May 2018: Fifth Edition

Revision History for the Fifth Edition

- 2018-05-08: First Release


The O’Reilly logo is a registered trademark of O’Reilly Media, Inc. Learning PHP, MySQL & JavaScript, the cover image, and related trade dress are trademarks of O’Reilly Media, Inc.

While the publisher and the author have used good faith efforts to ensure that the information and instructions contained in this work are accurate, the publisher and the author disclaim all responsibility for errors or omissions, including without limitation responsibility for damages resulting from the use of or reliance on this work. Use of the information and instructions contained in this work is at your own risk. If any code samples or other technology this work contains or describes is subject to open source licenses or the intellectual property rights of others, it is your responsibility to ensure that your use thereof complies with such licenses and/or rights.

978-1-491-97891-7

[M]
Dedication

For Julie
Preface

The combination of PHP and MySQL is the most convenient approach to dynamic, database-driven web design, holding its own in the face of challenges from integrated frameworks—such as Ruby on Rails—that are harder to learn. Due to its open source roots (unlike the competing Microsoft .NET Framework), it is free to implement and is therefore an extremely popular option for web development.

Any would-be developer on a Unix/Linux or even a Windows/Apache platform will need to master these technologies. And, combined with the partner technologies of JavaScript, jQuery, CSS, and HTML5, you will be able to create websites of the caliber of industry standards like Facebook, Twitter, and Gmail.

Audience

This book is for people who wish to learn how to create effective and dynamic websites. This may include webmasters or graphic designers who are already creating static websites but wish to take their skills to the next level, as well as high school and college students, recent graduates, and self-taught individuals.

In fact, anyone ready to learn the fundamentals behind responsive web design will obtain a thorough grounding in the core technologies of PHP, MySQL, JavaScript, CSS, and HTML5, and you’ll learn the basics of the jQuery and jQuery Mobile libraries, too.
Assumptions This Book Makes

This book assumes that you have a basic understanding of HTML and can at least put together a simple, static website, but does not assume that you have any prior knowledge of PHP, MySQL, JavaScript, CSS, or HTML5—although if you do, your progress through the book will be even quicker.

Organization of This Book

The chapters in this book are written in a specific order, first introducing all of the core technologies it covers and then walking you through their installation on a web development server so that you will be ready to work through the examples.

In the first section, you will gain a grounding in the PHP programming language, covering the basics of syntax, arrays, functions, and object-oriented programming.

Then, with PHP under your belt, you will move on to an introduction to the MySQL database system, where you will learn everything from how MySQL databases are structured to how to generate complex queries.

After that, you will learn how you can combine PHP and MySQL to start creating your own dynamic web pages by integrating forms and other HTML features. You will then get down to the nitty-gritty practical aspects of PHP and MySQL development by learning a variety of useful functions and how to manage cookies and sessions, as well as how to maintain a high level of security.

In the next few chapters, you will gain a thorough grounding in JavaScript, from simple functions and event handling to accessing the Document Object Model, in-browser validation, and error handling. You’ll also get a comprehensive primer on using the popular jQuery library for JavaScript.

With an understanding of all three of these core technologies, you will then learn how to make behind-the-scenes Ajax calls and turn your websites into highly dynamic environments.
Next, you’ll spend two chapters learning all about using CSS to style and lay out your web pages, before discovering how the jQuery libraries can make your development job a great deal easier. You’ll then move on to the final section on the interactive features built into HTML5, including geolocation, audio, video, and the canvas. After this, you’ll put together everything you’ve learned in a complete set of programs that together constitute a fully functional social networking website.

Along the way, you’ll find plenty of advice on good programming practices and tips that can help you find and solve hard-to-detect programming errors. There are also plenty of links to websites containing further details on the topics covered.

