



Practical Web Design for Absolute Beginners

Adrian W. West

Apress®

www.allitebooks.com

Practical Web Design for Absolute Beginners



Adrian W. West

Apress®

Practical Web Design for Absolute Beginners

Adrian W. West
Colyton, United Kingdom

ISBN-13 (pbk): 978-1-4842-1992-8
DOI 10.1007/978-1-4842-1993-5

ISBN-13 (electronic): 978-1-4842-1993-5

Library of Congress Control Number: 2016957887

Copyright © 2016 by Adrian W. West

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

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

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

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

Managing Director: Welmoed Spahr

Acquisitions Editor: Louise Corrigan

Development Editor: Mark Renfrow

Technical Reviewer: Massimo Nardone

Editorial Board: Steve Anglin, Pramila Balen, Laura Berendson, Aaron Black, Louise Corrigan,

Jonathan Gennick, Todd Green, Robert Hutchinson, Celestin Suresh John, Nikhil Karkal,

James Markham, Susan McDermott, Matthew Moodie, Natalie Pao, Gwenan Spearing

Coordinating Editor: Nancy Chen

Copy Editor: Karen Jameson

Compositor: SPi Global

Indexer: SPi Global

Artist: SPi Global, Cover image designed by freepik.com

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

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

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

Any source code or other supplementary materials referenced by the author in this text is available to readers at www.apress.com. For detailed information about how to locate your book's source code, go to www.apress.com/source-code/.

Printed on acid-free paper

I dedicate this book to the open source community. Without these committed people we would not have free text editors, free FTP programs, and free servers such as XAMPP/Apache. Because the software is free and supported by many forums run by knowledgeable enthusiasts, there are no obstacles preventing my readers and students from learning how to produce excellent websites from scratch.

—Adrian W. West

Contents at a Glance

About the Author	xxvii
About the Technical Reviewer	xxix
Acknowledgments	xxxi
Introduction	xxxiii
■ Chapter 1: Creating Websites	1
■ Chapter 2: Create Your First Website and Add Hyperlinks	13
■ Chapter 3: Styling the Website with CSS	19
■ Chapter 4: Create Web Pages with Three Columns Using CSS	27
■ Chapter 5: Create Web Pages with Four Columns Using CSS	35
■ Chapter 6: Add Pictures to Websites	43
■ Chapter 7: Enhancing the Website.....	51
■ Chapter 8: Rollover Menu Buttons.....	59
■ Chapter 9: Using Color Effectively	69
■ Chapter 10: Screen Size and Resolution: Useful Templates: Rounded Corners.....	79
■ Chapter 11: A Vertical Menu with a Picture Gallery	87
■ Chapter 12: A Horizontal Menu with an Enlarged Picture Gallery	99
■ Chapter 13: More about Website Images: Create a New Appearance with Tiles.....	111
■ Chapter 14: Vertical and Horizontal Menus on the Same Page: Colored Columns	121
■ Chapter 15: Create Tiles: Use Two New Tiles: Float-Drop and the Box Model ...	131

■ **Chapter 16: Create Tables for Data..... 139**

■ **Chapter 17: The Secret of Attractive and Useful Websites 147**

■ **Chapter 18: Design a Feedback Form..... 157**

■ **Chapter 19: Search Engine Optimization..... 169**

■ **Chapter 20: Positioning Elements on a Web Page 183**

■ **Chapter 21: Save Time and Reduce Tedium..... 195**

■ **Chapter 22: More on Using PHP *include*..... 213**

■ **Chapter 23: Receive Emails from a *Contact Us* Page..... 223**

■ **Chapter 24: Add Slideshows and Videos 239**

■ **Chapter 25: Create a Tab Menu 251**

■ **Chapter 26: Designing a Drop-Down Menu 261**

■ **Chapter 27: Drop Shadows 271**

■ **Chapter 28: User Name and Password for a Member’s Page 281**

■ **Chapter 29: Create a Printable Order Form 289**

■ **Chapter 30: Add a Search Field to Your Website 301**

■ **Chapter 31: Styled Bullet Points..... 309**

■ **Chapter 32: Indicating Which Horizontal Menu Button Has Been Clicked..... 317**

■ **Chapter 33: Indicating Which Vertical Menu Button Has Been Clicked..... 325**

■ **Chapter 34: Creating Multi-row Menus and Picture Galleries..... 333**

■ **Chapter 35: Building Responsive Websites for Mobile Devices Part 1 343**

■ **Chapter 36: Building Responsive Websites for Mobile Devices Part 2..... 359**

■ **Chapter 37: Building Responsive Websites for Mobile Devices Part 3 371**

■ **Chapter 38: Building Responsive Websites for Mobile Devices Part 4 381**

■ **Chapter 39: Avoiding Some of the Pitfalls of a CMS Website 397**

■ Chapter 40: Go Live and Validate Your Website	403
■ Chapter 41: Quick Reference: Graphics Programs: Resources.....	429
■ Chapter 42: Installing and Using Text Editors	443
Index.....	463

Contents

About the Author	xxvii
About the Technical Reviewer	xxix
Acknowledgments	xxx
Introduction	xxxiii
■ Chapter 1: Creating Websites	1
Definitions	1
The Two Website Creation Methods	2
The Advantages of Using Code Rather Than CMS	2
Who's Afraid of HTML?	4
But What About CSS?	4
Prepare Your Computer to Create Websites	5
Install Suitable Browsers.....	5
Install a Free HTML Text Editor	5
Create a Folder for Your First Web Page.....	6
The Basic Structure for Every Web Page.....	6
Enhancing the Structure.....	7
Create the Structure of Your First Web Page	9
Discussion of the First Six Lines of the HTML Code in Listing 1-1	11
Summary	12
■ Chapter 2: Create Your First Website and Add Hyperlinks	13
Create a Folder for the New Chapter.....	13
Let's Add Some Content to Your Web Page.....	13

Create a Home Page.....	15
Add a Second Page to the Website.....	16
Add Hyperlinks to the Two Pages	16
Explanation of the Code	18
Summary.....	18
■ Chapter 3: Styling the Website with CSS	19
How Do We Link HTML Pages to a CSS Style Sheet?	19
How Does the Link to the Style Sheet Work?	20
Create a Folder for the New Chapter.....	20
Creating a Simple CSS Style Sheet	22
Explanation of the Style Sheet <i>style.css</i>	23
How Much do you Need to Remember?	25
Summary.....	26
■ Chapter 4: Create Web Pages with Three Columns Using CSS	27
Create the Three-Column Page	27
Explanation of the Code.....	29
Create a Style Sheet for Producing Three columns.....	30
Explanation of the Amended CSS Code	30
Revise the CSS Code	31
Explanation of the Revised CSS Code	32
The Difference Between <i>id</i> and <i>class</i>	33
Summary.....	33
■ Chapter 5: Create Web Pages with Four Columns Using CSS	35
Create Web Pages with Four Columns	36
Explanation of the Code	38
Insert New Styles in the CSS Style Sheet.....	38
Explanation of the CSS Code	39

Create Two More Pages.....	39
The Importance of Forward Planning	40
Summary	41
■ Chapter 6: Add Pictures to Websites	43
The Problem with Some Older Browsers	44
Add Pictures to the Home Page.....	45
To Alter the Home Page Using a WYSIWYG Editor.....	45
To Alter the Home Page Using a Plain Text Editor.....	46
Explanation of the Code.....	47
Alter the CSS File to Style the New Home Page	47
Explanation of the CSS Code Alterations	48
Change the Heading in the Other Three Pages	48
Summary	49
■ Chapter 7: Enhancing the Website.....	51
Allowing Internet Explorer 8 to Understand Semantic Tags	51
Add the JavaScript and Some Hyperlink Tags That Can Be Styled	53
Explanation of the Code.....	54
Make a Minor Change to the CSS Style Sheet.....	55
Explanation of the CSS Code	56
A More Versatile Way of Setting Font Sizes.....	57
Summary	57
■ Chapter 8: Rollover Menu Buttons.....	59
Improving the Appearance of the Menu Hyperlinks	59
Explanation of the Code.....	62
Add the Rollover Feature.....	63
Create 3D Buttons with a Rollover Feature	63
Insert 3D-Colored Menu Buttons into a Web Page	65
Explanation of the Code.....	65
Summary	68

- **Chapter 9: Using Color Effectively 69**
 - Choosing Color Schemes..... 69
 - How Website Colors Are Produced 70
 - Understanding the CSS Color Codes 71
 - Tutorial: Adding More Color to our Website 72
 - Explanation of the Code..... 74
 - Using Color Pickers on WYSIWYG Text Editors..... 75
 - Tweaking Colors 77
 - Summary..... 78
- **Chapter 10: Screen Size and Resolution: Useful Templates: Rounded Corners..... 79**
 - Screen Sizes and Screen Resolutions 79
 - Other Monitor-Related Considerations 80
 - Will the Website Work on a Handheld Device? 80
 - Examining the Three Possible Layouts..... 81
 - The Advantages and Problems of Fixed-Width Layouts..... 81
 - The Advantages and Problems with Liquid Layouts 81
 - Semi-Liquid Layouts Provide the Best Compromise..... 81
 - Create a Template with a 3D Menu for Use in your Future Websites 82
 - Creating a Template with a Border with Rounded Corners..... 83
 - Creating Two Templates with Plain Menu Buttons 84
 - Summary..... 86
- **Chapter 11: A Vertical Menu with a Picture Gallery 87**
 - Adapting the Home Page..... 88
 - Change *page-2.html* to Create the About Us Page 89
 - Left-Align the Text in the About Us Page 91
 - Create the Location Page Using the File *page-4.html* 92
 - Creating the Gallery Page..... 93

Explanation of the Code.....	96
Adding the <i>figure</i> Style to the External Style Sheet.....	98
Summary	98
■ Chapter 12: A Horizontal Menu with an Enlarged Picture Gallery	99
Download and Install Four Templates with Horizontal Menus.....	99
Converting a Vertical Menu into a Horizontal Menu	100
Understanding the Horizontal Menu Templates.....	101
The Modified CSS Style for a Horizontal Menu with 3D Buttons	101
Explanation of the Code.....	102
The Modified CSS Style for a Horizontal Menu with Plain Buttons	103
Adding More Buttons to The Horizontal Menu	104
Tutorial: Taking Advantage of a Horizontal Menu.....	104
Tutorial	105
Explanation of the code for Listings 12-6 to 12-10	108
Summary.....	109
■ Chapter 13: More about Website Images: Create a New Appearance with Tiles	111
Image File Formats Suitable for Websites.....	111
Preparing Pictures for a Website	114
Creating a New Appearance Using Background Tiles	114
Tweaking the CSS Style Sheet	115
Explanation of the Code.....	116
Altering the home page <i>index.html</i>	118
Explanation of the Code Changes	119
Creating Pictures Suitable for the Header Section	120
Summary.....	120

- **Chapter 14: Vertical and Horizontal Menus on the Same Page: Colored Columns..... 121**
 - Tweak the Home Page *index.html*..... 122
 - Explanation of the Code..... 123
 - Change the Other Three Pages..... 124
 - Tweak the Style Sheet 125
 - Explanation of the Code..... 126
 - Create Four Templates Containing Both Horizontal and Vertical Menus..... 127
 - Summary..... 129
- **Chapter 15: Create Tiles: Use Two New Tiles: Float-Drop and the Box Model 131**
 - More About Using Tiles in the Background of a Website..... 131
 - Create Your Own Tiles for Backgrounds 132
 - Types of Tile..... 132
 - Create a Different Appearance Using Two New Tiles..... 134
 - Float-Drop 136
 - The Box Model..... 136
 - Summary..... 137
- **Chapter 16: Create Tables for Data..... 139**
 - Create a Simple Two-Column Table..... 139
 - Explanation of the Code..... 140
 - Inserting and Deleting a Table or Rows and Columns in a WYSIWYG Editor..... 141
 - Explore the Default Double Border Style 141
 - Placing a Four-Column Table within a Web Page 142
 - Explanation of the HTML Code..... 145
 - Explanation of the Internal Style..... 145
 - Summary..... 145
- **Chapter 17: The Secret of Attractive and Useful Websites 147**
 - Basic Rules for an Attractive Website..... 147
 - The Navigation Menu Must Be Prominent 148

The Use and Misuse of Text	149
Simple Is Good	150
That All-Important Home Page	150
State the Purpose of the Website Clearly and Concisely	150
Avoid Gimmicks	150
User Psychology: How Surfers Use Websites	151
User Experience (UX)	151
Slow Loading Websites Frustrate the User	151
Auto-start Audios, Videos, and Slide Shows	151
Other Annoyances	152
Websites Should Be Useful	152
Create a Web Page with an Encoded Email Address	152
Explanation of the <i>escramble</i> JavaScript	154
Summary	155
■ Chapter 18: Design a Feedback Form	157
Design a Feedback Form	157
Explanation of the Changes	160
Add an Internal Style to Position the Form Elements	161
Explanation of the File's Internal Style	162
Add the Form Code to the <i>contact.html</i> Page	163
Explanation of the Form Code	165
Summary	167
■ Chapter 19: Search Engine Optimization	169
How Search Engines Work	169
What Search Engines Look For	170
Choosing Keywords and Phrases	170
Popularity of a Product or Service and Its Keywords	171
The Importance of Keywords in the Title Tag	171
The Meta Tag/Keywords Controversy	171
Keywords in Headings	172

Keywords/Phrases Must Be Present in the Body of the Page	172
External Links	172
Well-Designed Internal Links.....	173
Restrictions on Excessive Repetition	173
Things You Should Never Do	174
A Web Page Containing No Search Engine Optimization.....	175
Explanation of the SEO Faults in Listing 19-1.....	176
Let People Know That Your Website Exists.....	177
Beware of False Promises.....	177
Optimize the Spring Garden Home Page for Search Engines.....	177
Explanation of the Code.....	180
Summary.....	182
■ Chapter 20: Positioning Elements on a Web Page	183
Absolute Positioning.....	183
Change the Vertical Position of the Header Text.....	185
Relatively Positioning an Image	185
An Experiment Using Relative Positioning.....	186
Positioning Images Across the Wrapper Boundary.....	187
Explanation of the Code.....	189
Positioning Images Next to Text	189
The <clear> Property Applied to Floated Elements	191
Positioning by Using the Margin Property	192
Summary	193
■ Chapter 21: Save Time and Reduce Tedium.....	195
The Time-Saving Code	195
A Note About PHP Code	196
Using a Server	197
1. Use an Existing Website Server on a Host.....	197
2. Install a Server for Free on Your Computer	197

Download and Install XAMPP for Windows.....	197
Starting XAMPP	199
Closing XAMPP	200
The XAMPP Security Console	200
The <i>htdocs</i> Folder.....	202
Testing PHP Files in XAMPP.....	203
Using the PHP <i>include</i> Command.....	204
Create the External Files	205
Create the External File for the Header	205
Create the External File for the Horizontal Menu.....	206
Create the External File for the Vertical Menu	206
Create the External File for the Footer.....	207
Inserting the PHP <i>include</i> Command to Replace the Code for the Four Included Elements.....	208
View the File Using the XAMPP Server on Your Computer.....	209
To use the Server on a Remote Host	211
Summary.....	211
■ Chapter 22: More on Using PHP <i>include</i>.....	213
Using the PHP <i>include</i> Command.....	214
Change the External Footer File (<i>footer.html</i>).....	214
Make Copies of Existing Pages to Create New Pages.....	216
Create a Template and a New Page (<i>spring-bulbs.php</i>)	216
Create the Page <i>spring-seeds.php</i>	217
Create the Page <i>spring-plants.php</i>	218
Create the Page <i>about.php</i>	218
Create the Page Named <i>location.php</i>	219
Create the Page Named <i>terms.php</i>	220
Modify the Page Named <i>contact.php</i>	220
Summary.....	222

■ Chapter 23: Receive Emails from a <i>Contact Us</i> Page.....	223
Examine the Code of the Downloaded Page <i>form-handler-typical.php</i>	225
Explanation of the Typical Form-Handler Code.....	227
Validating User Input	233
Create the Message Pages for the Project.....	234
Create the Style Sheet for the <i>thank-you</i> Page and All the <i>error message</i> Pages	234
Create the Thank-You Page	234
Create the Missing Essentials Error Message (<i>error.html</i>) Using the Code in Listing 23-4.....	235
Create the Email Error Message Using the Code in Listing 23-5.....	235
Create a Check Box Error Message (<i>boxerror.html</i>) Using Code in Listing 23-6	236
Create the Error Message to Prevent URLs Being Entered	236
Testing the Form-Handler and the Messages	237
Summary	237
■ Chapter 24: Add Slideshows and Videos	239
Using Other People’s Multimedia Clips	239
The Benefits of Using a Slideshow or a Video	240
What to Avoid.....	240
Good Housekeeping Tips	241
Add a Slideshow.....	241
The BarelyFitz Designs Slideshow.....	241
Test the Slideshow	244
Other Ways of Creating a Slideshow.....	245
Adding Video to a Web Page.....	245
Yesterday’s Video Formats.....	245
Today’s Video Formats	245
Explanation of the Code.....	246
Converting File Formats	247

How to Embed the HTML5 Video Code in a Web Page.....	247
Using YouTube	249
Signing up for a YouTube Account	249
To Host a Video with YouTube	250
Summary	250
■ Chapter 25: Create a Tab Menu	251
Examine the Code for the Home Page.....	253
Explanation of the Code.....	254
Other Items Also Change from Page to Page.....	254
Examine the Code for the CSS File.....	255
Explanation of the Code.....	256
Create Two New Tabs	257
Create Two New Pages.....	257
Summary	259
■ Chapter 26: Designing a Drop-Down Menu	261
Why Drop-Down Menus may not Suit all Types of Websites	261
Planning the Menu	262
Explanation of the Code.....	265
Showing and Hiding the Drop-Down Submenus	266
Explanation of the Code.....	267
Create a New Page to See how the Drop-Down Menu Works.....	267
Create the Publications Page	268
Summary	269
■ Chapter 27: Drop Shadows	271
Drop Shadows for Images and Websites with Sharp Corners.....	272
Explanation of the Code.....	273
Drop Shadows for a Website with Rounded Corners.....	274

Add a Drop Shadow and a White Border to an Image	275
Explanation of the Code.....	276
Add a Drop Shadow to Text	277
Explanation of the Code.....	278
Shadows on Four Sides of an HTML Element.....	278
Create a Four-Sided Drop Shadow	279
Summary.....	280
■ Chapter 28: User Name and Password for a Member's Page	281
Create the Login Page	282
Explanation of the Form Code	284
Styling the Login Page.....	284
Create a Members' Page.....	284
Create a Login-Handler	286
Explanation of the Code.....	287
Summary	288
■ Chapter 29: Create a Printable Order Form	289
Add an Order Form Button to the Vertical Menu.....	289
View and Modify the Order Form.....	290
Explanation of the New Code.....	295
Explanation of the Unfamiliar Code in Listing 29-1	295
Examine the Style Sheet for the Order Form (<i>order.css</i>)	297
Create the Style for the Printed Version of the Form (<i>print-order.css</i>)	298
Explanation of the Styling Code for the Printed Version of the Order Form	300
Summary	300
■ Chapter 30: Add a Search Field to Your Website	301
Bing Search Field	302
Yahoo! Search Field.....	303

Google Search Field	304
Testing the Google Search Field on One of My Websites.....	306
Summary	307
■ Chapter 31: Styled Bullet Points.....	309
The Problem with Default Bullets.....	309
Create a Test Page.....	310
Reduce the Gaps Caused by the Default Bullets	312
Explanation of the Code.....	313
Add Bold Headings to the Bulleted Paragraphs.....	313
More Bullet Styles Using CSS.....	314
Summary	316
■ Chapter 32: Indicating Which Horizontal Menu Button Has Been Clicked	317
View the Pages in the Downloaded Folder	317
Tweak the Menu So That It Can Be Styled to Indicate the Selected Menu Button	318
Make a Small Change to the Home Page Styling to Match the Revised Menu.....	319
Add the Indicator Style to the Other Pages	320
Emphasize the Page Indicator	322
Summary	323
■ Chapter 33: Indicating Which Vertical Menu Button Has Been Clicked	325
View the Pages in the Downloaded Files	325
Make Minor Changes to the Menu to Indicate That a Particular Button Has Been Clicked	328
Change the Menu on the Other Pages.....	329
Emphasize the Menu's Page Indicator	330
Summary	331

■ **Chapter 34: Creating Multi-row Menus and Picture Galleries..... 333**

 Create a Double Row of Menu Buttons..... 334

 Explanation of the Code..... 335

 The CSS Code 336

 Explanation of the Code..... 337

 To Accommodate an Odd Number of Buttons..... 337

 Create a Gallery with Four Rows of Pictures..... 338

 Explanation of the Code..... 340

 Summary..... 342

■ **Chapter 35: Building Responsive Websites for Mobile Devices Part 1 343**

 Definitions 344

 Is It Worth All the Effort?..... 345

 Different Websites? Or Different Style Sheets?..... 345

 The Responsive Solution 346

 The Viewport Statement..... 346

 Media Queries and the Responsive CSS Style Sheet 347

 Explanation of the Demonstration Media Query CSS Code..... 348

 Responsive Websites Use Proportional Dimensions..... 349

 Using Proportional Dimensions for the Elements on RWD Pages..... 349

 Responsive Images 350

 The Responsive Navigation Menu 351

 Explanation of the Styles (*style.css*)..... 355

 View the Other Responsive Pages..... 356

 Testing Responsive Websites 356

 Summary..... 357

■ Chapter 36: Building Responsive Websites for Mobile Devices Part 2	359
Adding a Drop-Down Feature to the Menu Buttons.....	359
Explanation of the Code.....	361
Creating a Two-Column Responsive Web Page	362
Explanation of the Code.....	365
Creating a Three-Column Responsive Web Page	366
Explanation of the Code.....	367
Explanation of the Code.....	368
Creating a Four-Column Responsive Web Page	369
Summary.....	370
■ Chapter 37: Building Responsive Websites for Mobile Devices Part 3	371
A Collapsible Responsive Menu	371
Explanation of the HTML Code.....	373
Explanation of the CSS Code	375
Adding a Header.....	378
Summary.....	380
■ Chapter 38: Building Responsive Websites for Mobile Devices Part 4	381
Summarizing the Use of Media Queries in CSS Style Sheets.....	381
View the Responsive Pages in This Chapter's Website	382
Brief Descriptions of the New Pages.....	383
The Supplementary Style Sheet Links.....	383
The Supplementary Style Sheets	384
Adding New Menu Buttons and Extra Drop-Down Items.....	386
Bullet Points and the Unordered List Problem.....	387
Responsive Videos.....	389
Responsive YouTube/Vimeo Videos.....	390
Explanation of the Code for <i>fitvids-test.html</i>	392

Additional Notes on Responsive Websites.....	393
Content Strategy.....	393
Fonts.....	393
Accelerated Mobile Pages (AMP).....	394
JavaScript.....	394
Other Avoidable Features That Cause Slow Loading	394
Items That Are Not Under the Control of the Designer.....	394
Summary.....	395
■ Chapter 39: Avoiding Some of the Pitfalls of a CMS Website	397
Pitfalls That Can Be Avoided.....	397
An Example of an Acceptable CMS Design.....	398
CMS Pitfalls That Are Very Difficult or Impossible to Avoid	400
Summary.....	401
■ Chapter 40: Go Live and Validate Your Website	403
Choosing a Domain Name	403
Choosing a Host	404
Download Your Free FTP Client	404
Allow the FTP Client to Connect to a Host	404
Getting to Know Your FTP Client.....	409
Connect to the Host with Your Free FTP Client	410
The Quickconnect Bar	410
Storing the FTP Details	410
Using the Advanced Tab to Create Automatic Connections	412
Set the Date Format	413
Uploading Folders and Files to the Remote Host.....	414
Downloading Folders and Files from the Host to Your Computer	416
Working within Folders.....	416
Validate your Web Pages.....	416
The DOCTYPE.....	417
Accessing the W3C Validator	417

Using the W3C Validator.....	417
Some Typical Reports on HTML5 Errors and How to Fix Them.....	420
Byte Order Mark Found	421
Validate the CSS.....	425
Sorry! We found the following errors (1)	426
Warnings (1)	426
Vendor-Specific CSS Errors	426
Embed the HTML5 Logo.....	426
The Solution for a Verifiable HTML5 Logo.....	427
Summary.....	428
■ Chapter 41: Quick Reference: Graphics Programs: Resources.....	429
A Quick Reference for HTML, CSS, and PHP.....	429
Web Page Structure.....	429
HTML Tags	430
Comments	431
The Main Structural Tags.....	431
Background Color Styles for Rollover 3D Menu Buttons	433
Elements within the Content Area	435
An Element Below the Content Area.....	436
Styling the Text with HTML	436
CSS Styles Sheets	437
PHP	438
Graphics Programs for Optimizing Pictures	438
Precautions when Downloading Free Programs.....	438
Which Graphics Program?	439
File Minimizer Pictures.....	439
Resources.....	442
Summary.....	442

■ **Chapter 42: Installing and Using Text Editors 443**

Which Free WYSIWYG Program Is Best?..... 443

 The WYSIWYG View in Expression Web May Look Odd 444

 Be Cautious When Downloading Text Editors 445

Download, Install, and Configure MS Expression Web 4 (Free) 445

 Configure the Toolbars in Expression Web..... 446

Download, Install, and Configure Blue Griffon (Free)..... 452

 Close Down Blue Griffon..... 454

Non-WYSIWYG Text Editors..... 461

 Downloading and Installing TextEdit..... 461

 Downloading and Installing Komodo Edit (Free)..... 461

 Downloading and Installing NoteTab Light (Free) 461

 Downloading and Installing Notepad++ (Free)..... 462

Summary 462

Index..... 463

About the Author

Adrian W. West resigned as a chartered engineer to become the UK director of a correspondence school. He has been teaching in one form or another since 1982. He introduced computers into his workplace in 1987 and taught the staff how to use them. For four years, he taught computer skills to undergraduates at a college in Cheshire in the United Kingdom.

Adrian lives in Colyton, a town in Devon, England, and for the past 17 years, he has designed and produced websites for national and local businesses and charities. For a time, he also served as a computer technician and teacher to his community. Then he decided to concentrate on his favorite occupation: designing websites. To avoid disappointing his former clients, he launched a free computer-help website at <http://www.colycomputerhelp.co.uk>. He also cofounded with Moira Wight the web design service <http://www.amwebdesign.co.uk>.

Adrian writes monthly computer-help articles for two local magazines. He is the author of two earlier Apress publications: *Practical HTML5 Projects* and *Practical PHP and MySQL. A simplified approach*.

About the Technical Reviewer

Massimo Nardone has more than 22 years of experience in Security, Web/Mobile development, Cloud, and IT Architecture. His true IT passions are Security and Android.

He has been programming and teaching how to program with Android, Perl, PHP, Java, VB, Python, C/C++, and MySQL for more than 20 years. He holds a Master of Science degree in Computing Science from the University of Salerno, Italy.

Massimo has worked as a Project Manager, Software Engineer, Research Engineer, Chief Security Architect, Information Security Manager, PCI/SCADA Auditor, and Senior Lead IT Security/Cloud/SCADA Architect for many years. His technical skills include the following: Security, Android, Cloud, Java, MySQL, Drupal, Cobol, Perl, Web and Mobile development, MongoDB, D3, Joomla, Couchbase, C/C++, WebGL, Python, Pro Rails, Django CMS, Jekyll, Scratch, etc.

He currently works as Chief Information Security Officer (CISO) for Cargotec Oyj. He worked as a visiting lecturer and supervisor for exercises at the Networking Laboratory of the Helsinki University of Technology (Aalto University). He holds four international patents (PKI, SIP, SAML, and Proxy areas).

Massimo has reviewed more than 40 IT books for different publishing companies and is the coauthor of *Pro Android Games* (Apress, 2015).

Acknowledgments

I thank my wife, Janice, for her love and support; for taking over my share of the chores so that I could concentrate on this book; for her encouragement; and for putting up with my absence as I hunched over the keyboard. I could never have managed without her meticulous proof reading, which she patiently repeated several times per chapter as each editorial stage was reached.

My thanks also go to the magnificent team at Apress: Mark Renfrow for his encouragement and for his advice on the layout and content of the chapters; Nancy Chen, who coordinated everybody and ensured that I sent chapters and files on. I thank Massimo Nardone, the technical reviewer, who checked my code and suggested several useful resources for inclusion in the book.

My special thanks go to my students who tested the chapters and made helpful suggestions. In particular I thank my star pupil Moira Wight who showed so much promise that she is now my partner in our joint venture <http://www.amwebdesign.co.uk>. Moira helped me test and improve every chapter in this book. My thanks go to all the people in Internet forums who helped me and replied to my queries; and to all those who placed information on the Internet from which I learned so much.

I express my gratitude to the Executive Project Manager DulcyNirmala, Chellappa and her team for their patience and diligence in correcting layout and formatting errors that crept into the book during the editing process.

—Adrian W. West

Introduction

The Aim of This Book

This book was written to provide a superior alternative to the increasing use of Content Management Systems (CMS) programs (WordPress, Joomla, Drupal, Go Daddy, etc.). CMS is a children's coloring book style of program that has resulted in millions of dreadful websites produced by amateurs who have no training in design and web-user psychology. The aim of this book is to teach the readers those essential skills in addition to teaching them how to use HTML and CSS code.

Producers of CMS programs promote their programs by implying that HTML and CSS are difficult to learn. However, HTML and CSS are not difficult. As languages go, they have a very small vocabulary, and better still, all the words are either English words or abbreviations of English words. This book will demonstrate that coding is not difficult and will also show you how to use HTML and CSS to produce unique, fast-loading, top-quality websites.

This book will teach you HTML and CSS using short, easily assimilated steps. *Practical Website Design for Absolute Beginners* avoids a tedious theoretical approach by introducing small amounts of new code in each short chapter. HTML and CSS are essential if you wish to take full control of a website's design and maintenance.

I have provided the reader with a wide range of attractive templates for web pages and websites. The great majority of chapters build on the templates created in previous chapters. The reader has a practical project to complete in almost all the chapters and is taught to produce practical web pages right from the start. The readers are guided to produce the majority of the templates themselves by following the instructions in each chapter. As a result they will have an intimate knowledge of the code in the templates and will have complete control over the quality and content of their sites.

The Origin of This Book

When teaching web design to my students, I prepared worksheets and emailed them so that they could have hands-on experience. I found that students learned faster if the chapters were short, they felt a real sense of achievement as they completed each small project. This book is a collection of those short worksheets so that readers can learn without the aid of an online human tutor.

Who Is This Book for?

This book will help people who would like to do the following:

- Design and produce their own unique, fast-loading websites
- Avoid using complex slow-loading JavaScript by using HTML5 and CMS
- Have complete control over their websites

- Maintain a website that they already own
- Upgrade from HTML 4 to HTML 5
- Escape from the restraints, disadvantages, and uniformity of CMS-based websites

In this book, you will not only learn to use HTML and CSS, but you will be taught vital features that you will not learn from CMS programs. For instance, you will be taught the following:

- The techniques for creating an attractive and useful website
- How to ensure that your site will be found by search engines
- The importance of focus and the composition of web pages
- The effective use of color
- How to create clear, intuitive navigation between the pages of a website

The Level of Skill Required

If you can type an email or a letter, you can create an HTML web page and a CSS style sheet. However, you must possess the basic skills for using Windows or Mac operating systems so that you know how to the following:

- Create a new folder
- Navigate from folder to folder
- Put a shortcut icon on your desktop
- Download and install a free program
- Save a copy of a file using *Save As...*
- Right-click and rename a file or folder
- Copy and paste using short-cut keys
- Be able to download folders from the book's website and unzip them

The Structure of This Book

Practical Website Design for Absolute Beginners is a project-oriented book that introduces small amounts of new code in short practical chapters. Each chapter builds on the templates created in the previous chapters. The many web pages and website templates can be easily adapted by readers for their own websites.

Chapter 1: This chapter explains the considerable advantages of using HTML and CSS rather than a coloring-book content management system (CMS). The next sections give advice on using free programs to enable you to produce the book's websites and adaptable templates. You will learn about the standard structure for a web page and then produce your first web page structure, which will form the basis of all the subsequent projects in the book.

Chapter 2: The previous chapter only produced the structure for a web page, so this chapter adds some content to that structure. The reader will add an extra page and learn how to link the two pages by means of hyperlinks.

Chapter 3: This chapter applies CSS styling (Cascading Style Sheets) to the previously created template. The chapter demonstrates how the appearance of all the pages in a whole website can be changed by an amendment to the single CSS file. The CSS amendment cascades into all the pages in the website.

Chapter 4: In this chapter you will create a template for a typical three-column web page.

Chapter 5: In this chapter you will create a template for a typical four-column web page.

Chapter 6: Pictures are an important part of a website; they must be optimized so that the browser downloads the picture to the screen quickly. This chapter adds pictorial content to your website template.

Chapter 7: This chapter further enhances the web pages and templates that you produced in earlier chapters. You will add style to the menu hyperlinks to produce attractive menu buttons. It also introduces a more versatile way of setting font sizes.

Chapter 8: Although they worked well, the menus in previous chapters were rather plain. In this chapter you will use CSS to improve the menu buttons' functions and appearance.

Chapter 9: The proper use of color can affect the user's decision whether to stay with your website or switch to a more appealing website. Color can affect the mood of a website and also reflect the purpose of the site.

Chapter 10: This chapter discusses the effect of various screen sizes and resolutions. You will create useful templates and learn how to add rounded corners using CSS styling.

Chapter 11: You will produce another practical website that will include a picture gallery. The site is an adaptation of one of the templates that you created in Chapter 10.

Chapter 12: This chapter uses template files that you can download within a zip file; this will save you hours of coding. I will describe how a vertical menu block is converted into a horizontal row of buttons. The template files are adaptations of the templates that you created in Chapter 10; however, the horizontal menu allows you to create a gallery with more pictures per row.

Chapter 13: This chapter discusses image file formats suitable for websites. You will learn how to create a different appearance using background tiles. Pictures suitable for the header section are discussed.

Chapter 14: If a website has many pages, you will need more menu buttons. You will learn how to create an additional menu so that you have both vertical and horizontal menus. You will also discover how to add color to columns.

Chapter 15: A previous chapter demonstrated how tiles can be used for backgrounds in websites. This chapter will help you to create your own background tiles to match the theme of your website. You will also learn about float-drop and the concept of box models.

Chapter 16: You will learn how to create tables for presenting data by placing a four-column table within a web page.

Chapter 17: Learn the secret of creating attractive and useful websites. You will also be taught how to display spam-proof live-linked email addresses.

Chapter 18: Most websites need a feedback form so that users can interact with the owner of the website. This chapter describes how to create a typical feedback form.

Chapter 19: When you launch a website on the World Wide Web, you hope it will be found as soon as possible by search engines such as Bing, Yahoo! and Google. This chapter teaches you about search engine optimization by means of a practical search engine optimization project.

Chapter 20: Elements can be positioned accurately on a web page by using CSS styling. A practical project will teach you about the two types of positioning: Absolute positioning and Relative positioning. You will be taught how to position images alongside paragraphs of text.

Chapter 21: Save Time and Reduce Tedium. This is the first chapter in the advanced section of the book. As the number of pages in your website grows, the task of adding the header, footer, and extra menu buttons to each new page becomes tedious. By using the shortcut described in this chapter, you will learn how to eliminate the tedium when altering the header, or the menu, or the footer. These alterations can be achieved on all the site's pages by changing only one file. The shortcut consists of a little piece of PHP code. The chapter also describes how to install a server in your computer for testing your PHP shortcuts.

Chapter 22: The previous chapter explained the advantages of using the PHP *include* command. In this chapter the considerable advantages will be demonstrated in greater detail. In Chapter 21 you only used the *include* commands in the home page *index.php*. In this chapter you will create new pages and use the *include* commands in every page of an eight-page website. All the new pages will be adaptations of existing pages.

Chapter 23: By using a PHP form-handler to process a feedback-form such as the *contact.php* created in Chapter 18, you will learn how to provide a spam-free way of receiving emails. Also the information sent by the user will be filtered to remove malicious content. The content of the pages will be deliberately brief to save time and space, and you will utilize all the pages from the previous chapter.

Chapter 24: This chapter will teach you how to add a slide show and a video to a web page.

Chapter 25: Tab menus are sometimes used in smaller websites with few pages, say a maximum of 12 pages. Tab menus are not particularly suitable for large sites because of the limited number of tabs that can be placed across a page. In this chapter you will view a ready-made website with six tabs so that you can discover how the tabs work.

Chapter 26: In the previous chapter we explored tabbed menus; tabs can also be created as drop-down menus. However, drop-down menus can be rather unfriendly because the page links are hidden until the user hovers over the drop-down tab. I have included a CSS drop-down menu for the sake of completeness. However, experienced users will not be afraid of exploring and using drop-down menus.

Chapter 27: CSS3 introduced many useful features and the CSS3 drop shadow technique simplifies a process that was previously rather complex. Therefore this chapter deals only with drop shadows for later versions of Internet Explorer, and of course with all the other modern browsers that are compatible with CSS3. The chapter also describes a drop shadow that wraps around a whole website page.

Chapter 28: This project is suitable for a small society or club. It hides certain information from the general public but makes it available to members (or a restricted group of members such as a committee). We achieve this by means of a user name and password. The only minor drawback is that each member must be given the same password and user name. However, the user name and password can be changed at any time if security becomes an issue. If a larger society or club would prefer a unique identification for each member, the members would register their user names and passwords via a registration page; their identification details would be stored in a database. Databases are beyond the scope of this book but are described in my recent book: *Practical PHP and MySQL Website Databases – A Simplified Approach* (Apress.com.)

Chapter 29: This chapter describes a printable form that will enable users to order goods or services and pay by check. The form has the advantage that the user can fill it in on-screen using a keyboard and mouse (except for the signature). The form prints out using only black ink to save the user's expensive color cartridge. Also some elements on the order form (as displayed on the screen) are automatically omitted from the printed form because they are not necessary (e.g., the header image, the footer, and any up-arrows).

Chapter 30: Larger websites can benefit from an in-built search facility; the chapter describes how you can achieve this.

Chapter 31: This chapter describes how bullet points can be styled using CSS.

Chapter 32: This chapter shows you how to indicate that a particular horizontal menu button has been clicked.

Chapter 33: This chapter shows you how to indicate that a particular vertical menu button has been clicked.

Chapter 34: A single row of horizontal menu buttons can limit the number of buttons to about 8 or 10. This chapter describes how a double row of menu buttons can provide up to 20 buttons. This chapter also demonstrates a multi-row picture gallery and a method of payment by PayPal and credit/debit card.

Chapter 35: This chapter will show you how to build responsive websites for mobile devices.

Chapter 36: This chapter will show you how to add extra features to the responsive website design that was described in Chapter 35.

Chapter 37: This chapter shows you how to build a collapsible responsive menu for use on websites for mobile devices.

Chapter 38: This chapter summarizes the use of media queries in CSS style sheets.

Chapter 39: This chapter will show you how to avoid some of the most common pitfalls of creating CMS websites.

Chapter 40: You will learn how to choose a domain name and a host in this chapter.

Chapter 41: This chapter provides a quick reference for HTML, CSS and PHP, graphics programs for optimizing pictures, and useful resources.

Chapter 42: This chapter will teach you how to install and use text editors.

Index

Minimum Use of Scripts

I have reduced the use of JavaScript to an absolute minimum. Most JavaScript functions can now be replaced by CSS2 or CSS3, which simplifies the task of producing websites from scratch. A small JavaScript function is used in a few projects to enable Internet Explorer 8 to understand the new HTML5 semantic tags. In Chapters 21, 22, and 23 a tiny PHP function is used to create a shortcut that greatly reduces the tedium of updating larger websites. Chapter 28 demonstrates the interactive aspect of PHP.

Conventions Used in This Book

The terms *code* and *listing* are used interchangeably. The words *tags* and *markup* are also synonymous. The words *client* and *website owner* are used to mean a person or organization that has commissioned you to produce a website.

In the code listings, **bold type** is used to highlight the items under discussion.

Using the Book's Code and Templates

Most of the worked examples are practical templates that readers can view and download from the companion website. Readers can easily and quickly adapt these examples for their own use. No permission is required for using the examples or the templates in a website. Permission will be required if you include the examples in commercial media, that is, printed matter or a CD. If you use the book's examples in a website offering instruction on web design, permission is required and you will be asked to acknowledge where you found the code. The attribution should give the source, as follows: *Practical Website Design for Absolute Beginners* by Adrian W. West. Copyright 2016 Adrian W. West. Published by Apress Media, LLC. ISBN 978-1-4842-1992-8.

If you think that your particular use of the book's markup is not covered by this paragraph, please contact permissions@apress.com.

Downloading the Source Code

The source code for each chapter is available from the book's website at Apress.com. The downloads include a folder containing the completed files for each chapter, a selection of templates, an occasional PDF document for demonstrating the colors discussed in the chapter, and all the images required in the projects. To unzip the files you will need 7-zip (free) or WinZip.

CHAPTER 1



Creating Websites

This chapter explains the considerable advantages of using HTML and CSS instead of a paint-by-numbers content management system (CMS) such as WordPress, Joomla, and Google web design. The chapter also contains advice on using free programs to enable you to produce the book's website projects. You will learn that all websites have a standard basic structure for web pages. You will then produce your first web page structure that will form the basis of many subsequent projects and most of the adaptable templates in the book.

This chapter contains the following sections:

- Definitions
- The two website creation methods
- The advantages of using code rather than CMS
- Who's afraid of HTML?
- But what about CSS?
- Prepare your computer to create websites
- Install suitable browsers
- Install a free HTML texteditor
- Create a folder for your first web page
- The basic structure for every web page
- Enhancing the structure
- Create the structure of your first web page
- Summary

Definitions

Browser: A program like Internet Explorer, Edge, Mozilla Firefox, Chrome, Opera, or Safari.

User: A person viewing your website.

HTML: The code language used to produce website pages. The code consists of logical English words or their abbreviations.

Electronic supplementary material The online version of this chapter (doi:[10.1007/978-1-4842-1993-5_1](https://doi.org/10.1007/978-1-4842-1993-5_1)) contains supplementary material, which is available to authorized users.

CSS: The code that formats and styles website pages. The code consists of logical English words or their abbreviations.

Websites can be produced and maintained by using one of the following two methods.

The Two Website Creation Methods

- Using HTML code and CSS code
- Using a Content Management System (CMS)

This book deals only with the first method, which gives you total control over your websites.

In the second method, CMS kits are like a child's coloring book: you add colors and text to ready-made templates. In effect the templates control you and your website.

■ **Tip** If you wish to skip the next sections and get on with learning HTML, jump down to the item labeled "Prepare your Computer to Create Websites." You could even use Microsoft's Notepad for the project in this chapter, but be sure to use one of the recommended text editors for future chapters.

The Advantages of Using Code Rather Than CMS

"DIY" (CMS) programs have helped millions of unskilled people to produce millions of dreadful websites. It is possible to produce an attractive and useful website with these kits, but the designer would need training in design and user psychology; tips are provided for this purpose in Chapter 39. Many so-called web design companies are now charging huge sums of money for building websites even though they are using one of the CMS platforms. This is crazy – anyone can use CMS coloring-book programs for free, so why pay someone to do it?

Content Management Systems stress the fact that you don't need to learn HTML and CSS; this promotes their programs and implies that HTML and CSS are difficult to learn. These languages are not difficult; they use plain English words and abbreviations of English words. More importantly, a good website is MUCH more than HTML and CSS. A web designer needs to know many other aspects of a good website as listed below. These aspects are built into this book, and you will assimilate them in short easy steps as you work through the chapters.

CMS kits do NOT provide you with the following vital aspects of good websites:

- How to focus the user's attention on key features of your web pages
- How to provide clear, search engine friendly navigation menus
- How to choose the best typography for your web pages
- How to optimize your websites so that search engines can find them
- How to optimize images and code for fast loading pages
- How to choose attractive color schemes
- How to ensure that your website is attractive and how to avoid irritating your visitors
- How to understand the way users explore websites (web-user psychology)

- How to make your websites useful
- How to include email addresses and “Contact Us” pages without attracting spam
- How to fine-tune your website, you can’t do this without HTML or CSS

This book and most other HTML/CMS manuals do cover these vital topics.

The great majority of CMS websites are produced by untrained amateurs, and the sites usually have very long home pages with all the content dotted around in a random fashion (lack of focus). This not only bewilders the visitor, but forces them to scroll down a mile-long page in the hope of finding what they are searching for. CMS kits include huge amounts of code to compensate for the user’s lack of HTML and CSS knowledge. CMS pages load slowly because each page contains 6 to 10 times more code than an equivalent professionally coded site. This extra code is necessary to convert the user’s text input and pictures into HTML and CSS.

The great majority of CMS pages use JavaScript menus; search engines cannot penetrate past the home page because search engines refuse to read JavaScript menus. Therefore search engines think the CMS site has only one page and it is ranked low because of this.

