PHP, 404
- passing `UPDATE` statement to MySQL from PHP, 406
- reading data from PHP, 361–374
- ways to connect with MySQL from PHP, 359

PEAR (PHP Extension and Application Repository)

- coding standard, 631
- example calculating age in days, 483–487
- example detecting visitor's browser, 448–449
- exercise solutions, 703–704
- exercises, 470
- `HTML_Table` package. See `HTML_Table` package
- installing dependencies, 447
- installing PEAR packages, 446–447
- overview of, 441–442
- `phpDocumentor` package. See `phpDocumentor` package
- summary, 470
- testing PEAR package manager on Mac OS X and MAMP, 443–444
- testing PEAR package manager on Ubuntu, 442–443
- testing PEAR package manager on WampServer on Windows, 444–446
- uninstalling packages, 447
- using PEAR packages, 448
- `Validate` package, 643

`preg_grep()`, for searching arrays, 556

Perl, 6

permissions, file

- changing, 314–315
- checking, 315–316
- overview of, 313–314

persistent storage mechanisms, 297

PHP, introduction to

- evolution of, 7
- reasons for using, 5–7
- summary, 9
- what is PHP, 3–5
- what's new in PHP 6, 7–9

PHP configuration

- about `php.ini`, 720–722
- data handling, 731–734
- dynamic extensions, 737–739
- error handling and logging, 727–731
- file uploads, 736
- `fopen()` wrappers, 736–737
- Language Options section of `php.ini`, 722–726
- miscellaneous section, 726–727
- module settings, 739–756
- Options section of `php.ini`, 722
- overview of, 719–720
- path and directories, 734–736
- resource limits and, 727

PHP Data Objects. See PDO (PHP Data Objects) **PHP Extension and Application Repository. See PEAR (PHP Extension and Application Repository)**

PHP installation/set up

- comments, 29–30
- compiling PHP, 23–24
- creating first script, 24–25
- embedding PHP within HTML, 25–27
- enhancing scripts, 28–29
- exercise, 30
- exercise solutions, 673
- on Mac OS X, 17–19
- prerequisites for creating/running scripts, 12
- running remotely, 24
- server options for running PHP, 23
- summary, 30
- testing PHP, 20–21
- testing Web server, 19–20
- time zone setting, 21–22
- on Ubuntu Linux, 12–14
- on Windows OSs, 15–17

PHP language basics

- arithmetic operators, 41
- assignment operators, 41–42
- bitwise operators, 42–43
- changing type by casting, 38–40
- changing variable type, 38
- comparison operators, 43–44
- constants, 48–50
- creating variables, 34–35
- data types, 35–36
- exercise solutions, 674
- exercises, 50
- increment/decrement operators, 44–45
- logical operators, 45–46

