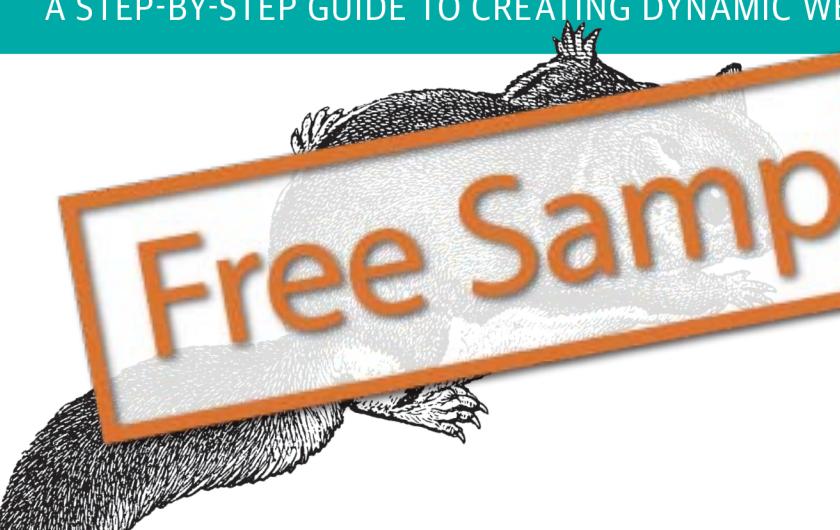


Learning PHP, MySQL, JavaScript, CSS & HTML5

A STEP-BY-STEP GUIDE TO CREATING DYNAMIC WEBSITES



Free Sampler

Robin Nixon

Learning PHP, MySQL, JavaScript, CSS & HTML5

Build interactive, data-driven websites with the potent combination of open-source technologies and web standards, even if you only have basic HTML knowledge. With this popular hands-on guide, you'll tackle dynamic web programming with the help of today's core technologies: PHP, MySQL, JavaScript, CSS, and HTML5.

Explore each technology separately and learn how to use them together—and pick up valuable web programming practices along the way. At the end of the book, you'll put everything together to build a fully functional social networking site.

- Learn PHP essentials and the basics of object-oriented programming
- Discover MySQL, from database structure to complex queries
- Create dynamic PHP web pages that integrate forms and other HTML features
- Manage cookies and sessions, and maintain a high level of security
- Work with JavaScript fundamentals, from functions and event handling to accessing the Document Object Model
- Use Ajax calls to turn your website into a highly dynamic environment
- Pick up CSS basics for formatting and styling your pages
- Learn HTML5 features, including geolocation, audio, video, and canvas
- Get up to speed on all of today's main web development technologies

“This is a great beginner's book that introduces several crucial web developer languages. It's a quick-paced, easy-to-read, information-packed book that will soon have you creating dynamically driven web sites, including a basic social networking site.”

—Albert Wiersch
developer of CSE HTML Validator

Robin Nixon, an IT journalist who's written hundreds of articles and several books on computing, has developed numerous websites using open source technologies, and is the developer who created the earliest pop-up windows. Robin has worked with and written about computers since the early 1980s.

WEB DEVELOPMENT

US \$49.99 CAN \$52.99

ISBN: 978-1-491-94946-7



9 781491 949467



Twitter: @oreillymedia
facebook.com/oreilly

Want to read more?

You can [buy this book](#) at [oreilly.com](#)
in print and ebook format.

Buy 2 books, get the 3rd FREE!

Use discount code: OPC10

All orders over \$29.95 qualify for **free shipping** within the US.

It's also available at your favorite book retailer,
including the iBookstore, the [Android Marketplace](#),
and [Amazon.com](#).



O'REILLY®

Spreading the knowledge of innovators

[oreilly.com](#)

Learning PHP, MySQL, JavaScript, CSS & HTML5, Third Edition

by Robin Nixon

Copyright © 2014 Robin Nixon. All rights reserved.

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (<http://my.safaribooksonline.com>). For more information, contact our corporate/institutional sales department: 800-998-9938 or corporate@oreilly.com.

Editor: Andy Oram

Production Editor: Kristen Brown

Copyeditor: Rachel Monaghan

Proofreader: Jasmine Kwityn

Indexer: Lucie Haskins

Cover Designer: Karen Montgomery

Interior Designer: David Futato

Illustrator: Rebecca Demarest

June 2014: Third Edition

Revision History for the Third Edition:

2014-05-19: First release

See <http://oreilly.com/catalog/errata.csp?isbn=9781491949467> for release details.

Nutshell Handbook, the Nutshell Handbook logo, and the O'Reilly logo are registered trademarks of O'Reilly Media, Inc. *Learning PHP, MySQL, JavaScript, CSS & HTML5*, the image of sugar gliders, and related trade dress are trademarks of O'Reilly Media, Inc.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and O'Reilly Media, Inc. was aware of a trademark claim, the designations have been printed in caps or initial caps.

While every precaution has been taken in the preparation of this book, the publisher and author assume no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.

ISBN: 978-1-491-94946-7

[LSI]

Table of Contents

Preface.....	xxi
1. Introduction to Dynamic Web Content.....	1
HTTP and HTML: Berners-Lee's Basics	2
The Request/Response Procedure	2
The Benefits of PHP, MySQL, JavaScript, CSS, and HTML5	5
Using PHP	6
Using MySQL	7
Using JavaScript	8
Using CSS	9
And Then There's HTML5	10
The Apache Web Server	11
About Open Source	12
Bringing It All Together	12
Questions	14
2. Setting Up a Development Server.....	15
What Is a WAMP, MAMP, or LAMP?	16
Installing a WAMP on Windows	16
Testing the Installation	28
Alternative WAMPs	31
Installing a MAMP on Mac OS X	31
Configuring MySQL	35
Ensuring MySQL Starts on Booting	36
Testing the Installation	36
Installing a LAMP on Linux	38
Working Remotely	38
Logging In	38
Using FTP	39

