A Step-by-Step Guide to Creating Dynamic Websites

2nd Edition

Learning

PHP, MySQL, JavaScript, & CSS

O'REILLY®

Robin Nixon

SECOND EDITION

Learning PHP, MySQL, JavaScript, and CSS

Robin Nixon

O'REILLY®
Beijing • Cambridge • Farnham • Köln • Sebastopol • Tokyo

Learning PHP, MySQL, JavaScript, and CSS, Second Edition
by Robin Nixon

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: Iris Febres
Copyeditor: Rachel Head
Proofreader: Kiel Van Horn

Indexer: Ellen Troutman Zaig
Cover Designer: Karen Montgomery
Interior Designer: David Futato
Illustrator: Robert Romano

August 2012: Second Edition.

Revision History for the Second Edition:
2012-08-10 First release
See *http://oreilly.com/catalog/errata.csp?isbn=9781449319267* for release details.

ISBN: 978-1-449-31926-7

[LSI]

1344626556

Table of Contents

Preface ..... **xvii**

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, and CSS 5
Using PHP 5
Using MySQL 6
Using JavaScript 7
Using CSS 9
The Apache Web Server 9
About Open Source 10
Bringing It All Together 10
Test Your Knowledge 12

2. Setting Up a Development Server **13**
What Is a WAMP, MAMP, or LAMP? 13
Installing a WAMP on Windows 14
Testing the Installation 22
Alternative WAMPs 24
Installing a MAMP on OS X 24
Configuring MySQL 25
Testing the Installation 28
Installing a LAMP on Linux 31
Working Remotely 31
Logging In 32
Using FTP 32
Using a Program Editor 33
Using an IDE 34
Test Your Knowledge 36

3. Introduction to PHP 37
Incorporating PHP Within HTML 37
Calling the PHP Parser 38
This Book's Examples 39
The Structure of PHP 40
Using Comments 40
Basic Syntax 41
Understanding Variables 42
Operators 46
Variable Assignment 49
Multiple-Line Commands 51
Variable Typing 54
Constants 54
The Difference Between the echo and print Commands 56
Functions 56
Variable Scope 57
Test Your Knowledge 62

4. Expressions and Control Flow in PHP 65
Expressions 65
Literals and Variables 66
Operators 67
Operator Precedence 68
Associativity 70
Relational Operators 71
Conditionals 75
The if Statement 75
The else Statement 76
The elseif Statement 78
The switch Statement 78
The ? Operator 81
Looping 82
while Loops 83
do...while Loops 85
for Loops 85
Breaking Out of a Loop 87
The continue Statement 88
Implicit and Explicit Casting 89
PHP Dynamic Linking 90
Dynamic Linking in Action 90
Test Your Knowledge 91

5. PHP Functions and Objects 93
PHP Functions 94
Defining a Function 95
Returning a Value 96
Returning an Array 97
Passing by Reference 98
Returning Global Variables 99
Recap of Variable Scope 100
Including and Requiring Files 100
The include Statement 100
Using include_once 101
Using require and require_once 101
PHP Version Compatibility 102
PHP Objects 102
Terminology 103
Declaring a Class 104
Creating an Object 105
Accessing Objects 105
Constructors 108
Writing Methods 109
Declaring Properties 110
Declaring Constants 111
Property and Method Scope in PHP 5 112
Inheritance 114
Test Your Knowledge 117

6. PHP Arrays 119
Basic Access 119
Numerically Indexed Arrays 119
Associative Arrays 121
Assignment Using the array Keyword 122
The foreach...as Loop 122
Multidimensional Arrays 124
Using Array Functions 127
is_array 127
count 127
sort 128
shuffle 128
explode 128
extract 129
compact 130
reset 131
end 131

