

LEARNING MADE EASY



Web Coding & Development

ALL-IN-ONE

for
dummies[®]
A Wiley Brand



Paul McFedries

Web Coding & Development

ALL-IN-ONE

for
dummies[®]
A Wiley Brand



Web Coding & Development

ALL-IN-ONE

by Paul McFedries

for
dummies[®]
A Wiley Brand

Web Coding & Development All-in-One For Dummies®

Published by: **John Wiley & Sons, Inc.**, 111 River Street, Hoboken, NJ 07030-5774, www.wiley.com

Copyright © 2018 by John Wiley & Sons, Inc., Hoboken, New Jersey

Published simultaneously in Canada

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without the prior written permission of the Publisher. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, (201) 748-6011, fax (201) 748-6008, or online at <http://www.wiley.com/go/permissions>.

Trademarks: Wiley, For Dummies, the Dummies Man logo, Dummies.com, Making Everything Easier, and related trade dress are trademarks or registered trademarks of John Wiley & Sons, Inc. and may not be used without written permission. All other trademarks are the property of their respective owners. John Wiley & Sons, Inc. is not associated with any product or vendor mentioned in this book.

LIMIT OF LIABILITY/DISCLAIMER OF WARRANTY: THE PUBLISHER AND THE AUTHOR MAKE NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE ACCURACY OR COMPLETENESS OF THE CONTENTS OF THIS WORK AND SPECIFICALLY DISCLAIM ALL WARRANTIES, INCLUDING WITHOUT LIMITATION WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE. NO WARRANTY MAY BE CREATED OR EXTENDED BY SALES OR PROMOTIONAL MATERIALS. THE ADVICE AND STRATEGIES CONTAINED HEREIN MAY NOT BE SUITABLE FOR EVERY SITUATION. THIS WORK IS SOLD WITH THE UNDERSTANDING THAT THE PUBLISHER IS NOT ENGAGED IN RENDERING LEGAL, ACCOUNTING, OR OTHER PROFESSIONAL SERVICES. IF PROFESSIONAL ASSISTANCE IS REQUIRED, THE SERVICES OF A COMPETENT PROFESSIONAL PERSON SHOULD BE SOUGHT. NEITHER THE PUBLISHER NOR THE AUTHOR SHALL BE LIABLE FOR DAMAGES ARISING HEREFROM. THE FACT THAT AN ORGANIZATION OR WEBSITE IS REFERRED TO IN THIS WORK AS A CITATION AND/OR A POTENTIAL SOURCE OF FURTHER INFORMATION DOES NOT MEAN THAT THE AUTHOR OR THE PUBLISHER ENDORSES THE INFORMATION THE ORGANIZATION OR WEBSITE MAY PROVIDE OR RECOMMENDATIONS IT MAY MAKE. FURTHER, READERS SHOULD BE AWARE THAT INTERNET WEBSITES LISTED IN THIS WORK MAY HAVE CHANGED OR DISAPPEARED BETWEEN WHEN THIS WORK WAS WRITTEN AND WHEN IT IS READ.

For general information on our other products and services, please contact our Customer Care Department within the U.S. at 877-762-2974, outside the U.S. at 317-572-3993, or fax 317-572-4002. For technical support, please visit <https://hub.wiley.com/community/support/dummies>.

Wiley publishes in a variety of print and electronic formats and by print-on-demand. Some material included with standard print versions of this book may not be included in e-books or in print-on-demand. If this book refers to media such as a CD or DVD that is not included in the version you purchased, you may download this material at <http://booksupport.wiley.com>. For more information about Wiley products, visit www.wiley.com.

Library of Congress Control Number: 2018935649

ISBN: 978-1-119-47392-3; ISBN: 978-1-119-47383-1 (ePDF); ISBN: 978-1-119-47379-4 (ePub)

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

Contents at a Glance

Introduction	1
Book 1: Getting Ready to Code for the Web	5
CHAPTER 1: How Web Coding and Development Work	7
CHAPTER 2: Setting Up Your Web Development Home	21
CHAPTER 3: Finding and Setting Up a Web Host	35
Book 2: Coding the Front End, Part 1: HTML & CSS	47
CHAPTER 1: Structuring the Page with HTML	49
CHAPTER 2: Styling the Page with CSS	79
CHAPTER 3: Sizing and Positioning Page Elements	103
CHAPTER 4: Creating the Page Layout	127
Book 3: Coding the Front End, Part 2: JavaScript	167
CHAPTER 1: An Overview of JavaScript	169
CHAPTER 2: Understanding Variables	183
CHAPTER 3: Building Expressions	197
CHAPTER 4: Controlling the Flow of JavaScript	225
CHAPTER 5: Harnessing the Power of Functions	249
CHAPTER 6: Working with Objects	267
CHAPTER 7: Working with Arrays	291
CHAPTER 8: Manipulating Strings, Dates, and Numbers	311
CHAPTER 9: Debugging Your Code	341
Book 4: Coding the Front End, Part 3: jQuery	363
CHAPTER 1: Developing Pages Faster with jQuery	365
CHAPTER 2: Livening Up Your Page with Events and Animation	387
CHAPTER 3: Getting to Know jQuery UI	411
Book 5: Coding the Back End: PHP and MySQL	433
CHAPTER 1: Learning PHP Coding Basics	435
CHAPTER 2: Building and Querying MySQL Databases	467
CHAPTER 3: Using PHP to Access MySQL Data	493

Book 6: Coding Dynamic Web Pages	507
CHAPTER 1: Melding PHP and JavaScript with Ajax and JSON	509
CHAPTER 2: Building and Processing Web Forms.	533
CHAPTER 3: Validating Form Data	565
 Book 7: Coding Web Apps	 591
CHAPTER 1: Planning a Web App.	593
CHAPTER 2: Laying the Foundation.	619
CHAPTER 3: Managing Data	637
CHAPTER 4: Managing App Users	673
 Book 8: Coding Mobile Web Apps	 721
CHAPTER 1: Exploring Mobile-First Web Development	723
CHAPTER 2: Building a Mobile Web App.	739
 Index	 769

Table of Contents

INTRODUCTION	1
About This Book	2
Foolish Assumptions	3
"I've never coded before!"	3
"I have coded before!"	3
Icons Used in This Book	4
Beyond the Book	4
BOOK 1: GETTING READY TO CODE FOR THE WEB	5
CHAPTER 1: How Web Coding and Development Work	7
The Nuts and Bolts of Web Coding and Development	8
How the web works	8
How the web works, take two	11
Understanding the Front End: HTML and CSS	12
Adding structure: HTML	13
Adding style: CSS	14
Understanding the Back End: PHP and MySQL	15
Storing data on the server: MySQL	16
Accessing data on the server: PHP	16
How It All Fits Together: JavaScript and jQuery	16
Front end, meet back end: JavaScript	16
Making your web coding life easier: jQuery	17
How Dynamic Web Pages Work	18
What Is a Web App?	19
What Is a Mobile Web App?	19
What's the Difference between Web Coding and Web Development?	20
CHAPTER 2: Setting Up Your Web Development Home	21
What Is a Local Web Development Environment?	22
Do You Need a Local Web Development Environment?	22
Setting Up the XAMPP for Windows Development Environment	23
Installing XAMPP for Windows	24
Running the XAMPP for Windows Control Panel	26
Accessing your local web server	27
Setting Up the XAMPP for OS X Development Environment	29
Installing XAMPP for OS X	29
Running the XAMPP Application Manager	30
Accessing your local web server	31
Choosing Your Text Editor	33

CHAPTER 3: Finding and Setting Up a Web Host	35
Understanding Web Hosting Providers	36
Using your existing Internet provider	36
Finding a free hosting provider	37
Signing up with a commercial hosting provider	37
A Buyer's Guide to Web Hosting	37
Finding a Web Host	40
Finding Your Way around Your New Web Home	41
Your directory and your web address	42
Making your hard disk mirror your web home	42
Uploading your site files	44
Making changes to your web files	45

BOOK 2: CODING THE FRONT END, PART 1: HTML & CSS 47

CHAPTER 1: Structuring the Page with HTML	49
Getting the Hang of HTML	50
Understanding Tag Attributes	52
Learning the Fundamental Structure of an HTML5 Web Page	53
Giving your page a title	54
Adding some text	56
Some Notes on Structure versus Style	57
Applying the Basic Text Tags	58
Emphasizing text	58
Marking important text	59
Nesting tags	60
Adding headings	60
Adding quotations	61
Creating Links	62
Linking basics	62
Anchors aweigh: Internal links	63
Building Bulleted and Numbered Lists	65
Making your point with bulleted lists	65
Numbered lists: Easy as one, two, three	67
Inserting Special Characters	68
Inserting Images	69
Carving Up the Page	71
The <header> tag	71
The <nav> tag	72
The <main> tag	73
The <article> tag	74
The <section> tag	74
The <aside> tag	75

	The <footer> tag	75
	Handling non-semantic content with <div>.....	76
	Handling words and characters with 	77
CHAPTER 2:	Styling the Page with CSS	79
	Figuring Out Cascading Style Sheets	80
	Styles: Bundles of formatting options	80
	Sheets: Collections of styles.....	80
	Cascading: How styles propagate.....	81
	Getting the Hang of CSS Rules and Declarations	81
	Adding Styles to a Page	83
	Inserting inline styles	83
	Embedding an internal style sheet	84
	Linking to an external style sheet	86
	Styling Page Text	87
	Setting the type size	87
	Getting comfy with CSS measurement units.....	88
	Applying a font family.....	89
	Making text bold	91
	Styling text with italics.....	91
	Styling links.....	91
	Aligning paragraph text	92
	Indenting a paragraph's first line	92
	Working with Colors	93
	Specifying a color.....	93
	Coloring text.....	94
	Coloring the background	94
	Getting to Know the Web Page Family.....	95
	Using CSS Selectors.....	96
	The class selector	97
	The id selector	98
	The descendant selector	99
	The child selector.....	99
	Revisiting the Cascade	100
CHAPTER 3:	Sizing and Positioning Page Elements	103
	Learning about the CSS Box Model	104
	Styling Sizes	105
	Adding Padding	107
	Building Borders	109
	Making Margins.....	110
	Resetting the padding and margin.....	111
	Collapsing margins ahead!.....	111
	Getting a Grip on Page Flow	113

Floating Elements	115
Clearing your floats	116
Collapsing containers ahead!	117
Positioning Elements	120
Using relative positioning	121
Giving absolute positioning a whirl	122
Trying out fixed positioning	125
CHAPTER 4: Creating the Page Layout	127
What Is Page Layout?	128
Laying Out Page Elements with Floats	128
Laying Out Page Elements with Inline Blocks	132
Making Flexible Layouts with Flexbox	136
Setting up the flex container	137
Aligning flex items along the primary axis	139
Aligning flex items along the secondary axis	140
Centering an element horizontally and vertically	141
Laying out a navigation bar with flexbox	143
Allowing flex items to grow	144
Allowing flex items to shrink	146
Laying out content columns with flexbox	149
Flexbox browser support	152
Shaping the Overall Page Layout with CSS Grid	153
Setting up the grid container	154
Specifying the grid rows and columns	154
Creating grid gaps	155
Assigning grid items to rows and columns	157
Aligning grid items	160
Laying out content columns with Grid	161
Grid browser support	163
Providing Fallbacks for Page Layouts	164

BOOK 3: CODING THE FRONT END, PART 2: JAVASCRIPT

167

CHAPTER 1: An Overview of JavaScript	169
JavaScript: Controlling the Machine	170
What Is a Programming Language?	171
Is JavaScript Hard to Learn?	172
What Can You Do with JavaScript?	173
What Can't You Do with JavaScript?	174
What Do You Need to Get Started?	175
Basic Script Construction	175
The <script> tag	175
Handling browsers with JavaScript turned off	176

Where do you put the <script> tag?	176
Example #1: Displaying a message to the user.	177
Example #2: Writing text to the page.	179
Adding Comments to Your Code.	180
Creating External JavaScript Files	181
CHAPTER 2: Understanding Variables	183
What Is a Variable?	184
Declaring a variable.	184
Storing a value in a variable.	185
Using variables in statements	186
Naming Variables: Rules and Best Practices	187
Rules for naming variables.	187
Ideas for good variable names	188
Understanding Literal Data Types	189
Working with numeric literals	189
Working with string literals	191
Working with Boolean literals	193
JavaScript Reserved Words	193
JavaScript Keywords	194
CHAPTER 3: Building Expressions	197
Understanding Expression Structure	197
Building Numeric Expressions.	199
A quick look at the arithmetic operators	199
Using the addition (+) operator	200
Using the increment (++) operator	200
Using the subtraction and negation (-) operators	201
Using the decrement (--) operator	202
Using the multiplication (*) operator	202
Using the division (/) operator	202
Using the modulus (%) operator	204
Using the arithmetic assignment operators	204
Building String Expressions	205
Building Comparison Expressions	208
The comparison operators.	208
Using the equal (==) operator	208
Using the not equal (!=) operator	209
Using the greater than (>) operator	209
Using the less than (<) operator	209
Using the greater than or equal (>=) operator	210
Using the less than or equal (<=) operator	210
The comparison operators and data conversion	211
Using the identity (===) operator	212
Using the non-identity (!==) operator.	212

Using strings in comparison expressions	213
Using the ternary (?:) operator	214
Building Logical Expressions	215
The logical operators	215
Using the AND (&&) operator	215
Using the OR () operator	216
Using the NOT (!) Operator	217
Advanced notes on the && and operators	217
Understanding Operator Precedence	219
The order of precedence	220
Controlling the order of precedence	221
CHAPTER 4: Controlling the Flow of JavaScript	225
Understanding JavaScript's Control Structures	226
Making True/False Decisions with if() Statements	226
Branching with if(). . .else Statements	228
Making Multiple Decisions	229
Using the AND (??) and OR () operators	230
Nesting multiple if() statements	230
Using the switch() statement	231
Understanding Code Looping	234
Using while() Loops	235
Using for() Loops	237
Using do. . .while() Loops	241
Controlling Loop Execution	243
Exiting a loop using the break statement	243
Bypassing loop statements using the continue statement	245
Avoiding Infinite Loops	246
CHAPTER 5: Harnessing the Power of Functions	249
What Is a Function?	250
The Structure of a Function	250
Where Do You Put a Function?	251
Calling a Function	252
Calling a function when the <script> tag is parsed	252
Calling a function after the page is loaded	253
Calling a function in response to an event	254
Passing Values to Functions	255
Passing a single value to a function	256
Passing multiple values to a function	257
Returning a Value from a Function	258
Understanding Local versus Global Variables	259
Working with local scope	260
Working with global scope	261
Using Recursive Functions	262

