VISUAL QUICKPRO GUIDE

Get up and running in no time!

PHP and MySQL for Dynamic Web Sites

Fourth Edition

LARRY ULLMAN

LEARN THE QUICK AND EASY WAY!

Table of Contents

Copyright 1
Special Thanks to: 3
Introduction 9
Chapter 1. Introduction to PHP 19
Basic Syntax 20
Sending Data to the Web Browser 24
Writing Comments 28
What Are Variables? 32
Introducing Strings 36
Concatenating Strings 39
Introducing Numbers 41
Introducing Constants 44
Single vs. Double Quotation Marks 47
Basic Debugging Steps 50
Review and Pursue 52
Chapter 2. Programming with PHP 53
Creating an HTML Form 54
Handling an HTML Form 59
Conditionals and Operators 63
Validating Form Data 67
Introducing Arrays 72
For and While Loops 87
Review and Pursue 90
Chapter 3. Creating Dynamic Web Sites 93
Including Multiple Files 94
Handling HTML Forms, Revisited 103
Making Sticky Forms 109
Creating Your Own Functions 113
Review and Pursue 128
Chapter 4. Introduction to MySQL 129
Naming Database Elements 130
Choosing Your Column Types 132
Choosing Other Column Properties 136
Accessing MySQL 139
Review and Pursue 146
Chapter 5. Introduction to SQL 147
Creating Databases and Tables 148
Inserting Records 151
Selecting Data 156
Using Conditionals 158
Using LIKE and NOT LIKE 161
Sorting Query Results 163
Limiting Query Results 165
Updating Data 167
Deleting Data 169
Using Functions 171
Review and Pursue 182
Chapter 6. Database Design 183
Normalization 184
Creating Indexes 197
Using Different Table Types 200
Languages and MySQL 202
Time Zones and MySQL 207
Foreign Key Constraints 213
Review and Pursue 220
Chapter 7. Advanced SQL and MySQL 221
Performing Joins 222
Grouping Selected Results 232
Advanced Selections 236
Performing FULLTEXT Searches 240
Optimizing Queries 248
Performing Transactions 252
Database Encryption 255
Review and Pursue 258
Chapter 8. Error Handling and Debugging 259
Error Types and Basic Debugging 260

Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

Displaying PHP Errors.... 266
Adjusting Error Reporting in PHP.... 268
Creating Custom Error Handlers.... 271
PHP Debugging Techniques.... 276
SQL and MySQL Debugging Techniques.... 280
Review and Pursue.... 282

Chapter 9. Using PHP with MySQL.... 283

Modifying the Template.... 284
Connecting to MySQL.... 286
Executing Simple Queries.... 291
Retrieving Query Results.... 299
Ensuring Secure SQL.... 303
Counting Returned Records.... 308
Updating Records with PHP.... 310
Review and Pursue.... 316

Chapter 10. Common Programming Techniques.... 317

Sending Values to a Script.... 318
Using Hidden Form Inputs.... 322
Editing Existing Records.... 327
Paginating Query Results.... 334
Making Sortable Displays.... 341
Review and Pursue.... 346

Chapter 11. Web Application Development.... 347

Sending Email.... 348
Handling File Uploads.... 354
PHP and JavaScript.... 366
Understanding HTTP Headers.... 373
Date and Time Functions.... 380
Review and Pursue.... 384

Chapter 12. Cookies and Sessions.... 385

Making a Login Page.... 386
Making the Login Functions.... 389
Using Cookies.... 394
Using Sessions.... 406
Improving Session Security.... 414
Review and Pursue.... 418

Chapter 13. Security Methods.... 419

Preventing Spam.... 420
Validating Data by Type.... 427
Validating Files by Type.... 432
Preventing XSS Attacks.... 436
Using the Filter Extension.... 439
Preventing SQL Injection Attacks.... 443
Review and Pursue.... 450

Chapter 14. Perl-Compatible Regular Expressions.... 451

Creating a Test Script.... 452
Defining Simple Patterns.... 456
Using Quantifiers.... 459
Using Character Classes.... 461
Finding All Matches.... 464
Using Modifiers.... 468
Matching and Replacing Patterns.... 470
Review and Pursue.... 474

Chapter 15. Introducing jQuery.... 475

What is jQuery?.... 476
Incorporating jQuery.... 478
Using jQuery.... 481
Selecting Page Elements.... 484
Event Handling.... 487
DOM Manipulation.... 491
Using Ajax.... 497
Review and Pursue.... 510

Chapter 16. An OOP Primer.... 511

Fundamentals and Syntax.... 512
Working with MySQL.... 515
The DateTime Class.... 529
Review and Pursue.... 536

Chapter 17. Example—Message Board.... 537

Making the Database.... 538
Writing the Templates.... 547

Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

Creating the Index Page.... 555
Creating the Forum Page.... 556
Creating the Thread Page.... 561
Posting Messages.... 566
Review and Pursue.... 576
Chapter 18. Example—User Registration.... 577
Creating the Templates.... 578
Writing the Configuration Scripts.... 584
Creating the Home Page.... 592
Registration.... 594
Activating an Account.... 604
Logging In and Logging Out.... 607
Password Management.... 612
Review and Pursue.... 622
Chapter 19. Example—E-Commerce.... 623
Creating the Database.... 624
The Administrative Side.... 630
Creating the Public Template.... 647
The Product Catalog.... 651
The Shopping Cart.... 663
Recording the Orders.... 672
Review and Pursue.... 677
Appendix A. Installation.... 679
Installation on Windows.... 679
Installation on Mac OS X.... 683
Managing MySQL Users.... 686
Testing Your Installation.... 691
Configuring PHP.... 693
Configuring Apache.... 696
bvdindexIndex.... 707

Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

VISUAL QUICKPRO GUIDE

PHP and MySQL for Dynamic Web Sites

Fourth Edition

LARRY ULLMAN

Peachpit Press

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

Visual QuickPro Guide
PHP and MySQL for Dynamic Web Sites, Fourth Edition
Larry Ullman

Peachpit Press
1249 Eighth Street
Berkeley, CA 94710
510/524-2178
510/524-2221 (fax)

Find us on the Web at: www.peachpit.com
To report errors, please send a note to: errata@peachpit.com
Peachpit Press is a division of Pearson Education.

Copyright © 2012 by Larry Ullman

Editor: Rebecca Gulick
Copy Editor: Patricia Pane
Technical Reviewer: Anselm Bradford
Production Coordinator: Myrna Vladic
Compositor: Debbie Roberti
Proofreader: Bethany Stough
Indexer: Valerie Haynes-Perry
Cover Design: RHDG / Riezebos Holzbaur Design Group, Peachpit Press
Interior Design: Peachpit Press
Logo Design: MINE™ www.minesf.com

Licensed by
Dena Laney
2335846

Notice of Rights

All rights reserved. No part of this book may be reproduced or transmitted in any form by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher. For information on getting permission for reprints and excerpts, contact permissions@peachpit.com.

Notice of Liability

The information in this book is distributed on an “As Is” basis, without warranty. While every precaution has been taken in the preparation of the book, neither the author nor Peachpit Press shall have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the instructions contained in this book or by the computer software and hardware products described in it.

Trademarks

Visual QuickPro Guide is a registered trademark of Peachpit Press, a division of Pearson Education. MySQL is a registered trademark of MySQL AB in the United States and in other countries. Macintosh and Mac OS X are registered trademarks of Apple, Inc. Microsoft and Windows are registered trademarks of Microsoft Corp. Other product names used in this book may be trademarks of their own respective owners. Images of Web sites in this book are copyrighted by the original holders and are used with their kind permission. This book is not officially endorsed by nor affiliated with any of the above companies, including MySQL AB.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and Peachpit was aware of a trademark claim, the designations appear as requested by the owner of the trademark. All other product names and services identified throughout this book are used in editorial fashion only and for the benefit of such companies with no intention of infringement of the trademark. No such use, or the use of any trade name, is intended to convey endorsement or other affiliation with this book.

ISBN-13: 978-0-321-78407-0
ISBN-10: 0-321-78407-3

9 8 7 6 5 4 3 2 1

Printed and bound in the United States of America

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

Dedication

Dedicated to the fine faculty at my alma mater, Northeast Missouri State University. In particular, I would like to thank: Dr. Monica Barron, Dr. Dennis Leavens, Dr. Ed Tyler, and Dr. Cole Woodcox, whom I also have the pleasure of calling my friend. I would not be who I am as a writer, as a student, as a teacher, or as a person if it were not for the magnanimous, affecting, and brilliant instruction I received from these educators.

Special Thanks to:

My heartfelt thanks to everyone at Peachpit Press, as always.

My gratitude to editor extraordinaire Rebecca Gulick, who makes my job so much easier. And thanks to Patricia Pane for her hard work, helpful suggestions, and impressive attention to detail. Thanks also to Valerie Haynes-Perry for indexing and Myrna Vladic and Deb Roberti for laying out the book, and thanks to Anselm Bradford for his technical review.

Kudos to the good people working on PHP, MySQL, Apache, phpMyAdmin, MAMP, and XAMPP, among other great projects. And a hearty "cheers" to the denizens of the various newsgroups, mailing lists, support forums, etc., who offer assistance and advice to those in need.

Thanks, as always, to the readers, whose support gives my job relevance. An extra helping of thanks to those who provided the translations in Chapter 17, "Example—Message Board," and who offered up recommendations as to what they'd like to see in this edition.

Thanks to Karnesha and Sarah for entertaining and taking care of the kids so that I could get some work done.

Finally, I would not be able to get through a single book if it weren't for the love and support of my wife, Jessica. And a special shout-out to Zoe and Sam, who give me reasons to, and not to, write books!

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

Table of Contents

Introduction ix

Chapter 1 Introduction to PHP. 1

Basic Syntax 2
Sending Data to the Web Browser. 6
Writing Comments. 10
What Are Variables?. 14
Introducing Strings 18
Concatenating Strings 21
Introducing Numbers 23
Introducing Constants 26
Single vs. Double Quotation Marks 29
Basic Debugging Steps. 32
Review and Pursue 34

Chapter 2 Programming with PHP 35

Creating an HTML Form 36
Handling an HTML Form 41
Conditionals and Operators 45
Validating Form Data 49
Introducing Arrays. 54
For and While Loops 69
Review and Pursue 72

Chapter 3 Creating Dynamic Web Sites. 75

Including Multiple Files 76
Handling HTML Forms, Revisited 85
Making Sticky Forms 91
Creating Your Own Functions 95
Review and Pursue 110

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

Chapter 4 Introduction to MySQL 111

Naming Database Elements 112
Choosing Your Column Types 114
Choosing Other Column Properties 118
Accessing MySQL 121
Review and Pursue 128

Chapter 5 Introduction to SQL 129

Creating Databases and Tables 130
Inserting Records 133
Selecting Data 138
Using Conditionals 140
Using LIKE and NOT LIKE 143
Sorting Query Results 145
Limiting Query Results 147
Updating Data 149
Deleting Data 151
Using Functions 153
Review and Pursue 164

Chapter 6 Database Design 165

Normalization 166
Creating Indexes 179
Using Different Table Types 182
Languages and MySQL 184
Time Zones and MySQL 189
Foreign Key Constraints 195
Review and Pursue 202

Chapter 7 Advanced SQL and MySQL 203

Performing Joins 204
Grouping Selected Results 214
Advanced Selections 218
Performing FULLTEXT Searches 222
Optimizing Queries 230
Performing Transactions 234
Database Encryption 237
Review and Pursue 240

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

Chapter 8 Error Handling and Debugging 241

Error Types and Basic Debugging 242
Displaying PHP Errors. 248
Adjusting Error Reporting in PHP 250
Creating Custom Error Handlers. 253
PHP Debugging Techniques 258
SQL and MySQL Debugging Techniques. 262
Review and Pursue 264

Chapter 9 Using PHP with MySQL 265

Modifying the Template. 266
Connecting to MySQL. 268
Executing Simple Queries 273
Retrieving Query Results 281
Ensuring Secure SQL 285
Counting Returned Records 290
Updating Records with PHP 292
Review and Pursue 298

Chapter 10 Common Programming Techniques 299

Sending Values to a Script 300
Using Hidden Form Inputs 304
Editing Existing Records 309
Paginating Query Results. 316
Making Sortable Displays 323
Review and Pursue 328

Chapter 11 Web Application Development 329

Sending Email 330
Handling File Uploads 336
PHP and JavaScript 348
Understanding HTTP Headers. 355
Date and Time Functions 362
Review and Pursue 366

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

Chapter 12 Cookies and Sessions 367

Making a Login Page 368
Making the Login Functions 371
Using Cookies 376
Using Sessions. 388
Improving Session Security 396
Review and Pursue 400

Chapter 13 Security Methods 401

Preventing Spam 402
Validating Data by Type. 409
Validating Files by Type. 414
Preventing XSS Attacks. 418
Using the Filter Extension 421
Preventing SQL Injection Attacks 425
Review and Pursue 432

Chapter 14 Perl-Compatible Regular Expressions. 433

Creating a Test Script 434
Defining Simple Patterns 438
Using Quantifiers 441
Using Character Classes 443
Finding All Matches 446
Using Modifiers 450
Matching and Replacing Patterns 452
Review and Pursue 456

Chapter 15 Introducing jQuery 457

What is jQuery? 458
Incorporating jQuery 460
Using jQuery 463
Selecting Page Elements. 466
Event Handling. 469
DOM Manipulation 473
Using Ajax 479
Review and Pursue 492

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

Chapter 16 An OOP Primer 493

Fundamentals and Syntax 494
Working with MySQL 497
The DateTime Class 511
Review and Pursue 518

Chapter 17 Example—Message Board 519

Making the Database 520
Creating the Index Page 537
Creating the Forum Page 538
Creating the Thread Page 543
Posting Messages 548
Review and Pursue 558

Chapter 18 Example—User Registration 559

Creating the Templates 560
Writing the Configuration Scripts 566
Creating the Home Page 574
Registration 576
Activating an Account 586
Logging In and Logging Out 589
Password Management 594
Review and Pursue 604

Chapter 19 Example—E-Commerce 605

Creating the Database 606
The Administrative Side 612
Creating the Public Template 629
The Product Catalog 633
The Shopping Cart 645
Recording the Orders 654
Review and Pursue 659

Index 661

BONUS APPENDIX

Appendix A Installation A1

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

Introduction

Today's Web users expect exciting pages that are updated frequently and provide a customized experience. For them, Web sites are more like communities, to which they'll return time and again. At the same time, Web-site administrators want sites that are easier to update and maintain, understanding that's the only reasonable way to keep up with visitors' expectations. For these reasons and more, PHP and MySQL have become the de facto standards for creating dynamic, database-driven Web sites.

This book represents the culmination of my many years of Web development experience coupled with the value of having written several previous books on the technologies discussed herein. The focus of this book is on covering the most important knowledge in the most efficient manner. It will teach you how to begin developing dynamic Web sites and give you plenty of example code to get you started. All you need to provide is an eagerness to learn.

Well, that and a computer.

What Are Dynamic Web Sites?

Dynamic Web sites are flexible and potent creatures, more accurately described as *applications* than merely sites. Dynamic Web sites

- Respond to different parameters (for example, the time of day or the version of the visitor's Web browser)
- Have a "memory," allowing for user registration and login, e-commerce, and similar processes
- Almost always integrate HTML forms, allowing visitors to perform searches, provide feedback, and so forth
- Often have interfaces where administrators can manage the site's content
- Are easier to maintain, upgrade, and build upon than statically made sites

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

There are many technologies available for creating dynamic Web sites. The most common are ASP.NET (Active Server Pages, a Microsoft construct), JSP (Java Server Pages), ColdFusion, Ruby on Rails (a Web development framework for the Ruby programming language), and PHP. Dynamic Web sites don't always rely on a database, but more and more of them do, particularly as excellent database applications like MySQL are available at little to no cost.

What is PHP?

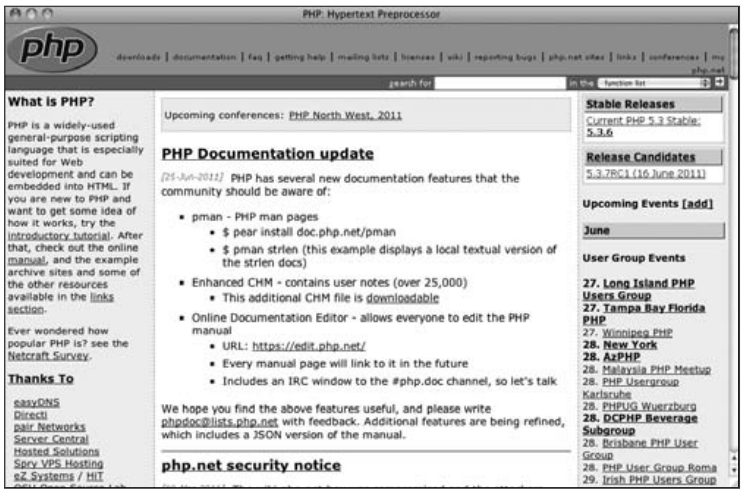
PHP originally stood for "Personal Home Page" as it was created in 1994 by Rasmus Lerdorf to track the visitors to his online résumé. As its usefulness and capabilities grew (and as it started being used in more professional situations), it came to mean "PHP: Hypertext Preprocessor."

According to the official PHP Web site, found at **`www.php.net`** Ⓐ, PHP is a "widely used general-purpose scripting language that is especially suited for Web development and can be embedded into HTML." It's a long but descriptive definition, whose meaning I'll explain.

Starting at the end of that statement, to say that PHP *can be embedded into HTML* means that you can take a standard HTML page, drop in some PHP wherever you need it, and end up with a dynamic result. This attribute makes PHP very approachable for anyone that's done even a little bit of HTML work.

Also, PHP is a *scripting* language, as opposed to a *compiled* language: PHP was designed to write Web scripts, not stand-alone applications (although, with some extra effort, you can now create applications in PHP). PHP scripts run only after an event occurs—for example, when a user submits a form or goes to a URL (Uniform Resource Locator, the technical term for a Web address).

I should add to this definition that PHP is a server-side, cross-platform technology, both descriptions being important. *Server-side* refers to the fact that everything PHP does occurs on the server. A Web server application, like Apache or Microsoft's IIS (Internet Information Services), is required and all PHP scripts must be accessed through a URL (**`http://`***something*). Its



Ⓐ The home page for PHP.

Licensed by Dena Laney 2335846
PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

What Happened to PHP 6?

When I wrote the previous version of this book, *PHP 6 and MySQL 5 for Dynamic Web Sites: Visual QuickPro Guide*, the next major release of PHP—PHP 6—was approximately 50 percent complete. Thinking that PHP 6 would therefore be released sometime after the book was published, I relied upon a beta version of PHP 6 for a bit of that edition's material. And then...
PHP 6 died.

One of the key features planned for PHP 6 was support for Unicode, meaning that PHP 6 would be able to work natively with any language. This would be a great addition to an already popular programming tool. Unfortunately, implementing Unicode support went from being complicated to quite difficult, and the developers behind the language tabled development of PHP 6. Not all was lost, however: Some of the other features planned for PHP 6, such as support for *namespaces* (an Object-Oriented Programming concept), were added to PHP 5.3.

At the time of this writing, it's not clear when Unicode support might be completed or what will happen with PHP 6. My hunch is that PHP will be making incremental developments along the version 5 trunk for some time to come.

cross-platform nature means that PHP runs on most operating systems, including Windows, Unix (and its many variants), and Macintosh. More important, the PHP scripts written on one server will normally work on another with little or no modification.

At the time this book was written, PHP was at version 5.3.6 and this book does assume you're using at least version 5.0. Some functions and features covered will require more specific or current versions, like PHP 5.2 or greater. In those cases, I will make it clear when the functionality was added to PHP, and provide alternative solutions if you have a slightly older version of the language.

If you're still using version 4 of PHP, you really should upgrade. If that's not in your plans, then please grab the second edition of this book instead.

More information about PHP can always be found at PHP.net or at Zend (`www.zend.com`), the minds behind the core of PHP.

Why use PHP?

Put simply, when it comes to developing dynamic Web sites, PHP is better, faster, and easier to learn than the alternatives. What you get with PHP is excellent performance, a tight integration with nearly every database available, stability, portability, and a nearly limitless feature set due to its extendibility. All of this comes at no cost (PHP is open source) and with a very manageable learning curve. PHP is one of the best marriages I've ever seen between the ease with which beginning programmers can start using it and the ability for more advanced programmers to do everything they require.

Finally, the proof is in the pudding: PHP has seen an exponential growth in use since its inception, and is the server-side

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

technology of choice on over 76 percent of all Web sites **B**. In terms of all programming languages, PHP is the fifth most popular **C**.

Of course, you might assume that I, as the author of a book on PHP (several, actually), have a biased opinion. Although not nearly to the same extent as PHP, I've also developed sites using Java Server Pages (JSP), Ruby on Rails (RoR), and ASP.NET. Each has its pluses and minuses, but PHP is the technology I always return to. You might hear that it doesn't perform or scale as well as other technologies, but Yahoo!, Wikipedia, and Facebook all use PHP, and you can't find many sites more visited or demanding than those.

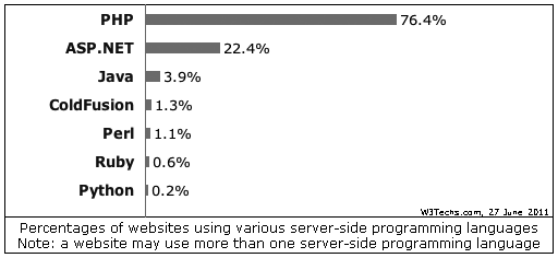
You might also wonder how secure PHP is. But *security isn't in the language*; it's in how that language is used. Rest assured that a complete and up-to-date discussion of all the relevant security concerns is provided by this book.

How PHP works

As previously stated, PHP is a server-side language. This means that the code you write in PHP sits on a host computer called a *server*. The server sends Web pages to the requesting visitors (you, the client, with your Web browser).

When a visitor goes to a Web site written in PHP, the server reads the PHP code and then processes it according to its scripted directions. In the example shown in **D**, the PHP code tells the server to send the appropriate data—HTML code—to the Web browser, which treats the received code as it would a standard HTML page.

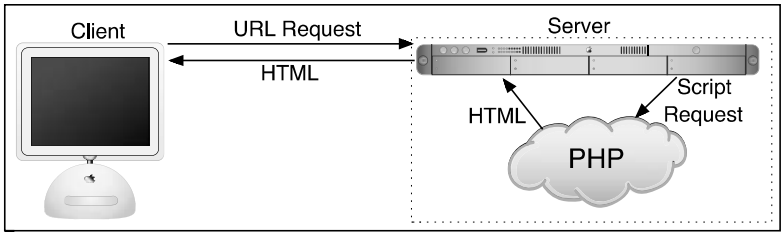
This differs from a static HTML site where, when a request is made, the server merely sends the HTML data to the Web browser and there is no server-side interpretation



B The Web Technology Surveys site provides this graphic regarding server-side technologies (`www.w3techs.com/technologies/overview/programming_language/all`).

