THE EXPERT'S VOICE® IN WEB DEVELOPMENT

Learn PHP 7

Object-Oriented Modular Programming using HTML5, CSS3, JavaScript, XML, JSON, and MySQL

Steve Prettyman

Apress®

Learn PHP 7

Object-Oriented Modular Programming using HTML5, CSS3, JavaScript, XML, JSON, and MySQL

Steve Prettyman

Apress®

Learn PHP 7: Object-Oriented Modular Programming using HTML5, CSS3, JavaScript, XML, JSON, and MySQL

Steve Prettyman
Stone Mountain, Georgia USA

ISBN-13 (pbk): 978-1-4842-1729-0 ISBN-13 (electronic): 978-1-4842-1730-6
DOI 10.1007/978-1-4842-1730-6

Library of Congress Control Number: 2015960461

Copyright © 2016 by Steve Prettyman

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

Trademarked names, logos, and images may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, logo, or image we use the names, logos, and images only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Managing Director: Welmoed Spahr
Lead Editor: Steve Anglin
Editorial Board: Steve Anglin, Louise Corrigan, Jonathan Gennick, Robert Hutchinson, Michelle Lowman, James Markham, Susan McDermott, Matthew Moodie, Jeffrey Pepper, Douglas Pundick, Ben Renow-Clarke, Gwenan Spearing
Coordinating Editor: Mark Powers
Copy Editor: Kezia Endsley
Compositor: SPi Global
Indexer: SPi Global
Artist: SPi Global

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a Delaware corporation.

For information on translations, please e-mail rights@apress.com, or visit www.apress.com.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales–eBook Licensing web page at www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this text is available to readers at www.apress.com. For additional information about how to locate and download your book's source code, go to www.apress.com/source-code/. Readers can also access source code at SpringerLink in the Supplementary Material section for each chapter.

Printed on acid-free paper

Contents at a Glance

Contents

About the Author

Steve Prettyman earned his Bachelor's of Arts Degree in education from Oglethorpe University in 1979. He quickly began his teaching career as a high school mathematics instructor while continuing his education by earning a Master's Degree in business information systems from Georgia State University (1985). Since then, Steve has spent over 30 years in the IT industry. He has been an instructor at Chattahoochee Technical College, Kennesaw State University, and Southern Polytechnic State University for almost 20 years. His primary teaching responsibilities include programming, web design, and web application development.

Acknowledgments

Thank you to everyone who has helped put this book together. Special thanks to the Introduction to PHP classes that have been the true testers and debuggers for this journey.

Special acknowledgement to all the open source developers and providers of free tutorials and training to every Internet user who wants to learn more about programming, especially w3schools and The New Boston.

Introduction

Learn PHP 7: Object-Oriented Modular Programming using HTML5, CSS3, JavaScript, XML, JSON, and MySQL is intended for use as a beginning level programming book. It is not the goal of this book to cover advanced techniques in the current versions of the PHP programming language. Some knowledge of general programming concepts is expected but no actual programming experience or education is assumed.

All code examples in this book are compatible with PHP 7. Most examples are compatible with PHP 5.6. The newest (as of the publication date) methods (functions) available in PHP have been used to provide the reader with the most current coding techniques. The examples use core methods provided in the PHP language. PHP includes many additional methods to accomplish similar tasks. The reader may, and should, research additional ways of improving security, performance, and other techniques. It is the goal of this book to prompt users to always consider using the most secure and efficient methods of program development. The code in this book provides some examples of using these techniques. The user should remember that *no program is 100% secure*. The programmer can only strive to make an application as secure as possible. It takes a team of developers, network personnel, security administrators, data center personnel, and others working together to provide the safest environment.

A Different Approach

There are quite a number of PHP books on the market today. What makes this book any different than any other?

- This book uses the concept of "learning by doing," which shows the reader how to develop applications with conditional statements, loops, arrays, and methods. Over 70 PHP methods (functions) are introduced and demonstrated in coding examples.
- From the very first examples, the reader is introduced to object-oriented programming techniques. Many other books only briefly cover OO programming (if at all) in the final chapters.
- Object-oriented set methods are used to verify and filter user input. Many other books simply show a `set` method accepting data and storing it.
- A major objective of the book is to convince the reader to create all programs as secure and efficient as possible. The newest password encryption techniques (`password_hash`) are demonstrated.
- The `try` and `catch` methods are introduced to capture exceptions and some errors. The newest versions of PHP have been created to handle exceptions and errors using this approach. Many other books use `die` or other techniques to shut down a program.
- Multi-tier program design is introduced in the early chapters. This allows the reader to discover what logic and coding should take place in each tier. Many PHP books do not even cover this topic.

- The majority of the examples in the book are used to develop one main application (ABC Canine Shelter Reservation System). As the book progresses, the application is built from the beginning, in stages, showing the reader that application development should be broken into stages. Only after each stage is completed and tested, can the next stage begin. This approach works hand in hand with multi-tier design. Additional programming exercises and a term project are provided to enhance the understanding of development.
- The creation of user, change, and error logs are introduced. This allows the reader to gain an understanding of how to provide backup and recovery ability to keep an application functioning properly when security breaches or exceptions occur.
- The introduction of data objects and the data tier demonstrates to the reader the importance of creating an application that provides the ability to change data storage techniques and data storage location without requiring a major rewrite of the application. XML, JSON, and MySQL examples are provided.
- A natural relationship between PHP, HTML5, CSS3, and JavaScript is demonstrated throughout the book. This relationship is one of the major strengths of PHP.
- Throughout the book, web links are provided to point the user to additional resources to help understand the material or to dig deeper into the subject matter. Updates to link locations are provided on the book's web site.

Special Note for Teachers

The design of the content of this book provides flexibility in teaching styles and approaches. Each college and university approaches the initial education of programming concepts in different ways. This book provides three different types of programming exercises, which allow teachers to pick and choose what would work best in their environment. "Do It" exercises are provided in each chapter to allow the student to gain hands-on experience with techniques shown by modifying existing examples to produce the desired results. These exercises provide a level of confidence before the student attempts to program exercises at the end of the chapters. In addition, a Term Project is provided that builds an application that uses the same types of algorithms and programming techniques shown in the book.