With CMS kits, if you want to tweak the web pages you need to learn HTML, CSS, PHP, and JavaScript. Unfortunately, if you try to do this you may find that the CMS program uses an idiosyncratic version of PHP that does not conform to the official standard.

When using a CMS program you will be forced to use the templates that thousands of others are using. By following the chapters in this book you will produce your own unique templates that you can modify to suit any website.

All CMS sites load an enormous amount of baggage into your host’s root folder. For example, a basic 6-page website using HTML5 and CSS2 results in only 2 folders and 6 files. Using a CMS package for the same website results in 17 folders with an average of 30 files in each, plus dozens of PHP and JavaScript files and several additional files for administering the website. Some CMS programs put a page or two of CSS on EVERY page of the website; this makes a mockery of CSS, which was designed to be a single file linked to each page.

If you need to move a CMS site to another web master, you may have difficulty finding someone willing to take on the learning curve necessary to grapple with the complexities of fine-tuning a CMS website.

Encouraging beginners to use a CMS program is like saying to someone, “So, you’re going to France on business for a year? Don’t bother to learn French, just take a translator with you.” Websites that can be designed online can be even more restrictive. These are heavily JavaScript based, resulting in even less designer control.

You will never be in full control of your web design process unless you learn how to use some HTML and CSS. If you wish to produce unique, lean, clean, easily managed websites, the only way to do this is to use HTML and CSS.

■ **Tip** If you decide against learning HTML and CSS, you can avoid some of the pitfalls of CMS websites by reading and absorbing Chapter 9 (The Effective Use of Color), Chapter 17 (Create Attractive and Useful Websites), Chapter 19 (Search Engine Optimization), and Chapter 39 (Avoiding Some of the Pitfalls of a CMS Website).

By following the advice in these chapters you may avoid launching another poorly designed website. However the CMS site will be slow to load because it will contain a huge amount of JavaScript.

Who's Afraid of HTML?

CMS kits state this: “No need to know any HTML code.” This implies that HTML and CSS are difficult to learn, which of course promotes the CMS programs. However, the truth is rather different. HTML is not so difficult; also the CMS programs don't tell the prospective user that the CMS kit must also be mastered in order to use them. The time spent learning to use a CMS kit could be spent more profitably by learning HTML and CSS; these languages are used in every website including CMS websites.

HTML was invented by Sir Tim Berners-Lee in the late 1980s. HTML stands for Hypertext Markup Language. This may sound very technical, but hypertexts are simply links, a way of moving from page to page in a website.

The HT part of HTML could be replaced by the word hyperlink. A hypertext or hyperlink is a piece of code that can be clicked to open a new page of the website. A group of hyperlinks on a web page is called a menu, and the menu allows users to choose which pages to view. Using the menu to move between web pages is called navigating.

Figure 1-1 shows some typical hyperlinks.

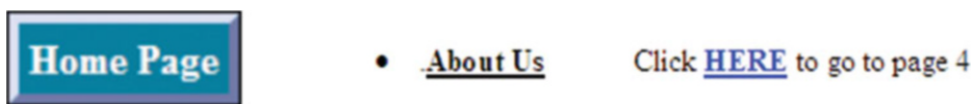


Figure 1-1. Three typical hyperlinks

The ML in HTML means “Markup Language.” A web page consists of English words containing markup that tells a browser (such as Internet Explorer) what to do. Take this code for instance: `<p>Some web text</p>`, the items in angle brackets are markup tags, and they don't display on the screen, but in this example they tell the browser to display the words “Some web text.” The HTML language is not a foreign language; it is English (simply because the inventor of HTML is British). In other words, the code uses logical English words or abbreviations of English words. For instance, `body` means the main content of a page, `table` means table, `div` is used for the division or section of a page, `p` means paragraph, `img` means image, and so on. Once you begin to use these simple English HTML words they will become so familiar that, with practice, you will be able to write HTML code as easily as you now write English prose. A web page is a plain text file that has a suffix `.html`; for instance, `mypage.html`

HTML undergoes gradual improvements. In this book you will be using HTML 5, which is the latest version. You can produce competent websites with just 40 English HTML words, and you can produce sophisticated websites with about 60 English HTML words. Although the words in HTML are easy, the syntax (grammar), as with all languages, can be a little harder to master. However, the syntax has logical rules that will gradually become second nature as you tackle the projects in this book.

But What About CSS?

CSS stands for cascading style sheets. It is a very convenient way of adding style to web pages. CMS kits state that you don't need to know CSS. That statement is true for CMS programs, but it is also a great pity because CSS is a wonderful time-saving device. It puts you in complete control of your website's appearance. Like HTML, it also uses plain English words.

Early in the development of HTML, web designers had to add the page styling code to every single page of a website. Let's assume that your pages have a picture in the banner heading. If some time later you decide to change the picture, every page in the website has to be altered. With CSS you need only change the picture in one single CSS file, and the new picture will magically appear on every page of the website. In other words, the change cascades throughout the whole website. A CSS file is a simple text file that has a suffix `.css`; for instance, `style.css`.

CSS was launched in 1996 and it undergoes improvements from time to time; the latest incarnation is CSS3. This new version makes it much easier to add enhancements such as rounded corners for borders and drop shadows for images and text.

Prepare Your Computer to Create Websites

You will need some tools for creating and testing your websites. There is no need to buy software for this purpose; a list of suitable free programs is provided next.

Install Suitable Browsers

You will need modern browsers such as Mozilla Firefox, Internet Explorer (version 11 or later), Opera, or Chrome. I recommend that you have these four free browsers installed so that you can check your websites on them all. MAC owners will have to use Safari.

Install a Free HTML Text Editor

You will need to download and install a free text editor because HTML and CSS files are simply text files saved with the extensions (suffixes) .html or .css. For instance, the home page for a website will be *index.html* because the home page is an index to all the other pages on a website. A style sheet file might be something like *style.css*.

Choose one text editor from the following list; they are all free and are listed in order of ease of use and versatility (the installation and configuration instructions are in Chapter 42.). The first two in the list are WYSIWYG editors. The rest are plain text editors (i.e., non-WYSIWYG). The acronym WYSIWYG means What You See on the screen is What You Get on the web page. What you type into the Design or WYSIWYG pane will appear on the web page, and you won't need to type much code; it feels like working with a word processor. Make sure that you have created a desktop icon for your HTML text editor, or pin its icon to the taskbar. When the instructions in this book say "open your HTML text editor," double-click the text editor's desktop icon to open it. Or single-click the icon on the taskbar.

■ **Warning** MS Word is a sophisticated text editor, but it should NEVER be used for producing web pages. A page saved as HTML in MS Word will have a word count 10 times bigger than necessary and will be very difficult to maintain.

MS Expression Web 4 (free) is an excellent, fully featured, and intuitive WYSIWYG text editor; however, it is only available for Windows.

<http://www.microsoft.com/en-us/download/details.aspx?id=36179>

BlueGriffon (free) is an excellent, up-to-date WYSIWYG text editor for Windows or Mac.

<http://www.bluegriffon.org/pages/download>

Important: If you are using a Mac you can use the Mac-bundled non-WYSIWYG text editor named TextWrangler.

TextEdit (free) is the easiest plain text editor to use. It is available for Windows or Mac.

<https://textedit.codeplex.com/>

Komodo Edit (free) is a plain text editor with several up-to-date enhancements.

<http://www.activestate.com/komodo-edit/downloads>

Notepad (free) is a very basic plain text editor that is bundled with Windows.

<http://www.notepad.com/download-now.php>

Notepad++ (free) is a much better plain text editor than Notepad.

<http://notepad-plus-plus.org>

■ **Tip** For the first few chapters, if you use a WYSIWYG program, I strongly recommend that you use code view to enter this book's code so that you can get used to the angle bracket keys and see how HTML works. However, later you can add or amend content by typing directly into the WYSIWYG pane, just like using a word processor.

Create a Folder for Your First Web Page

Create a folder called web-tutorial on your main hard drive (most likely to be the C: drive).

Within the web-tutorial folder create a folder named Chapter-1.

Now that you have a folder for the web page, you can create your first page, but first let's examine the basic structure common to all web pages.

■ **Tip** If you are eager to learn about creating a page, you can skip the explanation that follows and jump straight to the section headed Create Your First Web Page, then return to the explanation later.

The Basic Structure for Every Web Page

The minimum code required to produce a web page is as follows:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Some page title</title>
  </head>
  <body>
    <!--Your page content-->
  </body>
</html>
```

The first line tells the browser that you are using HTML5 code. The items such as <html>, <head>, <title>, and <body> are called tags and they are always surrounded by angle brackets.

To type angle brackets, use the Shift key together with the comma or full stop key.

The first line is the DOCTYPE tag; the content is not case sensitive so you could use lower case <!doctype html> if you wish. The DOCTYPE tag tells browsers what kind of document it is about to read. In this case it is an HTML5 document.

The tags <html> and </html> tell the browsers that everything between those tags is HTML code. The last tag has a forward slash to indicate to the browser that the HTML code is finished.

The `<head>` tags enclose an essential page title surrounded with `<title>` tags. The page title will be visible on the screen in a browser's tab; it is also used by search engines for indexing your website. The `<head></head>` tags can contain some optional information that you will learn about in later chapters. The last tag has a forward slash to indicate to the browser that the head code is finished. Any other information enclosed by the `<head>` tags will be hidden because the information is for the benefit of the browser and search engines; it will not be visible to the person viewing the page.

The `<body></body>` tags will contain all the content that you wish to display on the screen. Note that in this example, the words *Your page content* are surrounded by the tags `<!--and-->`.

These are called comment tags, and they prevent the surrounded words from displaying on the screen. If you deleted the comment tags in this example, you would see the words *Your page content* displayed on the screen by the browser. Comment tags are extremely useful; you can insert comments in a page as a reminder of what you intended the code to do at that point in the web page. The comment tags are also invaluable for troubleshooting.

Many text HTML editors will automatically produce the structural code for you: for example, when you open a new blank page in MS Expression Web 4 you will see this already entered:

```
<!DOCTYPE html>
<html>

<head>
<meta content="text/html; charset=utf-8" http-equiv="Content-Type">
<title>Untitled 1</title>
</head>

<body>

</body>

</html>
```

Note that all the words are English, but the line beginning `<meta` is not so straightforward. This will be explained in later chapters but because some text editor enters it for us, we need not worry about it for the present. In fact it can be omitted for the early chapters in this book.

Enhancing the Structure

The minimum code would be of little use, so we need to enhance the structure. All web pages have a wrapper and a minimum of four pieces of content between the `<body>` tags, which are the following:

- The wrapper keeps the web page content in a neat parcel,
- A header or banner at the top of the web page,
- Some text and perhaps some pictures in the main body of the page,
- A footer at the bottom of the page containing useful information such as a phone number and copyright details.

As previously stated, all website pages have a similar structure and the structure is completely logical. Figure 1-2 shows a diagram of the typical structure for web pages.

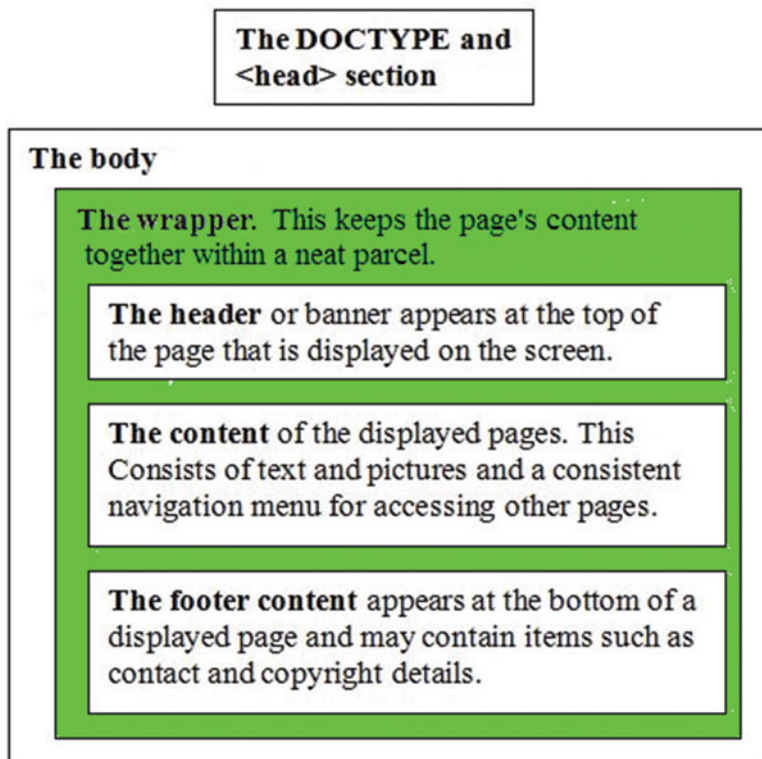


Figure 1-2. The structure of a typical web page

Listing 1-1. The HTML code for creating the structure shown in Figure 1-2. The HTML code is completely logical, although the first six lines need some explaining, and I will do this later in the chapter

```
<!DOCTYPE html>
<html>
  <head>
    <title>Template</title>
    <meta charset=utf-8>
  </head>
  <body>
    <div id="wrapper">
      <header>
      </header>
      <div id="content">
      </div>
      <footer>
      </footer>
    </div>
  </body>
</html>
```



Figure 1-3. Take note of the structure of the code. Compare it to one of those Russian dolls where the first doll contains a smaller doll, and that doll contains another smaller doll, and so on. The dolls are nested within one another; and in the same way, the blocks of HTML code are nested.

In the code Listing 1-1, the outer doll is `<html>` and `</html>`, the first inner doll is `<body>` and `</body>`, the next doll is `<div id="wrapper">` and `</div>`. The doll that is the `<div id="wrapper"></div>` section contains three individual dolls; they are the header, the content, and the footer. The analogy does not explain one of the blocks of code (i.e., `<head></head>`); in this case, the first doll has two small pieces of code located inside its `<head>` section; that is, `<title>Template</title>` and `<meta charset=utf-8>`.

The `<div>` tag is a very powerful tool that gives you complete command over the layout and styling of your web pages. This is especially true when the `<div>` is accompanied by an identity such as `<div id="wrapper">`. The `<div>` creates a division or section on the page, and the identity enables you to style that section any way you wish using CSS code (which is described and implemented in the next chapter).

■ **Tip** How much of the code and structure should you memorize? At the moment you need not memorize any of it, but it is important that you understand the underlying principles. Also try to remember the terminology such as tag, wrapper, body, footer, etc.

As you work through the chapters you will discover that you are beginning to memorize the jargon, the tags, and the structure. You will also realize that once you have created the initial web page, all future pages can simply be variants of that page. Web design is mainly a matter of copying, pasting, then adapting your existing pieces of code.

Create the Structure of Your First Web Page

1. Open your text editor in Code/Source view, and you will probably find that the text editor has already entered the code for a typical web page. You can use this code if you wish; however, so that you fully understand the structure and code that produces that structure; I strongly recommend that you delete any existing content so that you have an empty pane in the code view, then do the following:

Carefully type the code given in Listing 1-1 into the Code/Source view (not the Design or WYSIWYG view).

2. Use *File* then *Save As* to save the file in the folder *Chapter-1* and give it the file name *template.html*.
3. The resulting file is your first web page.

It is good practice to use the tab and return keys to copy (approximately) the indents and spaces as shown in the listing. The indents do not have to be exactly the same as the listing, but by using the tab key or space bar you will produce something similar. This formatting is known as beautifying the code. It is mainly done for clarity and is enormously helpful when troubleshooting. Each division `<div>` and section is visually isolated with Returns and Tabs so that you can quickly focus on the bit that is causing you a problem. Imagine trawling through a very big HTML file that was not beautified; it would be a nightmare.

Also, as the chapters progress you will encounter additional `<div>`s and sections that are nested within each other; the indents will helpfully reveal them clearly as separate sections.

■ **Note** The gaps (white spaces) in the code created by the space bar, tabs, and returns are completely ignored by browsers.

■ **Tip** Blue Griffon users will notice that some extra code is added; the line specifying the utf-8 code will automatically change to `<meta content="text/html; charset=utf-8" http-equiv="content-type">`. Ignore this because the extra code does not affect the outcome. Don't try to delete this extra code because it will only reappear when you save the file.

Access the page *template.html* in a modern browser to see the result. With the folder web tutorial on the screen, right-click the file *template.html* and select Open with, then choose your browser. If you have been successful, you should see the details shown in Figure 1-4.

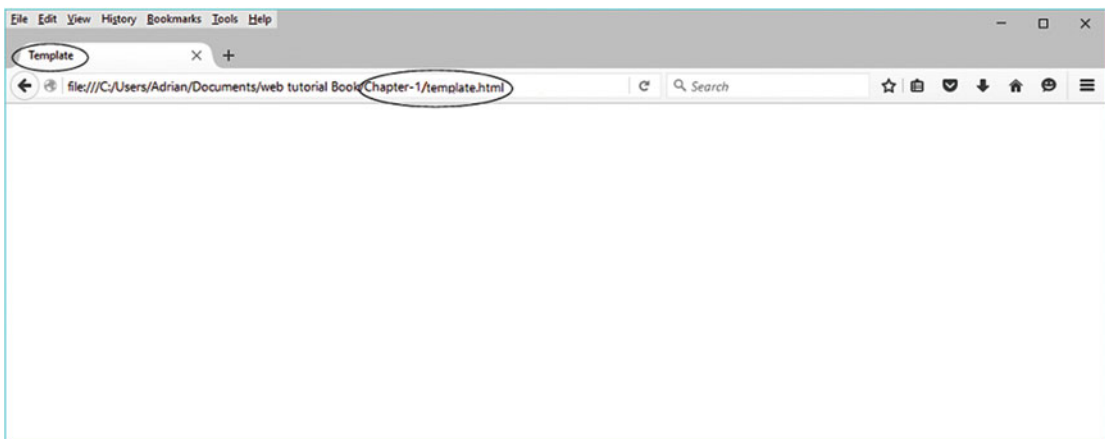


Figure 1-4. The file name is visible in the two circled locations but the main viewing pane is empty

The page title (shown circled) is visible in the tab and the address field. The text in front of the word 'Chapter-1' will be different for you, depending where you created the chapter.

Note that the HTML tags and the HTML code words are not displayed by the browser. The main window is empty because you have not yet added any content between the `<body>` tags.

If your result does not look like the above figure you must have typed something incorrectly, so use your HTML text editor to correct the code.

Discussion of the First Six Lines of the HTML Code in Listing 1-1

I have repeated the Listing 1-1 code below in order to explain the first six lines.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Template</title>
    <meta charset=utf-8>
  </head>
  <body>
    <div id="wrapper">
      <header>
      </header>
      <div id="content">
      </div>
      <footer>
      </footer>
    </div>
  </body>
</html>
```

The HTML code that you typed earlier creates the structure for a typical web page. This structure will be used in all the pages in this book. Using and reusing a template to create pages will greatly reduce the need to type raw HTML code.

The first six lines must appear at the beginning of every page of a website. Those lines tell the browser what to expect. There is no need to memorize them, because the template that you will use throughout this book (*template.html*) contains those lines of code. You don't even have to copy and paste them; just base all your pages on the template *template.html* and the lines are automatically included.

The title is very important for search engines, and the title also displays in the browser's tab for that page. Within the first set of lines, the title is the only item that changes on each page of a website.

The meta tag uses the term *utf-8*, which indicates to the browser that the web page uses multilingual characters. *<meta>* tags do not have a closing tag.

The body of the page contains three blocks of code: the header, the content, and the footer. Note that the three blocks are all enclosed within one *div* (division) called *<div id="wrapper">*. This keeps the page components together in a neat parcel.

The code between the *DOCTYPE* and the closing *</head>* tag is invisible to anyone viewing the website except for the page title, which will be visible in a browser tab. Let's check that statement. Open the *Chapter-1* folder and RIGHT-click the file *template.html*.

On the pop-up menu select *Open with* and then select your preferred browser: for example, Mozilla Firefox, Edge or Chrome.

■ **Troubleshooting** If you do see tags or any HTML elements displayed on the screen, you probably omitted a closing angle bracket.

Summary

Congratulations, you have produced the structure of your first (empty but rather useless) web page; however, you have learned some of the HTML buzzwords such as tag, nested, and DOCTYPE. You saw that the tags are English words, or abbreviations of English words, and that tags are always enclosed by angle brackets. You learned that the HTML code dictates the structure of the page, and that the same structure is used in all web pages. You produced a template that will form the basis of all the web pages in this book. This chapter also demonstrated that every page in a website must begin with the same block of code from the `<!DOCTYPE html>` to the `</head>` (except for the page title), and that by using your template for your web pages, these lines will be inserted automatically. You learned that the HTML tags are not displayed on the screen; they are provided only for the benefit of the browser.

In the next chapter you will add some content that will be visible in a browser. Then in Chapter 3 you will be introduced to hyperlinks so that you can navigate to additional pages.

CHAPTER 2



Create Your First Website and Add Hyperlinks

In this chapter you will learn how to produce hyperlinks that will enable you to move from page to page using the mouse, touch pad, or touch screen. This is essential for all websites that have more than one web page.

This chapter contains the following sections:

- Create a folder for the new chapter
- Let's add some content to your web page
- Create a home page
- Add a second page to the website
- Add hyperlinks to the two pages
- Explanation of the code
- Summary

Create a Folder for the New Chapter

For this lesson you need to create a new folder to contain the code for Chapter 2. This step is important to prevent confusing the code in the previous chapters with the new code.

1. Open your web-tutorial folder.
2. Create a folder named *Chapter-2*.
3. Open the folder *Chapter-1*.
4. Copy the file *template.html* (use *Ctrl+C*) and paste it into the new folder *Chapter-2*.

Let's Add Some Content to Your Web Page

■ **Note** When the following instructions say something like “type some text between the tags **as shown in bold type**,” you are not expected to use bold type in the text editor (but it does not matter if you do). The bold type in the instructions is present only to clarify what you need to type into the template.

In folder *Chapter-2*, open the *template.html* file in your text editor. Make sure you are in *Code/Source* view and then add some text between the tags **as shown in bold type** in the HTML Listing 2-1. If you wish to play safe here, first use *Save As* to make a copy of the template file named *original-template.html*, then go back to the file *template.html* and amend it as shown in bold type. If you mess up the template you can always go back to the original template and start again.

Listing 2-1. Add some content to the file *template.html*

```
<!DOCTYPE html>
<html>
  <head>
    <title>Template</title>
  <meta charset=utf-8>
</head>
  <body>
    <div id="wrapper">
      <header>
        This is the header
      </header>
      <div id="content">
        This is the content
      </div>
      <footer>
        This is the footer
      </footer>
    </div>
  </body>
</html>
```

Don't forget to save the revised file.

■ **Tip** Blue Griffon users will notice that some extra code is automatically added. Ignore this because the extra code does not affect the outcome. Don't try to delete the extra code because it will appear again when you save the file.

Minimize the text editor and open your folder named *Chapter-2*.

Now right-click *template.html* in the *Chapter-2* folder and select *Open with*. Then select your preferred modern browser. Figure 2-1 shows the result.

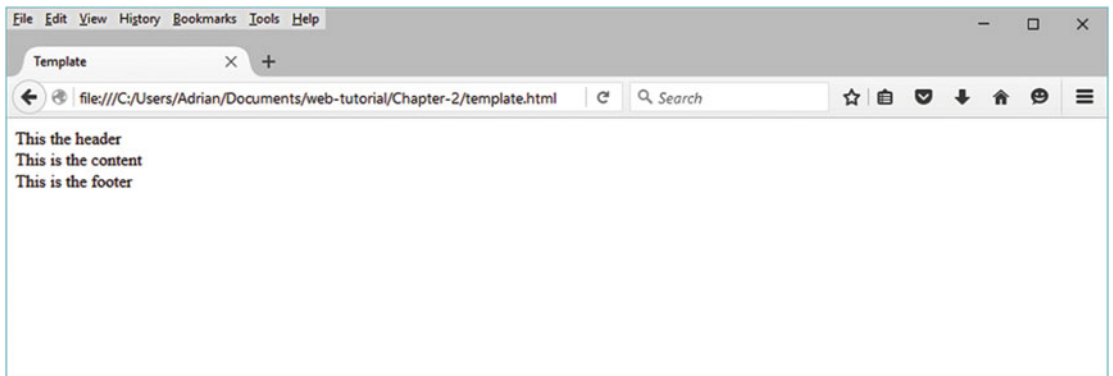


Figure 2-1. The browser now displays the text that you typed between the tags

The words in the address field will be different from the illustration depending on where you created the folder for Chapter 2. As before, the HTML tags are invisible (as they should be), but the text between the tags is now displayed. If you can see tags, then you have not entered the code accurately.

Create a Home Page

Hyperlinks allow users to move from page to page in a website. Before we can demonstrate hyperlinks we need to create a home page and a second page. This is simply a matter of copying and pasting from your template, then changing the page title.

1. In the Chapter-2 folder, right-click the *template.html* file and select *Open with...*, then choose to open it with your preferred text editor.
2. Save a copy of the file using *Save As...* and name it *index.html*. Keep this page on the text editor's screen ready for the next step.

■ **Note** The file *index.html* is the first page that appears when you access any website. It is always referred to as the Home Page.

3. Change the title on your new page to Home Page like this: `<title>Home Page</title>`
4. Also, beneath the line `<div id="content">` add the heading Home Page as shown in bold type.

```
<div id="content">
    This is the content
    <br>
    <strong>Home Page</strong>
    <br>
```

■ **Note** The `` tag causes the words *Home Page* to display in bold type. The tag `
` moves the next bit of text down one line; it stands for line break.” It does not need a closing tag. Think of it as the equivalent of pressing Enter on a word processor.

5. Save the file again.
6. Keep the *Home Page* code on the text editor screen ready for the next step.

Add a Second Page to the Website

Hyperlinks are devices for moving from one web page to another. To demonstrate this we need another page, which we will call page-2. We do not have to create the new page from scratch; we simply make a copy of an existing page as follows:

1. Save a copy of the *index.html* file using *Save As...* and name it *page-2.html*. Keep this page on the text editor’s screen ready for the next step.
2. Change the title on the new page to *Page Two* as shown below in bold type:

```
<title>Page Two</title>
```

3. Also, just beneath line `<div id="content">` add the heading **Page Two** as shown in bold type.

```
<div id="content">
    This is the content<br>
    <strong>Page Two</strong>
<br>
```

4. Save the file again as *page-2.html*.
5. Keep *page-2.html* on the text editor screen ready for the next step.

Add Hyperlinks to the Two Pages

We have a Home page and also a Page-2; the hyperlink on the home page will take us to Page-2. We need to be able to go back from Page-2 to the home page, so let’s add the same two hyperlinks to the two pages.

With *page-2.html* on the text editor’s screen, in code view, type the page heading and the following two hyperlinks beneath the line `Page Two` as shown in bold type in the next listing.

A full explanation will be given shortly.

```
<div id="content">
    This is the content<br>
    <strong>Page Two</strong>
    <br>
    <a href=page-2.html>Go to Page 2</a><br>
    <a href=index.html>Return to Home Page</a><br>
</div>
```

Save the file *page-2.html* again after adding the code.

Now open the home page file *index.html* in your text editor (in code view), and type the same hyperlinks as before as shown in bold (or copy and paste them from *page-2.html*).

```
<div id="content">
This is the content<br>
<strong>Home Page</strong>
<br>
      <a href=page-2.html>Go to Page 2</a><br>
      <a href=index.html>Return to Home Page</a><br>
</div>
```

Save the file with the same name *index.html*.

In the *Chapter-2* folder, right-click the file *index.html*. In the pop-up menu, select *Open with* and then select your preferred browser. Figure 2-2 shows the resulting display in your browser.

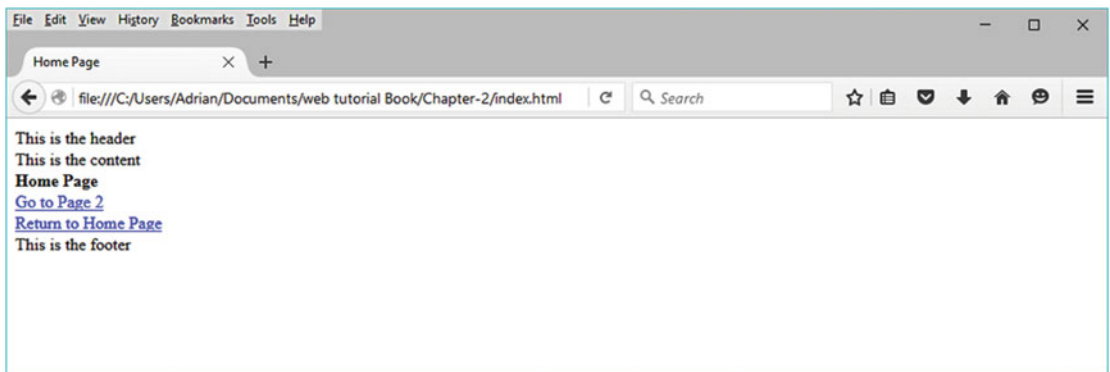


Figure 2-2. The two hyperlinks are displayed in blue and underlined. Note that the heading in bold type says “Home Page”

The appearance of the browser’s toolbar may be different in your case, depending on which browser you used. The content should be the same as shown in Figure 2-2.

Try clicking the blue underlined link to go to page 2. Figure 2-3 shows what you should see.

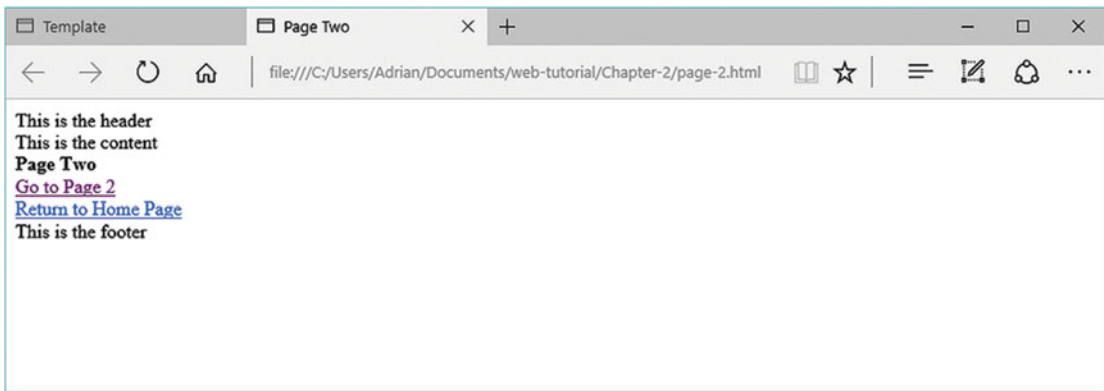


Figure 2-3. *Displaying page two. Note that the heading in bold type now says “Page Two”*

The hyperlinks may change color from blue to purple, but this is normal.

Now click the link to return to the home page.

You have added hyperlinks to a two-page website and you have been able to see how hyperlinks will move you from page to page. The code will now be explained.

Explanation of the Code

The code for a hyperlink looks complicated at first, but you will discover that it is reasonably logical.

A typical hyperlink takes this form: `Return to Home Page`.

The `<a...>` tag stands for anchor, which is rather cryptic; it would have been easier to understand if the word “access” had been used. For clarity this book assumes that `<a` stands for the word “access.”

`href` stands for hyperlink reference.

Therefore, the piece of code `Return to Home Page` displays a clickable hyperlink that will access the *index.html* file (which is the home page).

Note that the words *Return to Home Page* are located between the opening tag `` and the closing tag ``, any text between those tags is displayed on the screen.

The line break code `
` moves the next line of text down one line. Note that `
` tag does not need a closing tag.

Summary

You now have a very basic website. However, you learned how to copy the website’s template to create additional pages. You discovered how to display text on the screen by inserting it between tags in the content section. You learned how to make the text bold by wrapping it between `` tags. You discovered how to add hyperlinks using the `<a>` tags. You were able to move between the pages by clicking the hyperlinks. The two pages are currently very boring because no styling is applied. In the next chapter you will be introduced to some CSS styling and color to enliven the two pages.

CHAPTER 3



Styling the Website with CSS

Cascading Style Sheets (CSS) are based on the important principle of separating a page's appearance from a page's structure and content. CSS can convert dull pages into interesting and attractively styled pages. CSS is the language used for styling any element in an HTML page. It uses logical and simple English words or abbreviations of English words. A CSS style instruction has at least two simple parts: a *selector* and a *declaration*. For example, it could *select* paragraphs of text and *declare* that they should be displayed using Arial font.

This chapter contains the following sections:

- How do we link HTML pages to a CSS style sheet?
- How does the link to the style sheet work?
- Create a folder for the new chapter
- Creating a simple CSS style sheet
- Explanation of the style sheet *style.css*
- How much do you need to remember?
- Summary

The hard way to style the pages in a website would be to add the styling instructions within the `<head></head>` section of every page. This is called an internal style; for instance, the internal style could instruct the browser to display a pale green background on every page and to make all the fonts dark green using a 12-point font named Times New Roman. Let's suppose we later wished to change the background color to pale blue and the font to black Arial 10 point; we would have to change the internal style on every page of the website.

The easy way to style web pages is to put the styling instructions in a single external style sheet and link every page to that style sheet. A style sheet is simply a collection of instructions that tell the browser how to style the elements on a page. If we link each page to the single style sheet, the work involved in changing a website's appearance is reduced substantially. The appearance of all the pages in a whole website can be made by a change in the single external CSS style sheet. The change cascades into all the pages that are linked to the style sheet.

How Do We Link HTML Pages to a CSS Style Sheet?

We *link* them by means of a tag named *link* in the `<head>` section of a page. Assuming that the style sheet is called *style.css*, we simply add the same style link to all the HTML pages. In this project we will copy and paste the following line of HTML code into the `<head>` section of every page:

```
<link href="style.css" rel="stylesheet" type="text/css">
```

The *rel* attribute tells the browser that the link relates to a style sheet. The type attribute tells the browser the type of document it will be linked to. For style sheets the type is always "text/css," that is, a plain text document with a suffix .css. In our example this is *style.css*. Strictly speaking the type is not required in HTML5 pages because modern browsers assume that the style sheet will be *type="text/css."*

How Does the Link to the Style Sheet Work?

When users access a web page with a browser, the browser first absorbs all the information in the <head> section and stores it in its memory. Because the head section contains a link to a style sheet, the browser remembers the rules in that style sheet. Then the browser reads through the HTML code on any page accessed by the user. It then applies the style rules (declarations) to all the targets (selectors) it finds as it reads through the HTML code on every page.

For instance, if the style sheet contains this code, `p {font-size:14pt; text-align:center;}` it instructs the browser to look for the paragraphs (i.e., any text enclosed by the code `<p></p>`) in the HTML pages). Every time it finds a paragraph `<p>` it sets the paragraph's content to font size 14 pt. The style `text-align:center;` also tells the browser to center all the paragraphs on the page.

■ **Note** The style sheet uses a colon instead of an equals sign and each item is closed with a semicolon. The style must always be enclosed in curly brackets, for example, `p {font-size:14pt; text-align:center;}`. The word *center* is the U.S.spelling; it will not work if you use the U.K., Commonwealth, or European spelling *centre*.

Create a Folder for the New Chapter

For this lesson you need to create a new folder to contain the code for Chapter 3. This step is important to prevent confusing the new code with the code in the previous chapters.

1. Open your web-tutorial folder.
2. Create a folder named *Chapter-3*.
3. Open the folder *Chapter-2*.
4. Copy all the files in folder *Chapter-2* (use Ctrl+A then Ctrl+C) and paste those files into the new folder *Chapter 3* (using Ctrl+V)

In the folder *Chapter-3*, open *index.html* in your HTML text editor, then in code or source view, add the style sheet link just before the closing </head> tag. Also add some extra tags as shown in bold type in Listing 3-1.

■ **Tip** As you type a tag in the Expression Web 4 text editor, it tries to be helpful by automatically providing the closing tags. For the moment just make sure you move the closing tag to the end of its enclosed text. For instance, Expression Web will do this: `<p></p>sometext`

Move the closing tag to the end of the text like this: `<p>sometext</p>`. Later in the book, if you are using a WYSIWYG editor, you can use the Design/WYSIWYG view to avoid this problem.

Listing 3-1. Adding the link to the style sheet

```

<!DOCTYPE html>
<html>
  <head>
    <title>Home page</title>
    <meta charset=utf-8>
    <link href="style.css" rel="stylesheet" type="text/css">
  </head>
  <body>
    <div id="wrapper">

      <header>
        <b>h1>This the header</b>
      </header>

    <div id="content">
      <br>
      <b>h2>Home Page</b>
      <p>This is the content</p>
      <a href=page-2.html>Go to Page 2</a><br>
      <a href=index.html>Return to Home Page</a><br>

    </div>

      <footer>
        <p>This is the footer</p>
      </footer>

    </div>
  </body>
</html>

```

Save the file.

The tag `<h1></h1>` surrounds the words “This is the header,” the `<h1>` tag will give the heading a particular style called heading level-one. The largest heading available is `<h1></h1>`.

The tag `<h2></h2>` surrounds the words “Home Page” so that they can be given a particular style called heading level-two. The heading `<h2></h2>` is smaller than `<h1></h1>`. There are four more sizes and they progressively reduce in size.

The heading tags have default sizes and emphasis, but they can be styled using CSS to give any desired font size and font weight.

The tag `<p></p>` simply means paragraph. Any text between those tags will be a paragraph.

Now open *page-2.html* in your HTML text editor and add the style sheet link just before the closing `</head>` tag. Also add the extra `<h1></h1>` and `<p></p>` tags as shown in bold type in Listing 3-2:

Listing 3-2. Amending Page 2

```

<!DOCTYPE html>
<html>
  <head>
    <title>Page Two</title>
    <meta charset=utf-8>
    <link href="style.css" rel="stylesheet" type="text/css">
  </head>

```

```

<body>
  <div id="wrapper">

    <header>
      <h1>This the header</h1>
    </header>

    <div id="content">
      <br>
      <h2>Page Two</h2>
      <p>This is the content</p>
      <a href=page-2.html>Go to Page 2</a><br>
      <a href=index.html>Return to Home Page</a><br>
    </div>

    <footer>
      <p>This is the footer</p>
    </footer>
  </div>
</body>
</html>

```

Save the file.

We have just added a link to a style sheet that currently does not exist. In the next step you will create the style sheet.

Creating a Simple CSS Style Sheet

When typing the content of a CSS file you will notice the following features:

Each styling item contains a colon and is closed by a semicolon, for example,
width:1024px;

Curly brackets surround each style. These brackets are to the right of the “P” on your keyboard.

Some U.S. spelling must be used: for example, color and center.

The CSS words are plain English or abbreviations of English words.

■ **Important** Older versions of Blue Griffon did not allow you to create CSS code unless you paid for a plug-in. In the latest versions of Blue Griffon, make sure you save the CSS file with the suffix *.css* and in the field labeled *Save as Type*, ensure that you click the down-arrow, then select *All files*.

The code will seem like gobbledygook at the moment, but all will be revealed shortly.

```
#wrapper {margin:auto; width:1024px; border:2px blue solid;}
header {height:50px; text-align:center; color:red; background:yellow;}
h1 {font-size:22pt; font-weight:bold; text-align:center;}
h2 {font-size:18pt; text-align:center;}
p {font-size:14pt; text-align:center;}
a {font-size:14pt;}
```

If you are using a MAC with a U.K. keyboard, press Alt+3 to produce the # symbol.

Save the file with the name *style.css* and make sure it is saved into your *Chapter-3 folder*.

Your two pages will now have some style.

View the styled pages by viewing *index.html* and *page-2.html* in a modern browser (but not Internet Explorer 8). Provided you have configured Expression Web 4 to preview pages in a modern browser, while in the text editor, you can press the F12 key to see the display shown in Figure 3-1. (Configuration instructions for Expression Web 4 are provided in Chapter 42.)



Figure 3-1. The page now has some color and styling. The styling is not very special, but it is very different from the page with no styling in the previous chapter. The wrapper has a blue border, and the main heading is red against a yellow background.

Click the hyperlink to select *page 2* and you should see that the style has also cascaded onto that page.

Explanation of the Style Sheet *style.css*

We will now examine the CSS style sheet line by line.

The first line in Listing 3-3 shows the only tricky part of CSS; all the other lines are logical.

Line 1.

```
#wrapper {margin:auto; width:1024px; border:2px blue solid;}
```

A CSS file instructs the browser to look for a target in the HTML page.

In this example the CSS style `#wrapper` tells the browser to look for the HTML code `<div id="wrapper">`. The browser then applies the CSS style to the HTML item `<div id="wrapper">`.

This is the only cryptic part of CSS; all other HTML targets have the same name as the CSS code. For instance, the CSS code `p` targets the HTML code `<p>`, the CSS code `heading` targets `<heading>`, the CSS code `h1` targets `<h1>` and so on. The CSS identifier has the same name or word as the HTML code, but the angle brackets are omitted.

When the browser locates the target `<div id="wrapper">` in the HTML code it changes the wrapper according to the following instruction:

```
#wrapper {margin:auto;
```

This gives the wrapper equal margins; in other words, it centers the wrapper on the screen.

```
width:1024px;
```

This sets the width of the wrapper to 1024 pixels.

```
border:2px blue solid;
```

This gives the wrapper a 2-pixel thick solid blue border.

IMPORTANT: Between the curly brackets, each style must be separated from the next style by means of a semicolon.

Line 2.

```
header {height:50px; text-align:center; color:red; background:yellow;}
```

This CSS style tells the browser to look for a target named `<header>` in the HTML page. When the style sheet locates its target it changes the header as follows:

```
height:50px;
```

It gives the header a height of 50 pixels.

```
text-align:center;
```

Any text within the header is placed in the horizontal center of the header

```
color:red;
```

Any text within the header will be colored red.

```
background:yellow;
```

The background color of the header will be yellow.

Line 3.

```
h1 {font-size:22pt; font-weight:bold; text-align:center;}
```

This style ensures that any text within `<h1>` tags will be given a font size of 22 points, and it will be bold and centered.

The `<h1></h1>` tags indicate to the browser that the text within the `<h1></h1>` tags is the most important heading. There must be only one top level heading per page.

Line 4.

```
h2 {font-size:18pt; text-align:center;}
```

Any text with `<h2>` tags will be given a font size of 18 points, and will be centered.

The `<h2></h2>` tags indicate to the browser that the text within the `<h2></h2>` tags is second-level (i.e., the text is the second most important heading on the page). You can have more than one second-level heading on a page.

Line 5.

```
p {font-size:14pt; text-align:center;}
```

Any text within paragraph tags (i.e., `<p></p>` tags) will be given a font size of 14 points and it will be centered on the screen.

Line 6.

```
a {font-size:14pt;}
```

Any lines with `<a>` tags (i.e., link access) will be given a font size of 14 points.

How Much do you Need to Remember?

Should all the HTML and CSS code be committed to memory in order to learn and use HTML and CSS? The answer is “no” for the following reasons:

- As you progress through the chapters you will be creating templates that already contain all the code. You don’t need to type the code; just copy and paste it into your websites. These templates can be adapted for your own websites by tweaks to the existing code and changes to the textual content.
- You can always refer to the book’s chapters or to the quick reference in Chapter 41 to rediscover a piece of code.
- If you use one of the WYSIWYG editors, the great majority of the work involved in creating, adding to, or updating a website can be done in the Design/WYSIWYG pane of your editor, just as if you were modifying a document in MS Word for Windows. Therefore you don’t need to type code.
- As you practice the chapters or create your own websites, you will find that an increasing amount of the code and its syntax is actually stored in your brain without conscious effort.
- Even experts cannot remember all the code and syntax. After 16 years of website design, I still have to look up some things to refresh my memory.
- The important thing is to remember the “hows”: for example, how do I use the text editor? How do CSS sheets style a page of HTML? How do I find code or syntax that I have forgotten?
- Many editors will alert you to the fact that you have entered some faulty code or CSS. You will have noticed that the syntax for the CSS is rather different from HTML syntax for `<div>`s that have an identity. For instance, the target in some HTML code might be: `<div id=“wrapper”>`, but the CSS that tells the browser to look for that target would be `#wrapper {width:1024px;}`. You do need to know this difference. After a hard day of updating several websites, I might become weary and type something like this: `#wrapper {width=1024px;}`
I should have used a colon like this: `#wrapper {width:1024px;}`
- If you accidentally use an equals sign instead of a colon, the style simply won’t be applied. The lack of styling will alert you to the fact that you have made a mistake that can easily be corrected by changing the equals sign to a colon.

The CSS code and its HTML target can be summarized in Table 3-1 as follows:

Table 3-1. *The CSS equivalents of the HTML code*

CSS code	HTML target to be styled by the CSS code
body {some style;}	<body>some content</body>
h1 {some style;}	<h1>some content</h1>
h2 {some style;}	<h2>some content</h2>
h3 {some style;}	<h3>some content</h3>
h4 {some style;}	<h4>some content</h4>
p {some style;}	<p>some content</p>
br {some style;}	
a {some style;}	<a>some content</p>
header {some style;}	<header>some content</header>
footer {some style;}	<footer>some content</footer>
#wrapper {some style;}	<div id="wrapper"></div>

Note that the English words or abbreviations in a CSS file and the HTML file are the same except that the angle brackets are omitted in the CSS file. As discussed earlier, the CSS file uses # instead of the HTML code **id=**

All HTML text editors have an undo and redo feature, if you make a mistake you can always go back to an earlier change point, even if you saved the file. However, this won't work if you have closed down the text editor. Figure 3-2 shows the Undo/Redo symbol on the Toolbar of MS Expression Web; you will see a similar symbol in any HTML text editor except Notepad where you must click *Edit* on the menu and select *Undo*.