CHAPTER 6:	Working with Objects	267
	What Is an Object?	267
	The JavaScript Object Hierarchy	269
	Manipulating Object Properties	271
	Referencing a property	271
	Some objects are properties	272
	Changing the value of a property	273
	Working with Object Methods	273
	Playing Around with the window Object	275
	Referencing the window object	275
	Some window object properties you should know	275
	Working with JavaScript timeouts and intervals	276
	Interacting with the user	280
	Programming the document Object	284
	Specifying an element	284
	Working with elements	287
CHAPTER 7:	Working with Arrays	291
	What Is an Array?	291
	Declaring an Array	293
	Populating an Array with Data	294
	Declaring and populating an array at the same time	295
	Using a loop to populate an array	296
	Using a loop to work with array data	297
	Creating Multidimensional Arrays	299
	Using the Array Object	300
	The length property	300
	Concatenating to create a new array: concat()	301
	Creating a string from an array's elements: join()	302
	Removing an array's last element: pop()	303
	Adding elements to the end of an array: push()	303
	Reversing the order of an array's elements: reverse()	304
	Removing an array's first element: shift()	305
	Returning a subset of an array: slice()	305
	Ordering array elements: sort()	306
	Removing, replacing, and inserting elements: splice()	308
	Inserting elements at the beginning of an array: unshift()	310
CHAPTER 8:	Manipulating Strings, Dates, and Numbers	311
	Manipulating Text with the String Object	311
	Determining the length of a string	312
	Finding substrings	313
	Methods that extract substrings	315

Dealing with Dates and Times	323
Arguments used with the Date object	324
Working with the Date object	324
Extracting information about a date	325
Setting the date	330
Performing date calculations	332
Working with Numbers: The Math Object	335
Converting between strings and numbers	336
The Math object's properties and methods	338
CHAPTER 9: Debugging Your Code	341
Understanding JavaScript's Error Types	342
Syntax errors	342
Runtime errors	342
Logic errors	343
Getting to Know Your Debugging Tools	344
Debugging with the Console	345
Displaying the console in various browsers	346
Logging data to the Console	346
Executing code in the Console	347
Pausing Your Code	348
Entering break mode	348
Exiting break mode	350
Stepping through Your Code	350
Stepping into some code	351
Stepping over some code	351
Stepping out of some code	352
Monitoring Script Values	352
Viewing a single variable value	352
Viewing all variable values	353
Adding a watch expression	354
More Debugging Strategies	355
Top Ten Most Common JavaScript Errors	356
Top Ten Most Common JavaScript Error Messages	359
BOOK 4: CODING THE FRONT END, PART 3: jQuery	363
CHAPTER 1: Developing Pages Faster with jQuery	365
Getting Started with jQuery	366
How to include jQuery in your web page	366
Understanding the \$ function	368
Where to put jQuery code	368

Selecting Elements with jQuery	369
Using the basic selectors	370
Working with jQuery sets	371
Manipulating Page Elements with jQuery	373
Adding an element	374
Replacing an element's HTML	375
Replacing an element's text	376
Removing an element	377
Modifying CSS with jQuery	377
Working with CSS properties	378
Manipulating classes	382
Tweaking HTML Attributes with jQuery	385
Reading an attribute value	385
Setting an attribute value	385
Removing an attribute	386
 CHAPTER 2: Livening Up Your Page with Events and Animation	387
Building Reactive Pages with Events	388
What's an event?	388
Understanding the event types	389
Setting up an event handler	390
Using jQuery's shortcut event handlers	391
Getting data about the event	393
Preventing the default event action	394
Getting your head around event delegation	396
Turning off an event handler	398
Building Lively Pages with Animation	398
Hiding and showing elements	399
Fading elements out and in	400
Sliding elements	401
Controlling the animation duration and pace	402
Example: Creating a web page accordion	403
Animating CSS properties	406
Running code when an animation ends	408
 CHAPTER 3: Getting to Know jQuery UI	411
What's the Deal with jQuery UI?	412
Getting Started with jQuery UI	413
Working with the jQuery UI Widgets	415
Dividing content into tabs	415
Creating a navigation menu	418
Displaying a message in a dialog	420
Hiding and showing content with an accordion	422

Introducing jQuery UI Effects.	424
Applying an effect	424
Checking out the effects	426
Taking a Look at jQuery UI Interactions	428
Applying an interaction.	428
Trying out the interactions.	429

BOOK 5: CODING THE BACK END: PHP AND MYSQL

CHAPTER 1: Learning PHP Coding Basics	435
Understanding How PHP Scripts Work	436
Learning the Basic Syntax of PHP Scripts	436
Declaring PHP Variables	438
Building PHP Expressions.	438
Outputting Text and Tags.	439
Adding line breaks.	440
Mixing and escaping quotation marks	441
Outputting variables in strings	442
Outputting long strings.	443
Outputting really long strings	444
Working with PHP Arrays	445
Declaring arrays.	445
Giving associative arrays a look.	446
Outputting array values	447
Sorting arrays.	448
Looping through array values	450
Creating multidimensional arrays.	450
Controlling the Flow of Your PHP Code	451
Making decisions with if().	452
Making decisions with switch()	453
Looping with while()	454
Looping with for()	455
Looping with do. .while().	456
Working with PHP Functions	456
Passing values to functions	457
Returning a value from a function	458
Working with PHP Objects	458
Rolling your own objects	458
Creating an object	461
Working with object properties.	461
Working with object methods	462

Debugging PHP	463
Configuring php.ini for debugging	463
Accessing the PHP error log	464
Debugging with echo statements	465
Debugging with var_dump() statements	466
CHAPTER 2: Building and Querying MySQL Databases.....	467
What Is MySQL?	468
Tables: Containers for your data	468
Queries: Asking questions of your data	469
Introducing phpMyAdmin	470
Importing data into MySQL	471
Backing up MySQL data	473
Creating a MySQL Database and Its Tables	473
Creating a MySQL database	473
Designing your table	474
Creating a MySQL table	477
Adding data to a table	479
Creating a primary key	479
Querying MySQL Data	480
What Is SQL?	480
Creating a SELECT query	481
Understanding query criteria	482
Querying multiple tables	485
Adding table data with an INSERT query	490
Modifying table data with an UPDATE query	491
Removing table data with a DELETE query	492
CHAPTER 3: Using PHP to Access MySQL Data.....	493
Understanding the Role of PHP and MySQL in Your Web App	494
Using PHP to Access MySQL Data	495
Parsing the query string	495
Connecting to the MySQL database	497
Creating and running the SELECT query	499
Storing the query results in an array	500
Looping through the query results	501
Incorporating query string values in the query	501
Creating and Running Insert, Update, and Delete Queries	504
Separating Your MySQL Login Credentials	505

BOOK 6: CODING DYNAMIC WEB PAGES507

CHAPTER 1:	Melding PHP and JavaScript with Ajax and JSON	509
	What Is Ajax?	510
	Making Ajax Calls with jQuery	511
	Learning more about GET and POST requests	511
	Handling POST requests in PHP	513
	Using .load() to update an element with server data	514
	Using .get() or .post() to communicate with the server	523
	Introducing JSON	526
	Learning the JSON syntax	526
	Declaring and using JSON variables	527
	Returning Ajax Data as JSON Text	528
	Converting server data to the JSON format	528
	Handling JSON data returned by the server	530
CHAPTER 2:	Building and Processing Web Forms	533
	What Is a Web Form?	534
	Understanding How Web Forms Work	535
	Building an HTML5 Web Form	536
	Setting up the form	536
	Adding a form button	537
	Working with text fields	538
	Coding checkboxes	543
	Working with radio buttons	548
	Adding selection lists	551
	Programming pickers	555
	Handling and Triggering Form Events	557
	Setting the focus	558
	Monitoring the focus event	559
	Blurring an element	559
	Monitoring the blur event	560
	Listening for element changes	560
	Submitting the Form	561
	Triggering the submit event	562
	Preventing the default form submission	562
	Preparing the data for submission	563
	Submitting the form data	563
CHAPTER 3:	Validating Form Data	565
	Validating Form Data in the Browser	566
	Making a form field mandatory	566
	Restricting the length of a text field	567

Setting maximum and minimum values on a numeric field	568
Validating email fields	569
Making field values conform to a pattern	570
Styling invalid fields	571
Validating Form Data on the Server	574
Checking for required fields	575
Validating text data	578
Validating a field based on the data type	580
Validating against a pattern	582
Regular Expressions Reference	582
BOOK 7: CODING WEB APPS	591
CHAPTER 1: Planning a Web App	593
What Is a Web App?	594
Planning Your Web App: The Basics	595
What is my app's functionality?	595
What are my app's data requirements?	596
How will my app work?	597
How many pages will my app require?	597
What will my app's pages look like?	598
Planning Your Web App: Responsiveness	599
Planning Your Web App: Accessibility	605
Planning Your Web App: Security	608
Understanding the dangers	609
Defending your web app	612
CHAPTER 2: Laying the Foundation	619
Setting Up the Directory Structure	620
Setting up the public subdirectory	621
Setting up the private subdirectory	623
Creating the Database and Tables	624
Getting Some Back-End Code Ready	626
Defining PHP constants	626
Understanding PHP sessions	627
Securing a PHP session	628
Including code from another PHP file	629
Creating the App Startup Files	630
Creating the back-end initialization file	631
Creating the front-end common files	633
Building the app home page	635

CHAPTER 3: Managing Data	637
Handling Data the CRUD Way	638
Starting the web app's data class	639
Creating a data handler script	640
Creating New Data	643
Building the form	643
Sending the form data to the server	648
Adding the data item	649
Reading and Displaying Data	652
Getting the home page ready for data	652
Making an Ajax request for the data	654
Reading the data	655
Displaying the data	656
Filtering the data	657
Updating and Editing Data	661
Deleting Data	668
CHAPTER 4: Managing App Users	673
Configuring the Home Page	674
Setting Up the Back End to Handle Users	677
Starting the web app's user class	678
Creating a user handler script	679
Signing Up a New User	682
Building the form	683
Sending the data to the server	685
Sending a verification email	688
Adding the user to the database	689
Verifying the user	690
Signing a User In and Out	696
Checking for a signed-in user	696
Adding the form	697
Checking the user's credentials	700
Signing out a user	704
Resetting a Forgotten Password	704
Deleting a User	714
BOOK 8: CODING MOBILE WEB APPS	721
CHAPTER 1: Exploring Mobile-First Web Development	723
What Is Mobile-First Web Development?	724
Learning the Principles of Mobile-First Development	725
Mobile first means content first	725
Pick a testing width that makes sense for your site	726
Get your content to scale with the device	726

Build your CSS the mobile-first way	727
Pick a “non-mobile” breakpoint that makes sense for your content.	727
Going Mobile Faster with jQuery Mobile	729
What is jQuery Mobile?	729
Adding jQuery Mobile to your web app	730
Working with Images in a Mobile App	731
Making images responsive.	731
Delivering images responsively.	732
Storing User Data in the Browser	734
Understanding web storage	735
Adding data to storage	735
Getting data from web storage	736
Removing data from web storage.	737
CHAPTER 2: Building a Mobile Web App	739
Building the Button Builder App	740
Getting Some Help from the Web.	741
Building the App: HTML	741
Setting up the home page skeleton	741
Configuring the header.	744
Creating the app menu.	745
Adding the app’s controls.	745
Building the App: CSS	754
Building the App: JavaScript and jQuery	757
Setting up the app data structures.	757
Setting the app’s control values	758
Getting the app’s control values	761
Writing the custom CSS code.	763
Running the code.	765
Saving the custom CSS	765
Copying the custom CSS.	766
Resetting the CSS to the default	767
INDEX	769

Introduction

When the web first came to the attention of the world's non-geeks back in the mid-1990s, the vastness and variety of its treasures were a wonder to behold. However, it didn't take long before a few courageous and intrepid souls dug a little deeper into this phenomenon and discovered something truly phenomenal: *They* could make web pages, too!

Why was that so amazing? Well, think back to those old days and think, in particular, of what it meant to create what we now call *content*. Think about television shows, radio programs, magazines, newspapers, books, and the other media of the time. The one thing they all had in common was that their creation was a decidedly *uncommon* thing. It required a *team* of professionals, a *massive* distribution system, and a *lot* of money. In short, it wasn't something that your average Okie from Muskogee would have any hope of duplicating.

The web appeared to change all of that because learning HTML was within the grasp of anybody who could feed himself, it had a built-in massive distribution system (the Internet, natch), and it required little or no money. For the first time in history, content was democratized and was no longer defined as the sole province of governments and mega-corporations.

Then reality set in.

People soon realized that merely building a website wasn't enough to attract "eyeballs," as the marketers say. A site had to have interesting, useful, or fun content, or people would stay away in droves. Not only that, but this good content had to be combined with a solid site design, which meant that web designers needed a thorough knowledge of HTML and CSS.

But, alas, eventually even all of that was not enough. To make their websites dynamic and interesting, to make their sites easy to navigate, and to give their sites those extra bells and whistles that surfers had come to expect, something more than content, HTML, and CSS was needed.

That missing link was *code*.

What we've all learned the hard way over the past few years is that you simply can't put together a world-class website unless you have some coding prowess in your site design toolkit. You need to know how to program your way out of

the basic problems that afflict most sites; how to use scripting to go beyond the inherent limitations of HTML and CSS; and how to use code to send and receive data from a web server. And it isn't enough just to copy the generic scripts that are available on the web and paste them into your pages. First of all, most of those scripts are very poorly written, and second of all, they invariably need some customization to work properly on your site.

About This Book

My goal in this book is to give you a complete education on web coding and development. You learn how to set up the tools you need, how to use HTML and CSS to design and build your site, how to use JavaScript and jQuery to program your pages, and how to use PHP and MySQL to program your web server. My aim is to show you that these technologies aren't hard to learn, and that even the greenest rookie programmers can learn how to put together web pages that will amaze their family and friends (and themselves).

If you're looking for lots of programming history, computer science theory, and long-winded explanations of concepts, I'm sorry but you won't find it here. My philosophy throughout this book comes from Linus Torvalds, the creator of the Linux operating system: "Talk is cheap. Show me the code." I explain what needs to be explained and then I move on without further ado (or, most of the time, without any ado at all) to examples and scripts that do more to illuminate a concept than any verbose explanations I could muster (and believe me, I can muster verbosity with the best of them).

How you approach this book depends on your current level of web coding expertise (or lack thereof):

- » If you're just starting out, begin at the beginning with Book 1 and work at your own pace sequentially through to Books 2 and 3. This will give you all the knowledge you need to pick and choose what you want to learn throughout the rest of the book.
- » If you know HTML and CSS, you can probably get away with taking a fast look at Book 2, then settle in with Book 3 and beyond.
- » If you've done some JavaScript coding already, I suggest working quickly through the material in Book 3, then dig into Book 4 a little slower if you don't already know jQuery. You'll then be ready to branch out and explore the rest of the book as you see fit.
- » If you're a relatively experienced JavaScript programmer, use Books 3 and 4 as a refresher, then tackle Book 5 to learn how to code the back end. I've got a few tricks in there that you might find interesting. After that, feel free to

consider the rest of the book a kind of coding smorgasbord that you can sample as your web development taste buds dictate.