Teaching tools, including test banks, course outline, and PowerPoint slides are available for use from the book's web site and from `apress.com`.

Code Examples, Images, and Links

Every effort has been made to catch any errors in code (and grammar). Please let us know if/when you discover problems in this book. Please send all corrections to Steve Prettyman (`steve_prettyman@hotmail.com`).

All code examples, images, and links are available for download from `apress.com` and the following location. You can download code examples from either web site. Copying code from the book may cause errors due to format requirements for publishing.

Book's web site: `www.littleoceanwaves.com/securephp/`

Chapter Overview

Chapter 1: An Introduction to PHP 7

After completing this chapter, the student will be able to:

- Understand the difference between LAMP, WAMP, and MAMP
- Successfully install a version of LAMP, WAMP, or MAMP
- Search the Internet for troubleshooting problems
- Explain the difference between a programming language and a scripting language
- Create an error-free simple PHP program

Chapter 2: Interfaces, Platforms, Containers, Three Tier Programming

After completing this chapter, the student will be able to:

- Give examples of platforms or containers that can host PHP programs
- Create a simple, dynamic web application using PHP
- Explain three-tier design and determine what is contained in each tier
- Design a three-tier application
- Explain each step of the program development life cycle (PDLC)
- Define and explain MVC and dependency injection

Chapter 3: Modular Programming

After completing this chapter, the student will be able to:

- Create an error-free simple objected-oriented (OO) modular PHP program
- Create a PHP class and make an instance of the class (object)
- Create an OO PHP encapsulated program, including `get` and `set` methods
- Create PHP methods (functions) that accept parameters and return information
- Create PHP public and private properties (variables)
- Import existing PHP code from another file or library into a program
- Validate information received using ternary (conditional) operators

Chapter 4: Secure User Interfaces

After completing this chapter, the student will be able to:

- Explain why user input must be validated in the interface and business rules tiers
- Explain why user input must be filtered in the business rules tier
- Use HTML5 code to validate user input
- Use JavaScript code to validate user input
- Use PHP `if` statements (conditional statements) to validate and filter input
- Use `foreach` loops to dynamically create an HTML select box from an XML file
- Use simple arrays for filtering and validation
- Pass simple arrays into methods (functions)
- Understand how to use dependency injection to control code version changes

Chapter 5: Handling and Logging Exceptions

After completing this chapter, the student will be able to:

- Explain the difference between errors and exceptions
- Create a PHP program that can handle general exceptions
- Create a PHP program that can create, raise, and handle user exceptions
- Explain and use a `switch` and/or embedded `if/else` statement
- Create a PHP program that uses the `while` loop and/or the `for` loop
- Create a program that reads/updates a text file using a two-dimensional array
- Create a PHP program that logs exceptions and e-mails support personnel

Chapter 6: Data Objects

After completing this chapter, the student will be able to:

- Create a data class that inserts, updates, and deletes XML or JSON data
- Explain how to create a data class that updates MySQL Data using a SQL Script
- Create a PHP program that creates a change backup log
- Create a PHP program that can recover data from a previous backup
- Apply changes to create up-to-date valid information
- Use dependency injection to attach a data class to another class in the BR tier
- Create a three-tier PHP application

Chapter 7: Authentication

After completing this chapter, the student will be able to:

- Define sessions and explain how sessions are used for authentication
- Create a PHP program that authenticates user logon
- Create a PHP program that register users
- Create a PHP program that will allow users to change passwords
- Create a PHP program that logs invalid login attempts

Chapter 8: Multifunctional Interfaces

After completing this chapter, the student will be able to:

- Create a complete PHP application that deletes, updates, and inserts data
- Create a professional look to a completed application using CSS
- Use JavaScript to accept and manipulate data from another program
- Secure all programs within an application requiring user IDs/passwords
- Populate HTML objects with values from a JSON object

CHAPTER 1

An Introduction to PHP 7

"PHP is a popular general-purpose scripting language that is especially suited to web development. Fast, flexible, and pragmatic, PHP powers everything from your blog to the most popular web sites in the world." —`www.php.net`

Chapter Objectives/Student Learning Outcomes

After completing this chapter, the student will be able to:

- Understand the differences between LAMP, WAMP, and MAMP
- Successfully install a version of LAMP, WAMP, or MAMP
- Search the Internet for troubleshooting problems
- Explain the difference between a programming language and a scripting language
- Create an error-free simple PHP program

PHP 5.5+, PHP 7+, and PHP.NET

Today, **PHP** (Hypertext Preprocessor) is one of the most popular languages used for web application development. The language has evolved to allow the programmer to quickly develop well-formed error-free programs using both **procedural** and **objected-oriented** programming techniques. It provides the ability to use many preexisting libraries of code that either come with the basic installation or can be installed within the PHP environment. This gives you multiple ways to complete a particular task. It provides more flexibility than many other languages. The ease with which additional libraries of code can be added to the environment is one of the many driving forces in its popularity.

Procedural language—A procedural programming language includes functions/methods that can be called from the main flow of the program. The flow of the program jumps to the function/method, executes the code within the module, and then returns to the next statement in the main flow of the program. Some Procedural languages include a main function/method that automatically is called when the program is executed.

Electronic supplementary material The online version of this chapter (doi:10.1007/978-1-4842-1730-6_1) contains supplementary material, which is available to authorized users.

© Steve Prettyman 2015
S. Prettyman, *Learn PHP 7*, DOI 10.1007/978-1-4842-1730-6_1

Object-oriented language—An object-oriented language uses classes and objects. Classes are similar to blue prints. A class describes what an object can contain, including properties/variables and functions/methods. An object is an instance of a class (like a building that has been created from a blueprint). Object-oriented languages provide polymorphism, encapsulation, and inheritance. Objects are naturally encapsulated by containing all related functions/methods and properties/variables within the object itself. Polymorphism allows duplicate method/function names within object-oriented objects. However, the "signature" must be different. The "signature" is the combination of the types of variables (numbers and characters) passed into the method/function and the type of information passed out of a method/function. For example, several `add` *methods could be created—one that only accepts integers (whole numbers), one that only accepts floating point numbers (numbers with decimals), and one that accepts a combination. The program will determine which method/function to call by what has been passed into the method/function. Inheritance in object-oriented programming allows an object to inherit properties/variables and functions/methods from another object. The object can also override those items inherited. This is similar to a child inheriting characteristics from the parents. Object-oriented languages can also be event-driven. An event-driven program will "sleep" until an event occurs. This is similar to an ATM machine program waiting for a user to input an ATM card.*

PHP is an **open source** language. As such, each version of the language is created using input from the individuals who use it—the programmers themselves. This allows the language, over time, to evolve and float into the direction that is driven by the users. From its first release in 1995 as a Personal Home Page Tool (PHP) by Rasmus Lerdorf, the versions have been released on the Internet with forums to provide users the ability to make suggestions and even provide code changes and additions. Today `www.php.net` is the official PHP web site.