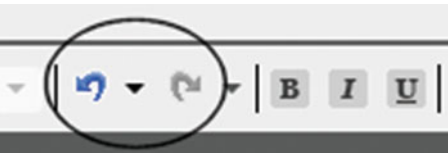


Figure 3-2. *The Undo and Redo symbols*

Summary

In the <head> section of our two pages you added a link to a style sheet. You also added some new target tags such as <h1> <h2>, <h2></h2> and <p></p> so that the style sheet could style the text between those target tags. You then created a CSS style sheet and discovered what each line meant. We tested the two pages and found that the style cascaded onto both pages. The style sheet was explained, and you learned that styles consist of plain English words arranged in a standard format using a colon and a semicolon, for example color:red; You learned that the style for each target tag must be enclosed in curly brackets, for example, **p {font-size:14pt; text-align:center;}** Each style must be separated from the next style by means of a semicolon.

In the next chapter, by using CSS, you will create some columns on the home page, and then you will experiment with the style sheet to see the styles cascade through to the other pages.

CHAPTER 4



Create Web Pages with Three Columns Using CSS

Web pages are normally divided into three or four vertical columns. Before the advent of browsers that were capable of fully understanding CSS, page columns were created by using tables; this practice has now been deprecated (made obsolete and unacceptable). In fact, the inventors of HTML never intended that tables should be used for page layout. Tables are strictly for presenting data. Page layout and columns should always be produced by using CSS and tags that create divisions. The opening and closing tags for the simplest division of a web page are as you would expect: `<div></div>`.

In a typical three-column web page, one column (on the far left or far right) is usually reserved for the hyperlink menu. The column on the opposite side of the page can contain information about the web designer and perhaps copyright details. The central column contains the page content (pictures and text).

This chapter has the following sections:

- Create the three-column page
- Explanation of the code
- Create a style sheet for producing three columns
- Explanation of the amended CSS code
- Revise the CSS code
- Explanation of the revised CSS code
- The difference between *id* and *class*
- Summary

For this lesson you need to create a new folder to contain the code for Chapter 4. This step is important to prevent confusing the previous chapter's code with the new code.

1. Create a folder named *Chapter-4*.
2. Copy all the files in your folder *Chapter-3* (Ctrl+A then Ctrl+C).
3. Paste (Ctrl+V) those files into the new folder *Chapter-4*.

Create the Three-Column Page

We will now create a three-column page with the hyperlink menu on the left.

Figure 4-1 shows you what you will achieve when you have completed this chapter.



Figure 4-1. This shows the three columns you should see when you have completed this chapter

1. Open your new folder *Chapter-4*. Right-click *index.html* and choose your HTML editor, then save a copy of the file as *original-index.html*. You now have an *index* file and an *original index* file. The original *index* file is purely a safeguard in case you mess up the code; you can always return to the original *index* and start again.
2. Close the file *original-index.html* and open *index.html* again in your HTML editor. Then change the page's HTML as follows in the next two steps.
3. Replace the `<link` to the style sheet with a link to the style sheet *style-3col.css*. The revised link is shown in bold type in the code Listing 4-1.
4. Then replace everything below the `<div id="content">` tag to just above the closing `</footer>` tag with the code shown in bold type in Listing 4-1.

■ **Tip** Look carefully at the code and try to discern the new structure of the page. I have abbreviated the name of each column as col (as in leftcol, rightcol, and midcol). However, when you create any *id* (identity) for a *div*, you can choose any name that is meaningful to you.

Listing 4-1. Modify the Page Structure to Include Three Columns

```
<!DOCTYPE html>
<html>
  <head>
    <title>Home Page</title>
    <meta charset=utf-8>
    <link href="style-3col.css" rel="stylesheet" type="text/css">
  </head>
  <body>
    <div id="wrapper">
      <header>
        <h1>This is the header</h1>
      </header>
      <div id="content">
        <div id="leftcol">
          <p>This is the far left column</p>
          <a href=page-2.html>Go to Page 2<br>
```

```

        <a href=index.html>Home Page</a><br>
    </div>
    <div id="rightcol">
        <p>This is the far right column</p>
    </div>
    <div id="midcol">
        <h2>Home Page</h2>
        <p>This is the content</p>
    </div>
</div><!--content div finishes here-->
    <footer>
        <p>This is the footer</p>
    </footer>
</div><!--wrapper div finishes here-->
</body>
</html>

```

■ **Note** In the HTML for *index.html*, the header and all three columns are confined within the wrapper. The blue border in Figure 4-1 marks the wrapper's boundary.

5. Save the file again as *index.html*
6. Using your HTML editor, open the file *page-2.html* and insert the same code (shown in bold type in Listing 4-1) into *page-2.html*
7. Save the file.

Explanation of the Code

The content of each section is surrounded by the tag `<p></p>`, which stands for paragraph. This acts as a target for the CSS style sheet so that it can style each paragraph.

The code also contains another three new target tags `id="leftcol," id="rightcol,"` and `id="midcol."`

A revised style sheet called *style-3col.css* will tell the browser to look for those three tags and then to style the three columns.

You will notice an amendment just above the `<footer>` tag as follows:

```
</div><!--content div finishes here-->
```

Also just before the closing `</body>` tag you will see an amendment as follows:

```
</div><!--wrapper div finishes here -->
```

As mentioned before, these items surrounded by the tags `<!--` and `-->` are comments. Comments are there to help you; they do not display in the browser. It is good practice to add comments to your code. Comments are not only useful when troubleshooting, but they help you to discern the structure of the page when you need to view the code at a later date.

To style columns using CSS, we need to create a suitable style sheet.

Create a Style Sheet for Producing Three columns

If you are using a Mac you may need the additional text editor named TextWrangler to produce and amend CSS files.

At the moment we don't have a style sheet named *style-3col.css*, so we will create one by amending the file *style.css*.

1. In your new folder *Chapter-4*, open *style.css* in your HTM editor and use *Save as...* to save a copy of the file as *style-3col.css*.
2. Don't close the new style sheet, but use code/source view to add the styles for the three columns as shown in bold type in the following snippet:

```
#wrapper {margin:auto; width: 1024px; border:2px blue solid;}
header {height:50px; text-align:center; color:red; background:yellow;}
h1 {font-size:22pt; font-weight:bold; text-align:center; }
h2 {font-size:18pt; text-align:center;}
p {font-size:14pt; text-align:center;}
a {font-size:14pt;}
#leftcol {float:left; width:150px; height:130px; background:aqua;}
#rightcol {float:right; width:150px; height:130px; background:aqua;}
#midcol {margin-left:155px; margin-right:155px;}
```

3. Save the style sheet again.
4. Now in the folder *Chapter-4* right-click the file *index.html*, then click *Open with* and select a modern browser (not IE8). You should see the display shown in Figure 4-2.



Figure 4-2. This display reveals a problem. The aqua-colored left and right columns are covering up some of the bottom border. We will correct this later

Explanation of the Amended CSS Code

#leftcol {float:left; width:150px; height:130px; background:aqua;}

The style **#leftcol** tells the browser to look in the file *index.html* for the target tag **<div id="leftcol">** It then applies the styles as follows:

The leftcol column is floated to the far left of the wrapper by the style **float:left;**

The width of the column is set at 150 pixels and its height is set at 130 pixels.

To enable you to see the column, I have given it a background color of **aqua** (pale turquoise).

#rightcol {float:right; width:150px; height:130px; background:aqua;}

The style **#rightcol** instructs the browser to look through the HTML code in file *index.html* for the target tag **<div id= "rightcol">**.

It applies the same style as **leftcol** except that the column is floated to the far right of the wrapper.

#midcol {margin-left:155px; margin-right:155px;}

Remember that the style instructions for the margins are relative to the wrapper's boundary. The middle column has been given two margins by the CSS style. The left margin pushes the **midcol** column 155 pixels to the right of the wrapper's left boundary. This ensures that the left edge of the middle column is clear of the left column by 5 pixels.

Similarly, the right margin pushes the edge of the **midcol** 155 pixels to the left of the wrapper's right boundary. This ensures that the right edge of the middle column is clear of the far right column.

The margins are 5 pixels bigger than the left and right column widths to make sure the **midcol** (middle column) really does sit clear of the outer columns. The outer columns and **midcol** bit are displayed on the screen as three separate columns.

If you find the explanation rather confusing, the diagram in Figure 4-3 will help you to understand.

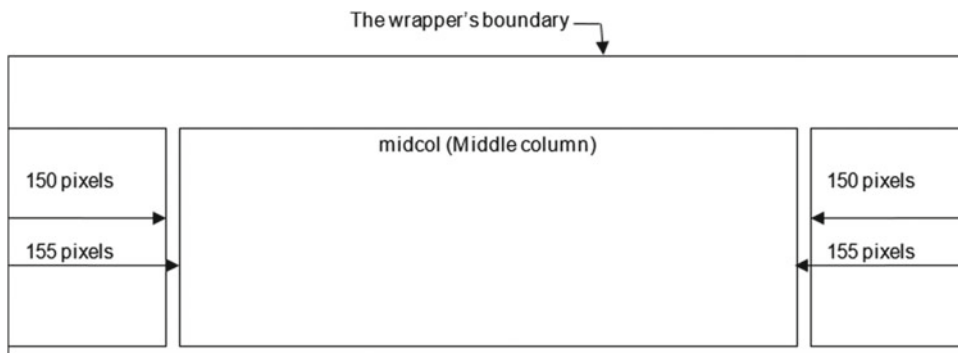


Figure 4-3. The 155 pixel margins set the middle column 5 pixels clear of the outer columns

Revise the CSS Code

In Figure 4-2 we saw that the two aqua-colored outer columns overlapped the bottom border of the wrapper. We will now correct this using a little trick with CSS. It will also introduce you to the tag attribute called *class*.

Open the CSS file *style-3col.css* in your HTML editor and add this new style to the bottom of the list of styles as shown in bold type in the following snippet:

```
header {height:50px; text-align:center; color:red; background:yellow;}
#wrapper {margin:auto; width:1024px; border:2px blue solid;}
h1 {font-size:22pt; font-weight:bold; text-align:center; }
h2 {font-size:18pt; text-align:center;}
p {font-size:14pt; text-align:center;}
a {font-size:14pt;}
#leftcol {float:left; width:150px; height:130px; background:aqua;}
#rightcol {float:right; width:150px; height:130px; background:aqua;}
#midcol {margin-left:155px; margin-right:155px;}
br.clear {clear:both}
```

Save the file as *style-3col.css*

The style code **br.clear {clear:both}** creates a class named `.clear`. The browser is instructed to do something to any line break **
** in the HTML code that has been given a class named `clear`. The line break just above the **<footer>** has been given the class `clear` and so the browser pushes the footer down clear of any floated columns. In our case, the floated columns are the aqua-colored columns on the far left and far right.

Explanation of the Revised CSS Code

```
br.clear {clear:both;}
```

The full stop in front of the word *clear* creates a class called *clear*. The tag `
` is a new line break, and *clear* is a *class* of style that will push the line clear of any floated columns that are located above the line break. This style tells the browser to look in the HTML page for any **
** with a class named *clear*. It then pushes the next line (and anything below it) clear of the floated columns.

```
{clear:both;}
```

The word *clear* is always associated with floats. Any element with the class *clear* is pushed clear of any floated items such as the floated left and right mid columns. The word *both* ensures that the element having the class *clear* will be pushed clear of both floated columns. In a future chapter you learn about two other versions of the *clear* style, that is, **clear:left;** and **clear:right;**

We will now apply that style to the two web pages.

Open *index.html* in your text editor, then immediately above the **<footer>** tag insert the line shown in bold type.

```
<br class="clear">
  <footer>
  <p>This is the footer</p>
  </footer>
```

Save the file as *index.html*

Open *page-2.html* in your text editor and immediately above the **<footer>** tag insert the line shown in bold type.

```
<br class="clear">
  <footer>
  <p>This is the footer</p>
  </footer>
</div>
```

Save the file as *page-2.html*

View *index.html* and *page-2.html* in your browser. The aqua-colored outer columns should no longer overlap the bottom border because the footer section has been styled to clear the three columns (see Figure 4-1).

The Difference Between *id* and *class*

You have already met the tag identifier named `id`, for example `<div id="wrapper">`. All `ids` are unique, that is, you can only have one `id="wrapper"` on a page. You can have many `<div>s` with `ids` on a page provided you give them different names after the equals sign. However, you can have an unlimited number of class identifiers with the same name after the equals sign.

For instance you can use the class identifier `<br class="red">` repeatedly on the same page to change the color of certain pieces of text to red.

Summary

You have learned some new HTML tags and several new CSS styling words. These enabled you to create a three-column page. The float style is used extensively to push items to the far right or far left of a boundary such as the wrapper. You were introduced to the class styling feature and learned how it differed from `ids`. You found that the CSS symbol for a class is a full stop, for example, `.clear`, and the CSS symbol for an `id` is `#`, for example, `#wrapper {margin:auto; width:1024px; border:2px blue solid;}`

You learned how the styling command `br.clear{clear:both;}` is used to position an element below two or more floated page elements. You discovered how to introduce more color, and also how margins can position an element on a page.

In the next chapter we will create a four-column web page.

CHAPTER 5



Create Web Pages with Four Columns Using CSS

A four-column page is a common format; it consists of a far left and a far right column, and a middle content section that is divided into two columns containing text or pictures. Normally the far left and far right columns are narrow, and the two *content* columns in the middle of the page are much wider.

This chapter contains the following sections:

- Create web pages with four columns
- Explanation of the code
- Insert new styles in the CSS style sheet
- Create two more pages
- The importance of forward planning
- Summary

For this lesson you need to create a new folder to contain the code for Chapter 5. This step is important to prevent confusing the previous chapters' code with the new code.

1. In the *web-tutorial* folder, create a new folder named *Chapter-5*.
2. In the *web-tutorial* folder, open the folder *Chapter-4*.
3. Copy all the files (Ctrl+A, then Ctrl+C) in the *Chapter-4* folder and paste them (Ctrl+V) into the new folder *Chapter-5*.

We will now create a four-column page with a hyperlink menu on the left. Figure 5-1 shows you what you will achieve when you have completed this chapter.

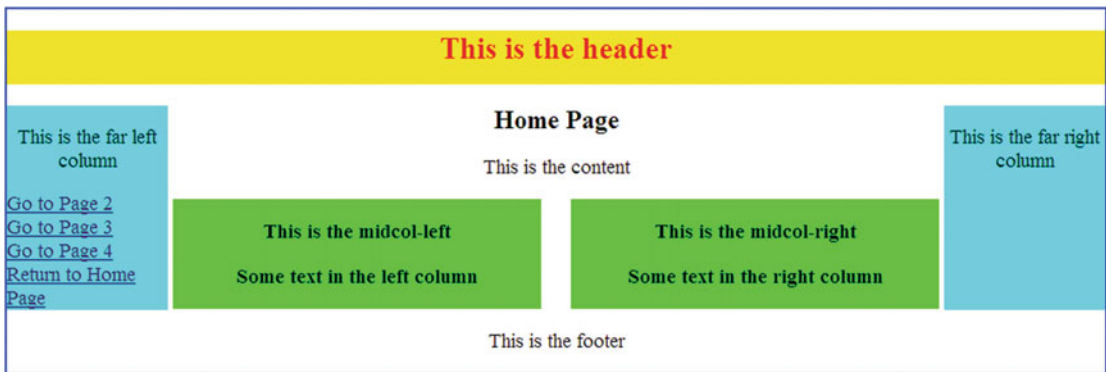


Figure 5-1. The appearance of the page when you have completed this chapter

You should see two new lime-green central columns.

Note The new style page will be created by means of small amendments to the files you created in the previous chapter.

Create Web Pages with Four Columns

To create the page with four columns shown in Figure 5-1, please take the following steps:

1. Open the new folder *Chapter-5*.
2. Open the *index.html* file in your HTML editor and save it as *original-index.html*.
You now have an index file and an original index file. This is purely a safeguard in case you mess up the code; you can always return to the *original-index.html* file and start again. Remember that you can also use the *Undo* icon on the tool bar of your HTML editor.
3. Close the file *original-index.html* and open *index.html* again in your HTML editor. Then change the page structure as follows.
4. Replace the <link to the CSS style sheet with a link to the file *style-4col.css* shown in bold type in the next code Listing 5-1.
5. To create two more columns, insert two divisions (<divs>) into the *midcol* section of the page as shown in bold type in the following Listing 5-1.

Listing 5-1. Create a home page with four columns

```
<!DOCTYPE html>
<html>
  <head>
    <title>Home Page</title>
    <meta charset=utf-8>
    <link href="style-4col.css" rel="stylesheet" type="text/css">
  </head>
```

```

<body>
  <div id="wrapper">
    <header>
      <h1>This is the header</h1>
    </header>
    <div id="content">
      <div id="leftcol">
        <p>This is the far left column</p>
        <a href=page-2.html>Go to Page 2<br>
        <a href=index.html>Home Page</a><br>
      </div>
      <div id="rightcol">
        <p>This is the far right column</p>
      </div>
      <div id="midcol">
        <h2>Home Page</h2>
        <p>This is the content</p>
        <div id="midcol-left">
          <p>This is the midcol-left</p>
          <p>Some text in the left column</p>
        </div>
        <div id="midcol-right">
          <p>This is the midcol-right</p>
          <p>Some text in the right column</p>
        </div>
      </div><!--the midcol div finishes here-->
    </div><!--the content div finishes here-->
    <br class="clear">
    <footer>
      <p>This is the footer</p>
    </footer>
  </div><!--the wrapper div finishes here-->
</body>
</html>

```

■ **Note** As previously stated, when modifying a web page or a style sheet, you never have to start from scratch by typing in all the code. You simply insert a small piece of code into your existing files.

6. Save the page.
7. Now insert the same code for the two new columns (shown in bold type in Listing 5-1 above) into the file *page-2.html*). You can type the code or you can copy and paste the revised code from *index.html*.

Explanation of the Code

```
<div id="midcol-left">
    <p>This is the midcol-left</p>
<p>Some text in the left column</p>
</div>
```

The code inserts an additional column within the *midcol* section. I have called it *midcol-left*; remember that you can call an id anything you like; it could be *left-middle-column*, *leftmidcol*, or *midleft-col*. Whatever id name you choose, be sure to give it the same name in the CSS style sheet so that the style sheet can tell the browser what to look for.

```
<div id="midcol-right">
    <p>This is the midcol-right</p>
<p>Some text in the right column</p>
</div>
```

The code inserts a second column into the *midcol* section called *midcol-right*.

```
</div><!--the midcol div finishes here-->
```

I added a new comment alongside the closing *midcol* **</div>**.

Note The new structure will have no effect until we insert a small piece of code into the style sheet.

Insert New Styles in the CSS Style Sheet

Open the file *style-3col.css* in your HTML editor (use TextWrangler if you are using a Mac) and save it as *style-4col.css*.

With the new style sheet *style-4col.css* in the editor's screen, we will increase the height of the two outer columns to 190 pixels in readiness for adding two more links to the menu. Then we will add two more styles to the style sheet to add the two extra columns. Please insert the lines shown in bold type in the next listing:

```
#wrapper {margin:auto; width:1024px; border:2px blue solid;}
header {height:50px; text-align:center; color:red; background:yellow;}
h1 {font-size:22pt; font-weight:bold; text-align:center; }
h2 {font-size:18pt; text-align:center;}
p {font-size:14pt; text-align:center;}
a {font-size:14pt;}
#leftcol {float:left; width:150px; height:190px; background:aqua;}
#rightcol {float:right; width:150px; height:190px; background:aqua;}
#midcol {margin-left:155px; margin-right:155px;}
br.clear {clear:both}
#midcol-left {float:left; width:48%; background:lime; font-weight:bold;}
#midcol-right {float:right; width:48%; background:lime; font-weight:bold;}
```

You can view the page *index.html* now; it should look like Figure 5-1 at the beginning of this chapter.

Explanation of the CSS Code

The height of the outer columns is increased to 190 pixels so that the aqua color embraces the two extra links that will be added later.

The new column styles look through the HTML file to find the `<div>`s with the identities `midcol-left` and `midcol-right`. It then styles those `<div>`s by floating one to the left and one to the right. This creates two columns within the `midcol` section. Each column is given a width equal to 48% of the width of the `midcol`. The columns are then given the background color lime-green so that you can see them clearly in a browser. The text within those columns has been made bold by the style `font-weight:bold`;

Create Two More Pages

To reinforce the concept that web design can be a process of copying and pasting from existing files, we will create two more pages. Our website will then have four pages and their design will be consistent and cohesive, which is the basis of a good website.

In the next step I will demonstrate how easy it is to create two more pages using the existing files as templates. We must bear in mind that new pages require new hyperlinks so that we can access any page from all the pages in the website.

Create the first new page by taking the following steps.

1. Open *index.html* in your HTML editor.
2. Add two more links to the menu as shown in bold type in the next snippet.

```
<div id="leftcol">
  <p>This is the far left column</p>
  <a href=page-2.html>Go to Page 2<br>
  <a href=page-3.html>Go to Page 3<br>
  <a href=page-4.html>Go to Page 4<br>
  <a href=index.html>Return to Home Page</a><br>
</div>
```

3. Save the file as *index.html*.
4. Add the same two menu links to the file *page-2.html* and save the page.
5. With *page-2.html* open in the HTML editor, create a copy using *Save as*, and name it *page-3.html*.
6. With the new *page-3.html* open in the HTML editor, change two items as shown in bold type in the code snippet:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Page Three</title>
    <meta charset=utf-8>
    <link href="style-4col.css" rel="stylesheet" type "text/css">
  </head>
```

```

<body>
    <div id="wrapper">
        <header>
            <h1>This is the header</h1>
        </header>
        <div id="content">
            <h2>Page Three</h2>
        </div>
    <div id="midcol-left">

```

7. Leave the rest of the code as it is and save the file as *page-3.html*.

Create the second new page by taking the following steps.

1. With *page-3.html* in your text editor, save a copy of it using *Save as* and name it *page-4.html*.
2. With the newly saved *page-4.html* open in the HTML editor, change the three items as shown in bold type in the next snippet.

```

<!DOCTYPE html>
<html>
    <head>
        <title>Page Four</title>
        <meta charset=utf-8>
        <link href="style-4col.css" rel="stylesheet" type "text/css" >
    </head>
    <body>
        <div id="wrapper">
            <header>
                <h1>This is the header</h1>
            </header>
            <div id="content">
                <br>
                <h2>Page 4</h2>
            </div>
        </div>
    </body>

```

3. Save the file again as *page-4.html*.
4. Open *index.html* in your browser and use the menu to navigate from page to page; you should discover that the new styles have cascaded into all four pages.

The Importance of Forward Planning

You could add as many pages as you like by repeating this process; remember you would also need to alter their titles and headings and add appropriate links to the extra pages, for example:

```

<a href=page-5.html>Go to Page 5<br>
<a href=page-6.html>Go to Page 6<br>

```

That would be rather tedious. However, the tedium can be greatly reduced by some forward planning. From the beginning, try to work out how many pages your website would contain. Allocate one page per topic; this will equate to one menu link per page.

For example, let's say you are Fred Bloggs and you have decided to produce a 9-page website to sell your paintings and greetings cards. Your forward plan might be to produce the following 10 topics/pages:

- About Fred
- Portraits
- Landscapes
- Seascapes
- Wild Life
- Still Life
- Prints
- Contact Fred
- Home Page

To save a great deal of time, you would first insert the nine links into the menu on the home page. Then you would copy the home page as a template for the other eight pages; by doing this you would automatically include the nine-link menu in every page.

On the template, and on all the other pages in this example, the code for the menu would be as follows:

```
<div id="leftcol">
    <p>This is the far left column</p>
    <a href=about.html>About Fred<br>
    <a href=portraits.html>Portraits<br>
    <a href=landscapes.html>Landscapes<br>
    <a href=seascapes.html>Seascapes<br>
    <a href=wildlife.html>Wild Life<br>
    <a href=still-life.html>Still life<br>
    <a href=prints.html>Prints<br>
    <a href=contact.html>Contact Fred<br>
    <a href=index.html>Return to Home Page</a><br>
</div>
```

In Chapter 21 you will learn a shortcut that will make an instant modification to a hyperlink menu on every page of a website. This is achieved by simply amending one file instead of amending every page in the website.

Summary

In this chapter you learned that when producing a website or modifying a web page or a style sheet, you rarely have to start from scratch by typing in all the code. You simply insert a small piece of code into your existing files.

You discovered how to split the middle column into two columns, thus creating a typical four-column layout. You learned how easy it is to create extra pages by copying existing pages and using *Save as*. You were then able to observe how the new styles cascaded into all the pages.

The four-page website you have created is rather plain and uninteresting. In the next chapter you will rectify this and learn how to insert pictures and position them on the page.

CHAPTER 6



Add Pictures to Websites

Pictures are an important part of a website, and of course they should be chosen to suit the theme of the website. Pictures must be optimized so that the browser downloads the picture to the screen quickly. We will add pictorial content to the website that you produced in the previous chapter. This chapter contains the following sections:

- The problem with some older browsers
- Add pictures to the home page
- To alter the home page using a WYSIWYG editor
- To alter the home page using a plain text editor
- Explanation of the code
- Alter the CSS file to style the new home page
- Explanation of the CSS code alterations
- Change the header in the other three pages
- Summary

When you have completed this chapter, the home page should look like [Figure 6-1](#).



Figure 6-1. The home page as displayed in a modern browser

The Problem with Some Older Browsers

In this book we are using the latest HTML5 code that employs the more meaningful semantic tags such as `<header></header>` instead of `<div id="header"></div>`. However, some older browsers don't understand semantic tags. For instance, Internet Explorer 8 (IE8) produces the unsatisfactory view of the HTML5 page as shown in Figure 6-2.



Figure 6-2. Showing the unsatisfactory appearance of the home page in Internet Explorer version 8

Because IE8 and old versions of other browsers will be with us for a while, I provide a temporary solution to this problem in Chapter 7.

Add Pictures to the Home Page

For this lesson you need to create a new folder to contain the code for Chapter 6. This step is important to prevent confusing the previous chapter's code with the new code. Please take the following steps:

1. In the *web-tutorial* folder, create a new folder named *Chapter-6*.
2. In the *web-tutorial* folder, go back to the *Chapter-5* folder and open it.
3. Copy all the files (Ctrl+A and Ctrl+C) in the *Chapter-5* folder and paste them (Ctrl+V) into the new folder *Chapter-6*.
4. Download the *Chapter-6* folder from the book's page at Apress.com and unzip it into your *Chapter-6* folder.

The *images* folder also contains a picture named **art1.jpg**, and we will eventually insert this image into the *midcol-right* section of the page.

First we need to make a few small changes to the HTML file *index.html*

To Alter the Home Page Using a WYSIWYG Editor

If you are using a WYSIWYG editor, instead of typing into the code/source view, try using the Design/WYSIWYG view to alter the text:

1. Open the folder named *Chapter-6*, then open the home page *index.html* in your HTML text editor and in the Design/WYSIWYG view and amend the text as follows:
2. Delete the line: "This is the content"
3. Replace "This is the Header" with the words: **The Rainbow Gallery**
4. Replace "This is the far left column" with the word: **Menu**
5. Replace "Return to Home page" with the words **Home Page**
6. Replace the words: "This is the midcol-left. Some text in the left column"

With the following text:

The Rainbow Gallery has a wide range of Original oil paintings, Greetings Cards and Prints. View and order items using this website or visit the gallery at 12 High Street, Townsville.

■ **Note** By using the WYSIWYG view you do NOT have to type any HTML tags. If you are using a non-WYSIWYG text editor, you obviously will have to type the text and include the tags. Microsoft Expression Web 4 has an irritating quirk: if you have specified a language in the header it will insert lots of `` codes and mess up the appearance of any inserted text. Therefore, in the `<head>` section make sure that the `<html>` tag does not specify a language such as `<html lang=eng>`; it should be just plain `<html>`.

Now switch to the Code/Source view and immediately below the line `<div id="midcol-right">` type the following new line to add a picture:

```

```

To Alter the Home Page Using a Plain Text Editor

In the Code/Source pane, type the alterations shown in bold type in Listing 6-1 and also delete this line: `<p>This is the content</p>` shown crossed through in Listing 6-1.

■ **Important** Ignore the left pointing arrow when typing in your code.

Listing 6-1. Alter the home page

```
<!DOCTYPE html>
<html >
<head>
    <title>Home Page</title>
    <meta charset=utf-8>
    <link href="style-4col.css" rel="stylesheet" type="text/css">
</head>
<body>
    <div id="wrapper">
        <header>
            <h1>The Rainbow Gallery</h1>
        </header>
        <div id="content">
            <div id="leftcol">
                <p>Menu</p>
                <a href=page-2.html>Go to Page 2<br>
                <a href=page-3.html>Go to Page 3<br>
                <a href=page-4.html>Go to Page 4<br>
                <a href=index.html>Home Page</a><br>
            </div>
            <div id="rightcol">
                <p>This is the far right column</p>
            </div>
            <div id="midcol">
                <h2>Home Page</h2>
                <p>This is the content</p>
                <div id="midcol-left">
                    <p>The Rainbow Gallery has a wide range of Original oil paintings, Greetings Cards ←
                    and Prints.</p>
                    <p>View and order items using this website or visit the gallery at 12 High Street, ←
                    Townsville.</p>
                </div>
            <div id="midcol-right">
                
            </div>
        </div>
    </div>
```

```

</div><!--content div finishes here-->
        <br class="clear">
    <footer>
        <p>This is the footer</p>
    </footer>
</div><!--wrapper div finishes here -->
</body>
</html>

```

Explanation of the Code

The still-life picture is inserted by using the line shown in bold type in the next code snippet:

```

<div id="midcol-right">

</div>

```

The image tag is ****. The **alt** and **title** attributes provide pop-up labels when the user's cursor is hovered over the picture. Most browsers accept *alt* but others require *title*.

The width and height of the picture are specified, and the term **src** (short for source) gives the source of the picture so that the browser can locate it.

Alter the CSS File to Style the New Home Page

We will now place a picture in the header or banner. For this you will need a panoramic picture. The *images* folder contains the panoramic image that is shown in Figure 6-3; it is 1200 pixels wide by 181 pixels high. The picture is inserted by the CSS style sheet.



Figure 6-3. The header image for this chapter

Open the CSS file *style-4col.css* in your HTML text editor in Code/Source view, and add or change the code shown in bold type in Listing 6-2. This code inserts the header picture and removes the lime-green backgrounds from the two middle columns.

Listing 6-2. Altering the CSS file

```

#wrapper {margin:auto; width:1024px; border:2px blue solid;}
header {margin-top:0; height:181px; background-image:url('images/rainbow-banner.jpg');}
h1 {position:absolute; top:45px; margin-left:30px; font-size:40pt; color:navy; ␣
    font-weight:bold;}
h2 {font-size:18pt; text-align:center;}
p {font-size:14pt; text-align:center;}
a {font-size:14pt;}

```

```
#leftcol {float:left; width:150px; height:190px; background:aqua;}
#rightcol {float:right; width:150px; height:190px; background:aqua;}
#midcol {margin-left:155px; margin-right:155px;}
br.clear {clear:both}
#midcol-left {float:left; width:46%; background:lime; font-weight:bold;}
#midcol-right {float:right; width:50%; background:lime; font-weight:bold;}
```

Explanation of the CSS Code Alterations

```
header {margin-top:0; height:181px; background-image:url('images/rainbow-banner.jpg');
```

This inserts the picture in the header. You may recall that in the previous chapters there was an unsightly gap between the top of the header and the border of the wrapper. By setting the top margin to zero, this gap is eliminated. There is no need to set the size of the margin as 0px, it can be just 0, but this is the only exception to the CSS rule for setting pixel sizes, all other sizes must add the px, for example, 5px.

By styling the header picture in the CSS file rather than in the HTML page, we are making it possible to automatically change the picture in every page of a website by just amending the single CSS file.

```
header {margin-top:0; height:181px; background-image:url('images/rainbow-banner.jpg');
```

Because the picture is 181 pixels high, I have amended the height of the header to match. The height of the header can be less than the picture height, but not more than the picture height.

The simple English term background-image is used to insert a picture, and the address of the picture is given to the browser so that it can find the picture that is in the *images* folder; the picture is named *rainbow-banner.jpg*. The URL address for the picture is therefore *images/rainbow-banner.jpg*.

```
h1 {position:absolute; top:45px; margin-left:30px; font-size:40pt; color:navy; ↵
    font-weight:bold;}
```

The code for the tag h1 is amended to accurately position the text in the header. We will be examining the positioning of page elements in more detail in Chapter 20.

```
#midcol-left {float:left; width:46%; background:lime; font-weight:bold;}
#midcol-right {float:right; width:50%; background:lime; font-weight:bold;}
```

The middle column widths have been adjusted a little to ensure that there is room to accommodate the picture *art-1.jpg*, which is 348 pixels wide. Delete the background color; it was there only for instructional purposes in the previous chapter.

The appearance of the completed home page is shown in Figure 6-1 at the start of this chapter.

Change the Heading in the Other Three Pages

In your HTML text editor, open *page-2.html*, *page-3.html*, and *page-4.html* in turn, and change the words between the <h1> and </h1> tags to *The Rainbow Gallery*.

Now view the home page *index.html* in Mozilla Firefox or any other modern browser and click the menu links to move through the four pages. The header image should now have cascaded into the other three pages.

Summary

You learned that a few small changes or amendments to the HTML code and the CSS code completely transformed the home page. You discovered how to add pictures into the header and the body of the page. You learned how to position the header image accurately using CSS and `position: absolute`;

In the next chapter you will insert some code so that Internet Explorer 8 can read HTML5 semantic tags. The home page will then look identical in both Internet Explorer 8 and any modern browser. You will discover a more versatile method of setting font sizes and learn how to make the hyperlink menu more attractive by using menu buttons.

CHAPTER 7



Enhancing the Website

In this chapter we will enhance the website that we produced in Chapter 6. The improvements consist of measures to make the site accessible to all browsers including the obsolescent Internet Explorer 8. Font sizes will be specified in percentages instead of point sizes to provide a more flexible way of adjusting fonts. Menus will be given tags that can be styled so that in future chapters, the menus can be converted to attractive buttons instead of bulleted underlined links.

In the latest HTML5 code, semantic tags replace some of the more complex `<div>` tags. These improved tags will be explained and we will use a tiny JavaScript file to enable Internet Explorer 8 to understand semantic tags.

This chapter contains the following sections:

- Allowing Internet Explorer 8 to understand semantic tags
- Add the JavaScript and some hyperlink tags that can be styled
- Explanation of the code
- Make a minor change to the CSS style sheet
- Explanation of the CSS code
- A more versatile way of setting font sizes
- Summary

For this lesson, please create a new folder to contain the code for Chapter 7. This step is important to prevent confusing the previous chapter's code with the new code.

1. In the *web-tutorial* folder, create a new folder named *Chapter-7*.
2. In the *web-tutorial* folder, go back to the *Chapter-6* folder and open it.
3. Copy all the files in the *Chapter-6* folder (*Ctrl+A* then *Ctrl+C*) and paste them (*Ctrl+V*) into the new folder *Chapter-7*.

Allowing Internet Explorer 8 to Understand Semantic Tags

Some users are reluctant to change from Windows XP or early versions of Vista; therefore they continue to use Internet Explorer 8 (IE8).

The problem with IE8 is that it does not understand the semantic tags used in HTML5.

Semantic tags have meaningful names such as `<header>` instead of `<div id="header">` and `<footer>` instead of `<div id="footer">`.

Internet Explorer 9 (IE9) does understand semantic tags but it cannot be used with Windows XP; however, IE9 was made available in Vista with Service pack 2. Fortunately this problem will not be with us for long, but meanwhile in the previous chapter, I promised to provide a solution to the IE8 problem. Because IE8 does not understand semantic tags, you must either dispense with semantic tags, or use a tiny bit of JavaScript to persuade IE8 to behave itself.

Chapter 6 stated that Internet Explorer 8 produces the unsatisfactory display shown in Figure 7-1.



Figure 7-1. A reminder of the unsatisfactory home page in Internet Explorer version 8

The next section of this chapter provides the JavaScript solution, so please take this step:

1. From *Apress.com* please download the *Chapter-7* folder and unzip it into your own *Chapter-7* folder. Assuming you have a program such as WinZip or 7-Zip, double-click the zip folder to unzip it. The downloaded folder contains the JavaScript file named *html5.js*, which enables IE8 to understand semantic tags. You don't have to understand the JavaScript. Just ensure that the unzipped JavaScript file is in your own *Chapter-7* folder. Later in the chapter you will be asked to enter some code that lets the browser IE8 know where *html5.js* is located.

■ **Note** You don't have to remember all the code in the next section because it will form part of many future templates that you produce in several future chapters. You will eventually be able to load these ready-made templates into your HTML editor and adapt them for your own websites. However, do try to understand the logic behind the code, for example, (i) in HTML5 some `<divs>` are replaced by HTML5 semantic tags, (ii) to make your pages acceptable in IE8 you need to add a little JavaScript so that IE8 understands the semantic tags, (iii) the next section will introduce you to menus constructed with unordered lists so that they can be styled.

Add the JavaScript and Some Hyperlink Tags That Can Be Styled

In this section we will insert some *conditional code*; if the code detects that a user is browsing with IE8, the conditional code loads the JavaScript file. We will use the semantic HTML5 tag `<nav></nav>` for the menu block instead of the old HTML 4 tag `<div id="leftcol"></div>`. The `<nav>` tag is shorthand for the navigation menu. We will also insert *unordered list* tags `` and `` so that we can eventually style the menu and its hyperlinks. Please follow these steps:

1. Starting with *index.html*, load each of your four pages in turn into your HTML text editor and amend the code as shown in bold type in Listings 7-1 and 7-2. The quickest way to do this would be change *index.html* and then copy and paste the changes into the other three pages.

Listing 7-1. Insert an IE8 conditional into each page of the website

```
<link href="style-4col.css" rel="stylesheet" type="text/css">
  <!--[if lte IE 8]>
    <script src="html5.js">
    </script>
  <![endif]-->

</head>
```

■ **Note** Because Internet Explorer 8 will not be with us for long, you may prefer to ignore IE8 and omit the IE8 conditional code (shown bold in the above snippet). If you choose to omit the IE8 conditional code, you will not need the JavaScript file *html5.js*, so also omit the file from your website's main folder.

For the sake of completeness, many of the projects and templates in this book contain the IE8 conditional code; therefore this note also applies to them. If you base your website on one of the templates in this book, the IE8 conditional and JavaScript file may be present but totally harmless. This is because the JavaScript will only be triggered if the user's browser is IE8.

Listing 7-2. Replace the `<div>` tags with semantic tags `<nav></nav>`, and insert unordered list tags

In *each page* of the website, just below the line `<div id="content">`, delete the two `<div>` lines shown crossed through, delete the `
` tags and insert the `<nav>`, `` and `<li class="btn">` tags shown in bold type.

```
<div id="content">
  <div id="leftcol">
  <nav>
    <ul>
    <p>Menu</p>
    <li class="btn"><a href=page-2.html>Go to Page 2</a></li>
    <li class="btn"><a href=page-3.html>Go to Page 3</a></li>
    <li class="btn"><a href=page-4.html>Go to Page 4</a></li>
    <li class="btn"><a href=index.html>Home Page</a></li>
  </ul>
  </nav>
</div>
```

2. Save the files.

Explanation of the Code

```
<!--[if lte IE 8]>
```

This piece of code is known as a *conditional*. The line is saying, “If the user’s browser is older than or equal to Internet Explorer 8, then do *something*...” On your keyboard, the keys for the square brackets are to the right of the *P* key; be careful you don’t use the Shift key when typing square brackets because you don’t want curly brackets here.

The letters *lte* stand for “less than or equal to.” There must be a space between IE and 8. Take care not to type *htm* followed by the numerals *15*. The letter between the *m* and the *5* is a lower case *L*.

```
<script src="html5.js">
</script>
```

This code creates *something* that the conditional code must do. If the user is viewing the page in Internet Explorer 8, the code instructs the browser to load a JavaScript file *html5.js*. The JavaScript file is located in the same folder as the HTML file. The extension *.js* indicates that the file is a JavaScript file. The abbreviation *src* means source. The script is surrounded by the tags `<script></script>`.

```
<![endif]-->
```

This line tells IE8 that the conditional has concluded its instruction to the browser.
Now we will examine the new menu code:

```
<nav>
  <ul>
    <p>Menu</p>
    <li class="btn"><a href=page-2.html>Go to Page 2</a></li>
    <li class="btn"><a href=page-3.html>Go to Page 3</a></li>
    <li class="btn"><a href=page-4.html>Go to Page 4</a></li>
    <li class="btn"><a href=index.html>Home Page</a></li>
  </ul>
</nav>
```

The tags `<nav>`, ``, and `` are logical abbreviations of English words as follows:

`<nav>` means **n**avigation menu, `` means **u**nordered list, `` means **l**ist-item. The new tags for the hyperlinks are ideal targets for CSS styles. In the next chapter we will use these to improve the appearance of the menu items.

We have replaced the `<div>` tag with the new semantic tag `<nav>`. The menu is now created by using an *unordered list* using `` and `` tags. You will be using unordered lists for menus in all the remaining chapters in this book. The menu block is enclosed in `` tags and each item in the list of hyperlinks has the list item tag ``. An *unordered list* means a list with the same bullet against all items in the list. Ordered bullets are in numerical or alphabetic order. See Figure 7-2.

Unordered bullets

Disc	Circle	Square
<ul style="list-style-type: none"> List item 1 List item 2 List item 3 	<ul style="list-style-type: none"> List item 1 List item 2 List item 3 	<ul style="list-style-type: none"> List item 1 List item 2 List item 3

Ordered bullets

Decimal	Upper Alpha	Lower alpha
<ol style="list-style-type: none"> List item 1 List item 2 List item 3 	<ol style="list-style-type: none"> List item 1 List item 2 List item 3 	<ol style="list-style-type: none"> List item 1 List item 2 List item 3

Figure 7-2. Unordered and ordered bullet points

```
<li class="btn">
```

In this line the list tag has been given a class called “*btn*,” which stands for *button*; you can call a class anything that is meaningful to you. It might be *butn* or *bbtn* or even *button*. The *class* is an attribute that will enable us to style the menu buttons in the next chapter. Remember that the same *class* can be used many times on a web page. Now we must make two small adjustments to the CSS style sheet.

Make a Minor Change to the CSS Style Sheet

Load the style sheet *style-4col.css* into your HTML text editor and in *Code/Source* view, insert one new line as shown in bold type in Listing 7-3; also replace the crossed-through line with the new line as shown in bold type.

Listing 7-3. Tweaking two lines in the CSS style sheet *style-4col.css*

```
#wrapper {margin:auto; width:1024px; border:2px blue solid;}
header,nav,article,section,footer {display:block;}
header {margin-top:0; height:180px;background-image:url('images/rainbow-banner.jpg');}
h1 {position:absolute; top:45px; margin-left:30px; font-size:40pt; color:navy;
font-weight:bold;}
h2 {font-size:18pt; text-align:center;}
p {font-size:14pt; text-align:center;}
a {font-size:14pt;}
/*#leftcol {float:left; width:150px; height:190px;}*/
nav {float:left; width:150px; height:190px;}
```

Explanation of the CSS Code

header,nav,article,section,footer {display:block;}

Note the words preceding the first curly bracket; they list five HTML5 semantic tags that the JavaScript will allow IE8 to understand. Without this line of code the JavaScript won't function. Don't worry if you find that particular code difficult to understand. Just accept it; and in any case when IE8 has become extinct, the JavaScript will no longer be needed.