Using a Program Editor	40
Using an IDE	41
Questions	43
3. Introduction to PHP.....	45
Incorporating PHP Within HTML	45
This Book's Examples	47
The Structure of PHP	48
Using Comments	48
Basic Syntax	49
Variables	50
Operators	55
Variable Assignment	57
Multiple-Line Commands	60
Variable Typing	62
Constants	63
Predefined Constants	64
The Difference Between the echo and print Commands	64
Functions	65
Variable Scope	66
Questions	71
4. Expressions and Control Flow in PHP.....	73
Expressions	73
TRUE or FALSE?	73
Literals and Variables	75
Operators	76
Operator Precedence	77
Associativity	78
Relational Operators	80
Conditionals	84
The if Statement	84
The else Statement	85
The elseif Statement	87
The switch Statement	88
The ? Operator	91
Looping	92
while Loops	93
do ... while Loops	94
for Loops	95
Breaking Out of a Loop	97
The continue Statement	98

Implicit and Explicit Casting	98
PHP Dynamic Linking	99
Dynamic Linking in Action	100
Questions	101
5. PHP Functions and Objects.....	103
PHP Functions	104
Defining a Function	106
Returning a Value	106
Returning an Array	108
Passing by Reference	108
Returning Global Variables	110
Recap of Variable Scope	111
Including and Requiring Files	111
The include Statement	111
Using include_once	112
Using require and require_once	112
PHP Version Compatibility	113
PHP Objects	113
Terminology	114
Declaring a Class	115
Creating an Object	116
Accessing Objects	116
Cloning Objects	118
Constructors	119
PHP 5 Destructors	120
Writing Methods	120
Static Methods in PHP 5	121
Declaring Properties	122
Declaring Constants	122
Property and Method Scope in PHP 5	123
Static Properties and Methods	124
Inheritance	125
Questions	129
6. PHP Arrays.....	131
Basic Access	131
Numerically Indexed Arrays	131
Associative Arrays	133
Assignment Using the array Keyword	134
The foreach ... as Loop	135
Multidimensional Arrays	137

Using Array Functions	140
is_array	140
count	140
sort	140
shuffle	141
explode	141
extract	142
compact	143
reset	144
end	144
Questions	144
7. Practical PHP.....	147
Using printf	147
Precision Setting	148
String Padding	150
Using sprintf	151
Date and Time Functions	151
Date Constants	154
Using checkdate	154
File Handling	155
Checking Whether a File Exists	155
Creating a File	155
Reading from Files	157
Copying Files	158
Moving a File	158
Deleting a File	158
Updating Files	159
Locking Files for Multiple Accesses	160
Reading an Entire File	162
Uploading Files	162
System Calls	167
XHTML or HTML5?	169
Questions	169
8. Introduction to MySQL.....	171
MySQL Basics	171
Summary of Database Terms	172
Accessing MySQL via the Command Line	172
Starting the Command-Line Interface	173
Using the Command-Line Interface	177
MySQL Commands	178

Data Types	183
Indexes	192
Creating an Index	192
Querying a MySQL Database	198
Joining Tables Together	207
Using Logical Operators	209
MySQL Functions	209
Accessing MySQL via phpMyAdmin	210
Using phpMyAdmin	214
Questions	214
9. Mastering MySQL.....	217
Database Design	217
Primary Keys: The Keys to Relational Databases	218
Normalization	219
First Normal Form	220
Second Normal Form	222
Third Normal Form	224
When Not to Use Normalization	226
Relationships	227
One-to-One	227
One-to-Many	228
Many-to-Many	229
Databases and Anonymity	230
Transactions	230
Transaction Storage Engines	231
Using BEGIN	232
Using COMMIT	232
Using ROLLBACK	233
Using EXPLAIN	234
Backing Up and Restoring	235
Using mysqldump	235
Creating a Backup File	237
Restoring from a Backup File	239
Dumping Data in CSV Format	239
Planning Your Backups	240
Questions	240
10. Accessing MySQL Using PHP.....	241
Querying a MySQL Database with PHP	241
The Process	242
Creating a Login File	242

Connecting to MySQL	243
A Practical Example	248
The \$_POST Array	251
Deleting a Record	252
Displaying the Form	252
Querying the Database	253
Running the Program	254
Practical MySQL	255
Creating a Table	255
Describing a Table	256
Dropping a Table	257
Adding Data	257
Retrieving Data	258
Updating Data	259
Deleting Data	260
Using AUTO_INCREMENT	260
Performing Additional Queries	262
Preventing SQL Injection	263
Using Placeholders	265
Preventing HTML Injection	266
Questions	268
11. Using the mysqli Extension.	269
Querying a MySQL Database with mysqli	269
Creating a Login File	269
Connecting to MySQL	270
A Practical Example	274
Using mysqli Procedurally	276
Questions	277
12. Form Handling.	279
Building Forms	279
Retrieving Submitted Data	281
register_globals: An Old Solution Hangs On	282
Default Values	283
Input Types	284
Sanitizing Input	291
An Example Program	292
What's New in HTML5?	295
The autocomplete Attribute	295
The autofocus Attribute	295
The placeholder Attribute	296

The required Attribute	296
Override Attributes	296
The width and height Attributes	297
Features Awaiting Full Implementation	297
The form Attribute	297
The list Attribute	297
The min and max Attributes	298
The step Attribute	298
The color Input Type	298
The number and range Input Types	298
Date and time Pickers	298
Questions	299
13. Cookies, Sessions, and Authentication.....	301
Using Cookies in PHP	301
Setting a Cookie	303
Accessing a Cookie	304
Destroying a Cookie	304
HTTP Authentication	304
Storing Usernames and Passwords	307
Salting	308
Using Sessions	312
Starting a Session	312
Ending a Session	315
Setting a Timeout	317
Session Security	317
Questions	320
14. Exploring JavaScript.....	323
JavaScript and HTML Text	324
Using Scripts Within a Document Head	325
Older and Nonstandard Browsers	325
Including JavaScript Files	326
Debugging JavaScript Errors	327
Using Comments	329
Semicolons	329
Variables	330
String Variables	330
Numeric Variables	330
Arrays	331
Operators	332
Arithmetic Operators	332