### Supporting Books

Once you have learned to develop using PHP, MySQL, JavaScript, CSS, and HTML5, you will be ready to take your skills to the next level using the following O’Reilly reference books:

- *Dynamic HTML: The Definitive Reference* by Danny Goodman
- *PHP in a Nutshell* by Paul Hudson
- *MySQL in a Nutshell* by Russell Dyer
- *JavaScript: The Definitive Guide* by David Flanagan
- *CSS: The Definitive Guide* by Eric A. Meyer and Estelle Weyl
- *HTML5: The Missing Manual* by Matthew MacDonald

### Conventions Used in This Book

The following typographical conventions are used in this book:

Plain text

Indicates menu titles, options, and buttons.
Using Code Examples

Supplemental material (code examples, exercises, etc.) is available for download at http://lpmj.net.

This book is here to help you get your job done. In general, if example code is offered with this book, you may use it in your programs and documentation. You do not need to contact us for permission unless you’re reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require
permission. Selling or distributing a CD-ROM of examples from O’Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product’s documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: “Learning PHP, MySQL & JavaScript by Robin Nixon (O’Reilly). Copyright 2018 Robin Nixon, 978-1-491-97891-7.”

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at permissions@oreilly.com.

**O’Reilly Safari**

*Safari* (formerly Safari Books Online) is a membership-based training and reference platform for enterprise, government, educators, and individuals.


For more information, please visit [http://oreilly.com/safari](http://oreilly.com/safari).

**How to Contact Us**

Please address comments and questions concerning this book to the publisher:

O’Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
(800) 998-9938 (in the United States or Canada)
(707) 829-0515 (international or local)
(707) 829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at http://bit.ly/lpmjch_5e.

To comment or ask technical questions about this book, send email to bookquestions@oreilly.com.

For more information about our books, courses, conferences, and news, see our website at http://www.oreilly.com.

Find us on Facebook: http://facebook.com/oreilly
Follow us on Twitter: http://twitter.com/oreillymedia
Watch us on YouTube: http://www.youtube.com/oreillymedia

Acknowledgments

I would like to once again thank my editor, Andy Oram, and everyone who worked so hard on this book, including Jon Reid, Michal Špaček, and John Craig for their comprehensive technical reviews, Melanie Yarbrough for overseeing production, Rachel Head for copy editing, Rachel Monaghan for proofreading, Rebecca Demarest for illustrations, Judy McConville for creating the index, Karen Montgomery for the original sugar glider front cover design, Randy Comer for the latest book cover, and everyone else too numerous to name who submitted errata and offered suggestions for this new edition.
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, connecting several hundred thousand computers, 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 Hypertext Markup Language, or HTML. To bring these together, he wrote the first web browser and web server.

Today we 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 5 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 using it a rich and dynamic experience.

**NOTE**

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 become clear as you read on.

**HTTP and HTML: Berners-Lee’s Basics**

HTTP is a communication standard governing the requests and responses that are sent 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. Some of these in-between devices can also help speed up the internet by storing pages or information locally in what is called a cache, and then serving this content up to clients directly from the cache rather than fetching it all the way from the source server.

A web server can usually handle multiple simultaneous connections, and when not communicating with a client, it 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

The steps in the request and response sequence are as follows:
1. You enter \textit{http://server.com} into your browser’s address bar.

2. Your browser looks up the Internet Protocol (IP) address for \textit{server.com}.

3. Your browser issues a request for the home page at \textit{server.com}.

4. The request crosses the internet and arrives at the \textit{server.com} web server.

5. The web server, having received the request, looks for the web page on its disk.

6. The web server retrieves the page and returns it to the browser.

7. Your browser displays the web page.

For an average web page, this process also 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 looks up the IP address of \textit{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 \textit{google.com}. As you probably know, the browser consults an additional internet service called the Domain Name Service (DNS) to find the server’s 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. For instance, you may click on a picture of a raincoat. Then PHP will put together a request using the standard database language, SQL—many of whose commands you will learn in this book—and send the request to the MySQL server. The MySQL server will return information about the raincoat you selected, and the PHP code will wrap it all up in some HTML, which the server will send to your browser (see Figure 1-2).
The steps are 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 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 SQL statements, which the PHP interpreter now passes to the MySQL database engine.