```
/*#leftcol {float:left; width:150px; height:190px;}*/  
nav {float:left; width:150px; height:190px;}
```

We no longer need #leftcol because the semantic tag *nav* will do nicely as a target for setting the width of the menu block. Therefore we have replaced the line shown crossed through with the new <nav> line shown in bold type. The new line tells the browser to target the *nav* tag in the HTML page.

Now try loading the web page in IE8 to see the result. Open the folder *Chapter-7* and right-click the file *index.html* (if you don't have IE8 on your computer, you may be able to load the folder *Chapter-7* into another computer that does have Internet Explorer 8.) Now use the menu to move from page to page.

■ **Note** Depending on how your IE8 browser is configured, you might see a cream-colored bar at the top saying that IE8 has restricted the file. Click the cream bar and select *Allow blocked content*.

Figure 7-3 shows the menu that you should see in your browser whether you use IE8 or any of the more modern browsers.

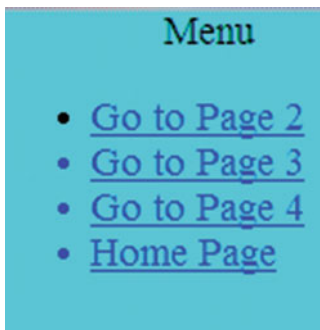


Figure 7-3. The menu is an unordered list

You will see that the menu's appearance has not changed, but in the code we now have some tags and <li class="btn"> that we can style by means of CSS. We will do this in the next chapter, where we will also use CSS to improve the appearance of the hyperlinks and introduce a colored rollover effect to the menu buttons.

A More Versatile Way of Setting Font Sizes

By using CSS, the font sizes can be set by many methods such as: points, pixels, ems, percentages, and keywords. Previous chapters used points such as `font-size:14pt`; I did this to protect you from information overload, and therefore I used a font sizing system that is familiar to anyone who has used a word processor. Unfortunately, fonts that are set in points can cause unexpected results.

CSS font sizes can be set by keywords, but they provide only seven sizes as follows:

`xx-small`, `x-small`, `small`, `medium`, `large`, `x-large` and `xx-large`

We could use the key word *medium* to set the general font size within the *wrapper* area of a page. The *medium* font size sets the font to the native (default) size for the browser. We could use *small* for less important words or paragraphs. However, there is a better way.

If we first set the base size for the *wrapper* area of a page to either *medium* or *100%*, we can then use *percentages* to provide an infinite range of font sizes within the *wrapper* area.

As an example, in your HTML text editor please alter the CSS file *style-4col.css* using the changes shown in bold type in the following listing:

```
#wrapper {margin:auto; width:1024px; border:2px blue solid; font-size:medium;}
header,nav,article,section,footer {display:block; }
header {margin-top:0; height:180px;background-image:url('images/rainbow-banner.jpg');}
h1 {position:absolute; top:45px; margin-left:30px; font-size:250%; color:navy; font-weight:bold;}
h2 {font-size:150%; text-align:center;}
p {font-size:120%; text-align:center;}
a {font-size:120%;}
nav { float:left; width:150px; height:190px; }
```

The percentages multiply the base size (*medium* or *100%*) so that for instance, the `h1` size (above) is 250% larger than the base size (medium or 100%), and the paragraph `<p>` size is set to 120% larger than the base size.

Now see how the pages look in a modern browser. If they are satisfactory, save the amended file and move to the next chapter

Summary

So far, we have been gradually building a template for a four-page website. The template contains all the code described in the first seven chapters of the book. In this chapter you were presented with a section describing how to overcome the deficiencies of Internet Explorer 8 by applying a small conditional JavaScript file. You learned how to use unordered (bulleted) lists for menus so that the HTML can provide tags for styling the hyperlinks. A more versatile method for sizing fonts was demonstrated.

In the next chapter you will discover how to tweak the CSS to improve the menu's appearance. By using CSS to target the `<li class="btn">` tags, you will be able to create colored "rollover" menu buttons. "Rollover" means that the buttons will change color when the mouse is hovered over them.

CHAPTER 8



Rollover Menu Buttons

Although they worked well, the menus in previous chapters were very plain and not very professional. In this chapter we will use CSS to improve the menu's function and appearance. This chapter contains the following sections:

- Improving the appearance of the menu hyperlinks
- Explanation of the code
- Add the rollover feature
- Create 3D buttons with a rollover feature
- Insert 3D-colored menu buttons into a web page
- Summary

For this lesson you need to create a new folder to contain the code for Chapter 8. Please follow these steps:

1. In the *web-tutorial* folder, create a new folder named *Chapter-8*.
2. In the *web-tutorial* folder, go back to the *Chapter-7* folder and open it.
3. Copy all the files in the *Chapter-7* folder (Ctrl+A then Ctrl+C).
4. Paste them into the new folder *Chapter-8* using (Ctrl+V).

Improving the Appearance of the Menu Hyperlinks

Menu Buttons can be plain-colored buttons or three dimensional (3D)-colored buttons (as shown in Figures [8-1a](#) and [8-1b](#)).



Figure 8-1a. Plain buttons

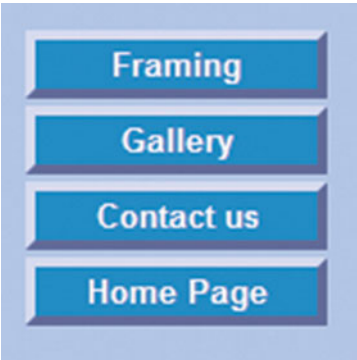


Figure 8-1b. 3D buttons

■ **Note** You don't have to remember all the code in this chapter because it will form part of most templates that you produce in future chapters. You will eventually be able to load those ready-made templates into your HTML editor and adapt them for your own websites. However, do try to understand the logic behind the code, for example, (i) flat plain-colored and 3D buttons can be produced by CSS code that targets an unordered list in the HTML pages, and (ii) the rollover color changes are produced by CSS code that targets the menu's hyperlinks.

Figure 8-2 shows a home Page with a menu with flat-colored buttons and a rollover feature.



Figure 8-2. The plain gray-colored buttons become blue when the cursor hovers over them

We will use some CSS styling to achieve the following three improvements:

Remove the bullets

Remove the underlines

Change the plain hyperlinks into colored buttons

To achieve this please follow these steps:

1. Open the file *index.html* in your text editor and use *Save As* to save a copy named *index-plain.html*.
2. Then make some minor changes in the file *index-plain.html* as shown in bold type in Listing 8-1.

Listing 8-1. Make some small changes in the file *index-plain.html*

```
<!DOCTYPE html>
<html>
  <head>
    <title>Home Page</title>
    <meta charset=utf-8>
    <link href="style-4col.css" rel="stylesheet" type="text/css">
    <link href="style-plain.css" rel="stylesheet" type="text/css">
    <!--[if lte IE 8]>
      <script src="html5.js">
    </script>
    <![endif]-->
  </head>
<body>
  <div id="wrapper">
    <header>
      <h1>The Rainbow Gallery</h1>
    </header>
    <div id="content">
      <nav>
        <ul>
          <p>Menu</p>
          <li class="btn"><a href=#>Go to Page 2</a></li>
          <li class="btn"><a href=#>Go to Page 3</a></li>
          <li class="btn"><a href=#>The firstGo to Page 4</a></li>
          <li class="btn"><a href=#>Home Page</a></li>
        </ul>
      </nav>
```

To save space and time, the menu has been changed by using `` so that the plain button links don't work. This will save you the chore of modifying all the pages; and also, if you click a link you will not see the "Page not available" error message.

The first change in Listing 8-1 illustrates an important point regarding links to style sheets.

```
<link href="style-4col.css" rel="stylesheet" type="text/css">
<link href="style-plain.css" rel="stylesheet" type="text/css">
```


We have two style links, so why don't they conflict? They do not conflict because browsers always obey the *last* styling link in a list of links.

Remember the phrase *the last style wins*. The second style can be a small file that only alters the style of one or two HTML elements. In our example the first linked CSS sheet will style the whole page, but only the menu styling will be overridden by the second linked CSS sheet.

3. Create a new file *style-plain.css* in the code view of your HTML text editor and insert the lines shown in bold type in Listing 8-2.

Listing 8-2. Improve the appearance of the menu buttons

```
/*set the styles for the side menu column */
nav ul {float:left; width:150px; margin-left:10px; list-style-type:none;}

/* set general side button styles */
li.btn {width:115px; line-height:20px; margin-bottom:3px; text-align:center;}

/* set general access (anchor) styles */
li.btn a {display:block; width:115px; color: white; background:gray; ↵
font-family:arial; font-size:small; font-weight:bold; text-decoration:none;}
```

View the file *index-plain.html* in your browser by, either going to the folder in Chapter 8 and right-clicking *index-plain.html*, then selecting a modern browser; or in Expression Web, press F12. You should see that the menu buttons no longer have bullets or underlines and the buttons are gray rectangles.

Explanation of the Code

```
/*set the styles for the side menu column */
nav ul {float:left; width:150px; margin-left:10px; list-style-type:none;}
```

The item **nav ul** means: style all HTML pages that have **ul** tags contained within a **<nav>** tag.

The code **list-style-type:none;** removes the bullets from the *unordered list* (refer back to Figure 7-3 in the previous chapter to see the bullets that will be removed).

```
/*set general side button styles */
li.btn {width:115px; line-height:20px; margin-bottom:3px; text-align:center;}
```

This style targets all list items **li** in the web pages that have a class named **btn**. It gives those items a width and height to produce rectangular buttons. The style also sets a bottom margin for the buttons so that they are separated by a gap of 3 pixels. The text on the buttons is centered.

```
/*set general access (anchor) styles */
li.btn a {display:block; width:115px; color:white; background:gray; font-family:arial; ↵
font-size:small; font-weight:bold; text-decoration:none;}
```

The code **li.btn a** styles all the list items **li**, but only if they have a class of **btn** and if the list contains a hyperlink **<a**.

Normally, to activate a hyperlink, the user must be careful to click the underlined label of the link. The new buttons use **li.btn a {display:block;}** which allows the user to activate the link by putting the cursor on *any* part of the button. The hyperlink applies to the whole button, not just to the text on the button. The text on the buttons is styled as Arial, white, bold, and a font size named *small*. The background of the buttons is set to gray (the USA spelling **gray** must be used).

The style **display:block;** ensures that the buttons fill the width and height specified for the buttons. It also ensures that the buttons will all be the same width even if the labels are different lengths.

The style **text-decoration:none;** removes the underlines from the hyperlinks.

Add the Rollover Feature

The plain buttons do not yet have the rollover feature, so we will add that next.

Rollover menus provide an interactive element. The colored buttons need a little more CSS code so that they change color when the mouse is hovered over them. The color can also change when the mouse button is held down by the menu button, although this is not essential and can be omitted from the code if you wish.

In the code/source view of your HTML text editor, load *style-plain.css* and add the following rollover styles to the end of the code as shown in Listing 8-3.

Listing 8-3. Adding the rollover feature

```
/* mouseover */
li.btn a:hover {background: blue;}

/* mousedown (optional)*/
li.btn a:active {background:green;}
```

The *mouseover* code decrees that hovering the mouse over a menu button causes it to change to blue.

The *mousedown* code decrees that holding the button down with the mouse causes the button to change to green.

View the page *index-plain.html* in your browser by either (i) going to the folder Chapter 8 and right-clicking *index-plain.html*, then selecting *Open with* and choose a modern browser, or (ii) in Expression Web, press F12. You should see that hovering the mouse pointer over the menu buttons produces the rollover color change.

Create 3D Buttons with a Rollover Feature

When you have completed the following section, the 3D rollover menu buttons should look like Figure 8-3.

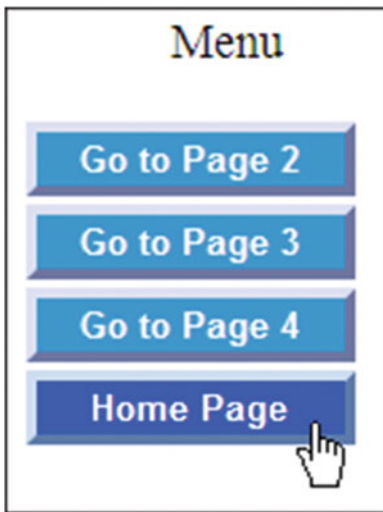


Figure 8-3. 3D rollover menu buttons

We will now use CSS to create a 3D effect on colored menu buttons.

To achieve this please follow these steps:

1. In the code/source view of your HTML text editor, open the file *style-plain.css* and use *Save As* to save a copy and name it *style-3d.css*.
2. Delete the items shown crossed through in Listing 8-4.
3. Add the last three lines as shown in bold in Listing 8-4.

Listing 8-4. Create a style sheet for 3D rollover menu buttons

```
/*set vertical button menu column*/
nav ul {float:left; width:150px; margin-left:10px; padding-left:0; list-style-type:none;}
/*set general side button styles*/
li.btn {width:115px; line-height:20px; margin-bottom:3px; text-align:center;}
/* set general access (anchor) styles */
li.btn a {display:block; width:115px; color:white; background:gray; font-family:arial;
font-size:small; font-weight:bold; text-decoration:none;}
/* mouseover */
li.btn a:hover {background:blue;}

/* mousedown (optional)*/
li.btn a:active {background:green;}

/* mouseout (default) */
li.btn a {background: #1A9CE0; border:4px outset #AABAFF;}
/* mouseover */
li.btn a:hover {background:#0A4ADF; border:4px outset #8ABAFF;}
/* mousedown (optional)*/
li.btn a:active {background:#AECBFF; border:4px inset #AECBFF;}
```

4. Save the file *style-3d.css*.

Insert 3D-Colored Menu Buttons into a Web Page

Now we need to add the new 3D menu style to a new index named *index-3d.html*.

To add the link to the new style sheet, please follow these steps:

1. In the code/source view of your HTML text editor, load *index.html*.
2. Save copy of the file with the new name *index-3d.html*.
3. Add the new style links shown in bold type in the next code snippet:

```
<link href="style-4col.css" rel="stylesheet" type="text/css">
<link href="style-plain.css" rel="stylesheet" type="text/css">
<link href="style-3d.css" rel="stylesheet" type="text/css">
```

4. Save the file.
5. View the file *index-3d.html* in your browser by either going to the folder Chapter 8 and right-clicking *index-3d.html*, then selecting *Open with* and choosing a modern browser. Or in Expression Web, press F12. You should see that the menu buttons now have a 3D appearance and a rollover feature. Once again, the last style in the list of links wins (remember that in CSS, *the last style wins*). The second linked style is overridden by the third linked style sheet.

■ **Tip** You can change between 3D buttons and flat buttons and vice versa simply by switching the relevant style link so that it comes last in the list of links.

Explanation of the Code

Remember that comments in CSS files use the tags */*some comment*/*

```
li.btn a {display:block; width:115px; color:white; background:gray; font-family:arial; ␣
        font-size:small; font-weight:bold; text-decoration:none;}
```

Delete the background color as shown, which is crossed through. This is because we are going to change the colors of the buttons and amend the next few lines to create the rollover effect.

```
/* mouseout (default) */
li.btn a {display:block; background:#1A9CE0; border:4px outset #AABAFF;}
```

The new background code has a color code **#0A4ADF** instead of a name such as gray or blue. Don't feel daunted by this code; just accept for the moment that the code stands for a particular color. The letters can be upper or lowercase. In Chapter 9 the CSS color codes will be fully explained, and they are more logical than they appear at first sight. The code **border:4px** adds a 4 pixels wide border to the menu buttons. The code **outset #AABAFF;** provides a top border and a left border with a much lighter color than the other two borders; this makes the buttons appear to be standing out in 3D.

```
/* mouseover */
li.btn a:hover {display:block; background:#0A4ADF; border:4px outset #8ABAFF;}
```

When the cursor hovers over a button the background and the borders change color.

```
/* mousedown (optional) */
li.btn a:active {background:#AECBFF; border:4px inset #AECBFF;}
```

When the cursor is placed over a button and the left mouse button is held down, the background and the borders change color. The code **inset #AECBFF;** produces the effect of a button that has been pushed in.

■ **Note** Chapter 41 contains the code for a selection of different-colored 3D buttons.

You have created the two index files, *index-plain.html* and *index-3d.html*. We will now create a file named *index.html* to get ready for the projects in Chapters 9 through 11. Because those projects will only use the 3D buttons, we will combine the button styles with the main style sheet for convenience. We will also reactivate the other three pages so that you can see how the 3D button styles cascade into them.

We need to amalgamate the small CSS file *style-3d.css* with the main style sheet *style-4col.css*. To avoid confusion we will call the amalgamated style sheet *style-3d.css*, which will overwrite the current file with the same name. If you wish to keep a copy of the basic 3d style file, open *style-3d.css* and use *Save As* to save a copy with the name *style-3d-basic.css*.

Whether you decide to save a basic copy or not, please follow these steps:

1. Open the file *style-4col.css* in your text editor and copy the contents into memory (Ctrl+A then Ctrl+C).
2. Open file *style-3d.css* in your text editor and using (Ctrl+V) paste the content of *style-4col.css* into the beginning of the file *style-3d.css* as shown in bold in Listing 8-5.

Listing 8-5. Create the amalgamated style sheet *style-3d.css*

```
#wrapper {margin:auto; width:1024px; border:2px blue solid; font-size:medium; ↵
background-color:white;}
header,nav,article,section,footer {display:block;}
header {margin-top:0; height:180px; background-image:url('images/rainbow-banner.jpg');}
h1 {position:absolute; top:45px; margin-left:30px; font-size:250%; color:navy; ↵
font-weight:bold;}
h2 {font-size:150%; text-align:center;}
p {font-size:120%; text-align:center;}
a {font-size:120%;}
nav {float:left; width:150px; height:190px; }
#rightcol {float:right; width:150px; height:190px; background:aqua;}
#midcol {margin-left:155px; margin-right:155px;}
br.clear {clear:both}
#midcol-left {float:left; width:46%; font-weight:bold;}
#midcol-right {float:right; width:50%; font-weight:bold;}
/*set vertical button menu column*/
nav ul {float:left; width:150px; margin-left:10px; padding-left:0; list-style-type:none;}
/* set general side button styles*/
li.btn {width:115px; line-height:20px; margin-bottom:3px; text-align:center;}
/* set general access (anchor) styles*/
li.btn a {display:block; width:115px; color:white; font-family:arial; font-size:small; ↵
font-weight:bold; text-decoration:none; }
```

```

/* mouseout (default) */
li.btn a {background:#1A9CE0; border:4px outset #AABAFF;}
/* mouseover */
li.btn a:hover {background:#0A4ADF; border:4px outset #8ABAFF;}
/* omousedown (optional)*/
li.btn a:active {background:#AECBFF; border:4px inset #AECBFF;}

```

3. Save the file with the name *style-3d.css*.

Now that the 3D button style and the main style sheet are amalgamated, we can dispense with the extra styles in readiness for Chapters 9 through 11, we also need to replace the dead links with live links to the other three pages. Please ensure that the only style link is the one linking to the amalgamated style sheet. The link is shown in bold in Listing 8-5. Please follow these steps:

1. Open the file *index.html* in your text editor and amend the code as shown in bold in Listing 8-6.

Listing 8-6. Change the index.html file (only the relevant part of the file code is shown)

```

<!DOCTYPE html>
<html>
  <head>
    <title>Home Page</title>
    <meta charset=utf-8>
    <link href="style-3d.css" rel="stylesheet" type="text/css">
      <!--[if lte IE 8]>
        <script src="html5.js">
        </script>
      <![endif]-->
  </head>
<body>
  <div id="wrapper">
    <header>
      <h1>The Rainbow Gallery</h1>
    </header>
    <div id="content">
      <nav>
        <ul>
          <p>Menu</p>
          <li class="btn"><a href=page-2.html>Go to Page 2</a></li>
          <li class="btn"><a href=page-3.html>Go to Page 3</a></li>
          <li class="btn"><a href=page-4.html>Go to Page 4</a></li>
          <li class="btn"><a href=index.html>Home Page</a></li>
        </ul>
      </nav>

```

2. Save the file as *index-2.html*
3. In each of the other pages, change the style sheet link so that it links to *style-3d.css* instead of *style-4col.css*, the link should be as follows:

```

<link href="style-3d.css" rel="stylesheet" type="text/css">

```

4. Save each of the pages after changing them.
5. View all four pages in your modern browsers, you should see that the 3D menu has cascaded into all the pages.

Summary

In this chapter you learned how to create menu buttons with flat colors, and you then added a rollover feature to those buttons. You were then shown how to create 3D rollover menu buttons. You learned that the head section of an HTML page may contain several linked styles. You also discovered the CSS rule that *the last style wins*. You were introduced briefly to CSS color codes.

In the next chapter you will learn how to choose suitable colors and discover how the CSS color codes work and how they can produce many new subtle shades of each color for web page backgrounds and menu buttons.

CHAPTER 9



Using Color Effectively

The proper use of color can affect the user's decision about whether to stay with your website or switch to a more appealing one. Color can affect the mood of a website and also reflects the purpose of the site.

This chapter contains the following sections:

- Choosing color schemes
- How website colors are produced
- Understanding the CSS color codes
- Tutorial: Adding more color to our website
- Using color pickers on WYSIWYG Text Editors
- Tweaking colors
- Summary

Choosing Color Schemes

The appropriate choice of color forms a vital aspect of a successful website design: in other words, the colors must be chosen to match the theme of the website. For instance, I was asked to take over a carpet cleaner's website that had been produced by an untrained amateur. The original site was predominantly gray and black, hardly suitable for a cleaner's site. I used pale pastel colors and white in the revised website to convey the theme of cleanliness.

I was also asked to look at the possibility of redesigning a website for a nature conservation society. In the original, the main background color was peach shading to orange, and the pages were plastered with a bewildering amount of colored links and thumbnail pictures in various colors. Common sense dictates that a nature conservation society's site should predominantly be made up of various shades of green.

The careful use of color can increase the appeal of a website, but some colors and combinations of colors are positively repellent. Sharp background colors such as bright red are irritating if they cover large areas of the page. If your site is selling something, use bright but not garish colors, and use them sparingly for the products or the menu buttons. Tasteless and amateurish websites seem to prefer a garish color mix. If you are not selling something but you are providing information, use backgrounds of pale pastel colors to create calm; the visitor will linger longer on a calm website.

Dark background colors are sinister and can be repellent (especially black, which is really only suitable for a funeral parlor or a witch's coven; also white or colored text tends to bleed into a black background making the text difficult to read).

If the client has a house style with a color scheme, or a colored brochure, or a colored logo, those items would make a good starting point for a color scheme. Showing a color palette to clients can also help them to choose a theme.

Ensure that the text and background color have enough contrast to enable partially sighted persons to read your website easily. The best text color for clarity and ease of reading is black text on a white background, and you should definitely avoid silver or pale blue text on a white background.

■ **Tip** Need ideas for colors? Try the following websites:

www.w3schools.com/css/css_colornames.asp

<http://somacon.com/p142.php>

<http://colorshemedesigner.com>

<http://colorshemer.com> or <http://www.elizabethcastro.com/html/colors/backflapcolors.html>

<http://www.december.com/html/spec/color.html>

<http://www.paciellogroup.com/resources/contrast-analyser.html>

How Website Colors Are Produced

If you have ever used paints you will know that mixing certain colors produces new colors. Well you can forget that for now because mixing colors for websites is quite different; this is because on websites you will not be mixing pigments but mixing different wave lengths of light. The primary colors of paints and light are different; paint primary colors are red, yellow, and blue; but primary colors in a beam of light are red, green, and blue. When you mix the three primary color paint pigments in equal amounts you produce something like the color of mud. In the mid-19th century, James Clerk Maxwell, the famous Scottish scientific mathematician, published his thesis on color in which he proclaimed that white light consisted of equal amounts of the three primary colors: red, green, and blue. Maxwell's discovery is the basis of all website colors.

■ **Tip** If you are reading this chapter in the printed version of this book, Figures 9-1 and 9-2 will not be in color and therefore are not very informative. However, I have included a PDF document in the downloads for this chapter so that you can view the colors.

■ **Note** If you feel daunted by that bit of physics, skip it and just refer to the color chart for light shown in Figure 9-1.



Figure 9-1. Light consists of the three primary colors: red, green, and blue

In this diagram, none of the primary colors exist outside the colored circles; if no primary color is present, the result is black.

Where all the primaries are mixed in equal quantities, the result is white.

Red	Green	Blue
Fuchsia	Lime	Aqua
Maroon	Olive	Navy
Orange	Yellow	Purple
Silver	Black	White
Gray		

Figure 9-2. The named colors. The CSS code for red would be `{color:red;}`

Figure 9-1 shows eight colors, and many new colors can be made: for instance, mixing some red with the pale blue will result in mauve. By mixing the colors in various proportions you can design over six million colors. CSS allows you to mix the colors easily by means of RGB code or hexadecimal code.

So how do we remember the colors and their codes? We don't have to remember them because CSS provides 142 colors with names like `color:red`; and the Internet has boatloads of charts showing the colors and their codes.

Understanding the CSS Color Codes

In the past, 17 basic CSS colors could be coded by name, and these are as shown in Figure 9-2:

■ **Tip** One hundred and twenty-three more colors have now been given names. For a vast number of named colors, visit <http://www.somacon.com/p142.php>. Then scroll down to the section headed General Purpose Colors. There you will find the colors, together with their names and hexadecimal codes.

Color codes can have four formats but we will only mention RGB code briefly and then we will concentrate on hexadecimal code. RGB code means red, green, blue code.

RGB color code uses intensities of the primaries, and the intensity of a primary color ranges from 0 to 255. For instance, the code for the color black is `color:rgb(0,0,0)`, white is `color:rgb(255,255,255)`, red is `color:rgb(255,0,0)`, purple is `color:rgb(128,0,128)`; and in this case the intensity of the red and blue is low, and the low light intensity results in a darker color. The order of the intensities inside the brackets is red, green, blue (r,g,b) as you would expect. As previously stated, you don't need to remember the details, just try to understand the logic.

Hexadecimal color code is shorter than RGB and is therefore more frequently used by web designers. It still relies on red, green, and blue in various intensities, but for a color called coral you could use some hexadecimal code like this: `color:#FF7F50`. The primary colors are in three groups of two letters or numbers per group and the three groups are in the order red, green, blue. For instance, in the first group FF is intense red, in the second group 7F is medium-intensity green (maximum intensity would have been FF), and in the third group 50 is low-intensity blue (maximum intensity would have been FF).

If you feel that the next bit of math is confusing you can skip it, but read it if you find codes interesting. Hexadecimal numbers use 6 characters. They can consist of 10 numbers (including zero) and 6 letters, and these relate to our normal base 10 numbering system as follows:

Base ten	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Hexadecimal	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

- A primary color with no intensity would be coded as 00.
- A primary color with maximum intensity would be coded as FF.
- If all three primary colors have absolutely no intensity the result would be black, therefore black is coded as `color:#000000`;
- If all three primary colors are at maximum intensity the code would be `color:#FFFFFF`; because each primary has maximum intensity, the resulting color is white.

Tutorial: Adding More Color to our Website

For this lesson you need to create a new folder to contain the code for Chapter 9. This step is important to prevent confusing the previous chapter's code with the new code.

Please follow these steps:

1. In the *web-tutorial* folder, create a new folder named *Chapter-9*.
2. In the *web-tutorial* folder, go back to the *Chapter-8* folder and open it.
3. Copy all the files (Ctrl+A) then (Ctrl+C).
4. In the *Chapter-8* folder, paste the files into the new folder *Chapter-9* using (Ctrl+V).

In the next section we will add more color to the web pages as shown in Figure 9-3.

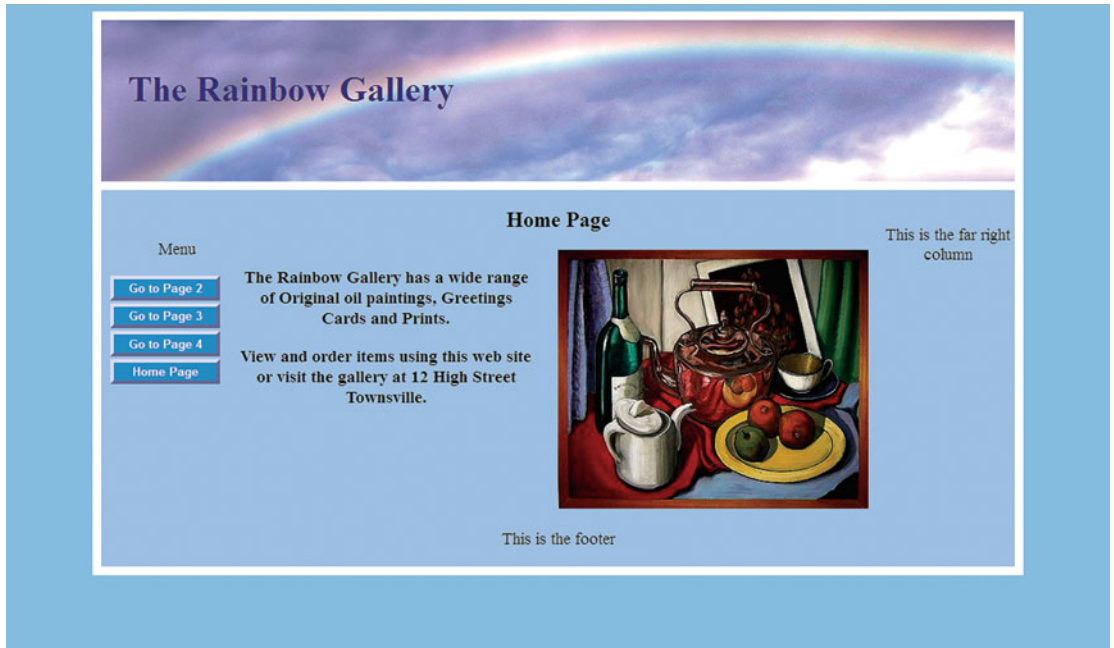


Figure 9-3. More colors have been added

To add the new colors, please follow these steps:

1. Use your HTML text editor to open the file *style-3d.css* and simply add a little extra styling as shown in bold type in listing 9-1. Delete the item that is crossed through.

Listing 9-1. Creating more colors in the file *style-3d.css*

```
body {background:#88CCFF;}
#wrapper {margin:auto; width:1024px; background:#A6CCFF; border:10px white solid;
    font-size:medium;}
header,nav,article,section,footer {display:block;}
header {margin-top:0; border-bottom:10px white solid; height:181px ↵;
    background-image:url('images/rainbow-banner.jpg');}
h1 {position:absolute; top:45px; margin-left:30px; font-size:250%; color:navy;↵
    font-weight:bold;}
h2 {font-size:150%; text-align:center;}
p {font-size:120%; text-align:center;}
a {font-size:120%;}
nav {float:left; width:150px; height:190px; }
#rightcol {float:right; width:150px; height:190px; background:aqua;}
#midcol {margin-left:155px; margin-right:155px;}
br.clear {clear:both}
#midcol-left {float:left; width:46%; font-weight:bold;}
#midcol-right {float:right; width:50%; font-weight:bold;}
```

```

/*set vertical button menu column*/
nav ul {float:left; width:150px; margin-left:10px; padding-left:0; list-style-type:none;}
/* set general side button styles*/
li.btn {width:115px; line-height:20px; margin-bottom:3px; text-align:center;}
/* set general access (anchor) styles*/
li.btn a {display:block; width:115px; color:white; font-family:arial; font-size:small;␣
        font-weight:bold; text-decoration:none; }
/* mouseout (default) */
li.btn a {background:#1A9CE0; border:4px outset #AABAFF;}
/* mouseover */
li.btn a:hover {background:#0A4ADF; border:4px outset #8ABAFF;}
/* omousedown */
li.btn a:active {background:#AECBFF; border:4px inset #AECBFF;}

```

2. Save the file.

Explanation of the Code

body {background:#88CCFF;}

This additional style fills the entire screen with a solid color that I chose from an online list of colors at <http://www.somacn.com/p142.php>.

```

#wrapper {margin:auto; width:1024px; background:#A6CCFF; ␣
        border:10px white solid; font-size:medium;}

```

The wrapper is filled with a slightly different colors chosen from an online list. Also the border of the wrapper is widened to 10 pixels and is colored white so that it shows against the blue background.

```

header {margin-top:0; border-bottom:10px white solid; height:181px;␣
        background-image:url('images/rainbow-banner.jpg');}

```

The header is given a 10 pixel white bottom border.

```

#rightcol {float:right; width:150px; height:190px; background:aqua;}

```

Delete the aqua background color of the far-right column because it clashes with the other colors.

Test the amended pages by either loading index.html into your HTML text editor and pressing F12 in Expression Web 4, or by clicking the blue globe in Blue Griffon. If you are using a non-WYSIWYG editor go to the folder Chapter 9 and right-click index.html and choose to open the file with a modern browser. Click the hyperlinks in the menu to move from page to page. The new style should cascade through all the pages.

Using Color Pickers on WYSIWYG Text Editors

In Expression Web 4 when compiling a CSS style sheet, if you type the word *background*, a color picker will pop up as shown in Figure 9-4.

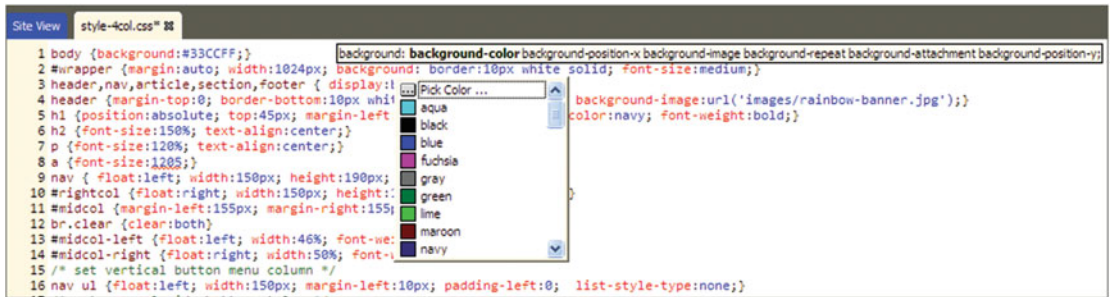


Figure 9-4. The Expression Web 4 color picker

You can click one of the 17 named colors, or for a wider range of colors click the topmost item Pick Color. You will then see a more comprehensive selection of colors as shown in Figure 9-5.

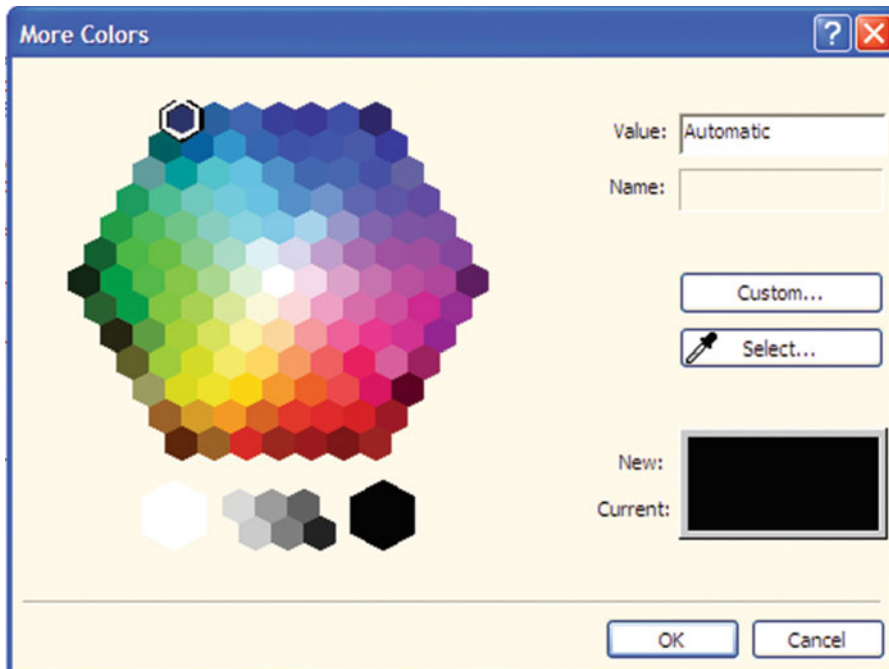


Figure 9-5. The extended color picker in Expression Web 4

If you click on a color and then click OK, the color code will be inserted into your CSS file.

In Blue Griffon: When a colored page is opened with Blue Griffon in WYSIWYG view, you will see two colored buttons as shown circled in Figure 9-6.

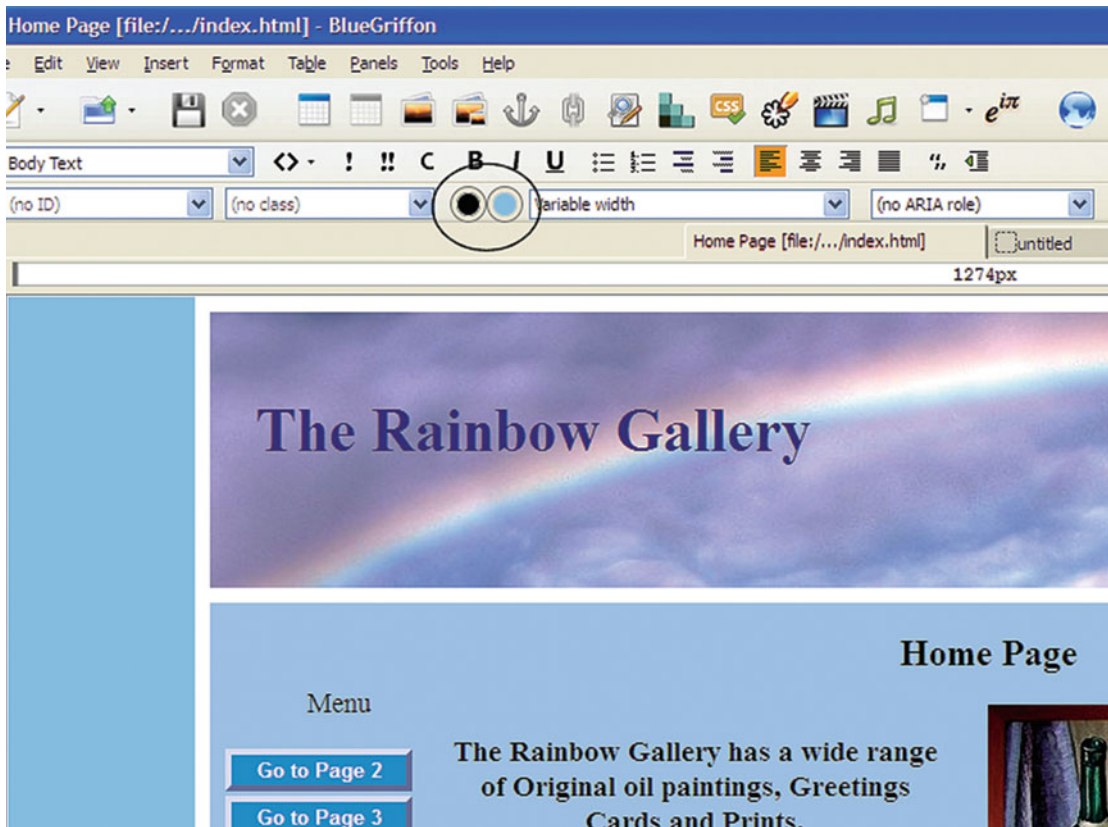


Figure 9-6. Showing the two colored buttons (circled) in Blue Griffon

The left button shows the color of the foreground (the text in the content section of the page).

The right button shows the background color of the body section of the page.

If you click one of the two buttons you will see a pop-up color picker as shown in Figure 9-7.

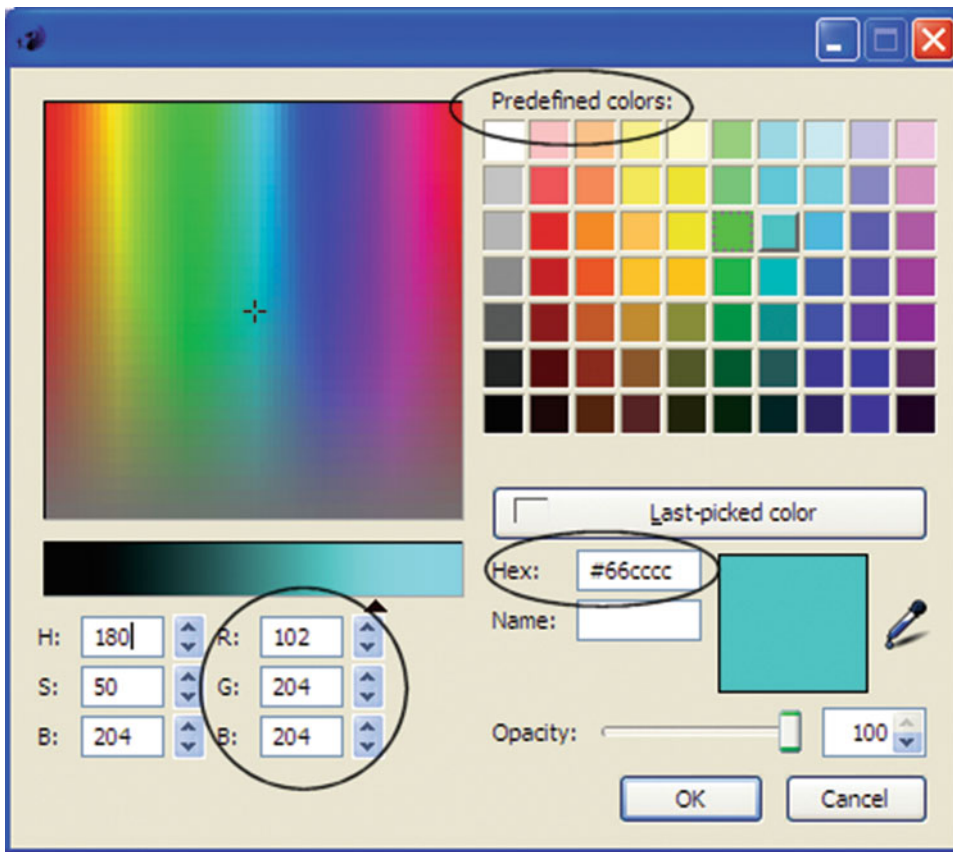


Figure 9-7. *The Blue Griffon color picker*

If you click on one of the 70 predefined colors you will see the RGB color code and also the hexadecimal color code. Make a note of the code and enter it into the appropriate place in your CSS style sheet. Move your cursor over the colors in the big panel (top left) to see the color codes (circled) change.

Tweaking Colors

Web designers have complete control over colors because color codes are infinitely adjustable. For instance, for this chapter, I chose a suitable background color from an online list. However, when I viewed the new background color in a browser it was a little too vivid. This can happen because the other colors in a display can affect your perception of the new background color. This is no problem because color codes are easy to tweak. Make a careful note of the color code before tweaking it so that you can return to it if you mess it up. The color I initially used was #33CCFF; it needed toning down a bit. I figured that adding more intensity to the red element might do the trick, so I changed 33 to 86. It was still not quite satisfactory so I changed the 88 to A6 giving a code #A6CCFF; I was then satisfied.

You can darken a color by decreasing intensity of the numbers and letters; for instance, pure blue is #0000FF; when FF is decreased to 80, it becomes #000080, which is navy blue.

To make a color lighter, increase the hexadecimal characters. For instance, royal blue is #0000FF; a paler version of the blue would be #7070FF.

Summary

This chapter contained guidelines for using color effectively in websites. You learned that a vast range of colors are completely under the control of the web designer. You learned the meaning of the color codes. You were told that online charts and color pickers are available to help you chose colors and code them. You learned how a small number of tweaks to the CSS file can completely transform the appearance of a page.

In the next chapter you will learn how to cope with the multitude of screen sizes that will be used to view your websites.

CHAPTER 10



Screen Size and Resolution: Useful Templates: Rounded Corners

This chapter describes a way of coping with the multitude of screen sizes and resolutions. It also provides a method for creating rounded corners on borders. You will also create four useful templates.

The chapter contains the following sections:

- Screen sizes and screen resolutions
- Other monitor-related considerations
- Will the website work on a handheld device?
- Examining the three possible layouts
- Create a template with a 3D menu for use in your future websites
- Creating a template with a border with rounded corners
- Creating two templates with plain button menus
- Summary

For this lesson you need to create a new folder to contain the code for Chapter 10. This step is important to prevent confusing the previous chapter's code with the new code.

1. In the web-tutorial folder, create a new folder named *Chapter-10*.
2. In the web-tutorial folder, go back to the *Chapter-9* folder and open it.
3. Copy all the files in the *Chapter-9* folder (use Ctrl+A then Ctrl+C) and paste them (using Ctrl+V) into the new folder *Chapter-10*.

Screen Sizes and Screen Resolutions

Web designers cannot possibly know the size of every user's screens or how their resolution is set up. Elderly folk with fading eyesight may set a 19-inch screen to 800 pixels wide by 600 pixels high, preferring big, blurred icons and fonts to small sharp icons and fonts. One pixel is the size of a single light cell on a screen. Look at a screen with a magnifying glass and you will see the individual cells. Table 10-1 shows some of the common screen resolutions for displaying websites.

Table 10-1. *Some of the resolutions currently in use (pixels: horizontal x vertical)*

4 x 3 ratio screens	1024 x 768	1152 x 864	1600 x 1200	1920 x 1440	2560 x 3070
Wide screens	1280 x 800	1600 x 900	1920 x 1080	2560 x 1440	3200 x 1800

When you design a website, the screen size and resolution you choose will always be a compromise; you just can't win. This chapter will suggest the best compromise; however, the resolution is not the only factor affecting the appearance of your pages on a user's screen.

Some users have a Favorites pane permanently pinned to the browser window, which reduces the usable width of the browser window; this causes the user to scroll horizontally. Most sites have both long and short pages. On short pages, no vertical scroll bar is displayed. On long pages, the vertical scroll bar appears and the page may move sideways a little so that items appear to jump around as the user moves from short pages to long pages. The real screen area can be less than the stated number of pixels for several reasons. Users lose vertical screen space by adding extra and unnecessary search bars, such as those inserted by Google, Yahoo, or AVG anti-virus. The Windows vertical scroll bar takes up some of the screen width.

The area within a browser's borders is called the viewport.

Other Monitor-Related Considerations

Other considerations can determine the layout of a website. You will need to consult your clients about the equipment they are using and their future intentions concerning that equipment. The advent of new handheld devices with different operating systems and browsers make the web designer's task even more difficult.

When designing a website for an office or a factory's internal use, you will need to visit the premises to determine the screen sizes and resolutions used by the staff. The employees will not be pleased if they have to continually scroll horizontally to access items on a page. Should the company invest in new equipment with bigger screens, you will need to change the wrapper's width to match. You may have to put the text into an increased number of columns for ease of reading.

Will the Website Work on a Handheld Device?

Handheld is a term describing mobile phones and tablets. Designing for handheld devices was once a major problem because the devices at first had resolutions as low as 170 pixels x 220 pixels. However, they now have resolutions as good as, and sometimes better than, the screens provided for desktop computers. Therefore, pages designed for desktop computers work well on the latest mobiles and tablets (provided that the pages validate and do not contain frames or table layouts).

Some websites provide an alternative style for the smaller and earlier handheld devices. These introduced a second link to the special style sheet something like this:

```
<link href="media-style.css" rel="stylesheet" type="text/css;" media="handheld">
<link href="stylesheet.css" rel="stylesheet" type="text/css;" media="screen">
```

This is not a satisfactory solution; therefore a process has been developed called Responsive Web Design (RWD), which is described in Chapters 35 through 38. But first, in preparation for those chapters, you will need to learn more of the basics of web design that you will find in the rest of this book.

The next section discusses the three types of layout: fixed, liquid, and semi-liquid. It describes their problems, limitations, and possible solutions.

Examining the Three Possible Layouts

The Advantages and Problems of Fixed-Width Layouts

Fixed-width websites are the easiest to design and control, and because the majority of websites are fixed width, users currently accept their limitations on bigger screens. At some future date, the fairly common 1024-pixel fixed width will increase to match the ever-increasing screen sizes and resolutions.

In previous chapters we used a 1024 fixed width; designers usually deduct 44 pixels to compensate for vertical scroll bars, and so the width of the wrapper is commonly set to 980 pixels. One advantage of the fixed-width layout is that when the user drags the right-hand edge of the browser window leftwards, the right-hand edge slides over the content, like drawing a curtain across it. This is good because the layout is not disturbed by shrinking the width; however, fixed-width sites are unsuitable for mobile phones.

The Advantages and Problems with Liquid Layouts

Liquid layouts were developed to cope with any variations in screen width.

Liquid layouts expand to fill the horizontal width of the screen. It was a reasonable solution for a while, but as screens became wider and screen resolutions increased, it became clear that the liquid layouts had their own problems. On smaller screens, the liquid website can go to pieces due to float drop. This means that an element on a page, such as an image, drops down below the other elements.

Text can stretch across a wide screen, making reading difficult; the reader's head waves to and fro like a windscreen wiper, therefore you should always present text in columns. On wide high-resolution screens, liquid layouts can result in large, unsightly gaps between side-by-side pictures. This can be partly overcome by placing pictures one above the other on a page with the text to the side of the pictures; however, that makes the pages longer; this is not good practice because it forces the user to scroll down unnecessarily.

In liquid layouts, a percentage is used for the width instead of pixels, like the next line of code.