Position Jun 2011	Position Jun 2010	Delta in Position	Programming Language	Ratings Jun 2011	Delta Jun 2010	Status
1	2	↑	Java	18.580%	+0.62%	A
2	1	↓	C	16.278%	-1.91%	A
3	3	=	C++	9.030%	-0.55%	A
4	6	↑↑	C#	6.844%	+2.06%	A
5	4	↓	PHP	6.602%	-2.47%	A
6	5	↓	(Visual) Basic	4.727%	-0.93%	A

C The Tiobe Index (`http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html`) uses a combination of factors to rank the popularity of programming languages.



D How PHP fits into the client/server model when a user requests a Web page.

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

occurring E. Because no server-side action is required, you can run HTML pages in your Web browser without using a server at all.

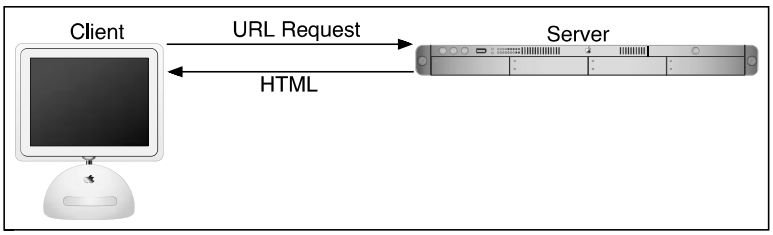
To the end user and the Web browser there is no perceptible difference between what `home.html` and `home.php` may look like, but how that page's content was created will be significantly different.

What is MySQL?

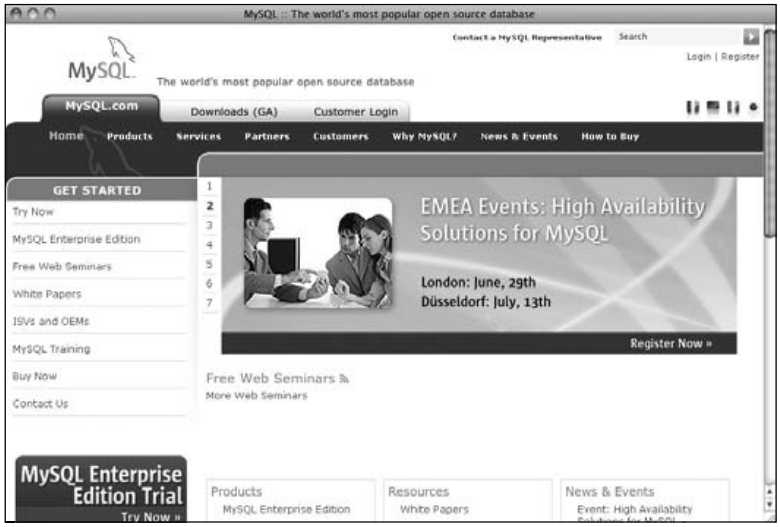
MySQL (`www.mysql.com`) F is the world's most popular open-source database. In fact, today MySQL is a viable competitor to the pricey goliaths such as Oracle and Microsoft's SQL Server (and, ironically, MySQL is owned by Oracle). Like PHP, MySQL offers excellent performance, portability, and reliability, with a moderate learning curve and little to no cost.

MySQL is a database management system (DBMS) for relational databases (therefore, MySQL is an RDBMS). A database, in the simplest terms, is a collection of data, be it text, numbers, or binary files, stored and kept organized by the DBMS.

There are many types of databases, from the simple flat-file to relational and object-oriented. A relational database uses multiple tables to store information in its most discernible parts. While relational databases may involve more thought in the design and programming stages, they offer improved reliability and data integrity that more than makes up for the extra effort required. Further, relational databases are more searchable and allow for concurrent users.



E The client/server process when a request for a static HTML page is made.



F The home page for the MySQL database application.

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

By incorporating a database into a Web application, some of the data generated by PHP can be retrieved from MySQL Ⓖ. This further moves the site's content from a static (hard-coded) basis to a flexible one, flexibility being the key to a dynamic Web site.

MySQL is an open-source application, like PHP, meaning that it is free to use or even modify (the source code itself is downloadable). There are occasions in which you should pay for a MySQL license, especially if you are making money from the sales or incorporation of the MySQL product. Check MySQL's licensing policy for more information on this.

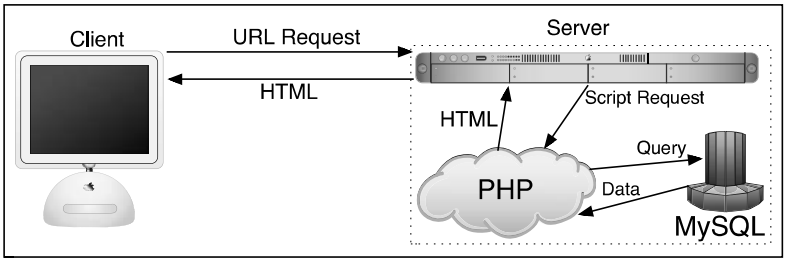
The MySQL software consists of several pieces, including the MySQL server (*mysqld*, which runs and manages the databases), the MySQL client (*mysql*, which gives you an interface to the server), and numerous utilities for maintenance and other purposes. PHP has always had good support for MySQL, and that is even more true in the most recent versions of the language.

MySQL has been known to handle databases as large as 60,000 tables with more than 5 billion rows. MySQL can work with tables as large as 8 million terabytes on some operating systems, generally a healthy 4 GB otherwise. MySQL is used by NASA and the United States Census Bureau, among many others.

At the time of this writing, MySQL is on version 5.5.13, with versions 5.6 and 6.0 in development. The version of MySQL you have affects what features you can use, so it's important that you know what you're working with. For this book, MySQL 5.1.44 and 5.5.8 were used, although you should be able to do everything in this book as long as you're using a version of MySQL greater than 5.0.

Pronunciation Guide

Trivial as it may be, I should clarify up front that MySQL is technically pronounced "My Ess Que Ell," just as SQL should be said "Ess Que Ell." This is a question many people have when first working with these technologies. While not a critical issue, it's always best to pronounce acronyms correctly.



Ⓖ How most of the dynamic Web applications in this book will work, using both PHP and MySQL.

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

What You'll Need

To follow the examples in this book, you'll need the following tools:

- A Web server application (for example, Apache, Abyss, or IIS)
- PHP
- MySQL
- A Web browser (Microsoft's Internet Explorer, Mozilla's Firefox, Apple's Safari, Google's Chrome, etc.)
- A text editor, PHP-capable WYSIWYG application (Adobe's Dreamweaver qualifies), or IDE (integrated development environment)
- An FTP application, if using a remote server

One of the great things about developing dynamic Web sites with PHP and MySQL is that all of the requirements can be met at no cost whatsoever, regardless of your operating system! Apache, PHP, and MySQL are each free; Web browsers can be had without cost; and many good text editors are available for nothing.

The appendix, which you can download from http://www.peachpit.com, discusses the installation process on the Windows and Mac OS X operating systems. If you have a computer, you are only a couple of downloads away from being able to create dynamic Web sites (in that case, your computer would represent both the client and the server in D and E). Conversely, you could purchase Web hosting for only dollars per month that will provide you with a PHP- and MySQL-enabled environment already online.

TIP To download this book's appendix from peachpit.com, create a free account at http://peachpit.com, and then register this book using ISBN number 0321784073. Once registered, you'll have access to the bonus content.

About This Book

This book teaches how to develop dynamic Web sites with PHP and MySQL, covering the knowledge that most developers might require. In keeping with the format of the Visual QuickPro series, the information is discussed using a step-by-step approach with corresponding images. The focus has been kept on real-world, practical examples, avoiding "here's something you could do but never would" scenarios. As a practicing Web developer myself, I wrote about the information that I use and avoided those topics immaterial to the task at hand. As a practicing writer, I made certain to include topics and techniques that I know readers are asking about.

The structure of the book is linear, and the intention is that you'll read it in order. It begins with three chapters covering the fundamentals of PHP (by the second chapter, you will have already developed your first dynamic Web page). After that, there are four chapters on SQL (Structured Query Language, which is used to interact with all databases) and MySQL. Those chapters teach the basics of SQL, database design, and the MySQL application in particular. Then there's one chapter on debugging and error management, information everyone needs. This is followed by a chapter introducing how to use PHP and MySQL together, a remarkably easy thing to do.

The following five chapters teach more application techniques to round out your knowledge. Security, in particular, is repeatedly addressed in those pages. Two new chapters, to be discussed momentarily, expand your newfound knowledge. Finally, I've included three example chapters, in which the heart of different Web applications are developed, with instructions.

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

Is this book for you?

This book was written for a wide range of people within the beginner-to-intermediate range. The book makes use of XHTML, so solid experience with XHTML or HTML is a must. Although this book covers many things, it does not formally teach HTML or Web-page design. Some CSS is sprinkled about these pages but also not taught.

Second, this book expects that you have one of the following:

- The drive and ability to learn without much hand holding, or...
- Familiarity with another programming language (even solid JavaScript skills would qualify), or...
- A cursory knowledge of PHP

Make no mistake: This book covers PHP and MySQL from A to Z, teaching everything you'll need to know to develop real-world Web sites, but particularly the early chapters cover PHP at a quick pace. For this reason I recommend either some programming experience or a curious and independent spirit when it comes to learning new things. If you find that the material goes too quickly, you should probably start off with the latest edition of my book *PHP for the World Wide Web: Visual QuickStart Guide*, which goes at a much more tempered pace.

No database experience is required, since SQL and MySQL are discussed starting at a more basic level.

What's new in this edition

The first three editions of this book have been very popular, and I've received a lot of positive feedback on them (thanks!). In writing this new edition, I wanted to do more than just update the material for the latest versions of PHP and MySQL, although that is an overriding consideration throughout the book. Other new features you'll find are:

- New examples demonstrating techniques frequently requested by readers
- Even more advanced MySQL and SQL instruction and examples
- A tutorial on using the jQuery JavaScript framework
- An introduction to the fundamentals and basic usage of Object-Oriented Programming
- Even more information and examples for improving the security of your scripts and sites
- Expanded and updated installation and configuration instructions
- Removal of outdated content (e.g., things used in older versions of PHP or no longer applicable)
- A "Review and Pursue" section at the end of each chapter, with review questions and prompts for ways in which you can further expand your knowledge based upon the information just covered

For those of you that also own a previous edition (thanks, thanks, thanks!), I believe that these new features will also make this edition a required fixture on your desk or bookshelf.

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

How this book compares to my other books

This is my fourth PHP and/or MySQL title, after (in order)

- *PHP for the World Wide Web: Visual QuickStart Guide*
- *PHP 5 Advanced for the World Wide Web: Visual QuickPro Guide*
- *MySQL: Visual QuickStart Guide*

I hope this résumé implies a certain level of qualification to write this book, but how do you, as a reader standing in a bookstore, decide which title is for you? Of course, you are more than welcome to splurge and buy the whole set, earning my eternal gratitude, but...

The *PHP for the World Wide Web: Visual QuickStart Guide* book is very much a beginner's guide to PHP. This title overlaps it some, mostly in the first three chapters, but uses new examples so as not to be redundant. For novices, this book acts as a follow-up to that one. The advanced book is really a sequel to this one, as it assumes a fair amount of knowledge and builds upon many things taught here. The MySQL book focuses almost exclusively on MySQL (there are but two chapters that use PHP).

With that in mind, read the section "Is this book for you?" and see if the requirements apply. If you have no programming experience at all and would prefer to be taught PHP more gingerly, my first book would be better. If you are already very comfortable with PHP and want to learn more of its advanced capabilities, pick up the second. If you are most interested in MySQL and are not concerned with learning much about PHP, check out the third.

That being said, if you want to learn everything you need to know to begin developing dynamic Web sites with PHP and MySQL today, then this is the book for you! It references the most current versions of both technologies, uses techniques not previously discussed in other books, and contains its own unique examples.

And whatever book you do choose, make sure you're getting the most recent edition or, barring that, the edition that best matches the versions of the technologies you'll be using.

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

Companion Web Site

I have developed a companion Web site specifically for this book, which you may reach at `www.LarryUllman.com`. There you will find every script from this book, a text file containing lengthy SQL commands, and a list of errata that occurred during publication. (If you have problems with a command or script, and you are following the book exactly, check the errata to ensure there is not a printing error before driving yourself absolutely mad.) At this Web site you will also find useful Web links, a popular forum where readers can ask and answer each other's questions (I answer many of them myself), and more!

Questions, comments, or suggestions?

If you have any questions on PHP or MySQL, you can turn to one of the many Web sites, mailing lists, newsgroups, and FAQ repositories already in existence. A quick search online will turn up virtually unlimited resources. For that matter, if you need an immediate answer, those sources or a quick Web search will most assuredly serve your needs (in all likelihood, someone else has already seen and solved your exact problem).

You can also direct your questions, comments, and suggestions to me. You'll get the fastest reply using the book's corresponding forum (I always answer those questions first). If you'd rather email me, my contact information is available on the Web site. I do try to answer every email I receive, although I cannot guarantee a quick reply.

Publisher's Tip: Check Out the Accompanying Video Training from Author Larry Ullman!

Visual QuickStart Guides are now even more visual: Building on the success of the top-selling Visual QuickStart Guide books, Peachpit now offers Video QuickStarts. As a companion to this book, Peachpit offers more than an hour of short, task-based videos that will help you master key features and techniques; instead of just reading about how to write PHP and MySQL scripts, you can watch it in action. It's a great way to learn all the basics and some of the newer or more complex features of the languages. Log on to the Peachpit site at www.peachpit.com/register to register your book, and you'll find a free streaming sample; purchasing the rest of the material is quick and easy.

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

1

Introduction to PHP

Although this book focuses on using MySQL and PHP in combination, you'll do a vast majority of your legwork using PHP alone. In this and the following chapter, you'll learn its basics, from syntax to variables, operators, and language constructs (conditionals, loops, and whatnot). At the same time you are picking up these fundamentals, you'll also begin developing usable code that you'll integrate into larger applications later in the book.

This introductory chapter will cruise through most of the basics of the PHP language. You'll learn the syntax for coding PHP, how to send data to the Web browser, and how to use two kinds of variables (strings and numbers) plus constants. Some of the examples may seem inconsequential, but they'll demonstrate ideas you'll have to master in order to write more advanced scripts further down the line. The chapter concludes with some quick debugging tips...you know...just in case!

In This Chapter

Basic Syntax	2
Sending Data to the Web Browser	6
Writing Comments	10
What Are Variables?	14
Introducing Strings	18
Concatenating Strings	21
Introducing Numbers	23
Introducing Constants	26
Single vs. Double Quotation Marks	29
Basic Debugging Steps	33
Review and Pursue	34

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

Basic Syntax

As stated in the book's introduction, PHP is an *HTML-embedded* scripting language, meaning that you can intermingle PHP and HTML code within the same file. So to begin programming with PHP, start with a simple Web page. **Script 1.1** is an example of a no-frills, no-content XHTML Transitional document, which will be used as the foundation for most Web pages in the book (this book does not formally discuss [X]HTML; see a resource dedicated to the topic for more information). Please also note that the template uses UTF-8 encoding, a topic discussed in the sidebar.

To add PHP code to a page, place it within PHP tags:

```
<?php
?>
```

Script 1.1 A basic XHTML 1.0 Transitional Web page.

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
    1.0 Transitional//EN" "http://www.w3.org/
    TR/xhtml1/DTD/xhtml1-transitional.dtd">
2   <html xmlns="http://www.w3.org/1999/
    xhtml" xml:lang="en" lang="en">
3   <head>
4      <meta http-equiv="Content-Type"
       content="text/html; charset=utf-8" />
5      <title>Page Title</title>
6   </head>
7   <body>
8      <!-- Script 1.1 - template.html -->
9   </body>
10  </html>
```

licensed by Dena Laney 2335846

Understanding Encoding

Encoding is a huge subject, but what you most need to understand is this: *the encoding you use in a file dictates what characters can be represented* (and therefore, what languages can be used). To select an encoding, you must first confirm that your text editor or Integrated Development Environment (IDE)—whatever application you're using to create the HTML and PHP scripts—can save documents using that encoding. Some applications let you set the encoding in the preferences or options area; others set the encoding when you save the file.

To indicate the encoding to the Web browser, there's the corresponding **meta** tag:

```
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
```

The *charset=utf-8* part says that UTF-8 encoding is being used, short for *8-bit Unicode Transformation Format*. Unicode is a way of reliably representing every symbol in every alphabet. Version 6 of Unicode—the current version at the time of this writing—supports over 99,000 characters!

If you want to create a multilingual Web page, UTF-8 is the way to go, and I'll be using it in this book's examples. You don't have to, of course. But whatever encoding you do use, make sure that the encoding indicated by the XHTML page matches the actual encoding set in your text editor or IDE. If you don't, you'll likely see odd characters when you view the page in a Web browser.

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

Script 1.2 This first PHP script doesn't do anything, but does demonstrate how a PHP script is written. It'll also be used as a test script, prior to getting into elaborate PHP code.

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
    1.0 Transitional//EN" "http://www.w3.org/
    TR/xhtml1/DTD/xhtml1-transitional.dtd">
2   <html xmlns="http://www.w3.org/1999/
    xhtml" xml:lang="en" lang="en">
3   <head>
4      <meta http-equiv="Content-Type"
       content="text/html; charset=utf-8" />
5      <title>Basic PHP Page</title>
6   </head>
7   <body>
8      <!-- Script 1.2 - first.php -->
9      <p>This is standard HTML.</p>
10  <?php
11  ?>
12  </body>
13  </html>
```

HTML5

At the time of this writing, the next major release of HTML—HTML5—is being actively developed and discussed, but is not production ready, which is why I chose not to use it in the book. In fact, I wouldn't be surprised if HTML5 is still not released by the time I start the fifth edition of this book, and it will take even longer for broad browser adoption of the language. Still, as HTML5 is an exciting future development, this book will occasionally mention features you can expect to see introduced and supported over time.

Anything written within these tags will be treated by the Web server as PHP, meaning the PHP interpreter will process the code. Any text outside of the PHP tags is immediately sent to the Web browser as regular HTML. (Because PHP is most often used to create content displayed in the Web browser, the PHP tags are normally put somewhere within the page's body.)

Along with placing PHP code within PHP tags, your PHP files must have a proper *extension*. The extension tells the server to treat the script in a special way, namely, as a PHP page. Most Web servers use **`.html`** for standard HTML pages and **`.php`** for PHP files.

Before getting into the steps, understand that *you must already have a working PHP installation*! This could be on a hosted site or your own computer, after following the instructions in Appendix A, "Installation," which is a free download from peachpit.com.

To make a basic PHP script:

1. Create a new document in your text editor or IDE, to be named **`first.php`** (**Script 1.2**).

   It generally does not matter what application you use, be it Adobe Dreamweaver (a fancy IDE), TextMate (a great and popular Macintosh plain-text editor), or vi (a plain-text Unix editor, lacking a graphical interface). Still, some text editors and IDEs make typing and debugging HTML and PHP easier (conversely, Notepad on Windows does some things that makes coding harder: *don't use Notepad*!). If you don't already have an application you're attached to, search the Web or use the book's corresponding forum (**`www.LarryUllman.com/forums/`**) to find one.

continues on next page

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

2. Create a basic HTML document:

```
<!DOCTYPE html PUBLIC "-//W3C//
→DTD XHTML 1.0 Transitional//EN"
→"http://www.w3.org/TR/xhtml1/DTD/
→xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/
→1999/xhtml" xml:lang="en"
→lang="en">
<head>
  <meta http-equiv="Content-Type"
  →content="text/html;
  →charset=utf-8" />
  <title>Basic PHP Page</title>
</head>
<body>
  <!-- Script 1.2 - first.php -->
<p>This is standard HTML.</p>
</body>
</html>
```

Although this is the syntax being used throughout the book, you can change the HTML to match whichever standard you intend to use (e.g., HTML 4.0 Strict). Again, see a dedicated (X)HTML resource if you're unfamiliar with any of this HTML code.

3. Before the closing **body** tag, insert the PHP tags:

```
<?php
?>
```

These are the *formal* PHP tags, also known as *XML-style* tags. Although PHP supports other tag types, I recommend that you use the formal type, and I will do so throughout this book.

4. Save the file as **first.php.**

Remember that if you don't save the file using an appropriate PHP extension, the script will not execute properly. (Just one of the reasons not to use Notepad is that it will secretly add the **.txt** extension to PHP files, thereby causing many headaches.)

5. Place the file in the proper directory of your Web server.

If you are running PHP on your own computer (presumably after following the installation directions in Appendix A), you just need to move, copy, or save the file to a specific folder on your computer. Check Appendix A or the documentation for your particular Web server to identify the correct directory, if you don't already know what it is.

If you are running PHP on a hosted server (i.e., on a remote computer), you'll need to use a File Transfer Protocol (FTP) application to upload the file to the proper directory. Your hosting company will provide you with access and the other necessary information.

6. Run **first.php** in your Web browser Ⓐ.

Because PHP scripts need to be parsed by the server, you *absolutely must* access them via a URL (i.e., the address in the browser must begin with **http://**). You cannot simply open them in your Web browser as you would a file in other applications (in which case the address would start with **file://** or **C:** or the like).

Ⓐ While it seems like any other (simple) HTML page, this is in fact a PHP script and the basis for the rest of the examples in the book.

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

If you are running PHP on your own computer, you'll need to use a URL like **`http://localhost/first.php`**, **`http://127.0.0.1/first.php`**, or **`http://localhost/~<user>/first.php`** (on Mac OS X, using your actual username for **`<user>`**). If you are using a Web host, you'll need to use **`http://`*`your-domain-name`*`/first.php`** (e. g., **`http://www.example.com/first.php`**).

7. If you don't see results like those in Ⓐ, start debugging!

   Part of learning any programming language is mastering debugging. It's a sometimes-painful but absolutely necessary process. With this first example, if you don't see a simple, but perfectly valid, Web page, follow these steps:

1. Confirm that you have a working PHP installation (see Appendix A for testing instructions).
2. Make sure that you are running the script through a URL. The address in the Web browser must begin with **`http://`**. If it starts with **`file://`**, that's a problem Ⓑ.
3. If you get a file not found (or similar) error, you've likely put the file in the wrong directory or mistyped the file's name (either when saving it or in your Web browser).

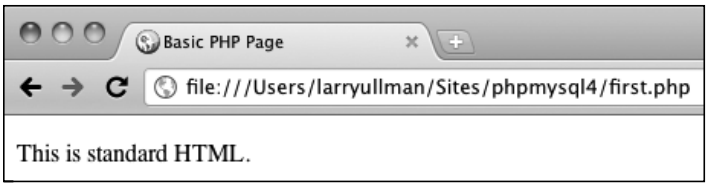
   If you've gone through all this and are still having problems, turn to the book's corresponding forum (**`www.LarryUllman.com/forums/`**).