Open source language—An open source programming language is developed by a community of interested parties. The community accepts input from fellow programmers for suggested upgrades and corrections. Several members of the community work together to present proposals and to make changes to the language. Open source languages are "free". A non-open source language (such as Microsoft C#) is created and updated by a company or major organization. Non-open source languages are not usually "free".

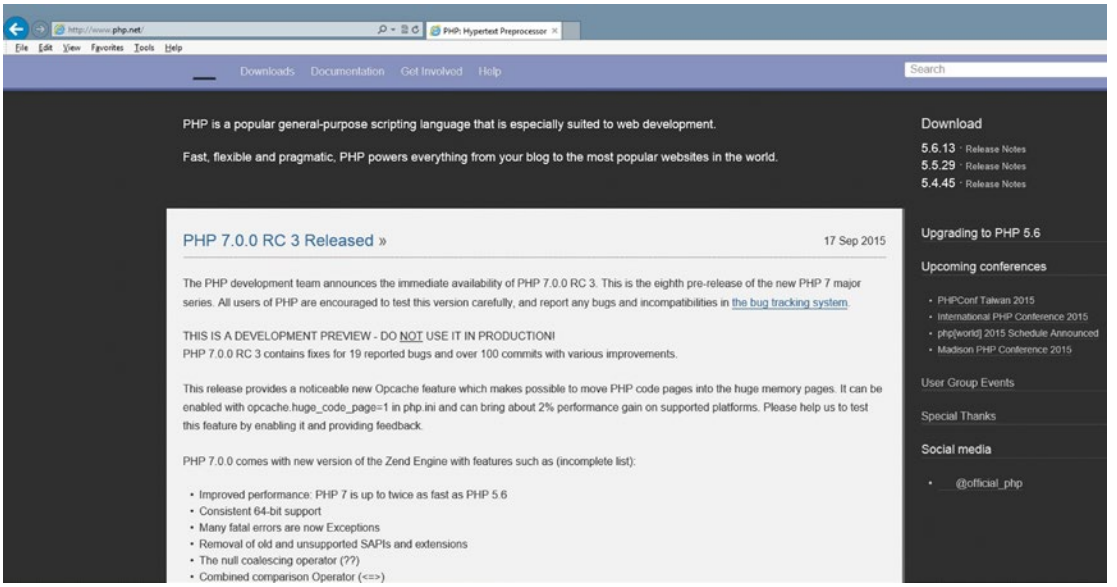

Figure 1-1. *PHP.NET (09/24/15)*

The www.php.net home page provides information on each of the latest releases of the language. It also provides information on future releases, the features planned for those releases, and the planned release dates. In addition, other related PHP information can be found, including links and information to major PHP conferences.

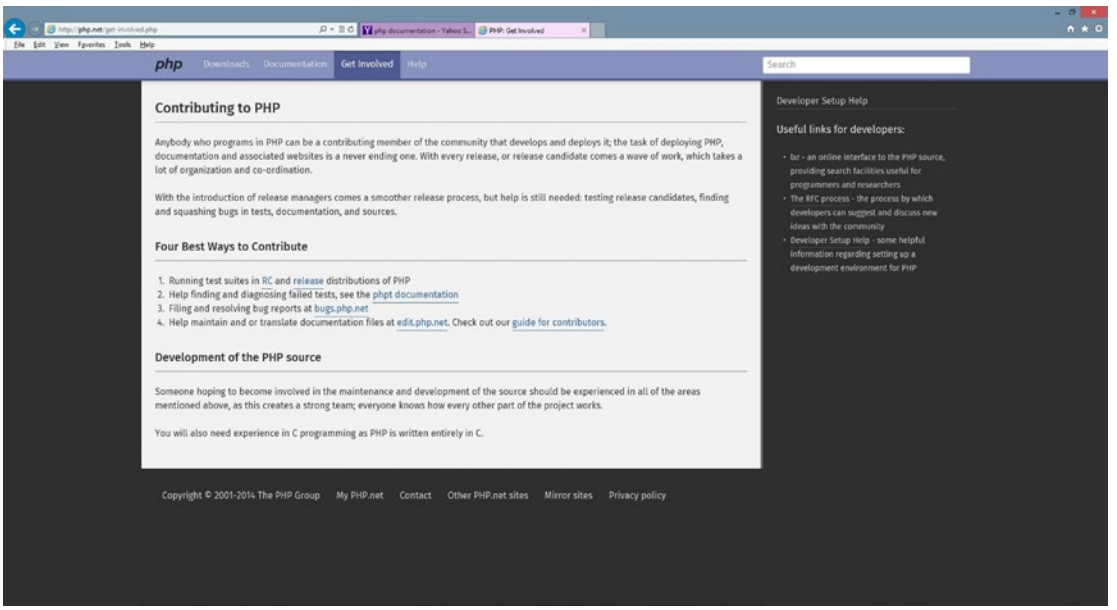

Figure 1-2. *Get involved (09/24/15)*

As mentioned, this site provides the ability for users to help with the future development of the language. Users can get involved with testing beta versions and reporting errors or program bugs. Visitors can also view documentation related to the development of possible future versions. This is a good way of discovering future enhancements or security fixes before major announcements have been made to the public.