```
#wrapper {width:90%; padding:0; text-align:center; margin:auto; background-color:white;}
```

Semi-Liquid Layouts Provide the Best Compromise

A semi-liquid layout uses two special attributes, namely, `min-width` and `max-width`.

We will now amend the style sheet so that our four HTML pages have a semi-liquid layout. Please follow these steps:

1. Open *style-3d.css* in code/source view in your HTML text editor.
2. Change the second line of code so that the code `width:1024px` is replaced by the two new widths as shown in bold in the Listing 10-1.

Listing 10-1. Creating a semi-liquid website

```
#wrapper {margin:auto; max-width:1200px; min-width:980px; background:#A6CCFF; ←
        border:10px white solid; font-size:medium;}
```

That's it! Job done. You now have a semi-liquid website. Test it by loading one of this chapter's HTML pages into a browser and then select the view that will allow you to shrink and expand the browser width.

If the title bar of the browser looks like Figure 10-1a, then click the button shown ringed. If it looks like Figure 10-1b you are in the correct view for this test.

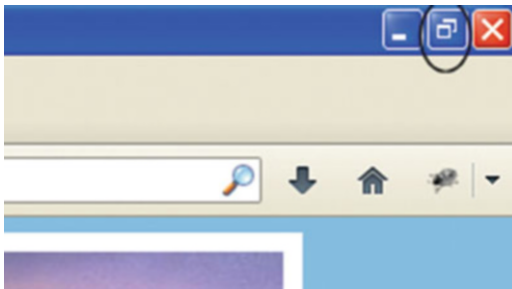


Figure 10-1a. Full screen is displayed

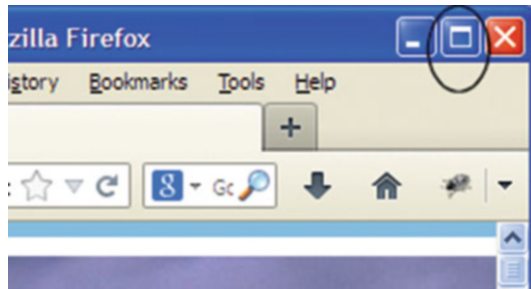


Figure 10-1b. Browser in resizable view

When you are in the view shown in Figure 10-1b, grab the right border of the browser with your mouse's left button and drag the browser to the left and right to see the effect of a semi-liquid layout.

As you move the border leftwards you will arrive at the minimum width, the right-hand edge of the browser will behave like a curtain being drawn over the page.

By using a combination of two styles: (i) flooding the body with color (introduced in Chapter 9), and (ii) a semi-liquid layout, we have a compromise that will look good on most desktop computer screen sizes and resolutions. The two styles are repeated below as a reminder.

```
body {background:#88CCFF;}
#wrapper {margin:auto; max-width:1200px; min-width:980px; background:#A6CCFF;
border:10px white solid; font-size:medium;}
```

■ **Note** Although a semi-liquid layout gives good results on desktop and laptop displays, semi-liquid, and fixed layouts are not suitable for handheld devices. These need a liquid layout with additional features that will cope with all viewport sizes. This is dealt with in the Advanced section of this book in Chapters 35 through 38.

Create a Template with a 3D Menu for Use in your Future Websites

We have just created a four-page website with a semi-liquid layout; this will be our first template. Because we will be creating more templates later, you will need this template (and several other templates) in future chapters. Therefore it would be sensible to store all the templates in a new folder called *templates*.

Please create the *templates* folder now by following these steps:

1. Within your folder *web-tutorial*, create a new folder named *templates*.
Within that folder, create a new folder named *template-3d-vmenu*.
2. Open the *chapter-10* folder and make a copy of all the files (Ctrl+A then Ctrl+C).
3. Open the *templates* folder and paste the files (Ctrl+V) into the new folder *template-3d-vmenu*.

You now have a template with ready-made files that you can use to create any number of websites that require a vertical menu with 3D buttons, a semi-liquid layout, and a white border. Unique websites can be created by simply changing the colors and the header picture in the CSS file, then altering the site's text and adding or removing pictures.

Creating a Template with a Border with Rounded Corners

Please follow these steps:

1. Within the *templates* folder, create a new folder named *template-3d-vmenu-round*.
2. Copy all the files in the folder *template-3d-vmenu* (Ctrl+A then Ctrl+C).
3. Paste the files (Ctrl+V) into the folder named *template-3d-vmenu-round*.
4. In the folder *template-3d-vmenu-round*, open *style-3d.css* in your HTML text editor.
5. In *style-3d.css* add the following three lines to the end of the existing CSS code:

```
#wrapper {border-radius:15px;}
#content {border-radius:6px;}
header {border-top-left-radius:6px; border-top-right-radius:6px; }
```
6. Save the file *style-3d.css*.
7. Open the file *index.html* and view it in a browser. The resulting display is shown in Figure 10-2.



Figure 10-2. Modern browsers display the border with rounded corners (but not Internet Explorer 8)

Explanation of the Code

The latest CSS code standard is known as CSS3, and the rounded corner is one of the new features in CSS3. The CSS3 styling words are logical and therefore easy to understand. IE 8 does not understand CSS3 so it just ignores it; as a result the wrapper's white border will have sharp corners.

```
#wrapper {border-radius:15px;}
```

This style's target is the `#wrapper`, which has a white border. It applies a 15 pixel radius to the four outer corners of the white border. The four inner corners are not affected, but the next two styles will set the inner corner radii.

```
#content {border-radius:6px;}
```

This style targets the corners of the `#content` area and so provides the inner radii of the white border. The inner radii must be smaller than the outer radii, so I have set it to 6 pixels instead of 15 pixels.

```
header {border-top-left-radius:6px; border-top-right-radius:6px; }
```

This style rounds the top corners of the header picture so that it fits neatly into the container's corners. You now have a template that provides a border with rounded corners.

Creating Two Templates with Plain Menu Buttons

We will now create two more templates giving the same features as the previous templates but using plain colored menu buttons, that is, with no 3D effect. Please follow these steps:

1. In the *templates* folder create two more folders, name one *template-plain-vmenu* and the other *template-plain-vmenu-round*.
2. In the *templates* folder, open the folder *template-3d-vmenu* and copy all the files (Ctrl+A then Ctrl+C) and paste them (Ctrl+V) into the folder named *template-plain-vmenu*.
3. In the folder *template-plain-vmenu*, use your HTML text editor to change the style link in the *index.html* file and in the other three pages as shown in bold type:

```
<link href="style-plain.css" rel="stylesheet" type="text/css">
```

4. In the folder *template-plain-vmenu*, RIGHT-click the file *style-3d.css*, and in the drop-down menu click *Rename*. Now rename the file *style-plain.css*.
5. Open *style-plain.css* in your HTML text editor and delete the items shown crossed through and amend the lines as shown in bold type.

```
/* set general access (anchor) styles */
li.btn a {display:block; width:115px; color:white; background:gray;
font-family:arial; font-size:small; font-weight:bold; text-
decoration:none;}
/* mouseout (default) */
li.btn a {background: #1A9CE0; border:4px-outset #AABAFF;}
/* mouseover */
li.btn a:hover {display:block; background:blue; border:4px-outset-
#8ABAFF;}
/* mousedown */
li.btn a:active {background:green; border:4px-inset #AECBFF; }
```

Make sure you save the file in the *templates* folder with the name *template-plain-vmenu*. To create the fourth template, please follow these steps:

1. In your *templates* folder create a new folder named *template-plain-vmenu-round*.
2. In the *templates* folder, open *template-plain-vmenu* and copy all the files (Ctrl+A) then (Ctrl+C).

3. Open the folder *template-plain-vmenu-round* and paste the files (use Ctrl+V).
4. In the same folder *template-plain-vmenu-round*, RIGHT-click the file *style-plain.css* and open it with your HTML text editor.
5. Add the following three lines to the end of the code to give rounded corners to the border of the wrapper:


```
#wrapper {border-radius:15px;}
#content {border-radius:6px;}
header {border-top-left-radius:6px; border-top-right-radius:6px; }
```
6. Save the file.

You now have four template folders containing ready-coded files for four websites, as shown in Figures 10-3, 10-4, 10-5, and 10-6. To save space, only the top-left corner of each web page is shown

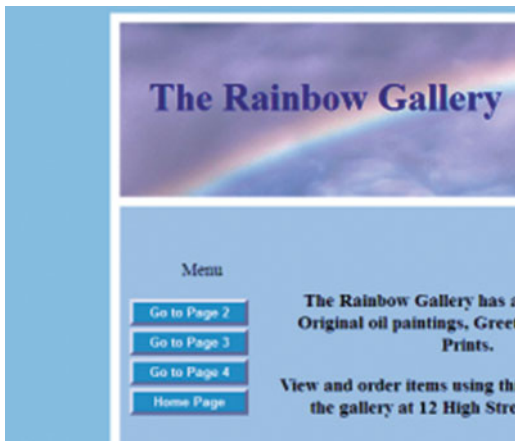


Figure 10-3. *template-3d-vmenu*



Figure 10-4. *template-3d-vmenu-round*



Figure 10-5. *template-plain-vmenu*



Figure 10-6. *template-plain-vmenu-round*

■ **Note** The templates folder and the four templates are available in the downloadable zip file for Chapter 10

Summary

This chapter discussed the problem caused by the wide variations in size and resolution of monitors. Fortunately, the physical size of monitors imposes its own limit; few users have room on their desks for anything wider than a 25-inch screen. The chapter described three solutions for coping with the wide variation in screen sizes: (i) fixed-width pages, (ii) liquid pages and, (iii) semi-liquid pages. You learned that a semi-liquid website using max-width and min-width attributes offers the best solution. We briefly covered the problems posed by handheld devices; solutions to these problems are provided in Chapters [35](#) through [38](#).

You learned how to create useful templates and then created four templates with vertical menus that will be used in future chapters. Feel free to use these templates as a basis for your own websites.

In the next chapter we will adapt one of the templates to produce a more fully worked website that includes a picture gallery.

CHAPTER 11



A Vertical Menu with a Picture Gallery

This chapter shows you how to produce a practical website by adapting one of the templates that we created in Chapter 10. Chapter 11 contains the following sections:

- Adapting the home page
- Change *page-2.html* to create the About Us page
- Create the Location page using the file *page-4.html*
- Create the Gallery page and displaying six pictures
- Summary

For this lesson you need to create a new folder to contain the code for Chapter 11. This step is important to prevent confusing the previous chapter's code with the new code.

1. In the *web-tutorial* folder, create a new folder named *Chapter-11*.
2. In the *web-tutorial* folder, go back to the *templates* folder and open it.
3. Copy all the files in the folder *template-3d-vmenu* (use Ctrl+A, then Ctrl+C) and paste them into the new folder *Chapter-11* (use Ctrl+V).
4. Rename the *images* folder as *old-images*.
5. Download the *Chapter 11 zip* file from the book's page at *Apress.com*.
6. Unzip the *Chapter 11 zip* file into your *Chapter 11* folder. Note that the new *images* folder contains seven new pictures.

For simplicity we will stay with the Rainbow Gallery theme. We will modify the home page and add content to the empty pages in the template. We will also add a gallery of paintings and tweak the -color within the wrapper.

■ **Note** My thanks go to Ann Roe Jones for allowing me to use pictures of her paintings for this chapter. Ann paints under the name of Ann L. Roe and she is a well-known Devon portrait painter. Her website can be found at <http://www.annroejones-artist.co.uk>

Adapting the Home Page

The new website will be for an imaginary gallery owner and portrait painter named Suzanne Rembrandt. The adaptation will be as follows:

- Change the menu labels to match three revised page names.
- Replace the still-life painting in the home page with a portrait (in fact, it is a double portrait).
- Change the text on the home page to match Suzanne Rembrandt's website.
- The background color in the template's wrapper is a little too strong; the portrait needs to stand out more against a paler background. We will therefore lighten the background color of the wrapper.

To achieve these changes please follow these steps:

1. Open the home page *index.html* in your HTML text editor and change the menu as shown in bold type in the next snippet of code:

```
<nav>
  <ul>
    <p>Menu</p>
    <li class="btn"><a href="about.html">About Us</a></li>
    <li class="btn"><a href="gallery.html">Gallery</a></li>
    <li class="btn"><a href="location.html">Location</a></li>
    <li class="btn"><a href="index.html">Home Page</a></li>
  </ul>
</nav>
```

2. Replace the still-life painting with a portrait as shown in bold type in the following snippet of code:

```
<div id="midcol-right">
  
</div>
```

3. Delete the line shown crossed through in the following snippet:

```
<div id="midcol">
  <h2>Home Page</h2>
  <p>This is the content</p>
```

4. Now we need to make a paler background color for the wrapper. To achieve the required color, I used a little trial and error. Open the CSS style sheet *style-3d.css* in your HTML text editor and change the background color of the wrapper from #A6CCFF; to #AEDEFF; this will create a paler version of the background color. The red component code A6 is intensified to AE, the green component is intensified from CC to DE. The blue component cannot be intensified further because it already has the maximum intensity FF.

If you use a WYSIWYG editor, try using the WYSIWYG view to amend the text in the *midleft-col* area. If you have a non-WYSIWYG text editor, type the text into the Code/Source view. The revised text is shown circled in Figure 11-1.

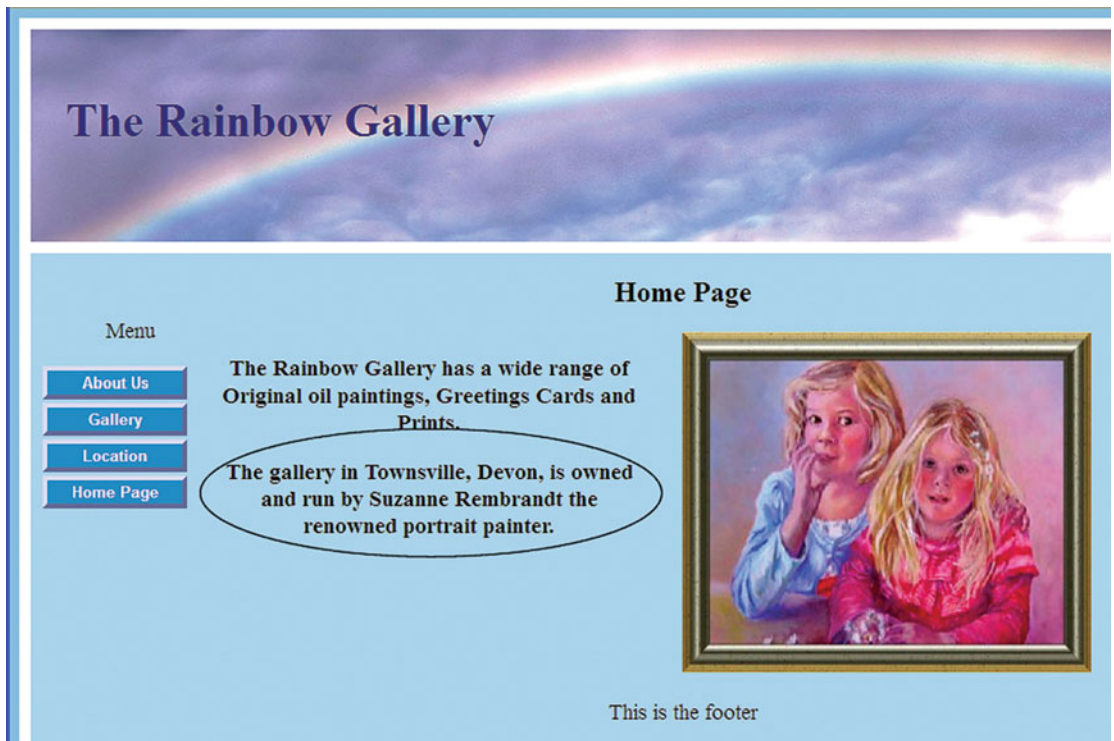


Figure 11-1. Only the left-hand portion of the home page is depicted

The menu buttons may not look exactly like those in the illustration because WYSIWYG editors sometimes display incorrectly; but don't worry, the buttons will look fine in a modern browser.

■ **Tip** For best results, start by placing the cursor *within* the existing text, type the new text, and then delete any unwanted text. This works in both views, WYSIWYG or Code view. You can always click the *Undo* icon on the toolbar if you spoil the layout.

5. Save the file *index.html*.

Change *page-2.html* to Create the About Us Page

Open the file *page-2.html* in your HTML text editor and save the file as *about.html*

If you are using a plain text editor, amend the content as shown in Figure 11-2.

If you are using a WYSIWYG editor, in the Design/WYSIWYG view, amend the text in both columns as if you are using a word processor; the amended text is shown in Figure 11-2.

Only part of the screen is shown to save space and to show the text clearly.

■ **Tip** For best results in WYSIWYG editors, start by placing the cursor *within* the existing text, type the new text, and then delete any unwanted text. You can always click the *Undo* icon on the toolbar if you spoil the layout.



Figure 11-2. Partial view showing the amended text in the *About Us* page

Now switch to Code/Source view and amend the code as shown in bold type in Listing 11-1:

Listing 11-1. The code for the file *about.html*

```
<!DOCTYPE html>
<html>
  <head>
    <title>About Us</title>
    <meta charset=utf-8>
    <link href="style-3d.css" rel="stylesheet" type="text/css">
      <!--[if lte IE 8]>
        <script src="html5.js">
        </script>
      <![endif]-->
  </head>
  <body>
    <div id="wrapper">
      <header>
        <h1>The Rainbow Gallery</h1>
      </header>
      <div id="content">
        <nav>
          <ul>
            <p>Menu</p>
            <li class="btn"><a href="about.html">About Us</a></li>
            <li class="btn"><a href="gallery.html">Gallery</a></li>
            <li class="btn"><a href="location.html">Location</a></li>
            <li class="btn"><a href="index.html">Home Page</a></li>
          </ul>
        </nav>
        <div id="rightcol">
          <p>This is the far right column</p>
        </div>
```

```

<div id="midcol">
    <h2>About Us</h2>
<div id="midcol-left">
<p>The Rainbow Gallery is owned and managed by the portrait painter Suzanne
Rembrandt. She specializes in paintings by local artists as well as her own work. </p>
<p>Suzanne is renowned for her portrait paintings and has received numerous
commissions to paint many well known people</p>
</div>
<div id="midcol-right">
<p>Suzanne also stages exhibitions of her own and local artists' work from time to
time. This can include pottery and sculpture, but paintings always predominate</p>
<p>The Gallery was established in 1936 and was purchased by Suzanne in 1970. Suzanne is
also a skilled picture framer.</p>
</div>
</div>
<!--content div finishes here-->
<br class="clear">
<footer>
    <p>This is the footer</p>
</footer>
</div><!--end of wrapper div-->
</body>
</html>

```

Test the page by pressing the F12 key in Expression Web. Or in Blue Griffon, click the blue globe and select your browser. If you are using a plain text editor right-click the *index.html* file and select *Open With* to view the file in a modern browser. You will see that the paragraphs are centered, but we will now correct that.

Left-Align the Text in the About Us Page

To left-align the text in the About Us page, please follow these steps:

1. Open the file *about.html* in the Code/Source view of your HTML text editor and insert the following internal style shown in bold type. (Internal styles will be explained later in the chapter.)

```

<!DOCTYPE html>
<html>
    <head>
        <title>About Us</title>
        <meta charset=utf-8>
        <link href="style-3d.css" rel="stylesheet" type="text/css">
        <style type="text/css">
            #midcol-left p, #midcol-right p {text-align:left;}
        </style>
        <!--[if lte IE 8]>
            <script src="html5.js">
            </script>
        <![endif]-->
    </head>

```

The realignment instruction `text-align:left`; is applied to any paragraph that lies within the `#midcol-left` or the `#midcol-right` columns.

2. Save the revised file and view the *about.html* page in your browser, you should see that the paragraphs have been realigned left.

Create the Location Page Using the File *page-4.html*

We will now use a copy of *page-4.html* to create the location page. Please follow these steps:

1. Open the file *page-4.html* in your HTML5 editor and save a copy of the file as *location.html*.
2. In Code/Source view, amend the code in the file *location.html* as shown in bold type in Listing 11-2; also delete the line shown crossed through.

Listing 11-2. Change the code in the file *location.html*

```
<!DOCTYPE html>
<html>
  <head>
    <title>Location</title>
    <meta charset=utf-8>
    <link href="style-3d.css" rel="stylesheet" type="text/css">
      <!--[if lte IE 8]>
        <script src="html5.js">
          </script>
        <![endif]-->
      </head>
  <body>
    <div id="wrapper">
      <header>
        <h1>The Rainbow Gallery</h1>
      </header>
      <div id="content">
        <nav>
          <ul>
            <p>Menu</p>
            <li class="btn"><a href="about.html">About Us</a></li>
            <li class="btn"><a href="gallery.html">Gallery</a></li>
            <li class="btn"><a href="location.html">Location</a></li>
            <li class="btn"><a href="index.html">Home Page</a></li>
          </ul>
        </nav>
        <div id="rightcol">
          <p>This is the far right column</p>
        </div>
        <div id="midcol">
          <h2>Location</h2>
          <p>This is the content</p>
```

Leave the rest of the code unchanged, then save the file and keep it open in your text editor.

You can change the content of the file *location.html* either in the Design/WYSIWYG view, or in code view. The Design/WYSIWYG view method will be described first.

Amend the text in both columns as shown in Figure 11-3. Only part of the Design/WYSIWYG screen is shown to save space, and also to show the text clearly.

■ **Tip** For best results in WYSIWYG view, start by placing the cursor *within* the existing text, type the new text, and then delete any unwanted text. You can always click the *Undo* icon on the toolbar if you spoil the layout.

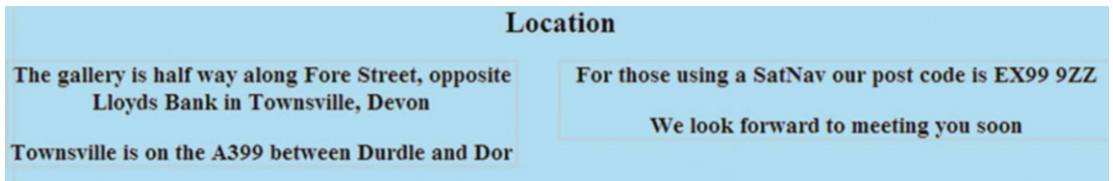


Figure 11-3. The new text in the *location.html* file

To change the text in the location page using a plain text editor, you will need to change the text shown bold in the following snippet:

```
<div id="midcol-left">
    <p>The gallery is half way along Fore Street, opposite Lloyds Bank in ↵
      Townsville, Devon</p>
    <p>Townsville is on the A399 between Durdle and Dor</p>
</div>
<div id="midcol-right">
    <p>For those using a Satnav our post code is EX99 9ZZ</p>
    <p>We look forward to meeting you soon</p>
</div>
```

3. Save the file as *location.html*.
4. View the page in a modern browser to check that all is well.

Creating the Gallery Page

The gallery page is shown in Figure 11-4.



Figure 11-4. The Gallery page

The gallery page is achieved by inserting the names and locations of the images into the HTML code and then applying a special CSS style to the images. We will use an *internal* style because the styling requirement is unique to that page. To prevent float-drop (where one or more pictures drop down below the other pictures) we must juggle with the margins and widths. This is best done by means of an internal style that can be tweaked without affecting the other pages.

■ **Important** In a gallery, the pictures in the top row of the display must have the same height. Adjust the pictures in your graphics program; the heights can vary by one or two pixels at the most. The heights of the pictures in the bottom row are less important. If you experience float-drop (i.e., pictures drop out of position), style the `<figure>` tag with a height large enough to cure the float-drop. Chapter 41 contains advice for adjusting pictures using free graphics programs.

To create the gallery page, please follow these steps:

1. Open *page-3.html* and use *Save As* to save it with the new name *gallery.html*.
2. With *gallery.html* in Code/Source view delete the lines shown crossed through, and amend the code as shown in bold type in Listing 11-3.

Listing 11-3. Create the code for the gallery page

```
<!DOCTYPE html>
<html>
  <head>
    <title>Gallery</title>
    <meta charset=utf-8>
    <link href="style-3d.css" rel="stylesheet" type="text/css">
<style type="text/css">
#rightcol {float:right;width:10px;}
#midcol {margin-left:170px;margin-right:10px;}
#wrapper {max-width:1200px;min-width:1080px; background:#AEDEFF; ~
border:10px white solid; font-size:medium;}
</style>
    <!--[if lte IE 8]>
      <script src="html5.js">
      </script>
    <![endif]-->
  </head>
<body>
  <div id="wrapper">
<header>
  <h1>The Rainbow Gallery</h1>
</header>
<div id="content">
  <nav>
    <ul>
      <p>Menu</p>
      <li class="btn"><a href="about.html">About Us</a></li>
      <li class="btn"><a href="gallery.html">Gallery</a></li>
      <li class="btn"><a href="location.html">Location</a></li>
      <li class="btn"><a href="index.html">Home Page</a></li>
    </ul>
  </nav>
<div id="rightcol">
  <p>This is the far right column</p>
</div>
<div id="midcol">
  <h2>Gallery</h2>
  <p>This is the content</p>
<div id="midcol-left">
<p>This is the midcol-left</p>
<p>Some text in the left column</p>
</div>
<div id="midcol-right">
<p>This is the midcol-right</p>
```

```

<p>Some text in the right column</p>
</div>
<figure>
  
  <p>The Late Ali Bongo&mdash;Magician</p>
</figure>

<figure>
  
  <p>Dr Alan Cotton</p>
</figure>

<figure>
  
  <p>Reuben</p>
</figure>

<br class="clear">

<figure>
  
  <p>The late Sir Patrick Moore</p>
</figure>

<figure>
  
  <p>Megan</p>
</figure>

<figure>
  
  <p>Chelsea Pensioners</p>
</figure>

```

The rest of the code remains unchanged.

3. Save the file.

Explanation of the Code

```

<link href="style-3d.css" rel="stylesheet" type="text/css">
<style type="text/css">
#rightcol {float:right; width:10px;}

```

```
#midcol {margin-left:170px; margin-right:10px;}
#wrapper {max-width:1200px; min-width:1080px; ←
background:#AEDEFF; border:10px white solid; font-size:medium;}
</style>
```

Until now you have always used an external CSS style sheet. We are now using an additional internal style sheet (styles that are typed into the <head> section of the HTML page). This is given in bold type in the above code snippet. The style is surrounded by the tags <style type="text/css"> and </style>. Between those tags the styling commands take exactly the same format as for an external style sheet.

We use internal style sheets when the styling for a particular page is unique to that page. The Gallery page is the only one in which the *rightcol* is shrunk to a width of 10 pixels; this gives us more room to fit several pictures into the *midcol* area. The *midcol*-right margin is shrunk to match the reduced *rightcol* width. The *wrapper* is given a larger minimum width; this ensures that when the browser is shrunk horizontally by the user, the pictures fit into the area without float-drop.

```
<div id="content">
  <nav>
    <ul>
      <p>Menu</p>
      <li class="btn"><a href="about.html">About Us</a></li>
      <li class="btn"><a href="gallery.html">Gallery</a></li>
      <li class="btn"><a href="location.html">Location</a></li>
      <li class="btn"><a href="index.html">Home Page</a></li>
    </ul>
  </nav>
  <div id="rightcol">
    <p>This is the far right column</p>
  </div>
  <div id="midcol">
    <h2>Gallery</h2>
    <div id="midcol">
      <div id="midcol-left">
        <p>This is the midcol-left</p>
        <p>Some text in the left column</p>
      </div>
      <div id="midcol-right">
        <p>This is the midcol-right</p>
        <p>Some text in the right column</p>
      </div>
    </div>
  </div>
```

The amended code changes the menu to match the revised pages and it also changes the name between the <h2> tags to *Gallery*. We also deleted the lines shown crossed through to make room for the pictures.

```
< figure>
  
  <p>The Late Ali Bongo&mdash;Magician</p>
</figure>
```

Six pictures have been inserted to replace the columns *midcol-left* and *midcol-right*. To save space, I will only explain the first of the six images.

The tag `<figure></figure>` is an HTML5 semantic tag that replaces the former `<div id=...>` tag that would have been `<div id="figure"></div>`. The tag `<figure>` encloses the image and a caption, and it is in fact a container. The new tag will work with Internet Explorer 8 provided the JavaScript workaround is used and that the word **figure** is included in the style sheet.

The abbreviation **img** means image and the code `src="images/alibongo.jpg"` points to the source (**src**) of the image, that is, it is in the folder named *images*. The width and height of the picture is given and also a caption that is enclosed in paragraph tags.

The code `alt="The late Ali Bongo—Magician"` provides the user with a pop-up description of the picture when the cursor is hovered over the picture; this pop-up is known as a *Tool tip*.

The code `title="The late Ali Bongo—Magician"` is an alternative to the `alt` tag and is for the benefit of those browsers that do not respond to the `alt` tag. The code `—` is an *entity* that creates a long hyphen on the user's screen (a dash that is as long as the letter "m"). A list of entities is provided in Chapter 41. Here are two more examples: `&` displays an ampersand, `£` displays a British pound symbol.

Adding the *figure* Style to the External Style Sheet

Using your HTML text editor in Code/Source view, amend the external style sheet *style-3d.css* as shown in bold type in the following code snippet:

```
body {background:#88CCFF;}
/*#wrapper {margin:auto; max-width:1200px; min-width:980px; background:#A6CCFF; ↵
border:10px white solid; font-size:medium;}*/
#wrapper {margin:auto; max-width:1200px; min-width:980px; background:#AEDEFF; ↵
border:10px white solid; font-size:medium;}
header,nav,article,section,footer,figure {display:block;}
figure {float:left; margin-left:15px; }
header {margin-top:0; border-bottom:10px white solid; height:181px; ↵
background-image:url('images/rainbow-banner.jpg');}
```

To allow Internet Explorer 8 to understand the semantic tag **figure**, the tag must be entered into the list of tags that can be manipulated by the JavaScript (fourth line from the bottom of the above code).

In the style sheet the figure tag is told to float left and provide a left margin of 15 pixels.

Save the file *style-3d.css*.

View the gallery page in a browser and then hover the cursor over the pictures to see the pop-up *alt* and *title* items.

The gallery page did not have enough room to accommodate more than three pictures in a row. If you require more pictures per row there are several solutions: (i) reduce the size of the pictures, (ii) increase the wrapper's max and min widths, or (iii) move the menu out of the way by using a horizontal menu; this is the approach that we will be exploring in the next chapter.

Summary

In this chapter you adapted one of the templates to create a small but practical website for the Rainbow Gallery. If you were using a WYSIWYG editor you practiced inserting text using the Design/WYSIWYG view. You learned how to left-align some paragraphs and how to create a picture gallery. The semantic tag `<figure>` was explained and it was added to the list of semantic tags that could be read by Internet Explorer 8 using the JavaScript workaround.

In the next chapter we will explore horizontal menus so that we can increase the number of pictures in the gallery page.

CHAPTER 12



A Horizontal Menu with an Enlarged Picture Gallery

This chapter uses one of four new template files that you can download in the zip file named *templates-h.zip*, and this will save you a great deal of coding time. I will describe how a vertical menu block is converted into a horizontal row of buttons. The template files are adaptations of the templates that we created in Chapter 10, the only difference being that the new templates have horizontal menus. By using a horizontal menu we will have enough room to add one extra picture per row.

This chapter contains the following sections:

- Download and install four templates with horizontal menus
- Converting a vertical menu into a horizontal menu
- Understanding the horizontal menu templates
- The modified CSS style for a horizontal menu with 3D buttons
- The modified CSS style for a horizontal menu with plain buttons
- Adding more buttons to the horizontal menu
- Tutorial: Taking advantage of a horizontal menu
- Tutorial:
- Summary

Download and Install Four Templates with Horizontal Menus

Note: For the first part of this chapter, you do NOT yet need to create a new folder for Chapter 12. Instead, please follow these steps:

1. Download the zip file *templates-h.zip* from this book's page at *Apress.com*.
2. Right-click the zip file and save it in your *templates* folder. You may find it helpful to create a subfolder named *templates-h*.
3. Unzip the file in either your *templates* folder or your *templates-h* folder; you will then have four new *template* folders for horizontal menus. The four horizontal menu templates are based on the previous Chapter 10 Rainbow Gallery pages with four columns as shown in Figure 12-1. However, in the gallery page we will dispense with the columns to provide plenty of room for an increased number of images.

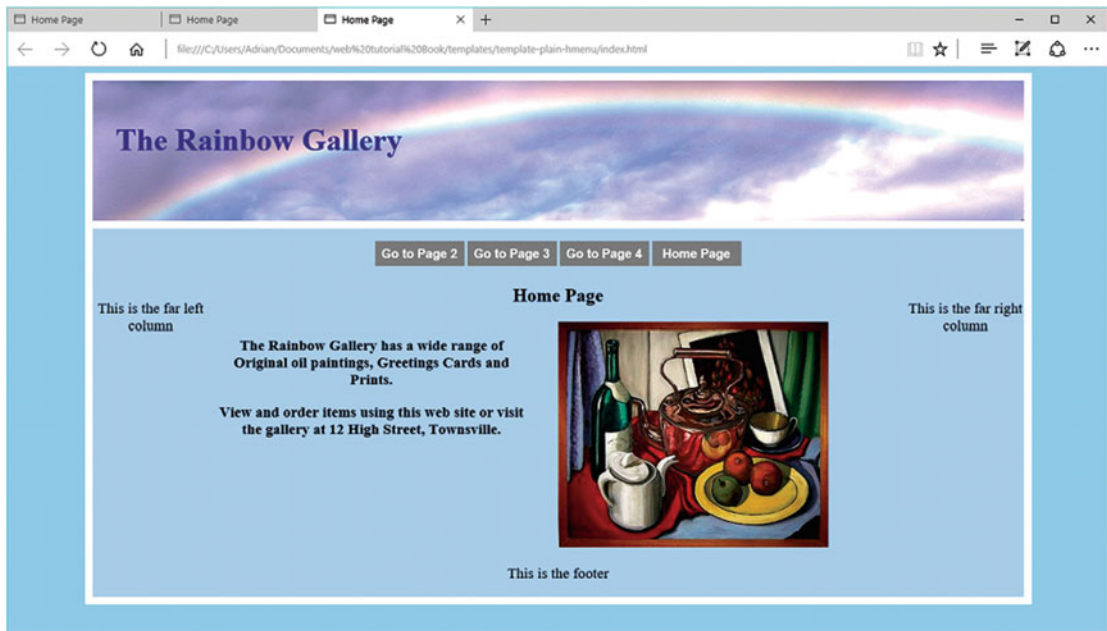


Figure 12-1. Showing the website template based on the Chapter 10 home page but incorporating a horizontal menu with plain buttons

In the downloadable folder *templates-h*, the four new template folders with horizontal menus are the following:

- template-3d-hmenu (a horizontal 3D menu, and a wrapper with square corners)
- template-3d-hmenu-round (a horizontal 3D menu, and a wrapper with rounded corners)
- template-plain-hmenu (a horizontal plain button menu and a wrapper with square corners)
- template-plain-hmenu-round (a horizontal plain button menu, the wrapper has rounded corners)

■ **Tip** The first sections of this chapter consist of explanations relating to a horizontal menu. **No practical work is required** in these sections and you can skip the explanations if you wish. However, I strongly recommend that you try to understand the logic in the steps taken to achieve a horizontal menu.

Converting a Vertical Menu into a Horizontal Menu

You will recall that the HTML page provides the structure of the page; in other words the items appear on the screen in the same order as the items in the HTML code. With vertical menu blocks, the menu appears after the line `<div id="content">`, that is, the menu block is within the content section.

For horizontal menus, the code for the menu is moved to a location immediately above the content section, that is, above the line `<div id="content">`. This causes the horizontal menu to be displayed just below the header and just before the main content (see Figure 12-1).

Listing 12-1 gives the amended code for every page in the four new horizontal menu templates.

Note that the menu now appears immediately below the `<header></header>` block of code.

Partial Listing 12-1. Creating a horizontal menu block for the four new templates

```
<header>
  <h1>The Rainbow Gallery</h1>
</header>
  <nav>
    <ul>
      <p>Menu</p>
      <li class="btn"><a href="page-2.html">Go to Page 2</a></li>
      <li class="btn"><a href="page-3.html">Go to Page 3</a></li>
      <li class="btn"><a href="page-4.html">Go to Page 4</a></li>
      <li class="btn"><a href="index.html">Home Page</a></li>
    </ul>
  </nav>
  <br class="clear">
<div id="content">
  <div id="leftcol">
    <p>This is the far left column</p>
  </div>
  <div id="rightcol">
    <p>This is the far right column</p>
  </div>
```

Understanding the Horizontal Menu Templates

In the new horizontal menu templates, I moved the menu upwards to a position immediately below the header and above the main content. I removed the line `<p>Menu</p>` because; by now it will be obvious to you that a block of rollover buttons is a menu.

The line `<br class="clear">` pushes the main content down below the horizontal menu.

```
<div id="leftcol">
  <p>This is the far left column</p>
</div>
```

When the menu was a vertical block, it automatically created a column for itself at the far left and inside the wrapper. I removed the vertical menu and created a far left column instead.

This retained the four-column layout for the new templates.

The horizontal menu has the same HTML code as the previous vertical menu. The vertical menu is changed to a horizontal menu by means of a minor modification to the CSS style sheet.

The Modified CSS Style for a Horizontal Menu with 3D Buttons

Most of the style sheet code remains unaltered, but a few tweaks are necessary to produce a horizontal menu and a far left column. The changes to the style sheet *style-3d.css* are shown in bold type in Listing 12-2.

Listing 12-2. The CSS style sheet for a horizontal menu with 3D buttons style-3d.css

```
body {background:#88CCFF;}
#wrapper {margin:auto; max-width:1200px; min-width:980px; background:#A6CCFF; ◀
border:10px white solid; font-size:medium;}
header,nav,article,section,figure,footer { display:block;}
header {margin-top:0; border-bottom:10px white solid; height:181px; ◀
background-image:url('images/rainbow-banner.jpg');}
h1 {position:absolute; top:45px; margin-left:30px; font-size:250%; color:navy; ◀
font-weight:bold;}
h2 {font-size:150%; text-align:center;}
p {font-size:120%; text-align:center;}
a {font-size:120%;}
/*add a left col*/
#leftcol {float:left; width:150px;}
#rightcol {float:right; width:150px; height:190px;}
#midcol {margin-left:155px; margin-right:155px;}
br.clear {clear:both}
#midcol-left {float:left; width:46%; font-weight:bold;}
#midcol-right {float:right; width:50%; font-weight:bold;}
/*set horizontal button menu*/
nav ul {float:left; width:1000px; height:40px; margin-left:100px; ◀
text-align:center; padding-left:0;list-style-type:none;}
/*set general horizontal menu button styles*/
nav ul li {display:inline-block;}
li.btn {width:115px; line-height:2; margin-bottom:3px; margin-right:5px; ◀
text-align :center;}
/* set general access (anchor) styles */
li.btn a {display:inline-block; width:115px; color:white; font-family:arial; ◀
font-size:100%; font-weight:bold; text-decoration:none;}

/* mouseout (default) */
li.btn a {background: #1A9CE0; border:4px outset #AABAFF;}
/* mouseover */
li.btn a:hover {background:#0A4ADF; border:4px outset #8ABAFF;}
/* mousedown */
li.btn a:active {background:#AECBFF; border:4px inset #AECBFF; }
```

Explanation of the Code

```
/*add a left col*/
#leftcol {float:left; width:150px;}
```

Note that a comment in a CSS style sheet is surrounded by the tags `/*` and `*/`. A far left column is specified in the HTML pages; therefore a style has been applied to the *leftcol*.

```
/*set horizontal button menu*/
nav ul {float:left; width:1000px; height:40px; margin-left:100px; ◀
text-align:center; padding-left:0; list-style-type:none;}
```

The first line is a comment. The second line creates a container 1,000 pixels wide by 40 pixels high for the horizontal menu. The container is given 100 pixels left margin, which centers the menu on the widest screen. The bullet points are removed by using `list-style-type:none`;

```
/*set general horizontal menu button styles*/
nav ul li {display:inline-block;}
li.btn {width:115px; line-height:2; margin-bottom:3px; margin-right:5px; ↵
text-align:center;}
```

The first line is a comment. The change in the second line ensures that the menu buttons will be displayed in a horizontal row. The width of each button is set to 115 pixels, and the line-height of the label text is increased to 2 to give some additional space above and below the labels on the buttons.