- loose typing, 36
 - naming variables, 34
 - operator precedence, 47–48
 - operator types, 40–41
 - overview of, 33
 - string operators, 46–47
 - summary, 50
 - testing variable type, 36–38
 - using variables, 33–34
 - `phpdoc` **command**, 634
 - phpDocumentor package**
 - example using, 636–640
 - for external documentation, 633–635
 - `php.ini`
 - Language Options section of, 722–726
 - module settings, 739–756
 - Options section of, 722
 - overview of, 720–722
 - setting configuration directives with, 719
 - PHPUnit**
 - automating code testing with, 666–667
 - example writing simple unit tests, 667–670
 - PIs (processing instructions), for XML documents, 576**
 - pixels, drawing individual, 512–513**
 - PNG**
 - image formats, 509
 - outputting PNG images, 511–512
 - polygons, drawing, 516–517**
 - `post` **method, for sending form data to server, 229**
 - `post_max_size`, **configuration directive, 731, 736**
 - PostgreSQL, 759–761**
 - compared with MySQL, 759–760
 - example script, 760–761
 - overview of, 759
 - precedence, operator, 47–48**
 - precision specifier, `printf()`, 93**
 - `preg_match()`, **for pattern matching, 541–542**
 - `preg_match_all()`
 - example finding all links in a Web page, 553–555
 - finding multiple matches with, 550–552
 - `preg_replace()`, **for replacing text, 557–559**
 - `preg_replace_callback()`, **for replacing text using a callback function, 560**
 - `preg_split()`, **for splitting strings, 157, 562–564**
 - presentation logic, separating application logic from, 662–666**
 - `prev()`, **for manipulating arrays elements, 110**
 - primary keys**
 - in relational databases, 342
 - UNIQUE keys compared with, 370
 - `print()`, **25**
 - `print_r()`, **for outputting entire array, 105–107**
 - `printf()`
 - padding specifiers and, 92
 - precision specifiers and, 93
 - sign specifiers and, 91
 - for swapping arguments, 93–94
 - type specifiers and, 90–91
 - `private` **methods, 174**
 - private properties, 170**
 - procedural programming, 166**
 - processing instructions (PIs), for XML documents, 576**
 - programs, in MySQL, 349**
 - properties**
 - accessing, 170–172
 - accessing from methods, 175–176
 - declaring, 170
 - overview of, 168
 - static, 172–173
 - visibility of, 169–170
 - `protected` **methods, 174**
 - protected properties, 170**
 - `public` **methods, 174**
 - public properties, 169**
 - Python, 6**
- ## Q
- quantifiers**
 - greedy and non-greedy matching, 546
 - for matching multiple characters, 545–546
 - queries, SQL**
 - aliases for simplifying, 381–382
 - eliminating duplicate results, 377–378
 - grouping results, 378–379
 - list of SQL statements, 348
 - pattern matching for flexible queries, 374–376
 - querying multiple tables, 379–381
 - sorting results, 373–374
 - SQLite and, 758
 - summarizing data returned by, 376–377

query SQL (continued)

`WHERE` clause for limiting number of rows returned by queries, 372–373

query strings

accessing data in, 270
building, 268–270
example using square numbers in pagination, 270–274
overview of, 268
passing session IDs in, 289–290

quotation marks, in string syntax, 74–75

R

radio buttons, on HTML forms, 226

RAM

setting resource limits during PHP configuration, 727
as temporary storage mechanism, 297

raster images, 509

RDBMS (Relational Database Management Systems). See also MySQL

date and time data types, 345
exercise solutions, 698–699
exercises, 366
indexes and keys, 347–348
MySQL data types, 344
normalization and, 341–342
numeric data types, 344–345
Oracle, 762
overview of, 341
SQL for communicating with, 343
SQL Server, 763
SQL statements, 348–349
string data types, 346–347
summary, 365–366
types of database models, 340

`readdir()`, for reading directories, 317

`readfile()`, for reading files, 312

reading/writing

entire files, 310–312
to files, 303–304
session data, 283
strings of characters, 304–305

`record()`, for recording page views, 419

records, in relational databases, 341

rectangles

drawing, 514
example drawing rectangle with rounded corners, 517–520

recursive functions

example creating Fibonacci sequence with recursion, 161–163
overview of, 160–161

red, green, blue (RGB) color model, 508

redirection, of HTML forms, 264–265

references

to array elements, 116
overview of, 158–159
passing to functions, 159
returning from functions, 160

Register Globals feature, removed in PHP 6, 8

registration form

example using `HTML_QuickForm` package, 462–469
with multi-value fields, 237–242

registration scripts, for member registration application, 411–416

regular expressions

anchors and, 548–550
backreferences, 547–548
checking input, 642–643
example finding all links in a Web page, 553–555
example validating form input, 564–570
exercise solutions, 712–713
exercises, 571–572
finding multiple matches, 550–552
greedy and non-greedy matching, 546
matching alternative patterns, 548
matching characters using character classes, 544–545
matching literal characters, 542–543
matching multiple characters, 545–546
overview of, 539
pattern matching and, 541–542
pattern modifiers and, 560–562
replacing text using callback function, 560
replacing text with `preg_replace()`, 557–559
searching arrays, 556
splitting strings, 562–564
subpatterns and, 546–547
summary, 571
what they are, 540–541