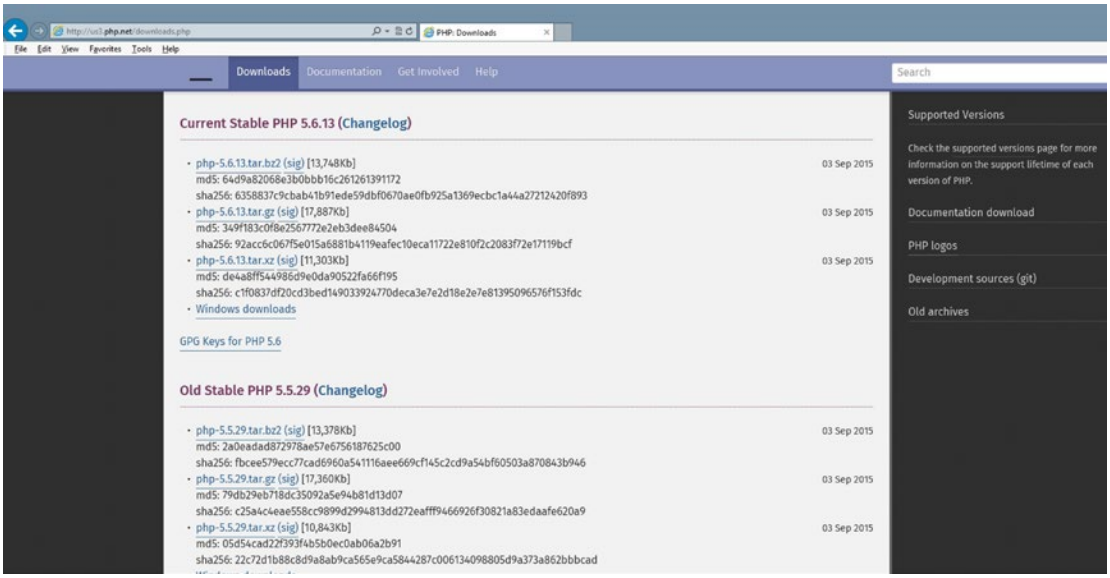

Figure 1-3. *Download page (09/24/15)*

The download page, as you might have guessed, provides the ability to gain easy access to the latest versions of the language. However, as you will note, only the language itself is provided. It is more common, and recommended, that the beginning user use a **WAMP** (Windows, Apache, MySQL, PHP); **LAMP** (Linux, Apache, MySQL, PHP); or **MAMP** (Mac, Apache, MySQL, PHP) package for initial installation. These packages (which we will look at later) allow for easy installation of multiple products at the same time. Otherwise, you have to run many separate installations, which can become complicated and error-prone if incompatible versions are installed.

> *WAMP/LAMP/MAMP—Open source (free) combinations, including Apache Web Server, MySQL, and PHP for a specific operating system (Windows, Linux, and Mac). These packages are open source. The combination of software is used for creating dynamic web sites and web applications.*

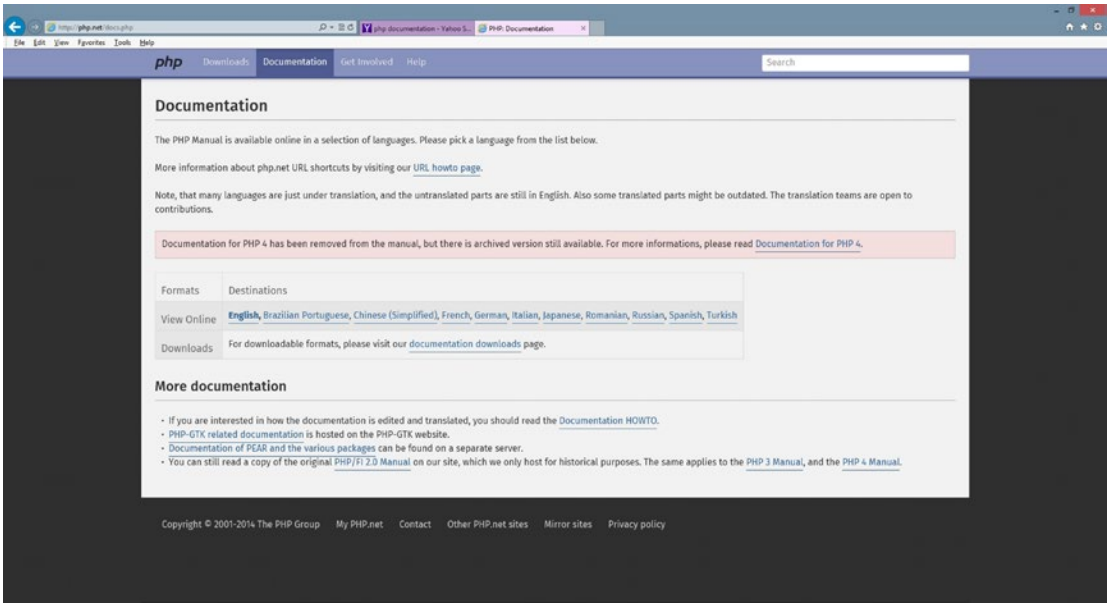

Figure 1-4. *Documentation pages (09/24/15)*

One of the more important pages of the PHP web site is the documentation page. This page allows users to search for descriptions and functionality of the language itself. You can also download the complete documentation. However, since this is a "live" site, with possible changes occurring, the most current information is best obtained by directly accessing it from the web site.