TIP To find more information about HTML and XHTML, check out Elizabeth Castro's excellent book *HTML, XHTML, and CSS, Sixth Edition: Visual QuickStart Guide,* (Peachpit Press, 2006) or search the Web.

TIP You can embed multiple sections of PHP code within a single HTML document (i.e., you can go in and out of the two languages). You'll see examples of this throughout the book.

TIP Prior to UTF-8, ISO-8859-1 was one of the more commonly used encodings. It represents most Western European languages. It's still the default encoding for many Web browsers and other applications.

TIP You can declare the encoding of an external CSS file by adding `@charset "utf-8";` as the first line in the file. If you're not using UTF-8, change the line accordingly.



Ⓑ PHP code will only be executed when run through *http://* (not that this particular script is affected either way).

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

Sending Data to the Web Browser

To create dynamic Web sites with PHP, you must know how to send data to the Web browser. PHP has a number of built-in functions for this purpose, the most common being **echo** and **print.** I personally tend to favor **echo**:

```
echo 'Hello, world!';
echo "What's new?";
```

You could use **print** instead, if you prefer (the name more obviously indicates what it does):

```
print 'Hello, world!';
print "What's new?";
```

As you can see from these examples, you can use either single or double quotation marks (but there is a distinction between the two types of quotation marks, which will be made clear by the chapter's end). The first quotation mark after the function name indicates the start of the message to be printed. The next matching quotation mark (i.e., the next quotation mark of the same kind as the opening mark) indicates the end of the message to be printed.

Along with learning how to send data to the Web browser, you should also notice that in PHP all statements—a line of executed code, in layman's terms—must end with a semicolon. Also, PHP is *case-insensitive* when it comes to function names, so **ECHO**, **echo**, **eCHo**, and so forth will all work. The all-lowercase version is easiest to type, of course.

Needing an Escape

As you might discover, one of the complications with sending data to the Web involves printing single and double quotation marks. Either of the following will cause errors:

```
echo "She said, "How are you?"";
echo 'I'm just ducky.';
```

There are two solutions to this problem. First, use single quotation marks when printing a double quotation mark and vice versa:

```
echo 'She said, "How are you?"';
echo "I'm just ducky.";
```

Or, you can *escape* the problematic character by preceding it with a backslash:

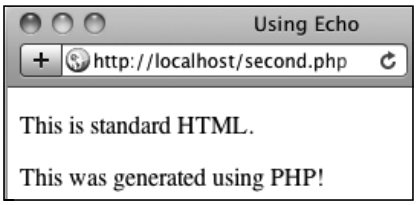
```
echo "She said, \"How are you?\"";
echo 'I\'m just ducky.';
```

An escaped quotation mark will merely be printed like any other character. Understanding how to use the backslash to escape a character is an important concept, and one that will be covered in more depth at the end of the chapter.

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

Script 1.3 Using **print** or **echo**, PHP can send data to the Web browser.

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
    1.0 Transitional//EN" "http://www.w3.org/
    TR/xhtml1/DTD/xhtml1-transitional.dtd">
2   <html xmlns="http://www.w3.org/1999/
    xhtml" xml:lang="en" lang="en">
3   <head>
4      <meta http-equiv="Content-Type"
       content="text/html; charset=utf-8" />
5      <title>Using Echo</title>
6   </head>
7   <body>
8      <!-- Script 1.3 - second.php -->
9      <p>This is standard HTML.</p>
10  <?php
11  echo 'This was generated using PHP!';
12  ?>
13  </body>
14  </html>
```



A The results still aren't glamorous, but this page was in part dynamically generated by PHP.

To send data to the Web browser:

1. Open **first.php** (refer to Script 1.2) in your text editor or IDE.
2. Between the PHP tags (lines 10 and 11), add a simple message (**Script 1.3**):

   ```
   echo 'This was generated using
   → PHP!';
   ```

   It truly doesn't matter what message you type here, which function you use (**echo** or **print**), or which quotation marks, for that matter—just be careful if you are printing a single or double quotation mark as part of your message (see the sidebar "Needing an Escape").
3. If you want, change the page title to better describe this script (line 5):

   ```
   <title>Using Echo</title>
   ```

   This change only affects the browser window's title bar.
4. Save the file as **second.php**, place it in your Web directory, and test it in your Web browser **A**.

   Remember that all PHP scripts must be run through a URL (**http://*something***)!

continues on next page

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

5. If necessary, debug the script.

   If you see a parse error instead of your message B, check that you have both opened and closed your quotation marks and escaped any problematic characters (see the sidebar). Also be certain to conclude each statement with a semicolon.

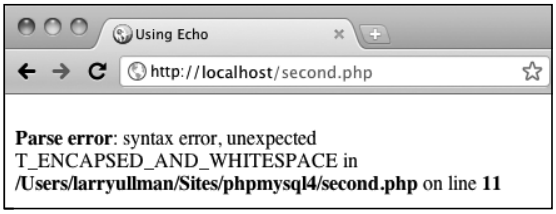
   If you see an entirely blank page, this is probably for one of two reasons:

   - There is a problem with your HTML. Test this by viewing the source of your page and looking for HTML problems there C.
   - An error occurred, but *display_errors* is turned off in your PHP configuration, so nothing is shown. In this case, see the section in Appendix A on how to configure PHP so that you can turn *display_errors* back on.

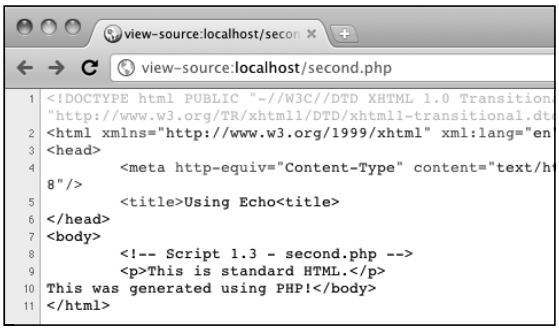
TIP Technically, `echo` and `print` are *language constructs*, not functions. That being said, don't be flummoxed as I continue to call them "functions" for convenience. Also, as you'll see later in the book, I include the parentheses when referring to functions—say `number_format( )`, not just `number_format`—to help distinguish them from variables and other parts of PHP. This is just my own little convention.

TIP You can, and often will, use `echo` and `print` to send HTML code to the Web browser, like so D:

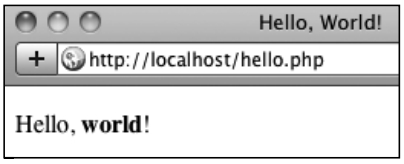
```
echo '<p>Hello, <b>world</b>!</p>';
```



B This may be the first of many parse errors you see as a PHP programmer (this one is caused by the omission of the terminating quotation mark).

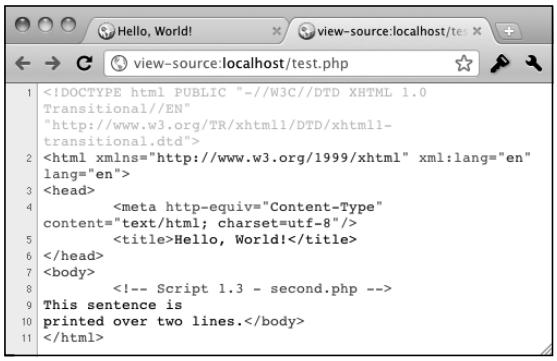


C One possible cause of a blank PHP page is a simple HTML error, like the closing `title` tag here (it's missing the slash).



D PHP can send HTML code (like the formatting here) as well as simple text A to the Web browser.

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.



E Printing text and HTML over multiple PHP lines will generate HTML source code that also extends over multiple lines. Note that extraneous white spacing in the HTML source will not affect the look of a page **F** but can make the source easier to review.

F The return in the HTML source **E** has no effect on the rendered result. The only way to alter the spacing of a displayed Web page is to use HTML tags (like `<br />` and `<p></p>`).

TIP **Echo and `print` can both be used over multiple lines:**

```
echo 'This sentence is
printed over two lines.';
```

TIP **What happens in this case is that the return (created by pressing Enter or Return) becomes part of the printed message, which isn't terminated until the closing quotation mark. The net result will be the "printing" of the return in the HTML source code E. This will not have an effect on the generated page F. For more on this, see the sidebar "Understanding White Space."**

Understanding White Space

With PHP you send data (like HTML tags and text) to the Web browser, which will, in turn, render that data as the Web page the end user sees. Thus, what you are often doing with PHP is creating the *HTML source* of a Web page. With this in mind, there are three areas of notable *white space* (extra spaces, tabs, and blank lines): in your PHP scripts, in your HTML source, and in the rendered Web page.

PHP is generally white space insensitive, meaning that you can space out your code however you want to make your scripts more legible. HTML is also generally white space insensitive. Specifically, the only white space in HTML that affects the rendered page is a single space (multiple spaces still get rendered as one). If your HTML source has text on multiple lines, that doesn't mean it'll appear on multiple lines in the rendered page (**E** and **F**).

To alter the spacing in a rendered Web page, use the HTML tags `<br />` (line break, `<br>` in older HTML standards) and `<p></p>` (paragraph). To alter the spacing of the HTML *source* created with PHP, you can

- Use `echo` or `print` over the course of several lines.

  or

- Print the newline character (`\n`) within double quotation marks, which is equivalent to Enter or Return.

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

Writing Comments

Creating executable PHP code is only a part of the programming process (admittedly, it's the most important part). A secondary but still crucial aspect to any programming endeavor involves documenting your code. In fact, when I'm asked what qualities distinguish the beginning programmer from the more experienced one, a good and thorough use of comments is my unwavering response.

In HTML you can add comments using special tags:

```
<!-- Comment goes here. -->
```

HTML comments are viewable in the source but do not appear in the rendered page (see E and F in the previous section).

PHP comments are different in that they aren't sent to the Web browser at all, meaning they won't be viewable to the end user, even when looking at the HTML source.

PHP supports three comment syntaxes. The first uses the pound or number symbol (**#**):

```
# This is a comment.
```

The second uses two slashes:

```
// This is also a comment.
```

Both of these cause PHP to ignore everything that follows until the end of the line (when you press Return or Enter). Thus, these two comments are for single lines only. They are also often used to place a comment on the same line as some PHP code:

```
print 'Hello!'; // Say hello.
```

A third style allows comments to run over multiple lines:

```
/* This is a longer comment
that spans two lines. */
```


PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

Script 1.4 These basic comments demonstrate the three comment syntaxes you can use in PHP.

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
    1.0 Transitional//EN" "http://www.w3.org/
    TR/xhtml1/DTD/xhtml1-transitional.dtd">
2   <html xmlns="http://www.w3.org/1999/
    xhtml" xml:lang="en" lang="en">
3   <head>
4      <meta http-equiv="Content-Type"
       content="text/html; charset=utf-8" />
5      <title>Comments</title>
6   </head>
7   <body>
8   <?php
9
10  # Script 1.4 - comments.php
11  # Created March 16, 2011
12  # Created by Larry E. Ullman
13  # This script does nothing much.
14
15  echo '<p>This is a line of text.<br />
    This is another line of text.</p>';
16
17  /*
18  echo 'This line will not be
    executed.';
19  */
20
21  echo "<p>Now I'm done.</p>";
    // End of PHP code.
22
23  ?>
24  </body>
25  </html>
```

To comment your scripts:

1. Begin a new PHP document in your text editor or IDE, to be named **comments.php**, starting with the initial HTML (**Script 1.4**):

```
<!DOCTYPE html PUBLIC "-//W3C//
→DTD XHTML 1.0 Transitional//EN"
→"http://www.w3.org/TR/xhtml1/DTD/
→xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/
→1999/xhtml" xml:lang="en"
→lang="en">
<head>
  <meta http-equiv="Content-Type"
  →content="text/html;
  →charset=utf-8" />
  <title>Comments</title>
</head>
<body>
```

2. Add the initial PHP tag and write your first comments:

```
<?php
# Script 1.4 - comments.php
# Created March 16, 2011
# Created by Larry E. Ullman
# This script does nothing much.
```

One of the first comments each script should contain is an introductory block that lists creation date, modification date, creator, creator's contact information, purpose of the script, and so on. Some people suggest that the shell-style comments (**#**) stand out more in a script and are therefore best for this kind of notation.

continues on next page

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

3. Send some HTML to the Web browser:

```
echo '<p>This is a line of
→text.<br />This is another line
→of text.</p>';
```

It doesn't matter what you do here, just make something for the Web browser to display. For the sake of variety, the **echo** statement will print some HTML tags, including a line break (**
**) to add some spacing to the generated HTML page.

4. Use the multiline comments to comment out a second **echo** statement:

```
/*
echo 'This line will not be
→executed.';
*/
```

By surrounding any block of PHP code with **/*** and ***/**, you can render that code inert without having to delete it from your script. By later removing the comment tags, you can reactivate that section of PHP code.

5. Add a final comment after a third **echo** statement:

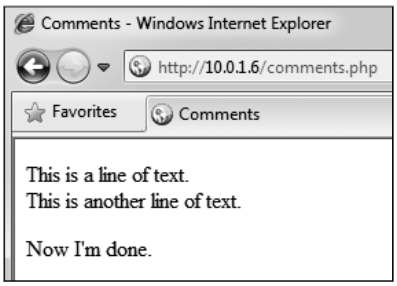
```
echo "<p>Now I'm done.</p>";
→// End of PHP code.
```

This last (superfluous) comment shows how to place a comment at the end of a line, a common practice. Note that double quotation marks surround this message, as single quotation marks would conflict with the apostrophe (see the "Needing an Escape" sidebar, earlier in the chapter).

6. Close the PHP section and complete the HTML page:

```
?>
</body>
</html>
```

7. Save the file as **comments.php**, place it in your Web directory, and test it in your Web browser (A).



(A) The PHP comments in Script 1.4 don't appear in the Web page or the HTML source (B).

Licensed by
Dena Laney
2335846

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

8. If you're the curious type, check the source code in your Web browser to confirm that the PHP comments do not appear there ❶.

TIP You shouldn't nest (place one inside another) multiline comments (`/* */`). Doing so will cause problems.

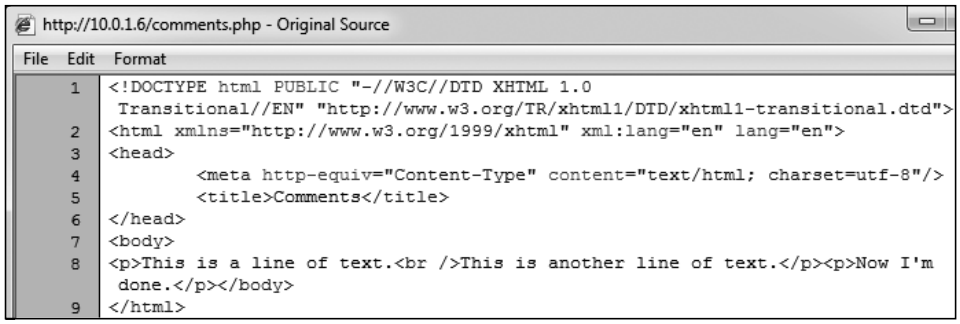
TIP Any of the PHP comments can be used at the end of a line (say, after a function call):

```
echo 'Howdy'; /* Say 'Howdy' */
```

Although this is allowed, it's far less common.

TIP It's nearly impossible to over-comment your scripts. Always err on the side of writing too many comments as you code. That being said, in the interest of saving space, the scripts in this book will not be as well documented as I would suggest they should be.

TIP It's also important that as you change a script you keep the comments up-to-date and accurate. There's nothing more confusing than a comment that says one thing when the code really does something else.



❶ The PHP comments from Script 1.4 are nowhere to be seen in the client's browser.

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

What Are Variables?

Variables are containers used to temporarily store values. These values can be numbers, text, or much more complex data. PHP supports eight types of variables. These include four scalar (single-valued) types—*Boolean* (**TRUE** or **FALSE**), *integer*, *floating point* (decimals), and *strings* (characters); two nonscalar (multivalued)—*arrays* and *objects*; plus *resources* (which you'll see when interacting with databases) and *NULL* (which is a special type that has no value).

Regardless of what type you are creating, all variable names in PHP follow certain syntactical rules:

- A variable's name must start with a dollar sign (**$**), for example, **$name**.
- The variable's name can contain a combination of letters, numbers, and the underscore, for example, **$my_report1**.
- The first character after the dollar sign must be either a letter or an underscore (it cannot be a number).
- *Variable names in PHP are case-sensitive!* This is a very important rule. It means that $**name** and **$Name** are entirely different variables.

To begin working with variables, this next script will print out the value of three *predefined variables*. Whereas a standard variable is assigned a value during the execution of a script, a predefined variable will already have a value when the script begins its execution. Most of these predefined variables reflect properties of the server as a whole, such as the operating system in use.

Before getting into this script, there are two more things you should know. First, variables can be assigned values using the equals sign (**=**), also called the *assignment operator*. Second, to display the value of a variable, you can print the variable without quotation marks:

```
print $some_var;
```

Or variables can be printed within *double* quotation marks:

```
print "Hello, $name";
```

You cannot print variables within single quotation marks:

```
print 'Hello, $name'; // Won't work!
```


PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

Script 1.5 This script prints three of PHP's many predefined variables.

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
    1.0 Transitional//EN" "http://www.w3.org/
    TR/xhtml1/DTD/xhtml1-transitional.dtd">
2   <html xmlns="http://www.w3.org/1999/
    xhtml" xml:lang="en" lang="en">
3   <head>
4      <meta http-equiv="Content-Type"
       content="text/html; charset=utf-8" />
5      <title>Predefined Variables</title>
6   </head>
7   <body>
8   <?php # Script 1.5 - predefined.php
9
10  // Create a shorthand version of the
    variable names:
11  $file = $_SERVER['SCRIPT_FILENAME'];
12  $user = $_SERVER['HTTP_USER_AGENT'];
13  $server = $_SERVER['SERVER_SOFTWARE'];
14
15  // Print the name of this script:
16  echo "<p>You are running the
    file:<br /><b>$file</b>.</p>\n";
17
18  // Print the user's information:
19  echo "<p>You are viewing this page
    using:<br /><b>$user</b></p>\n";
20
21  // Print the server's information:
22  echo "<p>This server is running:
    <br /><b>$server</b>.</p>\n";
23
24  ?>
25  </body>
26  </html>
```

To use variables:

1. Begin a new PHP document in your text editor or IDE, to be named **predefined.php**, starting with the initial HTML (**Script 1.5**):

```
<!DOCTYPE html PUBLIC "-//W3C//
→DTD XHTML 1.0 Transitional//EN"
→"http://www.w3.org/TR/xhtml1/DTD/
→xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/
→1999/xhtml" xml:lang="en"
→lang="en">
<head>
  <meta http-equiv="Content-Type"
  →content="text/html;
  →charset=utf-8" />
  <title>Predefined Variables
  →</title>
</head>
<body>
```

2. Add the opening PHP tag and the first comment:

```
<?php # Script 1.5 - predefined.php
```

From here on out, scripts will no longer comment on the creator, creation date, and so forth, although you should continue to document your scripts thoroughly. Scripts will, however, make a comment indicating the script's number and filename for ease of cross-referencing (both in the book and when you download them from the book's supporting Web site, **www.LarryUllman.com**).

continues on next page

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

3. Create a shorthand version of the first variable to be used in this script:

```
$file = $_SERVER['SCRIPT_FILENAME'];
```

This script will use three variables, each of which comes from the larger predefined **$_SERVER** variable. **$_SERVER** refers to a mass of server-related information. The first variable the script uses is **$_SERVER['SCRIPT_FILENAME']**. This variable stores the full path and name of the script being run (for example, **C:\Program Files\Apache\htdocs\predefined.php**).

The value stored in **$_SERVER['SCRIPT_FILENAME']** will be assigned to the new variable **$file**. Creating new variables with shorter names and then assigning them values from **$_SERVER** will make it easier to refer to the variables when printing them. (It also gets around another issue you'll learn about in due time.)

4. Create a shorthand version of two more variables:

```
$user = $_SERVER['HTTP_USER_AGENT'];
$server = $_SERVER
→ ['SERVER_SOFTWARE'];
```

$_SERVER['HTTP_USER_AGENT'] represents the Web browser and operating system of the user accessing the script. This value is assigned to **$user**.

$_SERVER['SERVER_SOFTWARE'] represents the Web application on the server that's running PHP (e.g., Apache, Abyss, Xitami, IIS). This is the program that must be installed (see Appendix A) in order to run PHP scripts on that computer.

5. Print out the name of the script being run:

```
echo "<p>You are running the
→ file:<br /><b>$file</b>.</p>\n";
```

The first variable to be printed is **$file**. Notice that this variable must be used within double quotation marks and that the statement also makes use of the PHP newline character (**\n**), which will add a line break in the generated HTML source. Some basic HTML tags—paragraph and bold—are added to give the generated page a bit of flair.

6. Print out the information of the user accessing the script:

```
echo "<p>You are viewing this page
→ using:<br /><b>$user</b></p>\n";
```

This line prints the second variable, **$user**. To repeat what's said in the fourth step, **$user** correlates to **$_SERVER['HTTP_USER_AGENT']** and refers to the operating system, browser type, and browser version being used to access the Web page.

7. Print out the server information:

```
echo "<p>This server is running:
→ <br /><b>$server</b>.</p>\n";
```

8. Complete the PHP block and the HTML page:

```
?>
</body>
</html>
```


PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

9. Save the file as `predefined.php`, place it in your Web directory, and test it in your Web browser **A**.

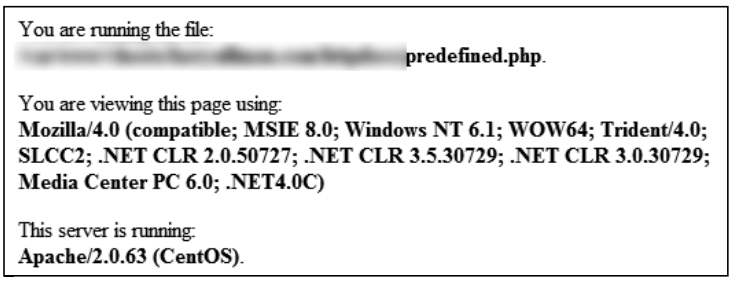
TIP If you have problems with this, or any other script, turn to the book's corresponding Web forum (`www.LarryUllman.com/forums/`) for assistance.

TIP If possible, run this script using a different Web browser and/or on another server B.

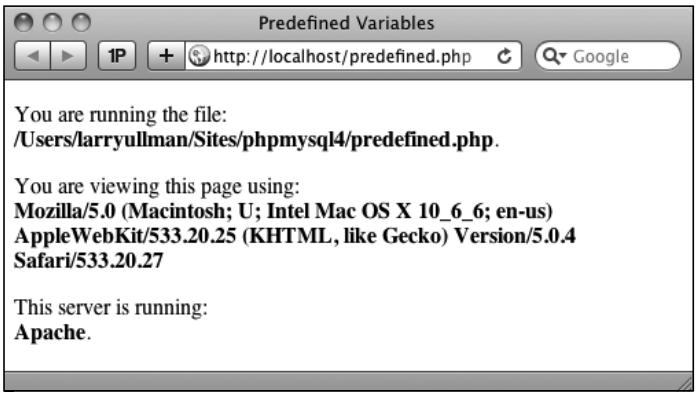
TIP Variable names cannot contain spaces. The underscore is commonly used in lieu of a space.

TIP The most important consideration when creating variables is to use a consistent naming scheme. In this book you'll see that I use all-lowercase letters for my variable names, with underscores separating words (`$first_name`). Some programmers prefer to use capitalization instead: `$FirstName` (known as "camel-case" style).

TIP PHP is very casual in how it treats variables, meaning that you don't need to initialize them (set an immediate value) or declare them (set a specific type), and you can convert a variable among the many types without problem.



A The `predefined.php` script reports back to the viewer information about the script, the Web browser being used to view it, and the server itself.



B This is the book's first truly dynamic script, in that the Web page changes depending upon the server running it and the Web browser viewing it (compare with **A**).

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

Introducing Strings

Now that you've been introduced to the general concept of variables, let's look at variables in detail. The first variable type to delve into is the *string*. A string is merely a quoted chunk of characters: letters, numbers, spaces, punctuation, and so forth. These are all strings:

- 'Tobias'
- "In watermelon sugar"
- '100'
- 'August 2, 2011'

To make a string variable, assign a string value to a valid variable name:

```
$first_name = 'Tobias';
$today = 'August 2, 2011';
```

When creating strings, you can use either single or double quotation marks to encapsulate the characters, just as you would when printing text. Likewise, you must use the same type of quotation mark for the beginning and the end of the string. If that same mark appears within the string, it must be escaped:

```
$var = "Define \"platitude\", please.";
```

Or you can also use the other quotation mark type:

```
$var = 'Define "platitude", please.';
```

To print out the value of a string, use either **echo** or **print**:

```
echo $first_name;
```

To print the value of string within a context, you must use double quotation marks:

```
echo "Hello, $first_name";
```

You've already worked with strings once—when using the predefined variables in the preceding section (the values of those variables happened to be strings). In this next example, you'll create and use your own strings.

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

Script 1.6 String variables are created and their values are sent to the Web browser in this script.

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
    1.0 Transitional//EN" "http://www.w3.org/
    TR/xhtml1/DTD/xhtml1-transitional.dtd">
2   <html xmlns="http://www.w3.org/1999/
    xhtml" xml:lang="en" lang="en">
3   <head>
4      <meta http-equiv="Content-Type"
       content="text/html; charset=utf-8" />
5      <title>Strings</title>
6   </head>
7   <body>
8   <?php # Script 1.6 - strings.php
9
10  // Create the variables:
11  $first_name = 'Haruki';
12  $last_name = 'Murakami';
13  $book = 'Kafka on the Shore';
14
15  // Print the values:
16  echo "<p>The book <em>$book</em>
    was written by $first_name
    $last_name.</p>";
17
18  ?>
19  </body>
20  </html>
```

To use strings:

1. Begin a new PHP document in your text editor or IDE, to be named **strings.php**, starting with the initial HTML and including the opening PHP tag (**Script 1.6**):

```
<!DOCTYPE html PUBLIC "-//W3C//
→DTD XHTML 1.0 Transitional//EN"
→"http://www.w3.org/TR/xhtml1/DTD/
→xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/
→1999/xhtml" xml:lang="en"
→lang="en">
<head>
  <meta http-equiv="Content-Type"
  →content="text/html;
  →charset=utf-8" />
  <title>Strings</title>
</head>
<body>
<?php # Script 1.6 - strings.php
```

2. Within the PHP tags, create three variables:

```
$first_name = 'Haruki';
$last_name = 'Murakami';
$book = 'Kafka on the Shore';
```

This rudimentary example creates **$first_name**, **$last_name**, and **$book** variables that will then be printed out in a message.

continues on next page

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

3. Add an **echo** statement:

```
echo "<p>The book <em>$book</em>
→ was written by $first_name
→ $last_name.</p>";
```

All this script does is print a statement of authorship based upon three established variables. A little HTML formatting (the emphasis on the book's title) is thrown in to make it more attractive. Remember to use double quotation marks here for the variable values to be printed out appropriately (more on the importance of double quotation marks at the chapter's end).

4. Complete the PHP block and the HTML page:

```
?>
</body>
</html>
```

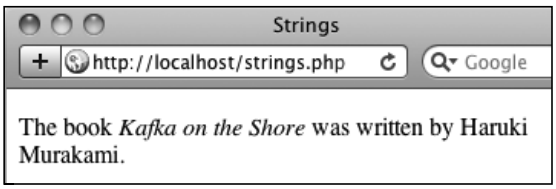
5. Save the file as **strings.php**, place it in your Web directory, and test it in your Web browser ❶.

6. If desired, change the values of the three variables, save the file, and run the script again ❷.

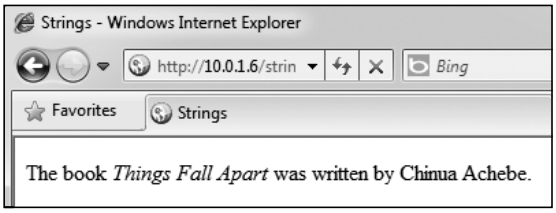
TIP **If you assign another value to an existing variable (say $book), the new value will overwrite the old one. For example:**

```
$book = 'High Fidelity';
$book = 'The Corrections';
/* $book now has a value of
'The Corrections'. */
```

TIP **PHP has no set limits on how big a string can be. It's theoretically possible that you'll be limited by the resources of the server, but it's doubtful that you'll ever encounter such a problem.**



❶ The resulting Web page is based upon printing out the values of three variables.



❷ The output of the script is changed by altering the variables in it.

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

Script 1.7 Concatenation gives you the ability to append more characters onto a string.

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
    1.0 Transitional//EN" "http://www.w3.org/
    TR/xhtml1/DTD/xhtml1-transitional.dtd">
2   <html xmlns="http://www.w3.org/1999/
    xhtml" xml:lang="en" lang="en">
3   <head>
4      <meta http-equiv="Content-Type"
       content="text/html; charset=utf-8" />
5      <title>Concatenation</title>
6   </head>
7   <body>
8   <?php # Script 1.7 - concat.php
9
10  // Create the variables:
11  $first_name = 'Melissa';
12  $last_name = 'Bank';
13  $author = $first_name . ' ' .
    $last_name;
14
15  $book = 'The Girls\' Guide to Hunting
    and Fishing';
16
17  //Print the values:
18  echo "<p>The book <em>$book</em>
    was written by $author.</p>";
19
20  ?>
21  </body>
22  </html>
```

Concatenating Strings

Concatenation is like addition for strings, whereby characters are added to the end of the string. It is performed using the *concatenation operator*, which is the period (**.**):

```
$city= 'Seattle';
$state = 'Washington';
$address = $city . $state;
```

The **$address** variable now has the value *SeattleWashington*, which almost achieves the desired result (*Seattle, Washington*). To improve upon this, you could write

```
$address = $city . ', ' . $state;
```

so that a comma and a space are concatenated to the variables as well.

Because of how liberally PHP treats variables, concatenation is possible with strings and numbers. Either of these statements will produce the same result (*Seattle, Washington 98101*):

```
$address = $city . ', ' . $state .
 ' 98101';
$address = $city . ', ' . $state .
 ' ' . 98101;
```

Let's modify **strings.php** to use this new operator.

To use concatenation:

1. Open **strings.php** (refer to Script 1.6) in your text editor or IDE.
2. After you've established the **$first_name** and **$last_name** variables (lines 11 and 12), add this line (**Script 1.7**):

   ```
   $author = $first_name . ' ' .
   → $last_name;
   ```

   As a demonstration of concatenation, a new variable—**$author**—will be created as the concatenation of two existing strings and a space in between.

continues on next page

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

3. Change the **echo** statement to use this new variable:

```
echo "<p>The book <em>$book</em>
→ was written by $author.</p>";
```

Since the two variables have been turned into one, the **echo** statement should be altered accordingly.

4. If desired, change the HTML page title and the values of the first name, last name, and book variables.

5. Save the file as **concat.php**, place it in your Web directory, and test it in your Web browser **A**.

A In this revised script, the end result of concatenation is not apparent to the user.

TIP **PHP has a slew of useful string-specific functions, which you'll see over the course of this book. For example, to calculate how long a string is (how many characters it contains), use strlen( ):**

```
$num = strlen('some string'); // 11
```

TIP **You can have PHP convert the case of strings with: strtolower( ), which makes it entirely lowercase; strtoupper( ), which makes it entirely uppercase; ucfirst( ), which capitalizes the first character; and ucwords( ), which capitalizes the first character of every word.**

TIP **If you are merely concatenating one value to another, you can use the concatenation assignment operator (.=). The following are equivalent:**

```
$title = $title . $subtitle;
$title .= $subtitle;
```

TIP **The initial example in this section could be rewritten using either**

```
$address = "$city, $state";
```

or

```
$address = $city;
$address .= ', ';
$address .= $state;
```

Using the PHP Manual

The PHP manual—accessible online at **www.php.net/manual**—lists every function and feature of the language. The manual is organized with general concepts (installation, syntax, variables) discussed first and ends with the functions by topic (MySQL, string functions, and so on).

To quickly look up any function in the PHP manual, go to **www.php.net/*functionname*** in your Web browser (for example, **www.php.net/print**). For each function, the manual indicates:

- The versions of PHP the function is available in.
- How many and what types of arguments the function takes (optional arguments are wrapped in square brackets).
- What type of value the function returns.

The manual also contains a description of the function.

You should be in the habit of checking out the PHP manual whenever you're confused by a function, how it's properly used, or need to learn more about any feature of the language. It's also critically important that you know what version of PHP you're running, as functions and other particulars of PHP do change over time.

Licensed by Dena Laney 23358846
PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

Introducing Numbers

In introducing variables, I stated that PHP has both integer and floating-point (decimal) number types. In my experience, though, these two types can be classified under the generic title *numbers* without losing any valuable distinction (for the most part). Valid number-type variables in PHP can be anything like

- 8
- 3.14
- 10980843985
- -4.2398508
- 4.4e2

Notice that these values are never quoted—quoted numbers are strings with numeric values—nor do they include commas to indicate thousands. Also, a number is assumed to be positive unless it is preceded by the minus sign (-).

Along with the standard arithmetic operators you can use on numbers (**Table 1.1**), there are dozens of functions built into PHP. Two common ones are **round( )** and **number_format( )**. The former rounds a decimal to the nearest integer:

```
$n = 3.14;
$n = round ($n); // 3
```

It can also round to a specified number of decimal places:

```
$n = 3.142857;
$n = round ($n, 3); // 3.143
```

The **number_format( )** function turns a number into the more commonly written version, grouped into thousands using commas:

```
$n = 20943;
$n = number_format ($n); // 20,943
```

This function can also set a specified number of decimal points:

```
$n = 20943;
$n = number_format ($n, 2); //
20,943.00
```

To practice with numbers, let's write a mock-up script that performs the calculations one might use in an e-commerce shopping cart.

TABLE 1.1 Arithmetic Operators

Operator	Meaning
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus
++	Increment
--	Decrement

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

To use numbers:

1. Begin a new PHP document in your text editor or IDE, to be named **numbers.php** (Script 1.8):

```
<!DOCTYPE html PUBLIC "-//W3C//
→DTD XHTML 1.0 Transitional//EN"
→"http://www.w3.org/TR/xhtml1/DTD/
→xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/
→1999/xhtml" xml:lang="en"
→lang="en">
<head>
  <meta http-equiv="Content-Type"
  →content="text/html;
  →charset=utf-8" />
  <title>Numbers</title>
</head>
<body>
<?php # Script 1.8 - numbers.php
```

2. Establish the requisite variables:

```
$quantity = 30;
$price = 119.95;
$taxrate = .05;
```

This script will use three hard-coded variables upon which calculations will be made. Later in the book, you'll see how these values can be dynamically determined (i.e., by user interaction with an HTML form).

3. Perform the calculations:

```
$total = $quantity * $price;
$total = $total + ($total *
→$taxrate);
```

The first line establishes the order total as the number of widgets purchased multiplied by the price of each widget. The second line then adds the amount of tax to the total (calculated by multiplying the tax rate by the total).

Script 1.8 The **numbers.php** script performs basic mathematical calculations, like those used in an e-commerce application.

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
    1.0 Transitional//EN" "http://www.w3.org/
    TR/xhtml1/DTD/xhtml1-transitional.dtd">
2   <html xmlns="http://www.w3.org/1999/
    xhtml" xml:lang="en" lang="en">
3   <head>
4      <meta http-equiv="Content-Type"
       content="text/html; charset=utf-8" />
5      <title>Numbers</title>
6   </head>
7   <body>
8   <?php # Script 1.8 - numbers.php
9
10  // Set the variables:
11  $quantity = 30; // Buying 30 widgets.
12  $price = 119.95;
13  $taxrate = .05; // 5% sales tax.
14
15  // Calculate the total:
16  $total = $quantity * $price;
17  $total = $total + ($total * $taxrate);
    // Calculate and add the tax.
18
19  // Format the total:
20  $total = number_format ($total, 2);
21
22  // Print the results:
23  echo '<p>You are purchasing <b>' .
    $quantity . '</b> widget(s) at a cost
    of <b>$' . $price . '</b> each. With
    tax, the total comes to <b>$' .
    $total . '</b>.</p>';
24
25  ?>
26  </body>
27  </html>
```


PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

4. Format the total:

```
$total = number_format ($total, 2);
```

The `number_format( )` function will group the total into thousands and round it to two decimal places. Applying this function will properly format the calculated value.

5. Print the results:

```
echo '<p>You are purchasing <b>' .
→$quantity . '</b> widget(s) at
→a cost of <b>$' . $price . '</b>
→each. With tax, the total comes
→to <b>$' . $total . '</b>.</p>';
```

The last step in the script is to print out the results. The `echo` statement uses both single-quoted text and concatenated variables in order to print out the full combination of HTML, dollar signs, and variable values. You'll see an alternative approach in the last example of this chapter.

6. Complete the PHP code and the HTML page:

```
?>
</body>
</html>
```

7. Save the file as `numbers.php`, place it in your Web directory, and test it in your Web browser **A**.

8. If desired, change the initial three variables and rerun the script **B**.

TIP **PHP supports a maximum integer of around two billion on most platforms. With numbers larger than that, PHP will automatically use a floating-point type.**

TIP **When dealing with arithmetic, the issue of *precedence* arises (the order in which complex calculations are made). While the PHP manual and other sources tend to list out the hierarchy of precedence, I find programming to be safer and more legible when I group clauses in parentheses to force the execution order (see line 17 of Script 1.8).**

TIP **Computers are notoriously poor at dealing with decimals. For example, the number 2.0 may actually be stored as 1.99999. Most of the time this won't be a problem, but in cases where mathematical precision is paramount, rely on integers, not decimals. The PHP manual has information on this subject, as well as alternative functions for improving computational accuracy.**

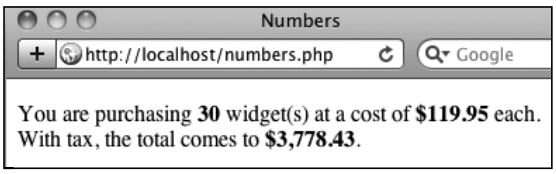
TIP **Many of the mathematical operators also have a corresponding assignment operator, letting you create a shorthand for assigning values. This line,**

```
$total = $total + ($total * $taxrate);
```

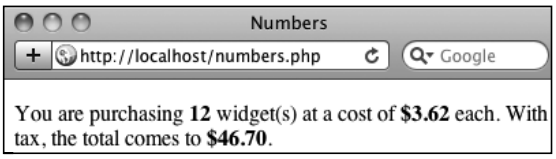
could be rewritten as

```
$total += ($total * $taxrate);
```

TIP **If you set a `$price` value without using two decimals (e.g., 119.9 or 34), you would want to apply `number_format( )` to `$price` before printing it.**



A The numbers PHP page (Script 1.8) performs calculations based upon set values.



B To change the generated Web page, alter any or all of the three variables (compare with **A**).

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

Introducing Constants

Constants, like variables, are used to temporarily store a value, but otherwise, constants and variables differ in many ways. For starters, to create a constant, you use the **define( )** function instead of the assignment operator (**=**):

```
define ('NAME', value);
```

Notice that, as a rule of thumb, constants are named using all capitals, although this is not required. Most importantly, constants do not use the initial dollar sign as variables do (because constants are not variables).

A constant can only be assigned a *scalar* value, like a string or a number:

```
define ('USERNAME', 'troutocity');
define ('PI', 3.14);
```

And unlike variables, a constant's value cannot be changed.

To access a constant's value, like when you want to print it, you cannot put the constant within quotation marks:

```
echo "Hello, USERNAME"; // Won't work!
```

With that code, PHP literally prints *Hello, USERNAME* Ⓐ and not the value of the **USERNAME** constant (because there's no indication that **USERNAME** is anything other than literal text). Instead, either print the constant by itself:

```
echo 'Hello, ';
echo USERNAME;
```

or use the concatenation operator:

```
echo 'Hello, ' . USERNAME;
```

PHP runs with several predefined constants, much like the predefined variables used earlier in the chapter. These include **PHP_VERSION** (the version of PHP running) and **PHP_OS** (the operating system of the server). This next script will print those two values, along with the value of a user-defined constant.

Ⓐ Constants cannot be placed within quoted strings.

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

Script 1.9 Constants are another temporary storage tool you can use in PHP, distinct from variables.

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
    1.0 Transitional//EN" "http://www.w3.org/
    TR/xhtml1/DTD/xhtml1-transitional.dtd">
2   <html xmlns="http://www.w3.org/1999/
    xhtml" xml:lang="en" lang="en">
3   <head>
4      <meta http-equiv="Content-Type"
       content="text/html; charset=utf-8" />
5      <title>Constants</title>
6   </head>
7   <body>
8   <?php # Script 1.9 - constants.php
9
10  // Set today's date as a constant:
11  define ('TODAY', 'March 16, 2011');
12
13  // Print a message, using predefined
    constants and the TODAY constant:
14  echo '<p>Today is ' . TODAY . '.<br />
    This server is running version <b>' .
    PHP_VERSION . '</b> of PHP on the <b>' .
    PHP_OS . '</b> operating system.</p>';
15
16  ?>
17  </body>
18  </html>
```

To use constants:

1. Begin a new PHP document in your text editor or IDE, to be named **constants.php** (**Script 1.9**).

```
<!DOCTYPE html PUBLIC "-//W3C//
→DTD XHTML 1.0 Transitional//EN"
→"http://www.w3.org/TR/xhtml1/DTD/
→xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/
→1999/xhtml" xml:lang="en"
→lang="en">
<head>
  <meta http-equiv="Content-Type"
  →content="text/html;
  →charset=utf-8" />
  <title>Constants</title>
</head>
<body>
<?php # Script 1.9 - constants.php
```

2. Create a new date constant:

```
define ('TODAY', 'March 16, 2011');
```

An admittedly trivial use of constants, but this example will illustrate the point. In Chapter 9, "Using PHP with MySQL," you'll see how to use constants to store your database access information.

3. Print out the date, the PHP version, and operating system information:

```
echo '<p>Today is ' . TODAY .
→'.<br />This server is running
→version <b>' . PHP_VERSION .
→'</b> of PHP on the <b>' . PHP_OS .
→'</b> operating system.</p>';
```

Since constants cannot be printed within quotation marks, use the concatenation operator in the **echo** statement.

continues on next page

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

4. Complete the PHP code and the HTML page:

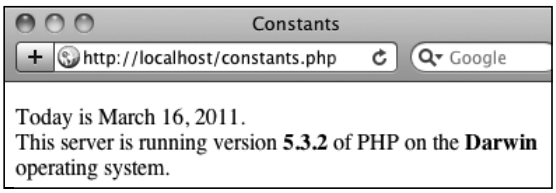
```
?>
</body>
</html>
```

5. Save the file as **`constants.php`**, place it in your Web directory, and test it in your Web browser **B**.

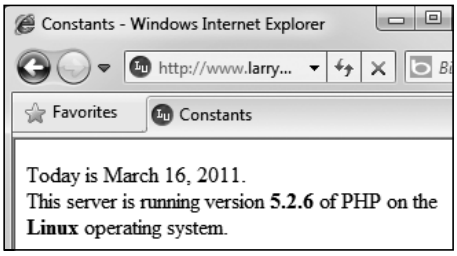
TIP **If possible, run this script on another PHP-enabled server C.**

TIP **The operating system called *Darwin* B is the technical term for Mac OS X.**

TIP **In Chapter 12, "Cookies and Sessions," you'll learn about another constant, `SID` (which stands for *session ID*).**



B By making use of PHP's constants, you can learn more about your PHP setup.



C Running the same script (refer to Script 1.9) on different servers garners different results.

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

Single vs. Double Quotation Marks

In PHP it's important to understand how single quotation marks differ from double quotation marks. With **echo** and **print**, or when assigning values to strings, you can use either, as in the examples used so far. But there is a key difference between the two types of quotation marks and when you should use which. You've seen this difference already, but it's an important enough concept to merit more discussion.

In PHP, *values enclosed within single quotation marks will be treated literally*, whereas *those within double quotation marks will be interpreted*. In other words, placing variables and special characters (**Table 1.2**) within double quotes will result in their represented values printed, not their literal values. For example, assume that you have

```
$var = 'test';
```

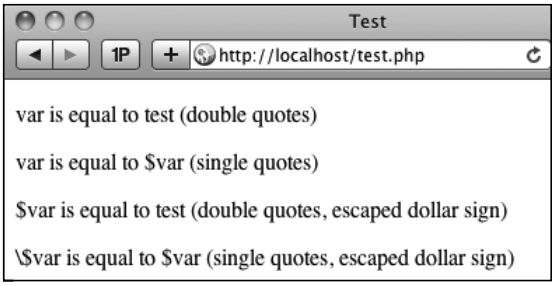
The code **echo "var is equal to $var";** will print out *var is equal to test*, but the code **echo 'var is equal to $var';** will print out *var is equal to $var*. Using an escaped dollar sign, the code **echo "\$var is equal to $var";** will print out *$var is equal to test*, whereas the code **echo '\$var is equal to $var';** will print out *\$var is equal to $var* Ⓐ.

As these examples should illustrate, double quotation marks will replace a variable's name (**$var**) with its value (*test*) and a special character's code (**\$**) with its represented value (*$*). Single quotes will always display exactly what you type, except for the escaped single quote (**\'**) and the escaped backslash (****), which are printed as a single quotation mark and a single backslash, respectively.

As another example of how the two quotation marks differ, let's modify the **numbers.php** script as an experiment.

TABLE 1.2 Escape Sequences

Code	Meaning
\"	Double quotation mark
\'	Single quotation mark
\\	Backslash
\n	Newline
\r	Carriage return
\t	Tab
\$	Dollar sign



Ⓐ How single and double quotation marks affect what gets printed by PHP.

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

To use single and double quotation marks:

1. Open **numbers.php** (refer to Script 1.8) in your text editor or IDE.
2. Delete the existing **echo** statement (**Script 1.10**).
3. Print a caption and then rewrite the original **echo** statement using double quotation marks:

```
echo "<h3>Using double quotation
→ marks:</h3>";
echo "<p>You are purchasing
→ <b>$quantity</b> widget(s) at
→ a cost of <b>\$$price</b> each.
→ With tax, the total comes to
→ <b>\$$total</b>.</p>\n";
```

In the original script, the results were printed using single quotation marks and concatenation. The same result can be achieved using double quotation marks. When using double quotation marks, the variables can be placed within the string.

There is one catch, though: trying to print a dollar amount as $12.34 (where 12.34 comes from a variable) would suggest that you would code **$$var**. That will not work (for complicated reasons). Instead, escape the initial dollar sign, resulting in **\$$var**, as you see twice in this code. The first dollar sign will be printed, and the second becomes the start of the variable name.

Script 1.10 This, the final script in the chapter, demonstrates the differences between using single and double quotation marks.

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
    1.0 Transitional//EN" "http://www.w3.org/
    TR/xhtml1/DTD/xhtml1-transitional.dtd">
2   <html xmlns="http://www.w3.org/1999/
    xhtml" xml:lang="en" lang="en">
3   <head>
4      <meta http-equiv="Content-Type"
       content="text/html; charset=utf-8" />
5      <title>Quotation Marks</title>
6   </head>
7   <body>
8   <?php # Script 1.10 - quotes.php
9
10  // Set the variables:
11  $quantity = 30; // Buying 30 widgets.
12  $price = 119.95;
13  $taxrate = .05; // 5% sales tax.
14
15  // Calculate the total.
16  $total = $quantity * $price;
17  $total = $total + ($total * $taxrate);
    // Calculate and add the tax.
18
19  // Format the total:
20  $total = number_format ($total, 2);
21
22  // Print the results using double
    quotation marks:
23  echo "<h3>Using double quotation
    marks:</h3>";
24  echo "<p>You are purchasing
    <b>$quantity</b> widget(s) at a cost
    of <b>\$$price</b> each. With tax,
    the total comes to <b>\$$total</b>.
    </p>\n";
25
26  // Print the results using single
    quotation marks:
27  echo '<h3>Using single quotation
    marks:</h3>';
28  echo '<p>You are purchasing
    <b>$quantity</b> widget(s) at a cost
    of <b>\$$price</b> each. With tax,
    the total comes to <b>\$$total</b>.
    </p>\n';
29
30  ?>
31  </body>
32  </html>
```


PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

4. Repeat the **echo** statements, this time using single quotation marks:

```
echo '<h3>Using single quotation
→ marks:</h3>';
echo '<p>You are purchasing
→ <b>$quantity</b> widget(s) at
→ a cost of <b>\$$price</b> each.
→ With tax, the total comes to
→ <b>\$$total</b>.</p>\n';
```

   This **echo** statement is used to highlight the difference between using single or double quotation marks. It will not work as desired, and the resulting page will show you exactly what does happen instead.