Assignment Operators	332
Comparison Operators	333
Logical Operators	333
Variable Incrementing and Decrementing	334
String Concatenation	334
Escaping Characters	334
Variable Typing	335
Functions	336
Global Variables	336
Local Variables	336
The Document Object Model	338
But It's Not That Simple	340
Another Use for the \$ Symbol	340
Using the DOM	341
Questions	342
15. Expressions and Control Flow in JavaScript.....	343
Expressions	343
Literals and Variables	344
Operators	345
Operator Precedence	346
Associativity	346
Relational Operators	347
The with Statement	350
Using onerror	351
Using try ... catch	352
Conditionals	353
The if Statement	353
The else Statement	353
The switch statement	354
The ? Operator	355
Looping	356
while Loops	356
do ... while Loops	357
for Loops	357
Breaking Out of a Loop	358
The continue Statement	359
Explicit Casting	360
Questions	360
16. JavaScript Functions, Objects, and Arrays.....	363
JavaScript Functions	363

Defining a Function	363
The arguments Array	364
Returning a Value	365
Returning an Array	367
JavaScript Objects	368
Declaring a Class	368
Creating an Object	369
Accessing Objects	370
The prototype Keyword	370
JavaScript Arrays	372
Numeric Arrays	373
Associative Arrays	374
Multidimensional Arrays	375
Using Array Methods	376
Questions	380
17. JavaScript and PHP Validation and Error Handling.....	381
Validating User Input with JavaScript	381
The validate.html Document (Part One)	382
The validate.html Document (Part Two)	384
Regular Expressions	387
Using Regular Expressions in JavaScript	395
Using Regular Expressions in PHP	396
Redisplaying a Form After PHP Validation	397
Questions	403
18. Using Ajax.....	405
What Is Ajax?	405
Using XMLHttpRequest	406
Your First Ajax Program	408
Using GET Instead of POST	413
Sending XML Requests	415
Using Frameworks for Ajax	420
Questions	421
19. Introduction to CSS.....	423
Importing a Style Sheet	424
Importing CSS from Within HTML	424
Embedded Style Settings	425
Using IDs	425
Using Classes	425

Using Semicolons	426
CSS Rules	426
Multiple Assignments	426
Using Comments	427
Style Types	428
Default Styles	428
User Styles	428
External Style Sheets	429
Internal Styles	429
Inline Styles	430
CSS Selectors	430
The Type Selector	430
The Descendant Selector	430
The Child Selector	431
The ID Selector	432
The Class Selector	433
The Attribute Selector	434
The Universal Selector	434
Selecting by Group	435
The CSS Cascade	435
Style Sheet Creators	436
Style Sheet Methods	436
Style Sheet Selectors	437
Calculating Specificity	437
The Difference Between Div and Span Elements	439
Measurements	440
Fonts and Typography	442
font-family	442
font-style	443
font-size	443
font-weight	444
Managing Text Styles	444
Decoration	445
Spacing	445
Alignment	446
Transformation	446
Indenting	446
CSS Colors	447
Short Color Strings	447
Gradients	448
Positioning Elements	449
Absolute Positioning	449

Relative Positioning	450
Fixed Positioning	450
Pseudo-Classes	452
Shorthand Rules	454
The Box Model and Layout	454
Setting Margins	455
Applying Borders	457
Adjusting Padding	458
Object Contents	459
Questions	459
20. Advanced CSS with CSS3.....	461
Attribute Selectors	461
The ^ Operator	462
The \$ Operator	462
The * Operator	463
The box-sizing Property	463
CSS3 Backgrounds	463
The background-clip Property	464
The background-origin Property	465
The background-size Property	466
Multiple Backgrounds	467
CSS3 Borders	469
The border-color Property	469
The border-radius Property	469
Box Shadows	472
Element Overflow	473
Multicolumn Layout	473
Colors and Opacity	475
HSL Colors	475
HSLA Colors	476
RGB Colors	476
RGBA Colors	477
The opacity Property	477
Text Effects	477
The text-shadow Property	477
The text-overflow Property	478
The word-wrap Property	479
Web Fonts	479
Google Web Fonts	480
Transformations	481
3D Transformations	483

Transitions	483
Properties to Transition	484
Transition Duration	484
Transition Delay	484
Transition Timing	485
Shorthand Syntax	485
Questions	487
21. Accessing CSS from JavaScript.....	489
Revisiting the getElementById Function	489
The O function	489
The S Function	490
The C Function	491
Including the Functions	492
Accessing CSS Properties from JavaScript	493
Some Common Properties	494
Other Properties	495
Inline JavaScript	497
The this Keyword	497
Attaching Events to Objects in a Script	498
Attaching to Other Events	499
Adding New Elements	500
Removing Elements	501
Alternatives to Adding and Removing Elements	502
Using Interrupts	503
Using setTimeout	503
Canceling a Timeout	504
Using setInterval	504
Using Interrupts for Animation	506
Questions	508
22. Introduction to HTML5.....	509
The Canvas	510
Geolocation	511
Audio and Video	513
Forms	514
Local Storage	515
Web Workers	515
Web Applications	515
Microdata	516
Summary	516

Questions	516
23. The HTML5 Canvas.....	517
Creating and Accessing a Canvas	517
The toDataURL Function	519
Specifying an Image Type	521
The fillRect Method	521
The clearRect Method	521
The strokeRect Method	522
Combining These Commands	522
The createLinearGradient Method	523
The addColorStop Method in Detail	525
The createRadialGradient Method	526
Using Patterns for Fills	528
Writing Text to the Canvas	530
The strokeText Method	530
The textBaseline Property	531
The font Property	531
The textAlign Property	531
The fillText Method	532
The measureText Method	533
Drawing Lines	533
The lineWidth Property	533
The lineCap and lineJoin Properties	533
The miterLimit Property	535
Using Paths	536
The moveTo and LineTo Methods	536
The stroke Method	537
The rect Method	537
Filling Areas	537
The clip Method	539
The isPointInPath Method	542
Working with Curves	543
The arc Method	543
The arcTo Method	546
The quadraticCurveTo Method	547
The bezierCurveTo Method	548
Manipulating Images	549
The drawImage Method	549
Resizing an Image	550
Selecting an Image Area	550
Copying from a Canvas	552

