SECOND EDITION
Fully revised and updated content

PHP Solutions

Dynamic Web Design Made Easy

DAVID POWERS

friendsof ED™
an Apress® company

PHP Solutions

Dynamic Web Design Made Easy
Second Edition

David Powers

friendsof ED™
DESIGNER TO DESIGNER™
an Apress® company

PHP Solutions: Dynamic Web Design Made Easy, Second Edition

ISBN-13 (pbk): 978-1-4302-3249-0

ISBN-13 (electronic): 978-1-4302-3250-6

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Credits

Lead Editor:
Ben Renow-Clarke

Technical Reviewers:
Kristian Besley and Jason Nadon

Editorial Board:
Steve Anglin, Mark Beckner, Ewan Buckingham, Tony Campbell, Gary Cornell, Jonathan Gennick, Michelle Lowman, Matthew Moodie, Jeffrey Pepper, Frank Pohlmann, Ben Renow-Clarke, Dominic Shakeshaft, Matt Wade, Tom Welsh

Project Manager:
Anita Castro

Copy Editor:
Heather Lang

Compositor:
Bronkella Publishing, LLC

Indexer:
Toma Mulligan

Artist:
April Milne

Cover Designer:
Anna Ishchenko

Cover Artist:
Corné van Doreen

Contents at a Glance

Contents

About the Author

David Powers is the author of a series of highly successful books on PHP and web development. He began developing websites in 1994 when—as Editor, BBC Japanese TV—he needed a way to promote his fledgling TV channel but didn't have an advertising budget. He persuaded the IT department to let him have some space on the BBC's server and hand-coded a bilingual website from scratch. That experience ignited a passion for web development that burns just as brightly as ever.

After leaving the BBC in 1999, David developed an online system with PHP and MySQL to deliver daily economic and political analysis in Japanese for the clients of a leading international consultancy. Since 2004, he has devoted most of his time to writing books and teaching web development.

David is an Adobe Community Professional and Adobe Certified Instructor for Dreamweaver. In 2010, he became one of the first people to qualify as a PHP 5.3 Zend Certified Engineer.

About the Technical Reviewers

Kristian Besley is the lead developer at Beetroot Design (www.beetrootdesign.co.uk) where he develops web applications, websites, educational interactions and games written mainly in various combinations of PHP, Flash and Javascript.

He has been working with computers and the web for far too long. He also spends far too much time hacking and developing for open-source applications - including Moodle - so that they work just so. Health warning: he has an unhealthy obsession with making his applications super-RSS compatible and overly configurable.

His past and current clients include the BBC, Pearson Education, Welsh Assembly Government and loads of clients with acronyms such as JISC, BECTA, MAWWFIRE and - possibly his favourite of all (well, just try saying it out loud) - SWWETN.

When he isn't working, he's working elsewhere lecturing in Interactive Media (at Gower College Swansea) or providing geeky technical assistance to a whole gamut of institutions or individuals in an effort to save them time and money (at his own expense!!!).

He has authored and co-authored a large number of books for friends of ED and Apress including the Foundation Flash series, Flash MX Video, Foundation ActionScript for Flash (with the wonderful David Powers) and Flash MX Creativity. His words have also graced the pages of Computer Arts a few times too.

Kristian currently resides with his family in Swansea, Wales and is a proud fluent Welsh speaker with a passion for pushing the language on the web and in bilingual web applications where humanly possible.

Jason Nadon has ten years experience building and supporting complex web applications. He is an active member of the web developer community and teaches several classes in his hometown in Michigan. He has been in the Information Technology field for more than twelve years and holds several industry certifications. He is currently working as an Infrastructure Manager for a global information company.

Acknowledgments

My thanks go to everyone who was involved in the production of this book. The original idea to write *PHP Solutions* came from Chris Mills, my editor for many years at Apress/friends of ED, who's now Developer Relations Manager at Opera and a passionate advocate of web standards. It was a great idea, Chris. Thanks to your help, the first edition of this book became my biggest seller. The invitation to write this second edition came from Chris's successor, Ben Renow-Clarke. Like Chris, Ben has given me free rein to shape this book according to my own ideas but has always put himself in the position of the reader, nudging me in the right direction when an explanation wasn't clear enough or a chapter was badly organized.

I'm grateful to Kris Besley and Jason Nadon, who scoured my text and code for errors. Much though I hate to admit it, they did find some. Kris, in particular, made some really good suggestions for improving the code. *Diolch yn fawr iawn*. Any mistakes that remain are my responsibility alone.

Most of all, thanks to you for reading. I hope you enjoy the book as much as I have enjoyed writing it.

Introduction

When the first edition of *PHP Solutions* was published, I was concerned that the subtitle, *Dynamic Web Design Made Easy*, sounded overambitious. PHP is not difficult, but nor is it like an instant cake mix: just add water and stir. Every website is different, so it's impossible to grab a script, paste it into a web page, and expect it to work. My aim was to help web designers with little or no knowledge of programming gain the confidence to dive into the code and adjust it to their own requirements.

The enduring popularity of the first edition suggests that many readers took up the challenge. Part of the book's success stemmed from the use of clear, straightforward language, highlighting points where you might make mistakes, with advice on how to solve problems. Another factor was its emphasis on forward and backward compatibility. The solutions were based on PHP 5, but alternatives were provided for readers still stuck on PHP 4.