9. The MySQL database returns the results of the statements 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.

The 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 running a local program) 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 Cascading Style Sheets (CSS) and HTML, now took on the even more muscular task of handling the client side of the asynchronous communication (exchanging data between a client and server after a web page has loaded). Using asynchronous communication, 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—especially as there is now a wide range of sophisticated frameworks of JavaScript functions you can call on to really speed up web development, such as the well-known jQuery, which is now probably the most common way programmers access asynchronous communication features.

**MariaDB: The MySQL Clone**
After Oracle purchased Sun Microsystems (the owners of MySQL), the community became wary that MySQL might not remain fully open source, so MariaDB was forked from it to keep it free under the GNU GPL. Development of MariaDB is led by some of the original developers of MySQL and it retains exceedingly close compatibility with MySQL. Therefore, you may well encounter MariaDB on some servers in place of MySQL—but not to worry, everything in this book works equally well on both MySQL and MariaDB, which is based on the same code base as MySQL Server 5.5. To all intents and purposes you can swap one with the other and notice no difference.

Anyway, as it turns out, many of the initial fears appear to have been allayed as MySQL remains open source, with Oracle simply charging for support and for editions that provide additional features such as geo-replication and automatic scaling. However, unlike MariaDB, MySQL is no longer community driven, so knowing that MariaDB will always be there if ever needed will keep many developers sleeping at night, and probably ensures that MySQL itself will remain open source.

**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
<?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
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 even more 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.

**NOTE**

If you intend to enter the PHP examples in this book into a program editor 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 information users provide to your website as they use it. 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 as follows:

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

You will previously have issued other commands to create the database and table and to set up all the correct fields, but the SQL INSERT command here shows how simple it can be to add new data to a database. SQL is a language designed in the early 1970s that is 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 the following:

```
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 combine related data sets to bring related pieces of information together, 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 directly access the MySQL command-line interface yourself. 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 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 to efficiently run common operations within MySQL, rather than creating them out of multiple PHP calls to MySQL.

**Using JavaScript**

The oldest of the three core technologies discussed 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 and displaying prompts such as “Did you really mean that?” (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 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 it is less necessary these days to have to optimize your code for different browsers. However, there remain millions of users using legacy browsers, and this will likely be the case 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 look at how to use basic JavaScript, accepted by all browsers:

```html
<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 does by writing the text `Today is` to the current document, along with the date, using the JavaScript function `Date`. The result will look something like this:

```
Today is Sun Jan 01 2017 01:23:45
```

NOTE

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 **asynchronous communication**, the process of accessing the web server in the background.

Asynchronous communication is what allows web pages to begin to resemble standalone programs, because they don’t have to be reloaded in
their entirety to display new content. Instead, an asynchronous 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.

Then, in Chapter 21, we take a good look at the jQuery framework, which you can use to save reinventing the wheel when you need fast, cross-browser code to manipulate your web pages. Of course, there are other frameworks available too, but jQuery is by far the most popular. Due to continuous maintenance, it is extremely reliable, and it’s a major tool in the utility kit of many seasoned web developers.

**Using CSS**

CSS is the crucial companion to HTML, ensuring that the HTML text and embedded images are laid out consistently and in a manner appropriate for the user’s screen. 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:

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

These rules change the default text alignment of the `<p>` tag so that paragraphs contained in it are fully justified and 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 not only lets you style your HTML precisely, but 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: *HTML5*. Its development began 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.