5. If you want, change the page's title.
6. Save the file as **quotes.php**, place it in your Web directory, and test it in your Web browser **B**.
7. View the source of the Web page to see how using the newline character (**\n**) within each quotation mark type also differs.

   You should see that when you place the newline character within double quotation marks it creates a newline in the HTML source. When placed within single quotation marks, the literal characters **** and **n** are printed instead.

TIP Because PHP will attempt to find variables within double quotation marks, using single quotation marks is theoretically faster. If you need to print the value of a variable, though, you must use double quotation marks.

TIP As valid HTML often includes a lot of double-quoted attributes, it's often easiest to use single quotation marks when printing HTML with PHP:

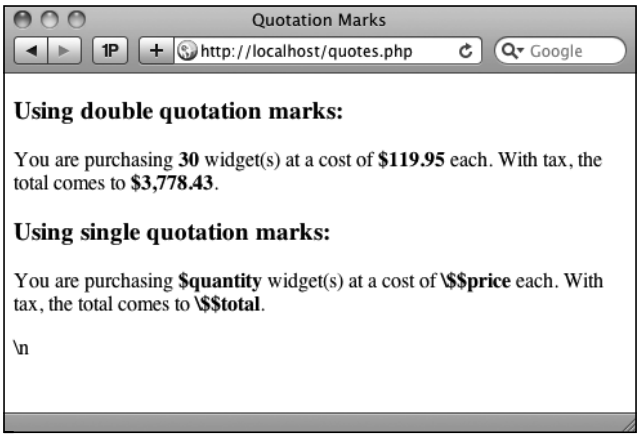
```
echo '<table width="80%" border="0"
→ cellspacing="2" cellpadding="3"
→ align="center">';
```

If you were to print out this HTML using double quotation marks, you would have to escape all of the double quotation marks in the string:

```
echo "<table width=\"80%\" border=
→ \"0\" cellspacing=\"2\" cellpadding=
→ \"3\" align=\"center\">";
```

TIP In newer versions of PHP, you can actually use `$$price` and `$$total` without preceding them with a backslash (thanks to some internal magic). In older versions of PHP, you cannot. To guarantee reliable results, regardless of PHP version, I recommend using the `\$$var` syntax when you need to print a dollar sign immediately followed by the value of a variable.

TIP If you're still unclear as to the difference between the types, use double quotation marks and you're less likely to have problems.



B These results demonstrate when and how you'd use one type of quotation mark as opposed to the other.

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

Basic Debugging Steps

Debugging is by no means a simple concept to grasp, and unfortunately, it's one that is only truly mastered by doing. The next 50 pages could be dedicated to the subject and you'd still only be able to pick up a fraction of the debugging skills that you'll eventually acquire and need.

The reason I introduce debugging in this somewhat harrowing way is that it's important not to enter into programming with delusions. Sometimes code won't work as expected, you'll inevitably create careless errors, and some days you'll want to pull your hair out, even when using a comparatively user-friendly language such as PHP. In short, prepare to be perplexed and frustrated at times. I've been coding in PHP since 1999, and occasionally I still get stuck in the programming muck. But debugging is a very important skill to have, and one that you will eventually pick up out of necessity and experience. As you begin your PHP programming adventure, I can offer the following basic but concrete debugging tips.

Note that these are just some general debugging techniques, specifically tailored to the beginning PHP programmer. Chapter 8, "Error Handling and Debugging," goes into other techniques in more detail.

licensed by
Dena Laney
2335846

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

To debug a PHP script:

- **Make sure you're always running PHP scripts through a URL!**

  This is perhaps the most common beginner's mistake. PHP code must be run through the Web server application, which means it must be requested via `http://something`. When you see actual PHP code instead of the result of that code's execution, most likely you're not running the PHP script through a URL.

- **Know what version of PHP you're running.**

  Some problems will arise from the version of PHP in use. Before you ever use any PHP-enabled server, run a `phpinfo.php` script (see Appendix A) or reference the `PHP_VERSION` constant to confirm the version of PHP in use.

- **Make sure `display_errors` is on.**

  This is a basic PHP configuration setting (also discussed in Appendix A). You can confirm this setting by executing the `phpinfo()` function (just use your browser to search for *display_errors* in the resulting page). For security reasons, PHP may not be set to display the errors that occur. If that's the case, you'll end up seeing blank pages when problems occur. To debug most problems, you'll need to see the errors, so turn this setting on while you're learning. You'll find instructions for doing so in Appendix A.

- **Check the HTML source code.**

  Sometimes the problem is hidden in the HTML source of the page. In fact, sometimes the PHP error message can be hidden there!

- **Trust the error message.**

  Another very common beginner's mistake is to not fully read or trust the error that PHP reports. Although an error message can often be cryptic and may seem meaningless, it can't be ignored. At the very least, PHP is normally correct as to the line on which the problem can be found. And if you need to relay that error message to someone else (like when you're asking me for help), do include the entire error message!

- **Take a break!**

  So many of the programming problems I've encountered over the years, and the vast majority of the toughest ones, have been solved by stepping away from the computer for a while. It's easy to get frustrated and confused, and in such situations, any further steps you take are likely to only make matters worse.

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

Review and Pursue

New in this edition of the book, each chapter ends with a "Review and Pursue" section. In these sections you'll find questions regarding the material just covered and prompts for ways to expand your knowledge and experience on your own. If you have any problems with these sections, either in answering the questions or pursuing your own endeavors, turn to the book's supporting forum (`www.LarryUllman.com/forums/`).

Review

- What tags are used to surround PHP code?
- What extension should a PHP file have?
- What does a page's *encoding* refer to? What impact does the encoding have on the page?
- What PHP functions, or language constructs, can you use to send data to the Web browser?
- How does using single versus double quotation marks differ in creating or printing strings?
- What does it mean to *escape* a character in a string?
- What are the three comment syntaxes in PHP? Which one can be used over multiple lines?
- What character do all variable names begin with? What characters can come next? What other characters can be used in a variable's name?
- Are variable names case-sensitive or case-insensitive?
- What is the assignment operator?
- How do you create a string variable?
- What is the concatenation operator? What is the concatenation assignment operator?
- How are constants defined and used?

Pursue

- If you don't already know—*for certain*—what version of PHP you're running, check now.
- Look up one of the mentioned string functions in the PHP manual. Then check out some of the other available string functions listed therein.
- Look up one of the mentioned number functions in the PHP manual. Then check out some of the other available number functions listed therein.
- Search the PHP manual for the `$_SERVER` variable to see what other information it contains.
- Create a new script, from scratch, that defines and displays the values of some string variables. Use double quotation marks in the `echo` or `print` statement that outputs the values. For added complexity include some HTML in the output. Then rewrite the script so that it uses single quotation marks and concatenation instead of double quotation marks.
- Create a new script, from scratch, that defines, manipulates, and displays the values of some numeric variables.

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

2

Programming with PHP

Now that you have the fundamentals of the PHP scripting language down, it's time to build on those basics and start truly programming. In this chapter you'll begin creating more elaborate scripts while still learning some of the standard constructs, functions, and syntax of the language.

You'll start by creating an HTML form, and then learn how you can use PHP to handle the submitted values. From there, the chapter covers conditionals and the remaining operators (Chapter 1, "Introduction to PHP," presented the assignment, concatenation, and mathematical operators), arrays (another variable type), and one last language construct, loops.

In This Chapter

Creating an HTML Form	36
Handling an HTML Form	41
Conditionals and Operators	45
Validating Form Data	49
Introducing Arrays	54
For and While Loops	69
Review and Pursue	72

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

Creating an HTML Form

Handling an HTML form with PHP is perhaps the most important process in any dynamic Web site. Two steps are involved: first you create the HTML form itself, and then you create the corresponding PHP script that will receive and process the form data.

It is outside the realm of this book to go into HTML forms in any detail, but I will lead you through one quick example so that it may be used throughout the chapter. If you're unfamiliar with the basics of an HTML form, including the various types of elements, see an HTML resource for more information.

An HTML form is created using the **form** tags and various elements for taking input. The **form** tags look like

```
<form action="script.php"
 method="post">
</form>
```

In terms of PHP, the most important attribute of your **form** tag is **action**, which dictates to which page the form data will be sent. The second attribute—**method**—has its own issues (see the "Choosing a Method" sidebar), but *post* is the value you'll use most frequently.

The different inputs—be they text boxes, radio buttons, select menus, check boxes, etc.—are placed within the opening and closing **form** tags. As you'll see in the next section, what kinds of inputs your form has makes little difference to the PHP script handling it. You should, however, pay attention to the names you give your form inputs, as they'll be of critical importance when it comes to your PHP code.

Choosing a Method

The **method** attribute of a form dictates how the data is sent to the handling page. The two options—***get*** and ***post***—refer to the HTTP (HyperText Transfer Protocol) method to be used. The **GET** method sends the submitted data to the receiving page as a series of name-value pairs appended to the URL. For example,

```
http://www.example.com/script.php?
→ name=Homer&gender=M&age=35
```

The benefit of using the **GET** method is that the resulting page can be bookmarked in the user's Web browser (since it's a complete URL). For that matter, you can also click Back in your Web browser to return to a **GET** page, or reload it without problems (none of which is true for **POST**). But there is a limit in how much data can be transmitted via **GET**, and this method is less secure (since the data is visible).

Generally speaking, ***GET*** *is used for requesting information*, like a particular record from a database or the results of a search (searches almost always use **GET**). *The* ***POST*** *method is used when an action is expected*: the updating of a database record or the sending of an email. For these reasons I will primarily use **POST** throughout this book, with noted exceptions.

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

Script 2.1 This simple HTML form will be used for several of the examples in this chapter.

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD
    XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/
    xhtml1-transitional.dtd">
2   <html xmlns="http://www.w3.org/1999/
    xhtml" xml:lang="en" lang="en">
3   <head>
4      <meta http-equiv="Content-Type"
       content="text/html; charset=utf-8" />
5      <title>Simple HTML Form</title>
6      <style type="text/css" title="text/
       css" media="all">
7      label {
8         font-weight: bold;
9         color: #300ACC;
10     }
11     </style>
12  </head>
13  <body>
14  <!-- Script 2.1 - form.html -->
15
16  <form action="handle_form.php"
    method="post">
17
18     <fieldset><legend>Enter your
       information in the form below:
       </legend>
19
20     <p><label>Name: <input type="text"
       name="name" size="20" maxlength=
       "40" /></label></p>
21
22     <p><label>Email Address: <input
       type="text" name="email" size="40"
       maxlength="60" /></label></p>
23
24     <p><label for="gender">Gender:
       </label><input type="radio"
       name="gender" value="M" /> Male
       <input type="radio" name="gender"
       value="F" /> Female</p>
25
26     <p><label>Age:
27     <select name="age">
28        <option value="0-29">Under 30
          </option>
29        <option value="30-60">Between 30
          and 60</option>
```

code continues on next page

To create an HTML form:

1. Begin a new HTML document in your text editor or IDE, to be named **form.html** (**Script 2.1**):

```
<!DOCTYPE html PUBLIC "-//W3C//
→DTD XHTML 1.0 Transitional//EN"
→"http://www.w3.org/TR/xhtml1/DTD/
→xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/
→1999/xhtml" xml:lang="en"
→lang="en">
<head>
  <meta http-equiv="Content-Type"
  →content="text/html;
  →charset=utf-8" />
  <title>Simple HTML Form</title>
  <style type="text/css"
  →title="text/css" media="all">
  label {
    font-weight: bold;
    color: #300ACC;
  }
  </style>
</head>
<body>
<!-- Script 2.1 - form.html -->
```

The document uses the same basic syntax for an HTML page as in the previous chapter. I have added some inline CSS (Cascading Style Sheets) in order to style the form slightly (specifically, making **label** elements bold and blue).

CSS is the preferred way to handle many formatting and layout issues in an HTML page. You'll see a little bit of CSS here and there in this book; if you're not familiar with the subject, check out a dedicated CSS reference.

Finally, an HTML comment indicates the file's name and number.

continues on next page

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

2. Add the initial **form** tag:

```
<form action="handle_form.php"
→ method="post">
```

Since the **action** attribute dictates to which script the form data will go, you should give it an appropriate name (*handle_form* to correspond with this page: **form.html**) and the **.php** extension (since a PHP script will handle this form's data).

3. Begin the HTML form:

```
<fieldset><legend>Enter your
→ information in the form below:
→ </legend>
```

I'm using the **fieldset** and **legend** HTML tags because I like the way they make the HTML form look (they add a box around the form with a title at the top). This isn't pertinent to the form itself, though.

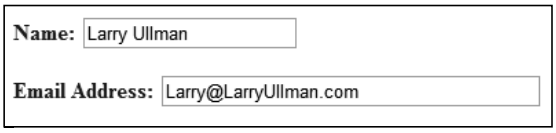
4. Add two text inputs:

```
<p><label>Name: <input type=
→ "text" name="name" size="20"
→ maxlength="40" /></label></p>
<p><label>Email Address: <input
→ type="text" name="email"
→ size="40" maxlength="60" />
→ </label></p>
```

These are just simple text inputs, allowing the user to enter their name and email address Ⓐ. In case you are wondering, the extra space and slash at the end of each input's tag are required for valid XHTML. With standard HTML, these tags would conclude with **maxlength="40">** instead. The **label** tags just tie each textual label to the associated element.

Script 2.1 *continued*

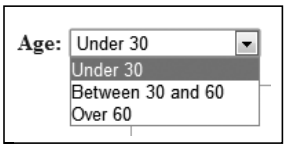
```
30          <option value="60+">Over 60
            </option>
31      </select></label></p>
32
33      <p><label>Comments: <textarea
        name="comments" rows="3" cols="40"></
        textarea></label></p>
34
35      </fieldset>
36
37      <p align="center"><input type=
        "submit" name="submit" value=
        "Submit My Information" /></p>
38
39  </form>
40
41  </body>
42  </html>
```



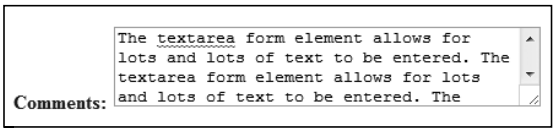
Ⓐ Two text inputs.

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

B If multiple radio buttons have the same `name` value, only one can be selected by the user.



C The pull-down menu offers three options, of which only one can be selected (in this example).



D The textarea form element type allows for lots and lots of text.

5. Add a pair of radio buttons:

```
<p><label for="gender">Gender:
→ </label><input type="radio"
→ name="gender" value="M" /> Male
→ <input type="radio" name=
→ "gender" value="F" /> Female</p>
```

The radio buttons **B** both have the same name, meaning that only one of the two can be selected. They have different values, though.

6. Add a pull-down menu:

```
<p><label>Age:
<select name="age">
  <option value="0-29">Under 30
  → </option>
  <option value="30-60">Between
  → 30 and 60</option>
  <option value="60+">Over 60
  → </option>
</select></label></p>
```

The `select` tag starts the pull-down menu, and then each `option` tag will create another line in the list of choices **C**.

7. Add a text box for comments:

```
<p><label>Comments: <textarea
→ name="comments" rows="3"
→ cols="40"></textarea></label></p>
```

Textareas are different from text inputs; they are presented as a box **D**, not as a single line. They allow for much more information to be typed and are useful for taking user comments.

continues on next page

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

8. Complete the form:

```
    </fieldset>
    <p align="center"><input type=
    → "submit" name="submit" value=
    → "Submit My Information" /></p>
</form>
```

The first tag closes the **fieldset** that was opened in Step 3. Then a submit button is created and centered using a **p** tag. Finally, the form is closed.

9. Complete the HTML page:

```
</body>
</html>
```

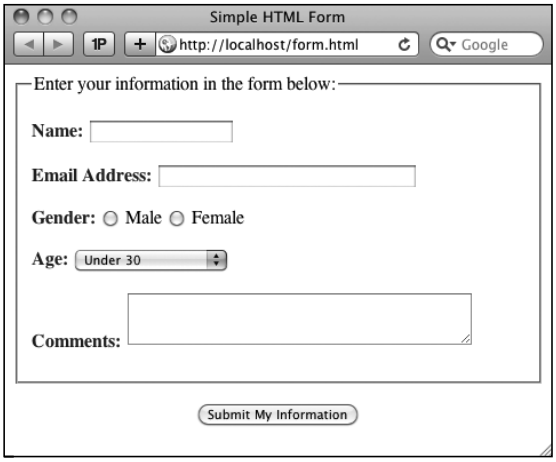
10. Save the file as **form.html**, place it in your Web directory, and view it in your Web browser **E**.

TIP Since this page contains just HTML, it uses an .html extension. It could instead use a .php extension without harm (since code outside of the PHP tags is treated as HTML).

TIP You can specify the encoding to accept in an HTML form tag, too:

```
<form accept-charset="utf-8">
```

By default, a Web page will use the same encoding as the page itself for any submitted data.



E The complete form, which requests some basic information from the user.

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

Handling an HTML Form

Now that the HTML form has been created, it's time to write a bare-bones PHP script to handle it. To say that this script will be *handling* the form means that the PHP page will do something with the data it receives (which is the data the user entered into the form). In this chapter, the scripts will simply print the data back to the Web browser. In later examples, form data will be stored in a MySQL database, compared against previously stored values, sent in emails, and more.

The beauty of PHP—and what makes it so easy to learn and use—is how well it interacts with HTML forms. PHP scripts store the received information in special variables. For example, say you have a form with an input defined like so:

```
<input type="text" name="city" />
```

Whatever the user types into that input will be accessible via a PHP variable named **`$_REQUEST['city']`**. It is very important that the spelling and capitalization match *exactly*! PHP is case-sensitive when it comes to variable names, so **`$_REQUEST['city']`** will work, but **`$_Request['city']`** and **`$_REQUEST['City']`** will have no value.

This next example will be a PHP script that handles the already-created HTML form (Script 2.1). This script will assign the form data to new variables (to be used as shorthand, just like in Script 1.5, **`predefined.php`**). The script will then print the received values.

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

To handle an HTML form:

1. Begin a new PHP document in your text editor or IDE, to be named **handle_form.php** starting with the HTML (**Script 2.2**):

```
<!DOCTYPE html PUBLIC "-//W3C//
→DTD XHTML 1.0 Transitional//EN"
→"http://www.w3.org/TR/xhtml1/DTD/
→xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/
→1999/xhtml" xml:lang="en"
→lang="en">
<head>
  <meta http-equiv="Content-Type"
  →content="text/html;
  →charset=utf-8" />
  <title>Form Feedback</title>
</head>
<body>
```

2. Add the opening PHP tag and create a shorthand version of the form data variables:

```
<?php # Script 2.2 -
→handle_form.php
$name = $_REQUEST['name'];
$email = $_REQUEST['email'];
$comments = $_REQUEST['comments'];
```

Following the rules outlined before, the data entered into the first form input, which is called *name*, will be accessible through the variable **$_REQUEST['name']** (**Table 2.1**). The data entered into the email form input, which has a **name** value of *email*, will be accessible through **$_REQUEST['email']**. The same applies to the comments data. Again, the spelling and capitalization of your variables here must exactly match the corresponding **name** values in the HTML form.

Script 2.2 This script receives and prints out the information entered into an HTML form (Script 2.1).

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD
    XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/
    xhtml1-transitional.dtd">
2   <html xmlns="http://www.w3.org/1999/
    xhtml" xml:lang="en" lang="en">
3   <head>
4      <meta http-equiv="Content-Type"
       content="text/html; charset=utf-8" />
5      <title>Form Feedback</title>
6   </head>
7   <body>
8   <?php # Script 2.2 - handle_form.php
9
10  // Create a shorthand for the form data:
11  $name = $_REQUEST['name'];
12  $email = $_REQUEST['email'];
13  $comments = $_REQUEST['comments'];
14  /* Not used:
15  $_REQUEST['age']
16  $_REQUEST['gender']
17  $_REQUEST['submit']
18  */
19
20  // Print the submitted information:
21  echo "<p>Thank you, <b>$name</b>,
    for the following comments:<br />
22  <tt>$comments</tt></p>
23  <p>We will reply to you at
    <i>$email</i>.</p>\n";
24
25  ?>
26  </body>
27  </html>
```

TABLE 2.1 Form Elements to PHP Variables

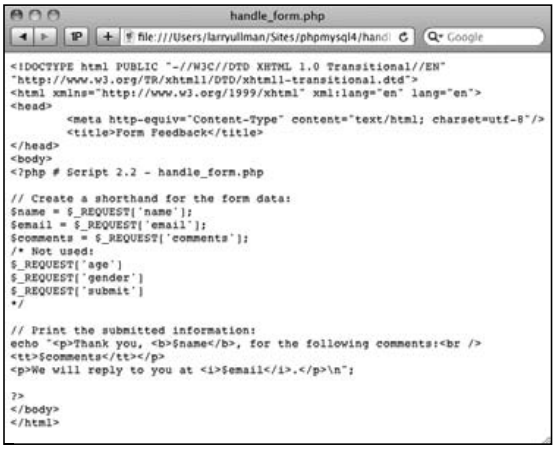
Element Name	Variable Name
name	$_REQUEST['name']
email	$_REQUEST['email']
comments	$_REQUEST['comments']
age	$_REQUEST['age']
gender	$_REQUEST['gender']
submit	$_REQUEST['submit']

Licensed by Dena Laney 233580

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

A To test **handle_form.php**, you must load the form through a URL, then fill it out and submit it.