reindexing arrays, 122

remote operation, of PHP, 24

`removeChild()`

moving elements in XML documents, 606
removing XML elements, 602

`rename()`, for renaming files, 316

repetition. See loops

`replaceData()`, for changing XML nodes and attributes, 603

replacing text

overview of, 557

with `preg_replace()`, 557–559

within strings, 81

using callback function, 560

request messages, HTTP, 489–490

`request_order`, configuration directive, 731

`require()`, for requiring files, 621–622

`require_once()`, for requiring files only once, 621–622

requiring files

dynamic includes, 625

include paths, 623–625

once only, 622–623

overview of, 621–622

writing of modular code and, 620

`reset()`, for manipulating array elements, 110

resource data types, 300

resource limits, PHP configuration and, 727

response messages, HTTP

modifying, 493–494

overview of, 490–493

result sets, of SQL queries, 348

return statement

functions returning values and, 150

returning references from functions, 160

returning values from a method, 175

`rewind()`, for random access to file data, 312–313

`rewinddir()`, for resetting directory pointer, 319

RGB (red, green, blue) color model, 508

`rmdir()`, for deleting directories, 320

root element, XML, 574

`root password`, setting up MySQL, 350–353

rows

in relational databases, 341

`WHERE` clause for limiting number returned by queries, 372–373

`rsort()`, for sorting indexed arrays, 121–122

RSS feeds, 573

`rtrim()`, for trimming strings, 95

Ruby, 6

running scripts, from command-line, 766–767

runtime errors, 644

S

Safe Mode, removed in PHP 6, 8

`saveFile()`, editing text files and, 332–333

scalar data types, 35

scheduling scripts, from command-line, 770–772

scripting languages, 4–5

scripts, example applications

age calculator, 483–487

browser detection, 448–449

car simulator, 177–179

directory hierarchy, traversing, 323–325

drawing lines, 513–514

drawing rectangle with rounded corners, 517–520

email contact form, 501–505

Fibonacci sequence displayed with `HTML_Table` package, 452–454

Fibonacci sequence with recursion, 161–163

file upload, 260–264

form handler, 231–234

form input validation, 564–570

greeting display, 57–58

hit counter, 305–307

homing pigeon simulator, 66–70

interactive Web form, 242–249

JPEG image, displaying, 521–523

links in a Web page, finding all, 553–555

list directory entries, 317–319

login script for members' area, 421–423

logout script for members' area, 424

member record viewer, 395–400

member registration, 411–416

multi-step HTML form, 250–256

OOP interface, 205–209

parent and child classes, 193–196

parsing XML files, 585–589

`phpDocumentor`, 636–640

PHPUnit testing, 667–670

reading MySQL tables, 362–364

reading XML documents, 591–594

registration form using `HTML_QuickForm` package, 462–469

registration form with multi-value fields, 237–242

remembering user information, 278–282

separating application logic from presentation logic, 662–666

shopping cart, 284–289

sorting words by length, 155–158

scripts, example applications (*continued*)

- spaceship flight simulator, 655–660
- SQLite database created with PDO, 758–759
- system fonts, displaying, 532–533
- text editor, 326–328
- TrueType font, displaying text with, 534–535
- user login system, 292–295
- wrapper for string class, 188–191
- XML documents created using DOM, 596–599

scripts, PHP

- accessing cookies in, 277
- checking input with, 642
- command-line options, 772–773
- creating first, 24–25
- creating interactive scripts from command-line, 768–770
- encoding output with, 643
- enhancing, 28–29
- error handling with, 648–651
- examples of common, 4
- functions reused in, 142
- passing arguments from command-line, 767–768
- PHP as server-side scripting language, 4
- PHP for command-line scripting, 5
- prerequisites for creating/running, 12
- running from command-line, 766–767
- scheduling from command-line, 770–772

searching arrays, with `preg_grep()`, 556

searching strings

- overview of, 78–79
- `strbrk()` for finding set of characters, 81
- `strpos()` and `strrpos()` for locating text, 79–80
- `strstr()` for searching, 79
- `substr_count()` for finding number of occurrences, 80–81

security

- checking input and, 641–643
- cookies and, 282
- encoding output and, 641, 643–644
- of form data, 234