It has taken a few years for HTML5 to develop, but now we are at a very solid and stable version 5.1 (since 2016). It’s a never-ending cycle of development, though, and more functionality is sure to be built into it over time. Some of the best features in HTML5 for handling and displaying media include the `<audio>`, `<video>`, and `<canvas>` elements, which add sound, video, and advanced graphics. Everything you need to know about these and all other aspects of HTML5 is covered in detail starting in Chapter 23.
NOTE

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 `<br>` element. Then, to ensure future compatibility with XHTML (the planned replacement for HTML that never happened), this was changed to `<br />`, in which a closing `/` character was added (since 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 types of elements. So, for the sake of brevity and fewer keystrokes, in this book I have reverted to the former style of `<br>`, `<hr>`, and so on.

The Apache Web Server

In addition to PHP, MySQL, JavaScript, CSS, and HTML5, there’s 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 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. And 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 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 use some of these modules to enhance the features provided by the three core technologies.

**Handling Mobile Devices**

We are now firmly in a world of interconnected mobile computing devices, and the concept of developing websites solely for desktop computers has become rather dated. Instead, developers now aim to develop responsive websites and web apps that tailor themselves to the environment in which they find themselves running.

So, new in this edition, I show how you can easily create these types of products using just the technologies detailed in this book, along with the powerful jQuery Mobile library of responsive JavaScript functions. With it, you’ll be able to focus on the content and usability of your websites and web apps, knowing that how they display will be automatically optimized for a range of different computing devices—one less thing for you to worry about.

To demonstrate how to make full use of its power, the final chapter of this book creates a simple social networking example website, using jQuery Mobile to make it fully responsive and ensure it displays well on anything from a small mobile phone screen to a tablet or a desktop computer.

**About Open Source**

The technologies in this book are open source: anyone is allowed to read and change the code. Whether or not this status 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 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 quickly 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 (sometimes aided by jQuery or other frameworks), 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 asynchronous communication 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).
Gmail uses asynchronous communication to check the availability of usernames

The steps involved in this asynchronous process will 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: 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 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 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 asynchronous communication, 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.

Asynchronous communication 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 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?

8. Why is a framework such as jQuery Mobile so important for developing modern websites and web apps?

See “Chapter 1 Answers” in Appendix A for the answers to these questions.
Chapter 2. Setting Up a Development Server

If you wish to develop internet applications but don’t have your own development server, you will have to upload every modification you make to a server somewhere else on the web before you can test it.

Even on a fast broadband connection, this can still represent a significant slowdown in development time. On a local computer, however, testing can be as easy as saving an update (usually just a matter of clicking once on an icon) and then hitting the Refresh button in your browser.

Another advantage of a development server is that you don’t have to worry about embarrassing errors or security problems while you’re writing and testing, whereas you need to be aware of what people may see or do with your application when it’s on a public website. It’s best to iron everything out while you’re still on a home or small office system, presumably protected by firewalls and other safeguards.

Once you have your own development server, you’ll wonder how you ever managed without one, and it’s easy to set one up. Just follow the steps in the following sections, using the appropriate instructions for a PC, a Mac, or a Linux system.

In this chapter, we cover just the server side of the web experience, as described in Chapter 1. But to test the results of your work—particularly when we start using JavaScript, CSS, and HTML5 later in this book—you should ideally have an instance of every major web browser running on some system convenient to you. Whenever possible, the list of browsers should include at least Microsoft Edge, Mozilla Firefox, Opera, Safari, and Google Chrome. If you plan to ensure that your sites look good on mobile devices too, you should try to arrange access to a wide range of iOS and Android devices.
What Is a WAMP, MAMP, or LAMP?

WAMP, MAMP, and LAMP are abbreviations for “Windows, Apache, MySQL, and PHP,” “Mac, Apache, MySQL, and PHP,” and “Linux, Apache, MySQL, and PHP.” These abbreviations each describe a fully functioning setup used for developing dynamic internet web pages.

WAMPs, MAMPs, and LAMPs come in the form of packages that bind the bundled programs together so that you don’t have to install and set them up separately. This means you can simply download and install a single program and follow a few easy prompts to get your web development server up and running fast, with minimal hassle.