Foolish Assumptions

This book is not a primer on the Internet or on using the World Wide Web. This is a coding and development book, pure and simple. This means I assume the following:

- » You know how to operate a basic text editor, and how to get around the operating system and file system on your computer.
- » You have an Internet connection.
- » You know how to use your web browser.

Yep, that's it.

"I've never coded before!"

If you've never done a stitch of computer programming before, even if you're not quite sure what programming really is, don't worry about it for a second because I had you in mind when I wrote this book. For too many years programming has been the property of "hackers" and other technowizards. That made some sense because the programming languages they were using — with bizarre names such as C++ and Perl — were exceedingly difficult to learn, and even harder to master.

This book's main coding technologies — HTML, CSS, JavaScript, jQuery, PHP, and MySQL — are different. They're nowhere near as hard to learn as those for-nerds-only languages. I honestly believe that *anyone* can become a savvy and successful web coder, and this book is, I hope, the proof of that assertion. Just follow along, examine my code carefully (particularly in the first few chapters), and practice what you learn, and you *will* master web coding and development.

"I have coded before!"

What if you've done some programming in the past? For example, you might have dipped a toe or two in the JavaScript waters already, or you might have dabbled with HTML and CSS. Will this book be too basic for you? No, not at all. My other main goal in this book is to provide you with a ton of truly *useful* examples that you can customize and incorporate into your own site. The book's first few chapters start slowly to avoid scaring off those new to this programming business. But

once you get past the basics, I introduce you to lots of great techniques and tricks that will take your web coding skills to a higher level.

Icons Used in This Book



REMEMBER

This icon points out juicy tidbits that are likely to be repeatedly useful to you — so please don't forget them.



TIP

Think of these icons as the fodder of advice columns. They offer (hopefully) wise advice or a bit more information about a topic under discussion.



WARNING

Look out! In this book, you see this icon when I'm trying to help you avoid mistakes that can cost you time, money, or embarrassment.



TECHNICAL
STUFF

When you see this icon, you've come across material that isn't critical to understand but will satisfy the curious. Think "inquiring minds want to know" when you see this icon.

Beyond the Book

Some extra content for this book is available on the web. Go online to find the following:

» **The examples used in the book:** You can find these here:

```
mcfedries.com/webcodingfordummies
```

The examples are organized by book and then by chapter within each book. For each example, you can view the code, copy it to your computer's clipboard, and run the code in the browser.

» **The WebDev Workshop:** To edit the book's examples and try your own code and see instant results, fire up the following site:

```
webdev.mcfedries.com
```

You won't break anything, so feel free to use the site run some experiments and play around with HTML, CSS, JavaScript, and jQuery.

1

Getting Ready to Code for the Web

Contents at a Glance

CHAPTER 1: How Web Coding and Development Work	7
CHAPTER 2: Setting Up Your Web Development Home	21
CHAPTER 3: Finding and Setting Up a Web Host	35

IN THIS CHAPTER

- » Learning how the web works
- » Understanding the front-end technologies of HTML and CSS
- » Understanding the back-end technologies of MySQL and PHP
- » Figuring out how JavaScript fits into all of this
- » Learning about dynamic web pages, web apps, and mobile web apps

Chapter 1

How Web Coding and Development Work

More than mere consumers of technology, we are makers, adapting technology to our needs and integrating it into our lives.

— DALE DOUGHERTY

The 1950s were a hobbyist's paradise with magazines such as *Mechanix Illustrated* and *Popular Science* showing the do-it-yourselfer how to build a go-kart for the kids and how to soup up a lawnmower with an actual motor! Sixty years later, we're now firmly entrenched in the age of do-it-yourself tech, where folks indulge their inner geek to engage in various forms of digital tinkering and hacking. The personification of this high-tech hobbyist renaissance is the *maker*, a modern artisan who lives to create things, rather than merely consume them. Today's makers exhibit a wide range of talents, but the skill most sought-after not only by would-be makers themselves, but by the people who hire them, is web coding and development.

Have you ever visited a website and thought, "Hey, I can do better than that!"? Have you found yourself growing tired of merely reading text and viewing images

that someone else has put on the web? Is there something creative in you — stories, images, expertise, opinions — that you want to share with the world? If you answered a resounding “Yes!” to any of these questions, then congratulations: You have everything you need to get started with web coding and development. You have, in short, the makings of a maker.

The Nuts and Bolts of Web Coding and Development

If, as the King said very gravely in Lewis Carroll’s *Alice in Wonderland*, it’s best to “begin at the beginning,” then you’ve come to the right place. My goal here is to get you off on the right foot by showing you what web coding and web development are.

How the web works

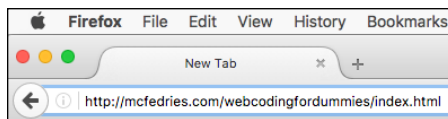
Before you can understand web coding and development, you need to take a step back and understand a bit about how the web itself works. In particular, you need to know what happens behind the scenes when you click a link or type a web page address into your browser. Fortunately, you don’t need to be a network engineer to understand this stuff, because I can explain the basics without much in the way of jargon. Here’s a high-level blow-by-blow of what happens:

1. You tell the web browser the web page you want to visit.

You do that either by clicking a link to the page or by typing the location — known as the *uniform resource locator* or *URL* (usually pronounced “you-are-ell,” but also sometimes “earl”) — into the browser’s address bar (see Figure 1-1).

FIGURE 1-1:

One way to get to a web page is to type the URL in the browser’s address bar.



2. The browser decodes the URL.

Decoding the URL means two things: First, it checks the prefix of the URL to see what type of resource you’re requesting; this is usually `http://` or `https://`, both of which indicate that the resource is a web page. Second, it gets the

URL's domain name — the something.com or whatever.org part — and asks the *domain name system* (DNS) to translate this into a unique location — called the IP (Internet Protocol) address — for the web server that hosts the page (see Figure 1-2).

FIGURE 1-2:
The browser extracts the prefix, domain, and the server address from the URL.

```
Decoding http://mcfedries.com/webcodingfordummies/index.html...

Results:

Prefix: http://
Domain name: mcfedries.com
Web server IP address: 162.144.120.37
```

3. The browser contacts the web server and requests the web page.

With the web server's unique IP address in hand, the web browser sets up a communications channel with the server and then uses that channel to send along a request for the web page (see Figure 1-3).

FIGURE 1-3:
The browser asks the web server for the web page.

```
Dear 162.144.120.37:

At your earliest convenience, please send
me the mcfedries.com web page located at
webcodingfordummies/index.html.

Sincerely,
W. Browser
```

4. The web server decodes the page request.

Decoding the page request involves a number of steps. First, if the web server is shared between multiple user accounts, the server begins by locating the user account that owns the requested page. The server then uses the page address to find the directory that holds the page and the file in which the page code is stored (see Figure 1-4).

FIGURE 1-4:
The server uses the page request to get the account, directory, and filename.

```
Decoding mcfedries.com/webcodingfordummies/index.html...

Results:

User account: paulmcfedries
Directory: webcodingfordummies
Filename: index.html
```

5. The web server sends the web page file to the web browser (see Figure 1-5).

FIGURE 1-5:
The web server sends the requested web page file to the browser.

Dear W. Browser:

Thank you for contacting us. Here is the file you requested. Let us know if you need anything else.

*Best,
mcfedries.com Web Server*

6. The web browser decodes the web page file.

Decoding the page file means looking for text to display, instructions on how to display that text, and other resources required by the page, such as images and fonts (see Figure 1-6).

FIGURE 1-6:
The web browser scours the page file to see if it needs anything else from the server.

Decoding index.html...

Results:

Text: Received
Formatting: Request styles.css
Images: Request logo.png, cover.jpg
Audio: None
Video: None
Data: Request book examples

7. If the web page requires more resources, the web browser asks the server to pass along those resources (see Figure 1-7).

FIGURE 1-7:
The web browser goes back to the server to ask for the other data needed to display the web page.

Dear 162.144.120.37:

Thank you for the page file. If it's not too much trouble, could you please also send along the following:

styles.css
logo.png
cover.jpg
Book examples from the database

8. For each of the requested resources, the web server locates the associated file and sends it to the browser (see Figure 1-8).

FIGURE 1-8:
The web server sends the browser the rest of the requested files.

Dear W. Browser:

You're very welcome. We're here to serve! We're gathering your order and will send along the extra data you requested shortly.

Best,
mcfedries.com Web Server

9. The web browser gathers up all the text, images, and other resources and displays the page in all its digital splendor in the browser's content window (see Figure 1-9).

FIGURE 1-9:
At long last, the web browser displays the web page.



How the web works, take two

Another way to look at this process is to think of the web as a giant mall or shopping center, where each website is a storefront in that mall. When you request a web page from a particular site, the browser takes you into that site's store and asks the clerk for the web page. The clerk goes into the back of the store, locates the page, and hands it to the browser. The browser checks the page and asks for any other needed files, which the clerk retrieves from the back. This process is repeated until the browser has everything it needs, and it then puts all the page pieces together for you, right there in the front of the store.

This metaphor might seem a bit silly, but it serves to introduce yet another metaphor, which itself illustrates one of the most important concepts in web development. In the same way that our website store has a front and a back, so, too, is web development separated into a front end and a back end:

- » **Front end:** That part of the web page that the web browser displays in the browser window. That is, it's the page stuff you see and interact with.
- » **Back end:** That part of the web page that resides on the web server. That is, it's the page stuff that the server gathers based on the requests it receives from the browser.

As a consumer of web pages, you only ever deal with the front end, and even then you only passively engage with the page by reading its content, looking at its images, or clicking its links or buttons.

However, as a maker of web pages — that is, as a web developer — your job entails dealing with both the front end and the back end. Moreover, that job includes coding what others see on the front end, coding how the server gathers its data on the back end, and coding the intermediate tasks that tie the two together.

Understanding the Front End: HTML and CSS

As I mention in the previous section, the *front end* of the web development process involves what users see and interact with in the web browser window. It's the job of the web developer to take a page design — which you might come up with yourself, but is more often something cooked up by a creative type who specializes in web design — and make it web-ready. Getting a design ready for the web means translating the design into the code required for the browser to display the page somewhat faithfully. (I added the hedge word “somewhat” there because it's not always easy to take a design that looks great in Photoshop or Illustrator and make it look just as good on the web. However, with the techniques you learn in this book, you'll almost always be able to come pretty close.)

You need code to create the front end of a web page because without it your page will be quite dull. For example, consider the following text:

```
COPENHAGEN—Researchers from Aalborg University announced today
that they have finally discovered the long sought-after
Soup-Nuts Continuum. Scientists around the world have been
```

searching for this elusive item ever since Albert Einstein's mother-in-law proposed its existence in 1922.

"Today is an incredible day for the physics community and for humanity as a whole," said senior researcher Lars Grüntwerk. "Today, for the first time in history, we are on the verge of knowing everything from soup to, well, you know, nuts."

If you plopped that text onto the web, you get the result shown in Figure 1-10. As you can see, the text is very plain, and the browser didn't even bother to include the paragraph break.

FIGURE 1-10:
Text-only
web pages are
dishwater-dull.

COPENHAGEN—Researchers from Aalborg University announced today that they have finally discovered the long sought-after Soup-Nuts Continuum. Scientists around the world have been searching for this elusive item ever since Albert Einstein's mother-in-law proposed its existence in 1922. "Today is an incredible day for the physics community and for humanity as a whole," said senior researcher Lars Grüntwerk. "Today, for the first time in history, we are on the verge of knowing everything from soup to, well, you know, nuts."

So, if you can't just throw naked text onto the web, what's a would-be web developer to do? Ah, that's where you start earning your web scout merit badges by adding code that tells the browser how you want the text displayed. That code comes in two flavors: structure and formatting.

Adding structure: HTML

The first thing you usually do to code a web page is give it some structure. This means breaking up the text into paragraphs, adding special sections such as a header and footer, organizing text into bulleted or numbered lists, dividing the page into columns, and much more. The web coding technology that governs these and other web page structures is called (deep breath) *Hypertext Markup Language*, or *HTML*, for short.

HTML consists of a few dozen special symbols called *tags* that you sprinkle strategically throughout the page. For example, if you want to tell the web browser that a particular chunk of text is a separate paragraph, you place the `<p>` tag (the *p* here is short for paragraph) before the text and the `</p>` tag after the text.

In the code that follows, I've added these paragraph tags to the plain text that I show earlier. As you can see in Figure 1-11, the web browser displays the text as two separate paragraphs, no questions asked.

```
<p>
COPENHAGEN—Researchers from Aalborg University announced today
that they have finally discovered the long sought-after
```

```

    Soup-Nuts Continuum. Scientists around the world have been
    searching for this elusive item ever since Albert Einstein's
    mother-in-law proposed its existence in 1922.
  </p>
  <p>
    "Today is an incredible day for the physics community and for
    humanity as a whole," said senior researcher Lars Grüntwerk.
    "Today, for the first time in history, we are on the verge of
    knowing everything from soup to, well, you know, nuts."
  </p>

```

FIGURE 1-11:
Adding paragraph
tags to the text
separates the
text into two
paragraphs.

COPENHAGEN—Researchers from Aalborg University announced today that they have finally discovered the long sought-after Soup-Nuts Continuum. Scientists around the world have been searching for this elusive item ever since Albert Einstein's mother-in-law proposed its existence in 1922.

"Today is an incredible day for the physics community and for humanity as a whole," said senior researcher Lars Grüntwerk. "Today, for the first time in history, we are on the verge of knowing everything from soup to, well, you know, nuts."



REMEMBER

HTML is one of the fundamental topics of web development, and you learn all about it in Book 2, Chapter 1.

Adding style: CSS

HTML takes care of the structure of the page, but if you want to change the formatting of the page, then you need to turn to a second front-end technology: *cascading style sheets*, known almost universally as just CSS. With CSS in hand, you can play around with the page colors and fonts, you can add margins and borders around things, and you can mess with the position and dimensions of page elements.

CSS consists of a large number of *properties* that enable you to customize many aspects of the page to make it look the way you want. For example, the `width` property lets you specify how wide a page element should be; the `font-family` property enables you to specify a typeface for an element; and the `font-size` property lets you dictate the type size of an element. Here's some CSS code that applies all three of these properties to every `p` element (that is, every `<p>` tag) that appears in a page (note that `px` is short for pixels):

```

p {
  width: 700px;
  font-family: sans-serif;
  font-size: 24px;
}

```

When used with the sample text from the previous two sections, you get the much nicer-looking text shown in Figure 1-12.