SELECT statement

- reading data from MySQL tables, 356–357
- SQL queries and, 348

SELECT statements

- aliases for simplifying queries, 381–382
- eliminating duplicate results from queries, 377–378
- grouping query results, 378–379

- overview of, 371–372
- pattern matching for flexible queries, 374–376
- querying multiple tables, 379–381
- sorting query results, 373–374
- summarizing query data, 376–377
- `WHERE` clause for limiting number of rows returned, 372–373

sender addresses, email and, 498–499

- `serialize()`, for storing objects as strings, 213

servers. See also Web servers

- MySQL, 349–350
- SQL Server, 763
- WampServer, 349–350, 444–446

server-side scripting, 4

session IDs (SIDs)

- overview of, 282
- passing in query strings, 289–290
- `session_destroy()`, for destroying sessions, 289
- `session_start()`, for creating sessions, 282–283

sessions

- changing behavior of, 290–291
- creating, 282–283
- destroying, 289
- example creating shopping cart, 284–289
- example creating user login system, 292–295
- overview of, 282
- passing session IDs in query strings, 289–290
- reading/writing session data, 283
- `_set()`, for overloading property access, 184–187

- `set_error_handler()`, 648–651

- `setcookie()`, 276

setting up PHP. See PHP installation/set up

- `settype()` function, for changing variable type, 38, 40

SGML (Standard Generalized Markup Language), 573

shapes, coordinate systems for drawing, 508–509

Shift left (<<), PHP bitwise operator, 43

Shift right (>>), PHP bitwise operator, 43

shopping cart script, 284–289

- `SHOW TABLE` command, MySQL, 355

SIDs (session IDs)

- overview of, 282
- passing in query strings, 289–290

sign specifiers, `printf()` and, 91

simple databases, 339**SimpleXML**

- converting between Simple XML and DOM objects, 612
- creating XML documents using, 610–612
- overview of, 606–608
- reading XML documents using, 608–610

single quotation marks (‘), in string syntax, 74–75**single-line comments, 29–30, 632****slash (/), in regular expression syntax, 543****snapshots, of PHP, 23****SOAP, 573****sort(), for sorting indexed arrays, 121–122****sorting**

- query results, 373–374
- words by length, 155–158

sorting arrays

- associative array keys, 123
- associative arrays, 122–123
- indexed arrays, 121–122
- multisorting, 124–128
- options for, 121

sparse arrays, 109**special data types, 36****sprintf(), for storing results instead of printing, 94****SQL (Structured Query Language)**

- for communicating with relational databases, 343
- Oracle’s use of, 762
- SQL statements used with relational databases, 348–349
- SQLite’s use of, 758

SQL queries. See queries, SQL**SQL Server, 763****SQLite, 757–759**

- example using PDO to create, 758–759
- features of/when to use, 758
- overview of, 757

square brackets ([])

- adding array elements and, 104–105
- handling multi-value fields in HTML forms, 236

square numbers, using in pagination, 270–274**ssh client, for running PHP remotely, 24****Standard Generalized Markup Language (SGML), 573****standards**

- for class names, 168
- coding standards for consistency, 630–631

state, preserving

- accessing cookies in scripts, 277
- accessing data in query strings, 270–274
- building query strings, 268–270
- changing session behavior, 290–291
- components of cookies, 274–275
- cookies, 274
- creating sessions, 282–283
- destroying sessions, 289
- example creating shopping cart, 284–289
- example creating user login system, 292–295
- example remembering user information, 278–282
- exercise solutions, 692–696
- exercises, 296
- overview of, 267
- passing session IDs in query strings, 289–290
- query strings, 268
- reading/writing session data, 283
- removing cookies, 278
- sessions, 282
- setting cookies, 276
- summary, 296

static keyword

- creating static method, 179–180
- creating static properties, 172–173

static methods, 179–180**static properties, 172–173****static variables, 153–154****static Web pages, 3****status codes, HTTP responses, 491****stepping through arrays, 109–113****storage**