Time has moved on. PHP 5 is now a mature and stable platform. This new edition of *PHP Solutions* requires PHP 5.2 and MySQL 4.1 or later. Some code will work with earlier versions, but most of it won't. The emphasis on future compatibility remains unchanged. All the code in this book avoids features destined for removal when work resumes on PHP 6 (at the time of this writing, it's not known when that will be).

The decision to drop support for older versions of PHP and MySQL has been liberating. When friends of ED asked me to prepare a new edition of this book, I initially thought it would involve just brushing away a few cobwebs. As soon as I started reviewing the code, I realized just how much the need to cater for PHP 4 had constrained me. It's also fair to say that my coding style and knowledge of PHP had expanded greatly in the intervening years.

As a result, this new edition is a major rewrite. The basic structure of the book remains the same, but every chapter has been thoroughly revised, and an extra two have been added. In some cases, little remains of the original chapter other than the title. For example, the file upload and thumbnail creation scripts in Chapters 6 and 8 have been completely refactored as PHP 5 custom classes, and the mail processing script in Chapter 5 has been rewritten to make it easier to redeploy in different websites. Other big changes include a class to check password strength in Chapter 9 and detailed coverage of the date and time classes introduced in PHP 5.2 and 5.3. Want to display the date of events on the second Tuesday of each month? Chapter 14 shows how to do it in half a dozen lines of code. Chapter 16 adds coverage of foreign key constraints in InnoDB, the default storage engine in MySQL 5.5.

I hesitated before devoting so much attention to using PHP classes. Many regard them as an advanced subject, not suitable for readers who don't have a programming background. But the advantages far outweighed my reservations. In simple terms, a class is a collection of predefined functions designed to perform related tasks. The beauty of using classes is that they're project-neutral. Admittedly, the file upload class in Chapter 6 is longer than the equivalent script in the first edition of *PHP Solutions*, but you can reuse it in multiple projects with just a few lines of code. If you're in hurry or are daunted by the prospect of building class definitions, you can simply use the finished files. However, I encourage you to explore the class definitions. The code will teach you a lot of PHP that you'll find useful in other situations.

Each chapter takes you through a series of stages in a single project, with each stage building on the previous one. By working through each chapter, you get the full picture of how everything fits together. You can later refer to the individual stages to refresh your memory about a particular technique. Although this isn't a reference book, Chapter 3 is a primer on PHP syntax, and some chapters contain short reference sections—notably Chapter 7 (reading from and writing to files), Chapter 9 (sessions), Chapter 10 (MySQL data types), Chapter 11 (MySQL prepared statements), Chapter 13 (the four essential SQL commands), and Chapter 14 (working with dates and times).

So, how easy is easy? I have done my best to ease your path, but there is no magic potion. It requires some effort on your part. Don't attempt to do everything at once. Add dynamic features to your site a few at a time. Get to understand how they work, and your efforts will be amply rewarded. Adding PHP and MySQL to your skills will enable you to build websites that offer much richer content and an interactive user experience.

Using the example files

All the files necessary for working through this book can be downloaded from the friends of ED website at `http://www.friendsofed.com/downloads.html`. Make sure you select the download link for *PHP Solutions: Dynamic Web Design Made Easy, Second Edition*. The code is very different from the first edition.

Set up a PHP development environment, as described in Chapter 2. Unzip the files, and copy the `phpsols` folder and all its contents into your web server's document root. The code for each chapter is in a folder named after the chapter: `ch01`, `ch02`, and so on. Follow the instructions in each PHP solution, and copy the relevant files to the site root or the work folder indicated.

Where a page undergoes several changes during a chapter, I have numbered the different versions like this: `index_01.php`, `index_02.php`, and so on. When copying a file that has a number, remove the underscore and number from the filename, so `index_01.php` becomes `index.php`. If you are using a program like Dreamweaver that prompts you to update links when moving files from one folder to another, do *not* update them. The links in the files are designed to pick up the right images and style sheets when located in the target folder. I have done this so you can use a file comparison utility to check your files against mine.

If you don't have a file comparison utility, I strongly urge you to install one. It will save you hours of head scratching when trying to spot the difference between your version and mine. A missing semicolon or mistyped variable can be hard to spot in dozens of lines of code. Windows users can download WinMerge for free from `http://winmerge.org/`. I use Beyond Compare (`www.scootersoftware.com`). It's not free but is excellent and reasonably priced. BBEdit on a Mac includes a file comparison utility. Alternatively, use the file comparison feature in TextWrangler, which can be downloaded free from `www.barebones.com/products/textwrangler/`.

The HTML code in the example files and text uses HTML5 syntax, but I have avoided using elements that are not supported by older browsers. Even Internet Explorer 6 understands the HTML5 `DOCTYPE` declaration, and new form elements that older browsers don't recognize are rendered as text input fields.

Layout conventions

To keep this book as clear and easy to follow as possible, the following text conventions are used throughout.

Important words or concepts are normally highlighted on the first appearance in **bold type**.

Code is presented in `fixed-width font`.

New or changed code is normally presented in **`bold fixed-width font`**.

Pseudo-code and variable input are written in *`italic fixed-width font`*.

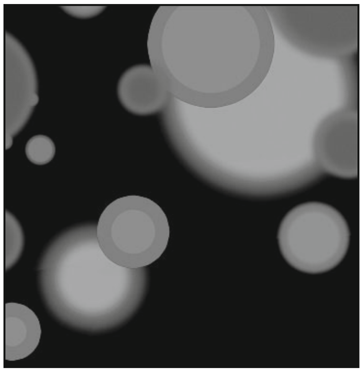
Menu commands are written in the form **Menu ➤ Submenu ➤ Submenu**.