Test Your Knowledge 132

7. Practical PHP 133
Using printf 133
Precision Setting 134
String Padding 136
Using sprintf 137
Date and Time Functions 137
Date Constants 140
Using checkdate 140
File Handling 141
Checking Whether a File Exists 141
Creating a File 141
Reading from Files 143
Copying Files 143
Moving a File 144
Deleting a File 144
Updating Files 145
Locking Files for Multiple Accesses 146
Reading an Entire File 147
Uploading Files 148
System Calls 153
XHTML 154
The Benefits of XHTML 155
XHTML Versions 155
What's Different? 155
HTML 4.01 Document Types 156
The HTML5 Document Type 157
XHTML 1.0 Document Types 157
XHTML Validation 158
Test Your Knowledge 159

8. Introduction to MySQL 161
MySQL Basics 161
Summary of Database Terms 162
Accessing MySQL via the Command Line 162
Starting the Command-Line Interface 163
Using the Command-Line Interface 166
MySQL Commands 168
Data Types 172
Indexes 181
Creating an Index 182
Querying a MySQL Database 187

Joining Tables Together 196
Using Logical Operators 198
MySQL Functions 199
Accessing MySQL via phpMyAdmin 199
Windows Users 199
OS X Users 201
Linux Users 201
Using phpMyAdmin 201
Test Your Knowledge 202

9. Mastering MySQL 205
Database Design 205
Primary Keys: The Keys to Relational Databases 206
Normalization 207
First Normal Form 208
Second Normal Form 210
Third Normal Form 212
When Not to Use Normalization 214
Relationships 215
One-to-One 215
One-to-Many 216
Many-to-Many 216
Databases and Anonymity 218
Transactions 218
Transaction Storage Engines 219
Using BEGIN 220
Using COMMIT 220
Using ROLLBACK 221
Using EXPLAIN 221
Backing Up and Restoring 223
Using mysqldump 223
Creating a Backup File 224
Restoring from a Backup File 226
Dumping Data in CSV Format 226
Planning Your Backups 227
Test Your Knowledge 227

10. Accessing MySQL Using PHP 229
Querying a MySQL Database with PHP 229
The Process 229
Creating a Login File 230
Connecting to MySQL 231
A Practical Example 236

The $_POST Array 238
Deleting a Record 239
Displaying the Form 239
Querying the Database 240
Running the Program 241
Practical MySQL 242
Creating a Table 242
Describing a Table 243
Dropping a Table 244
Adding Data 244
Retrieving Data 245
Updating Data 246
Deleting Data 246
Using AUTO_INCREMENT 247
Performing Additional Queries 248
Preventing SQL Injection 249
Preventing HTML Injection 252
Test Your Knowledge 254

11. Form Handling 255
Building Forms 255
Retrieving Submitted Data 256
register_globals: An Old Solution Hangs On 258
Default Values 258
Input Types 259
Sanitizing Input 266
An Example Program 267
Test Your Knowledge 270

12. Cookies, Sessions, and Authentication 271
Using Cookies in PHP 271
Setting a Cookie 273
Accessing a Cookie 273
Destroying a Cookie 274
HTTP Authentication 274
Storing Usernames and Passwords 277
Salting 277
Using Sessions 281
Starting a Session 281
Ending a Session 284
Session Security 286
Test Your Knowledge 289

13. Exploring JavaScript 291
JavaScript and HTML Text 291
Using Scripts Within a Document Head 293
Older and Nonstandard Browsers 293
Including JavaScript Files 294
Debugging JavaScript Errors 295
Using Comments 297
Semicolons 297
Variables 298
String Variables 298
Numeric Variables 298
Arrays 299
Operators 299
Arithmetic Operators 300
Assignment Operators 300
Comparison Operators 301
Logical Operators 301
Variable Incrementing and Decrementing 301
String Concatenation 302
Escaping Characters 302
Variable Typing 303
Functions 303
Global Variables 304
Local Variables 304
The Document Object Model (DOM) 305
But It's Not That Simple 307
Using the DOM 308
Test Your Knowledge 309