- deciding how to store data, 338
- persistent vs. temporary, 297

str_pad(), for padding strings, 96**str_replace(), for replacing text within strings, 81–82****strbrk(), for finding set of characters, 81****string data types, MySQL, 346–347****string matching. See pattern matching****string operators, 46–47****strings**

- accessing characters within, 78
- case conversion, 87–89
- casting values to, 39
- complex expressions included within, 75–76
- converting to/from arrays, 136–137
- creating/accessing, 74–75
- delimiters, 76–77

strings (continued)

- exercise solutions, 677–678
- exercises, 100
- finding length of, 77–78
- formatting, 89
- formatting date strings, 478–481
- formatting numbers, 98–99
- MySQL functions, 385
- options for creating, 77
- overview of, 73
- padding, 96
- padding specifier, 92
- precision specifier, 93
- `printf()` for swapping arguments, 93–94
- replacing text within, 81
- searching, 78–79
- sign specifier, 91
- splitting with regular expressions, 562–564
- `sprintf()` for storing results instead of printing, 94
- storing functions in string variables, 144
- storing objects as, 213–214
- `str_replace()` for replacing all occurrences, 81–82
- `strbrk()` for finding set of characters, 81
- `strpos()` and `strrpos()` for locating text, 79–80
- `strstr()` for searching, 79
- `strtr()` for translating characters, 87
- `substr_count()` for finding number of occurrences, 80–81
- `substr_replace()` for replacing portion of a string, 82–87
- summary, 99
- trimming, 95
- type specifier, 90–91
- wrapping lines of text, 96–98
- `strlen()`, for finding length of strings, 77–78
- strongly-typed languages, 36**
- `strpos()`, for locating text within strings, 79–80
- `strrpos()`, for locating text within strings, 80
- `strstr()`, for searching strings, 79**
- `strtolower()`, for case conversion, 88
- `strtotime()`, creating timestamps from date and time strings, 474–475
- `strtoupper()`, for case conversion, 88
- `strtr()`, for translating characters, 87
- Structured Query Language. See SQL (Structured Query Language)**

subdirectories, 298

subpatterns

- backreferences and, 547–548
- as means of grouping patterns, 546–547

subroutines. See functions

`substr()`, for extracting characters from strings, 78

`substr_count()`, for finding number of occurrences, 80–81

`substr_replace()`, for replacing portion of a string, 82–87

subtraction (-), PHP arithmetic operator, 35

Sun Java, 631

superglobal arrays

`$_COOKIE`, 277

`$_FILES`, 257–258

`$_GET`, 230

`$_POST`, 230

`$_REQUEST`, 230

`$_SERVER`, 494–496

`$_SESSION[]`, 283, 289

superglobal variables, 153

`switch` statements, for testing one expression many times, 55–56

Synaptic Package Manager, 12–14

syntax errors, 644

syntax rules, XML, 577–578

T

tables

- creating HTML tables with `HTML_Table` package, 450–452
- in relational databases, 341

tables, MySQL

- adding data to, 355
- creating, 354–355
- deleting, 358–359
- deleting data from, 358
- inserting records, 403–405
- pulling query data from multiple tables, 379–381
- reading data in, 356–357
- updating data in, 357

tab-separated-value (TSV), 309

testing

- automating code testing with PHPUnit, 666–667
- member manager application, 437–438
- member record viewer application, 400–401