Where I want to draw your attention to something, I've highlighted it like this:

Ahem, don't say I didn't warn you.

Sometimes code won't fit on a single line in a book. Where this happens, I use an arrow like this: ➥.

```
This is a very, very long section of code that should be written all on the same ➥
line without a break.
```



Chapter 1

What Is PHP—And Why Should I Care?

One of the first things most people want to know about PHP is what the initials stand for. Then they wish they had never asked. Officially, PHP stands for **PHP: Hypertext Preprocessor**. It's an ugly name that gives the impression that it's strictly for nerds or propellerheads. Nothing could be further from the truth.

PHP is a scripting language that brings websites to life in the following ways:

- Sending feedback from your website directly to your mailbox
- Uploading files through a web page
- Generating thumbnails from larger images
- Reading and writing to files
- Displaying and updating information dynamically
- Using a database to display and store information
- Making websites searchable
- And much more . . .

By reading this book, you'll be able to do all that. PHP is easy to learn; it's platform-neutral, so the same code runs on Windows, Mac OS X, and Linux; and all the software you need to develop with PHP is open source and therefore free. Several years ago, there was a lighthearted debate on the PHP General mailing list (`http://news.php.net/php.general`) about changing what PHP stands for. Among the suggestions were Positively Happy People and Pretty Happy Programmers. The aim of this book is to help you put PHP to practical use—and in the process understand what makes PHP programmers so happy.

In this chapter, you'll learn about the following:

- How PHP has grown into the most widely used technology for dynamic websites
- How PHP makes web pages dynamic
- How difficult—or easy—PHP is to learn
- Whether PHP is safe
- What software you need to write PHP

How PHP has grown

Although PHP is now the most widely used technology for creating dynamic websites, it started out with rather modest ambitions—and a different name—in 1995. Originally called Personal Home Page Tools (PHP Tools), one of its goals was to create a guestbook by gathering information from an online form and displaying it on a web page. Shortly afterward, the ability to communicate with a database was added. When version 3 was released in 1998, it was decided to drop Personal Home Page from the name, because it sounded like something for hobbyists and didn't do justice to the range of sophisticated features that had been added. PHP 3 was described as "a very programmer-friendly scripting language suitable for people with little or no programming experience as well as the seasoned web developer who needs to get things done quickly."

Since then, PHP has developed even further, adding extensive support for object-oriented programming (OOP) in PHP 5. One of the language's great attractions, though, is that it remains true to its roots. You can start writing useful scripts without the need to learn lots of theory, yet be confident in the knowledge that you're using a technology with the capability to develop industrial-strength applications. PHP is the language that drives the highly popular content management systems (CMSs), Drupal (`http://drupal.org/`), Joomla! (`www.joomla.org`), and WordPress (`http://wordpress.org/`). It also runs some of the most heavily used websites, including Facebook (`www.facebook.com`) and Wikipedia (`www.wikipedia.org`).

PHP can now be regarded as a mature technology in the sense that it has a large user base, is widely supported, and has many advanced features. New features are being continually added, although these are mainly of interest to advanced users.

At the time of this writing, the current version is PHP 5.3. Development of PHP 6 was suspended indefinitely in early 2010, when it was realized the original plans had been too ambitious.

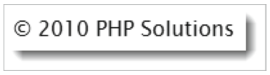
The emphasis in this book is on code that works now, not on what might work at some unspecified time in the future. Care has also been taken to avoid using features that have been deprecated—in other words, marked for removal from the next major version of PHP.

How PHP makes pages dynamic

PHP was originally designed to be embedded in the HTML of a web page, and that's the way it's often still used. For example, if you want to display the current year in a copyright notice, you could put this in your footer:

```
<p>&copy; <?php echo date('Y'); ?> PHP Solutions</p>
```

On a PHP–enabled web server, the code between the `<?php` and `?>` tags is automatically processed and displays the year like this:


© 2010 PHP Solutions

This is only a trivial example, but it illustrates some of the advantages of using PHP:

- You can enjoy your New Year's party without worrying about updating your copyright notice. Anyone accessing your site after the stroke of midnight sees the correct year.
- Unlike using JavaScript to display the date, the processing is done on the web server, so it doesn't rely on JavaScript being enabled in the user's browser.
- The date is calculated by the web server, so it's not affected if the clock in the user's computer is set incorrectly.

Although it's convenient to embed PHP code in HTML like this, it often results in typing the same code repeatedly, which is boring and leads to mistakes. It can also make your web pages difficult to maintain, particularly once you start using more complex PHP code. Consequently, it's common practice to store a lot of dynamic code in separate files and use PHP to build your pages from the different components. The separate files—or *include files*, as they're usually called—can contain either only PHP, only HTML, or a mixture of both.

At first, it can be difficult to get used to this way of working, but it's much more efficient. As a simple example, you can put your website's navigation menu in an include file and use PHP to include it in each page. Whenever you need to make any changes to the menu, you edit just one file—the include file—and the changes are automatically reflected in every page that includes the menu. Just imagine how much time that saves on a website with dozens of pages.

With an ordinary HTML page, the content is fixed by the web developer at design time and uploaded to the web server. When somebody visits the page, the web server simply sends the HTML and other assets, such as images and style sheet. It's a simple transaction—the request comes from the browser, and the fixed content is sent back by the server. When you build web pages with PHP, much more goes on. Figure 1-1 shows what happens.