14. Expressions and Control Flow in JavaScript 311
Expressions 311
Literals and Variables 312
Operators 313
Operator Precedence 314
Associativity 314
Relational Operators 315
The with Statement 318
Using onerror 318
Using try...catch 319
Conditionals 320
The if Statement 320
The switch statement 321
The ? Operator 323

Looping 323
while Loops 323
do...while Loops 324
for Loops 324
Breaking Out of a Loop 325
The continue Statement 326
Explicit Casting 327
Test Your Knowledge 327

15. JavaScript Functions, Objects, and Arrays 329
JavaScript Functions 329
Defining a Function 329
Returning a Value 331
Returning an Array 333
JavaScript Objects 333
Declaring a Class 333
Creating an Object 335
Accessing Objects 335
The prototype Keyword 336
JavaScript Arrays 338
Numeric Arrays 338
Associative Arrays 339
Multidimensional Arrays 340
Using Array Methods 341
Test Your Knowledge 345

16. JavaScript and PHP Validation and Error Handling 347
Validating User Input with JavaScript 347
The validate.html Document (Part One) 348
The validate.html Document (Part Two) 350
Regular Expressions 353
Matching Through Metacharacters 353
Fuzzy Character Matching 354
Grouping Through Parentheses 355
Character Classes 355
Some More Complicated Examples 356
Summary of Metacharacters 359
General Modifiers 360
Using Regular Expressions in JavaScript 361
Using Regular Expressions in PHP 361
Redisplaying a Form After PHP Validation 362
Test Your Knowledge 367

17. Using Ajax 369
What Is Ajax? 369
Using XMLHttpRequest 370
Implementing Ajax via POST Requests 372
The readyState Property 374
The Server Half of the Ajax Process 375
Using GET Instead of POST 377
Sending XML Requests 379
About XML 381
Why Use XML? 383
Using Frameworks for Ajax 383
Test Your Knowledge 383

18. Introduction to CSS 385
Importing a Style Sheet 386
Importing a Style Sheet from Within HTML 386
Embedded Style Settings 387
Using IDs 387
Using Classes 387
CSS Rules 388
Using Semicolons 388
Multiple Assignments 388
Using Comments 389
Style Types 390
Default Styles 390
User Styles 390
External Style Sheets 390
Internal Styles 391
Inline Styles 391
CSS Selectors 392
The Type Selector 392
The Descendant Selector 392
The Child Selector 393
The Adjacent Sibling Selector 394
The ID Selector 395
The Class Selector 395
The Attribute Selector 396
The Universal Selector 396
Selecting by Group 397
The CSS Cascade 398
Style Sheet Creators 398
Style Sheet Methods 398
Style Sheet Selectors 399

The Difference Between and 401
Measurements 402
Fonts and Typography 404
font-family 404
font-style 405
font-size 406
font-weight 406
Managing Text Styles 407
Decoration 407
Spacing 407
Alignment 408
Transformation 408
Indenting 408
CSS Colors 408
Short Color Strings 409
Gradients 410
Positioning Elements 411
Absolute Positioning 411
Relative Positioning 412
Fixed Positioning 412
Comparing Positioning Types 412
Pseudoclasses 413
Pseudoelements 415
Shorthand Rules 416
The Box Model and Layout 416
Setting Margins 417
Applying Borders 418
Adjusting Padding 419
Object Contents 420
Test Your Knowledge 421

19. Advanced CSS with CSS3 423
Attribute Selectors 423
Matching Parts of Strings 424
The box-sizing Property 425
CSS3 Backgrounds 425
The background-clip Property 426
The background-origin Property 426
The background-size Property 428
Multiple Backgrounds 428
CSS3 Borders 430
The border-color Property 430
The border-radius Property 431