```
/* set general access (anchor) styles */
li.btn a {display:inline-block; width:115px; color:white; font-family:arial; ↵
font-size:100%; font-weight:bold; text-decoration:none;}
```

The first line is a comment. The change in the second line styles the width, height, and padding for the 3D buttons.

■ **Note** The same HTML changes are applied to every page in the four new templates. To produce the differences between the templates, each template contains a link to a slightly different style sheet, just like the links in the four vertical menu templates in Chapter 10.

The style sheet for *template-3d-hmenu-round* is the same as for *template-3d-hmenu* except for the three lines that make the rounded corners; these are added to the end of the code as shown in Listing 12-3.

Listing 12-3. Add rounded corners to the wrapper

```
#wrapper {border-radius:15px;}
#content {border-radius:6px;}
header {border-top-left-radius:6px; border-top-right-radius:6px;}
```

The Modified CSS Style for a Horizontal Menu with Plain Buttons

The style sheet for *template-plain-hmenu* is the same as Listing 12-2 except that the 3D buttons are changed to plain buttons as shown in bold type in Listing 12-4.

Listing 12-4. Style the horizontal menu and the buttons

```
/*set horizontal button menu*/
nav ul {float:left; width:1000px; height:40px; margin-left:100px; ↵
text-align:center; padding-left:0; list-style-type:none;}
/*set general horizontal menu button styles*/
nav ul li {display:inline-block;}
/* set general side button styles */
li.btn {width:115px; line-height:2; margin-bottom:3px; text-align:center;}
/* set general access (anchor) styles */
li.btn a {display:block; width:115px; color:white; background:gray; ↵
```

```
font-family:arial; font-size:100%; font-weight:bold; text-decoration:none;}
/* mouseover */
li.btn a:hover {background:blue;}
/* mousedown */
li.btn a:active {background:green;}
```

The style sheet for *template-plain-hmenu-round* is the same as for *template-plain-hmenu* except for the three lines added to the end of the code as shown in Listing 12-5:

Listing 12-5. Adding rounded corners to the wrapper

```
#wrapper {border-radius:15px;}
#content {border-radius:6px;}
header {border-top-left-radius:6px; border-top-right-radius:6px;}
```

Adding More Buttons to The Horizontal Menu

The templates are for a website with four pages; therefore the menu has four buttons. If you add more pages to the website, you will need more buttons to access those pages. When you add extra buttons, the menu will not be centered on the screen.

To solve this, reduce the left margin of the menu block until the menu is centered. Use trial and error to center the menu by reducing the left margin, and make it smaller than the 100 pixels that you can see in the following line:

```
nav ul {float:left; width:1000px; height:40px; margin-left:100px; text-align:center;
padding-left:0; list-style-type:none;}
```

The horizontal menu could accommodate 8 buttons for an 8-page website. You could double this to 16 by means of a second horizontal row of buttons like the website www.bbc.co.uk. A horizontal menu with a double row of buttons is described in Chapter 34.

Tutorial: Taking Advantage of a Horizontal Menu

In Chapter 11, I suggested that if the vertical menu was removed, there would be room to squeeze one more picture into each row of the gallery. For this tutorial we will use a template for a horizontal menu taken from the *templates* folder. The next exercise will demonstrate this.

For this tutorial you need to create a new folder to contain the code for *Chapter 12*. This step is important to prevent confusing the previous chapter's code with the new code. Please follow these steps:

1. In your *web-tutorial* folder, create a new folder named *Chapter-12*.
2. Download the zip file for Chapter 12 and unzip it in your Chapter 12 folder. The folder contains all the pictures that you will need for this next tutorial.
3. In your *web-tutorial* folder, go back to the *templates* folder (or *templates-h* folder) and open it.
4. Copy all the files in the downloaded folder *template-plain-hmenu* (use Ctrl+A then Ctrl+C) and paste them into the new folder *Chapter-12* (use Ctrl+V).

To save time and space we will only create the Gallery page and this is shown in Figure 12-2.



Figure 12-2. A horizontal menu will allow us to dispense with the far left column on the gallery page so that we have room to insert one more picture in each row

Because the special style for the gallery page is not used in other pages, I chose to use an internal style for the page as shown in Listing 12-6.

Tutorial

This section will show you how to produce a new page by combining ready-made snippets of code from other pages. We will use the gallery code from Chapter 11 and combine it with the style for a horizontal menu used in the new template, that is, *template-plain-hmenu.css*.

The tutorial will also demonstrate the following:

- How, by moving a menu upwards within the HTML code, we can provide the basis for a horizontal menu
- How the menu is styled by using the CSS file from a horizontal menu template
- How to add two extra pictures to the gallery

Typing all the picture details again would be very tedious; therefore we will make use of the gallery page from Chapter 11 because it already contains the code for six pictures. Please follow these steps:

■ **Warning** In step 2 below, don't be tempted to use *Save As* to save the Chapter 11 gallery file directly into the folder *chapter-12*. If you do, the result will be a file containing multiple links to Chapter 11.

1. Open the folder *Chapter-11*.
2. In *File Explorer*, click the file *gallery.html* then use Ctrl+C to place a copy of the file in the computer's memory.
3. Open the *Chapter-12* folder, use Ctrl+V to paste the file *gallery.html* into the *chapter-12* folder.
4. In the *Chapter-12* folder, open file *gallery.html* in the Code/Source pane of your HTML text editor and change the linked style to *style-plain.css*.
5. Delete the current internal style and replace it with the new internal style shown in bold type in Listing 12-6:

Listing 12-6. Insert the style link for the horizontal menu block and adjust the gallery page layout

```
<!DOCTYPE html>
<html>
<head>
  <title>Gallery</title>
  <meta charset=utf-8>
  <link href="style-plain.css" rel="stylesheet" type="text/css">
<style type="text/css">
#rightcol {float:right; width:10px;}
#midcol {margin-left:170px; margin-right:10px;}
#wrapper {max-width:1200px; min-width:1080px; background:#AEDEFF; border:10px white solid;
font-size:medium;}
figure {float:left; margin:0 20px 0 0;}
#midcol {width:1100px; margin:auto;}
p {margin-top:0;}
</style>
```

6. In Listing 12-7 we will move the block of menu code upwards to a new location just below the header code and just above the content. Add the code <br class="clear"> below the menu block. Be sure to change the two menu items shown in bold type because the template we are using does not contain the files *about.html* and *location.html*. The amendments are shown in bold in Listing 12-7.

Listing 12-7. Move the menu block of code up to a position just below the header code, and amend the menu as shown in bold type

```
<header>
  <h1>The Rainbow Gallery</h1>
</header>
<nav>
  <ul>
```

```

        <p>Menu</p>
        <li class="btn"><a href=page-2.html>Go to Page 2</a></li>
        <li class="btn"><a href=gallery.html>Gallery</a></li>
        <li class="btn"><a href=page-4.html>Go to Page 4</a></li>
        <li class="btn"><a href=index.html>Home Page</a></li>
    </ul>
</nav>
<br class="clear">

```

It would be a good idea to update the menu on the other three pages by amending the menu link to access the file *gallery.html* (second menu item in the above code). In Listing 12-7 we will move the block of menu code upwards to a new location.

7. We will dispense with the far right column to provide room for extra pictures; therefore delete the unwanted item shown crossed through in Listing 12-8.

Listing 12-8. Delete the unwanted right column

```

<div id="content">
    <div id="rightcol">
        <p>This is the far right column</p>
    </div>
<div id="midcol">

```

8. Add a new picture to the upper row of pictures just after the “Reuben” image as shown in bold type in Listing 12-9.

Listing 12-9. Insert another image into the top row of pictures after the line `<p>Reuben</p>`

```

<p>Reuben</p>
</figure>
<figure>
    
    <p>Morning news</p>
</figure>
<br class="clear">

```

9. Just after the “Chelsea Pensioners” image, add a new picture to the end of the bottom row of pictures as shown in bold type in Listing 12-10.

Listing 12-10. Add one extra picture to the end of the bottom row of images

```

<p>Chelsea Pensioners</p>
</figure>
<figure>
    
    <p>Mr Friedman</p>
</figure>

```

10. Save the file then view it in a modern browser.

Explanation of the code for Listings 12-6 to 12-10

```
<title>Gallery</title>
<meta charset=utf-8>
<link href="style-plain.css" rel="stylesheet" type="text/css">
<style type="text/css">
    figure {float:left; margin:0 20px 0 0;}
    #midcol {width:1100px; margin:auto; }
    p {margin-top:0;}
</style>
```

The figures (which are just containers for the pictures and captions) are floated left and some shorthand has been used for styling the margins. The four margin numbers **margin:0 20px 0 0;** represent the sizes of the margins starting with the top margin and moving round the figure clockwise. They are therefore: top margin zero, right margin 20 pixels, and zero margins at the bottom of the figure and left of the figure. The *midcol* is given a fixed width to prevent float-drop on smaller screens, the auto margin gives equal margins on each side of the *midcol*.

```
<nav>
  <ul>
    <p>Menu</p>
    <li class="btn"><a href=page-2.html>Go to Page 2</a></li>
    <li class="btn"><a href=gallery.html>Gallery</a></li>
    <li class="btn"><a href=page-4.html>Go to Page 4</a></li>
    <li class="btn"><a href=index.html>Home Page</a></li>
  </ul>
```

The menu is made to match the pages in the template, and the menu will now link to the gallery page.

```
    <div id ="rightcol">
      <p>This is the far right column</p>
    </div>
<div id="midcol">
  <h2>Gallery</h2>
  <p>This is the content</p>
```

The right column is deleted, the <h2> heading is changed to *Gallery*, and the line <p>This is the content</p> is deleted.

A new picture is added at the end of each row, that is, *Morning news* and *Mr. Friedman*.

■ **Congratulations!** You have now learned sufficient HTML and CSS to produce acceptable websites and you have eight adaptable templates to help you. However, good web design is not just about HTML and CSS, so be sure to read Chapter 17 because the topics such as focus, quality, and usefulness are just as important as coding. Later in this book you will encounter topics such as choosing and processing images, search engine optimization, typography, and user psychology. Several useful CSS techniques will also be described such as tabbed menus, adding a logo to a header, and drop shadows. In Chapter 40 you will find instructions on hosting your website and transferring the site to a host.

In the Advanced section of the book, you will be introduced to a little piece of PHP code that will make the design and maintenance of your websites much easier and quicker. The PHP examples will also allow you to introduce a safe feedback form and passwords into your websites.

Summary

You downloaded four new templates that will provide you with horizontal menus. You were shown the logical process that produced horizontal menus. You learned how to combine ready-made code from another chapter with a template to produce a new gallery page. You then removed the vertical menu and the far right column, and this provided extra horizontal space so that one more picture could be added to each row of pictures in the gallery page.

In the next chapter you will learn how to prepare pictures suitable for websites.

CHAPTER 13



More about Website Images: Create a New Appearance with Tiles

Browsers will display three image file formats. In this chapter, the three formats are described and their pros and cons discussed.

This chapter contains the following sections:

- Image file formats suitable for websites
- Preparing pictures for a website
- Creating a new appearance using background tiles
- Tweaking the CSS style sheet
- Explanations of the code
- Creating pictures suitable for the header section
- Summary

Image File Formats Suitable for Websites

Modern browsers and also Internet Explorer 8 can display three types of graphic files:

GIF, JPEG, and PNG. The files have the following formats: *mypic.gif*, *mypic.jpg*, and *mypic.png*

The meaning and pronunciation of each acronym is as follows:

GIF (pronounced 'giff' not 'jiff') means Graphics Interchange Format

JPEG or JPG (pronounced jaypeg) means Joint Photographic Experts Group

PNG (pronounced "ping") means Portable Network Graphics

Use GIF images for displaying pictures with large areas of flat color, for example, logos, diagrams, cartoons, and text; this is because GIF is restricted to 256 colors. GIF files allow one color to be transparent. A GIF file can be given loss-less compression, that is, it won't lose quality if it is compressed.



Figure 13-1. A GIF image. The .gif picture on the left was quite a large file that I compressed to 67 kilobytes. The area surrounding the man and the computer was originally yellow. I chose to make the yellow color transparent, and as a result it looks white on this white background. I use this figure on my computer-help website, which has a pale turquoise background color; the transparent area therefore allows the pale turquoise background to show through. To see this, view my website: <http://www.colycomputerhelp.co.uk>

JPEG images are used for photographs and they can display 16 million colors with smooth gradation between colors. The files do not support transparent colors. However, JPEG files can be compressed to produce very small file sizes that are ideal for fast-loading web images. The quality of the image deteriorates if the compression is great, or if the file is saved several times.



Figure 13-2. The JPEG picture on the left was a photograph that started off with a file size of 1.29MB. It was resized and compressed and is now only 95 kilobytes.

Use PNG files when advanced features are required; these will be described later. Meanwhile accept the fact that PNG files combine the best features of GIFs and JPEGs. Because of these extra features, PNG files are a little larger than JPEGs.



Figure 13-3. When I received the PNG image on the left, it already had a relatively small file size of 485 kilobytes. I reduced the dimensions of the picture then compressed it to produce this 188 kilobyte PNG image. I had no need to make use of the transparent properties of PNG for this image.

Figure 13-4. Because it has very few colors and required a transparent background, the rosette could have been a GIF file, but I chose to save it as a PNG file. I made use of the transparency properties of PNG files to surround it with a transparent area. I was then able to superimpose it on the header image of a website so that the header's background image was visible in the transparent area. Its final file size was only 6 kilobytes.



Preparing Pictures for a Website

Pictures produced by a camera or scanner are usually much too large and need cropping, compressing, and enhancing. You will also need to decide which file format would be best for your web page. The procedure for achieving this requires the following steps:

1. Decide which would be the best format to suit your web page, that is, GIF, JPEG, or PNG. If you need a transparent area you will be limited to GIF or PNG.
2. Reduce the physical dimensions of the picture (resize it) and crop it to fit the layout of your page. Record the dimensions (in pixels) so that you can include them in the HTML code for the picture.
3. If the picture is a PNG or a JPEG, compress the file so that it loads quickly.
4. Adjust the colors, sharpness, and contrast if necessary.

You can achieve the last three steps using a graphics program such as Paint Shop Pro, Adobe Photoshop Elements, the free program Irfan view, or the free program Gimp. Be sure to read Chapter 41 to learn more about optimizing an image. In Windows 7, 8, or 10 a combination of Paint and Windows Live Photo Gallery can be used for steps 2 and 3. Use the program's *Help* files for instructions on resizing and compressing JPEG and PING files.

■ **Warning** Many unsafe free image compression programs are offered on the Internet. Many contain content such as search bars that will corrupt your computer. Chapter 41 contains tips on how to safely download free graphics manipulators.

For your guidance, there were eight pictures in the gallery page in the previous chapter and each picture had the following properties:

Average width: 230 pixels. Average height: 300 pixels.

Average original file size: 1.5MB. Average compressed file size: 22.9 kilobytes.

Creating a New Appearance Using Background Tiles

We will now use the home page to demonstrate how a website's appearance can be transformed by small changes to the page and the CSS styles. Please follow these steps:

1. Create a new folder called *Chapter-13*.
2. Go to your *templates* folder, and open *template-3d-hmenu*.
3. Make a copy of all the files in the folder *template-3d-hmenu* (use Ctrl+A, then Ctrl+C) and paste them (use Ctrl+V) into the *Chapter-13* folder.
4. Download the *Chapter-13.zip* file and unzip it in your *Chapter-13* folder.
5. The downloaded folder contains a folder named *new-images*; open it and copy and paste the new images into the *Chapter-13 images* folder.

This exercise demonstrates how, by using your own templates, you have complete control over colors, layouts, backgrounds, and contents. By taking the pale blue template with 3D buttons, and tweaking it with the CSS style sheet, you will produce a completely different website. The home page is shown in Figure 13-5. To save space, only the home page will be demonstrated in this chapter.

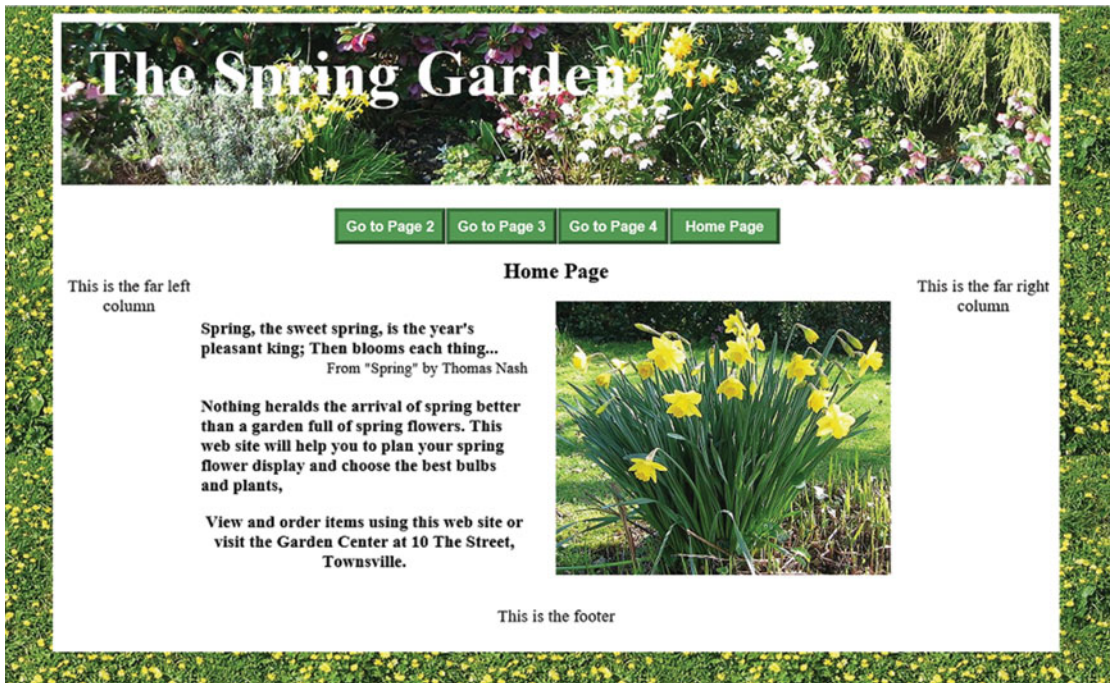


Figure 13-5. The template-3d-hmenu.css was used to create this completely different appearance

Tweaking the CSS Style Sheet

We will now make some small changes to the CSS style sheet. These will create the new appearance shown in Figure 13-5. Please follow these steps:

1. Open the style sheet *style-3d.css* in your HTML text editor.
2. Use *Save As* to save it as *style-3d-gdn.css*.
3. Change the items shown in bold type in Listing 13-1.
4. At the end of the code, add the two extra lines shown in bold type in Listing 13-1.

Listing 13-1. Tweaking the style sheet *style-3d-gdn.css*.

```
body {background-image:url(images/spring-tile-2.jpg);}
#wrapper {margin:auto; max-width:1100px; min-width:980px; background:white; ←
border:10px white solid; font-size:medium;}
header,nav,article,section,figure,footer {display:block;}
header {margin-top:0; border-bottom:10px white solid; height:181px;
background-image:url('images/spring-header.jpg');}
h1 {position:absolute; top:-15px; margin-left:30px; font-size:450%; color:white; ←
font-weight:bold;}
h2 {font-size:150%; text-align:center;}
p {font-size:120%; text-align:center;}
a {font-size:120%;}
```

```

/*add a left col*/
#leftcol {float:left; width:150px;}
#rightcol {float:right; width:150px; height:190px;}
#midcol {margin-left:155px; margin-right:155px;}
br.clear {clear:both}
#midcol-left {float:left; width:46%; font-weight:bold;}
#midcol-right {float:right; width:50%; font-weight:bold;}
/*set horizontal button menu*/
nav ul {float:left; width:960px; height:40px; margin-left:70px; text-align:center; ↵
padding-left:0; list-style-type:none;}
/*set general horizontal menu button styles*/
nav ul li { display:inline-block;}
li.btn {width:115px; line-height:20px; margin-bottom:3px; margin-right:5px; ↵
text-align :center;}
/* set general access (anchor) styles */
li.btn a {display:inline-block; width:115px; color:white; font-family:arial; ↵
font-size:small; font-weight:bold; text-decoration:none;}

/* mouseout (default) */
li.btn a {background:#559A55; border:4px outset #559A55;}
/* mouseover */
li.btn a:hover {background:red; border:4px outset red;}
/* mousedown */
li.btn a:active {background:maroon; border:4px inset maroon;}
.left {text-align:left;}
.quote {float:right; font-weight:normal; font-size:90%;}

```

5. Save the file.

Explanation of the Code

```
body {background-image:url(images/spring-tile-2.jpg);}
```

This code introduces you to a new way of creating a background, and it is called tiling. In previous chapters the background was a plain color, but tiling allows you to use a pictorial background using tiles. We will use a tile named *spring-tile.jpg* shown in Figure 13-6.



Figure 13-6. The tile named *spring-tile.jpg* is a 500 pixel square and was created from a photograph of a bank of celandines

The code for tiling the body (see Listing 13-1) fills the whole area of the body with tiles. This is the default style, but it is possible to place a single row of tiles vertically or horizontally. This will be described in Chapter 15.

```
h1 {position:absolute; top:-15px; margin-left:30px; font-size:450%; color:white; ↵
    font-weight:bold;}
```

A negative top position pulls the text “The Spring Garden” upwards so that it appears over a darker area of the header image. This makes the header text easier to read.

```
#wrapper {margin:auto; max-width:1100px; min-width:980px; background:white;
```

In this line, the width of the wrapper is reduced to show more of the tiling on the left and right of the page. The background color of the wrapper is changed to white to give a white panel for the content. You can see the effect of these changes in Figure 13-5.

```
background-image:url('images/spring-header.jpg'));
```

The header now has a background image that is more appropriate for a Spring Garden website.

```
/*set horizontal button menu*/
nav ul {float:left; width:960px; height:40px; margin-left:70px; text-align:center; ↵
padding-left:0;
```

The width of the horizontal menu is reduced to match the reduced width of the wrapper. The left margin is changed to 70 pixels to center the horizontal menu block on larger screens.

```
/* mouseout (default) */
li.btn a {background:#559A55; border:4px outset #559A55;}
/* mouseover */
li.btn a:hover {background:red; border:4px outset red;}
/* mousedown */
li.btn a:active {background:maroon; border:4px inset maroon;}
```

The 3D blue buttons are changed to a shade of green to match the theme of the website.

.left {text-align:left;}

This line has been added to change the text in the *middleleft* column from centered text to left-aligned text.

.quote {float:right; font-weight:normal; font-size:90%;}

This line was added to change the appearance of the poem's citation text and to push it to the right; see Figure 13-5.

Altering the home page *index.html*

We will now alter the home page to a spring garden page. Please follow these steps:

1. In your HTML text editor, open the file *index.html*.
2. In the Code/Source pane make the changes shown in bold type in Listing 13-2.

To save space, only the changed part of the code is shown.

Partial Listing 13-2. Showing the changed parts of the code for the home page.

```
<link href="style-3d-gdn.css" rel="stylesheet" type="text/css">
  <!--[if lte IE 8]>
    <script src="html5.js">
      </script>
  <![endif]-->
</head>
<body>
  <div id="wrapper">
    <header>
      <h1>The Spring Garden</h1>
    </header>
    <nav>
      <ul>
        <!--<p>Menu</p>-->
        <li class="btn"><a href="page-2.html" title="Go to page 2">Go to Page 2</a></li>
        <li class="btn"><a href="page-3.html" title="Go to page 3">Go to Page 3</a></li>
        <li class="btn"><a href="page-4.html" title="Go to page 4">Go to Page 4</a></li>
        <li class="btn"><a href="index.html" title="Return to Home page">Home Page</a></li>
      </ul>
```



```

        </nav>
<div id="content">
<div id="leftcol">
    <p>This is the far left column</p>
</div>
<div id="rightcol">
    <p>This is the far right column</p>
</div>
<div id="midcol">
    <h2>Home Page</h2>
<div id="midcol-left">
    <p class="left">Spring, the sweet spring, is the year's pleasant king; ~
    Then blooms each thing...<br>
    <span class="quote">From "Spring" by Thomas Nash</span><br>
    <p class="left">Nothing heralds the arrival of spring better than a garden full of ~
    spring flowers. This website will help you to plan your spring flower display and ~
    choose the best bulbs and plants./p>
    <p>View and order items using this website or visit the Garden Center at 10 The ~
    Street, Townsville.</p>
</div>
<div id="midcol-right">
    
</div>

```

3. Save the file

Explanation of the Code Changes

```
<link href="style-3d-gdn.css" rel="stylesheet" type="text/css">
```

This code changes the link so that it will access the new style sheet *style-3d-gdn.css*.

```

<div id="wrapper">
  <header>
    <h1>The Spring Garden</h1>
  </header>

```

This code changes the text in the page header.

■ **Note** Please add the two changes above to the other three pages in the website.

```

<nav>
  <ul>
    <p>Menu</p>
    <li class="btn"><a href="page-2.html" title="Go to page 2">Go to Page 2</a></li>
    <li class="btn"><a href="page-3.html" title="Go to page 3">Go to Page 3</a></li>

```

```

    <li class="btn"><a href="page-4.html" title="Go to page 4">Go to Page 4</a></li>
    <li class="btn"><a href="index.html" title="Return to Home page">Home ◀
Page</a></li>
</ul>
</nav>

```

Here you are introduced to four additional items beginning **title=** "... These provide pop-up *Tool tips* to help blind and partially sighted persons who use a reader that speaks out the name of each link.

```

<div id="midcol-left">
  <p class="left">Spring, the sweet spring, is the year's pleasant king; Then ◀
    blooms each thing...<br>
  <span class="quote">From "Spring" by Thomas Nash</span><br>
  <p class="left">Nothing heralds the arriva ofspring betterthan a garden full ◀
    of spring flowers.This website will help you to plan your spring flower display ◀
    and choose the bes bulbs and Plants.</p>
    <p>View and order items using this website or visit the Garden Centerat10 ◀
    The Street, Townsville</p>
</div>

```

The text is changed to something more appropriate for a Spring Garden website:

```

<div id="midcol-right">
  
</div>

```

Add a more suitable picture and include its width and height in the code.

Now view the home page in a modern browser; it should look like Figure 13-5. Try shrinking the horizontal width of the browser to ensure that no float-drop occurs. In the home page, test the rollover effect on the menu buttons and rest your cursor on them to see the pop-up Tool tips.

■ **Note** The appearance of the home page in the Design pane of Expression Web 4 may not be quite the same as Figure 13-5, but this is normal so don't worry. Press F12 to see how it will look in a modern browser.

Creating Pictures Suitable for the Header Section

Headers often need panoramic pictures, and digital camera photos usually have a natural width greater than 1600 pixels. Therefore by cropping a horizontal slice from a wide picture you can produce a letter-box shaped image suitable for the header area. The header picture in this chapter was cropped from a large picture; its final size is 1199 pixels wide by 2017 pixels high. It was optimized for fast loading to a file size of 140 kilobytes.

Summary

This chapter described suitable file formats for website images, and it described the general procedure for preparing pictures for a website. You created a completely different website using one of your templates. You learned about the technique of *tiling* for background images. A suggestion for creating header images was provided.

In the next chapter you will create a website containing both horizontal and vertical menus.

CHAPTER 14



Vertical and Horizontal Menus on the Same Page: Colored Columns

A website with many pages will need many more menu buttons, so the simplest approach is to create an additional menu block so that pages have both vertical and horizontal menus.

This chapter contains the following sections:

- Tweak the home page *index.html*
- Change the other three pages
- Tweak the style sheet
- Create four templates containing both horizontal and vertical menus
- Summary

Please follow these steps:

1. Create a new folder called *Chapter-14*.
2. Go to the folder *Chapter-13*.
3. Make a copy of all the completed files (use Ctrl+A then Ctrl+C) and paste them (use Ctrl+V) into the *Chapter-14* folder.
4. From this book's page on Apress.com, download the zip folder for Chapter-14 and unzip it into your *Chapter-14* folder.
5. From within the downloaded *Chapter-14* folder, insert the image **grenswrl.gif** into your images folder.

We will use the Spring Garden website to create a template with vertical and horizontal menus, and also with tiled side columns. By tweaking the HTML page and the CSS style sheet, you will produce a website as shown in Figure 14-1.

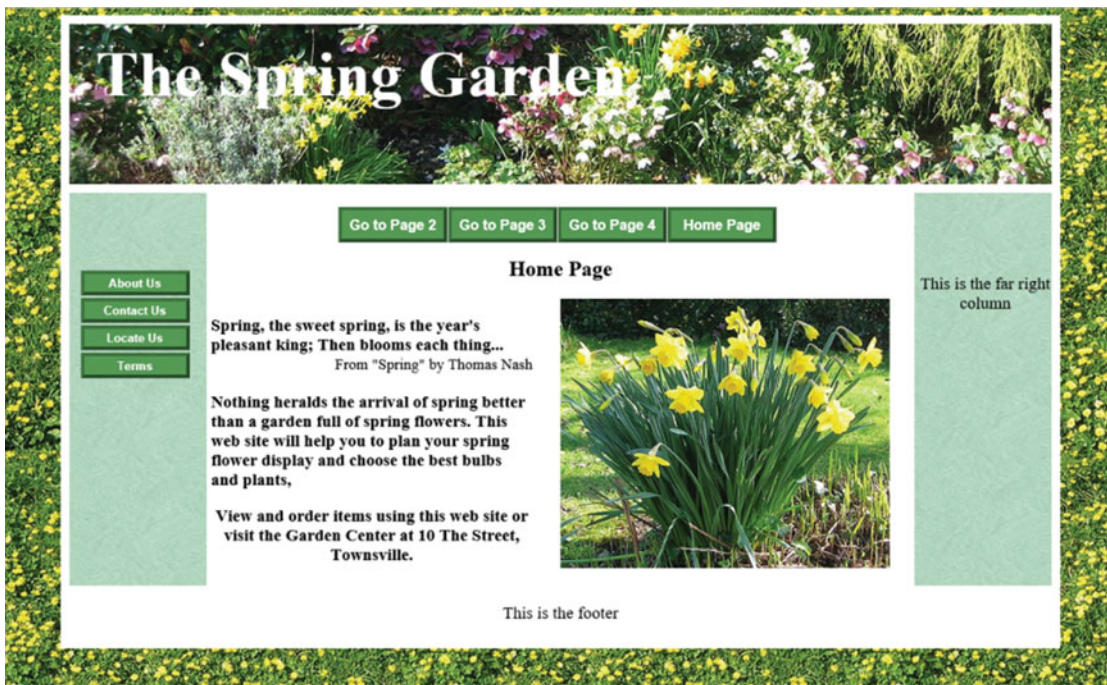


Figure 14-1. A page with vertical menu, horizontal menu, tiled background, and tiled side columns

Tweak the Home Page *index.html*

1. In folder *Chapter-14*, open the file *index.html* in your HTML text editor. This file already has a horizontal menu.
2. Just below the `</nav>` tag, at the end of the first navigation block, delete the lines shown crossed through, and insert a block of code for the additional vertical menu shown in bold type in Listing 14-1.

Listing 14-1. Tweak the home page *index.html*

```

</nav>
<br class="clear">
<div id="content">
<div id="leftcol">
  <p>This is the far left column</p>
  <nav id="vertical">
    <ul>
      <li class="vbtn"><a href="#" title="About Us">About Us</a></li>
      <li class="vbtn"><a href="#" title="Contact Us">Contact Us</a></li>
      <li class="vbtn"><a href="#" title="Locate Us">Locate Us</a></li>
      <li class="vbtn"><a href="#" title="Terms and Conditions">Terms</a></li>
    </ul>
  </nav>
</div>

```

```
<div id="rightcol">
    <p>This is the far right column</p>
</div>
```

The rest of the code is unchanged, but make sure the code **<br class=clear>** exists in the last section of the HTML as shown in bold type in the next snippet of code.

```
<div id="midcol-right">
    
</div>
    <br class="clear"><!--push the next closing divs downwards clear of the two ↵
    floated columns-->
</div><!--midcol div finishes here-->
</div><!--content div finishes here-->
<footer>
    <p>This is the footer</p>
</footer>
</div><!--wrapper div finishes here -->
</body>
</html>
```

3. Save the page.

■ **Important** The symbol, ↵, is not part of the code so ignore it when typing the code into your file. The symbol indicates that a line of code is continued on the next line.

Explanation of the Code

```
</nav>
<br class="clear">
<div id="content">
<div id="leftcol">
    <p>This is the far left column</p>
```

The two lines shown crossed through are no longer required because the extra line space **<br class="clear">** is not necessary, and the far left column no longer needs a heading.

```
<nav id="vertical">
    <ul>
        <li class="vbtn"><a href="#" title="About Us">About Us</a></li>
        <li class="vbtn"><a href="#" title="Contact Us">Contact Us</a></li>
        <li class="vbtn"><a href="#" title="Locate Us">Locate Us</a></li>
        <li class="vbtn"><a href="#" title="Terms and Conditions">Terms</a></li>
    </ul>
</nav>
</div>
```

This code inserts the vertical block of menu buttons. To differentiate the vertical buttons from the horizontal buttons they have been given a new class, that is, `class="vbtn."`

The text displayed on the vertical buttons has been changed in anticipation of creating four new pages. However, we will not be creating those new pages in this chapter because they are not necessary for the purpose of this exercise. The vertical buttons therefore have dead links, that is, `href="#"`.

■ **Tip** A reminder of why we sometimes use a hash symbol "#" instead of a file name: this little trick avoids the appearance of the "Page not available" error message when the link is clicked. In a real world situation, you would, of course, create the four new pages and use their *actual* file names instead of the hash symbols. The hash symbol "#" creates a dead link.

Change the Other Three Pages

This next step is important because the resulting pages will demonstrate how the colored side columns automatically grow and shrink to match different page lengths.

Change *page-2.html*, *page-3.html*, and *page-4.html* in the same way as you changed the home page. The instructions for the amendments are the same as or the home page and are given in Listing 14-2.

1. In folder *Chapter-14*, open each file in your HTML text editor. These file already have a horizontal menu.
2. Just below the `</nav>` tag, at the end of the first navigation block, delete the lines shown crossed through, and insert a block of code for the additional vertical menu shown in bold type in Listing 14-2.

Listing 14-2. Change the other three pages

```
</nav>
<br class="clear">
<div id="content">
<div id="leftcol">
<p>This is the far left column</p>
<nav id="vertical">
<ul>
<li class="vbtn"><a href="#" title="About Us">About Us</a></li>
<li class="vbtn"><a href="#" title="Contact Us">Contact Us</a></li>
<li class="vbtn"><a href="#" title="Locate Us">Locate Us</a></li>
<li class="vbtn"><a href="#" title="Terms and Conditions">Terms</a></li>
</ul>
</nav>
</div>
<div id="rightcol">
  <p>This is the far right column</p>
</div>
```

The rest of the code is unchanged, but make sure the code `<br class=clear>` exists in the last section of the HTML as shown in bold type in the next snippet of code.

```

<div id="midcol-right">
  
</div>
  <br class="clear"><!--push the next closing divs downwards clear of the two ↵
  floated columns-->
</div><!--midcol div finishes here-->
</div><!--content div finishes here-->
  <footer>
    <p>This is the footer</p>
  </footer>
</div><!--wrapper div finishes here -->
</body>
</html>

```

3. Save the three pages.

Tweak the Style Sheet

Because the vertical menu block needs its own CSS style, we will now insert a new style into the style sheet *style-3d-gdn.css*. We will also introduce some color into the far left and far right columns; this will be produced by means of *tiling*. By using tiling we can make the colored columns automatically grow or shrink vertically to match the length of individual pages. To achieve this without tiling, we would have to specify the height of the columns on every individual page.

1. Open the file *style-3d-gdn.css* in the code/source pane of your HTML text editor.
2. Amend the lines in the style sheet as shown in bold type in Listing 14-2.
3. Insert the styling for the vertical menu buttons as shown in bold type in Listing 14-2.

Listing 14-2. Tweak the CSS file *style-3d-gdn.css* to add color to the side columns

```

body {background-image:url(images/spring-tile-2.jpg);}
#wrapper {margin:auto; max-width:1110px; min-width:1045px; background:white; ↵
border:10px white solid; font-size:medium;}
header,nav,article,section,figure,footer {display:block;}
header {margin-top:0; border-bottom:10px white solid; height:181px; ↵
background-image:url('images/spring-header.jpg');}
#content {background-image:url('images/grenswrl.gif'); min-width:980px;}
h1 {position:absolute; top:-15px; margin-left:30px; font-size:450%; color:white; ↵
font-weight:bold;}
h2 {font-size:150%; text-align:center;}
p {font-size:120%; text-align:center;}
a {font-size:120%;}
#leftcol {float:left; width:150px;}
#rightcol {float:right; width:150px; height:190px;}
#midcol {margin-left:155px; margin-right:155px; background:white; padding:5px;}

```

The rest of the code is unaltered, but at the end of the code add the following new styles for the vertical menu buttons. These are targeted by the CSS class *.vbtn*

```
/*set vertical button menu column*/
nav#vertical ul {float:left; width:150px; margin-left:-5px; padding-left:0; ←
list-style-type:none;}
/*set general side button styles*/
li.vbtn {width:115px; line-height:20px; margin-bottom:3px; text-align:center;}
/* set general anchor styles */
li.vbtn a {display:block; width:115px; color:white; font-family:arial; ←
font-size:small;
font-weight:bold; text-decoration:none;}
/* mouseout (default) */
li.vbtn a {display:block; background: #559a55; border:4px outset #559a55;}
/* mouseover */
li.vbtn a:hover { background:red; border:4px outset red;}
/* mousedown */
li.vbtn a:active {background:maroon; border:4px inset maroon; }
```

4. Save the CSS file.

Explanation of the Code

```
#wrapper {margin:auto; max-width:1110px; min-width:1045px; ←
background:white; border:10px white solid; font-size:medium;}
```

In case the user shrinks the browser window horizontally, the max and min widths are altered to prevent the right-hand column from sliding over the pictures and text in the *midcol* area.

```
#content {background-image:url('images/grenswrl.gif'); min-width:980px;}
```

The content area is filled with tiles to create colored columns.

```
#midcol {margin-left:155px; margin-right:155px; background:white; padding:5px;}
```

We need to change the *midcol* area to white so that it stands out from the tiled content area. The padding gives a five-pixel space around the inside of the *midcol* area. This prevents text from butting against the sides of the *midcol* area.

```
/*set vertical button menu column*/
nav#vertical ul {float:left; width:150px; margin-left:-5px; padding-left:0; ←
list-style-type:none;}
/*set general side button styles*/
li.vbtn {width:115px; line-height:20px; margin-bottom:3px; text-align:center;}
/* set general anchor styles */
li.vbtn a {display:block; width:115px; color:white; font-family:arial; ←
font-size:small;
font-weight:bold; text-decoration:none;}
```



```

/* mouseout (default) */
li.vbtn a {display:block; background: #559a55; border:4px outset #559a55;}
/* mouseover */
li.vbtn a:hover {background:red; border:4px outset red;}
/* mousedown */
li.vbtn a:active {background:maroon; border:4px inset maroon;}

```

The vertical block of menu buttons is contained within a side column that is floated left. This column is given a width of 150 pixels. The vertical 3D buttons are targeted by using the class **.vbtn**

Open the home page in a modern browser and move from page to page using the *horizontal* menu. Watch how the length of the colored columns varies automatically to match the different page lengths. Also test the vertical menu to see how the use of a hash symbol prevents nonexistent pages from displaying the error message “Page not available.”

Create Four Templates Containing Both Horizontal and Vertical Menus

We will now create four more templates containing horizontal and vertical menus.

1. Open your *templates* folder and create four new folders (the letters h-v indicate horizontal and vertical menus) Name the folders as follows:

 template-3d-h-v
 template-3d-h-v-round
 template-plain-h-v
 template-plain-h-v-round
2. Open your folder *Chapter-14* and copy all the files and paste them into three of the four template folders you have just created but **not** in the folder *template-plain-h-v-round*.
3. Open the folder *template-3d-h-v-round*.
4. In the folder *template-3d-h-v-round*, open the file *style-3d-gdn.css* in your HTML text editor.

To provide the rounded borders, at the end of the existing code, add the code shown in the next snippet.