FIGURE 1-12:
With the judicious use of a few CSS properties, you can greatly improve the look of a page.

COPENHAGEN—Researchers from Aalborg University announced today that they have finally discovered the long sought-after Soup-Nuts Continuum. Scientists around the world have been searching for this elusive item ever since Albert Einstein's mother-in-law proposed its existence in 1922.

"Today is an incredible day for the physics community and for humanity as a whole," said senior researcher Lars Grüntwerk. "Today, for the first time in history, we are on the verge of knowing everything from soup to, well, you know, nuts."



REMEMBER

CSS is a cornerstone of web development. You learn much more about it in Book 2, Chapters 2, 3, and 4.

Understanding the Back End: PHP and MySQL

Many web pages are all about the front end. That is, they consist of nothing but text that has been structured by HTML tags and styled by CSS properties, plus a few extra files such as images and fonts. Sure, all these files are transferred from the web server to the browser, but that's the extent of the back end's involvement.

These simple pages are ideal when you have content that doesn't change very often, if ever. With these so-called *static* pages, you plop in your text, add some HTML and CSS, perhaps point to an image or two, and you're done.

But there's another class of page that has content that changes frequently. It could be posts added once or twice a day, or sports or weather updates added once or twice an hour. With these so-called *dynamic* pages, you might have some text, HTML, CSS, and other content that's static, but you almost certainly don't want to be updating the changing content by hand.

Rather than making constant manual changes to such pages, you can convince the back end to do it for you. You do that by taking advantage of two popular back-end technologies: MySQL and PHP.

Storing data on the server: MySQL

MySQL is a relational database management system that runs on the server. You use it to store the data you want to use as the source for some (or perhaps even all) of the data you want to display on your web page. Using a tool called Structured Query Language (SQL, pronounced “ess-kew-ell,” or sometimes “sequel”), you can specify which subset of your data you want to use.



REMEMBER

If phrases such as “relational database management system” and “Structured Query Language” have you furrowing your brow, don’t sweat it: I explain all in Book 5, Chapter 2.

Accessing data on the server: PHP

PHP is a programming language used on the server. It’s a very powerful and full-featured language, but for the purposes of this book, you use PHP mostly to interact with MySQL databases. You can use PHP to extract from MySQL the subset of data you want to display, manipulate that data into a form that’s readable by the front end, and then send the data to the browser.



REMEMBER

You learn about the PHP language in Book 5, Chapter 1, and you learn how to use PHP to access MySQL data in Book 5, Chapter 3.

How It All Fits Together: JavaScript and jQuery

Okay, so now you have a front end consisting of HTML structure and CSS styling, and a back end consisting of MySQL data and PHP code. How do these two seemingly disparate worlds meet to create a full web page experience?

In the website-as-store metaphor that I introduce earlier in this chapter, I use the image of a store clerk taking an order from the web browser and then going into the back of the store to fulfill that order. That clerk is the obvious link between the front end and the back end, so what technology does that clerk represent? She actually represents two technologies that I use in this book: JavaScript and jQuery.

Front end, meet back end: JavaScript

The secret sauce that brings the front end and the back end together to create the vast majority of the web pages you see today, is JavaScript. JavaScript is a

programming language and is the default language used for coding websites today. JavaScript is, first and foremost, a front-end web development language. That is, JavaScript runs inside the web browser and it has access to everything on the page: the text, the images, the HTML tags, the CSS properties, and more. Having access to all the page stuff means that you can use code to manipulate, modify, even add and delete web page elements.

But although JavaScript runs in the browser, it's also capable of reaching out to the server to access back-end stuff. For example, with JavaScript you can send data to the server to store that data in a MySQL database. Similarly, with JavaScript you can request data from the server and then use code to display that data on the web page.



REMEMBER

JavaScript is very powerful, very useful, and very cool, so Book 3 takes nine full chapters to help you learn it well. Also, you learn how JavaScript acts as a bridge between the front end and the back end in Book 6, Chapter 1.

Making your web coding life easier: jQuery

JavaScript is extremely powerful, but sometimes using certain JavaScript statements and structures can be a bit unwieldy. For example, here's a bit of JavaScript code:

```
var subheads = document.getElementsByClassName('subheadings');
```

This will no doubt look like gibberish to you now, but my purpose here is only to have you remark the length of that statement. Now compare the following:

```
var subheads = $('.subheadings');
```

Believe it or not, these statements do exactly the same thing, except the second one is written using a JavaScript package called jQuery. jQuery is a collection — called a *library* — of JavaScript code that makes it easier and faster to code for the web. Not only does jQuery give you shorter ways to reference web page elements, but it also incorporates routines that make it easier for you to manipulate HTML tags and CSS properties, navigate and manipulate web page elements, add animation effects, and much more.



REMEMBER

jQuery is extremely powerful and useful stuff, and you'll be thankful you've got it in your web development toolkit. You learn just enough jQuery to be dangerous in Book 4.

How Dynamic Web Pages Work

It's one thing to know about HTML and CSS and PHP and all the rest, but it's quite another to actually do something useful with these technologies. That, really, is the goal of this book, and to that end the book spends several chapters later covering how to create wonderful things called *dynamic web pages*. A dynamic web page is one that includes content that, rather than being hard-wired into the page, is generated on-the-fly from the web server. This means the page content can change based on a request by the user, by data being added to or modified on the server, or in response to some event, such as the clicking of a button or link.

It likely sounds a bit like voodoo to you now, so perhaps a bit more detail is in order. For example, suppose you want to use a web page to display some data that resides on the server. Here's a general look at the steps involved in that process:

- 1. JavaScript determines the data that it needs from the server.**

JavaScript has various ways it can do this, such as extracting the information from the URL, reading an item the user has selected from a list, or responding to a click from the user.

- 2. JavaScript sends a request for that data to the server.**

In most cases, and certainly in every case you see in this book, JavaScript sends this request by calling a PHP script on the server.

- 3. The PHP script receives the request and passes it along to MySQL.**

The PHP script uses the information obtained from JavaScript to create an SQL command that MySQL can understand.

- 4. MySQL uses the SQL command to extract the required information from the database and then return that data to the PHP script.**

- 5. The PHP script manipulates the returned MySQL data into a form that JavaScript can use.**

JavaScript can't read raw MySQL data, so one of PHP's most important tasks is to convert that data into a format called JavaScript Object Notation (JSON, for short, and pronounced like the name Jason) that JavaScript is on friendly terms with (see Book 6, Chapter 1 for more about this process).

- 6. PHP sends the JSON data back to JavaScript.**

- 7. JavaScript displays the data on the web page.**

One of the joys of JavaScript is that you get tremendous control over how you display the data to the user. Through existing HTML and CSS, and by manipulating these and other web page elements using JavaScript, you can show your data in the best possible light.



REMEMBER

To expand on these steps and learn how to create your own dynamic web pages, check out the three chapters in Book 6.

What Is a Web App?

You no doubt have a bunch of apps residing on your smartphone. If you use Windows 10 on your PC, then you have not only the pre-installed apps such as Mail and Calendar, but you might also have one or more apps downloaded from the Windows Store. If the Mac is more your style, then you're probably quite familiar with apps such as Music and Messages, and you might have installed a few others from the App Store. We live, in other words, in a world full of apps which, in the context of your phone or computer, are software programs dedicated to a single topic or task.

So what then is a *web app*? It's actually something very similar to an app on a device or PC. That is, it's a website, built using web technologies such as HTML, CSS, and JavaScript, that has two main characteristics:

- » The web app is focused on a single topic or task.
- » The web app offers some sort of interface that enables the user to operate the app in one or more ways.

In short, a web app is a website that looks and acts like an app on a device or computer. This is opposed to a regular website, which usually tackles several topics or tasks and has an interface that for the most part only enables users to navigate the site.



REMEMBER

To get the scoop on building your very own web apps, head on over to the four chapters in Book 7.

What Is a Mobile Web App?

In late 2016, the world reached a milestone of sorts when the percentage of people accessing the web via mobile devices such as smartphones and tablets surpassed the percentage of people doing the web thing using desktops and notebooks. The gap between mobile web users and everyone else has only widened since then, so it's safe to say that we live in a mobile web world now.

What does that mean for you as a web developer? It means you can't afford to ignore mobile users when you build your web pages. It means you can't code your web pages using a gigantic desktop monitor and assume that everything will look great on a relatively tiny smartphone screen. It means that you'd do well to embrace the mobile web in a big old bear hug by creating not just web apps, but *mobile web apps*. What's the difference? A mobile web app is the same as a web app — that is, it has content and an interface dedicated to a single topic or task — but with a design built from the ground up to look good and work well in a mobile device. This is known as the *mobile-first* approach to web development, and it's one of the hottest topics in the web coding world.



REMEMBER

To learn how to create your own mobile web apps, look no farther than the two chapters in Book 8.

What's the Difference between Web Coding and Web Development?

After all this talk of HTML, CSS, MySQL, JavaScript, and jQuery, after the bird's-eye view of dynamic sites, web apps, and mobile web apps, you might be wondering when the heck I'm going to answer the most pressing question of the all: What in the name of Sir Tim Berners-Lee (inventor of the web) is the difference between web coding and web development?

I'm glad you asked! Some people would probably answer that question by saying that there's no real difference at all, because "web coding" and "web development" are two ways of referring to the same thing: Creating web pages using programming tools.

Hey, it's a free country, but to my mind I think there's a useful distinction to be made between web coding and web development:

- » *Web coding* is the pure programming part of creating a web page, particularly using JavaScript/jQuery on the front end and PHP on the back end.
- » *Web development* is the complete web page creation package, from building a page with HTML tags, to formatting the page with CSS, to storing data on the back end with MySQL, to accessing that data with PHP, to bridging the front and back ends using JavaScript and jQuery.

However you look at it, this book teaches you everything you need to know to become both a web coder and a web developer.

IN THIS CHAPTER

- » Understanding the need for a web development environment
- » Gathering the tools you need for a local development setup
- » Installing a local web development environment on a Windows PC
- » Installing a local web development environment on a Mac
- » Learning what to look for in a good text editor

Chapter 2

Setting Up Your Web Development Home

He is happiest, be he king or peasant, who finds peace in his home.

— JOHANN WOLFGANG VON GOETHE

One of the truly amazing things about web development is that, with the exception of the databases on the server, all you ever work with are basic text files. But surely all the structure you add with HTML tags requires some obscure and complex file type? No way, José: It's text all the way down. What about all that formatting stuff associated with CSS? Nope: nothing but text. PHP? Text. JavaScript and jQuery? Text and, again, text.

What this text-only landscape means is that you don't need any highfalutin, high-priced software to develop for the web. A humble text editor is all you require to dip a toe or two in the web coding waters.

But what if you want to get more than your feet wet in web coding? What if you want to dive in, swim around, perhaps do a little snorkeling? Ah, then you need to take things up a notch or three and set up a proper web development environment

on your computer. This will give you everything you need to build, test, and refine your web development projects. In this chapter, you get your web coding adventure off to a rousing start by exploring how to set up a complete web development environment on your Windows PC or Mac.

What Is a Local Web Development Environment?

In programming circles, an *integrated development environment* (IDE) is a collection of software programs that make it easy and efficient to write code. Most development environments are tailored to a particular programming language and come with tools for editing, testing, and compiling code (that is, converting the code to its final form as an application).

In the web coding game, we don't have IDEs, *per se*, but we do have a similar beast called a *local web development environment*, which is also a collection of software. It usually includes the following:

- » A web server
- » A relational database management system (RDBMS) to run on the web server
- » A server-side programming language
- » An interface for controlling (starting, stopping, and so on) the web server
- » An interface for accessing and manipulating the RDBMS

The key point to grok here is that this is a “local” web development environment, which means that it gets installed on your PC or Mac. This enables you to build and test your web development projects right on your computer. You don't need a web hosting service or even an Internet connection, for that matter. Everything runs conveniently on your computer, so you can concentrate on coding and leave the deployment of the site until you're ready.

Do You Need a Local Web Development Environment?

Okay, if it's possible to use a simple text editor to develop web pages, why not do just that? After all, every Windows PC and Mac in existence comes with a

pre-installed text editor, and there are lots of free third-party text editors ripe for downloading, so why bother installing the software for a local web development environment?

To be perfectly honest, I'm not going to stand here and tell you that a local web development setup is a must. Certainly if all you're doing for now is getting started with a few static web pages built using HTML, CSS, and JavaScript, then you don't yet need access to the back end. Similarly, if you're building websites and web apps for your own use and you already have a web host that gives you access to MySQL and PHP, then you can definitely get away with using just your trusty text editor.

However, there are two major exceptions that pretty much require you to build your web stuff locally:

- » If you're building a website or app for someone else and you don't have access to their web server.
- » If you're building a new version of an existing website or app, which means that you don't want to mess with the production code while tinkering (and therefore making mistakes) with the new code.

That said, there's also something undeniably cool about having a big-time web server purring away in the background of your computer. So, even if you don't think you'll need a full-blown web development environment in the short term, think about installing one anyway, if only so you can say you're "running Apache 2.4 locally" at your next cocktail party.

Setting Up the XAMPP for Windows Development Environment

If you're running Windows, then I highly recommend the web development environment XAMPP for Windows, which in its most recent version (at least as I write this in early 2018) requires Windows Vista or later. XAMPP for Windows is loaded with dozens of features, but for our needs the following are the most important:

- » **Apache:** This is an open-source web server that runs about half of all the websites on Earth.
- » **MariaDB:** This is an open-source server database that is fully compatible with MySQL (discussed in Book 1, Chapter 1).

- » **PHP:** This is the server-side programming language that I talk about briefly in Book 1, Chapter 1.
- » **phpMyAdmin:** This is an interface that enables you to access and manipulate MariaDB databases.

So all of this requires big bucks, right? Nope. XAMPP for Windows is completely free.

To get started, head for the Apache Friends website at www.apachefriends.org, and then download XAMPP for Windows. Be sure to get the most recent version.

Installing XAMPP for Windows

Once the download is complete, follow these steps to install XAMPP for Windows:

- 1. Open the installation file that you downloaded.**

The download is an executable file, so you can double-click it to get the installation off the ground.

- 2. Enter your User Account Control (UAC) credentials to allow the install.**

If you're the administrator of your PC, click Yes. Otherwise, you need to enter the username and password of the PC's administrator account.