Box Shadows 434
Element Overflow 435
Multicolumn Layout 435
Colors and Opacity 436
HSL Colors 437
HSLA Colors 437
RGB Colors 438
RGBA Colors 438
The opacity Property 438
Text Effects 439
The text-shadow Property 439
The text-overflow Property 439
The word-wrap Property 440
Web Fonts 440
Google Web Fonts 441
Transformations 441
Transitions 444
Properties to Transition 444
Transition Duration 444
Transition Delay 444
Transition Timing 445
Shorthand Syntax 445
Test Your Knowledge 447

20. Accessing CSS from JavaScript 449
Revisiting the getElementById Function 449
The O Function 449
The S Function 450
The C Function 451
Including the Functions 452
Accessing CSS Properties from JavaScript 453
Some Common Properties 453
Other Properties 455
Inline JavaScript 456
The this Keyword 457
Attaching Events to Objects in a Script 457
Attaching to Other Events 458
Adding New Elements 459
Removing Elements 460
Alternatives to Adding and Removing Elements 461
Using Interrupts 462
Using setTimeout 462
Canceling a Timeout 463

Using setInterval 463
Using Interrupts for Animation 465
Test Your Knowledge 467

21. Bringing It All Together 469
Designing a Social Networking Site 469
On the Website 470
functions.php 470
The Functions 470
header.php 472
setup.php 474
index.php 475
signup.php 475
Checking for Username Availability 476
checkuser.php 478
login.php 479
profile.php 481
Adding the "About Me" Text 482
Adding a Profile Image 482
Processing the Image 482
Displaying the Current Profile 483
members.php 485
Viewing a User's Profile 486
Adding and Dropping Friends 486
Listing All Members 486
friends.php 488
messages.php 491
logout.php 493
styles.css 495

A. Solutions to the Chapter Questions 499

B. Online Resources 513

C. MySQL's FULLTEXT Stopwords 517

D. MySQL Functions 521

Index 533

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. Tim Berners-Lee (a CERN fellow) devised a method of navigating between them using a hyperlinking framework, which came to be known as the Hyper Text Transfer Protocol, or HTTP. He also created a markup language called HTML, or Hyper Text Markup Language. To bring these together, he wrote the first web browser and web server.

We now take these tools 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 request. 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).

These are the steps in the request and 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 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 server retrieves the web page and returns it 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