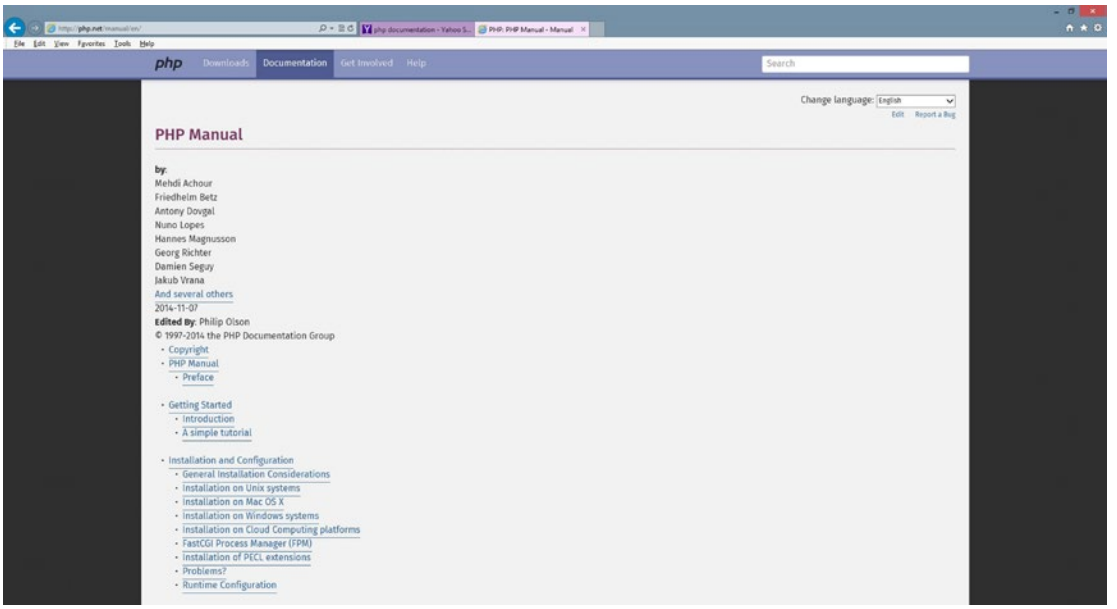

Figure 1-5. *The Manual (11/11/14)*

You can use the manual as if it were a textbook by clicking through each link from the beginning. The limited amount of explanation provided with each section of the manual might cause a beginner to want to give up on programming and change interests to something ghastly like networking! The manual does provide a great guide for experienced programmers, as the syntax of the language is similar to other languages such as JavaScript, Perl, and Java.

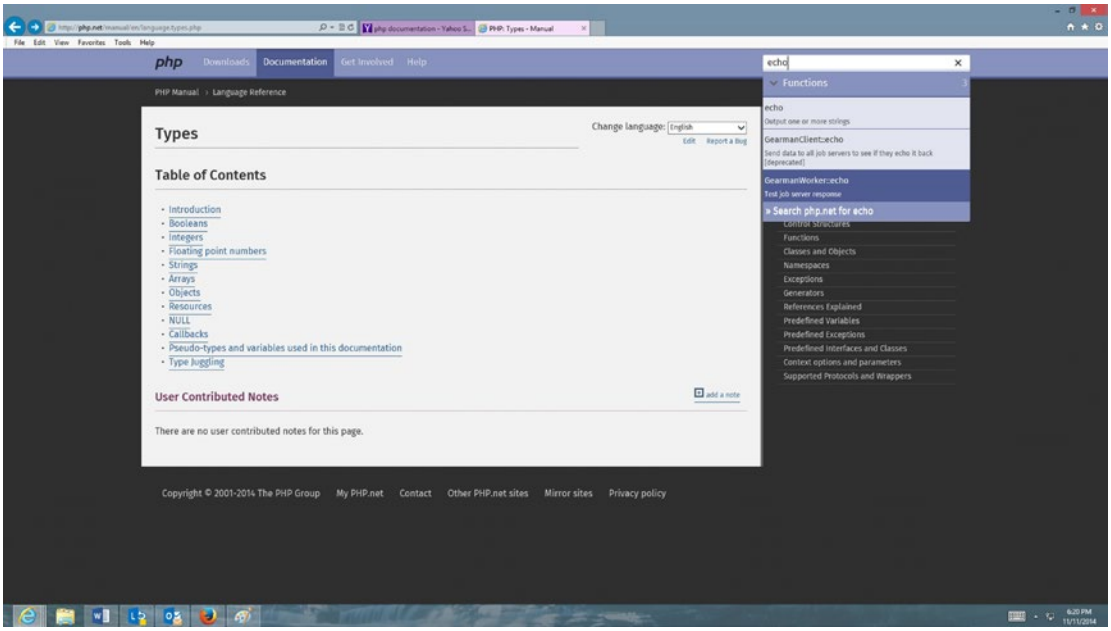

Figure 1-6. *Search (11/11/14)*

On any page of the web site, the user can enter a term, an expression, or even a function name to find more information. As the information is entered in search box, the web page will provide the user with one or more options below the box for the user to select.

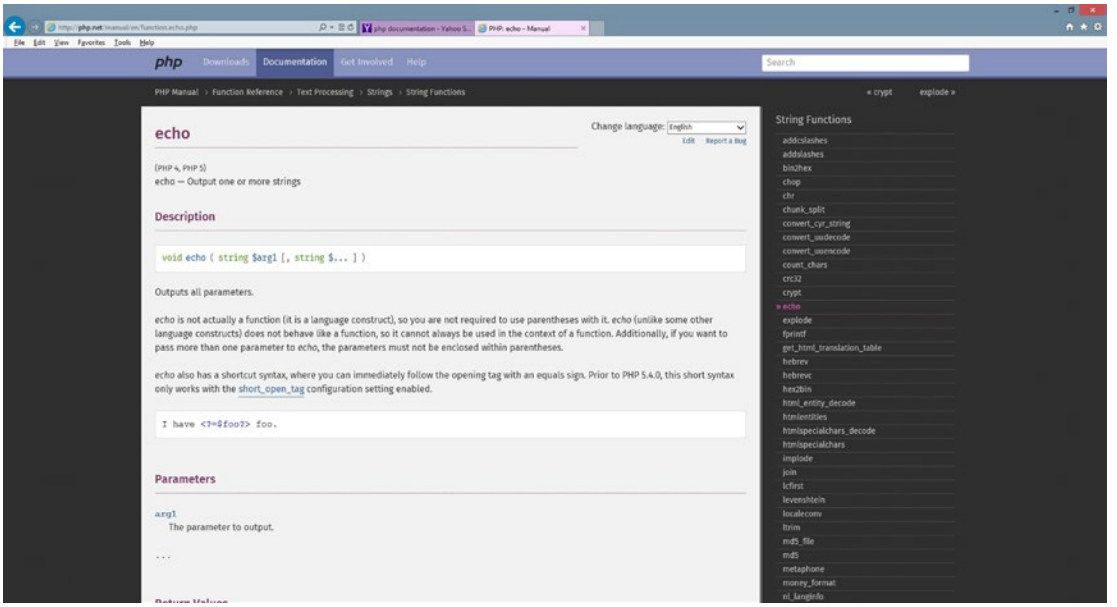

Figure 1-7. *Echo (11/11/14)*

Once the user has selected an option (such as echo shown in Figure 1-7), the results of the search provide the user with a general description of the item requested, any inputs or outputs for a function (parameters), and example code.