During installation, several default settings are created for you. The security configurations of such an installation will not be as tight as on a production web server, because it is optimized for local use. For these reasons, you should never install such a setup as a production server.

However, for developing and testing websites and applications, one of these installations should be entirely sufficient.

**WARNING**

If you choose not to go the WAMP/MAMP/LAMP route for building your own development system, you should know that downloading and integrating the various parts yourself can be very time-consuming and may require a lot of research in order to configure everything fully. But if you already have all the components installed and integrated with one another, they should work with the examples in this book.

Installing AMPPS on Windows

There are several available WAMP servers, each offering slightly different configurations. Of the various open source and free options, one of the best is AMPPS. You can download it by clicking the button on the website’s home page, shown in Figure 2-1.
I recommend that you always download the latest stable release (as I write this, it’s 3.8, which is about 128 MB in size). The various Windows, macOS, and Linux installers are listed on the download page.

![Figure 2-1. The AMPPS website](image)

**NOTE**

During the lifetime of this edition, some of the screens and options shown in the following walk-through may change. If so, just use your common sense to proceed in as similar a manner as possible to the sequence of actions described.

Once you’ve downloaded the installer, run it to bring up the window shown in **Figure 2-2**. Before arriving at that window, though, if you use an antivirus program or have User Account Control activated on Windows, you may first be shown one or more advisory notices, and will have to click Yes and/or OK to continue with the installation.
Click Next, after which you must accept the agreement. Click Next once again, and then once more to move past the information screen. You will now need to confirm the installation location. This will probably be suggested as something like the following, depending on the letter of your main hard drive, but you can change this if you wish:

C:\Program Files (x86)\Ampps

Once you have decided where to install AMPPS, click Next, choose a Start menu folder name, and click Next again. You can choose which icons you wish to install, as shown in Figure 2-3. On the screen that follows, click the Install button to start the process.
Installation will take a few minutes, after which you should see the completion screen in Figure 2-4, and you can click Finish.
The final thing you must do is install C++ Redistributable Visual Studio, if you haven’t already. Visual Studio is an environment in which you’ll be doing development work. A window will pop up to prompt you, as shown in Figure 2-5. Click Yes to start the installation or No if you are certain you already have it.

If you choose to go ahead and install, you will have to agree to the terms and conditions in the pop-up window that appears, and then click Install. Installation of this should be fairly fast. Click Close to finish.
Once AMPPS is installed, the control window shown in Figure 2-6 should appear at the bottom right of your desktop. You can also call up this window using the AMPPS application shortcut in the Start menu or on the desktop, if you allowed these icons to be created.

![AMPPS control window](image)

*Figure 2-6. The AMPPS control window*

Before proceeding, I recommend you acquaint yourself with the AMPPS documentation. Once you have digested this, should you still have an issue, there’s a Support link at the bottom of the control window that will take you to the AMPPS website, where you can open up a trouble ticket.
NOTE

You may notice that the default version of PHP in AMPPS is 5.6. In other sections of this book I detail some of the more important changes in PHP 7. If you wish to try them out for yourself, click the Options button (nine white boxes in a square) within the AMPPS control window, and then select Change PHP Version, whereupon a new menu will appear from which you can choose a version between 5.6 and 7.1.

Testing the Installation

The first thing to do at this point is verify that everything is working correctly. To do this, enter either of the following two URLs into the address bar of your browser:

localhost
127.0.0.1

This will call up an introductory screen, where you will have the opportunity to secure AMPPS by giving it a password (see Figure 2-7). I recommend you don’t check the box and just click the Submit button to proceed without setting a password.
Once this has been done you will be taken to the main control page at `localhost/ampps/` (from now on I will assume you are accessing AMPPS through `localhost` rather than `127.0.0.1`). From here you can configure and control all aspects of the AMPPS stack, so make a note of this for future reference, or perhaps set a bookmark in your browser.