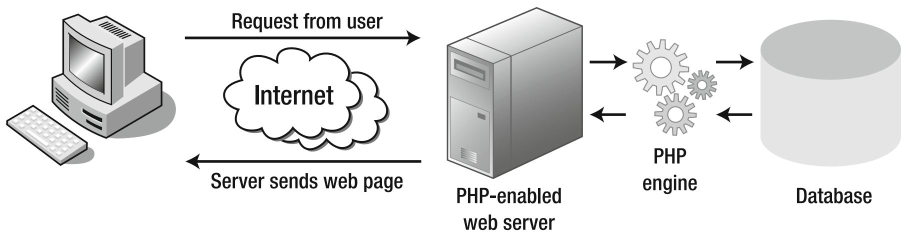


Figure 1-1. The web server builds each PHP page dynamically in response to a request.

When a PHP–driven website is visited, it sets in train the following sequence of events:

1. The browser sends a request to the web server.
2. The web server hands the request to the PHP engine, which is embedded in the server.
3. The PHP engine processes the code. In many cases, it might also query a database before building the page.
4. The server sends the completed page back to the browser.

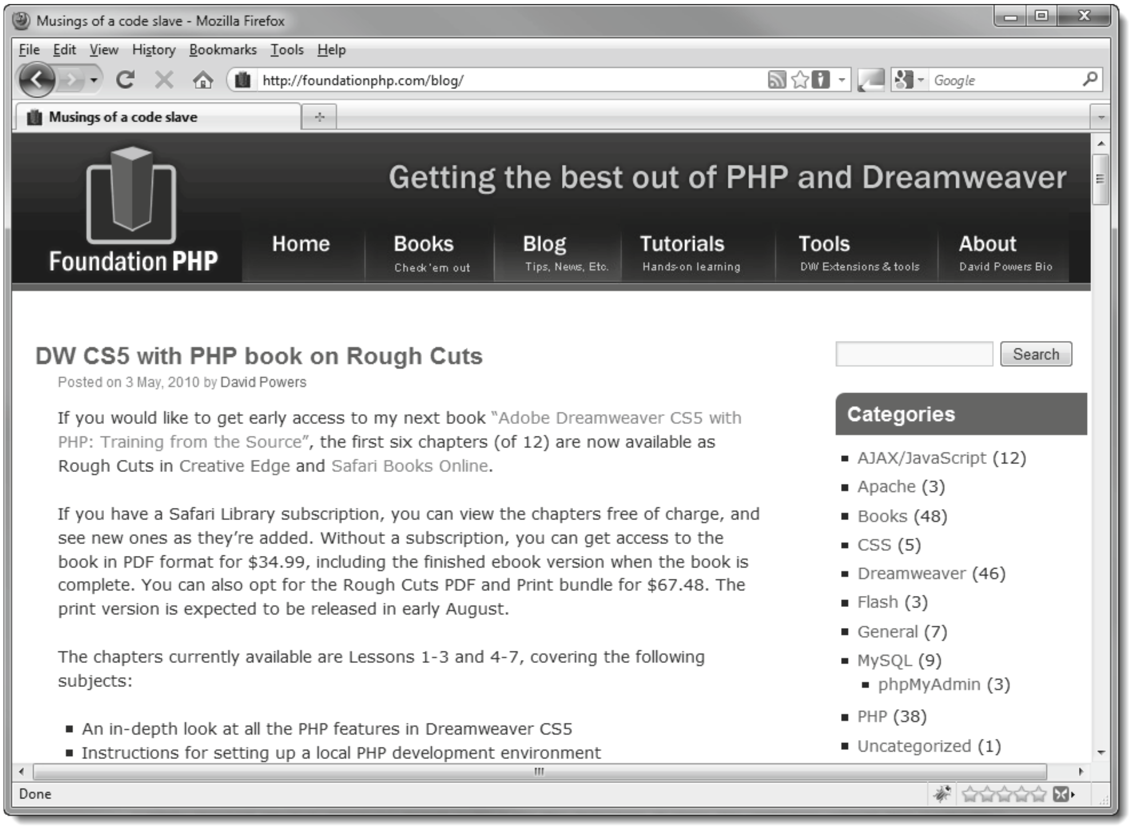
This process usually takes only a fraction of a second, so the visitor to a PHP website is unlikely to notice any delay. Because each page is built individually, PHP pages can respond to user input, displaying different content when a user logs in or showing the results of a database search.

Creating pages that think for themselves

PHP is a **server-side language**. The PHP code remains on the web server. After it has been processed, the server sends only the output of the script. Normally, this is HTML, but PHP can also be used to generate other web languages, such as Extensible Markup Language (XML).

PHP enables you to introduce logic into your web pages. This logic is based on alternatives. Some decisions are based on information that PHP gleans from the server: the date, the time, the day of the week, information in the page's URL, and so on. If it's Wednesday, show Wednesday's TV schedules. At other times, decisions are based on user input, which PHP extracts from online forms. If you have registered with a site, display your personalized information . . . that sort of thing.

As a result, you can create an infinite variety of output from a single script. For example, if you visit my blog at `http://foundationphp.com/blog/` (see Figure 1-2), and click various internal links, what you see is always the same page but with different content. Admittedly, I tend to write always about the same kinds of subjects, but that's my fault, not PHP's.