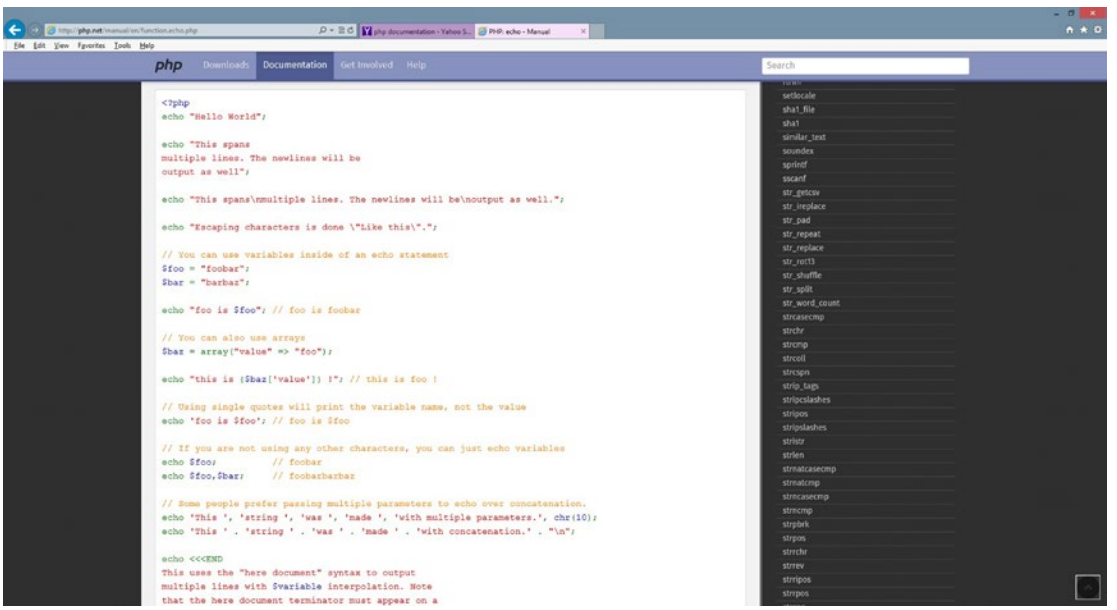

Figure 1-8. *Echo code (11/11/14)*

The example code provides explanations of the use of the function within the code itself by using comments (indicated by the `//` and gold color in Figure 1-8). The comments are not executable code. The executable code is color-coded to highlight strings (red), variables (blue), keywords (green), and the PHP opening and closing tags (blue). Color-coding helps make the code more readable. It also can make it easier to find syntax errors when creating programs. Many PHP editors provide similar color schemes.

PHP 5.6+ and PHP 7+

With the release of the PHP 7 environment, great improvements have taken place. PHP 5.5+ has dramatically improved security. In this book, we will use the newest PHP encryption tool "password hash" instead of MD5 which many current books use. Over the last several years, MD5 has proven to be vulnerable to hacking.

> *"PHP 7 is based on the PHPNG project (PHP Next-Gen), that was led by Zend to speed up PHP applications. The performance gains realized from PHP 7 are huge! They vary between 25% and 70% on real-world apps, and all of that just from upgrading PHP, without having to change a single line of code!"* —`www.zend.com`

PHP 7 also replaces fatal errors, which previously would crash a program, with exceptions that can be handled within the program itself.

If you are migrating from a previous version of PHP to PHP 7, please review the following link:

`http://php.net/manual/en/migration70.php`

The code used in the examples in this book is compatible with PHP 7. Most examples are also compatible with PHP 5.5 and PHP 5.6.

Do It

1. Go to `www.php.net`. Search for information on the `print` and `printf` functions. How are these functions similar? How are they different?
2. How do you "join the team" and help with the creation of the next version of PHP? Hint: Go to the "Get Involved" section of `www.php.net`, select "Guide for Contributors," and then find the "Join the team" link. Of course, the web site changes, so you may need to find a different route to the information.
3. Which ways can the `www.php.net` web site be useful for a beginning PHP programmer?
4. What language is used to create PHP? Hint: The answer is somewhere on the `www.php.net` web site.
5. Go to `www.php.net`. Search for the list of improvements and changes with PHP 7. List those improvements and changes. Which of these do you think will affect a beginning level programmer?

PHP, JavaScript, CSS, HTML, and Apache Web Server

PHP is a scripting language. A **scripting language** is different than an actual programming language. **Programming languages** (such as Java) are written by the programmer in an English-like syntax. The program is compiled, which means it's converted from the English syntax into machine code (0s and 1s). This code is then executed (run) within a compatible operating system and hardware. Scripting languages do not use a compiler. The first time the code is accessed it is interpreted line by line as the program is executed.

You may wonder if this causes the code to be slower than compiled code. The answer is no. Once the code has been executed once, the interpreted code stays in the memory of the computer, or server, for other executions. If the programmer changes this code, a new version will replace the previous version in memory.

JavaScript is also a scripting language. As you may be aware, JavaScript code can be seen within a **web browser** by viewing the source, as shown in Figure 1-9.