```

#wrapper {border-radius:15px;}
#content {border-radius:6px;}
header {border-top-left-radius:6px; border-top-right-radius:6px;}

```

5. Save the file.
6. Open the folder *template-plain-h-v* and in your HTML text editor then open the file *style-3d-gdn.css*.
7. Save it as *style-plain-gdn.css*.
8. In *style-plain.css* please change the buttons' backgrounds to plain colors and set the *mouseover* and *mousedown* styles as shown in bold type in Listing 14-2:

Listing 14-2. Create the plain gray menu buttons

```
/* set general access (anchor) styles */
li.btn a {display:block; width:115px; color:white; background:gray; font-family:arial; ↵
font-size:100%; font-weight:bold; text-decoration:none;}
/* mouseover */
li.btn a:hover {background:blue;}
/* mousedown */
li.btn a:active {background:black;}

/*set vertical button menu column*/
nav#vertical ul {float:left; width:150px; margin-left:-5px; padding-left:0; ↵
list-style-type:none;}
/*set general side button styles*/
li.vbtn {width:115px; line-height:20px; margin-bottom:3px; text-align:center;}
/* set general access (anchor) styles */
li.vbtn a {display:block; width:115px; color:white; font-family:arial; font-size:small; ↵
font-weight:bold; text-decoration:none;}
/* mouseout (default) */
li.vbtn a {display:block;background:gray;}
/* mouseover */
li.vbtn a:hover {background:blue;}
/* mousedown */
li.vbtn a:active {background:black;}
```

9. Save the file.
10. In the folder *template-plain-h-v* we now need to change the styling links in the four pages so that they link to the plain menu style sheet *style-plain-gdn.css*.
Open each page and change the style link as follows:
`<link href="style-plain-gdn.css" rel="stylesheet" type="text/css">`
11. Save each of the amended pages.
12. Copy all the files in the folder *template-plain-h-v* (use Ctrl+A then Ctrl+C) and paste them into the folder *template-plain-h-v-round* (use Ctrl+V).
13. Open the folder *template-plain-h-v-round* and in your HTML text editor open *style-plain-gdn.css*.
14. To style the border's rounded corners, add the following three lines of code to the end of the style sheet *style-plain-gdn.css*:
#wrapper {border-radius:15px;}
#content {border-radius:6px;}
header {border-top-left-radius:6px; border-top-right-radius:6px;}
15. Save the file.

You now have a total of 12 templates and these can be tweaked to provide almost any style of website. By using a template, the structure and the styling are all coded and ready for you to add your own content and pictures. Although the four new templates in this chapter used the spring garden theme, the body background can be changed easily by using different tiles or reverting to plain white or using a plain color. You can also change the color of the wrapper's border or use *border:none*: to remove the border altogether.

Feel free to adapt the templates to produce your own websites. The picture in the *header* background can be changed by substituting another picture, or you can fill the header area with tiles. You can change the tiling in the left and right columns of the content area, or you can choose to omit the tiling to produce white side columns. If you need more horizontal space in the content area, just shrink the **width** of the far right column in the CSS style sheet. The colors of the menu buttons are easily changed by specifying different colors in the style sheet.

You can add more pages by making “Save as...” copies of existing pages, change the page titles, and then add more menu buttons to all the pages to allow them to access the new pages. The page shown in Figure 14-1 was created in this way.

Summary

In this chapter you learned how to produce a website with both vertical and horizontal menu blocks. You also discovered how to add extendable colored side columns by using tiles. We then created four new templates with horizontal and vertical menus based on the new home page. Dozens of background tiles are included in the folder named backgrounds.

In the next chapter you will learn how to create your own tiles for backgrounds.

CHAPTER 15



Create Tiles: Use Two New Tiles: Float-Drop and the Box Model

The previous chapter demonstrated how tiles can be used for backgrounds in websites. This chapter contains instructions for creating your own background tiles to match the theme of your website.

You will learn how to change the appearance of a website using two new tiles. In addition you will learn about float-drop and the box model.

This chapter contains the following sections:

- More about using tiles in the background of a website
- Create your own tiles for backgrounds
- Create a different appearance using two new tiles
- Float-drop
- The box model
- Summary

Tip So that readers of this book can see the colors of the tiles and of the finished project, I have included a PDF document in the downloadable folder for this chapter. I have also included a folder packed with colored tiles that you can use it in future websites.

More About Using Tiles in the Background of a Website

As already described in the previous chapter, tiling is a method of repeating an image either horizontally or vertically or both. Tiling can be achieved with any JPEG, GIF, or PING image, and the image can be square or rectangular. To fill the entire background of a page or any container such as a `<div>` or a semantic tag, the tile is repeated both horizontally and vertically using the *default* code as follows:

```
body {background-image: url("tile.jpg");}
```

In the example code above, the container (the body) will be filled with tiles no matter whether the body is large or small. By using the default code, tiles can be made to completely fill any container of any size (not just the body).

This property is very useful because tiling can accommodate web pages of any length or width. Moreover, the container's widths and depths do not have to be multiples of the tile size. For example, using tiles 100 pixels square in a horizontal `<div>` with a height of 150 pixels, displays a horizontal strip one and a half tiles high.

■ **Note** If you wish to fill a container with tiles and the container is not a multiple of either the height or width of the tile, then use the default setting, that is, don't include a *background-repeat* command in the styling. Using the default, the tiles will never extend beyond the boundary of a container.

The following code will tile the background image horizontally across the page (only one tile high):

```
body {background-image: url("tile.jpg"); background-repeat:repeat-x;}
```

The following code will tile the background image vertically down the page (only one tile wide):

```
body {background-image: url("tile.jpg"); background-repeat:repeat-y;}
```

However, it is rarely practical to have a banner that is only one tile deep or a sidebar only one tile wide.

Create Your Own Tiles for Backgrounds

The downloadable folder for this chapter contains a folder full of background tiles. However, it is fun to create your own original and unique tiles to match your website's theme. All you need is a digital camera or a scanner and a graphics program such as Gimp, IrfanView, Paint Shop Pro, or Adobe Photoshop Elements.

Types of Tile

Distinguishable tiles

If you can see where each tile is abutting against the next tile, they are *distinguishable tiles*.

They are easy to create; just photograph or scan something, resize the image to the size of the desired tile, then compress the file so that the tiled background will load into the user's display quickly. You might take photographs of textured items such as a towel, a brick wall, a quilted silk bedspread, a closely patterned carpet, a piece of slightly crumpled colored paper, or aluminum foil, and so on. There will inevitably be a gradient of lighting across the image that will cause the tiles to be distinguishable.

Semi-seamless tiles

When it is just possible to see where adjoining tiles butt against one another, they are called *semi-seamless tiles*. These can be created by photographing or scanning surfaces that have no clearly discerned repeated pattern. The bank of celandines tile in Chapter 14 was a semi-seamless tile; you would need to look very carefully to discern where one tile butts against another. Figure 15-1 shows four semi-seamless tiles. I photographed the outside wall of my house, which has a coating of thick paint that has a pronounced but non-repeated texture. I then made several copies of the photograph and changed the color balance of each tile in my graphics program.

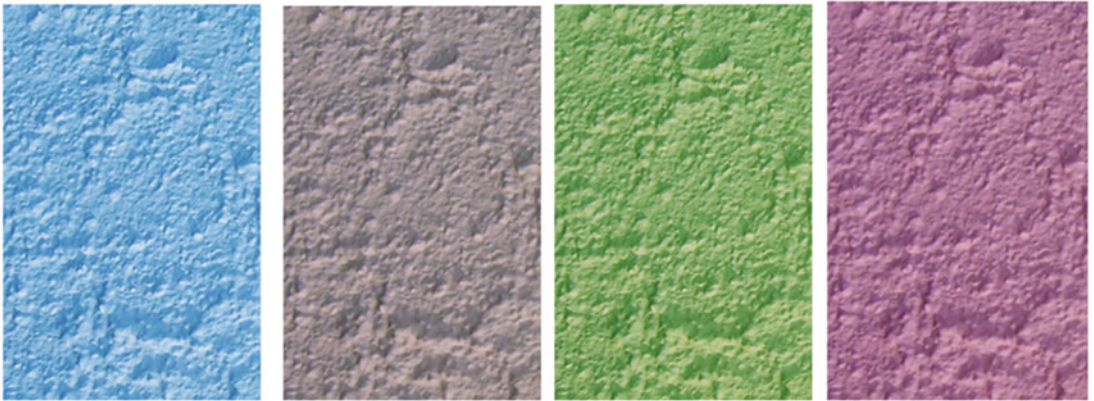


Figure 15-1. Four semi-seamless tiles created from a photograph of a painted wall

Seamless tiles

When you cannot see where abutting tiles begin or end they are called *seamless tiles*. Creating these can be tricky. The problem is how do you prevent the inevitable gradient of light from top to bottom or side to side?

You might think this can be overcome by having the light source (the sun or an electric light) directly above the item being photographed; however, when you stand over the object to take the photo, your own shadow spoils the image.

A possible solution would be to set up the object and the camera, then darken the room (night time is best), and take a flash photograph.

For better results, produce tiles using a scanner. With a good quality scanner the lighting will automatically be reasonably uniform across the tile so that seamless tiles will be produced.



Figure 15-2. This shows a pictorial tile produced by scanning a leaf placed on a piece of colored paper. Such tiles are sometimes used in the body section of the HTML code.

Pictorial tiles

However, be wary of this technique because you may cause the page to look spotty and spoil the focus, that is, the user will be distracted away from the center of focus, which should always be the content panel and the menu buttons. To reduce the distraction you could use a blurred image in the body's background.

Create a Different Appearance Using Two New Tiles

We will now use one of our templates to produce a home page with a mauve theme. To save space and time we will use the Spring Garden template. Mauve is hardly suitable for a garden theme, but it will demonstrate how easy it is to change the appearance of a page with CSS and some tiles. The result of the next project should be as shown in Figure 15-3.

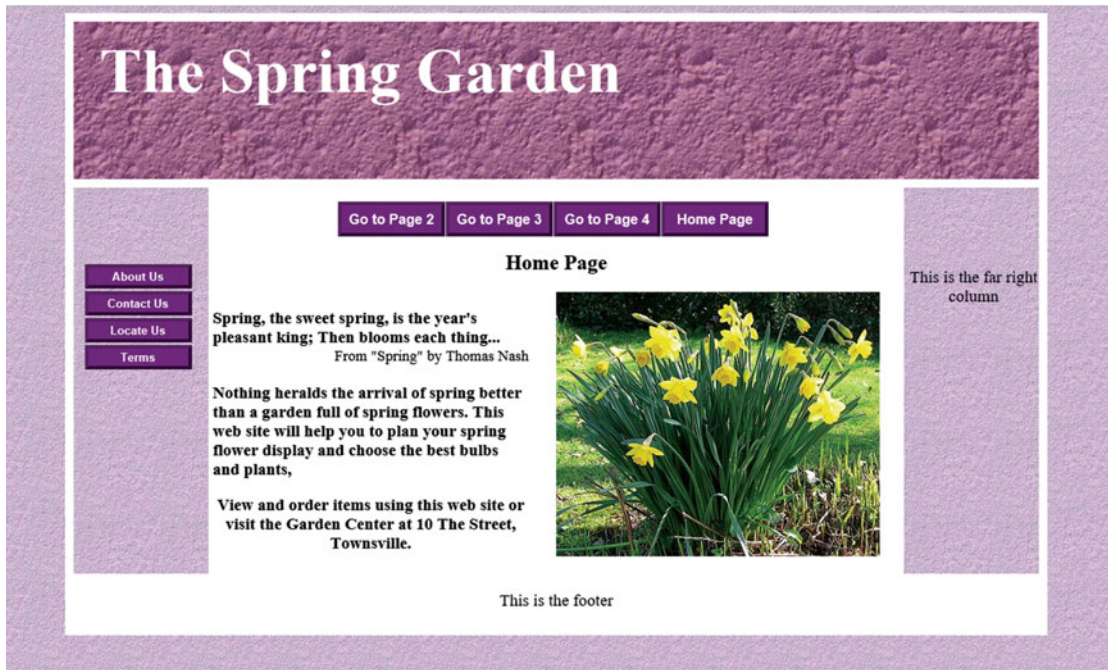


Figure 15-3. Showing the application of a mauve theme to one of our templates

Create a New Folder for Chapter 15

To create the page shown in Figure 15-3, follow these steps:

1. In the *web-tutorial* folder, create a new folder named *Chapter-15*.
2. Open your *templates* folder and then open the folder *template-h-v-3d*.
3. Copy all the files in the folder *template-h-v-3d*, use Ctrl+A then Ctrl+C to copy them into the computer's memory.
4. Open the folder *Chapter-15*.
5. Paste the files into the folder *Chapter-15* using Ctrl+V).
6. Download the Chapter-15 zip-file and unzip it in your Chapter-15 folder. The unzipped folder contains three new tile images.
7. In the folder *Chapter-15*, right-click the file *style-3d-gdn.css* and rename it *tiles.css*.
8. Right-click *tiles.css* and choose to open it with your HTML editor.
9. In Code/Source view amend the file *tiles.css* as shown in bold type in Listing 15-1.

Listing 15-1. Make some small amendments to the style sheet tiles.css

```

body {background-image:url('images/lilacstone.gif');}
#wrapper {margin:auto; max-width:1110px; min-width:1045px; background:white; ↵
border:10px white solid; font-size:medium;}
header,nav,article,section,figure,footer {display:block;}
header {margin-top:0; border-bottom:10px white solid; height:181px; ↵
background-image:url('images/tile-lilac.jpg');}
/*h1 {position:absolute; top:45px; margin-left:30px; font-size:450%; color:white; ↵
font-weight:bold;}*/
#content {background-image:url('images/lilacstone.gif'); min-width:980px;}
h1 {position:absolute; top:-15px; margin-left:30px; font-size:450%; color:white; ↵
font-weight:bold;}
h2 {font-size:150%; text-align:center;}
p {font-size:120%; text-align:center;}
a {font-size:120%;}
/*add a left col*/
#leftcol {float:left; width:150px;}
#rightcol {float:right; width:150px;}
#midcol {margin-left:155px; margin-right:155px; background:white; padding:5px;}
br.clear {clear:both}
#midcol-left {float:left; width:46%; font-weight:bold;}
#midcol-right {float:right; width:50%; font-weight:bold;}
/*set horizontal button menu*/
nav ul {float:left; width:960px; height:40px; margin-left:70px; text-align:center; ↵
padding-left:0;
list-style-type:none;}
/*set general horizontal menu button styles*/
nav ul li { display:inline-block;}
li.btn {width:115px; line-height:2; margin-bottom:3px; margin-right:5px;text-align :center;}
/* set general access (anchor) styles */
li.btn a {display:inline-block; width:115px; color:white; font-family:Arial; font-size:100%; ↵
font-weight:bold;text-decoration:none;}
/* mouseout (default) */
li.btn a {background:purple; border:4px outset purple;}
/* mouseover */
li.btn a:hover {background:red; border:4px outset red;}
/* mousedown */
li.btn a:active {background:maroon; border:4px inset maroon;}
.left {text-align:left;}
.quote {float:right; font-weight:normal; font-size:90%;}

/*set vertical button menu column*/
nav#vertical ul {float:left; width:150px; margin-left:-5px; padding-left:0; ↵
list-style-type:none;}
/*set general side button styles*/
li.vbtn {width:115px; line-height:20px; margin-bottom:3px; text-align:center;}
/* set general anchor styles */
li.vbtn a {display:block; width:115px; color:white; font-family:arial; font-size:small; ↵
font-weight:bold;text-decoration:none;}
/* mouseout (default) */
li.vbtn a {display:block; background:purple; border:4px outset purple;}

```



```
/* mouseover */
li.vbtn a:hover {background:red; border:4px outset red;}
/* mousedown */
li.vbtn a:active {background:maroon; border:4px inset maroon; }
```

10. Save the file in your *Chapter-15* folder as *tiles.css*.

Now Link the New Style Sheet to the Home Page

1. In the Chapter-15 folder open the file *index.html* in your HTML text editor.
2. In Code/Source view amend the link as shown in bold type in the next snippet of code:

```
<link href="tiles.css" rel="stylesheet" type="text/css">
```

3. Save the file.
4. Amend the style link in the other three pages to the same code as in step 2 and save the files.
5. View the revised files in your browser (or press F12 in Expression web); they should show the new appearance as shown in Figure 15-3.

Float-Drop

As a beginner you need to know about a problem called float-drop and also to learn about the box model, which is the basis of all web page elements.

Sometimes a picture or another page element drops down below the row where it is supposed to be located. This can happen to elements surrounded by tags (<div>, , , <p>, <h1> etc) if they are floated within a container that is too narrow to accommodate all the floated items within a row. We call this phenomenon *float-drop*. Figure 15-4 shows a typical float-drop.



Figure 15-4. The floated menu buttons were not given sufficient room in the menu's container

Float-drop can be avoided by calculating the total width of the floated elements in the row, then making the container large enough to match the total width. The math sounds simple enough, but there is a catch. To calculate the total width of an element, you must add border thicknesses, padding dimensions, and margins to the width of the content as shown in Figure 15-5.

The Box Model

You should try to memorize the principle of the box model because it applies to all HTML tags and can save you time and frustration when elements unexpectedly drop down on a web page.

The width and height of an image must always be specified; this ensures that a space can be reserved for it in the page displayed on the screen. If the image's dimensions are not specified, or are not exactly the same as the image, the resulting display will not be what you intended.

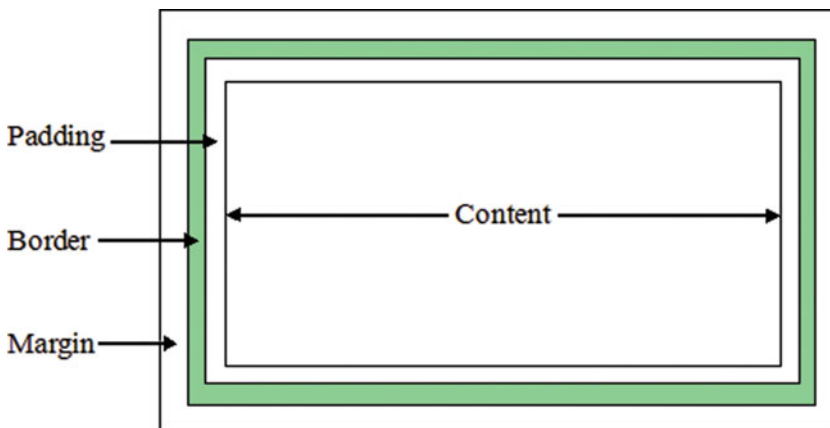


Figure 15-5. *The box model*

Let's assume that the box in Figure 15-5 has content that is an image 300 pixels wide with a border, some padding, and some margins. Let's give those items the following dimensions:

```
img {width:300px; height:150px; padding:5px; border:1px black solid; margin:5px;}
```

The total width of the box is $300 + 2 \times 5 + 2 \times 1 + 2 \times 5 = 322$ pixels

Assume that the web page has three of these pictures floated to the left in a horizontal row, the width of the row would be $3 \times 322 = 966$ pixels

To avoid float-drop, the container for the row of pictures must have a fixed width greater than 966 pixels, or if, in the case of a semi-liquid layout, a minimum width greater than 966 pixels.

Some browsers interpret padding and margins differently. Though the differences are slight they can prove puzzling to beginners when trying to avoid float-drop. The best solution is to zero-out the differences by setting all tag margins and padding to zero at the start of your style sheet; this is called a CSS browser reset. The following zero reset can be inserted at the beginning of your style sheets:

```
h1, h2, h3, h4, h5, h6, p, ol, ul, li, form, fieldset {padding, margin:0;}
img {border:0}
```

Summary

In this chapter we summarized the code for applying tiled backgrounds. We then discussed a few ways of creating your own background tiles. We demonstrated how easy it is to tweak a template using CSS and so produce a completely different color theme.

You then learned how to avoid float-drop, and you were introduced to the important concept of the box model. You also learned how to zero-out any differences in padding and margins caused by some browsers.

In the next chapter you will learn how to create data tables using HTML and CSS.

CHAPTER 16



Create Tables for Data

Tables should **never** be used for page layout, although in the past many web designers used them for this purpose. The practice is now deprecated (obsolete and bad practice). Tables must now be used only for the purpose for which they were originally intended, i.e., for presenting data.

This chapter contains the following sections:

- Create a simple two-column table
- Inserting and deleting a table or rows and columns in a WYSIWYG editor
- Explore the default double border style
- Placing a four-column table within a web page
- Summary

Create a Simple Two-Column Table

A simple two-column data table is shown in Figure 16-1.

Events at the Town Hall, Townsville	
22nd August 2014	10.00am. Coffee morning in aid of The Underwater Basket Weavers
3rd September 2014	3.00pm. Jumble sale in aid of the Society for the Protection of Mice

Figure 16-1. A simple two-column data table

Take the following steps:

1. In the folder named *web-tutorial* create a folder named *Chapter-16*.
2. In the Code/Source view of your HTML text editor, type the code shown in Listing 16-1.
3. Save it as *simple-2col-table.html*.

Listing 16-1 has an internal style sheet for instructional purposes only. The style should normally be contained in the main style sheet.

Listing 16-1. Create a two column table *simple-2col-table.html*

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset=utf-8>
    <title>simple-2col</title>
    <style type="text/css">
      table {width:670px; border:1px black solid; border-collapse:collapse;
        font-family:Arial; font-size:100%;}
      td {border:1px black solid; padding:5px;}
      caption {font-weight:bold;}
    </style>
  </head>
  <body>
    <table>
      <caption>Events at the Town Hall,Townsville</caption>
      <tr>
        <td><strong>22nd August 2014</strong></td>
        <td>10.00am. Coffee morning in aid of The Underwater Basket Weavers</td>
      </tr>
      <tr>
        <td><strong>3rd September 2014</strong></td>
        <td>3.00pm. Jumble sale in aid of the Society for the Protection of
          Mice </td>
      </tr>
    </table>
    <br>
  </body>
</html>

```

4. Use *Save As* to save the file in the folder *Chapter-16*. Give the file the name *simple-2col-table.html*.
5. View the file in your browser.

Explanation of the Code

```

<style type="text/css">
  table {width:670px; border:1px black solid; border-collapse:collapse;
    font-family:Arial; font-size:100%;}
  td {border:1px black solid; padding:5px;}
  caption {font-weight:bold;}
</style>

```

In the internal CSS style code, the second line creates a table 670 pixels wide, with a one pixel black border. Always specify a table width; otherwise it will spread out to fit the space required to display the table data. Note the format for configuring the table and cell borders. The border is given the attribute `border-collapse:collapse`; this ensures that the borders are single lines. Each cell is surrounded by the tags `<td></td>` and the cells have a padding of 5 pixels to provide a gap between the text and the borders.

When using the table in your own websites, extract the code between the `<style>` tags and paste it into the CSS style sheet. Then delete the internal style and the `<style>` tags from the web page.

```
<body>
  <table>
    <caption>Events at the Town Hall, Townsville</caption>
    <tr>
      <td><strong>22nd August 2014</strong></td>
      <td>10.00am. Coffee morning in aid of The Underwater Basket Weavers</td>
    </tr>

    <tr>
      <td><strong>3rd September 2014</strong></td>
      <td>3.00pm. Jumble sale in aid of the Society for the Protection of Mice</td>
    </tr>
  </table>
</body>
</html>
```

Tables are created using the tags `<table></table>`

The text that describes a table is called a *caption* and it has a `<caption>` tag as shown in the following line:

```
<caption>Events at the Town Hall, Townsville</caption>
```

The tag `<tr></tr>` stands for table **row**

The tag `<td></td>` stands for table **data**, that is, the content of a table cell

Inserting and Deleting a Table or Rows and Columns in a WYSIWYG Editor

You can also use Expression Web 4 or Blue Griffon in the Design/WYSIWYG view to create, amend, or delete a table. Click *Table* in the top menu bar. In the drop-down menu rest your cursor on *Insert* or *Delete*, then select the appropriate action in the fly-out menu. In Expression Web 4 or Blue Griffon, use the *Design/WYSIWYG* view to insert or delete rows or columns as follows: click anywhere in the row or column and then click *Table* in the top menu bar. In the drop-down menu rest your cursor on *Insert* or *Delete*, then select the appropriate action in the fly-out menu. You can then type in the cell contents just as if you were using a word processor.

Explore the Default Double Border Style

If you do not collapse the borders they will be double line borders as shown in Figure 16-2.

Events at the Town Hall, Townsville	
22nd August 2014	10.00am. Coffee morning in aid of The Underwater Basket Weavers
3rd September 2014	3.00pm. Jumble sale in aid of the Society for the Protection of Mice

Figure 16-2. The default style's double borders

Tip The default double border is not popular and it does look rather fussy compared with the clean single line borders.

To explore the default border style, please follow these steps:

- 1. Open the file *simple-2col-table.html* in your text editor and use *Save As* to save a copy of the file with the name *no-collapse-2col.html*.
- 2. In that file delete the code in the internal style shown crossed through in Listing 16-2.
- 3. Save the file.

Listing 16-2. Demonstrate the default double borders

```
<style type="text/css">
  table {width:670px; border:1px black solid; border-collapse:collapse;
        font-family:Arial; font-size:100%;}
  td {border:1px black solid; padding:5px;}
  caption {font-weight:bold;}
</style>
```

- 4. Open the file *no-collapse-2col.html* in a browser to view the default table double borders.

Placing a Four-Column Table within a Web Page

We will use one of our templates to create a home page for a travel firm.
The page is shown in Figure 16-3.

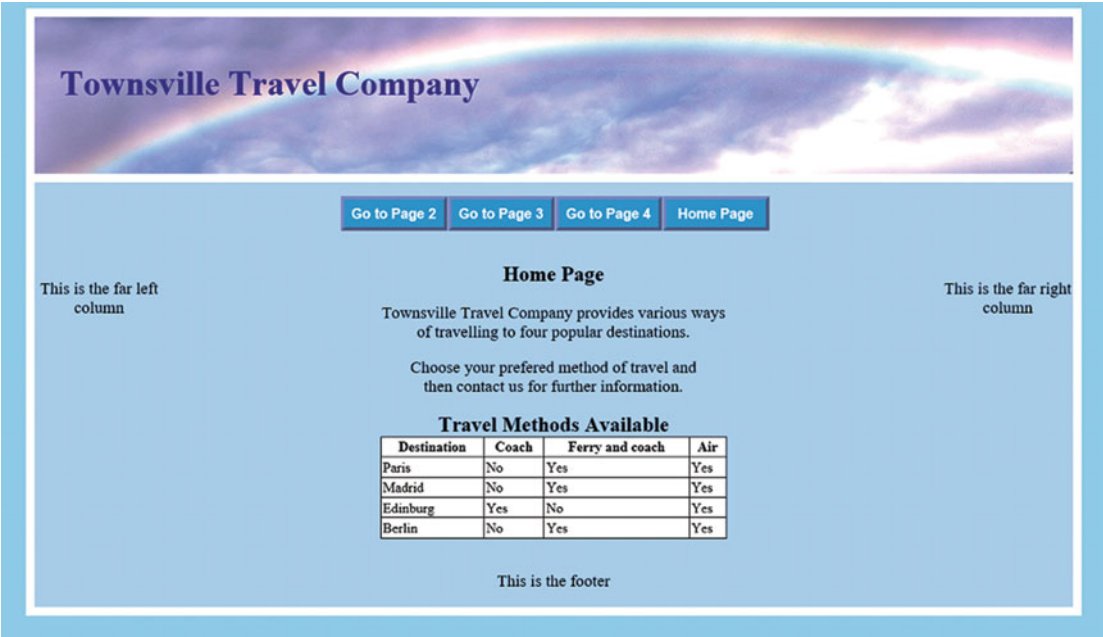


Figure 16-3. Shows a four-column five-row table embedded in a web page

To produce the page shown in Figure 16-3 follow these steps:

1. Open your *Templates* folder, then open the folder *template-3d-hmenu*.
2. Hold down the *Ctrl* key and tap the *A* key to select all the files.
3. Then hold down the *Ctrl* key and tap the *C* key to copy all the files into the computer's memory.
4. Open the folder *Chapter-16* and paste in the files (Ctrl+V).
5. Open the file *index.html* in your text editor's code view and delete the items shown crossed through.
6. Make the additions and amendments to the file *index.html* as shown in bold in Listing 16-3.

The page contains an internal style because the table and its style are unique to that demonstration page.

Listing 16-3. Create a web page containing a four-column table

```
<!DOCTYPE html>
<html>
  <head>
    <title>Home Page</title>
    <meta charset=utf-8>
    <link href="style-3d.css" rel="stylesheet" type="text/css">
    <!--[if lte IE 8]>
      <script src="html5.js">
      </script>
    <![endif]-->
    <style type="text/css">
      table {width:400px; margin:auto; border-collapse:collapse; ←
        border:1px black solid;background:white;}
      td, th {padding:2px; border-collapse: collapse; border:1px black solid;}
      caption { font-size:150%; font-weight:bold;}
    </style>
  </head>
  <body>
    <div id="wrapper">
      <header>
        <h1>Townsville Travel Company</h1>
      </header>
      <nav>
        <ul>
          <li class="btn"><a href="page-2.html">Go to Page 2</a></li>
          <li class="btn"><a href="page-3.html">Go to Page 3</a></li>
          <li class="btn"><a href="page-4.html">Go to Page 4</a></li>
          <li class="btn"><a href="index.html">Home Page</a></li>
        </ul>
      </nav>
      <br class="clear">
    <div id="content">
    <div id="leftcol">
```

```

    <p>This is the far left column</p>
</div>
<div id="rightcol">
    <p>This is the far right column</p>
</div>
<div id="midcol">
    <h2>Home Page</h2>
    <div id="midcol-left">
        <p>Townsville Travel Company provides various ways<br>of travelling to ←
            four popular destinations</p>
    <div id="midcol-right">
        <p>Choose your preferred method of travel and<br>then contact us for ←
            further information.</p>
    </div>
</div>
<br class="clear">
<table>
    <caption><strong>Travel Methods Available</strong></caption>
    <tr>
        <th>Destination</th>
        <th>Coach</th>
        <th>Ferry and coach</th>
        <th>Air</th>
    </tr>
    <tr>
        <td>Paris</td>
        <td>No</td>
        <td>Yes</td>
        <td>Yes</td>
    </tr>
    <tr>
        <td>Madrid</td>
        <td>No</td>
        <td>Yes</td>
        <td>Yes</td>
    </tr>
    <tr>
        <td>Edinburgh</td>
        <td>Yes</td>
        <td>No</td>
        <td>Yes</td>
    </tr>
    <tr>
        <td>Berlin</td>
        <td>No</td>
        <td>Yes</td>
        <td>Yes</td>
    </tr>
</table></div>
<br class="clear">
<footer>

```



```

        <p>This is the footer</p>
    </footer>
</div>
</div><!--wrapper div finishes here -->

</body>
</html>

```

Explanation of the HTML Code

A table with five rows and four columns has been added using *Code View*.

Note the new tag `<th>` in the first block of table code.

```

<tr>
    <th>Destination</th>
    <th>Coach</th>
    <th>Ferry and coach</th>
    <th>Air</th>
</tr>

```

`<th>` is a special type of table cell, and it stands for **table header cell**. This tag behaves exactly the same as the standard cell tag `<td>` except that text within a `<th>` cell is bold and centered by default. Text within a `<td>` cell has *normal* font weight and is *left-aligned* by default.

Explanation of the Internal Style

```

<style type="text/css">
    table {width:400px; margin-left:400px; border-collapse:collapse;
        border:1px black solid; background:white;}
    td, th {padding:2px; border-collapse: collapse; border:1px black solid;}
    caption { font-size:150%; font-weight:bold;}
</style>

```

The table is positioned and set with collapsed black borders and a white background. The white background makes the table stand out from the background color of the page. The cells are set with collapsed borders. When using a table in your own websites, extract the code between the `<style>` tags and paste it into the CSS style sheet. Then delete the internal style and the `<style>` tags from the web page.

Summary

In this chapter you learned how to produce a simple two-column table. You explored the default view, which displays double borders. You then produced a four-column table embedded into a web page using a plain text editor. Instructions were given for inserting and deleting tables or rows and columns in the Design/WYSIWYG view of an HTML text editor.

In the next chapter, you will learn some of the important rules for designing an attractive and successful website. You will also discover how to design a feedback form.

CHAPTER 17



The Secret of Attractive and Useful Websites

To ensure success a website must be: Carefully planned, attractive and useful. It should also be based on a knowledge of web-user psychology (how users access and view websites).

This chapter contains the following sections:

- Basic rules for an attractive website
- That all-important home page
- User psychology: How surfers use websites
- Websites should be useful
- Create a web page with an encoded email address
- Summary

Basic Rules for an Attractive Website

Good websites will follow the commonsense rules clearly expressed in Steve Krug's groundbreaking book *Don't Make Me Think* (now in its third edition, ISBN 978-0-321-96557-6). His usability rule is used by all properly trained web masters; it states the following:

"When I look at a web page it should be self-evident. Obvious. Self-explanatory. I should be able to 'get it' – what it is and how to use it – without expending any effort thinking about it."

To achieve this goal I have listed the following rules:

Planning: Careful planning is essential whether you are designing your own website or a client's website. A client's website should of course be planned in collaboration with the client. Start with commonsense questions such as the following: Who is the intended audience for the website? What is the main purpose of the website? What is the secondary purpose of the website? Keep the answers to these questions firmly in mind as you continue to plan. Sketch a flow diagram to indicate the main topics and show how the user will navigate to those topics.

Attractiveness is judged by four criteria: focus, color, ease of navigation, and clarity.

Focus: Focus is the strategy employed by a web designer to persuade the user to concentrate on the important aspects of a web page. Focus is achieved by avoiding clutter. Each web page should focus on *one* topic only. Pages that repeat previous information are exasperating to users; it makes them feel that they are getting nowhere fast. The principle of one topic per page is also extremely helpful to the designer and also to the client when planning the website. By discussing the topics with the client and listing them, the designer will know how many pages and how many menu buttons will be required. This also assists the designer when quoting a price for the website.

Color: The careful use of color can attract a user. Garish colors and predominantly dark colors are repellent. Users will linger longer on a web page that uses restful pastel colors or various shades of one color. A page covered in many patches of different color can be irritating and will certainly spoil the focus. Be sure to read the section later in this chapter about using a strongly contrasting color for text.

An inappropriate color choice indicates that the web producer not only has no training but also lacks common sense. Examples of inappropriate colors: (i) a carpet cleaner's DIY site that was predominately black and gray. This should have used clean pastel colors. (ii) a wedding photographer's DIY site that was predominately black. Black is quite unsuitable for a wedding photographer; surely a pale peach and white would have been more suitable? Also white or colored text bleeds when placed in a black background. (iii) a nature conservation DIY site that was orange shading to peach. Colors associated with nature, such as shades of green and browns, would have been more appropriate.

Clarity is achieved when this happens:

- Each web page focuses on one topic only,
- The menu, header, and footer are in exactly the same position on every page ,
- All pages have a consistent color theme and layout,
- All pages are free from clutter,
- Links to other pages are not duplicated in the body of the home page.

The Navigation Menu Must Be Prominent

A navigation menu provides the means by which users move to the particular page of information they are searching for. The user should not have to scroll around or use trial and error to find a link to the topic they wish to view. Steve Krug states this:

“The user should never be more than one or two clicks away from the topic they are searching for.”

If you do need three clicks, make it logical and easy to get to that third click; anything that looks obscure or difficult will be ignored by users.

All this is common sense and is thoroughly tried and tested. The labels on the menu buttons must indicate clearly what is covered in the other pages. The menu must not be overpowered by surrounding clutter; it must stand out clearly. On the home page, the menu is the second most important item that you want the user to focus on (the first important item is the brief description of the website). The menu draws the user into the rest of the website. It must allow users to access the information they require quickly and easily. Use large prominent menu buttons rather than icons or pictures. Using thumbnail pictures or a geographical map with clickable hot spots would cost more and would download slower than HTML buttons.

Drop-down menus contradict the above requirement because they hide the content of the website's pages from the user. Microsoft and other software companies have realized the fact that drop-down menus are unfriendly and old fashioned, so they have replaced them with ribbon menus. The very popular BBC news website uses several rows of menu buttons instead of drop-down menus. Paint-by-numbers websites such as WordPress nearly always use drop-down menus that are constructed using JavaScript; search engines refuse to read JavaScript so that those websites appear to consist of only a home page, all subsequent pages are invisible to the search engines.

■ **Note** Having stated that drop-down menus are not user friendly, the advent of smartphones means that drop-down menus are now an acceptable way to avoid cluttering a smartphone's viewport with menu buttons rather than content. Chapters 35 to 38 deal with the topic of designing websites so that they are dual purpose, that is, websites that fit into a smartphone's viewport and will also expand to fill the viewport of a desktop computer.

The Use and Misuse of Text

Users want information quickly. If they have to wade through a sea of text they will go elsewhere for the information. Always précis the text to remove unnecessary verbiage. When I précis some text, I come back to it a day or two later to see if anything else can be removed or condensed without losing the meaning of the article. Insert appropriate pictures at intervals, and split the text into bite-size chunks; use bullets or headings where appropriate. Selling points are best highlighted by bullets.

Justified paragraphs are tiring to read on a printed page and are doubly so on a computer screen. The reason is clear: justified paragraphs have variable gaps between each word, so the brain has to keep adjusting to the changing gaps and this requires great concentration. Ragged right paragraphs have constant gaps so that the brain does not have to work so hard; this enables the user to concentrate on the content of a page. Justified paragraphs give an immediate impression of boring gray rectangular blocks, whereas ragged right paragraphs have a lively and more interesting appearance. Quality magazines always make use of this feature and use ragged right paragraphs.

Never spread text across a page. When the eye has reached the end of a wide line of text, the user turns his head back to the beginning of the line and is usually lost (and he eventually gives up the struggle). Use two columns instead. The accepted maximum width of a column is 5.25 inches (133 mm) on a resolution of 1024 pixels x 768 pixels. A website for children should have an even smaller column width.

Be particularly careful to use a contrasting color for text. Pale text on a pale background is extremely tiring to read, and users with less than perfect eyesight will skip to another website that can be read clearly. If you want users to read your text, use something like black or navy text on a white or pale background. Note that in some countries, such as UK and Australia, a poor contrast between text and background is illegal due to laws on the accessibility of websites for users with visual impairment.

Small fonts are not user friendly. The text in the main body should never be smaller than 100% for Times New Roman and never less than 90% for Arial or Verdana. Nobody will bother to fetch a magnifying glass to peer at a website set in 8 point font.

Try not to use more than two fonts on a page. Arial and Verdana are the preferred sans serif fonts for websites because they were designed for easy reading on computer screens. Times New Roman is the best of the serif fonts because it was specifically designed and tested for good readability in printed matter; however, it is not as good as sans serif for computer screens.

Simple Is Good

Visitors are not interested in your extensive vocabulary and erudite phrases; they just want facts clearly and briefly presented. Use plain English. If simplicity is well done, it can be more exciting than innovation or gimmicks.

That All-Important Home Page

The home page is the important introduction to your website and should be a model of clarity. It is the index to the other pages on the site. Of course, all the rules in the previous sections also apply to the home page, but the home page has some additional rules as follows:

The visitor to your site's home page only needs to know the answer to these two questions:

“Will I find what I am looking for on this website, and can I get to the various topics easily without having to scroll down and search around the home page?”

The important features on the home page are (i) a brief description of the website's purpose and (ii) an obvious and prominent means of navigating to the other pages.

State the Purpose of the Website Clearly and Concisely

A user will only stay with your website and navigate to the other pages if the content of the home page clearly matches the topic that they have searched for. This just requires a few concise but clear headings. The headings could use bullet points to highlight what the site contains. Sometimes a picture can proclaim the purpose of the website, in which case very little textual explanation will be required. Because the golden rule for websites is “One web page per topic,” do not repeat the content of these topics on the home page and do not have duplicate links to the other pages.

Keep it short. If possible, the home page should not exceed the height of one screen; one-screen deep is ideal. Don't force users to scroll down to find what they want, instead, tempt them with distinctive menu buttons to find the information they are seeking.

Avoid Gimmicks

On the home page, avoid gimmicks that spoil the focus such as blinking or flashing items, scrolling messages (marquees), slide shows, and audio or video – especially if they start automatically when the page is loaded. Items that move or jiggle are extremely irritating. If you have video or slides, put them on a separate video or slide show page.

Compare the home pages in Figures 17-1 and 17-2. The page on the left obeys all the rules for a good home page. The words “We specialise in embroidered logos” tells the user briefly and exactly what the website is about. The menu on the left tells the user what the subsequent pages contain. The colors are tasteful and the three bullet points give a little more information, just enough to tempt the user to explore the menu for more detailed information.



Figure 17-1. A clean uncluttered home page



Figure 17-2. Yuck!

The home page on the right Figure 17-2 is the top third of a very long and bewildering home page produced by an amateur using a CMS (paint-by numbers) program such as WordPress. The two huge menus are on either side of the page, so which should the user focus on? The user is confronted with a mass of text that is guaranteed to turn the visitor away. The center image is a self-starting video with loud music that is irritating and distracting.

User Psychology: How Surfers Use Websites

The great majority of users most certainly do not visit websites for a good read. Users search for some information, and they want that information quickly. An exception to this rule might be a website that is a showcase for an author's poems or anecdotes; this is an acceptable exception because the user is searching for poetry or anecdotes and is actually hoping for a good read.

User Experience (UX)

UX (user experience) is a tried and tested technique for discovering how users view websites. Some website development firms even employ UX designers. Their job is to test the websites with a cross-section of users to detect if they are puzzled by any aspect of a new website. As the UX designers experience grows, he or she can advise the design departments during the production stage of the website. I recommend that you test your websites on a range of users varying from experienced surfers to absolute beginners.

Slow Loading Websites Frustrate the User

Websites that are slow to load can be the result of using many uncompressed pictures per page. A page produced using a CMS program such as WordPress will also load slowly because it is packed with JavaScript that compensates for the designer's lack of knowledge of HTML and CSS code. A website created by a knowledgeable person using HTML and CSS will always be much leaner and faster. I am sometimes asked to change a CMS site to a properly coded site, and I have often been able to reduce the loading time from 18 seconds per page to less than 1 second per page.

Auto-start Audios, Videos, and Slide Shows

These gimmicks can cause users to quickly switch to another site. A sudden burst of sound can cause users to jump with fright and never to return to that site. Noise, and flickering or sliding pictures are irritating distractions that inevitably ruin the focus.

Audio video and slides must be used only where they explain things better than some text. They are best located on a special page and must have controls giving the user a choice whether or not to start the item.

Other Annoyances

Inconsistent page themes will put users off

Navigation menus that appear in a different position from page to page are particularly irritating. Changes of design or color can cause users to wonder if they are still in the same website.

If every available space on a page is filled

A cramped page will turn visitors away. Beginners feel they must fill a page with pictures and text, but space is a great aid to clarity. Make sure there is a margin between a container's border and the edge of a block of text or a picture, give containers at least 5 pixels of padding.

Websites Should Be Useful

A useful website will become a popular website

For example, a manufacturer of quiz buzzers could become more popular than other quiz buzzer manufacturers if his website contained useful things such as a selection of quiz questions and instructions on a variety of alternative quiz games. A website is useful if it has a recognizable branding, an easily understood navigation system, and a simple means of searching the website.

E-commerce sites should show the price of goods or services

Without prices, a commercial site is not very useful; the lack of prices will usually kill a prospective sale. Sensible users look for prices before committing themselves to a purchase. They certainly won't move from their computer to telephone for a price. Some bad websites announce the price only at the last moment when you click the Add-to-cart button. Sites that add sales tax and delivery charges at the very last stage of a transaction are worse still.

Remember that the Internet is a World Wide Web

Many smaller clubs and organizations seem to regard their website as a sort of internal news device, so the fact that some of the millions of viewers might be interested enough to join the organization and attend its events takes second place. A more useful and a better economic view would be to assume that the main target is the vast audience outside the organization, and the members would then be the secondary target.

Make it easy for users to contact the owner of the website

Display a menu button linking to a feedback form on a Contact Us page. A feedback form provides the safest and most useful method of contacting the owner; this will be introduced in Chapter 18. However, if the form is not carefully designed, hackers can hijack a form and use it to send malware to the site owner. Chapters 18 and 23 will teach you how to design a safe feedback form.

Information such as telephone number and postal address should be included in the footer of each page. Or if you prefer to have a live email address displayed on the pages instead of a feedback form, make sure it is encoded to prevent an avalanche of spam. If an email address is not encoded it will be harvested and sold to spammers. Both actions cause distress and an influx of spam to the site owner; and they don't enhance the reputation of the web designer.

We will now create a page containing a spam-proof email address.

Create a Web Page with an Encoded Email Address

To create a page containing a spam-proof email address, follow these steps:

1. In the *web-tutorial* folder, create a new folder named *Chapter-17*.
2. Create a new folder named *images*.
3. Create an image of the email address. Most graphics programs can create a *.png* or *.gif* image for the mail address.

■ **Tip** Microsoft's *Paint* program is ideal for this task. In *Paint* click the 'A' icon and draw a box big enough to contain the email address. Type the email address in the box and select the text to choose the font color and size. Choose to make the background transparent so that the background color of the page will show through. Select the font color by clicking the 'color 1' icon. Use the *Select* tool to select an area to fit snugly around the text. Click *Crop* then click anywhere outside the rectangle to make the box and its handles vanish. Then Click *File* in the top menu and hover over *Save As*. A selection of file formats will slide out. Select PNG and save the file with the name *youremail.png* into the *images* folder in your *Chapter-17* folder. Because it was saved as a transparent *.png* file, the background color of your web page will show through it. The image is shown in Figure 17-3.



Figure 17-3. The email address image

4. Open a new blank page in your text editor, save the file as *email-test.html*, and enter the code given in Listing 17-1. I have included an image of the dummy email address and a text file *java-script.txt* in the downloadable folder for Chapter 17. Copy the JavaScript and paste it into the code in your text editor. Of course you can use your own email address if you wish; in which case you will need to create an image of your own email address. The code is given in Listing 17-1:

Listing 17-1. Create an encoded email address within a web page *email-test.html*

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<style type="text/css">
body {text-align:center; margin:auto; margin-top:10px; font-size:120%; font-weight:bold;
width:500px;font-family:Arial; border:1px black solid;}
h2 {margin-bottom::5px;}
</style>
<title>Email test</title>
</head>
<body>
  <h2>Email Test</h2>
  To email me click:<br>
  <script type="text/javascript">
  <!--
    function escramble(){
      var a,b,c,d,e,f,g,h,i
      a='<a href=\"mai'
      b='youremail'
      c='\">'
      a+='lto:'
      b+='@'
```



```

        b+='yourisp.com'
        e='</a>'
        f=''
        g='<img src=\"'
        h='images/youreemail-1.png'
        i='\" alt="Email Me" border="0">'
        if(f) d=f
        else if(h) d=g+h+i
        else d=b
        document.write(a+b+c+d+e)
    }
    escramble()
//-->
</script>
<br><br>
</body>
</html>

```

■ **Caution** The last bracket in the function `escramble()` is a curly bracket. The variable `f=''` is followed by two single quotes, not a double quote.

The line `g='<img src=\"'` ends with a double quote followed by a single quote.

Explanation of the *escramble* JavaScript

The *escramble* script hides the email address `youreemail@yourisp.com`

```

<script type="text/javascript">
<!--
    function escramble(){
    var a,b,c,d,e,f,g,h,i
    a='<a href=\"mai'
    b='youreemail'
    c='\">'
    a+='lto:'
    b+='@'
    b+='yourisp.com'
    e='</a>'
    f=''
    g='<img src=\"'
    h='images/youreemail.png'
    i='\" alt="Email Me" border="0">'
    if(f) d=f
    else if(h) d=g+h+i
    else d=b
    document.write(a+b+c+d+e)
    }
    escramble()
//-->
</script>

```

The JavaScript breaks up both the email instruction and the email address (shown bold) so that spam spiders are unable to make sense of it. The line: `document.write()` adds together the items *a+b+c+d+e* and fills out the *To:* field of an email client such as *Windows Live Mail* on the computer.

If the image (item *h*) is not present it displays the clickable message *Email Me* instead of the clickable image (item *h*).

You don't need to understand the JavaScript, just grasp the principle. When inserting encoded email addresses in websites, simply enter your own or your client's details in place of the items *b* and the second instance of *b*+

Escramble is a free, anti-spam device. It was originally produced with a plain live link, and was later enhanced by the addition of an image of the email address by *InnerPeace.org*. Their web page at <http://innerpeace.org/escrambler.shtml> will quickly generate the code for you.

View the page *email-test.html* in a browser and you should see the page shown in Figure 17-4.

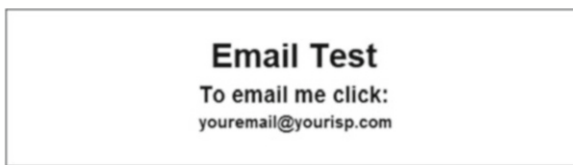


Figure 17-4. Showing the *email-test* page with its spam-proof live email link

If your page is correct and you have entered your own email address correctly, try clicking the link and it should allow you to send an email to yourself.

■ **Tip** You will not see the email picture in a text editor, but on your computer it will be visible in Mozilla Firefox. When it is uploaded users will be able to see the picture in any browser.

Clicking either the image or the words “Email Me” will cause the user's default email program to open with the email address already filled in. Users may need to be told that the email may go into their *Outbox* and therefore it must be sent by also clicking the *Send/Receive* button.

Summary

In this chapter you learned the main aspects of good web design. The chapter then described a simple JavaScript method of encoding email addresses. In the next chapter you will learn how to design a feedback form.

CHAPTER 18



Design a Feedback Form

Many websites use a feedback form so that users can interact with the owner of the website. This chapter describes how to create a typical form. The project will demonstrate once again that HTML code uses only plain English words or abbreviations of English words.

This chapter contains the following sections:

- Design a feedback form
- Add an internal style to position the form elements
- Add the form code to the *contact.html* page
- Summary

■ **Caution** At this stage the form will not send an email to the owner because that requires a tiny piece of PHP code that will be introduced in the Advanced section of this book. Forms need to be located on a server before they will function; this topic and PHP are dealt with in Chapter-23.

Design a Feedback Form

The form that you will create is shown in Figure [18-1](#).



Figure 18-1. The feedback form

To create the page shown in Figure 18-1, follow these steps:

1. In the *web-tutorial* folder, create a new folder named *Chapter-18*.
2. Open the *templates* folder and then open the folder *template-h-v-3d*.
3. Use Ctrl+A, then Ctrl+C to copy all the files into the computer's memory
4. Open the new folder *Chapter-18*.
5. Paste the items into the folder *Chapter-18* using Ctrl+V.
6. Download the file *form.txt* from the book's page at Apress.com and place the file in your *Chapter-18* folder.

7. Open the file *index.html* and change the second item in the vertical menu to provide a *Contact Us* button. The amendment is shown in bold type in the code snippet Listing 18-1.
8. In the other three pages change the second item in the vertical menu to provide a *Contact Us* button. The amendment is shown in bold type in the code snippet Listing 18-1.

Listing 18-1. Amend the vertical menu in the home page, and also in the other three pages

```
<div id="leftcol">
  <nav id="vertical">
    <ul>
      <li class="vbtn"><a href="#" title="About Us">About Us</a></li>
      <li class="vbtn"><a href="contact.html" title="Contact Us">Contact Us</a></li>
      <li class="vbtn"><a href="#" title="Locate Us">Locate Us</a></li>
      <li class="vbtn"><a href="#" title="Terms and Conditions">Terms</a></li>
    </ul>
  </nav>
</div>
```

9. Save the file four files.
10. Open file *page-2.html* and save a copy with the name *contact.html* using *Save As* and leave it on the screen ready for the next step.
11. First we need to change the page title to <title>Contact Us</title>
12. Change the <h2> heading in file *contact.html* to *Contact Us*.
13. Then delete the two mid-columns as shown crossed through in listing 18-2. This will make room on the page for the fields of the feedback form.

Listing 18-2. Creating space for a feedback form in the file *contact.html*

```
<!DOCTYPE html>
<html>
  <head>
    <title>Contact us</title>
    <meta charset=utf-8>
    <link href="style-3d-gdn.css" rel="stylesheet" type="text/css">
    <!--[if lte IE 8]>
      <script src="html5.js">
    </script>
  <![endif]-->
  </head>
  <body>
    <div id="wrapper">
      <header>
        <h1>The Spring Garden</h1>
      </header>
      <nav>
        <ul>
          <li class="btn"><a href="page-2.html" title="Go to page 2">Go to Page 2</a></li>
```

```

        <li class="btn"><a href="page-3.html" title="Go to page 3">Go to Page 3</a></li>
        <li class="btn"><a href="page-4.html" title="Go to page 4">Go to Page 4</a></li>
        <li class="btn"><a href="index.html" title="Return to Home page">Home Page</a></li>
    </ul>
</nav>
<div id="content">
<div id="leftcol">
    <nav id="vertical">
        <ul>
            <li class="vbtn"><a href="#" title="About Us">About Us</a></li>
            <li class="vbtn"><a href="contact.html" title="Contact Us">Contact Us</a></li>
            <li class="vbtn"><a href="#" title="Locate Us">Locate Us</a></li>
            <li class="vbtn"><a href="#" title="Terms and Conditions">Terms</a></li>
        </ul>
    </nav>
</div>
<div id="rightcol">
    <p>This is the far right column</p>
</div>
<div id="midcol">
    <h2>Contact Us</h2>
    <div id="midcol-left">
        <p>This is the midcol-left</p>
        <p>Some text in the left column</p>
    </div>
    <div id="midcol-right">
        <p>This is the midcol-right</p>
        <p>Some text in the right column</p>
    </div>
    <br class="clear">
</div><!--content div finishes here-->
</div>
<br class="clear">
<footer>
    <p>This is the footer</p>
</footer>
</div><!--wrapper div finishes here -->
</body>
</html>

```

Explanation of the Changes

```

<!DOCTYPE html>
<html>
  <head>
    <title>Contact Us</title>

```

Change the title to Contact Us

```

<div id="leftcol">
  <nav id="vertical">

```

```

<ul>
  <li class="vbtn"><a href="#" title="About Us">About Us</a></li>
  <li class="vbtn"><a href="contact.html" title="Contact Us">Contact Us</a></li>
  <li class="vbtn"><a href="#" title="Locate Us">Locate Us</a></li>
  <li class="vbtn"><a href="#" title="Terms and Conditions">Terms</a></li>
</ul>
</nav>
</div>

```

In the second menu item, replace the # with *contact.html* as shown above.