Next, type the following to view the document root (described in the following section) of your new Apache web server:

```
localhost
```
This time, rather than seeing the initial screen about setting up security, you should see something similar to Figure 2-8.

![Figure 2-8. Viewing the document root](image)

**Accessing the Document Root (Windows)**

The *document root* is the directory that contains the main web documents for a domain. This directory is the one that the server uses when a basic URL without a path is typed into a browser, such as `http://yahoo.com` or, for your local server, `http://localhost`.

By default AMPPS will use the following location as the document root:

```
C:\Program Files (x86)\Ampps\www
```

To ensure that you have everything correctly configured, you should now create the obligatory “Hello World” file. So, create a small HTML file along the following lines using Windows Notepad or any other program or text editor, but not a rich word processor such as Microsoft Word (unless you save as plain text):

```
<html>
<body>
  <p>Hello, World!</p>
</body>
</html>
```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>A quick test</title>
  </head>
  <body>
    Hello World!
  </body>
</html>

Once you have typed this, save the file into the document root directory, using the filename test.html. If you are using Notepad, make sure that the value in the “Save as type” box is changed from “Text Documents (*.txt)” to “All Files (*.*)”.

You can now call this page up in your browser by entering the following URL in its address bar (see Figure 2-9):

localhost/test.html

![Screenshot of a web browser showing "A quick test" and "Hello World!"](image)

Figure 2-9. Your first web page

NOTE

Remember that serving a web page from the document root (or a subfolder) is different from loading one into a web browser from your computer’s filesystem. The former will ensure access to PHP, MySQL, and all the features of a web server, while the latter will simply load the file into the browser, which will do its best to display it but will be unable to process any PHP or other server instructions. So, you should generally run examples using the localhost preface from your browser’s address bar, unless you are certain that the file doesn’t rely on web server functionality.
Alternative WAMPs

When software is updated, it sometimes works differently from how you expect, and bugs can even be introduced. So, if you encounter difficulties that you cannot resolve in AMPPS, you may prefer to choose one of the other solutions available on the web.

You will still be able to make use of all the examples in this book, but you’ll have to follow the instructions supplied with each WAMP, which may not be as easy to follow as the preceding guide.

Here’s a selection of some of the best, in my opinion:

- EasyPHP
- XAMPP
- WAMPServer
- Glossword WAMP

**UPDATES TO AMPPS**

Over the life of this edition of the book, it is very likely that the developers of AMPPS will make improvements to the software, and therefore the installation screens and method of use may evolve over time, as may versions of Apache, PHP, or MySQL. So, please don’t assume something is wrong if the screens and operation look different. The AMPPS developers take every care to ensure it is easy to use, so just follow any prompts given, and refer to the documentation on the website.

**Installing AMPPS on macOS**

AMPPS is also available on macOS, and you can download it from the website, as shown previously in Figure 2-1 (as I write, the current version is 3.8 and its size is around 270 MB).

If your browser doesn’t open it automatically once it has downloaded, double-click the .dmg file, and then drag the AMPPS folder over to your Applications folder (see Figure 2-10).
Now open your *Applications* folder in the usual manner, and double-click the AMPPS program. If your security settings prevent the file being opened, hold down the Control key and click the icon once. A new window will pop up asking if you are sure you wish to open it. Click Open to do so. When the app starts you may have to enter your macOS password to proceed.

Once AMPPS is up and running, a control window similar to the one shown in Figure 2-6 will appear at the bottom left of your desktop.

**NOTE**

You may notice that the default version of PHP in AMPPS is 5.6. In other sections of this book I detail some of the more important changes in PHP 7. If you wish to try them out for yourself, click the Options button (nine white boxes in a square) within the AMPPS control window, and then select Change PHP Version, whereupon a new menu will appear in which you can choose a version between 5.6 and 7.1.