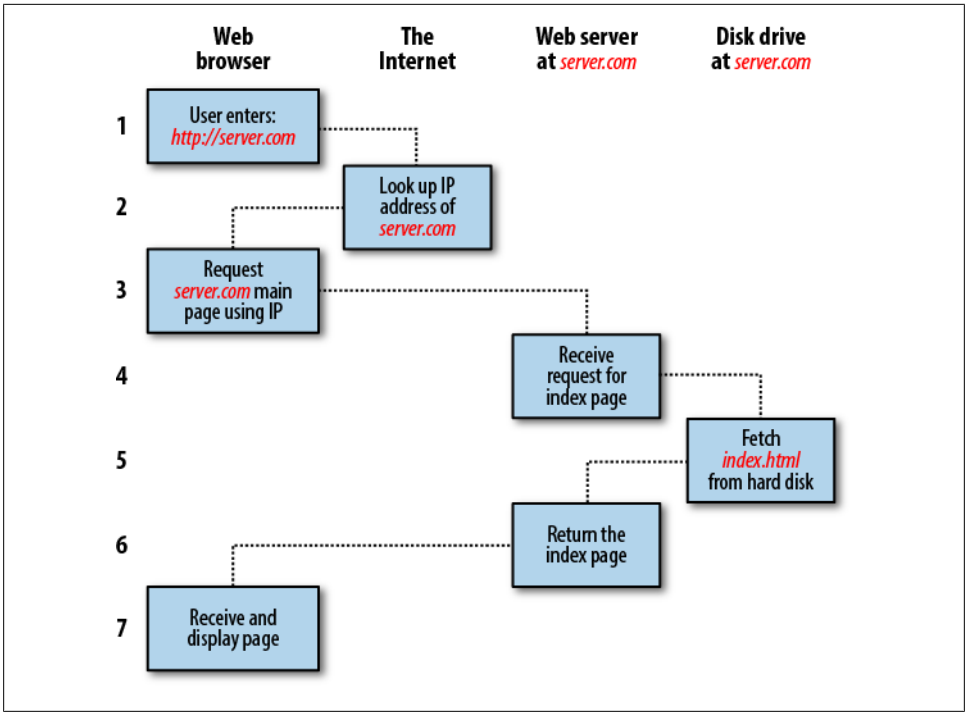

Figure 1-1. The basic client/server request/response sequence

consults an additional Internet service called the Domain Name Service (DNS) to find the server's associated IP address and then uses that 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).

Here are the steps:

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.

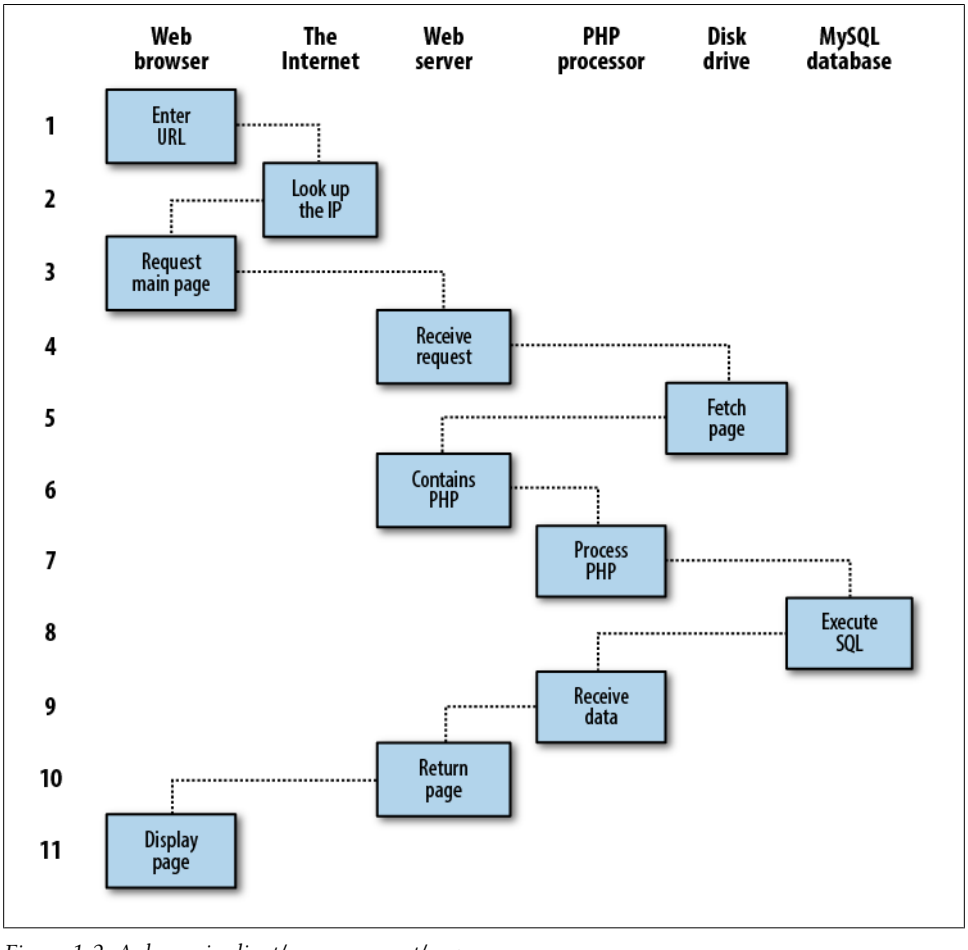

Figure 1-2. A dynamic client/server request/response sequence

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 it all happens 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, and CSS

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 (dynamically inserting the contents of one file—or the output of a system call—into another one).