B The script should display results like this.



C If you see the PHP code after submitting the form, the problem is likely that you did not access the form through a URL.

At this point, you won't make use of the age, gender, and submit form elements.

3. Print out the received name, email, and comments values:

```
echo "<p>Thank you, <b>$name</b>,
→ for the following comments:<br />
<tt>$comments</tt></p>
<p>We will reply to you at
→ <i>$email</i>.</p>\n";
```

The submitted values are simply printed out using the **echo** statement, double quotation marks, and a wee bit of HTML formatting.

4. Complete the page:

```
?>
</body>
</html>
```

5. Save the file as **handle_form.php** and place it in the same Web directory as **form.html.**

6. Test both documents in your Web browser by loading **form.html** through a URL (**http://something**) and then filling out **A** and submitting the form **B**.

Because the PHP script must be run through a URL (see Chapter 1), the form must also be run through a URL. Otherwise, when you go to submit the form, you'll see PHP code **C** instead of the proper result **B**.

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

TIP **$_REQUEST is a special variable type, known as a *superglobal*. It stores all of the data sent to a PHP page through either the GET or POST method, as well as data accessible in cookies. Superglobals will be discussed later in the chapter.**

TIP **If you have any problems with this script, apply the debugging techniques suggested in Chapter 1. If you still can't solve the problem, check out the extended debugging techniques listed in Chapter 8, "Error Handling and Debugging." If you're still stymied, turn to the book's supporting forum for assistance (`www.LarryUllman.com/forums/`).**

TIP **If the PHP script shows blank spaces where a variable's value should have been printed, it means that the variable has no value. The two most likely causes are: you failed to enter a value in the form; or you mis-spelled or mis-capitalized the variable's name.**

TIP **If you see any *Undefined variable: variablename* errors, this is because the variables you refer to have no value and PHP is set on the highest level of error reporting. The previous tip provides suggestions as to why a variable wouldn't have a value. Chapter 8 discusses error reporting in detail.**

TIP **To see how PHP handles the different form input types, print out the `$_REQUEST['age']` and `$_REQUEST['gender']` values D.**

D The values of gender and age correspond to those defined in the form's HTML.

Magic Quotes

Earlier versions of PHP had a feature called *Magic Quotes*, which has since been deprecated and will eventually be removed entirely. Magic Quotes—when enabled—automatically escapes single and double quotation marks found in submitted form data (there were actually three kinds of Magic Quotes, but this one kind is most important here). As an example, Magic Quotes would turn the string *I'm going out* into *I\'m going out*.

The escaping of potentially problematic characters can be useful and even necessary in some situations. But if Magic Quotes are enabled on your PHP installation, you'll see these backslashes when the PHP script prints out the form data. You can undo the effect of Magic Quotes using the **`stripslashes()`** function:

```
$var = stripslashes($var);
```

This function will remove any backslashes found in **`$var`**. This will have the result of turning an escaped submitted string back to its original, non-escaped value.

To use this in **`handle_form.php`** (Script 2.2), you would write:

```
$name = stripslashes($_REQUEST
→['name']);
```

If you're not seeing backslashes added to your form data, then you don't need to worry about Magic Quotes.

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

Conditionals and Operators

PHP's three primary terms for creating conditionals are **if**, **else**, and **elseif** (which can also be written as two words, **else if**).

Every conditional begins with an **if** clause:

```
if (condition) {
        // Do something!
}
```

An **if** can also have an **else** clause:

```
if (condition) {
        // Do something!
} else {
        // Do something else!
}
```

TABLE 2.2 Comparative and Logical Operators

Symbol	Meaning	Type	Example
`==`	is equal to	comparison	`$x == $y`
`!=`	is not equal to	comparison	`$x != $y`
`<`	less than	comparison	`$x < $y`
`>`	greater than	comparison	`$x > $y`
`<=`	less than or equal to	comparison	`$x <= $y`
`>=`	greater than or equal to	comparison	`$x >= $y`
`!`	not	logical	`!$x`
`&&`	and	logical	`$x && $y`
`AND`	and	logical	`$x and $y`
`||`	or	logical	`$x || $y`
`OR`	or	logical	`$x or $y`
`XOR`	and not	logical	`$x XOR $y`

An **elseif** clause allows you to add more conditions:

```
if (condition1) {
        // Do something!
} elseif (condition2) {
        // Do something else!
} else {
        // Do something different!
}
```

If a condition is true, the code in the following curly braces (**{ }**) will be executed. If not, PHP will continue on. If there is a second condition (after an **elseif**), that will be checked for truth. The process will continue—you can use as many **elseif** clauses as you want—until PHP hits an **else**, which will be automatically executed at that point, or until the conditional terminates without an **else**. For this reason, it's important that the **else** always come last and be treated as the default action unless specific criteria—the conditions—are met.

A condition can be true in PHP for any number of reasons. To start, these are true conditions:

- **$var**, if **$var** has a value other than 0, an empty string, **FALSE**, or **NULL**
- **isset($var)**, if **$var** has any value other than **NULL**, including 0, **FALSE**, or an empty string
- **TRUE**, **true**, **True**, etc.

In the second example, a new function, **isset( ),** is introduced. This function checks if a variable is "set," meaning that it has a value other than **NULL** (as a reminder, **NULL** is a special type in PHP, representing no set value). You can also use the comparative and logical operators (**Table 2.2**) in conjunction with parentheses to make more complicated expressions.

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

To use conditionals:

1. Open **handle_form.php** (refer to Script 2.2) in your text editor or IDE, if it is not already.
2. Before the **echo** statement, add a conditional that creates a **$gender** variable (**Script 2.3**):

```
if (isset($_REQUEST['gender'])) {
  $gender = $_REQUEST['gender'];
} else {
  $gender = NULL;
}
```

This is a simple and effective way to validate a form input (particularly a radio button, check box, or select). If the user checks either gender radio button, then **$_REQUEST['gender']** will have a value, meaning that the condition **isset($_REQUEST['gender'])** is true. In such a case, the shorthand version of this variable—**$gender**—is assigned the value of **$_REQUEST['gender']**, repeating the technique used with **$name**, **$email**, and **$comments**. If the user does not click one of the radio buttons, then this condition is not true, and **$gender** is assigned the value of **NULL**, indicating that it has no value. Notice that **NULL** is not in quotes.

Script 2.3 In this remade version of **handle_form.php**, two conditionals are used to validate the gender radio buttons.

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD
    XHTML 1.0 Transitional//EN"
          "http://www.w3.org/TR/xhtml1/DTD/
          xhtml1-transitional.dtd">
2   <html xmlns="http://www.w3.org/1999/
    xhtml" xml:lang="en" lang="en">
3   <head>
4      <meta http-equiv="Content-Type"
       content="text/html; charset=utf-8" />
5      <title>Form Feedback</title>
6   </head>
7   <body>
8   <?php # Script 2.3 - handle_form.php #2
9
10  // Create a shorthand for the form data:
11  $name = $_REQUEST['name'];
12  $email = $_REQUEST['email'];
13  $comments = $_REQUEST['comments'];
14
15  // Create the $gender variable:
16  if (isset($_REQUEST['gender'])) {
17     $gender = $_REQUEST['gender'];
18  } else {
19     $gender = NULL;
20  }
21
22  // Print the submitted information:
23  echo "<p>Thank you, <b>$name</b>,
    for the following comments:<br />
24  <tt>$comments</tt></p>
25  <p>We will reply to you at
    <i>$email</i>.</p>\n";
26
27  // Print a message based upon the
    gender value:
28  if ($gender == 'M') {
29     echo '<p><b>Good day, Sir!</b>
       </p>';
30  } elseif ($gender == 'F') {
31     echo '<p><b>Good day, Madam!
       </b></p>';
32  } else { // No gender selected.
33     echo '<p><b>You forgot to enter
       your gender!</b></p>';
34  }
35
36  ?>
37  </body>
38  </html>
```


PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

Thank you, **Marge Simpson**, for the following comments:
`Bart, don't use the Touch of Death on your sister.`

We will reply to you at *marge@example.edu*.

Good day, Madam!

A The gender-based conditional prints a different message for each choice in the form.

Thank you, **Ralph Waldo Emerson**, for the following comments:
`Difficulties exist to be surmounted.`

We will reply to you at *rwe@example.org*.

Good day, Sir!

B The same script will produce different salutations (compare with **A**) when the gender value changes.

Thank you, **Alistair Cooke**, for the following comments:
`A professional is someone who can do his best work when he doesn't feel like it.`

We will reply to you at *a.cooke@example.net*.

You forgot to enter your gender!

C If no gender was selected, a message is printed indicating the oversight to the user.

3. After the **echo** statement, add another conditional that prints a message based upon **$gender**'s value:

```
if ($gender == 'M') {
  echo '<p><b>Good day, Sir!</b>
  → </p>';
} elseif ($gender == 'F') {
  echo '<p><b>Good day, Madam!
  → </b></p>';
} else {
  echo '<p><b>You forgot to enter
  → your gender!</b></p>';
}
```

This **if-elseif-else** conditional looks at the value of the **$gender** variable and prints a different message for each possibility. It's very important to remember that the double equals sign (**==**) means equals, whereas a single equals sign (**=**) assigns a value. The distinction is important because the condition **$gender == 'M'** may or may not be true, but **$gender = 'M'** will always be true.

Also, the values used here—*M* and *F*—must be exactly the same as those in the HTML form (the values for each radio button). Equality is a case-sensitive comparison with strings, so *m* will not equal *M*.

4. Save the file, place it in your Web directory, and test it in your Web browser **A**, **B**, and **C**.

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

TIP **Although PHP has no strict formatting rules, it's standard procedure and good programming form to make it clear when one block of code is a subset of a conditional. Indenting the block is the norm.**

TIP **You can—and frequently will—nest conditionals (place one inside another).**

TIP **The first conditional in this script (the `isset( )`) is a perfect example of how to use a default value. The assumption (the `else`) is that `$gender` has a `NULL` value unless the one condition is met: that `$_REQUEST['gender']` is set.**

TIP **The curly braces used to indicate the beginning and end of a conditional are not required if you are executing only one statement. I would recommend that you almost always use them, though, as a matter of clarity.**

TIP **Both *and* and *or* have two representative operators, with slight, technical differences between them. For no particular reason, I tend to use `&&` and `||` instead of `AND` and `OR`.**

TIP **XOR is called the *exclusive or* operator. The conditional `$x XOR $y` is true if `$x` is true or if `$y` is true, but not both.**

Switch

PHP has another type of conditional, called the **`switch`**, best used in place of a long **`if-elseif-else`** conditional. The syntax of **`switch`** is

```
switch ($variable) {
    case 'value1':
          // Do this.
          break;
    case 'value2':
          // Do this instead.
          break;
default:
    // Do this then.
    break;
}
```

The **`switch`** conditional compares the value of **`$variable`** to the different cases. When it finds a match, the following code is executed, up until the **`break`**. If no match is found, the **`default`** is executed, assuming it exists (it's optional). The **`switch`** conditional is limited in its usage in that it can only check a variable's value for equality against certain cases; more complex conditions cannot be easily checked.

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

Validating Form Data

A critical concept related to handling HTML forms is that of validating form data. In terms of both error management and security, you should absolutely never trust the data being submitted by an HTML form. Whether erroneous data is purposefully malicious or just unintentionally inappropriate, it's up to you—the Web architect—to test it against expectations.

Validating form data requires the use of conditionals and any number of functions, operators, and expressions. One standard function to be used is **`isset()`**, which tests if a variable has a value (including 0, **`FALSE`**, or an empty string, but not **`NULL`**). You saw an example of this in the preceding script.

One issue with the **`isset()`** function is that an empty string tests as true, meaning that **`isset()`** is not an effective way to validate text inputs and text boxes from an HTML form. To check that a user typed something into textual elements, you can use the **`empty()`** function. It checks if a variable has an *empty* value: an empty string, 0, **`NULL`**, or **`FALSE`**.

The first aim of form validation is seeing if *something* was entered or selected in form elements. The second goal is to ensure that submitted data is of the right type (numeric, string, etc.), of the right format (like an email address), or a specific acceptable value (like **`$gender`** being equal to either *M* or *F*). As handling forms is a main use of PHP, validating form data is a point that will be re-emphasized time and again in subsequent chapters. But first, let's create a new **`handle_form.php`** to make sure variables have values before they're referenced (there will be enough changes in this version that simply updating Script 2.3 doesn't make sense).

To validate your forms:

1. Begin a new PHP script in your text editor or IDE, to be named **`handle_form.php`** starting with the initial HTML (**Script 2.4**):

```
<!DOCTYPE html PUBLIC "-//W3C//
→ DTD XHTML 1.0 Transitional//EN"
→ "http://www.w3.org/TR/xhtml1/DTD/
xhtml1-transitional.dtd">
```

continues on page 50

Script 2.4 Validating HTML form data before you use it is critical to Web security and achieving professional results. Here, conditionals check that every referenced form element has a value.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/
DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
   <title>Form Feedback</title>
   <style type="text/css" title="text/css" media="all">
   .error {
      font-weight: bold;
      color: #C00;
   }
   </style>
</head>
<body>
<?php # Script 2.4 - handle_form.php #3

```

code continues on next page

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

Script 2.4 *continued*

```
16  // Validate the name:
17  if (!empty($_REQUEST['name'])) {
18     $name = $_REQUEST['name'];
19  } else {
20     $name = NULL;
21     echo '<p class="error">You forgot to enter your name!</p>';
22  }
23
24  // Validate the email:
25  if (!empty($_REQUEST['email'])) {
26     $email = $_REQUEST['email'];
27  } else {
28     $email = NULL;
29     echo '<p class="error">You forgot to enter your email address!</p>';
30  }
31
32  // Validate the comments:
33  if (!empty($_REQUEST['comments'])) {
34     $comments = $_REQUEST['comments'];
35  } else {
36     $comments = NULL;
37     echo '<p class="error">You forgot to enter your comments!</p>';
38  }
39
40  // Validate the gender:
41  if (isset($_REQUEST['gender'])) {
42
43     $gender = $_REQUEST['gender'];
44
45     if ($gender == 'M') {
46        echo '<p><b>Good day, Sir!</b></p>';
47     } elseif ($gender == 'F') {
48        echo '<p><b>Good day, Madam!</b></p>';
49     } else { // Unacceptable value.
50        $gender = NULL;
51        echo '<p class="error">Gender should be either "M" or "F"!</p>';
52     }
53
54  } else { // $_REQUEST['gender'] is not set.
55     $gender = NULL;
56     echo '<p class="error">You forgot to select your gender!</p>';
57  }
58
59  // If everything is OK, print the message:
60  if ($name && $email && $gender && $comments) {
61
62     echo "<p>Thank you, <b>$name</b>, for the following comments:<br />
63     <tt>$comments</tt></p>
64     <p>We will reply to you at <i>$email</i>.</p>\n";
65
66  } else { // Missing form value.
67     echo '<p class="error">Please go back and fill out the form again.</p>';
68  }
69
70  ?>
71  </body>
72  </html>
```


PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

```
<html xmlns="http://www.w3.org/
→1999/xhtml" xml:lang="en"
→lang="en">
<head>
  <meta http-equiv="Content-Type"
  →content="text/html;
  →charset=utf-8" />
  <title>Form Feedback</title>
</head>
<body>
```

2. Within the HTML **head**, add some CSS code:

```
<style type="text/css"
→title="text/css" media="all">
.error {
  font-weight: bold;
  color: #C00;
}
</style>
```

This code defines one CSS class, called *error*. Any HTML element that has this class name will be formatted in a bold, red color (which will be more apparent in your Web browser than in this black-and-white book).

3. In PHP block, check if the name was entered:

```
if (!empty($_REQUEST['name'])) {
  $name = $_REQUEST['name'];
} else {
  $name = NULL;
  echo '<p class="error">You
  →forgot to enter your name!</p>';
}
```

A simple way to check that a form text input was filled out is to use the **empty( )** function. If **$_REQUEST['name']** has a value other than an empty string, 0, **NULL**, or **FALSE**, assume that their name was entered and a shorthand variable is assigned that value. If **$_REQUEST['name']** is empty, the **$name** variable is set to **NULL** and an error message is printed. This error message uses the CSS class.

4. Repeat the same process for the email address and comments:

```
if (!empty($_REQUEST['email'])) {
  $email = $_REQUEST['email'];
} else {
  $email = NULL;
  echo '<p class="error">You
  →forgot to enter your email
  →address!</p>';
}
if (!empty($_REQUEST['comments'])) {
  $comments = $_REQUEST['comments'];
} else {
  $comments = NULL;
  echo '<p class="error">You
  →forgot to enter your
  →comments!</p>';
}
```

Both variables receive the same treatment as **$_REQUEST['name']** in Step 3.

5. Begin validating the gender variable:

```
if (isset($_REQUEST['gender'])) {
  $gender = $_REQUEST['gender'];
```

The validation of the gender is a two-step process. First, check if it has a value or not, using **isset( )**. This starts the main **if-else** conditional, which otherwise behaves like those for the name, email address, and comments.

continues on next page

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

6. Check **$gender** against specific values:

```
if ($gender == 'M') {
  $greeting = '<p><b>Good day,
  → Sir!</b></p>';
} elseif ($gender == 'F') {
  $greeting = '<p><b>Good day,
  → Madam!</b></p>';
} else {
  $gender = NULL;
  echo '<p class="error">Gender
  → should be either "M" or
  → "F"!</p>';
}
```

Within the gender **if** clause is a nested **if-elseif-else** conditional that tests the variable's value against what's acceptable. This is the second part of the two-step gender validation.

The conditions themselves are the same as those in the last script. If gender does not end up being equal to either *M* or *F*, a problem occurred and an error message is printed. The **$gender** variable is also set to **NULL** in such cases, because it has an unacceptable value.

If **$gender** does have a valid value, a gender-specific message is assigned to a new variable, so that the message can be printed later in the script.

7. Complete the main gender **if-else** conditional:

```
} else {
  $gender = NULL;
  echo '<p class="error">You
  forgot to select your
  → gender!</p>';
}
```

This **else** clause applies if **$_REQUEST['gender']** is not set. The complete, nested conditionals (see lines 41–57 of Script 2.4) successfully check every possibility:

- **$_REQUEST['gender']** is not set
- **$_REQUEST['gender']** has a value of *M*
- **$_REQUEST['gender']** has a value of *F*
- **$_REQUEST['gender']** has some other value

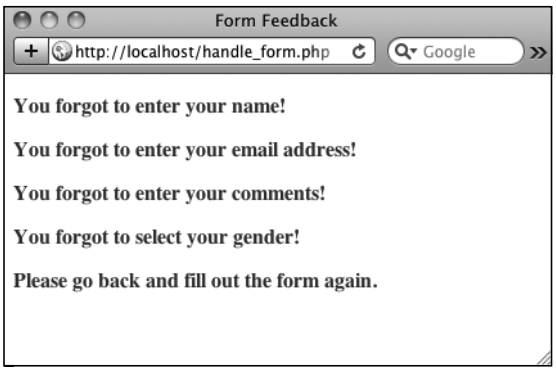
You may wonder how this last case may be possible, considering the values are set in the HTML form. If a malicious user creates their own form that gets submitted to your **handle_form.php** script (which is very easy to do), they could give **$_REQUEST['gender']** any value they want.

8. Print messages indicating the validation results:

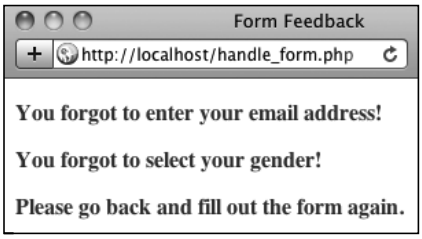
```
if ($name && $email && $gender
→ && $comments) {
  echo "<p>Thank you, <b>$name
  → </b>, for the following
  → comments:<br />
  <tt>$comments</tt></p>
  <p>We will reply to you at
  → <i>$email</i>.</p>\n";
  echo $greeting;
} else {
  echo '<p class="error">Please
  → go back and fill out the form
  → again.</p>';
}
```

Licensed by Dena Laney 2335846

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.



A The script now checks that every form element was filled out (except the age) and reports on those that weren't.



B If even one or two fields were skipped, the *Thank you* message is not printed.

The main condition is true if every listed variable has a true value. Each variable will have a value if it passed its test but have a value of `NULL` if it didn't. If every variable has a value, the form was completed, so the *Thank you* message will be printed, as will the gender-specific greeting. If any of the variables are `NULL`, the second message will be printed (**A** and **B**).

9. Close the PHP section and complete the HTML page:

```
?>
</body>
</html>
```

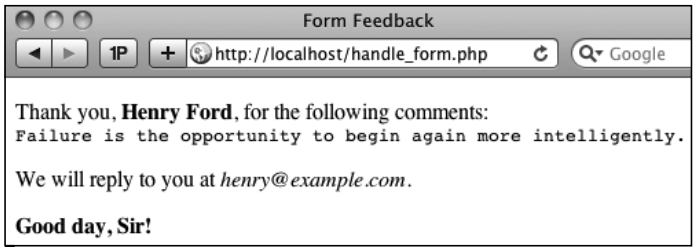
10. Save the file as `handle_form.php`, place it in the same Web directory as `form.html`, and test it in your Web browser.