Musings of a code slave - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://foundationphp.com/blog/
Google
Musings of a code slave
Getting the best out of PHP and Dreamweaver
Foundation PHP
Home
Books Check 'em out
Blog Tips, News, Etc.
Tutorials Hands-on learning
Tools DW Extensions & tools
About David Powers Bio
Search
Done

DW CS5 with PHP book on Rough Cuts

Posted on 3 May, 2010 by David Powers

If you would like to get early access to my next book "Adobe Dreamweaver CS5 with PHP: Training from the Source", the first six chapters (of 12) are now available as Rough Cuts in Creative Edge and Safari Books Online.

If you have a Safari Library subscription, you can view the chapters free of charge, and see new ones as they're added. Without a subscription, you can get access to the book in PDF format for $34.99, including the finished ebook version when the book is complete. You can also opt for the Rough Cuts PDF and Print bundle for $67.48. The print version is expected to be released in early August.

The chapters currently available are Lessons 1-3 and 4-7, covering the following subjects:

- An in-depth look at all the PHP features in Dreamweaver CS5
- Instructions for setting up a local PHP development environment

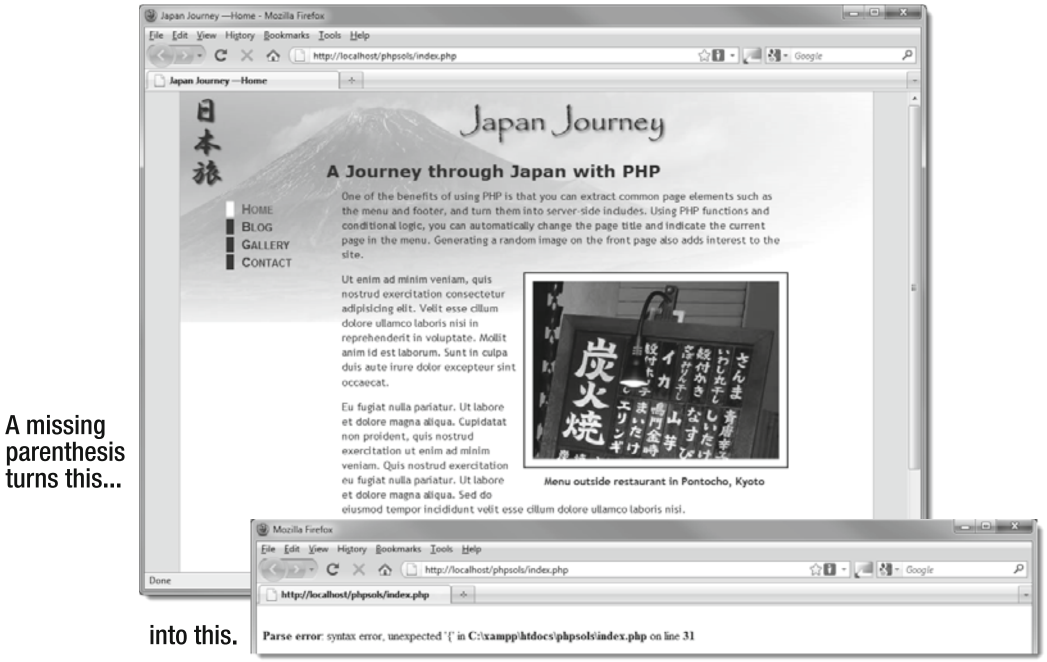
Categories

- AJAX/JavaScript (12)
- Apache (3)
- Books (48)
- CSS (5)
- Dreamweaver (46)
- Flash (3)
- General (7)
- MySQL (9)
  - phpMyAdmin (3)
- PHP (38)
- Uncategorized (1)

Figure 1-2. Blogs are a good example of sites ideally suited to PHP.

How hard is PHP to use and learn?

PHP isn't rocket science, but at the same time, don't expect to become an expert in five minutes. Perhaps the biggest shock to newcomers is that PHP is far less tolerant of mistakes than browsers are with HTML. If you omit a closing tag in HTML, most browsers will still render the page. If you omit a closing quote, semicolon, or brace in PHP, you'll get an uncompromising error message like the one shown in Figure 1-3. This isn't just a feature of PHP but of all server-side technologies, including ASP, ASP.NET, and ColdFusion.


A missing parenthesis turns this...
into this.
Japan Journey —Home - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost/phpsols/index.php
Google
Japan Journey —Home
日本旅
HOME
BLOG
GALLERY
CONTACT
Done
Mozilla Firefox
http://localhost/phpsols/index.php

Japan Journey

A Journey through Japan with PHP

One of the benefits of using PHP is that you can extract common page elements such as the menu and footer, and turn them into server-side includes. Using PHP functions and conditional logic, you can automatically change the page title and indicate the current page in the menu. Generating a random image on the front page also adds interest to the site.

Ut enim ad minim veniam, quis nostrud exercitation consectetur adipisicing elit. Velit esse cillum dolore ullamco laboris nisi in reprehenderit in voluptate. Mollit anim id est laborum. Sunt in culpa duis aute irure dolor excepteur sint occaecat.

Eu fugiat nulla pariatur. Ut labore et dolore magna aliqua. Cupidatat non proident, quis nostrud exercitation ut enim ad minim veniam. Quis nostrud exercitation eu fugiat nulla pariatur. Ut labore et dolore magna aliqua. Sed do eiusmod tempor incididunt velit esse cillum dolore ullamco laboris nisi.