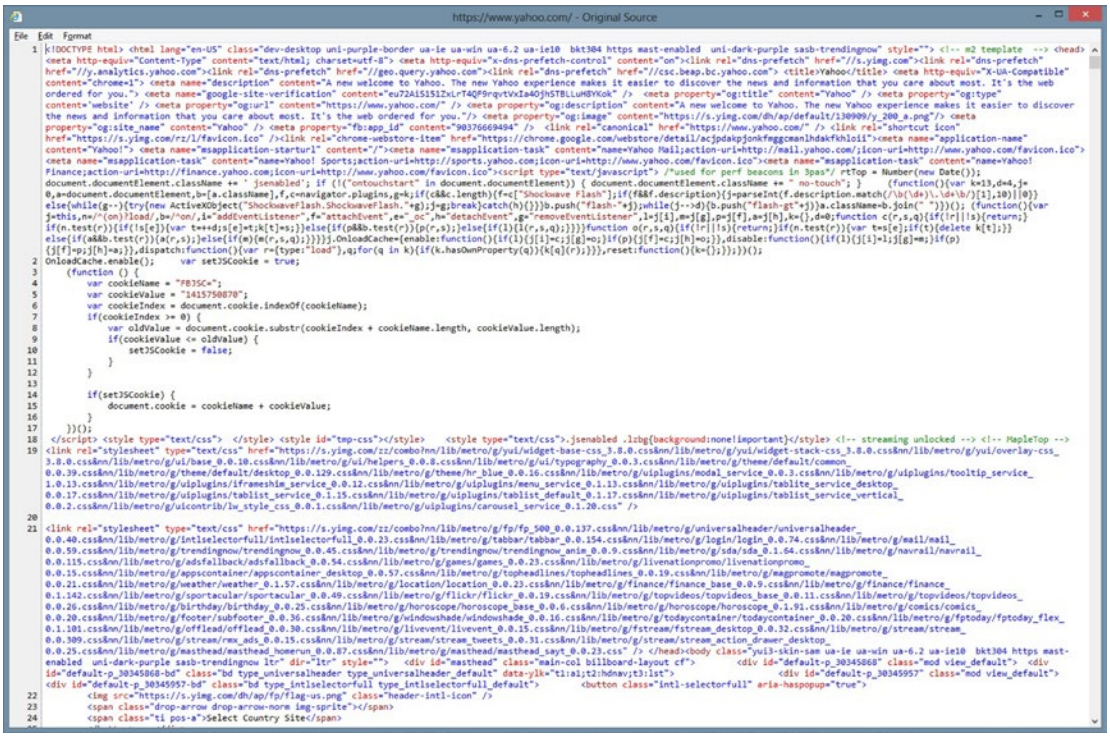

Figure 1-9. *JavaScript, HTML, and CSS code from yahoo.com (11/11/14)*

The **source code** displayed in Figure 1-9 is from `www.yahoo.com` and it shows a combination of several languages, including **HTML**, **CSS**, and JavaScript. The JavaScript code (displayed in black) is located between **script tags** (`<script type="text/JavaScript">` and `</script>`). This JavaScript code will attempt to create a cookie on your machine, if your browser allows cookies.

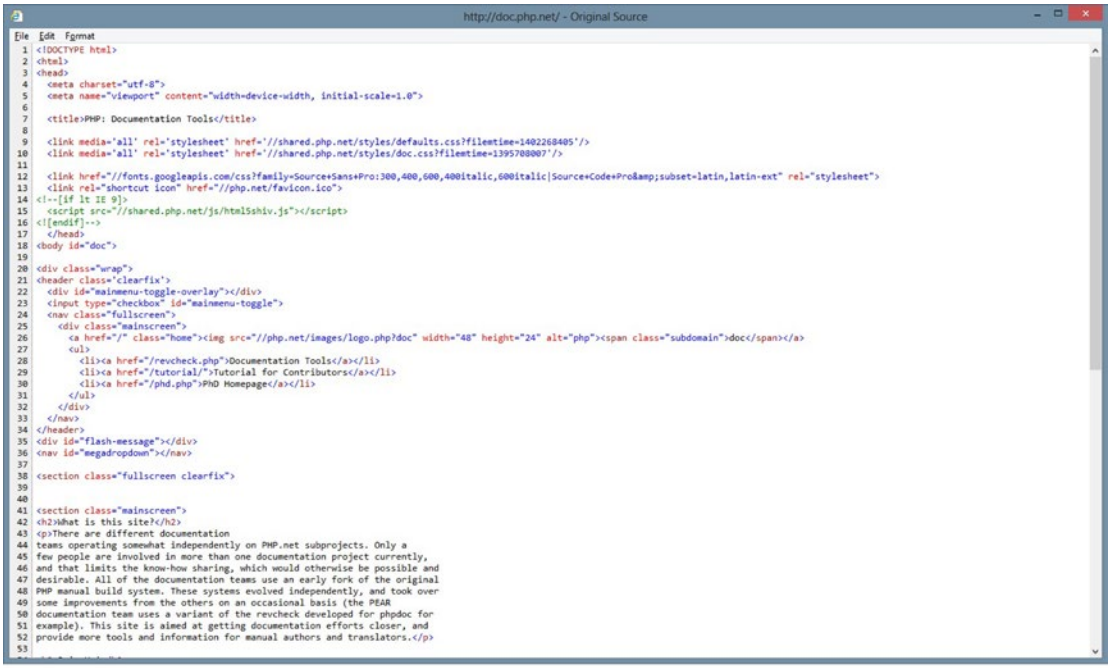

Figure 1-10. *The `www.php.net` source code (11/11/14)*

However, when we look at the `www.php.net` source code (in Figure 1-10), we cannot see any PHP script code. There are links to some PHP files present, but no actual PHP code is displayed. Why?

JavaScript code resides on the user's computer. It is interpreted and executed within the browser. PHP code resides on a **web server**. The code is also interpreted and executed, but by the web server, not by the browser. The results of executing the PHP code are returned to the browser, not to the actual code itself.

```
<?php
Print "Hello";
?>
```

■ **Note** You may notice other formats for using PHP (such as: `<%`, `<%=`, `%>`, or `<script language="php">`); with PHP 7 these styles are no longer valid. There were actually depreciated previously, but still usuable.

You might guess that this code will display `Hello`. While this is correct, the question is, what processes happen to produce this result?

If this code is placed in a file (such as `hello.php`) on a web server, we would use our web browser to request this file by entering its name and location in the URL (address) box (such as `http://servera.com/hello.php`). The address entered instructs the browser to send an HTTP `Get` request to the web server (`servera.com`) to return the web page (`hello.php`).