Adding Shadows	552
Editing at the Pixel Level	554
The getImageData Method	554
The data Array	555
The putImageData Method	557
The createImageData Method	557
Advanced Graphical Effects	558
The globalCompositeOperation Property	558
The globalAlpha Property	561
Transformations	561
The scale Method	561
The save and restore Methods	562
The rotate Method	562
The translate Method	564
The transform Method	565
The setTransform Method	567
Summary	567
Questions	567
24. HTML5 Audio and Video.....	569
About Codecs	570
The <audio> Element	571
Supporting Non-HTML5 Browsers	573
The <video> Element	574
The Video Codecs	575
Supporting Older Browsers	578
Summary	580
Questions	580
25. Other HTML5 Features.....	581
Geolocation and the GPS Service	581
Other Location Methods	582
Geolocation and HTML5	583
Local Storage	586
Using Local Storage	587
The localStorage Object	587
Web Workers	589
Offline Web Applications	591
Drag and Drop	593
Cross Document Messaging	595
Microdata	598
Other HTML5 Tags	601

Summary	601
Questions	602
26. Bringing It All Together.....	603
Designing a Social Networking Site	603
On the Website	604
functions.php	604
The Functions	605
header.php	607
setup.php	608
index.php	610
signup.php	610
Checking for Username Availability	611
Logging In	611
checkuser.php	614
login.php	615
profile.php	617
Adding the “About Me” Text	618
Adding a Profile Image	618
Processing the Image	618
Displaying the Current Profile	619
members.php	622
Viewing a User’s Profile	622
Adding and Dropping Friends	622
Listing All Members	622
friends.php	625
messages.php	628
logout.php	631
styles.css	632
javascript.js	636
A. Solutions to the Chapter Questions.....	639
B. Online Resources.....	659
C. MySQL’s FULLTEXT Stopwords.....	663
D. MySQL Functions.....	667
Index.....	677

CHAPTER 1

Introduction to Dynamic Web Content

The World Wide Web is a constantly evolving network that has already traveled far beyond its conception in the early 1990s, when it was created to solve a specific problem. State-of-the-art experiments at CERN (the European Laboratory for Particle Physics—now best known as the operator of the Large Hadron Collider) were producing incredible amounts of data—so much that the data was proving unwieldy to distribute to the participating scientists who were spread out across the world.

At this time, the Internet was already in place, with several hundred thousand computers connected to it, so Tim Berners-Lee (a CERN fellow) devised a method of navigating between them using a hyperlinking framework, which came to be known as Hypertext Transfer Protocol, or HTTP. He also created a markup language called HTML, or Hypertext Markup Language. To bring these together, he wrote the first web browser and web server, tools that we now take for granted.

But back then, the concept was revolutionary. The most connectivity so far experienced by at-home modem users was dialing up and connecting to a bulletin board that was hosted by a single computer, where you could communicate and swap data only with other users of that service. Consequently, you needed to be a member of many bulletin board systems in order to effectively communicate electronically with your colleagues and friends.

But Berners-Lee changed all that in one fell swoop, and by the mid-1990s, there were three major graphical web browsers competing for the attention of five million users. It soon became obvious, though, that something was missing. Yes, pages of text and graphics with hyperlinks to take you to other pages was a brilliant concept, but the results didn't reflect the instantaneous potential of computers and the Internet to meet the particular needs of each user with dynamically changing content. Using the Web was a very dry and plain experience, even if we did now have scrolling text and animated GIFs!

Shopping carts, search engines, and social networks have clearly altered how we use the Web. In this chapter, we'll take a brief look at the various components that make up the Web, and the software that helps make it a rich and dynamic experience.



It is necessary to start using some acronyms more or less right away. I have tried to clearly explain them before proceeding. But don't worry too much about what they stand for or what these names mean, because the details will all become clear as you read on.

HTTP and HTML: Berners-Lee's Basics

HTTP is a communication standard governing the requests and responses that take place between the browser running on the end user's computer and the web server. The server's job is to accept a request from the client and attempt to reply to it in a meaningful way, usually by serving up a requested web page—that's why the term *server* is used. The natural counterpart to a server is a *client*, so that term is applied both to the web browser and the computer on which it's running.

Between the client and the server there can be several other devices, such as routers, proxies, gateways, and so on. They serve different roles in ensuring that the requests and responses are correctly transferred between the client and server. Typically, they use the Internet to send this information.

A web server can usually handle multiple simultaneous connections and—when not communicating with a client—spends its time listening for an incoming connection. When one arrives, the server sends back a response to confirm its receipt.

The Request/Response Procedure

At its most basic level, the request/response process consists of a web browser asking the web server to send it a web page and the server sending back the page. The browser then takes care of displaying the page (see [Figure 1-1](#)).

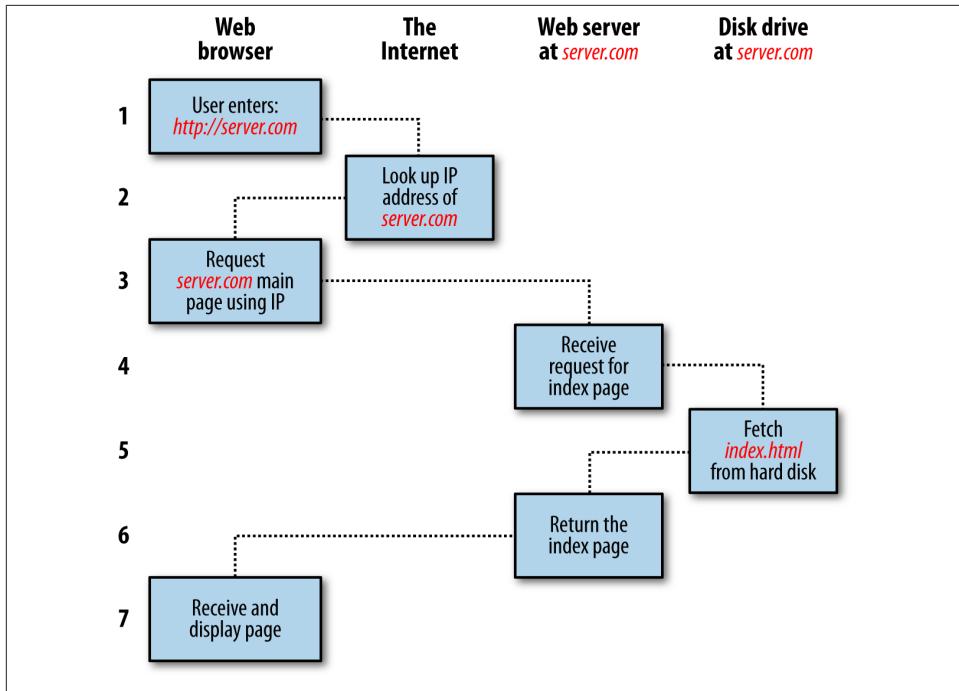


Figure 1-1. The basic client/server request/response sequence

Each step in the request and response sequence is as follows:

1. You enter *http://server.com* into your browser's address bar.
2. Your browser looks up the IP address for *server.com*.
3. Your browser issues a request for the home page at *server.com*.
4. The request crosses the Internet and arrives at the *server.com* web server.
5. The web server, having received the request, looks for the web page on its hard disk.
6. The web page is retrieved by the server and returned to the browser.
7. Your browser displays the web page.

For an average web page, this process takes place once for each object within the page: a graphic, an embedded video or Flash file, and even a CSS template.

In step 2, notice that the browser looked up the IP address of *server.com*. Every machine attached to the Internet has an IP address—your computer included. But we generally access web servers by name, such as *google.com*. As you probably know, the browser consults an additional Internet service called the Domain Name Service (DNS) to find its associated IP address and then uses it to communicate with the computer.

For dynamic web pages, the procedure is a little more involved, because it may bring both PHP and MySQL into the mix (see [Figure 1-2](#)).

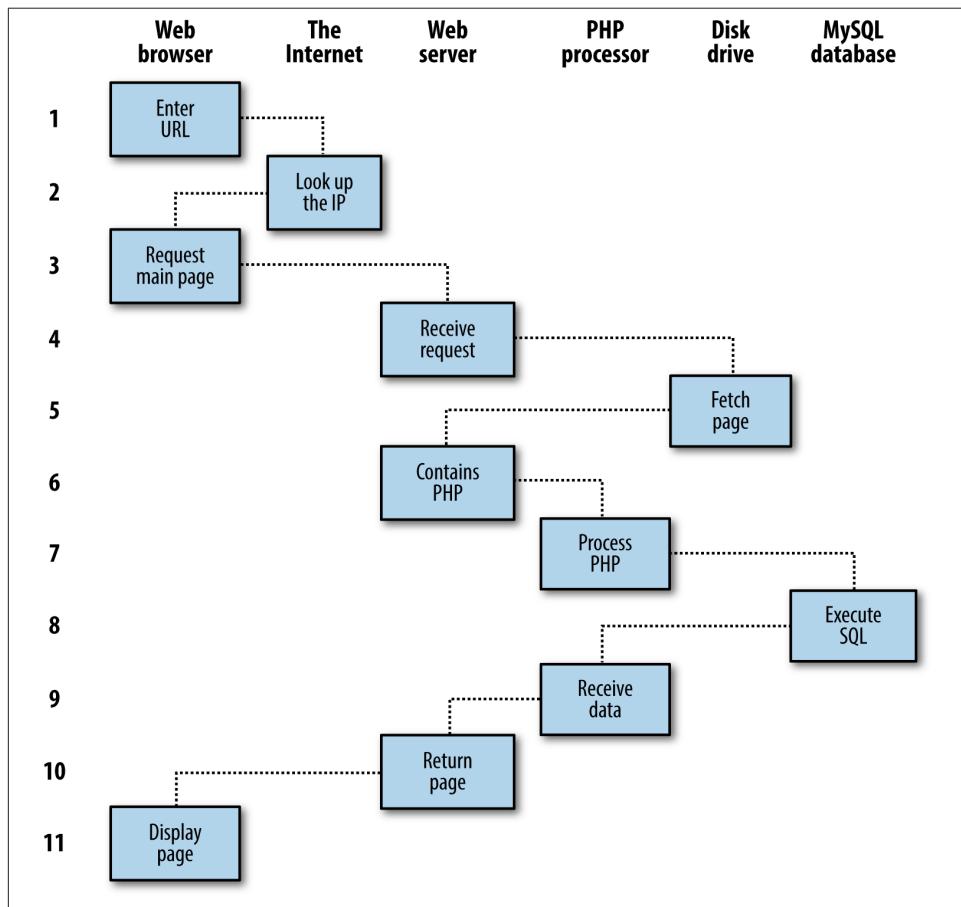


Figure 1-2. A dynamic client/server request/response sequence

Here are the steps for a dynamic client/server request/response sequence:

1. You enter `http://server.com` into your browser's address bar.
2. Your browser looks up the IP address for `server.com`.
3. Your browser issues a request to that address for the web server's home page.
4. The request crosses the Internet and arrives at the `server.com` web server.
5. The web server, having received the request, fetches the home page from its hard disk.

6. With the home page now in memory, the web server notices that it is a file incorporating PHP scripting and passes the page to the PHP interpreter.
7. The PHP interpreter executes the PHP code.
8. Some of the PHP contains MySQL statements, which the PHP interpreter now passes to the MySQL database engine.
9. The MySQL database returns the results of the statements back to the PHP interpreter.
10. The PHP interpreter returns the results of the executed PHP code, along with the results from the MySQL database, to the web server.
11. The web server returns the page to the requesting client, which displays it.

Although it's helpful to be aware of this process so that you know how the three elements work together, in practice you don't really need to concern yourself with these details, because they all happen automatically.

HTML pages returned to the browser in each example may well contain JavaScript, which will be interpreted locally by the client, and which could initiate another request—the same way embedded objects such as images would.

The Benefits of PHP, MySQL, JavaScript, CSS, and HTML5

At the start of this chapter, I introduced the world of Web 1.0, but it wasn't long before the rush was on to create Web 1.1, with the development of such browser enhancements as Java, JavaScript, JScript (Microsoft's slight variant of JavaScript), and ActiveX. On the server side, progress was being made on the Common Gateway Interface (CGI) using scripting languages such as Perl (an alternative to the PHP language) and *server-side scripting*—inserting the contents of one file (or the output of a system call) into another one dynamically.