Menu outside restaurant in Pontocho, Kyoto

```
Parse error: syntax error, unexpected '{' in C:\xampp\htdocs\phpsols\index.php on line 31
```

Figure 1-3. Server-side languages like PHP are intolerant of most coding errors.

If you're the sort of web designer or developer who uses a visual design tool, such as Adobe Dreamweaver or Microsoft Expression Web, and never looks at the underlying code, it's time to rethink your approach. Mixing PHP with poorly structured HTML is likely to lead to problems. PHP uses loops to perform repetitive tasks, such as displaying the results of a database search. A **loop** repeats the same section of code—usually a mixture of PHP and HTML—until all results have been displayed. If you put the loop in the wrong place, or if your HTML is badly structured, your page is likely to collapse like a house of cards. If you're not already in the habit of doing so, it's a good idea to check your pages using the World Wide Web Consortium's (W3C) Markup Validation Service (`http://validator.w3.org/unicorn`).

The W3C is the international body that develops standards—such as HTML and CSS—and guidelines to ensure the long-term growth of the Web. It's led by the inventor of the World Wide Web, Tim Berners-Lee. To learn about the W3C's mission, see `www.w3.org/Consortium/mission`.

Can I just copy and paste the code?

There's nothing wrong with copying the code in this book. That's what it's there for. Copying is the way we all learn as children, but most of us progress from the copycat stage by asking questions and beginning to experiment on our own. Rather than attempt to teach you PHP by going through a series of boring exercises that have no immediate value to your web pages, I've structured this book so that you jump straight into applying your newfound knowledge to practical projects. At the same time, I explain what the code is for and why it's there. Even if you don't understand exactly how it all works, this should give you sufficient knowledge to know which parts of the code to adapt to your own needs and which parts are best left alone.

PHP is a toolbox full of powerful features. It has thousands of built-in functions that perform all sorts of tasks, such as converting text to uppercase, generating thumbnail images from full-sized ones, or connecting to a database. The real power comes from combining these functions in different ways and adding your own conditional logic. To get the best out of this book, you need to start experimenting with the tools you learn about in these pages and come up with your own solutions.

How safe is PHP?

PHP is like the electricity or kitchen knives in your home: handled properly, it's very safe; handled irresponsibly, it can do a lot of damage. One of the inspirations for the first edition of this book was a spate of malicious attacks that erupted in late 2005. The attacks exploited a vulnerability in email scripts, turning websites into spam relays. Few people were immune. I certainly wasn't, but once I was alerted to the problem, I plugged the hole and stopped the attacks in their tracks. However, day after day, people were sending frantic pleas for help to online forums. Even when they were told how to deal with the problem, their response became even more frantic. Many admitted they didn't know the first thing about any of the code they were using in their websites. For someone building websites as a hobby, this might be understandable, but many of these people were "professionals" who had built sites on behalf of clients. The clients were naturally unhappy when their mailboxes started filling with spam. They were no doubt even unhappier when their domains were suspended by hosting companies fed up with insecure scripts on their servers.

The moral of this story is not that PHP is unsafe; nor does everyone need to become a security expert to use PHP. What is important is to understand the basic principle of PHP safety: *always check user input before processing it*. You'll find that to be a constant theme throughout this book. Most security risks can be eliminated with very little effort.

Perhaps the most worrying aspect is that, more than five years after this exploit was first revealed, I still see people using insecure email scripts. The best way to protect yourself is to understand the code you're using. Even if you can't solve a problem yourself, you can implement any remedies suggested to you by the author of the script or another expert.

What software do I need to write PHP?

Strictly speaking, you don't need any special software to write PHP scripts. PHP code is plain text and can be created in any text editor, such as Notepad on Windows or TextEdit on Mac OS X. Having said that, you would need to be a masochist to use a plain text editor. Your current web development program might already support PHP. If it doesn't there's a wide choice of programs—both paid-for and free—that have features designed to speed up the development process.

What to look for when choosing a PHP editor

If there's a mistake in your code, your page will probably never make it as far as the browser, and all you'll see is an error message. You should choose a script editor that has the following features:

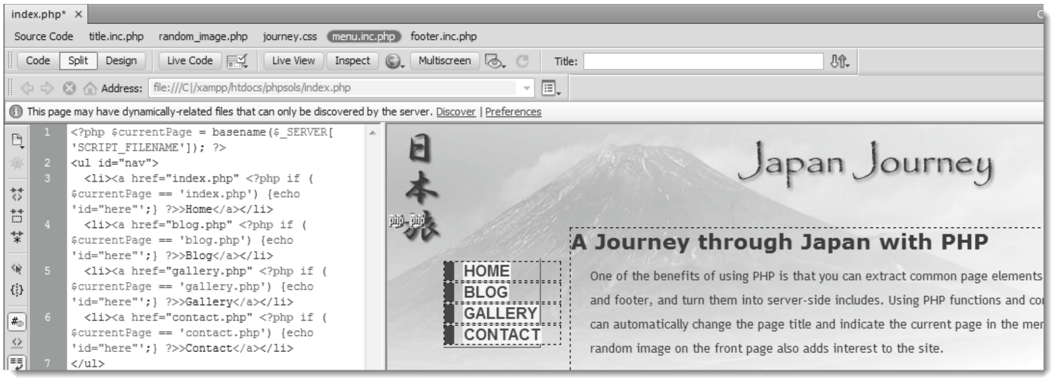
- **PHP syntax checking**: This used to be found only in expensive, dedicated programs, but it's now a feature in several free programs. Syntax checkers monitor the code as you type and highlight errors, saving a great deal of time and frustration.
- **PHP syntax coloring**: Code is highlighted in different colors according to the role it plays. If your code is in an unexpected color, it's a sure sign you've made a mistake.
- **PHP code hints**: PHP has so many built-in functions, it can be difficult to remember how to use them—even for an experienced user. Many script editors automatically display tooltips with reminders of how a particular piece of code works.
- **Line numbering**: Finding a specific line quickly makes troubleshooting a lot simpler.
- **A "balance braces" feature**: Parentheses (()), square brackets ([]), and curly braces ({}) must always be in matching pairs. It's easy to forget to close a pair. All good script editors help find the matching parenthesis, bracket, or brace.