- member registration application, 416
 - members' area of Web site, 426–428
 - PHP install, 20–21
 - switch statements for testing one expression
 - many times, 55–56
 - text editors, 328–330
 - Web server, 19–20
 - text**
 - adding to images, 531
 - example displaying system fonts, 532–533
 - example displaying with TrueType font, 534–535
 - justifying, 83–87
 - replacing text using callback function, 560
 - replacing text with `preg_replace()`, 557–559
 - TrueType fonts, 533–534
 - wrapping lines of, 96–98
 - text, finding within strings**
 - `strbrk()` for finding set of characters, 81
 - `strpos()` and `strrpos()` for locating, 79–80
 - `strstr()` for finding, 79
 - `substr_count()` for finding number of occurrences, 80–81
 - text, replacing within strings**
 - overview of, 81
 - `str_replace()` for replacing all occurrences, 81–82
 - `strtr()` for translating characters, 87
 - `substr_replace()` for replacing portion of a string, 82–87
 - text editors**
 - `createFile()`, 333–334
 - `displayEditForm()`, 332
 - `displayFileList()`, 331–332
 - `displayPageHeader()`, 334
 - examining editor code, 330
 - main logic of editor code, 330–331
 - overview of, 325–326
 - `saveFile()`, 332–333
 - script for, 326–328
 - testing, 328–330
 - throwing exceptions, 653–654**
 - thumbnails, of images, 528–531**
 - time. See dates and times**
 - `time()`, for getting current date and time, 472
 - time zone settings, 21–22**
 - time-related file functions, 299**
 - TIMESTAMP data type, 371**
 - timestamps, 472**
 - creating, 473–475
 - extracting date and time values from, 475–477
 - UNIX timestamps and, 276, 299
 - `track_errors`, **configuration directive, 728**
 - transparency, of images, 525–527**
 - `trigger_error()`, **646–647**
 - `trim()`, **for trimming strings, 95**
 - true color, 508**
 - TrueType fonts**
 - example displaying text with TrueType font, 534–535
 - for text used with images, 533–534
 - `try...catch` **blocks, for catching exceptions, 654**
 - TSV (tab-separated-value), 309**
 - type casting, 38–40**
 - type specifier, 90–91**
- ## U
- Ubuntu**
 - PHP installation on, 12–14
 - starting MySQL server on UNIX/Linux OSs, 349
 - testing PEAR package manager on, 442–443
 - `ucfirst()`, **for case conversion, 88**
 - `ucwords()`, **for case conversion, 88**
 - Unicode**
 - MySQL and PHP 6 and, 365
 - support for, 8–9
 - working with in PHP 6, 302–303
 - `uninstall` **command, PEAR packages, 447**
 - UNIQUE keys, compared with primary keys, 370**
 - unit testing**
 - automating code testing with PHPUnit, 666–667
 - example use of PHPUnit tests, 667–670
 - overview of, 666
 - Universal Coordinated Time (UTC), 371, 472**
 - UNIX OSs, 298. See also Ubuntu**
 - UNIX timestamps**
 - setting cookies and, 276
 - time-related file functions, 299
 - `unlink()`, **for deleting files, 316**
 - `unserialize()`, **for storing objects as strings, 213**
 - `_unset()`, **overloading method, 192**
 - `update()`, **for adding update methods, 428**
 - UPDATE statement**
 - `update()` and, 430
 - updating data in MySQL tables, 357
 - updating records, using PHP scripts, 406**

upload_max_filesize, configuration directive

`upload_max_filesize`, **configuration directive**, 736

`upload_tmp_dir`, **configuration directive**, 736

uploading files. See **file uploads**

uppercase, converting strings to/from, 87–89

`urlencode()`, 269

URLs

- encoding, 269
- redirection, 264–265
- wrappers and, 736

user information, cookies for remembering, 278–282

user input

- handling Unicode in, 564
- regular expressions for validating, 564

user login system, 292–295

user preferences, cookies for storing, 274

`user.ini` **file, setting configuration directives**, 719–720

usernames, authentication methods for, 418

`usort()`, **for sorting array of words**, 157

UTC (Universal Coordinated Time), 371, 472

UTF-8

- MySQL and PHP 6 and, 365
- working with Unicode files in PHP 6, 302

V

valid documents, XML, 575–576, 578–579

`Validate` **package, PEAR**, 643

validation, of form input, 564–570

validation rules, HTML_QuickForm package, 460–462

values

- functions returning, 148–150
- static variables for preserving, 153–154

`VARCHAR` **data type, MySQL**, 346–347

variable scope

- global variables, 152–153
- local variables, 151–152
- overview of, 150–151
- static variables for preserving values, 153–154

variables. See **also arrays**

- changing variable type, 38
- converting an array to a list of variables, 137–138
- creating, 34–35
- defined, 33
- functions, 144–145