Once the dust had settled, three main technologies stood head and shoulders above the others. Although Perl was still a popular scripting language with a strong following, PHP's simplicity and built-in links to the MySQL database program had earned it more than double the number of users. And JavaScript, which had become an essential part of the equation for dynamically manipulating CSS (Cascading Style Sheets) and HTML, now took on the even more muscular task of handling the client side of the Ajax process. Under Ajax, web pages perform data handling and send requests to web servers in the background—without the web user being aware that this is going on.

No doubt the symbiotic nature of PHP and MySQL helped propel them both forward, but what attracted developers to them in the first place? The simple answer has to be the ease with which you can use them to quickly create dynamic elements on websites. MySQL is a fast and powerful, yet easy-to-use, database system that offers just about

anything a website would need in order to find and serve up data to browsers. When PHP allies with MySQL to store and retrieve this data, you have the fundamental parts required for the development of social networking sites and the beginnings of Web 2.0.

And when you bring JavaScript and CSS into the mix too, you have a recipe for building highly dynamic and interactive websites.

Using PHP

With PHP, it's a simple matter to embed dynamic activity in web pages. When you give pages the *.php* extension, they have instant access to the scripting language. From a developer's point of view, all you have to do is write code such as the following:

```
<?php  
    echo " Today is " . date("l") . ". ";  
?>
```

Here's the latest news.

The opening `<?php` tells the web server to allow the PHP program to interpret all the following code up to the `?>` tag. Outside of this construct, everything is sent to the client as direct HTML. So the text `Here's the latest news.` is simply output to the browser; within the PHP tags, the built-in `date` function displays the current day of the week according to the server's system time.

The final output of the two parts looks like this:

Today is Wednesday. Here's the latest news.

PHP is a flexible language, and some people prefer to place the PHP construct directly next to PHP code, like this:

```
Today is <?php echo date("l"); ?>. Here's the latest news.
```

There are also other ways of formatting and outputting information, which I'll explain in the chapters on PHP. The point is that with PHP, web developers have a scripting language that, although not as fast as compiling your code in C or a similar language, is incredibly speedy and also integrates seamlessly with HTML markup.



If you intend to enter the PHP examples in this book to work along with me, you must remember to add `<?php` in front and `?>` after them to ensure that the PHP interpreter processes them. To facilitate this, you may wish to prepare a file called *example.php* with those tags in place.

Using PHP, you have unlimited control over your web server. Whether you need to modify HTML on the fly, process a credit card, add user details to a database, or fetch

information from a third-party website, you can do it all from within the same PHP files in which the HTML itself resides.

Using MySQL

Of course, there's not much point to being able to change HTML output dynamically unless you also have a means to track the changes that users make as they use your website. In the early days of the Web, many sites used “flat” text files to store data such as usernames and passwords. But this approach could cause problems if the file wasn't correctly locked against corruption from multiple simultaneous accesses. Also, a flat file can get only so big before it becomes unwieldy to manage—not to mention the difficulty of trying to merge files and perform complex searches in any kind of reasonable time.

That's where relational databases with structured querying become essential. And MySQL, being free to use and installed on vast numbers of Internet web servers, rises superbly to the occasion. It is a robust and exceptionally fast database management system that uses English-like commands.

The highest level of MySQL structure is a database, within which you can have one or more tables that contain your data. For example, let's suppose you are working on a table called `users`, within which you have created columns for `surname`, `firstname`, and `email`, and you now wish to add another user. One command that you might use to do this is:

```
INSERT INTO users VALUES('Smith', 'John', 'jsmith@mysite.com');
```

Of course, as mentioned earlier, you will have issued other commands to create the database and table and to set up all the correct fields, but the `INSERT` command here shows how simple it can be to add new data to a database. The `INSERT` command is an example of SQL (Structured Query Language), a language designed in the early 1970s and reminiscent of one of the oldest programming languages, COBOL. It is well suited, however, to database queries, which is why it is still in use after all this time.

It's equally easy to look up data. Let's assume that you have an email address for a user and need to look up that person's name. To do this, you could issue a MySQL query such as:

```
SELECT surname,firstname FROM users WHERE email='jsmith@mysite.com';
```

MySQL will then return `Smith`, `John` and any other pairs of names that may be associated with that email address in the database.

As you'd expect, there's quite a bit more that you can do with MySQL than just simple `INSERT` and `SELECT` commands. For example, you can join multiple tables according to various criteria, ask for results in a variety of orders, make partial matches when you know only part of the string that you are searching for, return only the *n*th result, and a lot more.

Using PHP, you can make all these calls directly to MySQL without having to run the MySQL program yourself or use its command-line interface. This means you can save the results in arrays for processing and perform multiple lookups, each dependent on the results returned from earlier ones, to drill right down to the item of data you need.

For even more power, as you'll see later, there are additional functions built right into MySQL that you can call up for common operations and extra speed.

Using JavaScript

The oldest of the three core technologies in this book, JavaScript, was created to enable scripting access to all the elements of an HTML document. In other words, it provides a means for dynamic user interaction such as checking email address validity in input forms, displaying prompts such as “Did you really mean that?”, and so on (note, however, that it cannot be relied upon for security, which should always be performed on the web server).

Combined with CSS (see the following section), JavaScript is the power behind dynamic web pages that change in front of your eyes rather than when a new page is returned by the server.

However, JavaScript can also be tricky to use, due to some major differences in the ways different browser designers have chosen to implement it. This mainly came about when some manufacturers tried to put additional functionality into their browsers at the expense of compatibility with their rivals.

Thankfully, the developers have mostly now come to their senses and have realized the need for full compatibility with one another, so they don't have to write multi-exception code. But there remain millions of legacy browsers that will be in use for a good many years to come. Luckily, there are solutions for the incompatibility problems, and later in this book we'll look at libraries and techniques that enable you to safely ignore these differences.