- 3. When XAMPP displays a warning about installing with UAC activated, click OK.**

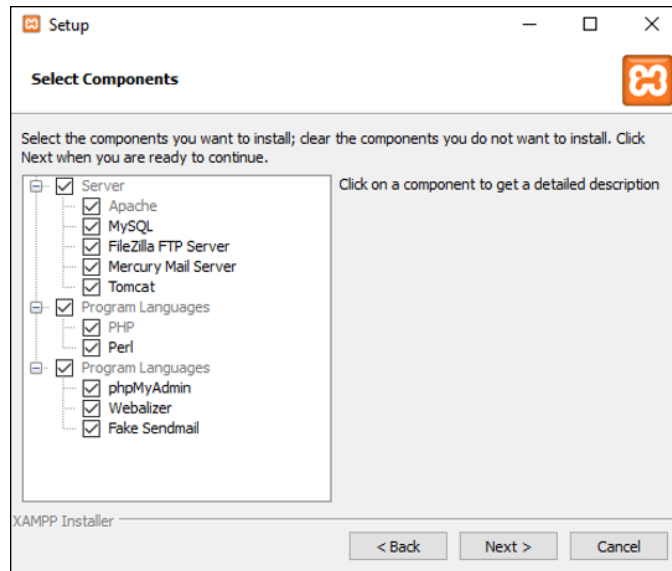
This oddly worded warning means that if you install XAMPP in the default folder (usually `C:\Program Files`), then it might have problems running normally because UAC imposes restrictions on that folder. You can ignore this because later (see Step 6) I show you how to install XAMPP in a different folder that doesn't suffer from this problem.

- 4. When the XAMPP Setup Wizard appears, click Next.**

- 5. In the Select Components dialog box (see Figure 2-1), deselect the check box beside any component you don't want installed, and then click Next.**

For a basic install, you only need Apache, MySQL, PHP, and phpMyAdmin. If your PC is running low on disk space, consider not installing the other components. If you're rich in disk space, go ahead and install everything because, hey, after all of this you might be inspired to learn Perl (which is another server-side programming language).

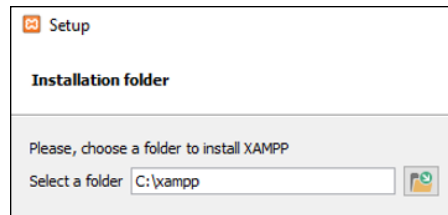
FIGURE 2-1:
Use this Setup Wizard dialog box to deselect the check box beside any component you don't want installed.



6. In the Installation Folder dialog box, type the location where you want XAMPP installed, then click Next.

Be sure to avoid the folders `C:\Program Files` and `C:\Program Files (x86)`, for the reason I described back in Step 3. Most folks create a `xampp` folder in `C:\` and install everything there (see Figure 2-2).

FIGURE 2-2:
To install XAMPP, use a subfolder in the main `C:\` folder (such as `C:\xampp`).



7. The Setup Wizard lets you know that Bitnami for XAMPP can install content management systems such as WordPress and Drupal. Click OK.

If you don't care about any of this, be sure to deselect the Learn More About Bitnami for XAMPP check box before you click OK.

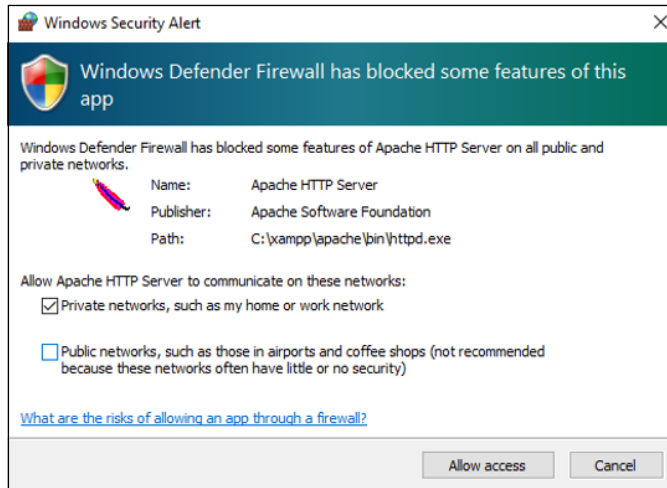
8. Click Next to begin the installation.

9. If you see a Windows Security Alert similar to the one shown in Figure 2-3, select the Private Networks check box, deselect the Public Networks check box, and then click Allow Access.



However, just because you select the Private Networks check box, it doesn't mean that people on your network can access (much less mess with) your local web server. XAMPP for Windows is configured out of the box to be accessible only from the computer on which it's installed.

FIGURE 2-3: If the Windows Security Alert dialog box shows up, be sure to allow Apache to communicate on your private network, but not on any public networks.



10. When the install is complete, click Finish.

Be sure to deselect the Do You Want to Start the Control Panel Now check box. I talk about the correct way to start the Control Panel in the next section.

Running the XAMPP for Windows Control Panel

The XAMPP Control Panel enables you to start, stop, and configure the XAMPP apps, particularly the Apache web server and the MySQL database system. For best results, you should start the program with administrator privileges, which you can do by following these steps:

1. **Click Start.**
2. **Find and open the XAMPP folder in the All Apps list.**

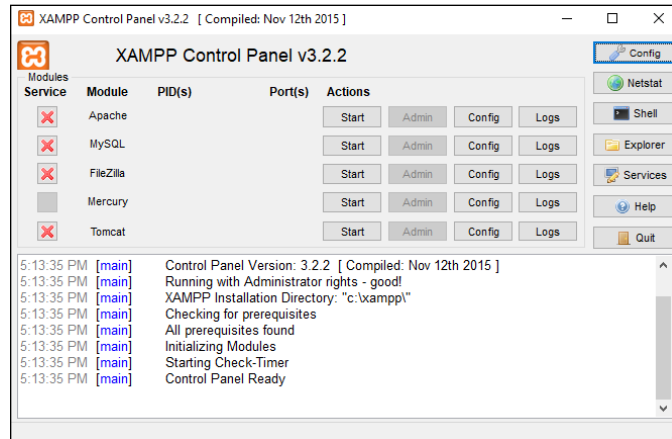
Depending on your version of Windows, you might have to click All Apps to get to the All Apps list.
3. **Right-click XAMPP Control Panel, click More, and then click Run as Administrator.**

Depending on your version of Windows, you might not have to click More to get to the Run as Administrator command.

4. **If you're the administrator of your PC, click Yes. Otherwise, you need to enter the username and password of the PC's administrator account.**
5. **The first time you run the Control Panel, you're asked to choose a language. Select the radio button for the language you prefer, then click Save.**

The XAMPP Control Panel appears, as shown in Figure 2-4.

FIGURE 2-4:
You use the XAMPP Control Panel to control and configure apps such as Apache and MySQL.



To start an app, click the corresponding Start button. That button name changes to Stop, meaning you can later stop the service by clicking its Stop button.



TIP

You'll always want the Apache and MySQL apps running, so you can save a bit of time by having the XAMPP Control Panel launch these two apps automatically when you open the program. Click Config, select the Apache and MySQL check boxes, and then click Save.



REMEMBER

If when you start an app you see a Windows Security Alert dialog box similar to the one shown earlier in Figure 2-3. Select the Private Networks check box, deselect the Public Networks check box, and then click Allow Access.

Accessing your local web server

With XAMPP for Windows installed and Apache up and running, congratulations are in order: You've got a web server running on your PC! That's great, but how do

you access your shiny, new web server? There are two ways, depending on what you're doing:

- » **Adding files and folders to the web server:** Place the files and folders in the `htdocs` subfolder of your main XAMPP install folder. For example, if you installed XAMPP to `C:\xampp`, then your web server's root folder will be `C:\xampp\htdocs`.
- » **Viewing the files and folders on the server:** Open your favorite web browser and navigate to the `localhost` address (or to `127.0.0.1`, which gets you to the same place). If you have the XAMPP Control Panel open, you can also click the Apache app's Admin button.

By default, your local website is configured to automatically redirect `localhost` to `localhost/dashboard/`, shown in Figure 2-5, which gives you access to several XAMPP tools.

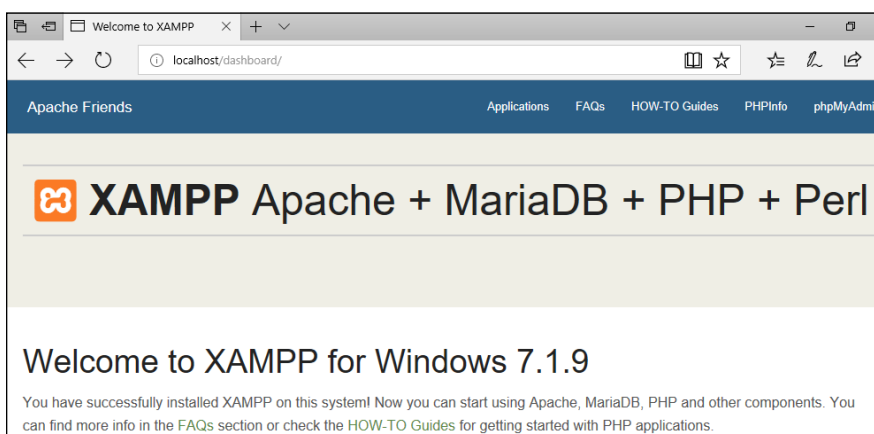


FIGURE 2-5:
The `localhost/dashboard/` address gives you access to a few XAMPP tools.

In the page header, you can use the following links:

- » **Apache Friends:** Returns you to the main Dashboard page.
- » **Applications:** Provides information about installing Bitnami applications on the server.
- » **FAQs:** Displays a list of XAMPP frequently asked questions.
- » **How-To Guides:** Displays a list of links to step-by-step guides for a number of XAMPP for Windows tasks.
- » **PHPInfo:** Displays a for-geeks-only page of information about the version of PHP that you have installed.

- » **phpMyAdmin:** Opens the phpMyAdmin tool, which lets you create and manipulate MariaDB/MySQL databases. You can also open phpMyAdmin by navigating directly to `localhost/phpmyadmin/`, or in the XAMPP Control Panel, by clicking the MySQL app's Admin button. However you get there, just be sure to have the MySQL app running before you open phpMyAdmin.

Setting Up the XAMPP for OS X Development Environment

If you'll be doing your web work on a Mac, then I recommend the web development environment XAMPP for OS X, which in its most recent version (at least as I write this in early 2018) requires OS X Snow Leopard (10.6) or later. XAMPP for OS X is packed with programs and features, but you'll probably only concern yourself with the following:

- » **Apache:** This is an open-source web server that runs about half of all the websites on Earth.
- » **MariaDB:** This is an open-source server database that is fully compatible with MySQL (discussed in Book 1, Chapter 1).
- » **PHP:** This is the server-side programming language that I mention in Book 1, Chapter 1.
- » **phpMyAdmin:** This is an interface that enables you to access and work with MariaDB databases.

The best news of all is XAMPP for OS X is completely, utterly, and forever free. Nice! To get the show on the road, surf to the Apache Friends website at www.apachefriends.org, and then download the most recent version of XAMPP for OS X.

Installing XAMPP for OS X

Once the download is done, follow these steps to install XAMPP for OS X:

1. Double-click the installation file that you downloaded.
2. Double-click the XAMPP icon.
3. If macOS warns you about opening an application downloaded from the Internet, say "It's cool, bro" and click Open.
4. Enter your macOS administrator password and then click OK.

5. When the XAMPP Setup Wizard appears, click Next.
6. In the Select Components dialog, deselect the XAMPP Developer Files check box, as shown in Figure 2-6, and then click Next.

The developer files might sound like they're right up your alley, but they're actually for people who want to add to or modify the code for XAMPP itself.

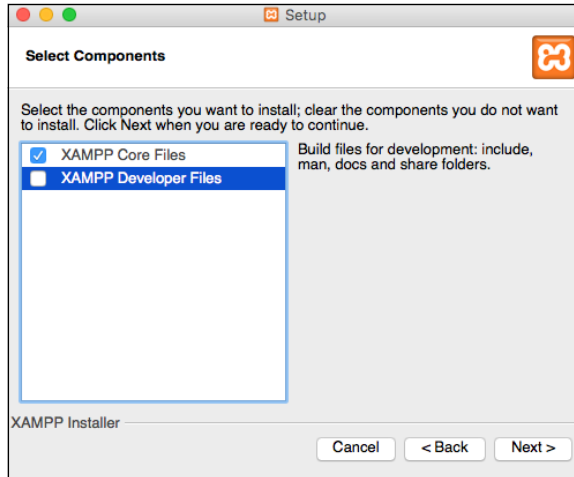


FIGURE 2-6: Use this Setup Wizard dialog to deselect the check box beside XAMPP Developer Files.

7. In the Installation Directory dialog, click Next.
 8. The Setup Wizard lets you know that Bitnami for XAMPP can install content management systems such as WordPress and Drupal. Click Next.
- If you don't care about any of this, be sure to deselect the Learn More About Bitnami for XAMPP check box before you click Next.
9. Click Next to launch the installation.
 10. When the install is complete, click Finish.

If you want to head right into the XAMPP Manager, leave the Launch XAMPP check box selected.



REMEMBER

What about the security of your local web server? Fortunately, that's not an issue because people on your network can't access your web server. XAMPP is configured by default to be accessible only from the Mac on which it's installed.

Running the XAMPP Application Manager

The XAMPP Application Manager enables you to start, stop, and configure the XAMPP servers, particularly the Apache web server and the MySQL database system. To launch the XAMPP Application Manager, you have two choices:

- » If you still have the final Setup Wizard dialog onscreen, leave the Launch XAMPP check box selected and click Finish.
- » In Finder, open the Applications folder, open the XAMPP folder, and then double-click Manager-OSX.

The XAMPP Application Manager appears. To work with the XAMPP servers, click the Manage Servers tab, shown in Figure 2-7.

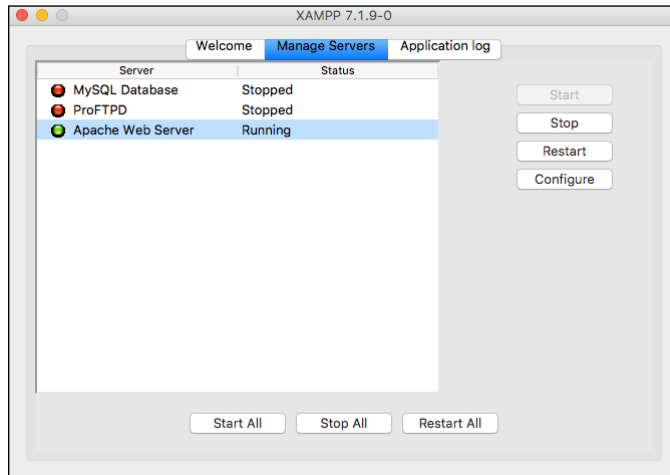


FIGURE 2-7:
You use the XAMPP Control Panel to control and configure services such as Apache and MySQL.

In the Manage Servers tab, you can perform the following actions:

- » **Start a server.** Click the server and then click Start.
- » **Start all the servers.** Click Start All.
- » **Restart a server.** Click the server and then click Restart.
- » **Restart all the servers.** Click Restart All.
- » **Stop a server.** Click the server and then click Stop.
- » **Stop all the servers.** Click Stop All.