The following sections describe some of the script editors you might like to consider. It's by no means an exhaustive list but is based on personal experience.

General purpose web development tools with PHP support

Two of the most widely used integrated development environments (IDEs) for building websites, Adobe Dreamweaver (`www.adobe.com/products/dreamweaver/`) and Microsoft Expression Web (`www.microsoft.com/expression/products/web_overview.aspx`), have built-in support for PHP.

- **Dreamweaver CS5**: Dreamweaver is a good, standards-compliant visual editor. PHP support was taken to a completely new level in Dreamweaver CS5 with the addition of syntax checking, embedded documentation (complete with examples), and autocompletion of variables. Particularly useful is the ability to work in PHP includes, while keeping the main page visible in the workspace (see Figure 1-4).



This page may have dynamically-related files that can only be discovered by the server. Discover | Preferences

```
<?php $currentPage = basename($_SERVER[
'SCRIPT_FILENAME']); ?>
<ul id="nav">
  <li><a href="index.php" <?php if (
$currentPage == 'index.php') {echo
'id="here"';} ?>>Home</a></li>
  <li><a href="blog.php" <?php if (
$currentPage == 'blog.php') {echo
'id="here"';} ?>>Blog</a></li>
  <li><a href="gallery.php" <?php if (
$currentPage == 'gallery.php') {echo
'id="here"';} ?>>Gallery</a></li>
  <li><a href="contact.php" <?php if (
$currentPage == 'contact.php') {echo
'id="here"';} ?>>Contact</a></li>
</ul>
```

A Journey through Japan with PHP

One of the benefits of using PHP is that you can extract common page elements
and footer, and turn them into server-side includes. Using PHP functions and co
can automatically change the page title and indicate the current page in the me
random image on the front page also adds interest to the site.

Figure 1-4. Dreamweaver CS5 lets you edit PHP include files and view the results in Live View.

- **Expression Web**: The level of PHP support in versions 2, 3, and 4 of Expression Web is similar to that offered in older versions of Dreamweaver—in other words, syntax coloring, code hints for PHP core functions, and line numbers. The big drawback at the time of this writing is there's no support for syntax checking.

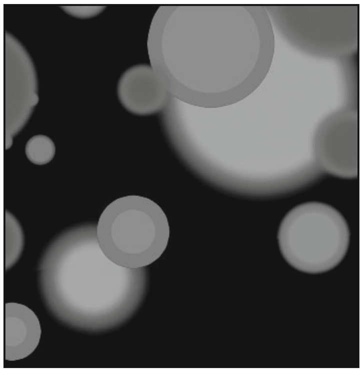
Dedicated script editors

Even if you don't plan to do a lot of PHP development, you should consider using a dedicated script editor if your web development IDE doesn't support syntax checking. The following dedicated script editors have all the essential features, such as syntax checking and code hints. They also support HTML and CSS but lack the visual display offered by Dreamweaver or Expression Web.

- **Zend Studio** (`www.zend.com/en/products/studio/`): If you're really serious about PHP development, Zend Studio is the most fully featured IDE for PHP. It's created by Zend, the company run by leading contributors to the development of PHP. Zend Studio runs on Windows, Mac OS X, and Linux. Its main drawback is cost, although the price includes 12 months of free upgrades and support.
- **PhpED** (`www.nusphere.com/products/phped.htm`): This is available in three different versions. The least expensive version has all the features you need as a beginner. If you need the more advanced features later, you can upgrade to one of the other versions. Windows only.
- **PHP Development Tools** (`www.eclipse.org/pdt/`): PDT is actually a cut-down version of Zend Studio and has the advantage of being free. The disadvantage is that at the time of this writing, the documentation for PDT is almost nonexistent. It runs on Eclipse, the open source IDE that supports multiple computer languages. If you have used Eclipse for other languages, you should find it relatively easy to use. PDT runs on Windows, Mac OS X, and Linux and is available either as an Eclipse plug-in or as an all-in-one package that automatically installs Eclipse and the PDT plug-in.
- **Komodo Edit** (`www.activestate.com/komodo-edit`): This is a free, open source IDE for PHP and a number of other popular computer languages. It's available for Windows, Mac OS X, and Linux. It's a cut-down version of Komodo IDE, which is a paid-for program with more advanced features. There are separate download links for a free trial of Komodo IDE, which is time-limited, and for Komodo Edit, which doesn't expire.

So, let's get on with it . . .

This chapter has provided only a brief overview of what PHP can do to add dynamic features to your websites and what software you need. The first stage in working with PHP is to set up a testing environment. The next chapter covers the process for both Windows and Mac OS X.