For now, let's take a quick look at how you can use basic JavaScript, accepted by all browsers:

```
<script type="text/javascript">
  document.write("Today is " + Date() );
</script>
```

This code snippet tells the web browser to interpret everything within the `script` tags as JavaScript, which the browser then does by writing the text `Today is` to the current document, along with the date, by using the JavaScript function `Date`. The result will look something like this:

Today is Sun Jan 01 2017 01:23:45



Unless you need to specify an exact version of JavaScript, you can normally omit the `type="text/javascript"` and just use `<script>` to start the interpretation of the JavaScript.

As previously mentioned, JavaScript was originally developed to offer dynamic control over the various elements within an HTML document, and that is still its main use. But more and more, JavaScript is being used for Ajax. This is a term for the process of accessing the web server in the background. (It originally meant “Asynchronous JavaScript and XML,” but that phrase is already a bit outdated.)

Ajax is the main process behind what is now known as Web 2.0 (a term popularized by Tim O'Reilly, the founder and CEO of this book's publishing company), in which web pages have started to resemble standalone programs, because they don't have to be reloaded in their entirety. Instead, a quick Ajax call can pull in and update a single element on a web page, such as changing your photograph on a social networking site or replacing a button that you click with the answer to a question. This subject is fully covered in [Chapter 18](#).

Using CSS

With the emergence of the CSS3 standard in recent years, CSS now offers a level of dynamic interactivity previously supported only by JavaScript. For example, not only can you style any HTML element to change its dimensions, colors, borders, spacing, and so on, but now you can also add animated transitions and transformations to your web pages, using only a few lines of CSS.

Using CSS can be as simple as inserting a few rules between `<style>` and `</style>` tags in the head of a web page, like this:

```
<style>
  p {
    text-align:justify;
    font-family:Helvetica;
  }
</style>
```

These rules will change the default text alignment of the `<p>` tag so that paragraphs contained in it will be fully justified and will use the Helvetica font.

As you'll learn in [Chapter 19](#), there are many different ways you can lay out CSS rules, and you can also include them directly within tags or save a set of rules to an external file to be loaded in separately. This flexibility not only lets you style your HTML precisely, but it can also, for example, provide built-in hover functionality to animate objects as the mouse passes over them. You will also learn how to access all of an element's CSS properties from JavaScript as well as HTML.

And Then There's HTML5

As useful as all these additions to the web standards became, they were not enough for ever more ambitious developers. For example, there was still no simple way to manipulate graphics in a web browser without resorting to plug-ins such as Flash. And the same went for inserting audio and video into web pages. Plus, several annoying inconsistencies had crept into HTML during its evolution.

So, to clear all this up and take the Internet beyond Web 2.0 and into its next iteration, a new standard for HTML was created to address all these shortcomings. It was called HTML5 and it began development as long ago as 2004, when the first draft was drawn up by the Mozilla Foundation and Opera Software (developers of two popular web browsers). But it wasn't until the start of 2013 that the final draft was submitted to the World Wide Web Consortium (W3C), the international governing body for web standards.

With nine years for it to develop, you might think that would be the end of the specification, but that's not how things work on the Internet. Although websites come and go at great speed, the underlying software is developed slowly and carefully, and so the stable recommendation for HTML5 is not expected until after this edition of the book has been published—in late 2014. And then guess what? Work will move on to versions 5.1 and higher, beginning in 2015. It's a never-ending cycle of development.

However, while HTML5.1 is planned to bring some handy improvements (mainly to the canvas), basic HTML5 is the new standard web developers now need to work to, and it will remain in place for many years to come. So learning everything you can about it now will stand you in very good stead.

There's actually a great deal of new stuff in HTML (and quite a few things that have been changed or removed), but in summary, here's what you get:

Markup

Including new elements such as `<nav>` and `<footer>`, and deprecated elements like `` and `<center>`.

New APIs

For example, the `<canvas>` element for writing and drawing on a graphics canvas, `<audio>` and `<video>` elements, offline web apps, microdata, and local storage.

Applications

Including two new rendering technologies: MathML (Math Markup Language) for displaying mathematical formulae and SVG (Scalable Vector Graphics) for creating graphical elements outside of the new `<canvas>` element. However, MathML and SVG are somewhat specialist, and are so feature-packed they would need a book of their own, so I don't cover them here.

All these things (and more) are covered in detail starting in [Chapter 22](#).



One of the little things I like about the HTML5 specification is that XHTML syntax is no longer required for self-closing elements. In the past you could display a line break using the `
` element. Then, to ensure future compatibility with XHTML (the planned replacement for HTML that never happened), this was changed to `
`, in which a closing `/` character was added (because all elements were expected to include a closing tag featuring this character). But now things have gone full circle, and you can use either version of these element types. So, for the sake of brevity and fewer keystrokes, in this book I have reverted to the former style of `
`, `<hr>`, and so on.

The Apache Web Server

In addition to PHP, MySQL, JavaScript, CSS, and HTML5, there's actually a sixth hero in the dynamic Web: the web server. In the case of this book, that means the Apache web server. We've discussed a little of what a web server does during the HTTP server/client exchange, but it actually does much more behind the scenes.

For example, Apache doesn't serve up just HTML files—it handles a wide range of files from images and Flash files to MP3 audio files, RSS (Really Simple Syndication) feeds, and so on. To do this, each element a web client encounters in an HTML page is also requested from the server, which then serves it up.

But these objects don't have to be static files such as GIF images. They can all be generated by programs such as PHP scripts. That's right: PHP can even create images and other files for you, either on the fly or in advance to serve up later.

To do this, you normally have modules either precompiled into Apache or PHP or called up at runtime. One such module is the GD (Graphics Draw) library, which PHP uses to create and handle graphics.

Apache also supports a huge range of modules of its own. In addition to the PHP module, the most important for your purposes as a web programmer are the modules that handle security. Other examples are the Rewrite module, which enables the web server to handle a varying range of URL types and rewrite them to its own internal requirements, and the Proxy module, which you can use to serve up often-requested pages from a cache to ease the load on the server.

Later in the book, you'll see how to actually use some of these modules to enhance the features provided by the three core technologies.

About Open Source

Whether the open source quality of these technologies is the reason they are so popular has often been debated, but PHP, MySQL, and Apache *are* the three most commonly used tools in their categories.