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), now took on the even more muscular task of handling the client side of the Ajax process. Under Ajax (described in "Using JavaScript" on page 7), 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 these technologies 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 might 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 "Hello World. Today is ".date("l").". ";
?>

How are you?
```

The opening `<?php` tells the web server to allow the PHP program to interpret all the following code up to the `?>` command. Outside of this construct, everything is sent to the client as direct HTML. So, the text "How are you?" 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:

```
Hello World. Today is Wednesday. How are you?
```

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

```
Hello World. Today is <?php echo date("l"); ?>. How are you?
```

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 that also integrates seamlessly with HTML code.

If you intend to type in 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 a lot of 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 (which stands for 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 you 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 different 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 core technologies described 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 (although 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 the server returns a new page.

However, JavaScript can also be tricky to use, due to some major differences between 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 between their products, so web developers don't have to write multiexception 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 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("Hello World. Today is " + Date() );
</script>
```

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

```
Hello World. Today is Thu Jan 01 2015 01:23:45
```

It's worth knowing that 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 17.

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 justification 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 18, 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 lets you do more than style your HTML precisely; you will also see how it can (for example) provide built-in hover functionality to animate objects as the mouse pointer passes over them. You will also learn how to access all of an element's CSS properties from JavaScript as well as HTML.

The Apache Web Server

In addition to PHP, MySQL, JavaScript, and CSS, there's actually a fifth 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 more. 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 library (short for Graphics Draw), 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 core technologies we cover.

About Open Source

Whether or not being open source is the reason these technologies 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, though, is that 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, and CSS 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).

Without using program code, it's a good idea at this point to summarize the contents of this chapter by looking at the process of combining our core 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).

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

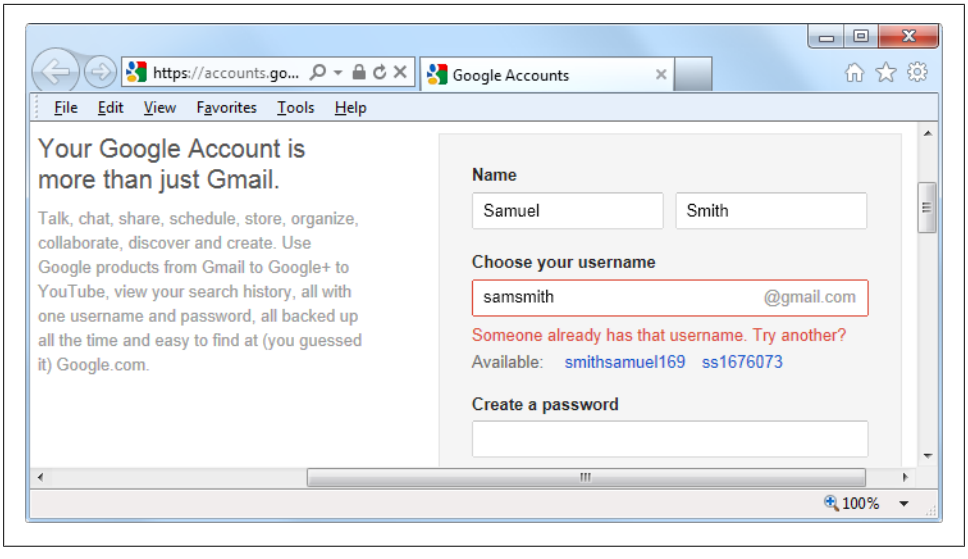

Figure 1-3. Gmail uses Ajax to check the availability of usernames

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: whether some text has been typed into it, and 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 typed in 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 check mark 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 using 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, and CSS (as well as Apache), and have learned how they work together with each other. 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. First, though, consider these questions.

Test Your Knowledge

1. What four components 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. 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 499 in Appendix A for the answers to these questions.