- naming, 34
- overview of, 33–34
- parameters as. See **parameters**
- storing PHP variables in HTML forms, 249–256
- testing variable type, 36–38

`variables_order`, **configuration directive**, 731

vector images, 509

`view_member.php` **script**

- member manager application, 431–437
- member record viewer application, 399–400

`view_members.php` **script**

- member manager application, 431
- member record viewer application, 395–398

visibility

- of methods, 174
- of properties, 169–170

visitors

- example detecting visitor's browser, 448–449
- example remembering user information, 278–282

W

Wamp (Windows, Apache, MySQL, and PHP), 15

WampServer

- starting MySQL server on Windows OSs, 349–350
- testing PEAR package manager on Windows OSs, 444–446

watermarks, applying to images, 523–525

WBMP image format, 509

Web browsers, detecting visitor's browser, 448–449

Web forms

- creating, 242
- creating interactive, 242–249
- creating with `HTML_QuickForm` package, 455
- file upload forms. See **file uploads**

Web pages

- creating for members' area of Web site, 424–426
- dynamic vs. static, 3

Web servers

- getting information from, 494–497
- options for running PHP, 23
- testing, 20–21
- working with files and directories stored on, 297

Web services, XML and, 573

well formed document, XML, 575–577

WHERE clause, for limiting number of rows returned, 372–373

while loops, 59–60

whitelisting, 642–643

Windows, Apache, MySQL, and PHP (Wamp), 15

Windows OSs

- directories and, 298
- ODBC and, 763
- PHP installation on, 15–17
- testing PEAR package manager on, 444–446

words, sorting by length, 155–158

wordwrap(), for wrapping lines of text, 96–98

wrapper string class, object overloading and, 188–191

wrapping lines of text, 96–98

writing/reading. See reading/writing

X

x coordinates, coordinate systems for drawing, 508

XHTML

- creating XHTML document, 581
- `displayPageHeader()`, 334
- DTDs, 579–580

XML (eXtensible Markup Language), 583

- adding elements to existing document, 600–602
- changing nodes and attributes, 603–605
- converting between Simple XML and DOM objects, 612
- creating XHTML document, 581
- creating XML documents using DOM, 595–599
- creating XML documents using SimpleXML, 610–612
- document structure, 575–576
- DOM and, 590
- DTDs for XHTML, 579–580
- elements and attributes, 578
- event handlers for parsing, 583
- exercise solutions, 714–716
- exercises, 616–617
- manipulating XML documents, 599
- moving elements, 605–606
- namespaces, 581
- overview of, 573–574
- parse errors, 584–585

- parsing XML document, 584
 - parsing XML file using DOM, 585–589
 - parts of well formed document, 576–577
 - reading XML documents using DOM, 591–594
 - reading XML documents using SimpleXML, 608–610
 - reading XML documents with XML Parser, 582
 - referencing DTDs, 580–581
 - removing elements from existing document, 602–603
 - SimpleXML and, 606–608
 - summary, 615–616
 - syntax rules, 577–578
 - valid documents, 578–579
 - what it is, 574–575
 - writing/manipulating XML documents, 589–590
 - XSL and XSLT and, 613–615

XML declaration, 574, 576

XML Parser

- event handlers for, 583
- example parsing XML file, 585–589
- parse errors, 584–585
- for reading XML documents, 582

XML Schema definitions (XSDs)

- major parts of XML documents, 575–576
- valid XML documents and, 578–579

xml_parse(), 584–585

xml_parser_create(), 582

xmlns (XML Namespace) attribute, 581

Xor (^), PHP bitwise operator, 43

xor, PHP logical operator, 46

XSDs (XML Schema definitions)

- major parts of XML documents, 575–576
- valid XML documents and, 578–579

XSL (Extensible Stylesheet Language), 613–615

XSLT (XSL Transformations), 613–615

XSLTProcessor class, 613

XSS (cross-site scripting attacks), 642

Y

y coordinates, coordinate systems for drawing, 508

Z

Zeus server, 23