**Accessing the Document Root (macOS)**
By default, AMPPS will use the following location as the document root:

/Applications/Ampps/www

To ensure that you have everything correctly configured, you should now create the obligatory “Hello World” file. So, create a small HTML file along the following lines using the TextEdit program or any other program or text editor, but not a rich word processor such as Microsoft Word (unless you save as plain text):

```html
<html>
<head>
  <title>A quick test</title>
</head>
<body>
  Hello World!
</body>
</html>
```

Once you have typed this, save the file into the document root directory using the filename `test.html`.

You can now call this page up in your browser by entering the following URL in its address bar (to see a similar result to Figure 2-9):

`localhost/test.html`

**NOTE**

Remember that serving a web page from the document root (or a subfolder) is different from loading one into a web browser from your computer’s filesystem. The former will ensure access to PHP, MySQL, and all the features of a web server, while the latter will simply load the file into the browser, which will do its best to display it but will be unable to process any PHP or other server instructions. So, you should generally run examples using the `localhost` preface from your browser’s address bar, unless you are certain that the file doesn’t rely on web server functionality.

### Installing a LAMP on Linux
This book is aimed mostly at PC and Mac users, but its contents will work equally well on a Linux computer. However, there are dozens of popular flavors of Linux, and each of them may require installing a LAMP in a slightly different way, so I can’t cover them all in this book.

That said, many Linux versions come preinstalled with a web server and MySQL, and the chances are that you may already be all set to go. To find out, try entering the following into a browser and see whether you get a default document root web page:

```
localhost
```

If this works, you probably have the Apache server installed and may well have MySQL up and running too; check with your system administrator to be sure.

If you don’t yet have a web server installed, however, there’s a version of AMPPS available that you can download from the [website](#). Installation is similar to the sequence shown in the preceding section. If you need further assistance on using the software, please refer to the [documentation](#).

### Working Remotely

If you have access to a web server already configured with PHP and MySQL, you can always use that for your web development. But unless you have a high-speed connection, it is not always your best option. Developing locally allows you to test modifications with little or no upload delay.

Accessing MySQL remotely may not be easy either. You should use the secure SSH protocol to log into your server to manually create databases and set permissions from the command line. Your web hosting company will advise you on how best to do this and provide you with any password it has set for your MySQL access (as well as, of course, for getting into the
server in the first place). Unless you have no choice, I recommend you do not use the insecure Telnet protocol to remotely log into any server.

### Logging In

I recommend that, at minimum, Windows users should install a program such as PuTTY, for Telnet and SSH access (remember that SSH is much more secure than Telnet).

On a Mac, you already have SSH available. Just select the *Applications* folder, followed by *Utilities*, and then launch Terminal. In the Terminal window, log in to a server using SSH as follows:

```
ssh mylogin@server.com
```

where *server.com* is the name of the server you wish to log into and *mylogin* is the username you will log in under. You will then be prompted for the correct password for that username and, if you enter it correctly, you will be logged in.

### Using FTP

To transfer files to and from your web server, you will need an FTP program. If you go searching the web for a good one, you’ll find so many that it could take you quite a while to come across one with all the right features for you.

My preferred FTP program is the open source FileZilla, for Windows, Linux, and macOS 10.5 or newer (see *Figure 2-11*). Full instructions on how to use FileZilla are available on the [wiki](https://wiki).
Of course, if you already have an FTP program, all the better—stick with what you know.

**Using a Program Editor**

Although a plain-text editor works for editing HTML, PHP, and JavaScript, there have been some tremendous improvements in dedicated program editors, which now incorporate very handy features such as colored syntax highlighting. Today’s program editors are smart and can show you where you have syntax errors before you even run a program. Once you’ve used a modern editor, you’ll wonder how you ever managed without one.