What can be said definitively, though, is that their being open source means that they have been developed in the community by teams of programmers writing the features they themselves want and need, with the original code available for all to see and change. Bugs can be found and security breaches can be prevented before they happen.

There's another benefit: all these programs are free to use. There's no worrying about having to purchase additional licenses if you have to scale up your website and add more servers. And you don't need to check the budget before deciding whether to upgrade to the latest versions of these products.

Bringing It All Together

The real beauty of PHP, MySQL, JavaScript, CSS, and HTML5 is the wonderful way in which they all work together to produce dynamic web content: PHP handles all the main work on the web server, MySQL manages all the data, and the combination of CSS and JavaScript looks after web page presentation. JavaScript can also talk with your PHP code on the web server whenever it needs to update something (either on the server or on the web page). And with the powerful new features in HTML5, such as the canvas, audio and video, and geolocation, you can make your web pages highly dynamic, interactive, and multimedia packed.

Without using program code, let's summarize the contents of this chapter by looking at the process of combining some of these technologies into an everyday Ajax feature that many websites use: checking whether a desired username already exists on the site when a user is signing up for a new account. A good example of this can be seen with Gmail (see [Figure 1-3](#)).

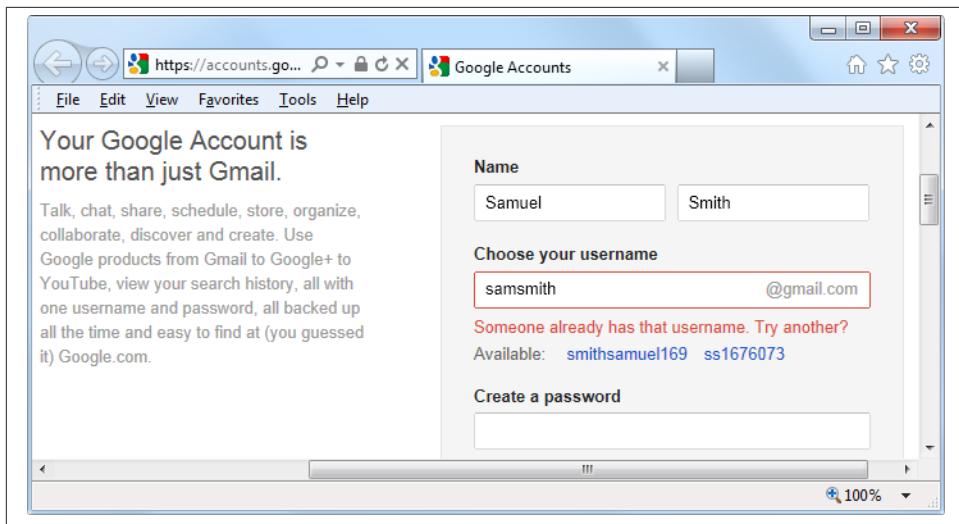


Figure 1-3. Gmail uses Ajax to check the availability of usernames

The steps involved in this Ajax process would be similar to the following:

1. The server outputs the HTML to create the web form, which asks for the necessary details, such as username, first name, last name, and email address.
2. At the same time, the server attaches some JavaScript to the HTML to monitor the username input box and check for two things: (a) whether some text has been typed into it, and (b) whether the input has been deselected because the user has clicked on another input box.
3. Once the text has been entered and the field deselected, in the background the JavaScript code passes the username that was entered back to a PHP script on the web server and awaits a response.
4. The web server looks up the username and replies back to the JavaScript regarding whether that name has already been taken.
5. The JavaScript then places an indication next to the username input box to show whether the name is one available to the user—perhaps a green checkmark or a red cross graphic, along with some text.
6. If the username is not available and the user still submits the form, the JavaScript interrupts the submission and reemphasizes (perhaps with a larger graphic and/or an alert box) that the user needs to choose another username.
7. Optionally, an improved version of this process could even look at the username requested by the user and suggest an alternative that is currently available.

All of this takes place quietly in the background and makes for a comfortable and seamless user experience. Without Ajax, the entire form would have to be submitted to the server, which would then send back HTML, highlighting any mistakes. It would be a workable solution, but nowhere near as tidy or pleasurable as on-the-fly form field processing.

Ajax can be used for a lot more than simple input verification and processing, though; we'll explore many additional things that you can do with it in the Ajax chapters later in this book.

In this chapter, you have read a good introduction to the core technologies of PHP, MySQL, JavaScript, CSS, and HTML5 (as well as Apache), and have learned how they work together. In [Chapter 2](#), we'll look at how you can install your own web development server on which to practice everything that you will be learning.

Questions

1. What four components (at the minimum) are needed to create a fully dynamic web page?
2. What does HTML stand for?
3. Why does the name MySQL contain the letters *SQL*?
4. PHP and JavaScript are both programming languages that generate dynamic results for web pages. What is their main difference, and why would you use both of them?
5. What does CSS stand for?
6. List three major new elements introduced in HTML5.
7. If you encounter a bug (which is rare) in one of the open source tools, how do you think you could get it fixed?

See “[Chapter 1 Answers](#)” on page 639 in [Appendix A](#) for the answers to these questions.

O'Reilly Ebooks—Your bookshelf on your devices!



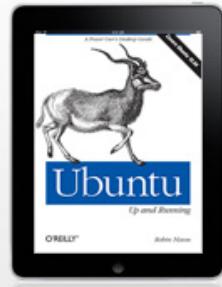
PDF



ePub



Mobi



APK



DAISY

When you buy an ebook through oreilly.com you get lifetime access to the book, and whenever possible we provide it to you in five, DRM-free file formats—PDF, .epub, Kindle-compatible .mobi, Android .apk, and DAISY—that you can use on the devices of your choice. Our ebook files are fully searchable, and you can cut-and-paste and print them. We also alert you when we've updated the files with corrections and additions.

Learn more at ebooks.oreilly.com

You can also purchase O'Reilly ebooks through the iBookstore, the [Android Marketplace](http://Android.Marketplace), and Amazon.com.

O'REILLY®

Spreading the knowledge of innovators

oreilly.com