    Fill out the form to different levels of completeness to test the new script **C**.

TIP **To test if a submitted value is a number, use the `is_numeric( )` function.**

TIP **In Chapter 14, "Perl-Compatible Regular Expressions," you'll see how to validate form data using regular expressions.**

TIP **It's considered good form (pun intended) to let a user know which fields are required when they're filling out the form, and where applicable, the format of that field (like a date or a phone number).**



C If the form was completed properly, the script behaves as it previously had.

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

Introducing Arrays

Chapter 1 introduced two *scalar* (single valued) variable types: strings and numbers. Now it's time to learn about another type, the *array*. Unlike strings and numbers, an *array* can hold multiple, separate pieces of information. An array is therefore like a list of values, each value being a string or a number or even another array.

Arrays are structured as a series of *key-value* pairs, where one pair is an item or *element* of that array. For each item in the list, there is a *key* (or *index*) associated with it (**Table 2.3**).

PHP supports two kinds of arrays: *indexed*, which use numbers as the keys (as in Table 2.3), and *associative*, which use strings as keys (**Table 2.4**). As in most programming languages, with indexed arrays, arrays will begin with the first index at 0, unless you specify the keys explicitly.

An array follows the same naming rules as any other variable. This means that, offhand, you might not be able to tell that **`$var`** is an array as opposed to a string or number. The important syntactical difference arises when accessing individual array elements.

To refer to a specific value in an array, start with the array variable name, followed by the key within square brackets:

```
$band = $artists[0]; // The Mynabirds
echo $states['MD']; // Maryland
```

You can see that the array keys are used like other values in PHP: numbers (e.g., 0) are never quoted, whereas strings (*MD*) must be.

TABLE 2.3 Array Example 1: `$artists`

Key	Value
0	The Mynabirds
1	Jeremy Messersmith
2	The Shins
3	Iron and Wine
4	Alexi Murdoch

TABLE 2.4 Array Example 2: `$states`

Key	Value
MD	Maryland
PA	Pennsylvania
IL	Illinois
MO	Missouri
IA	Iowa

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

Superglobal Arrays

PHP includes several predefined arrays called the *superglobal* variables. They are: **$_GET**, **$_POST**, **$_REQUEST**, **$_SERVER**, **$_ENV**, **$_SESSION**, and **$_COOKIE**.

The **$_GET** variable is where PHP stores all of the values sent to a PHP script via the **GET** method (possibly but not necessarily from an HTML form). **$_POST** stores all of the data sent to a PHP script from an HTML form that uses the **POST** method. Both of these—along with **$_COOKIE**—are subsets of **$_REQUEST**, which you've been using.

$_SERVER, which was used in Chapter 1, stores information about the server PHP is running on, as does **$_ENV**. **$_SESSION** and **$_COOKIE** will both be discussed in Chapter 12, "Cookies and Sessions."

One aspect of good security and programming is to be precise when referring to a variable. This means that, although you can use **$_REQUEST** to access form data submitted through the **POST** method, **$_POST** would be more accurate.

Because arrays use a different syntax than other variables, and can contain multiple values, printing them can be trickier. This will not work ❶:

```
echo "My list of states: $states";
```

However, printing an individual element's value is simple if it uses indexed (numeric) keys:

```
echo "The first artist is
→ $artists[0].";
```

But if the array uses strings for the keys, the quotes used to surround the key will muddle the syntax. The following code will cause a parse error ❷:

```
echo "IL is $states['IL']."; // BAD!
```

To fix this, wrap the array name and key in curly braces when an array uses strings for its keys ❸:

```
echo "IL is {$states['IL']}.";
```

If arrays seem slightly familiar to you already, that's because you've already worked with two: **$_SERVER** (in Chapter 1) and **$_REQUEST** (in this chapter). To acquaint you with another array and to practice printing array values directly, one final, but basic, version of the **handle_form.php** page will be created using the more specific **$_POST** array (see the sidebar on "Superglobal Arrays").

My list of states: Array

❶ Attempting to print an array using only the variable's name results in the word *Array* being printed.

Parse error: syntax error, unexpected T_ENCAPSED_AND_WHITESPACE, expecting T_STRING or T_VARIABLE or T_NUM_STRING in **/Users/larryullman/Sites/phpmysql4/test.php** on line **13**

❷ Attempting to print an element in an associative array *without* using curly braces results in a parse error.

IL is Illinois.

❸ Attempting to print an element in an associative array *while* using curly braces works as desired.

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

To use arrays:

1. Begin a new PHP script in your text editor or IDE, to be named **handle_form.php** starting with the initial HTML (**Script 2.5**):

```
<!DOCTYPE html PUBLIC "-//W3C//
→DTD XHTML 1.0 Transitional//EN"
→"http://www.w3.org/TR/xhtml1/DTD/
→xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/
→1999/xhtml" xml:lang="en" lang="en">
<head>
  <meta http-equiv="Content-Type"
  →content="text/html;
  →charset=utf-8" />
  <title>Form Feedback</title>
</head>
<body>
<?php # Script 2.5 -
→handle_form.php #4
```

2. Perform some basic form validation:

```
if ( !empty($_POST['name']) &&
→!empty($_POST['comments']) &&
→!empty($_POST['email']) ) {
```

In the previous version of this script, the values are accessed by referring to the **$_REQUEST** array. But since these variables come from a form that uses the POST method (see Script 2.1), **$_POST** would be a more exact, and therefore more secure, reference.

This conditional checks that these three text inputs are all not empty. Using the *and* operator (**&&**), the entire conditional is only true if each of the three subconditionals is true.

3. Print the message:

```
echo "<p>Thank you, <b>{$_POST
→['name']}</b>, for the following
→comments:<br />
<tt>{$_POST['comments']}</tt></p>
<p>We will reply to you at
→<i>{$_POST['email']}</i>.</p>\n";
```

After you comprehend the concept of an array, you still need to master the syntax involved in printing one. When printing an array element that uses a

Script 2.5 The superglobal variables, like **$_POST** here, are just one type of array you'll use in PHP.

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/
    DTD/xhtml1-transitional.dtd">
2   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
3   <head>
4      <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
5      <title>Form Feedback</title>
6   </head>
7   <body>
8   <?php # Script 2.5 - handle_form.php #4
9
10  // Print the submitted information:
11  if ( !empty($_POST['name']) && !empty($_POST['comments']) && !empty($_POST['email']) ) {
12     echo "<p>Thank you, <b>{$_POST['name']}</b>, for the following comments:<br />
13  <tt>{$_POST['comments']}</tt></p>
14  <p>We will reply to you at <i>{$_POST['email']}</i>.</p>\n";
15  } else { // Missing form value.
16     echo '<p>Please go back and fill out the form again.</p>';
17  }
18  ?>
19  </body>
20  </html>
```


PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

string for its key, use the curly braces (as in **{$_POST['name']}** here) to avoid parse errors.

4. Complete the conditional begun in Step 2:

```
} else {
  echo '<p>Please go back and
  → fill out the form again.</p>';
}
```

If any of the three subconditionals in Step 2 is not true (which is to say, if any of the variables has an empty value), then this **else** clause applies and an error message is printed ❶.

5. Complete the PHP and HTML code:

```
?>
</body>
</html>
```

6. Save the file as **handle_form.php**, place it in the same Web directory as **form.html**, and test it in your Web browser ❷.

TIP Because PHP is lax with its variable structures, an array can even use a combination of numbers and strings as its keys. The only important rule is that the keys of an array must each be unique.

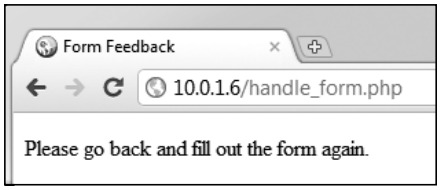
TIP If you find the syntax of accessing superglobal arrays directly to be confusing (e.g., $_POST['name']), you can continue to use the shorthand technique at the top of your scripts as you have been:

```
$name = $_POST['name'];
```

In this script, you would then need to change the conditional and the echo statement to refer to $name et al.

TIP You only need to use the curly brackets to surround an associated array used within quotation marks. All of these array references are fine:

```
echo $_POST['name'];
echo "The first item is $item[0].";
$total = number_format($cart['total']);
```



❶ If any of the three tested form inputs is empty, this generic error message is printed.

❷ The fact that the script now uses the **$_POST** array has no effect on the visible result.

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

Creating arrays

The preceding example uses a PHP-generated array, but there will frequently be times when you want to create your own. There are two primary ways to define your own array. First, you could add an element at a time to build one:

```
$band[] = 'Jemaine';
$band[] = 'Bret';
$band[] = 'Murray';
```

As arrays are indexed starting at 0, **$band[0]** has a value of *Jemaine*; **$band[1]**, *Bret*, and **$band[2]**, *Murray*.

Alternatively, you can specify the key when adding an element. But it's important to understand that if you specify a key and a value already exists indexed with that same key, the new value will overwrite the existing one:

```
$band['fan'] = 'Mel';
$band['fan'] = 'Dave'; // New value
$fruit[2] = 'apple';
$fruit[2] = 'orange'; // New value
```

Instead of adding one element at a time, you can use the **array( )** function to build an entire array in one step:

```
$states = array (
'IA' => 'Iowa',
'MD' => 'Maryland'
);
```

(As PHP is generally insensitive to white space, you can use this function over multiple lines for added clarity.)

The **array( )** function can be used whether or not you explicitly set the key:

```
$artists = array ('Clem Snide',
→'Shins', 'Eels');
```

Or, if you set the first numeric key value, the added values will be keyed incrementally thereafter:

```
$days = array (1 => 'Sun', 'Mon',
→'Tue');
echo $days[3]; // Tue
```

The **array( )** function is also used to initialize an array, prior to referencing it:

```
$tv = array( );
$tv[] = 'Flight of the Conchords';
```

Initializing an array (or any variable) in PHP isn't required, but it makes for clearer code and can help avoid errors.

Finally, if you want to create an array of sequential numbers, you can use the **range( )** function:

```
$ten = range (1, 10);
```

Accessing entire arrays

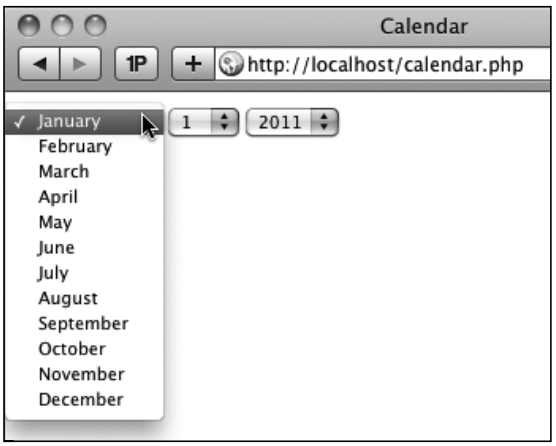
You've already seen how to access individual array elements using its keys (e.g., **$_POST['email']**). This works when you know exactly what the keys are or if you want to refer to only a single element. To access every array element, use the **foreach** loop:

```
foreach ($array as $value) {
        // Do something with $value.
}
```

The **foreach** loop will iterate through every element in **$array**, assigning each element's value to the **$value** variable. To access both the keys and values, use

```
foreach ($array as $key => $value) {
        echo "The value at $key is
        →$value.";
}
```


PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.



F These pull-down menus will be created using arrays and the **foreach** loop.

Script 2.6 This form uses arrays to dynamically create three pull-down menus.

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD
    XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/
    xhtml1-transitional.dtd">
2   <html xmlns="http://www.w3.org/1999/
    xhtml" xml:lang="en" lang="en">
3   <head>
4      <meta http-equiv="Content-Type"
       content="text/html; charset=utf-8" />
5      <title>Calendar</title>
6   </head>
7   <body>
8   <form action="calendar.php"
    method="post">
9   <?php # Script 2.6 - calendar.php
10
11  // This script makes three pull-down
    menus
12  // for an HTML form: months, days,
    years.
13
14  // Make the months array:
15  $months = array (1 => 'January',
    'February', 'March', 'April', 'May',
    'June', 'July', 'August', 'September',
    'October', 'November', 'December');
16
17  // Make the days and years arrays:
18  $days = range (1, 31);
19  $years = range (2011, 2021);
```

code continues on next page

(You can use any valid variable name in place of **$key** and **$value**, like just **$k** and **$v**, if you'd prefer.)

Using arrays, this next script will demonstrate how easy it is to make a set of form pull-down menus for selecting a date F.

To create and access arrays:

1. Begin a new PHP document in your text editor or IDE, to be named **calendar.php** starting with the initial HTML (**Script 2.6**):

```
<!DOCTYPE html PUBLIC "-//W3C//
→DTD XHTML 1.0 Transitional//EN"
→"http://www.w3.org/TR/xhtml1/DTD/
→xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/
→1999/xhtml" xml:lang="en"
→lang="en">
<head>
  <meta http-equiv="Content-Type"
  →content="text/html;
  →charset=utf-8" />
  <title>Calendar</title>
</head>
<body>
<form action="calendar.php"
→method="post">
<?php # Script 2.6 - calendar.php
```

One thing to note here is that even though the page won't contain a complete HTML form, the form tags are still required to create the pull-down menus.

continues on next page

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

2. Create an array for the months:

```
$months = array (1 => 'January',
→'February', 'March', 'April',
→'May', 'June', 'July', 'August',
→'September', 'October',
→'November', 'December');
```

This first array will use numbers for the keys, from 1 to 12. Since the value of the first key is specified, the following values will be indexed incrementally (in other words, the `1 =>` code creates an array indexed from 1 to 12, instead of from 0 to 11).

3. Create the arrays for the days of the month and the years:

```
$days = range (1, 31);
$years = range (2011, 2021);
```

Using the `range( )` function, you can easily make an array of numbers.

4. Generate the month pull-down menu:

```
echo '<select name="month">';
foreach ($months as $key =>
→$value) {
  echo "<option value=\"$key\">
  →$value</option>\n";
}
echo '</select>';
```

The `foreach` loop can quickly generate all of the HTML code for the month pull-down menu. Each execution of the loop will create a line of code like `<option value="1">January</option>` Ⓖ.

5. Generate the day and year pull-down menus:

```
echo '<select name="day">';
foreach ($days as $value) {
  echo "<option value=\"$value\">
  →$value</option>\n";
}
```

Script 2.6 *continued*

```
20
21  // Make the months pull-down menu:
22  echo '<select name="month">';
23  foreach ($months as $key =>
    $value) {
24     echo "<option value=\"$key\">
       $value</option>\n";
25  }
26  echo '</select>';
27
28  // Make the days pull-down menu:
29  echo '<select name="day">';
30  foreach ($days as $value) {
31     echo "<option value=\"$value\">
       $value</option>\n";
32  }
33  echo '</select>';
34
35  // Make the years pull-down menu:
36  echo '<select name="year">';
37  foreach ($years as $value) {
38     echo "<option value=\"$value\">
       $value</option>\n";
39  }
40  echo '</select>';
41
42  ?>
43  </form>
44  </body>
45  </html>
```

```
8  <form action="calendar.php" method="post">
9  <select name="month"><option value="1">January</option>
10 <option value="2">February</option>
11 <option value="3">March</option>
12 <option value="4">April</option>
13 <option value="5">May</option>
14 <option value="6">June</option>
15 <option value="7">July</option>
16 <option value="8">August</option>
17 <option value="9">September</option>
18 <option value="10">October</option>
19 <option value="11">November</option>
20 <option value="12">December</option>
21 </select><select name="day"><option value="1">1</option>
22 <option value="2">2</option>
23 <option value="3">3</option>
24 <option value="4">4</option>
25 <option value="5">5</option>
26 <option value="6">6</option>
27 <option value="7">7</option>
```

Ⓖ Most of the HTML source was generated by just a few lines of PHP.

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

```
echo '</select>';
echo '<select name="year">';
foreach ($years as $value) {
  echo "<option value=\"$value\">
  → $value</option>\n";
}
echo '</select>';
```

Unlike the month example, both the day and year pull-down menus will use the same data for the option's value and label (a number, G). For that reason, there's no need to also fetch the array's key with each loop iteration.

6. Close the PHP, the form tag, and the HTML page:

```
?>
</form>
</body>
</html>
```

7. Save the file as **calendar.php**, place it in your Web directory, and test it in your Web browser.

TIP To determine the number of elements in an array, use `count( )`:

```
$num = count($array);
```

TIP The `range( )` function can also create an array of sequential letters:

```
$alphabet = range ('a', 'z');
```

TIP An array's key can be multiple-worded strings, such as *first name* or *phone number*.

TIP The `is_array( )` function confirms that a variable is of the array type.

TIP If you see an *Invalid argument supplied for foreach()* error message, that means you are trying to use a `foreach` loop on a variable that is not an array.

Multidimensional arrays

When introducing arrays, I mentioned that an array's values could be any combination of numbers, strings, and even other arrays. This last option—an array consisting of other arrays—creates a *multidimensional array*.

Multidimensional arrays are much more common than you might expect but remarkably easy to work with. As an example, start with an array of prime numbers:

```
$primes = array(2, 3, 5, 7, …);
```

Then create an array of *sphenic* numbers (don't worry: I had no idea what a sphenic number was either; I had to look it up):

```
$sphenic = array(30, 42, 66, 70, …);
```

These two arrays could be combined into one multidimensional array like so:

```
$numbers = array ('Primes' =>
→ $primes, 'Sphenic' => $sphenic);
```

Now, **$numbers** is a multidimensional array. To access the prime numbers sub-array, refer to **$numbers['Primes']**. To access the prime number 5, use **$numbers['Primes'][2]** (it's the third element in the array, but the array starts indexing at 0). To print out one of these values, surround the whole construct in curly braces:

```
echo "The first sphenic number is
→ {$numbers['Sphenic'][0]}.";
```

Of course, you can also access multidimensional arrays using the **foreach** loop, nesting one inside another if necessary. This next example will do just that.

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

To use multidimensional arrays:

1. Begin a new PHP document in your text editor or IDE, to be named **multi.php** beginning with the initial HTML (**Script 2.7**):

```
<!DOCTYPE html PUBLIC "-//W3C//
→DTD XHTML 1.0 Transitional//EN"
→"http://www.w3.org/TR/xhtml1/DTD/
→xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/
→1999/xhtml" xml:lang="en"
→lang="en">
<head>
  <meta http-equiv="Content-Type"
  →content="text/html;
  →charset=utf-8" />
  <title>Multidimensional
  →Arrays</title>
</head>
<body>
<p>Some North American States,
→Provinces, and Territories:</p>
<?php # Script 2.7 - multi.php
```

This PHP page will print out some of the states, provinces, and territories found in the three North American countries (Mexico, the United States, and Canada ❶).

2. Create an array of Mexican states:

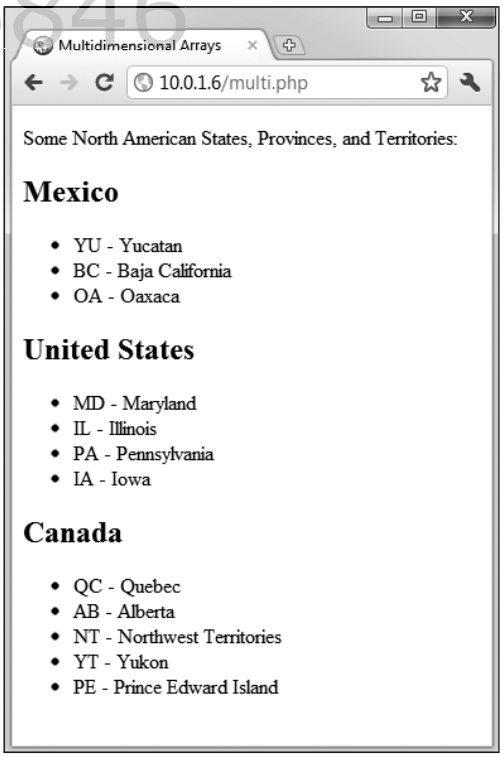
```
$mexico = array(
'YU' => 'Yucatan',
'BC' => 'Baja California',
'OA' => 'Oaxaca'
);
```

This is an associative array, using the state's postal abbreviation as its key. The state's full name is the element's value. This is obviously an incomplete list, just used to demonstrate the concept.

Script 2.7 The multidimensional array is created by using other arrays for its values. Two **foreach** loops, one nested inside of the other, can access every array element.

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD
    XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/
    xhtml1-transitional.dtd">
2   <html xmlns="http://www.w3.org/1999/
    xhtml" xml:lang="en" lang="en">
3   <head>
4      <meta http-equiv="Content-Type"
       content="text/html; charset=utf-8" />
5      <title>Multidimensional Arrays</
       title>
6   </head>
7   <body>
8   <p>Some North American States,
    Provinces, and Territories:</p>
9   <?php # Script 2.7 - multi.php
10
```

code continues on next page



❶ The end result of running this PHP page (Script 2.7), where each country is printed, followed by an abbreviated list of its states, provinces, and territories.

Licensed by Dena Laney 2335846

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

Script 2.7 *continued*

```
// Create one array:
$mexico = array(
'YU' => 'Yucatan',
'BC' => 'Baja California',
'OA' => 'Oaxaca'
);

// Create another array:
$us = array (
'MD' => 'Maryland',
'IL' => 'Illinois',
'PA' => 'Pennsylvania',
'IA' => 'Iowa'
);

// Create a third array:
$canada = array (
'QC' => 'Quebec',
'AB' => 'Alberta',
'NT' => 'Northwest Territories',
'YT' => 'Yukon',
'PE' => 'Prince Edward Island'
);

// Combine the arrays:
$n_america = array(
'Mexico' => $mexico,
'United States' => $us,
'Canada' => $canada
);

// Loop through the countries:
foreach ($n_america as $country =>
$list) {

   // Print a heading:
   echo "<h2>$country</h2><ul>";

   // Print each state, province, or
   territory:
   foreach ($list as $k => $v) {
      echo "<li>$k - $v</li>\n";
   }

   // Close the list:
   echo '</ul>';

} // End of main FOREACH.