Accessing your local web server

With XAMPP for OS X installed and Apache up and running, it's time for high-fives all around because you've got a web server running on your Mac! That's

awesome, but how do you access your web server? There are two ways, depending on what you're doing:

- » **Adding files and folders to the web server:** Place the files and folders in the htdocs subfolder of your main XAMPP install folder. To get there, open Applications, then XAMPP, then double-click htdocs. If you have the XAMPP Application Manager open, click the Welcome tab, click Open Application Folder, then open htdocs.
- » **Viewing the files and folders on the server:** Open your favorite web browser and navigate to the localhost address (or to 127.0.0.1, which gets you to the same place). If you have the XAMPP Application Manager running, click the Welcome tab and then click Go To Application.

By default, your local website is configured to automatically redirect localhost to localhost/dashboard/, shown in Figure 2-8, which gives you access to several XAMPP tools.

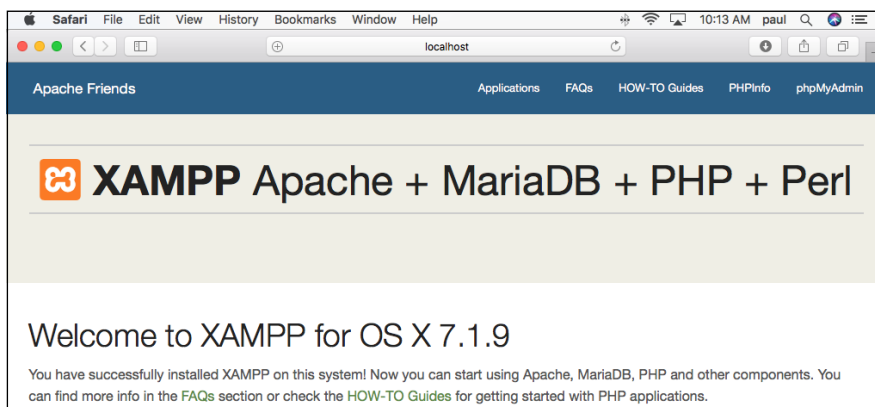


FIGURE 2-8:
The localhost/
dashboard/
address gives you
access to a few
XAMPP for OS X
features.

In the page header, you can use the following links:

- » **Apache Friends:** Returns you to the main Dashboard page.
- » **Applications:** Provides information about installing Bitnami applications on the server.
- » **FAQs:** Displays a list of XAMPP frequently asked questions.
- » **How-To Guides:** Displays a list of links to step-by-step guides for a number of XAMPP for OS X tasks.

- » **PHPInfo:** Displays a for-geeks-only page of information about the version of PHP that you have installed.
- » **phpMyAdmin:** Opens the phpMyAdmin tool, which lets you create and manipulate MariaDB/MySQL databases. You can also open phpMyAdmin by navigating directly to `localhost/phpmyadmin/`. Either way, make sure you have the MySQL Database server running before you open phpMyAdmin.

Choosing Your Text Editor

I mention at the beginning of this chapter that all you need to develop web pages is a text editor. However, saying that all you need to code is a text editor is like saying that all you need to live is food: It's certainly true, but more than a little short on specifics. After all, to a large extent the quality of your life depends on the food you eat. If you survive on nothing but bread and water, well "surviving" is all you're doing. What you really need is a balanced diet that supplies all the nutrients your body needs. And pie.

The bread-and-water version of a text editor is the barebones program that came with your computer: Notepad if you run Windows, or TextEdit if you have a Mac. You can survive as a web developer using these programs, but that's not living, if you ask me. You need the editing equivalent of vitamins and minerals (and, yes, pie) if you want to flourish as a web coder. These nutrients are the features and tools that are crucial to being an efficient and organized developer:

- » **Syntax highlighting:** *Syntax* refers to the arrangement of characters and symbols that create correct programming code, and *syntax highlighting* is an editing feature that color-codes certain syntax elements for easier reading. For example, while regular text might appear black, all the HTML tags might be shown in blue and the CSS properties might appear red. The best text editors let you choose the syntax colors, either by offering prefab themes, or by letting you apply custom colors.
- » **Line numbers:** It might seem like a small thing, but having a text editor that numbers each line, as shown in Figure 2-9, can be a major timesaver. When the web browser alerts you to an error in your code (see Book 3, Chapter 9), it gives you an error message and, crucially, the line number of the error. This enables you to quickly locate the culprit and (fingers crossed) fix the problem pronto.
- » **Code previews:** A good text editor will let you see a preview of how your code will look in a web browser. The preview might appear in the same window as your code, or in a separate window, and it should update automatically as you modify and save your code.

FIGURE 2-9: Line numbers, as seen here down the left side of the window, are a crucial text editor feature.

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="utf-8">
5   <title>Web Coding and Web Dev AIO for Dummies</title>
6   <meta name="author" content="Paul McFedries">
7   <meta name="viewport" content="width=device-width, initial-scale=1.0">
8   <link rel="stylesheet" href="/css/styles.css">
9   <script src="/js/jquery-1.11.1.min.js"></script>
10  <script src="/js/clipboard.min.js"></script>
11  <script src="/js/my-js.js"></script>
12  <script>
13    $(document).ready(function () {
14      // Load the nav bar
15      $("#navbar-code").load("/includes/navbar-code.txt");
16
17      // Load the footer
18      $("#footer-code").load("/php/write-footer.php");
19
20      // Get the example files
21      $.getJSON('get-examples.php', function(data) {
22        var str = '';
23        var strCode = '';
24        var strExample = '';
25        var currBook = data[0].Book;
26        var currChapter = data[0].Chapter;
27
28        // Write the initial item
29        str += '<div class="wcd-book" id="book" + currBook + ">';
30        str += '    Book ' + data[0].Book + ': ' + data[0].BookTitle;
31        str += '    <div class="wcd-chapters" id="book" + currBook + '-chapters">';
32        str += '        <div class="wcd-chapter" id="book" + currBook + '-chapter' + currChapter + ">';
33        str += '            Chapter ' + data[0].Chapter + ': ' + data[0].ChapterTitle;
```

- » **Code completion:** This is a handy feature that, when you start typing something, displays a list of possible code items that complete your typing. You can then select the one you want and press Tab or Enter to add it to your code without having to type the whole thing.
- » **Text processing:** The best text editors offer a selection of text processing features, such as automatic indentation of code blocks, converting tabs to spaces and vice versa, shifting chunks of code right or left, removing unneeded spaces at the end of lines, hiding blocks of code, and more.

The good news is that there's no shortage of text editors that support all these features and many more. That's also the bad news, because it means you have a huge range of programs to choose from. To help you get started, here, in alphabetical order, are a few editors to take for test drives:

- » **Atom:** Available for Windows and Mac. Free! <http://atom.io>
- » **Brackets:** Available for Windows and Mac. Also free! <http://brackets.io/>
- » **Coda:** Available for Mac only. \$99, but a free trial is available. www.panic.com/coda
- » **Notepad++:** Available for Windows only. Another freebie. <https://notepad-plus-plus.org/>
- » **Sublime Text:** Available for both Windows and Mac. \$80, but a free trial is available. www.sublimetext.com
- » **TextMate:** Available for Mac only. \$60, but a free trial is available. <http://macromates.com/>

IN THIS CHAPTER

- » Understanding web hosting providers
- » Examining the various choices for hosting your site
- » Choosing the host that's right for you
- » Looking around your new web home
- » Getting your site files to your web host

Chapter 3

Finding and Setting Up a Web Host

You will end up with better software by releasing as early as practically possible, and then spending the rest of your time iterating rapidly based on real-world feedback. So trust me on this one: Even if version 1 sucks, ship it anyway.

— JEFF ATTWOOD

You build your web pages from the comfort of your Mac or PC, and if you've chosen your text editor well (as I describe in Book 1, Chapter 2), then you can even use your computer to preview how your web pages will look in a browser.

That's fine and dandy, but I think you'll agree that the whole point of building a web page is to, you know, put it on the web! First, you need to subject your code to the wilds of the wider web to make sure it works out there. Even if it seems to be running like a champ on your local server, you can't give it the seal of approval until you've proven that it runs champlike on a remote server. Second, once your code is ready, then the only way the public can appreciate your handiwork is to get it out where they can see it.

Whether you're testing or shipping your code, you need somewhere to put it, and that's what this chapter is about. Here you explore the wide and sometimes wacky world of web hosts. You delve into what they offer, investigate ways to choose a good one, and then take a tour of your web home away from home.

Understanding Web Hosting Providers

A common question posed by web development newcomers is “Where the heck do I put my web page when it's done?” If you've asked that question, you're doing okay because it means you're clued in to something crucial: Just because you've created a web page and you have an Internet connection doesn't mean your site is automatically a part of the web.

After all, people on the web have no way of getting to your computer. Even if you're working with a local web development environment (which I discuss in Book 1, Chapter 2), you're working in splendid isolation because no one either on your network or on the Internet can access that environment.

In other words, your computer isn't set up to hand out documents (such as web pages) to remote visitors who ask for them. Computers that can do this are called *servers* (because they “serve” stuff out to the web), and computers that specialize in distributing web pages are called *web servers*. So your web page isn't on the web until you store it on a remote web server. Because this computer is, in effect, playing “host” to your pages, such machines are also called *web hosts*. Companies that run these web hosts are called *web hosting providers*.

Now, just how do you go about finding a web host? Well, the answer to that depends on a bunch of factors, including the type of site you have, how you get connected to the Internet in the first place, and how much money (if any) you're willing to fork out for the privilege. In the end, you have three choices:

- » Your existing Internet provider
- » A free hosting provider
- » A commercial hosting provider

Using your existing Internet provider

If you access the Internet via a corporate or educational network, your institution might have its own web server you can use. If you get online via an Internet service provider (ISP), phone or email its customer service department to ask

whether the company has a web server available. Almost all ISPs provide space so their customers can put up personal pages free of charge.

Finding a free hosting provider

If cash is in short supply, a few hosting providers will bring your website in from the cold out of the goodness of their hearts. In some cases, these services are open only to specific groups such as students, artists, nonprofit organizations, and so on. However, plenty of providers put up personal sites free of charge.

What's the catch? Well, there are almost always restrictions both on how much data you can store and on the type of data you can store (no ads, no dirty pictures, and so on). You might also be required to display some kind of "banner" advertisement for the hosting provider on your pages.

Signing up with a commercial hosting provider

For personal and business-related websites, many web artisans end up renting a chunk of a web server from a commercial hosting provider. You normally hand over a setup fee to get your account going and then you're looking at a monthly fee.

Why shell out all that dough when there are so many free sites lying around? Because, as with most things in life, you get what you pay for. By paying for your host, you generally get more features, better service, and fewer annoyances (such as the ads that some free sites have to display).

A Buyer's Guide to Web Hosting

Unfortunately, choosing a web host isn't as straightforward as you might like it to be. For one thing, hundreds of hosts are out there clamoring for your business; for another, the pitches and come-ons your average web host employs are strewn with jargon and technical terms. I can't help reduce the number of web hosts, but I can help you understand what those hosts are yammering on about. Here's a list of the terms you're most likely to come across when researching web hosts:

- » **Storage space:** Refers to the amount of room allotted to you on the host's web server to store your files. The amount of acreage you get determines the amount of data you can store. For example, if you get a 1MB (1 megabyte) limit, you can't store more than 1MB worth of files on the server. HTML files

don't take up much real estate, but large graphics sure do, so you need to watch your limit. For example, you could probably store about 200 pages in 1MB of storage (assuming about 5KB per page), but only about 20 images (assuming about 50KB per image). Generally speaking, the more you pay for a host, the more storage space you get.

» **Bandwidth:** A measure of how much of your data the server serves. For example, suppose the HTML file for your page is 1KB (1 kilobyte) and the graphics associated with the page consume 9KB. If someone accesses your page, the server ships out a total of 10KB; if ten people access the page (either at the same time or over a period of time), the total bandwidth is 100KB. Most hosts give you a bandwidth limit (or “cap”), which is most often a certain number of megabytes or gigabytes per month. (A gigabyte is equal to about 1,000 megabytes.) Again, the more you pay, the greater the bandwidth you get.



WARNING

If you exceed your bandwidth limit, users will usually still be able to get to your pages (although some hosts shut down access to an offending site). However, almost all web hosts charge you an extra fee for exceeding your bandwidth, so check this out before signing up. The usual penalty is a set fee per every megabyte or gigabyte over your cap.

» **Domain name:** A general Internet address, such as `wiley.com` or `whitehouse.gov`. They tend to be easier to remember than the long-winded addresses most web hosts supply you by default, so they're a popular feature. Two types of domain names are available:

- A regular domain name (such as `yourdomain.com` or `yourdomain.org`)
- A subdomain name (such as `yourdomain.webhostdomain.com`)

To get a regular domain, you either need to use one of the many domain registration services such as GoDaddy or Register.com. A more convenient route is to choose a web hosting provider that will do this for you. Either way, it will usually cost you \$35 per year (although some hosts offer cheap domains as a “loss leader” and recoup their costs with hosting fees; also, discount domain registrars such as GoDaddy offer domains for as little as \$9.99 per year). If you go the direct route, almost all web hosts will host your domain, which means that people who use your domain name will get directed to your website on the host's web server. For this to work, you must tweak the domain settings on the registrar. This usually involves changing the DNS servers associated with the domain so that they point at the web host's domain name servers. Your web host will give you instructions on how to do this.

With a subdomain name, “webhostdomain.com” is the domain name of the web hosting company, and it simply tacks on whatever name you want to the beginning. Many web hosts will provide you with this type of domain, often for free.

- » **Email addresses:** Most hosts offer you one or more email addresses along with your web space. The more you pay, the more mailboxes you get. Some hosts offer *email forwarding*, which enables you to have messages that are sent to your web host address rerouted to some other email address.
- » **Shared server:** If the host offers a *shared server* (or *virtual server*), it means that you'll be sharing the server with other websites — dozens or even hundreds of them. The web host takes care of all the highly technical server management chores, so all you have to do is maintain your site. This is by far the best (and cheapest) choice for individuals or small business types.
- » **Dedicated server:** You get your very own server computer on the host. That may sound like a good thing, but it's usually up to you to manage the server, which can be a dauntingly technical task. Also, dedicated servers are much more expensive than shared servers.
- » **Operating system:** The operating system on the web server. You usually have two choices: Unix (or Linux) and Windows Server. Unix systems have the reputation of being very reliable and fast, even under heavy traffic loads, so they're usually the best choice for a shared server. Windows systems are a better choice for dedicated servers because they're easier to administer than their Unix brethren. Note, too, that Unix servers are case sensitive in terms of file and directory names, while Windows servers are not.
- » **Databases:** The number of databases you get to create with your account. Unix systems usually offer MySQL databases, whereas Windows servers offer SQL Server databases.
- » **Administration interface:** This is the host app that you use to perform tasks on the server, such as uploading files or creating users. Many hosts offer the excellent cPanel interface, and most Unix-based systems offer the phpMyAdmin app for managing your MySQL data.
- » **Ad requirements:** A few free web hosts require you to display some type of advertising on your pages. This could be a banner ad across the top of the page, a "pop-up" ad that appears each time a person accesses your pages, or a "watermark" ad, usually a semitransparent logo that hovers over your page. Fortunately, free hosts that insist on ads are rare these days.
- » **Uptime:** The percentage of time the host's server is up and serving. There's no such thing as 100 percent uptime because all servers require maintenance and upgrades at some point. However, the best hosts have uptime numbers over 99 percent. (If a host doesn't advertise its uptime, it's probably because it's very low. Be sure to ask before committing yourself.)
- » **Tech support:** If you have problems setting up or accessing your site, you want to know that help — in the form of *tech support* — is just around the corner. The best hosts offer 24/7 tech support, which means you can contact the company — either by phone or email — 24 hours a day, 7 days a week.

- » **FTP support:** You usually use the Internet's *FTP* service to transfer your files from your computer to the web host. If a host offers *FTP access* (some hosts have their own method for transferring files), be sure you can use it any time you want and there are no restrictions on the amount of data you can transfer at one time.
- » **Website statistics:** Tell you things such as how many people have visited your site, which pages are the most popular, how much bandwidth you're consuming, which browsers and browser versions surfers are using, and more. Most decent hosts offer a ready-made stats package, but the best ones also give you access to the "raw" log files so you can play with the data yourself.
- » **Ecommerce:** Some hosts offer a service that lets you set up a web "store" so you can sell stuff on your site. That service usually includes a "shopping script," access to credit card authorization and other payment systems, and the ability to set up a secure connection. You usually get this only in the more expensive hosting packages, and you'll most often have to pay a setup fee to get your store built.
- » **Scalability:** The host is able to modify your site's features as required. For example, if your site becomes very popular, you might need to increase your bandwidth limit. If the host is scalable, it can easily change your limit (or any other feature of your site).

Finding a Web Host

Okay, you're ready to start researching the hosts to find one that suits your web style. As I mention earlier, there are hundreds, perhaps even thousands, of hosts, so how is a body supposed to whittle them down to some kind of short list? Here are some ideas:

- » **Ask your friends and colleagues.** The best way to find a good host is that old standby, word of mouth. If someone you trust says a host is good, chances are you won't be disappointed. This is assuming you and your pal have similar hosting needs. If you want a full-blown ecommerce site, don't solicit recommendations from someone who has only a humble home page.
- » **Solicit host reviews from experts.** Ask existing webmasters and other people "in the know" about which hosts they recommend or have heard good things about. A good place to find such experts is Web Hosting Talk (www.webhostingtalk.com), a collection of forums related to web hosting.

- » **Contact web host customers.** Visit sites that use a particular web host, and send an email message to the webmaster asking what she thinks of the host's service.
- » **Peruse the lists of web hosts.** A number of sites track and compare web hosts, so they're an easy way to get in a lot of research. Careful, though, because there are a lot of sketchy lists out there that are only trying to make a buck by getting you to click ads. Here are some reputable places to start:
 - **CNET Web Hosting Solutions:** www.cnet.com/web-hosting
 - **PC Magazine Web Site Hosting Services Reviews:** www.pcmag.com/reviews/web-hosting-services
 - **Review Hell:** www.reviewhell.com
 - **Review Signal Web Hosting Reviews:** <http://reviewsignal.com/webhosting>

Finding Your Way around Your New Web Home

After you sign up with a web hosting provider and your account is established, the web administrator creates two things for you: a directory on the server you can use to store your website files, and your very own web address. (This is also true if you're using a web server associated with your corporate or school network.) The directory — which is known in the biz as your *root directory* — usually takes one of the following forms:

```
/yourname/  
/home/yourname/  
/yourname/public_html/
```

In each case, *yourname* is the login name (or username) the provider assigns to you, or it may be your domain name (with or without the .com part). Remember, your root directory is a slice of the host's web server, and this slice is yours to monkey around with as you see fit. This usually means you can do all or most of the following to the root:

- » Add files to the directory.
- » Add subdirectories to the directory.
- » Move or copy files from one directory to another.

- » Rename files or directories.
- » Delete files from the directory.

Your web address normally takes one of the following shapes:

```
http://provider/yourname/  
http://yourname.provider/  
http://www.yourname.com/
```

Here, *provider* is the host name of your provider (for example, `www.hostcompany.com` or just `hostcompany.com`), and *yourname* is your login name or domain name. Here are some examples:

```
http://www.hostcompany.com/mywebsite/  
http://mywebsite.hostcompany.com/  
http://www.mywebsite.com/
```

Your directory and your web address

There's a direct and important relationship between your server directory and your address. That is, your address actually “points to” your directory and enables other people to view the files you store in that directory. For example, suppose I decide to store a file named `thingamajig.html` in my directory and my main address is `http://mywebsite.hostcompany.com/`. This means someone else can view that page by typing the following URL into a web browser:

```
http://mywebsite.hostcompany.com/thingamajig.html
```

Similarly, suppose I create a subdirectory named `stuff` and use it to store a file named `index.html`. A surfer can view that file by convincing a web browser to head for the following URL:

```
http://mywebsite.hostcompany.com/stuff/index.html
```

In other words, folks can surf to your files and directories by strategically tacking on the appropriate filenames and directory names after your main web address.

Making your hard disk mirror your web home

As a web developer, one of the key ways to keep your projects organized is to set up your directories on your computer, and then mirror those directories on your web host. Believe me, this will make your uploading duties immeasurably easier.



REMEMBER

Moving a file from your computer to a remote location (such as your web host's server) is known in the file transfer trade as *uploading*.

This process begins at the root. On the web host, you already have a root directory assigned to you by the hosting provider, so now you need to designate a folder on your computer to be the root mirror. If you're using the XAMPP web development environment (see Book 1, Chapter 2), then the XAMPP installation's `htdocs` subfolder is perfect as your local root. Otherwise, choose or create a folder on your computer to use as the local root.

What you do from here depends on the number of web development projects you're going to build, and the number of files in each project:

- » **A single web development project consisting of just a few files:** In this case, just put all the files into the root directory.
- » **A single web development project consisting of many files:** The more likely scenario for a typical web development project is to have multiple HTML, CSS, JavaScript, and PHP files, plus lots of ancillary files such as images and fonts. Although it's okay to place all your HTML files in the root directory, do yourself a favor and organize all your other files into subfolders by file type: a `css` subfolder for CSS files, a `js` subfolder for JavaScript files, and so on.
- » **Multiple web development projects:** As a web developer, you'll almost certainly create tons of web projects, so it's crucial to organize them. The ideal way to do that is to create a separate root subdirectory for each project. Then within each of these subdirectories, you can create sub-subdirectories for file types such as CSS, JavaScript, images, and so on.

To help you see why mirroring your local and remote directory structures is so useful, suppose you set up a subfolder on your computer named `graphics` that you use to store your image files. To insert into your page a file named `mydog.jpg` from that folder, you'd use the following reference:

```
graphics/mydog.jpg
```

When you send your HTML file to the server and you then display the file in a browser, it looks for `mydog.jpg` in the `graphics` subdirectory. If you don't have such a subdirectory — either you didn't create it or you used a different name, such as `images` — the browser won't find `mydog.jpg` and your image won't show. In other words, if you match the subdirectories on your web server with the subfolders on your computer, your page will work properly without modifications both at home and on the web.



WARNING

One common faux pas beginning web developers make is to include the local drive and all the folder names when referencing a file. Here's an example:

```
C:\xampp\htdocs\graphics\mydog.jpg
```

This image will show up just fine when it's viewed from your computer, but it will fail miserably when you upload it to the server and view it on the web. That's because the `C:\xampp\htdocs\` part exists only on your computer.



WARNING

The Unix (or Linux) computers that play host to the vast majority of web servers are downright finicky when it comes to the uppercase and lowercase letters used in file and directory names. It's crucial that you check the file references in your code to be sure the file and directory names you use match the combination of uppercase and lowercase letters used on your server. For example, suppose you have a CSS file on your server that's named `styles.css`. If your HTML references that file as, say, `STYLES.CSS`, the server won't find the file and your styles won't get applied.

Uploading your site files

Once your web page or site is ready for its debut, it's time to get your files to your host's web server. If the server is on your company or school network, you send the files over the network to the directory set up by your system administrator. Otherwise, you upload the files to the root directory created for you on the hosting provider's web server.

How you go about uploading your site files depends on the web host, but here are the four most common scenarios:

- » **Use an FTP program.** It's a rare web host that doesn't offer support for the File Transfer Protocol (FTP, for short), which is the Internet's most popular method for transfer files from here to there. To use FTP, you usually need to get a piece of software called an *FTP client*, which enables you to connect to your web host's FTP server (your host can provide you with instructions for this) and offers an interface for standard file tasks, such as navigating and creating folders, uploading the files, deleting and renaming files, and so on. Popular Windows clients are CuteFTP (www.globalscape.com/cuteftp) and Cyberduck (<https://cyberduck.io>). For the Mac, try Transmit (<https://panic.com/transmit>) or FileZilla (<https://filezilla-project.org>).
- » **Use your text editor's file upload feature.** Some text editors come with an FTP client built-in, so you can edit a file and then immediately upload it with a single command. The Coda text editor (<https://panic.com/coda>) supports this too-handy-for-words feature.

- » **Use the File Manager feature of cPanel.** I mention earlier that lots of web hosts offer an administration tool called cPanel that offers an interface for hosting tasks such as email and domain management. cPanel also offers a File Manager feature that you can use to upload files and perform other file management chores.
- » **Use the web host's proprietary upload tool.** For some reason, a few web hosts only offer their own proprietary interface for uploading and messing around with files and directories. See your host's Help or Support page for instructions.

Making changes to your web files

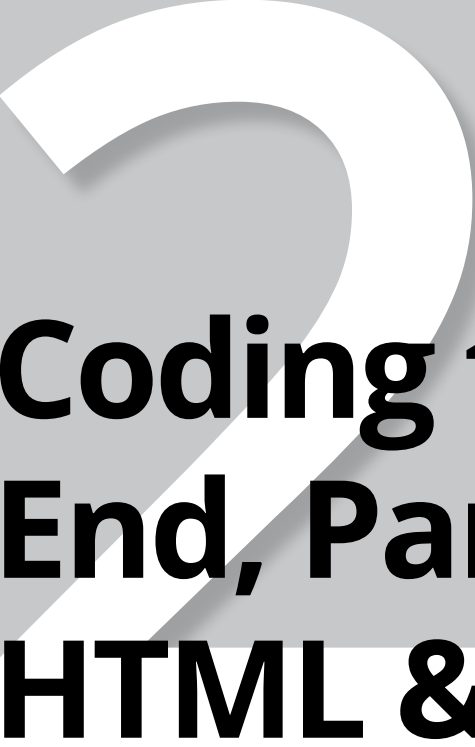
What happens if you send a web development file to your web host and then realize you've made a typing gaffe or you spy a coding mistake? Or what if you have more information to add to one of your web pages? How do you make changes to the files you've already sent?

Well, here's the short answer: You don't. That's right, after you've sent your files, you never have to bother with them again. That doesn't mean you can never update your site, however. Instead, you make your changes to the files that reside on your computer and then send these revised files to your web host. These files replace the old files, and your site is updated just like that.



WARNING

Be sure you send the updated file to the correct directory on the server. Otherwise, you may overwrite a file that happens to have the same name in some other directory.



Coding the Front End, Part 1: HTML & CSS

Contents at a Glance

CHAPTER 1: Structuring the Page with HTML	49
CHAPTER 2: Styling the Page with CSS	79
CHAPTER 3: Sizing and Positioning Page Elements	103
CHAPTER 4: Creating the Page Layout	127

- » Getting comfy with HTML
- » Figuring out HTML tags and attributes
- » Understanding the basic blueprint for all web pages
- » Adding text, images, and links to your page
- » Building bulleted and numbered lists

Chapter **1**

Structuring the Page with HTML

I am always fascinated by the structure of things; why do things work this way and not that way.

— URSUS WEHRLI

When it comes to web development, it's no exaggeration to say that the one indispensable thing, the *sine qua non* for those of you who studied Latin in school, is HTML. That's because absolutely everything else you make as a web developer — your CSS rules, your JavaScript code, even your PHP scripts — can't hang its hat anywhere but on some HTML. These other web development technologies don't even make sense outside of an HTML context.

So, in a sense, this chapter is the most important for you as a web coder because all the rest of the book depends to a greater or lesser degree on the HTML know-how found in the following pages. If that sounds intimidating, not to worry: One of the great things about HTML is that it's not a huge topic, so you can get up to full HTML speed without a massive investment of time and effort.

Because HTML is so important, you'll be happy to know that I don't rush things. You'll get a thorough grounding in all things HTML, and when you're done you'll be more than ready to tackle the rest of your web development education.

Getting the Hang of HTML

Building a web page from scratch using your bare hands may seem like a daunting task. It doesn't help that the codes you use to set up, configure, and format a web page are called the Hypertext Markup Language (HTML for short), a name that could only warm the cockles of a geek's heart. I take a mercifully brief look at each term:

- » **Hypertext:** In prehistoric times — that is, the 1980s — tall-forehead types referred to any text that, when selected, takes you to a different document, as *hypertext*. So this is just an oblique reference to the links that are the defining characteristic of web pages.
- » **Markup:** Instructions that specify how the content of a web page should be displayed in the web browser.
- » **Language:** The set of codes that comprise all the markup possibilities for a page.

But even though the name HTML is intimidating, the codes used by HTML aren't even close to being hard to learn. There are only a few of them, and in many cases they even make sense!

At its most basic, HTML is nothing more than a collection of markup codes — called *tags* — that specify the structure of your web page. In HTML, “structure” is a rubbery concept that can refer to anything from the overall layout of the page all the way down to a single word or even just a character or two.

You can think of a tag as a kind of container. What types of things can it contain? Mostly text, although lots of tags contain things like chunks of the web page and even other tags.

Most tags use the following generic format:

```
<tag>content</tag>
```

What you have here are a couple codes that define a container. Most of these codes are one- or two-letter abbreviations, but sometimes they're entire words. You always surround these codes with angle brackets `<>`; the brackets tell the web browser that it's dealing with a chunk of HTML and not just some random text.

The first of these codes — `<tag>` — is called the *start tag* and it marks the opening of the container; the second of the codes — `</tag>` — is called the *end tag* and it marks the closing of the container. (Note the extra slash (/) that appears in the end tag.)

In between you have the *content*, which refers to whatever is contained in the tag. For example, I start with a simple sentence that might appear in a web page:

```
Okay, listen up people because this is important!
```

Figure 1-1 shows how this might look in a web browser.

FIGURE 1-1:

The sample sentence as it appears in a web browser.

Okay, listen up people because this is important!

Ho hum, right? Suppose you want to punch this up a bit by emphasizing “important.” In HTML, the tag for emphasis is ``, so you’d modify your sentence like so:

```
Okay, listen up people because this is <em>important</em>!
```

See how I’ve surrounded the word *important* with `` and ``? The first `` is the start tag and it says to the browser, “Yo, Browser Boy! You know the text that comes after this? Be a good fellow and treat it as emphasized text.” This continues until the browser reaches the end tag ``, which lets the browser know it’s supposed to stop what it’s doing. So the `` tells the browser, “Okay, okay, that’s enough with the emphasis already!”

All web browsers display emphasized text in italics, so that’s how the word now appears, as you can eyeball in Figure 1-2.

FIGURE 1-2:

The sentence revised to italicize the word *important*.

Okay, listen up people because this is *important*!

There are tags for lots of other structures, including important text, paragraphs, headings, page titles, links, and lists. HTML is just the sum total of all these tags.



WARNING

One of the most common mistakes rookie web weavers make is to forget the slash (/) that identifies an end tag. If your page looks wrong when you view it in a browser, look for a missing slash. Also look for a backslash (\) instead of a slash, which is another common error.

Understanding Tag Attributes

You'll often use tags straight up, but all tags are capable of being modified in various ways. This might be as simple as supplying a unique identifier to the tag for use in a script or a style, or it might be a way to change how the tag operates. Either way, you modify a tag by adding one or more *attributes* to the start tag. Most attributes use the following generic syntax:

```
<tag attribute="value">
```

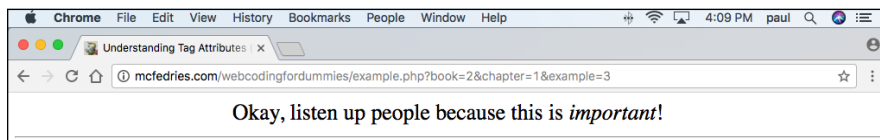
Here, you replace *attribute* with the name of the attribute you want to apply to the tag, and you replace *value* with the value you want to assign the attribute.

For example, the `<hr>` tag adds a horizontal line across the web page (`hr` stands for *horizontal rule*). You use only the start tag in this case (as a simple line, it can't "contain" anything, so no end tag is needed), as demonstrated in the following example:

```
Okay, listen up people because this is <em>important</em>!  
<hr>
```

As you can see in Figure 1-3, the web browser draws a line right across the page.

FIGURE 1-3:
When you add the `<hr>` tag, a horizontal line appears across the page.



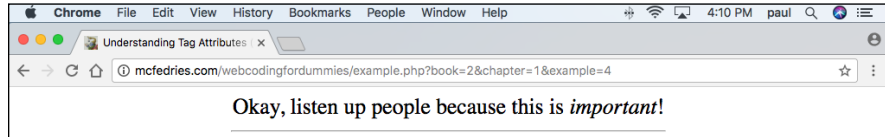
You can also add the `width` attribute to the `<hr>` tag and specify the width you prefer. For example, if you only want the line to traverse half the page width, set the `width` attribute to `"50%"`, as shown here:

```
Okay, listen up people because this is <em>important</em>!  
<hr width="50%">
```

As Figure 1-4 shows, the web browser obeys your command and draws a line that takes up only half the width of the page.

FIGURE 1-4:

The `<hr width="50%">` tag creates a horizontal line across half the page.



Learning the Fundamental Structure of an HTML5 Web Page

In this section, I show you the tags that serve as the basic blueprint you'll use for all your web pages.

Your HTML files will always lead off with the following tag:

```
<!DOCTYPE html>
```

This tag (it has no end tag) is the so-called *Doctype declaration*, and it lets the web browser know what type of document it's about to process (an HTML document, in this case).

Next up you add the `<html lang="en">` tag. This tag doesn't do a whole lot except tell any web browser that tries to read the file that it's dealing with a file that contains HTML doodads. It also uses the `lang` attribute to specify the document's language, which in this case is English.

Similarly, the last line in your document will always be the corresponding end tag: `</html>`. You can think of this tag as the HTML equivalent for "The End." So, each of your web pages will include this on the second line:

```
<html lang="en">
```

and this on the last line:

```
</html>
```

The next items serve to divide the page into two sections: the head and the body. The head section is like an introduction to the page. Web browsers use the head to glean various types of information about the page. A number of items can appear in the head section, but the only one that makes any real sense at this early stage is the title of the page, which I talk about in the next section.

To define the head, add `<head>` and `</head>` tags immediately below the `<html>` tag you typed in earlier. So your web page should now look like this:

```
<!DOCTYPE html>
<html lang="en">
<head>
</head>
</html>
```



REMEMBER

Although technically it makes no difference if you enter your tag names in uppercase or lowercase letters, the HTML powers-that-be prefer to see HTML tags in lowercase letters, so that's the style I use in this book, and I encourage you to do the same.

While you're in the head section, let's add a head-scratcher:

```
<meta charset="utf-8">
```

You place this between the `<head>` and `</head>` tags (indented four spaces for easier reading). It tells the web browser that your web page uses the UTF-8 character set, which you can mostly ignore except to know that UTF-8 contains almost every character (domestic and foreign), punctuation mark, and symbol known to humankind.

The body section is where you enter the text and other fun stuff that the browser will actually display. To define the body, place `<body>` and `</body>` tags after the head section (that is, below the `</head>` tag):

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
</head>
<body>
</body>
</html>
```



WARNING

A common page error is to include two or more copies of these basic tags, particularly the `<body>` tag. For best results, be sure you use each of these seven basic structural tags only one time on each page.

Giving your page a title

When you surf the web, you've probably noticed that your browser displays some text in the current tab. That tab text is the web page title, which is a short (or

sometimes long) phrase that gives the page a name. You can give your own web page a name by adding the `<title>` tag to the page's head section.

To define a title, surround the title text with the `<title>` and `</title>` tags. For example, if you want the title of your page to be “My Home Sweet Home Page,” enter it as follows:

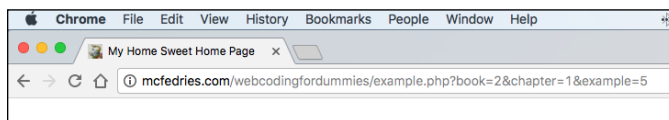
```
<title>My Home Sweet Home Page</title>
```

Note that you always place the title inside the head section, so your basic HTML document now looks like this:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>My Home Sweet Home Page</title>
</head>
<body>
</body>
</html>
```

Figure 1-5 shows this HTML file loaded into a web browser. Notice how the title appears in the browser's tab bar.

FIGURE 1-5:
The text you insert into the `<title>` tag shows up in the browser tab.



Here are a few things to keep in mind when thinking of a title for your page:

- » Be sure your title describes what the page is all about.
- » Don't make your title too long. If you do, the browser might chop it off because there's not enough room to display it in the tab. Fifty or 60 characters are usually the max.
- » Use titles that make sense when someone views them out of context. For example, if someone really likes your page, that person might add it to his or her list of favorites or bookmarks. The browser displays the page title in the favorites list, so it's important that the title makes sense when she looks at the bookmarks later on.

- » Don't use cryptic or vague titles. Titling a page "Link #42" or "My Web Page" might make sense to you, but your readers will almost certainly be scratching their heads.

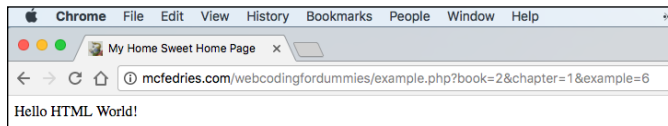
Adding some text

Now it's time to put some flesh on your web page's bones by entering the text you want to appear in the body of the page. For the most part, you can type the text between the `<body>` and `</body>` tags, like this:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>My Home Sweet Home Page</title>
</head>
<body>
Hello HTML World!
</body>
</html>
```

Figure 1-6 shows how a web browser displays this HTML.

FIGURE 1-6:
Text you add
to the page
body appears
in the browser's
content window.



Before you start typing willy-nilly, however, there are a few things you should know:

- » You might think you can line things up and create some interesting effects by stringing together two or more spaces. Ha! Web browsers chew up all those extra spaces and spit them out into the nether regions of cyberspace. Why? Well, the philosophy of the web is that you can use only HTML tags to lay out a document. So a run of multiple spaces (or *white space*, as it's called) is ignored.
- » Tabs also fall under the rubric of white space. You can enter tabs all day long, but the browser ignores them completely.

- » Browsers also like to ignore the carriage return. It might sound reasonable to the likes of you and me that pressing Enter (or Return on a Mac) starts a new paragraph, but that's not so in the HTML world.
- » If you want to separate two chunks of text, you have multiple ways to go, but here are the two easiest:
 - **If you want no space between the texts:** Place a `
` (for line break) tag between the two bits of text.
 - **If you want some breathing room between the texts:** Surround each chunk of text with the `<p>` and `</p>` (for paragraph) tags.
- » If HTML documents are just plain text, does that mean you're out of luck if you need to use characters such as © and €? Luckily, no. For the most part, you can just add these characters to your file. However, HTML also has special codes for these kinds of characters. I talk about them a bit later in this chapter.
- » If, for some reason, you're using a word processor instead of a text editor, know that it won't help to format your text using the program's built-in commands. The browser cheerfully ignores even the most elaborate formatting jobs because browsers understand only HTML (and CSS and JavaScript). And besides, a document with formatting is, by definition, not a pure text file, so a browser might bite the dust trying to load it.

Some Notes on Structure versus Style

One of the key points of front-end web development is to separate the structure of the web page from its styling. This makes the page faster to build, easier to maintain, and more predictable across a range of browsers and operating systems. HTML provides the structure side, while CSS handles the styling.

That's fine as far as it goes, but HTML performs its structural duties with a couple of quirks you need to understand:

- » **This isn't your father's idea of structure.** That is, when you think of the structure of a document, you probably think of larger chunks such as articles, sections, and paragraphs. HTML does all that, but it also deals with structure at the level of sentences, words, and even characters.
- » **HTML's structures often come with some styling attached.** Or, I should say, all web browsers come with predefined styling that they use when they render some HTML tags. Yes, I know I just said that it's best to separate

structure and style, so this can be a tad confusing. Think of it this way: When you build a new deck using cedar, your completed deck has a natural “cedar” look to it, but you’re free to apply a coat of varnish or paint. HTML is the cedar, whereas CSS is the paint.

I mention these quirks because they can help to answer some questions that might arise as you work with HTML tags.



REMEMBER

Another key to understanding why HTML does what it does, is that much of HTML — especially its most recent incarnation, HTML5 — has been set up so that a web page is “understandable” to an extent by software that analyzes the page. One important example is a screen reader used by some visually impaired surfers. If a screen reader can easily figure out the entire structure of the page from its HTML tags, then it can present the page properly to the user. Similarly, software that seeks to index, read, or otherwise analyze the page will only be able to do this successfully if the page’s HTML tags are a faithful representation of the page’s intended structure.

Applying the Basic Text Tags

HTML has a few tags that enable you to add structure to text. Many web developers use these tags only for the built-in browser formatting that comes with them, but you really should try and use the tags *semantically*, as the geeks say, which means to use them based on the meaning you want the text to convey.

Emphasizing text

One of the most common meanings you can attach to text is emphasis. By putting a little extra oomph on a word or phrase, you tell the reader to add stress to that text, which can subtly alter the meaning of your words. For example, consider the following sentence:

```
You'll never fit in there with that ridiculous thing on your head!
```

Now consider the same sentence with emphasis added to one word:

```
You'll never fit in there with that ridiculous thing on your head!
```

You emphasize text on a web page by surrounding that text with the `` and `` tags:

```
You'll <em>never</em> fit in there with that ridiculous thing on  
your head!
```

All web browsers render the emphasized text in italics, as shown in Figure 1-7.

FIGURE 1-7:
The web
browser renders
emphasized
text using italics.

You'll *never* fit in there with that ridiculous thing on your head!

I should also mention that HTML has a closely related tag: `<i>`. The `<i>` tag's job is to mark up *alternative text*, which refers to any text that you want treated with a different mood or role than regular text. Common examples include book titles, technical terms, foreign words, or a person's thoughts. All web browsers render text between `<i>` and `</i>` in italics.

Marking important text

One common meaning that you'll often want your text to convey is importance. It might be some significant step in a procedure, a vital prerequisite or condition for something, or a crucial passage within a longer text block. In each case, you're dealing with text that you don't want your readers to miss, so it needs to stand out from the regular prose that surrounds it.

In HTML, you mark text as important by surrounding it with the `` and `` tags, as in this example:

```
Dear reader: Do you see the red button in the upper-right  
corner of this page? <strong>Never click the red  
button!</strong> You have been warned.
```

All web browsers render text marked up with the `` tag in bold, as shown in Figure 1-8.

FIGURE 1-8:
The browser
renders
important text
using bold.

Dear reader: Do you see the red button in the upper-right corner of this page?
Never click the red button! You have been warned.

Just to keep us all on our web development toes, HTML also offers a close cousin of the `` tag: the `` tag. You use the `` tag to mark up keywords in the text. A *keyword* is a term that you want to draw attention to because it plays a different role than the regular text. It could be a company name or a person's name (think of those famous “bold-faced names” that are the staple of celebrity gossip columns). The browser renders text between the `` and `` tags in a bold font.

Nesting tags

It's perfectly legal — and often necessary — to combine multiple tag types by nesting one inside the other. For example, check out this code:

```
Dear reader: Do you see the red button in the upper-right
corner of this page? <strong>Never, I repeat <em>never</em>,
click the red button!</strong> You have been warned.
```

See what I did there? In the text between the `` and `` tags, I marked up a word with the `` and `` tags. The result? You got it: bold, italic text, as shown in Figure 1-9.

FIGURE 1-9:

The browser usually combines nested tags, such as the bold, italic text shown here.

Dear reader: Do you see the red button in the upper-right corner of this page?
Never, I repeat *never*, click the red button! You have been warned.

Adding headings

Earlier you saw that you can give your web page a title using the aptly named `<title>` tag. However, that title only appears in the browser's title bar and tab. What if you want to add a title that appears in the body of the page? That's almost easier done than said because HTML comes with a few tags that enable you to define *headings*, which are bits of text that appear in a separate paragraph and usually stick out from the surrounding text by being bigger, appearing in a bold typeface, and so on.

There are six heading tags in all, ranging from `<h1>`, which uses the largest type size, down to `<h6>`, which uses the smallest size. Here's some web page code that demonstrates the six heading tags, and Figure 1-10 shows how they look in a web browser:

```
<h1>This is Heading 1</h1>
<h2>This is Heading 2</h2>
```