Figure 1-11. *Requesting an HTML/JavaScript web page*

The web server receiving the request will determine that PHP code must first be interpreted and executed. It determines this simply by looking at the file extension (`.php`) of the file requested. Any PHP code within the file is then sent to the PHP processor for interpretation and execution. The results of the execution of the code is returned to the web server, which in turn sends it (and any other HTML and/or JavaScript code) back to the browser. In this example, `Hello` would be returned and displayed by the browser. If we then viewed the source code, as mentioned, we would only see the actual word `Hello`. We would not see any HTML or PHP. Why? Because we did not send any HTML back to the browser.

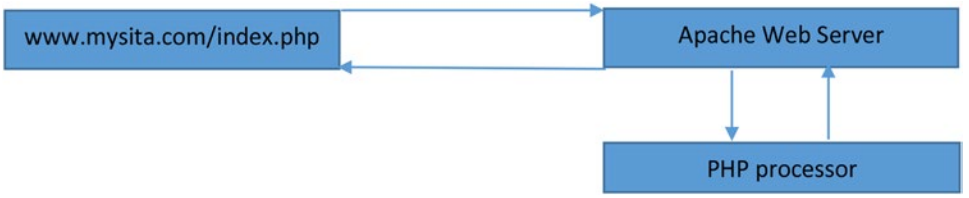

Figure 1-12. *Requesting a web page with PHP code*

You may be wondering if you can use this process to send back actual HTML (and/or JavaScript) code to create a dynamic web page. The answer is yes. The PHP `print` function will return any HTML (or JavaScript) code that has been placed between the `""`. The browser will interpret any code returned by the web server.

> *Print function. The print function is not actually a function. It is a language construct. Functions require that strings be included in quotes when passed. Language constructs do not require quotes around strings. However, it is still recommended. Print will pass whatever has been passed into it to the browser. It will attempt to convert any item that is not a string to string (text) format since all items displayed within a web page are in text format.*
>
> *For more information, visit:*
>
> *http://php.net/manual/en/function.print.php.*
>
> *For a more in-depth explanation of the print command, visit the free "The New Boston" (thenewboston.com) video(s) at:*
>
> *https://www.thenewboston.com/videos.php?cat=11&video=16996.*

■ **Note** All links provided in this book can be accessed from http://www.littleoceanwaves.com/securephp.

```
<?php
Print "<h1>Hello</h1>";
?>
```

If we change our code to the listing above, the browser will display `Hello` as an HTML header (`h1`). The disadvantage of using the `print` function is that the program will have no control over where the statement is displayed on the web page. The statement will actually be displayed as the first line of code, even before any other existing HTML tags. This might be okay if we are just returning a statement to the user, such as "Your process has been completed". However, this might not be acceptable if your goal is to format output at an exact location on the page. There are other techniques and functions that we could choose to eliminate this problem. However, it is beyond our current discussion.

Now that we know we must interpret and execute PHP code with the help of a web server, what server should we use?

The Apache web server is the server that is most commonly used to host and handle PHP web page requests. Like other web servers, Apache can also accept and return requests for other types of files, including HTML, JavaScript, PERL, images, and RSS feeds. Apache, as mentioned, determines what processes need to be completed from HTTP requests by first looking at the file extensions of the requested files.

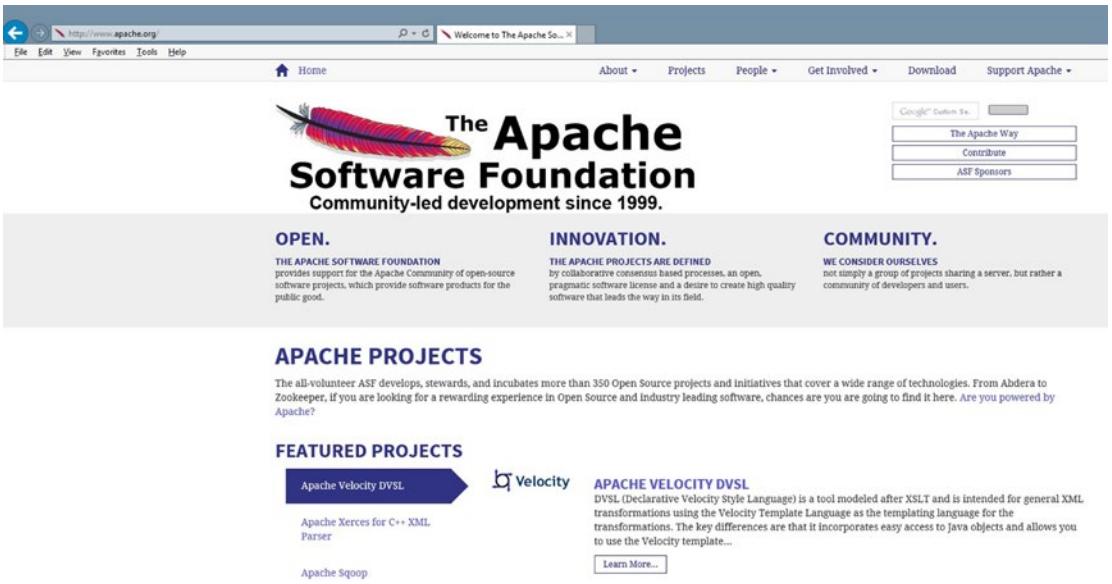

Figure 1-13. *Apache.org web site (09/24/15)*

Apache, like PHP, is an open source product. All changes to the Apache web server are coordinated by the Apache Software Foundation. ASP maintains the `apache.org` web site to provide users and developers the ability to discover projects currently under development and the ability to download the latest versions of Apache. However, as mentioned, downloading separate versions of PHP, Apache, and MySQL can cause issues with incompatible versions. Unless you know what you are doing, it's much wiser to download a complete WAMP, LAMP, or MAMP version.

Figure 1-14. *Apache's Get Involved (09/24/15)*

The Apache Software Foundation also encourages all users of their products to keep up to date and to get involved in the development of future products. Users are encouraged to join discussion and mailing groups, test out new releases, and even help fix bugs or add new features to their products.

Do It

1. What are the differences in executing PHP code compared to executing Java code?
2. What is the difference between a scripting language and a programming language? What type of language is PHP?
3. How does the Apache web server handle requests for a PHP web page?
4. Why can we see JavaScript code within a web browser but we can't see PHP code?
5. Go to `www.apache.org`. What are some of the ways that you can become involved with the development of Apache projects, even though you have limited experience?