?>
</body>
</html>
```

3. Create the second and third arrays:

```
$us = array (
'MD' => 'Maryland',
'IL' => 'Illinois',
'PA' => 'Pennsylvania',
'IA' => 'Iowa'
);
$canada = array (
'QC' => 'Quebec',
'AB' => 'Alberta',
'NT' => 'Northwest Territories',
'YT' => 'Yukon',
'PE' => 'Prince Edward Island'
);
```

4. Combine all of the arrays into one:

```
$n_america = array(
'Mexico' => $mexico,
'United States' => $us,
'Canada' => $canada
);
```

You don't have to create three arrays and then assign them to a fourth in order to make the desired multidimensional array, but I think it's easier to read and understand this way (defining a multidimensional array in one step makes for some ugly code).

The **$n_america** array now contains three elements. The key for each element is a string, which is the country's name. The value for each element is the array of states, provinces, and territories found within that country.

5. Begin the primary **foreach** loop:

```
foreach ($n_america as $country
→ => $list) {
  echo "<h2>$country</h2><ul>";
```

continues on next page

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

Following the syntax outlined earlier, this loop will access every element of **$n_america**. This means that this loop will run three times. Within each iteration of the loop, the **$country** variable will store the **$n_america** array's key (*Mexico*, *Canada*, or *United States*). Also within each iteration of the loop, the **$list** variable will store the element's value (the equivalent of **$mexico**, **$us**, and **$canada**).

To print out the results, the loop begins by printing the country's name within **H2** tags. Because the states and so forth should be displayed as an HTML list, the initial unordered list tag (**<ul>**) is printed as well.

6. Create a second **foreach** loop:

```
foreach ($list as $k => $v) {
  echo "<li>$k - $v</li>\n";
}
```

This loop will run through each sub-array (first **$mexico**, then **$us**, and then **$canada**). With each iteration of this loop, **$k** will store the abbreviation and **$v** the full name. Both are printed out within HTML list tags. The newline character is also used, to better format the HTML source code.

7. Complete the outer **foreach** loop:

```
  echo '</ul>';
} // End of main FOREACH.
```

After the inner **foreach** loop is done, the outer **foreach** loop has to close the unordered list begun in Step 5.

8. Complete the PHP and HTML:

```
?>
</body>
</html>
```

9. Save the file as **multi.php**, place it in your Web directory, and test it in your Web browser (H).

10. If you want, check out the HTML source code to see what PHP created.

TIP Multidimensional arrays can also come from an HTML form. For example, if a form has a series of checkboxes with the name *interests[]*—

```
<input type="checkbox" name=
→"interests[]" value="Music" /> Music
<input type="checkbox" name=
→"interests[]" value="Movies" /> Movies
<input type="checkbox" name=
→"interests[]" value="Books" /> Books
```

—the $_POST variable in the receiving PHP page will be multidimensional. $_POST['interests'] will be an array, with $_POST['interests'][0] storing the value of the first checked box (e.g., Movies), $_POST['interests'][1] storing the second (Books), etc. Note that only the checked boxes will get passed to the PHP page.

TIP You can also end up with a multidimensional array if an HTML form's select menu allows for multiple selections:

```
<select name="interests[]"
→multiple="multiple">
   <option value="Music">Music
   →</option>
   <option value="Movies">Movies
   →</option>
   <option value="Books">Books
   →</option>
   <option value="Napping">Napping
   →</option>
</select>
```

Again, only the selected values will be passed to the PHP page.

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

Arrays and Strings

Because arrays and strings are so commonly used together, PHP has two functions for converting between them:

```
$array = explode (separator,
→ $string);
$string = implode (glue, $array);
```

The key to using and understanding these two functions is the *separator* and *glue* relationships. When turning an array into a string, you establish the glue—the characters or code that will be inserted between the array values in the generated string. Conversely, when turning a string into an array, you specify the separator, which is the token that marks what should become separate array elements. For example, start with a string:

```
$s1 = 'Mon-Tue-Wed-Thu-Fri';
$days_array = explode ('-', $s1);
```

The **$days_array** variable is now a five-element array, with *Mon* indexed at **0**, *Tue* indexed at **1**, etc.

```
$s2 = implode (', ', $days_array);
```

The **$s2** variable is now a comma-separated list of days: *Mon, Tue, Wed, Thu, Fri*.

Sorting arrays

One of the many advantages arrays have over the other variable types is the ability to sort them. PHP includes several functions you can use for sorting arrays, all simple in syntax:

```
$names = array ('Moe', 'Larry',
→ 'Curly');
sort($names);
```

The sorting functions perform three kinds of sorts. First, you can sort an array by value, discarding the original keys, using **sort( )**. It's important to understand that the array's keys will be reset after the sorting process, so *if the key-value relationship is important, you should not use* **sort( )**.

Second, you can sort an array by value *while maintaining the keys*, using **asort( )**. Third, you can sort an array by key, using **ksort( )**. Each of these can sort in reverse order if you change them to **rsort( )**, **arsort( )**, and **krsort( )** respectively.

To demonstrate the effect sorting arrays will have, this next script will create an array of movie titles and ratings (how much I liked them on a scale of 1 to 10) and then display this list in different ways.

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

To sort arrays:

1. Begin a new PHP document in your text editor or IDE, to be named **sorting.php** starting with the initial HTML (**Script 2.8**):

```
<!DOCTYPE html PUBLIC "-//W3C//
→DTD XHTML 1.0 Transitional//EN"
→"http://www.w3.org/TR/xhtml1/DTD/
→xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/
→1999/xhtml" xml:lang="en"
→lang="en">
<head>
  <meta http-equiv="Content-Type"
  →content="text/html;
  →charset=utf-8" />
  <title>Sorting Arrays</title>
</head>
<body>
```

2. Create an HTML table:

```
<table border="0" cellspacing="3"
→cellpadding="3" align="center">
  <tr>
    <td><h2>Rating</h2></td>
    <td><h2>Title</h2></td>
  </tr>
```

To make the ordered list easier to read, it'll be printed within an HTML table. The table is begun here.

3. Add the opening PHP tag and create a new array:

```
<?php # Script 2.8 - sorting.php
$movies = array (
'Casablanca' => 10,
'To Kill a Mockingbird' => 10,
'The English Patient' => 2,
'Stranger Than Fiction' => 9,
'Story of the Weeping Camel' => 5,
'Donnie Darko' => 7
);
```

Script 2.8 An array is defined, then sorted in two different ways: first by key, then by value (in reverse order).

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD
    XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/
    xhtml1-transitional.dtd">
2   <html xmlns="http://www.w3.org/1999/
    xhtml" xml:lang="en" lang="en">
3   <head>
4      <meta http-equiv="Content-Type"
       content="text/html; charset=utf-8" />
5      <title>Sorting Arrays</title>
6   </head>
7   <body>
8   <table border="0" cellspacing="3"
    cellpadding="3" align="center">
9      <tr>
10        <td><h2>Rating</h2></td>
11        <td><h2>Title</h2></td>
12     </tr>
13  <?php # Script 2.8 - sorting.php
14
15  // Create the array:
16  $movies = array (
17  'Casablanca' => 10,
18  'To Kill a Mockingbird' => 10,
19  'The English Patient' => 2,
20  'Stranger Than Fiction' => 9,
21  'Story of the Weeping Camel' => 5,
22  'Donnie Darko' => 7
23  );
24
25  // Display the movies in their
    original order:
26  echo '<tr><td colspan="2"><b>In
    their original order:</b></td></tr>';
27  foreach ($movies as $title =>
    $rating) {
28     echo "<tr><td>$rating</td>
29     <td>$title</td></tr>\n";
30  }
31
32  // Display the movies sorted by title:
33  ksort($movies);
34  echo '<tr><td colspan="2"><b>Sorted by
    title:</b></td></tr>';
```

code continues on next page

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

Script 2.8 *continued*

```
35  foreach ($movies as $title =>
    $rating) {
36     echo "<tr><td>$rating</td>
37     <td>$title</td></tr>\n";
38  }
39
40  // Display the movies sorted by
    rating:
41  arsort($movies);
42  echo '<tr><td colspan="2"><b>Sorted by
    rating:</b></td></tr>';
43  foreach ($movies as $title =>
    $rating) {
44     echo "<tr><td>$rating</td>
45     <td>$title</td></tr>\n";
46  }
47
48  ?>
49  </table>
50  </body>
51  </html>
```

This array uses movie titles as the keys and their respective ratings as their values. This structure will open up several possibilities for sorting the whole list. Feel free to change the movie listings and rankings as you see fit (just don't chastise me for my taste in films).

4. Print out the array as is:

```
echo '<tr><td colspan="2"><b>In
→their original order:</b>
→</td></tr>';
foreach ($movies as $title =>
→$rating) {
  echo "<tr><td>$rating</td>
  <td>$title</td></tr>\n";
}
```

At this point in the script, the array is in the same order as it was defined. To verify this, print it out. A caption is first printed across both table columns. Then, within the **foreach** loop, the key is printed in the first column and the value in the second. A newline is also printed to improve the readability of the HTML source code.

5. Sort the array alphabetically by title and print it again:

```
ksort($movies);
echo '<tr><td colspan="2"><b>
→Sorted by title:</b></td></tr>';
foreach ($movies as $title =>
→$rating) {
  echo "<tr><td>$rating</td>
  <td>$title</td></tr>\n";
}
```

The **ksort( )** function will sort an array by key, in ascending order, while maintaining the key-value relationship. The rest of the code is a repetition of Step 4.

continues on next page

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

6. Sort the array numerically by descending rating and print again:

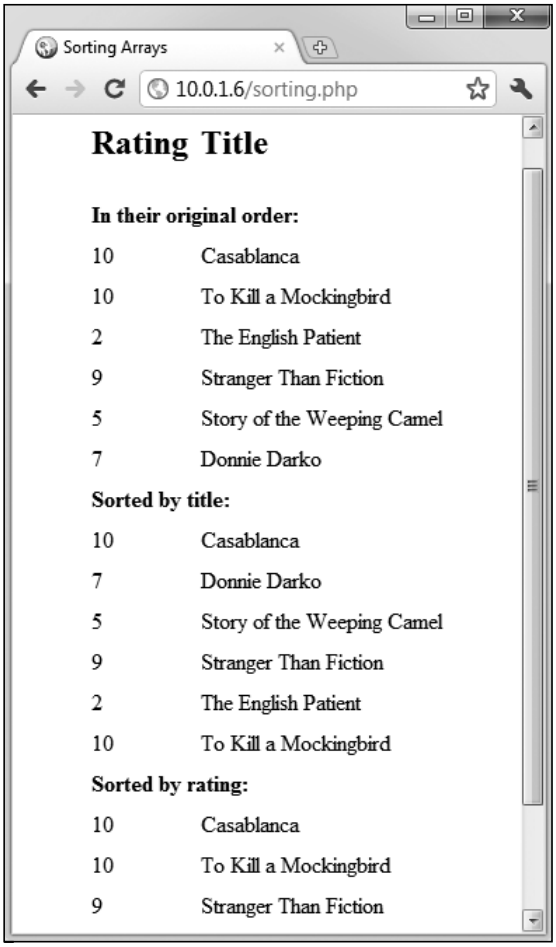
```
arsort($movies);
echo '<tr><td colspan="2"><b>
→Sorted by rating:</b></td></tr>';
foreach ($movies as $title =>
→$rating) {
  echo "<tr><td>$rating</td>
  <td>$title</td></tr>\n";
}
```

To sort by values (the ratings), while maintaining the keys, one would use the `asort( )` function. But since the highest-ranking films should be listed first, the order must be reversed, using `arsort( )`.

7. Complete the PHP, the table, and the HTML:

```
?>
</table>
</body>
</html>
```

8. Save the file as `sorting.php`, place it in your Web directory, and test it in your Web browser ❶.



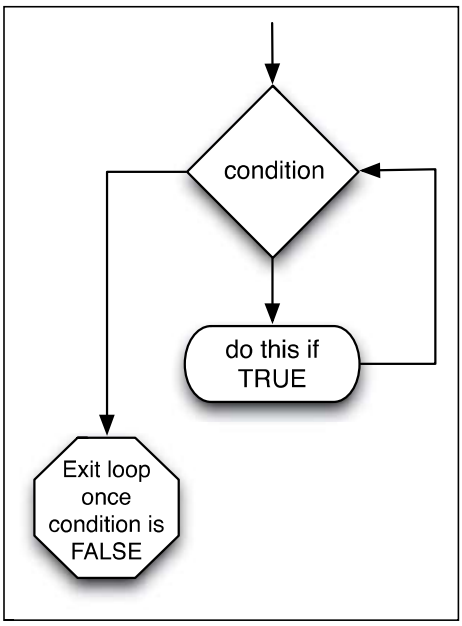
❶ This page demonstrates different ways arrays can be sorted.

TIP To randomize the order of an array, use `shuffle( )`.

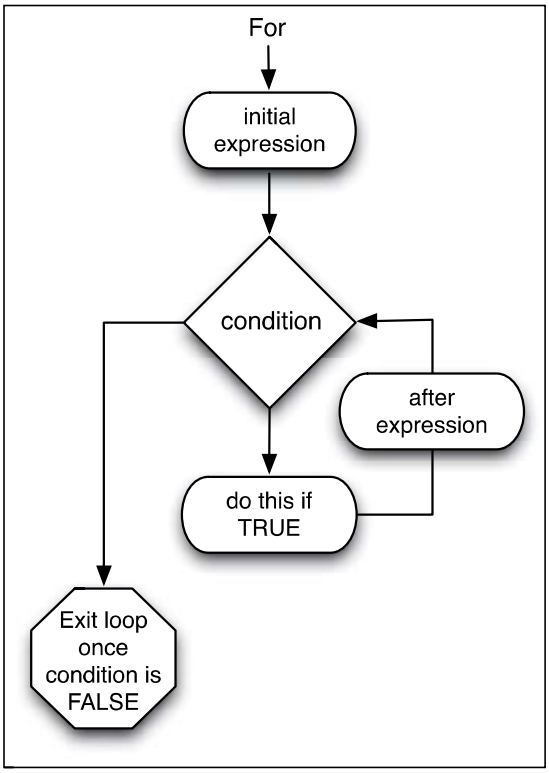
TIP PHP's `natsort( )` function can be used to sort arrays in a more natural order (primarily handling numbers in strings better).

TIP Multidimensional arrays can be sorted in PHP with a little effort. See the PHP manual for more information on the `usort( )` function or check out my *PHP 5 Advanced: Visual QuickPro Guide* book.

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.



Ⓐ A flowchart representation of how PHP handles a **`while`** loop.



Ⓑ A flowchart representation of how PHP handles the more complex **`for`** loop.

For and While Loops

The last language construct to discuss in this chapter is the loop. You've already used one, **`foreach`**, to access every element in an array. The other two types of loops you'll use are **`for`** and **`while`**.

The **`while`** loop looks like this:

```
while (condition) {
        // Do something.
}
```

As long as the *condition* part of the loop is true, the loop will be executed. Once it becomes false, the loop is stopped Ⓐ. If the condition is never true, the loop will never be executed. The **`while`** loop will most frequently be used when retrieving results from a database, as you'll see in Chapter 9, "Using PHP with MySQL."

The **`for`** loop has a more complicated syntax:

```
for (initial expression; condition;
closing expression) {
        // Do something.
}
```

Upon first executing the loop, the initial expression is run. Then the condition is checked and, if true, the contents of the loop are executed. After execution, the closing expression is run and the condition is checked again. This process continues until the condition is false Ⓑ. As an example,

```
for ($i = 1; $i <= 10; $i++) {
        echo $i;
}
```

The first time this loop is run, the **`$i`** variable is set to the value of 1. Then the condition is checked (*is 1 less than or equal to 10?*). Since this is true, 1 is printed out (**`echo $i`**). Then, **`$i`** is incremented to 2 (**`$i++`**), the condition is checked, and so forth. The result of this script will be the numbers 1 through 10 printed out.

continues on next page

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

The functionality of both loops is similar enough that **for** and **while** can often be used interchangeably. Still, experience will reveal that the **for** loop is a better choice for doing something *a known number of times*, whereas **while** is used when a condition will be true an *unknown number of times*.

In this chapter's last example, the calendar script created earlier will be rewritten using **for** loops in place of two of the **foreach** loops.

To use loops:

1. Open **calendar.php** (refer to Script 2.6) in your text editor or IDE.
2. Delete the creation of the **$days** and **$years** arrays (lines 18–19).

   Using loops, the same result of the two pull-down menus can be achieved without the extra code and memory overhead involved with creating actual arrays. So these two arrays should be deleted, while still keeping the **$months** array.
3. Rewrite the **$days foreach** loop as a **for** loop (**Script 2.9**):

```
for ($day = 1; $day <= 31;
→ $day++) {
  echo "<option value=\"$day\">$day
  → </option>\n";
}
```

   This standard **for** loop begins by initializing the **$day** variable as 1. It will continue the loop until **$day** is greater than 31, and upon each iteration, **$day** will be incremented by 1. The content of the loop itself (which is executed 31 times) is an **echo** statement.

Script 2.9 Loops are often used in conjunction with or in lieu of an array. Here, two **for** loops replace the arrays and **foreach** loops used in the script previously.

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD
    XHTML 1.0 Transitional//EN"
          "http://www.w3.org/TR/xhtml1/DTD/
          xhtml1-transitional.dtd">
2   <html xmlns="http://www.w3.org/1999/
    xhtml" xml:lang="en" lang="en">
3   <head>
4      <meta http-equiv="Content-Type"
       content="text/html; charset=utf-8" />
5      <title>Calendar</title>
6   </head>
7   <body>
8   <form action="calendar.php"
    method="post">
9   <?php # Script 2.9 - calendar.php #2
10
11  // This script makes three pull-down
    menus
12  // for an HTML form: months, days, years.
13
14  // Make the months array:
15  $months = array (1 => 'January',
    'February', 'March', 'April', 'May',
    'June', 'July', 'August', 'September',
    'October', 'November', 'December');
16
17  // Make the months pull-down menu:
18  echo '<select name="month">';
19  foreach ($months as $key => $value) {
20     echo "<option value=\"$key\">$value
       </option>\n";
21  }
22  echo '</select>';
23
24  // Make the days pull-down menu:
25  echo '<select name="day">';
26  for ($day = 1; $day <= 31; $day++) {
27     echo "<option value=\"$day\">
       $day</option>\n";
28  }
29  echo '</select>';
30
```

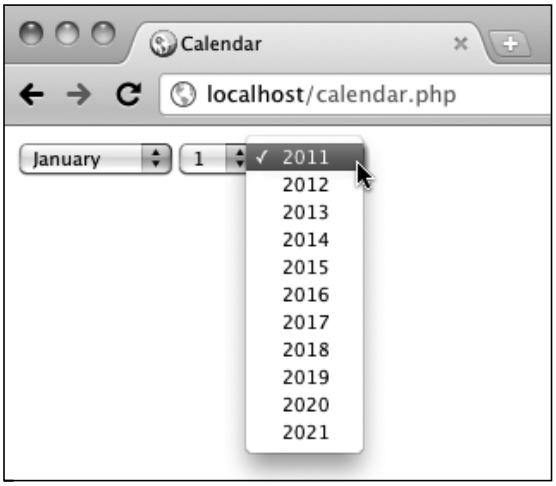
code continues on next page

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

Script 2.9 *continued*

```
// Make the years pull-down menu:
echo '<select name="year">';
for ($year = 2011; $year <= 2021;
$year++) {
   echo "<option value=\"$year\">
   $year</option>\n";
}
echo '</select>';

?>
</form>
</body>
</html>
```



C This calendar form looks the same as it had previously but was created with two fewer arrays (compare Script 2.9 with Script 2.6).

4. Rewrite the **$years foreach** loop as a **for** loop:

```
for ($year = 2011; $year <= 2021;
→ $year++) {
  echo "<option value=\"$year\">
  → $year</option>\n";
}
```

The structure of this loop is fundamentally the same as the **$day for** loop, but the **$year** variable is initially set to 2011 instead of 1. As long as **$year** is less than or equal to 2021, the loop will be executed. Within the loop, the **echo** statement is run.

5. Save the file, place it in your Web directory, and test it in your Web browser **C**.

TIP **PHP also has a do…while loop with a slightly different syntax (check the manual). This loop will always be executed at least once.**

TIP **When using loops, watch your parameters and conditions to avoid the dreaded infinite loop, which occurs when a loop's condition is never going to be false.**

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

Review and Pursue

If you have any problems with the review questions or the pursue prompts, turn to the book's supporting forum (**`www.LarryUllman.com/forums/`**).

Note: Some of these questions and prompts rehash information covered in Chapter 1, in order to reinforce some of the most important points.

Review

- What is the significance of a form's **`method`** attribute? Of its **`action`** attribute?
- Why must an HTML form that gets submitted to a PHP script be loaded through a URL? What would happen upon submitting the form if it were not loaded through a URL?
- What are the differences between using single and double quotation marks to delineate strings?
- What control structures were introduced in this chapter?
- What new variable type was introduced in this chapter?
- What operator tests for equality? What is the assignment operator?
- Why are textual form elements validated using **`empty()`** but other form elements are validated using **`isset()`**?
- What is the difference between an *indexed* array and an *associative* array?
- With what value do indexed arrays begin (by default)? If an indexed array has ten elements in it, what would the expected index be of the last element in the array?
- What are the *superglobal arrays*? From where do the following superglobals get their values?
  - **`$_GET`**
  - **`$_POST`**
  - **`$_COOKIE`**
  - **`$_REQUEST`**
  - **`$_SESSION`**
  - **`$_SERVER`**
  - **`$_ENV`**
- How can you print an individual *indexed* array item? How can you print an individual *associative* array item? Note: there is more than one answer to both questions.
- What does the **`count()`** function do?
- What impact does printing **`\n`** have on the Web browser?
- Generally speaking, when would you use a **`while`** loop? When would you use a **`for`** loop? When would you use a **`foreach`** loop? What is the syntax of each loop type?
- What is the **`++`** operator? What does it do?

Licensed by Dena Laney 2335846

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.

Pursue

- What version of PHP are you using? If you don't know, find out now!
- Create a new form that takes some input from the user (perhaps base it on a form you know you'll need for one of your projects). Then create the PHP script that validates the form data and reports upon the results.
- Rewrite the gender conditional in **`handle_form.php`** (Script 2.4) as one conditional instead of two nested ones. Hint: You'll need to use the **`AND`** operator.
- Rewrite **`handle_form.php`** (Script 2.4) to use **`$_POST`** instead of **`$_REQUEST`**.
- Rewrite **`handle_form.php`** (Script 2.4) so that it validates the age element. Hint: Use the **`$gender`** validation as a template, this time checking against the corresponding pull-down option values (*0-29*, *30-60*, *60+*).
- Rewrite the **`echo`** statement in the final version of **`handle_form.php`** (Script 2.5) so that it uses single quotation marks and concatenation instead of double quotation marks.
- Look up in the PHP manual one of the array functions introduced in this book. Then check out some of the other array-related functions built into the language.
- Create a new array and then display its elements. Sort the array in different ways and then display the array's contents again.
- Create a form that contains a select menu or series of check boxes that allow for multiple sections. Then, in the handling PHP script, display the selected items along with a count of how many the user selected.
- For added complexity, take the suggested PHP script you just created (that handles multiple selections), and have it display the selections in alphabetical order.

PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide, Fourth Edition. PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide,
Prepared for denalaney@gmail.com, Dena Laney
Copyright © 2012 by Larry Ullman. This download file is made available for personal use only and is subject to the Terms of Service. Any other use requires prior written consent from the copyright owner. Unauthorized use, reproduction and/or distribution are strictly prohibited and violate applicable laws. All rights reserved.