There are a number of good programs available, but I have settled on Editra (see Figure 2-12), because it’s free and available on macOS, Windows, and Linux/Unix, and it suits the way I program. You can download a copy by visiting Editra’s website and selecting the Download link at the top of the page, where you can also find a link to the documentation. Everyone has
different programming styles and preferences, though, so if you don’t get on with it, there are plenty more program editors available to choose from—or you may wish to go directly for an integrated development environment (IDE), as described in the following section.

As you can see from Figure 2-12, Editra highlights the syntax appropriately, using colors to help clarify what’s going on. What’s more, you can place the cursor next to brackets or braces, and Editra will highlight the matching ones so that you can check whether you have too many or too few. In fact, Editra does a lot more in addition, which you will discover and enjoy as you use it.

Again, if you have a different preferred program editor, use that; it’s always a good idea to use programs you’re already familiar with.

Using an IDE
As good as dedicated program editors can be for your programming productivity, their utility pales into insignificance when compared to integrated development environments, which offer many additional features such as in-editor debugging and program testing, as well as function descriptions and much more.

**Figure 2-13** shows the popular phpDesigner IDE with a PHP program loaded into the main frame, and the righthand Code Explorer listing the various classes, functions, and variables that it uses.

![Figure 2-13](image)

*Figure 2-13. When you're using an IDE such as phpDesigner, PHP development becomes much quicker and easier*

When developing with an IDE, you can set breakpoints and then run all (or portions) of your code, which will stop at the breakpoints and provide you with information about the program’s current state.

As an aid to learning programming, the examples in this book can be entered into an IDE and run there and then, without the need to call up your web browser. There are several IDEs available for different platforms, most
of which are commercial, but there are some free ones too. Table 2-1 lists some of the most popular PHP IDEs, along with their download URLs.

Table 2-1. A selection of PHP IDEs

<table>
<thead>
<tr>
<th>IDE</th>
<th>Download URL</th>
<th>Cost</th>
<th>Win</th>
<th>Mac</th>
<th>Lin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eclipse PDT</td>
<td><a href="http://eclipse.org/pdt/downloads/">http://eclipse.org/pdt/downloads/</a></td>
<td>Free</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Komodo IDE</td>
<td><a href="http://activestate.com/Products/komodo_ide">http://activestate.com/Products/komodo_ide</a></td>
<td>$295</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>NetBeans</td>
<td><a href="http://www.netbeans.org">http://www.netbeans.org</a></td>
<td>Free</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>phpDesigner</td>
<td><a href="http://mpsoftware.dk">http://mpsoftware.dk</a></td>
<td>$39</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>PHPeclipse</td>
<td><a href="https://sourceforge.net/projects/phpeclipse/">https://sourceforge.net/projects/phpeclipse/</a></td>
<td>Free</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>PhpED</td>
<td><a href="http://nusphere.com">http://nusphere.com</a></td>
<td>$99</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>PHPEdit</td>
<td><a href="https://phpedit.en.softonic.com">https://phpedit.en.softonic.com</a></td>
<td>$117</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>

Choosing an IDE can be a very personal thing, so if you intend to use one, I advise you to download a couple or more to try them out first; they all either have trial versions or are free to use, so it won’t cost you anything.

You should take the time to install a program editor or IDE you are comfortable with now; you’ll then be ready to try out the examples in the coming chapters.

Armed with these tools, you are now ready to move on to Chapter 3, where we’ll start exploring PHP in further depth and find out how to get HTML and PHP to work together, as well as how the PHP language itself is structured. But before moving on, I suggest you test your new knowledge with the following questions.

Questions

1. What is the difference between a WAMP, a MAMP, and a LAMP?
2. What do the IP address 127.0.0.1 and the URL http://localhost have in common?
3. What is the purpose of an FTP program?

4. Name the main disadvantage of working on a remote web server.

5. Why is it better to use a program editor instead of a plain-text editor?

See “Chapter 2 Answers” in Appendix A for the answers to these questions.