Chapter 2

Getting Ready to Work with PHP

Now that you've decided to use PHP to enrich your web pages, you need to make sure that you have everything you need to get on with the rest of this book. Although you can test everything on your remote server, it's usually more convenient to test PHP pages on your local computer. Everything you need to install is free. In this chapter, I'll explain the various options and give instructions for both Windows and Mac OS X.

What this chapter covers:

- Determining what you need
- Deciding whether to create a local testing setup
- Using a ready-made package
- Making sure PHP has the right settings

Checking whether your website supports PHP

The easiest way to find out whether your website supports PHP is to ask your hosting company. The other way to find out is to upload a PHP page to your website and see if it works. Even if you know that your site supports PHP, do the following test to confirm which version is running:

1. Open a text editor, such as Notepad or TextEdit, and type the following code into a blank page:

   ```
   <?php echo phpversion(); ?>
   ```

2. Save the file as `phpversion.php`. It's important to make sure that your operating system doesn't add a `.txt` filename extension after the `.php`. Mac users should also make sure that TextEdit doesn't save the file in Rich Text Format (RTF). If you're at all unsure, use `phpversion.php` from the `ch02` folder in the files accompanying this book.

3. Upload `phpversion.php` to your website in the same way you would an HTML page, and then type the URL into a browser. Assuming you upload the file to the top level of your site, the URL will be something like `http://www.example.com/phpversion.php`.

   If you see a three-part number like **5.3.3** displayed onscreen, you're in business: PHP is enabled. The number tells you which version of PHP is running on your server. *You need a minimum of 5.2.0 to use the code in this book.*

   If you get a message that says something like **Parse error**, it means PHP is supported but that you have made a mistake in typing the file. Use the version in the `ch02` folder instead.

   If you just see the original code, it means PHP is not supported.

Official support for PHP 4 was terminated in August 2008. Although PHP 4 was excellent, the time to lay it to rest has long since passed. PHP 5 has been around since 2004. It's faster and has more features, and most important of all, it's actively maintained, making it more secure.

At the time of this writing, two series are being currently maintained: PHP 5.2 and PHP 5.3. All the code in this book has been designed to run on both versions, and it avoids using features that are scheduled to be removed from future versions. If your server is running a version earlier than PHP 5.2, contact your host and tell them you want the most recent stable version of PHP. If your host refuses, it's time to change your hosting company.

Deciding where to test your pages

Unlike ordinary web pages, you can't just double-click PHP pages in Windows Explorer or Finder on a Mac and view them in your browser. They need to be **parsed**—processed—through a web server that supports PHP. If your hosting company supports PHP, you can upload your files to your website and test them there. However, you need to upload the file every time you make a change. In the early days, you'll probably find you have to do this often because of some minor mistake in your code. As you become more experienced, you'll still need to upload files frequently because you'll want to experiment with different ideas.

If you want to get working with PHP straight away, by all means use your own website as a test bed. However, you'll soon discover the need for a local PHP test environment. The rest of this chapter is devoted to showing you how to do it, with instructions for Windows and Mac OS X.

What you need for a local test environment

To test PHP pages on your local computer, you need to install the following:

- A web server (Apache or IIS)
- PHP

To work with a database, you'll also need MySQL and a web-based front end for MySQL called phpMyAdmin. All the software you need is free. The only cost to you is the time it takes to download the necessary files, plus, of course, the time to make sure everything is set up correctly. In most cases, you should be up and running in less than an hour, probably considerably less.

You don't need any special equipment. A web server is a piece of software that displays web pages, not a separate computer. As long as you have at least 1GB of free disk space, you should be able to install all the software on your computer—even one with modest specifications.

If you already have a PHP test environment on your local computer, there's no need to reinstall. Just check the section at the end of the chapter titled "Checking your PHP (Windows and Mac)."

Individual programs or an all-in-one package?

For many years, I advocated installing each component of a PHP testing environment separately, rather than using a package that installs Apache, PHP, MySQL, and phpMyAdmin automatically in a single operation. My advice was based on the dubious quality of some early all-in-one packages, which installed easily but were next to impossible to uninstall or upgrade. The all-in-one packages currently available are excellent, and I have no hesitation in recommending them. On my computers, I use XAMPP for Windows (`www.apachefriends.org/en/xampp-windows.html`) and MAMP for Mac OS X (`www.mamp.info/en/mamp/index.html`).

*Setting up a PHP testing environment with an all-in-one package is normally trouble free. The main cause of difficulty is a conflict with another program using port 80, which Apache and IIS use to listen for page requests. If Skype is installed, go to the **Advanced** section of **Skype Preferences**, and make sure it's not using port 80. Try 42815 as the incoming port instead.*

Setting up on Windows

These instructions have been tested on Windows 7, Windows Vista, and Windows XP. Make sure that you're logged on as an Administrator before proceeding.

Getting Windows to display filename extensions

By default, most Windows computers hide the three- or four-letter filename extension, such as `.doc` or `.html`, so all you see in dialog boxes and Windows Explorer is `thisfile` instead of `thisfile.doc` or `thisfile.html`. The ability to see these filename extensions is essential for working with PHP.

Use these instructions to enable the display of filename extensions in Windows 7 and Windows Vista:

1. Open **Start ➤ Computer**.
2. Select **Organize ➤ Folder and Search Options**.
3. In the dialog box that opens, select the **View** tab.
4. In the **Advanced settings** section, uncheck the box marked **Hide extensions for known file types**.
5. Click **OK**.