```

<div id="midcol">
  <h2>Contact Us</h2>
  <div id="midcol-left">
    <p>This is the midcol-left</p>
    <p>Some text in the left column</p>
  </div>
  <div id="midcol-right">
    <p>This is the midcol-right</p>
    <p>Some text in the right column</p>
  </div>
  <br class="clear">

```

The code shown crossed through removes the columns *midcol-left* and *midcol-right*. Save the file and view the page in a browser; it should look like Figure 18-2.

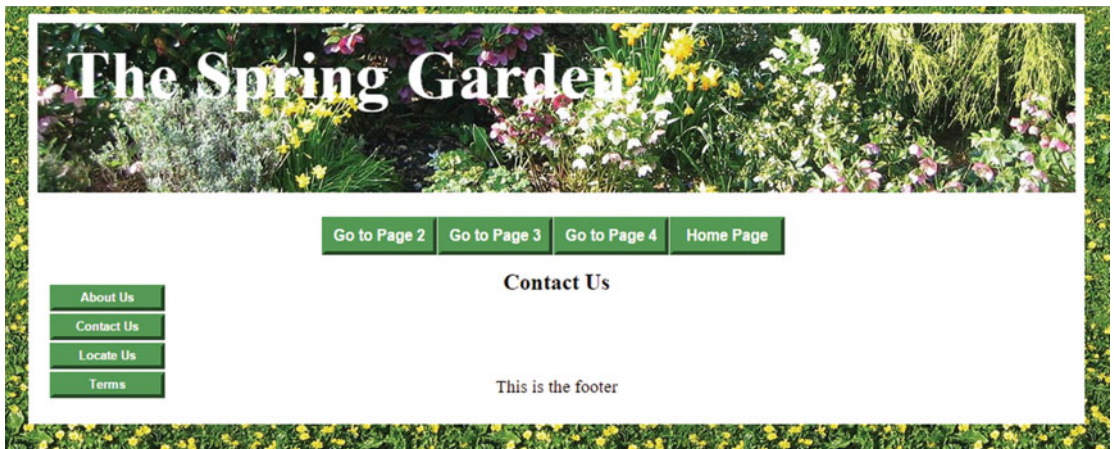


Figure 18-2. The Contact Us page with the two mid-columns deleted

Add an Internal Style to Position the Form Elements

In the next step we will add an internal style because the form style is unique to the “Contact Us” page. Now add the internal style to the *contact.html* page as shown in bold type in Listing 18-3.

Listing 18-3. Add an internal style for the feedback form

```
<!DOCTYPE html>
<html>
  <head>
    <title>Contact Us</title>
    <meta charset=utf-8>
<link href="style-3d-gdn.css" rel="stylesheet" type="text/css">
<style type="text/css">
  #midcol, #content {background:#aaddaa;}
  h3 {text-align:center;}
  /*Position the form elements on the page*/
  form {width:500px; margin:auto; text-align:center;}
  .label {float:left; width:400px; text-align:right; clear:left;}
  .chk1 {text-align:left; padding-left:30%;}
  #rad {text-align:left; padding-left:30%;}
  #message {text-align:center; margin:auto;}
  #message-field {text-align:left;}
  #submit { text-align:center;}
  footer {background:#aaddaa; text-align:center;}
  .red {color:red; font-weight:bold;}
</style>
<!--[if lte IE 18]>
```

Save the file *contact.html* and leave it on the screen ready for inserting the form into the space vacated by the deleted mid-columns.

Explanation of the File's Internal Style

Note that the internal style is in *addition* to the main style *style-3d-gdn.css*.

```
<style type="text/css">
```

The internal style begins with the `<style>` tag.

```
#midcol, #content {background:#aaddaa;}
```

This line provides a pale green background for the midcol area and the content

```
h3 {text-align:center;}
```

The h3 heading is centered on the page

```
/*Position the form elements on the page*/
```

This comment states that the next few lines style the form elements and position them

```
form {width:500px; margin:auto; text-align:center;}
```

This line sets the width of the form and centers it on the page

```
.label {float:left; width:400px; text-align:right; clear:left;}
```


This line floats the field label to the left and sets its width to 400 pixels

```
.chk1 {text-align:left; padding-left:30%;}
```

This class **.chk1** positions the check boxes to the left and gives them the equivalent of a left margin of 30% of the form width

```
#rad {text-align:left; padding-left:30%;}
```

This line positions the radio buttons to the left and gives them the equivalent of a left margin of 30% of the form width

```
#message {text-align:center; margin:auto;}
```

This line left aligns the content of the text area and centers the text area

```
#message-field {text-align:left;}
```

This line ensures the text entered by the user will be left aligned

```
#submit {text-align:center;}
```

This line centers the text in the submit button and centers the button in the form

```
footer {background:#aaddaa; text-align:center;}
```

This line provides a pale green background for the footer and centers the text in the footer

```
.red {color:red; font-weight:bold;}
```

This class line provides a class **.red** for changing the asterisk color to red

```
</style>
```

This line is the tag that closes the internal style

Add the Form Code to the *contact.html* Page

The downloadable file named *form.txt* contains the code for displaying the form elements. Open the file *form.txt* in Windows Notepad or Notepad++ (or in TextWrangler if you are using a Mac). With *form.txt* file open, press **Ctrl+A** then **Ctrl+C** to copy the form code into the computer's memory. Then open the file *contact.html* in the Code/Source pane, and immediately below the line `<h2>Contact Us</h2>` create a line space (i.e., press the enter key) and paste the form code into that line space using **Ctrl+V**.

Listing 18-4 shows the form code.

Listing 18-4. Inserting the code for the form into the file *contact.html*

```
<div id="midcol">
    <h2>Contact Us</h2>
    <h3>Required items <span class="red">*</span></h3>
    <form action="form-handler.php" method="post">
```

```
<label class="label" for="username"><strong>Your Name</strong><span class="red">*</span>
<input id="username" name="username" size="30"></label>
<br><br>
```

```
<label class="label" for="useremail"><strong>Your Email Address</strong><span
class="red">*</span>
<input id="useremail" name="useremail" size="30"></label>
<br><br>
```

```
<label class="label" for="userphone"><strong>Telephone Number </strong>
<span class="red">*</span>
<input id="userphone" name="userphone" size="30"></label>
<br>
```

```
<div>
<h3>Please email me the following brochure(s) <span class="red">*</span></h3>
</div>
```

```
<div class="chk1">
<input id="checkbox1" name="bulbs" value="Yes" type="checkbox">
<label for="checkbox1"><strong>Bulbs</strong></label>
</div><br>
```

```
<div class="chk1">
<input id="checkbox2" name="seeds" value="Yes" type="checkbox">
<label for="checkbox2"><strong>Seeds</strong></label>
</div><br>
```

```
<div class="chk1">
<input id="checkbox3" name="potting" value="Yes" type="checkbox">
<label for="checkbox3"><strong>Plants ready for potting on</strong></label>
</div>
```

```
<div>
<h3>How can we respond to your message?</h3>
</div>
<div id="rad">
<input id="email" type="radio" name="contact" value="email" checked>
<label for="contact"><strong>Email</strong></label>
<input id="phone" type="radio" name="contact" value="phone">
<label for="contact"><strong>Phone</strong></label>
</div>
<div id="message">
<label for="message"><h3><strong>Please type your enquiry</strong></h3></label>
<textarea id="message-field" name="message" rows="12" cols="40"></textarea>
</div>
```

```
<div id="submit">
<input id="submit" value="Submit" title="Submit" type="submit">
</div>
</form>
```

```
</div><!--content div finishes here-->
```

■ **Note** If you click the submit button you will see a ‘Page not available’ message; this is because, as yet, you have no file named *form-handler.php*. This file will be described in Chapter 23.

■ **Tip** Don't worry if you have difficulty with the following explanation. When adapting the code for your own use, you only need to load this code into a web page and then change some of the labels and attributes to match your requirements. However, try to understand the logic of the form elements and note that the code consists entirely of English words. Some additional information about form code is included in Chapter 41.

Explanation of the Form Code

```
<h3>Required items <span class="red">*</span></h3>
<form action="form-handler.php" method="post">
```

The first line informs the user that some items in the form with a red asterisk are essential.

The second line starts with the **<form** tag that begins the form section. The attributes **action** and **method** will post the user's data entries to a page named *form-handler.php*; That page sends an email to the owner of the website.

```
<label class="label" for="username"><strong>Your Name</strong><span class="red">*</span>
<input id="username" name="username" size="30"></label>
<br><br>
```

A form field with room for a single line entry is called a *text field*.

Each *text field* on the form requires two lines of code, a *label* and an *input*.

The label in this case is **Your Name**. The size of the text field is 30, which means that the *text field* can accept a data entry up to 30 characters long.

```
<label class="label" for="useremail"><strong>Your Email Address</strong><span
class="red">*</span>
<input id="useremail" name="useremail" size="30"></label>
<br><br>
```

The label in this case is **Your Email Address**. The size of the field is 30, which means that *text field* can accept a data entry up to 30 characters long.

```
<label class="label" for="userphone"><strong>Telephone Number </strong>
<span class="red">*</span>
<input id="userphone" name="userphone" size="30"></label>
<br>
```

The label in this case is **Telephone Number**. The size of the field is 30, which means that field is 30 characters long.

```
<div>
<h3>Please email me the following brochures</h3>
</div>
```

This **<h3>** subheading describes what the user is expected to do with the check boxes.

```
<div class="chk1">
<input id="chkbox1" name="bulbs" value="Yes" type="checkbox">
<label for="chkbox1"><strong>Bulbs</strong></label>
</div><br>
```

The first line defines a class **chk1** for the check box so that it can be styled and positioned. The form element is specified as **type="checkbox"**. The code for a check box is similar to the code for a text field except that the input and label lines are reversed; in this case the label is **Bulbs**. The value is **Yes**; this means that if the user ticks the box labeled “Bulbs” the email that is sent to the user will contain an instruction for the owner to send a bulb brochure. This bit of magic is the task of the form-handler page that will be described in Chapter 23.

```
<div class="chk1">
<input id="chkbox2" name="seeds" value="Yes" type="checkbox">
<label for="chkbox2"><strong>Seeds</strong></label>
</div><br>
```

This is the same as the “bulb” check box except that the *id* is *checkbox2* and the label is for **Seeds**.

```
<div class="chk1">
<input id="chkbox3" name="potting" value="Yes" type="checkbox">
<label for="chkbox3"><strong>Plants ready for potting on</strong></label>
</div>
```

This is the same as the “bulb” check box except that the *id* is *checkbox3* and the label is **Plants ready for potting on**.

```
<div>
<h3>How can we respond to your enquiry?</h3>
</div>
<div id="rad">
<input id="email" type="radio" name="contact" value="email" checked>
<label for="contact"><strong>Email</strong></label>
</div>
```

The second line in the above code poses a choice; the user is asked how the website owner should respond to the user’s message. The fourth line defines an identity for the radio button so that it can be styled and positioned. The code for a *radio* button is similar to the code for a check box. In this case the label is **Email**. The value is **email**, which means that if the user selects the radio button labeled “Email,” the email that is sent to the website owner will contain an instruction for the owner to contact the sender by email. The attribute **checked** instructs browsers to indicate that the email button is pre-selected, a black dot is placed inside the radio button (see Figure 18-1).

```
<input id="phone" type="radio" name="contact" value="phone">
<label for="contact"><strong>Phone</strong></label>
</div>
```

In this case the label is **Phone**. The value is *phone*, which means that if the user selects the radio button labeled “**Phone**,” the email that is sent to the website owner will contain an instruction for the owner to use the phone to reply to the user’s message.

```
<div id="message">
<label for="message"><h3><strong>Please type your enquiry</strong></h3></label>
<textarea id="message-field" name="message" rows="12" cols="40"></textarea>
</div>
```

An identified div **<div id="message">** is provided so that the position of the *textarea* can be styled.

A *text area* differs from a *text field* because it allows the user to insert several rows of data. The *textarea* in this case is set with 12 rows; each row is 40 characters long. Instead of an *input* line we use a *textarea* line in the third line of the code. The *textarea* is given an identity **id="message-field"** so that the message entered by the user can be styled; in this case the text has been set as *aligned left* by the internal style.

```
<div id="submit">
<input id="submit" value="Submit" title="Submit" type="submit">
</div>
</form>
```

The submit button when clicked will pass the user's data entries to the *form-handler* page and the information will be sent to the website owner as an email. It will not work at the moment because we do not have a *form-handler* page. The *form-handler* page is described later in Chapter 23.

■ **Tip** When designing your own feedback form, if the labels appear above the fields instead of alongside them, you need to reduce the number of characters in the labels, or you can increase the width of the form in the CSS style sheet. You would amend the following line to change the width: **form {width:500px; margin:auto; text-align:center;}**

Summary

In this chapter you learned how to design a feedback form and style it with an internal style. You learned the names of the form elements such as *text field*, *check box*, *radio button*, *text area*, and the *submit button*. The code for the various form elements was explained.

In the next chapter you will learn how to maximize the possibility of your website being found by search engines like Bing, Yahoo, and Google.

CHAPTER 19



Search Engine Optimization

When you eventually launch a website on the World Wide Web, you hope it will be found as soon as possible by search engines such as Bing, Yahoo! and Google. This chapter contains the following sections that will help you to maximize the probability that your site will be found:

- How search engines work
- What search engines look for
- Choosing keywords and phrases
- Keywords in headings
- Keywords/phrases must be present in the body of the page
- Restrictions on excessive repetition
- Things you should never do
- A web page containing no search engine optimization
- Let people know that your website exists
- Beware of false promises
- Optimize the spring garden home page for search engines
- Summary

How Search Engines Work

Search engines use a robot named a *web crawler* (sometimes called a *spider*, but that name is usually associated with spam spiders). The crawler trawls through every website, looking for keywords and key phrases. The search engines can then index the website's keywords so that users will be able to find website pages containing those keywords. The crawlers read through the source code on the home page first, and then they look at the source code in other pages via the links on the home page. You can improve the chances of appearing in the first few pages of search results, but an early appearance cannot always be guaranteed (however, you could pay for a favorable position; but that is beyond the scope of this book).

This chapter shows you how you might improve your rating and what to avoid. You will be given information to help you understand how search engines choose web pages that correspond with the user's search words (keywords).

Search Engines (SEs) take between three and five months to index a new optimized website. This is because the robots used by search engines have to crawl through almost 50 million new ones each year. At the same time they crawl through 240 million existing websites to update their search results. About 40 million become inactive annually and may be removed from an SE's index.

Although SEs may index a new domain within three to five months, this does not mean it will then have a good ranking in the search results. Sometimes a further five or six months may pass before your website moves up nearer to the first page of results. Some affluent companies will purchase a well-established domain name that is already high ranking rather than start from scratch with a new domain. When producing a new website, the less affluent should upload some basic content containing keywords immediately, rather than waiting until the website is complete. This lets the search engines index it and start the “probationary” period while the rest of the website is being developed.

What Search Engines Look For

Search engines look for the following elements; they are listed in order of importance:

- Keywords and key phrases, especially in the <title> and <body> sections of a page
- Keywords in the headings <h1> and <h2>
- Keywords in the meta description
- Well-designed internal links to and from all the pages in the website
- External links in related websites that point to your site

Choosing Keywords and Phrases

Users searching for relevant websites use keywords and key phrases. For instance, if a user has a problem with a computer, he will probably enter the key phrase “computer problem.” Imagine you are the person searching for the information offered on your own website; what would you type into the search field? Don't just rely on your own judgment; ask other people what they would type in. Choosing keywords and phrases is not as easy as that, but it starts you thinking. Sensible people who know the address (URL) of the website would enter the URL into the proper place: that is, the browser's address field. When you have chosen some keywords and phrases for your website, try entering them into the search field of one or two SEs and note the number of results found.

The resulting figures do not indicate the number of websites containing those keywords/phrases; it tells you the number of web pages that contain the keywords/phrases. Many results will be duplicates or triplicates. Some results may not be very relevant; this is particularly so with Google at the moment because Google places a greater emphasis on external links, paid-for links, and Google-related organizations.

The following items are keyword variations that should be considered and added to your list.

- *Singulars and plurals*: Lists and pages should include both forms of an important keyword.
- *Hyphens*: If hyphens are likely to be inserted by searchers, add the hyphenated keyword.
- *Misspellings*: If a particular keyword is commonly misspelled, add the misspelled keyword.

- *Images and links:* Add keywords in the “titles” for links. Add keywords in “alts” and “titles” for images. For instance, a picture showing a packet of seeds might have the following HTML code:

```

```

■ **Tip** The following useful websites will help you select keywords:

<http://googlekeywordtool.com> and <https://adwords.google.com/select/KeywordToolExternal>.

The Google Webmaster Tools are a must-see for learning more about search engine optimization.

Popularity of a Product or Service and Its Keywords

If your product or service is not unique, you will be competing against millions of other web pages for a good position in the list of search results. However, if you are the only company producing anti-thrombosis socks for elephants, you will be top of the first page of search results. Therefore, if a website owner concentrates on keywords relating to some unique aspect of the website’s products or services, this will result in a better search engine ranking. For instance, millions of antique dealers compete for a place in the search engine results. If the dealer concentrated on a very special type of antique, this would narrow the search field considerably and improve the website’s ranking. If your product or service is not unique, you may be able to concentrate on a particular geographic area; for instance a carpet supplier or carpet cleaner could use the local counties or towns as keywords in conjunction with the keywords *carpets* and *cleaning*.

The Importance of Keywords in the Title Tag

In the <head></head> section the <title></title> is a vitally important tag and must appear as close as possible to the top of the <head> tag for maximum effect. Your *main* keyword or key phrase should also appear in the wording of the <title> tag in HTML pages. Make sure that the keywords are relevant to that page. The title will be displayed on the browser’s tabs.

The Meta Tag/Keywords Controversy

The keywords/phrases in the <head></head> section are called *meta* tags. Put your *main* keyword or *main* key phrase at the beginning of every list of keywords in the *meta* tags in HTML pages. Putting your company or website name in the title or in meta tags is not very productive unless your company is reasonably well known. If you are Tesco or Walmart, then you must put the company name in the <title> and in the *description* meta tags because users do search using well-known names. Provide a good and relevant *description* meta tag containing key phrases or keywords because this is always displayed on the results listing.

Google states that it indexes the content of the *title* tag and the *description* meta tag; but Google ignores the meta tag for *keywords*. However, I always put keywords in the *keywords* meta tag because Bing and Yahoo! say that they take note of them, and also there is a remote possibility that Google might start looking at the *keywords* meta tag again sometime in the future. Despite the controversy surrounding the *keywords* meta tag, you should play safe and still use the *keywords* meta tag for your keywords.

Keywords in Headings

Try to use keywords in the domain name, for instance: www.devon-electrician.com, this is not always practical but worth bearing in mind.

Body text headings such as `<h1>` `</h1>`, `<h2>` `</h2>`, are ideal locations for keywords and key phrases. Search engines give a high rating to the content of headings. They also give a high rating to bold text; therefore bold headings are very important, so don't waste them by using non-keywords such as *Welcome*. Nobody searches for *welcome* or *welcome to*. The banner heading at the top of all the pages of a website is important if it incorporates a keyword. Therefore try to incorporate your product or service within the banner heading, although this is not always possible.

■ **Caution** Because search engines cannot read text that is part of an image, ensure that you use real HTML text to overlay an image as in this example:

```
<header>

<h1>The Rainbow Gallery</h1>

</header>
```

The underlying image is inserted by a line of code in the CSS file as follows:

```
Header {margin-top:0; height:181px; background-image:url('images/rainbow-banner.jpg');}
```

Keywords/Phrases Must Be Present in the Body of the Page

The keywords/phrases in the body section of your HTML file are very important. The keywords/phrases should be present between the `<body>` tags on every page of the website. The more keywords you have in the body of each page the better.

External Links

A link to your website from another website is valuable provided the other website is relevant to your product. For instance a website selling potting compost would be a suitable site for a link to the spring garden website.

Google and Bing place a great emphasis on a website's popularity; they assume that if many websites have links pointing to a particular website, they rate that website highly and it will be included in the first pages of results. *Popularity has nothing to do with the number of hits received by a website*. Although Bing, and especially Google, determine popularity mainly by detecting the number of external links pointing to your website, Yahoo! detects the number of keywords and the number of internally linked pages within your website, giving results that are much more relevant to the keyword typed into the search field.

I have web pages that regularly appear on the first page of Yahoo! because the web pages are well optimized. However, using the same search words, Google and Bing do not rate them so highly simply because fewer external links point to them. Links on popular websites that point to your site have a greater effect than links from a less popular website. External links have a strong effect, but unfortunately they are not directly under your control. Fortunately, all SEs also rely on well-organized internal links. These will be discussed first because they are directly under your control.

■ **Tip** For maximum optimization create many pages; the more pages the better. Sticking to the rule “One topic per page” will be helpful here. More than one topic on a page will confuse the search engines. Make most pages short; SEs will only scan a certain amount on each page. Ensure that they contain keywords and key phrases relevant to those pages. Pages produced by untrained amateurs using CMS programs can be excessively long and contain many topics; this, coupled with the use of JavaScript menus, means that the CMS website will not be optimized for search engines.

Well-Designed Internal Links

Insert the most important keyword/ phrase into the file names of the web pages. A four-page spring garden website might have the following file names:

- *index.html* (this is the home page and so the page name cannot be a keyword)
- *spring-bulbs.html*
- *spring-seeds.html*
- *spring-plants.html*

The menu buttons will automatically contain the keywords three times, for instance:

```
<ul class="menu">
  <li><a href="spring-bulbs.html" title="Spring garden bulbs">Spring Bulbs</a></li>
  <li><a href="spring-seeds.html" title="Spring garden seeds">Spring Seeds</a></li>
  <li><a href="spring-plants.html" title="Spring garden plants">Spring ↵
    Plants</a></li>
  <li><a href="index.html" title="Return to home page">Home page</a></li>
</ul>
```

■ **Tip** In your own websites, make sure that every page has a link back to the home page. Note also that the footer is an ideal location for internal links and keywords *provided the pages are short*.

Website designers need to know how search engines behave so that you can ensure that the website's keywords and links are optimized. The next section lists what the search engines look for and the restrictions they apply.

Restrictions on Excessive Repetition

(a) Do not repeat a phrase (or a plural of it) in the *title* tags or the *meta* tags; for instance, don't do the following:

```
<title>Garden supplies, garden supplies, garden supplies, garden supplies, garden ↵
supplies, garden supplies, garden supplies </title>
<meta name="description" content=" garden supplies, garden supplies, garden supplies, ↵
garden supplies, garden supplies, garden supplies, garden supplies ">
```

```
<meta name="keywords" content=" garden supplies, garden supplies, garden supplies, ↵
garden supplies, garden supplies, garden supplies, garden supplies">
```

(b) You can put the same phrase/word twice in the *title* tag and also use different phrases containing the keywords in the *title* tag. SEs will not read more than 60 characters including spaces in the title.

Here is a suitable title for our Spring Garden website:

```
<title>Spring garden supplies, garden supplies, garden supplies online, plants and garden ↵
supplies, bulb and seed suppliers.</title>
```

(c) You can put the same word up to four times in different phrases in the *<meta name="description">* tag. Search engines will read up to 250 characters (including spaces).

```
<meta name="description" content=" Spring garden supplies, spring bulbs, spring plants and ↵
garden supplies, bulb and seed suppliers, garden supplies, garden supplies online.">
```

(d) You can put the same word four times in different phrases in the *keywords meta* tag. Search engines will read 12 words maximum in the *keywords meta* tag.

```
<meta name="keywords" content=" Spring garden supplies, garden supplies, garden supplies ↵
online, plants and garden supplies, bulb and seed suppliers">
```

There is no other popular synonym for garden; therefore you have a problem finding enough alternatives for the word “garden” because of the restriction on repetition. If the website sells bicycles, a number of synonyms could be used in the keywords *meta* tag without undue repetition: for instance, bicycle, bike, pushbike, and cycle.

(e) Although keywords/phrases in the *title* are the most important, the *keywords in the body section are more highly rated* than those in the *meta* tags.

You can put many keywords in the *<body>* section of a page and you can repeat them often. But don’t overdo it. Most experts say the body keywords should not constitute more than 10 percent to 15 percent of the total words on a page.

SEs restrict the number of words they will scan on each page. If a long page has keywords near the bottom of the page they have little value because SEs stop scanning the page before it reaches the end of a long page.

Things You Should Never Do

The following list of “things you should never do” was taken from the advice pages of the three major search engines.

- Do not put links to other websites on your home page; this would be detrimental to your own ranking.
- Do not use JavaScript menus and links; they prevent web crawlers from accessing your other pages.
- Do not use tricks such as invisible writing: for example, white keywords on a white background. Web crawlers can even detect and penalize closely related colors such as a very pale gray text on a white background.
- Do not use images for keywords/key phrases; these are invisible to web crawlers.

- Do not use *cookie cutters*; that is, pages that are identical or practically identical that have been added to boost the number of linked pages. Search engines may penalize the website.
- Do not use link farms. These are websites that ask you to add a link to your site in return for a link on their site. This is acceptable if the other website has very few links to other sites but link farms have a page or pages crammed with external links. Search engines refuse to acknowledge these links.

You can submit new websites to the search engines; the URLs for doing this are listed in the next tip. Only submit once, or otherwise the search engines will assume you are spamming. Some SEO experts prefer not to submit; they feel that it is better to wait for the search engines to find the website. The best approach can be difficult to determine.

■ **Tip** Submit website URLs to the three main search engines at the following addresses:

Bing: <https://ssl.bing.com/webmaster/SubmitSitePage.aspx>

Yahoo: <http://www.search.yahoo.com/info/submit.html>

Google: <http://www.google.co.uk/addurl/> and http://www.google.com/submit_content.html

Do not use mass-submission programs. In my experience, some of these programs sell the site's email address to spammers; don't risk it, do the job yourself.

A Web Page Containing No Search Engine Optimization

Figure 19-1 shows a web page that has absolutely no search engine optimization.



Figure 19-1. A web page that appears to contain the key phrase “teddy bears”

The obvious key phrase for this website is *teddy bears* or *Colyton teddy bears*. At first glance, you might think that the key phrase appears five times on this page. However, search engines scan the source code looking for keywords and phrases. Listing 19-1 shows the source code for the page; pretend you are a search engine and see if you can spot any suitable key words or phrases. The SEO faults are shown in bold type.

Listing 19-1. Demonstrating a Web Page with no Search Engine Optimization

```
<!doctype html>
<head>
<title>Welcome to our website</title>
<meta charset=utf-8>
    <script type="text/javascript" src="coolmenu.js"></script>
    <script type="text/javascript" src="menu_items2.js"></script>
    <link rel="stylesheet" type="text/css" href="menu_styles.css"/>
<style type="text/css">
#container {margin:0 auto 0 auto; width:790px;}
</style>
</head>
<body>
    <div id="container">
        <header>

        </header>
        <div>
            <script type="text/javascript">
            var m1 = new COOLjsMenu("menu1", MENU_ITEMS)
            </script>
        </div>
        <p>&nbsp;</p>
        <div>
            
            
            
        </div>
    </div>
</body>
</html>
```

Explanation of the SEO Faults in Listing 19-1

- The key phrases *teddy bears* and *Colyton teddy bears* do not appear anywhere in the source code, nor do the words *teddy* or *bears*. The key phrases are all images, so the key phrases cannot be detected by web crawlers. The designer should have used background colors for the colored panels and then overlaid real HTML text on the panels so that the SE could read the text.
- The page title is a waste of an opportunity; it contains no keywords and nobody searches using the word *welcome*. The head section contains no *meta* information.
- There was a chunk of JavaScript for a drop-down menu that links the other pages on the website. Search engines do not read JavaScript. Subsequent pages will never be seen by the search engines.

- The header image is called *banner.jpg* and no *alts* or *titles* were used. Three opportunities for inserting keywords were missed. It could have read:

```

```

- The same fault can be seen in each of the three panels; for example, *black-panel.jpg* could have been:

```

```

The conclusion: A user typing *teddy bears*, *teddy*, or *bears* into the search field would never find this web page or any of the website's subsequent pages in the search engine results.

■ **Note** Search Engines cannot read computer screens, they can only read the source code.

Let People Know That Your Website Exists

To maximize the number of visitors to your site, supplement the search engine results by other forms of advertising. A commercial site may be ranked low, yet it gets many more visits than highly ranked competitors; this is probably because the low ranked site is advertised widely in magazines, trade journals, trade directories, and brochures. Regular, small advertisements in the right journals and brochures get good results. On every brochure, flyer, wrapper, and all other pieces of printed matter, make sure your website address is prominently displayed. If the website owner has a retail store, ensure that the website address is clearly displayed on the shop front and on delivery vans.

Satisfied customers should be encouraged to tell others about your product. These people may then look for your product on a search engine.

Beware of False Promises

Warn your clients that they will be pestered by emails from unscrupulous people trying to persuade your client to pay for the website to be optimized. If an owner is duped by this, she will pay a lot of money and the spammer will do absolutely nothing. *In fact, there is nothing the spammer can do.* Some of these dreadful individuals even claim to be working for Google, Yahoo!, or Bing, or they claim to have contacts within those organizations. One particularly nasty specimen used the telephone to contact victims. He said he worked for Google and that if the victim refused to use his "service," he would remove the website from Google's index. Genuine SE optimizers do exist, but why pay for a job you can do yourself with the help of this chapter?

Optimize the Spring Garden Home Page for Search Engines

This project demonstrates where to locate keywords in a web page. To save time and space we will work only on the home page. The resulting optimized page is shown in Figure 19-2.



Figure 19-2. Keywords have been added to the horizontal menu, the main text, and the footer

To create the optimized spring garden page, please follow these steps:

1. In the *web-tutorial* folder, create a new folder named *Chapter-19*.
2. Open the *templates* folder and then open the folder *template-3d-h-v*.
3. Use Ctrl+A then Ctrl+C to copy all the files into the computer's memory.
4. Open the new folder *Chapter-19*.
5. Paste the items from the computer's memory (use Ctrl+V) into the folder *Chapter-19*.
6. Open the file *style-3d-gdn.css* and delete the code shown bold and crossed through in the snippet below:


```
#rightcol{float:right; width:150px; height:190px;}
```
7. Open the file *index.html* in the Code/Source view of your HTML text editor and add the items shown in bold type in Listing 19-2.

Listing 19-2. Optimizing the home page index.html for search engines.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Spring garden bulbs, seeds, plants. Home page</title>
    <meta charset=utf-8>
```

```

<meta name="description" content="Spring bulbs, seeds and plants">
<meta name="keywords" content=" Spring bulbs, seeds, plants, bulbs, ↵
  spring garden seeds, spring garden plants">
<link href="style-3d-gdn.css" rel="stylesheet" type="text/css">
  <style type="text/css">
    .small {font-size:75%; color:black;}
  </style>
<!--[if lte IE 8]>
  <script src="html5.js">
</script>
<![endif]-->
</head>
<body>
<div id="wrapper">
  <header>
    <h1>The Spring Garden</h1>
  </header>
<nav>
  <ul>
    <li class="btn"><a href="spring-bulbs.html" title="Spring garden bulbs">Spring ↵
      Bulbs</a></li>
    <li class="btn"><a href="spring-seeds.html" title="Spring garden seeds">Spring ↵
      Seeds</a></li>
    <li class="btn"><a href="spring-plants.html" title="Spring garden plants">Spring ↵
      Plants</a></li>
    <li class="btn"><a href="index.html" title="Return to Home page">Home Page</a></li>
  </ul>
</nav>
<!--<br class="clear">-->
<div id="content">
<div id="leftcol">
  <!--<p>This is the far left column</p>-->
  <nav id="vertical">
    <ul>
      <li class="vbtn"><a href="#" title="About Us">About Us</a></li>
<!-- Change the Contact.html link to # -->
      <li class="vbtn"><a href="#" title="Contact Us">Contact Us</a></li>
      <li class="vbtn"><a href="#" title="Locate Us">Locate Us</a></li>
      <li class="vbtn"><a href="#" title="Terms and Conditions">Terms</a></li>
    </ul>
  </nav>
</div>
<div id="rightcol">
  <p>This is the far right column</p>
</div>
<div id="midcol">
  <h2>Spring Garden Bulbs, Seeds and Plants</h2>
<div id="midcol-left">
  <p class="left">Spring, the sweet spring, is the year's pleasant king; ↵
    Then blooms each thing...<br>
    <span class="quote">From "Spring" by Thomas Nash</span><br>

```


[illegible]

8. Save the page and view it in a modern browser.

■ **Important** To save space and concentrate on SEO, we have not created the three pages *spring-bulbs.html*, *spring-seeds.html*, and *spring-plants.html*; any attempt to click the horizontal menu buttons or the footer links will result in a “Page not available” error message.

Explanation of the Code

The keywords/phrases chosen for optimizing the website are *Spring*, *garden*, *bulbs*, *seeds*, and *plants*.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Spring garden bulbs, seeds, plants. Home page</title>
    <meta charset=utf-8>
    <meta name="description" content="Spring bulbs, seeds and plants">
    <meta name="keywords" content=" Spring bulbs, seeds, plants, bulbs, spring garden ←
      seeds, spring garden plants">
    <link href="style-3d-gdn.css" rel="stylesheet" type="text/css">
```

```
<style type="text/css">
  .small {font-size:75%; color:black;}
</style>
```

The keywords/phrases are inserted into the title, and also in the new meta names. An internal style is added to format the footer so that it has a smaller font (using a class `.small {font-size:75%; color:black;}`).

```
<nav>
  <ul>
    <li class="btn"><a href="spring-bulbs.html" title="Spring garden bulbs">Spring ←
      Bulbs</a></li>
    <li class="btn"><a href="spring-seeds.html" title="Spring garden seeds">Spring ←
      Seeds</a></li>
    <li class="btn"><a href="spring-plants.html" title="Spring garden plants">Spring ←
      Plants</a></li>
    <li class="btn"><a href="index.html" title="Return to Home page">Home Page</a></li>
  </ul>
</nav>
```

The horizontal menu now incorporates the keywords/phrases. The file names of three web pages have been changed to include keywords/phrases

```
<div id="midcol">
  <h2>Spring Garden Bulbs, Seeds and Plants</h2>
  <div id="midcol-left">
    <p class="left">Spring, the sweet spring, is the year's pleasant king; Then ←
      blooms each thing..<br>
    <span class="quote">From "Spring" by Thomas Nash</span><br>
    <p class="left">Nothing heralds the arrival of spring better than a garden full ←
      of spring flowers. This website will help you to plan your spring flower ←
      display and choose the best spring bulbs, seeds and plants.</p>
    <p>View and order spring bulbs, seeds and plants using this website, or visit ←
      the Spring Garden Center at 10 The Street, Townsville.</p>
  </div>
```

Note the keywords/phrases in bold that are now added to the content

```
<!--<p>This is the far left column</p>-->
  <nav id="vertical">
    <ul>
      <li class="vbtn"><a href="#" title="About Us">About Us</a></li>
      <li class="vbtn"><a href="contact.html" title="Contact Us">Contact Us</a></li>
      <li class="vbtn"><a href="#" title="Locate Us">Locate Us</a></li>
      <li class="vbtn"><a href="#" title="Terms and Conditions">Terms</a></li>
    </ul>
  </nav>
</div>
```

Note, to save space we have not optimized the *contact.html* file. Therefore, if you click the vertical menu button labeled *Contact Us*, you will see none of the new optimized buttons and footer links.

[illegible]

The line `<br class="clear">` is now deleted.

The footer is expanded to include links to the renamed pages. The footer now has links that contain keywords/phrases to make maximum use of the links for optimizing the page. The code ` ` provides three spaces between each link; the entity for a space is ` `;

Summary

To ensure that people can find your website, it must be advertised. This chapter mentioned the two main methods of advertising your website: (i) search engine optimization and (ii) printed matter. The chapter began with an overview of search engines and how they work. You learned how to choose keywords and phrases. There was also a section describing what you must never do. Advice was given on well-designed internal and external links. You were given a warning about the search engine scam that afflicts website owners. You examined and analyzed a web page that was not optimized. You then practiced optimizing the home page of the spring garden website.

In the next chapter, you will learn more about positioning elements on a web page, in particular the use of absolute and relative positioning for logos and decorative items.

CHAPTER 20



Positioning Elements on a Web Page

Elements can be positioned accurately on a web page by using CSS styling. The two types of positioning are *Absolute* positioning and *Relative* positioning. You used absolute positioning in many of the previous projects; this enabled you to locate an element (such as the header text) at a fixed position relative to the top and left edges of a browser's viewport. Relative positioning moves an element from where it would normally appear on the page. Several other positioning techniques are also described in this chapter.

This chapter contains the following sections:

- Absolute positioning
- Change the vertical position of the header text
- Relatively positioning an image
- An experiment using relative positioning
- Positioning images across the wrapper boundary
- Positioning images next to text
- The <clear> property applied to floated elements
- Positioning by using the margin property
- Summary

Absolute Positioning

The next project will be an absolute positioning experiment using the “The Spring Garden” home page header text as shown in [Figure 20-1](#).



Figure 20-1. The vertical position of the header text is positioned absolutely from the top edge of the browser's viewport (to save space, only the top portion of the page is shown)

The CSS style for the header text in Figure 20-1 is as follows:

```
h1 {position:absolute; top:-15px; margin-left:30px; font-size:450%; color:white;
    font-weight:bold;}
```

■ **Note** The horizontal position of the text is set by the 30 pixels left margin; the text begins 30 pixels from the left edge of the header. The horizontal position was not set by means of absolute positioning because when the viewport shrinks on smaller screens, the text would move to the right, away from the left edge of the header. The format for absolute positioning is as follows:

```
h1 {position:absolute; top:nnpx; left:nnpx;},
```

in our example *left* position is replaced by a *left margin*.

Of course, you would substitute numbers for the items shown as *nn*.

To explore the effect of vertical absolute positioning, please follow these steps:

1. In the *web-tutorial* folder, create a new folder named *Chapter-20*.
2. Open the *Chapter-19* folder.
3. Use Ctrl+A then Ctrl+C to copy all the files into the computer's memory.
4. Open the new folder *Chapter-20*.
5. Paste the items from the computer's memory into the folder *Chapter-20* (use Ctrl+V).
6. Download the *Chapter-20* folder zip file from this book's page on the *apress.com* website and insert it into your *Chapter-20* folder.
7. Unzip the downloaded *Chapter-20* folder and copy the two images *rosette -128.png* and *crocus -3.gif* (Ctrl+C), then close the downloaded folder.
8. In your own *Chapter-20* folder, paste the two images into your *images* folder.
9. Open the file *index.html* in a modern browser.
10. Minimize the browser window so that it sits on the task bar ready for the next experiment.

Change the Vertical Position of the Header Text

11. Open the file *style-3d-gdn.css* in the Code/Source view of your HTML text editor and amend the absolute vertical position as shown in bold type in the following code snippet:

```
/*h1 {position:absolute; top:-15px; margin-left:30px; font-size:450%; color:white; ↵
    font-weight:bold;}*/
h1 {position:absolute; top:45px; margin-left:30px; font-size:450%; color:white; ↵
    font-weight:bold;}
```

Place the cursor at the end of the line that styles the *h1* tag and press Enter to create an empty line.

Copy (Ctrl+C) the line that styles the *h1* tag. Paste the copied line into the empty line (Ctrl+V).

Comment-out the first line that styles the *h1* tag by using */** and **/*. The line that is commented-out exists so that you can return to the original style when you have finished experimenting.

Amend the second *h1* line to change the absolute position as shown in bold type, that is, change the negative position *-15px* to *45px*. The original negative position moved the text upwards from its natural position by 15 pixels. That new position pushes the text down to a position 45 pixels below the top edge of the browser window.

12. Save the file.
13. Click the minimized *index.html* file on the task bar to maximize it, and then click the browser's refresh button to see the header text move downwards to a new position.
14. We have seen the result of the repositioning, but we must admit that the new position of the header text is not as good as the original position; therefore please change it back to the original position; the code will then be as before:

```
h1 {position:absolute; top:-15px; margin-left:30px; font-size:450%; ↵
    color:white; font-weight:bold;}
```

15. Save the file.
16. Minimize the home page once more to get ready for the next experiment.

Relatively Positioning an Image

Relative positioning is frequently used to reposition an object relative to its original location on the page. For instance, a logo will normally appear in an unsatisfactory position within its containing box; we can move it to any position by using relative positioning as shown in Figure 20-2.

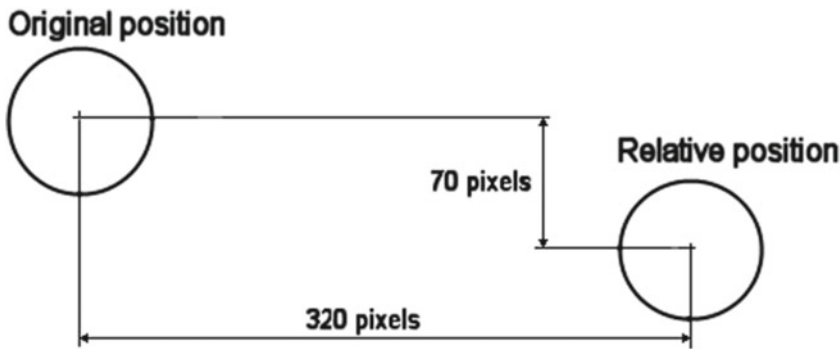


Figure 20-2. The circle is repositioned using relative positioning

CSS relative positioning displaces the item relative to where it would normally appear. With relative positioning, the circle on the left in Figure 20-2 has been made to appear 320 pixels to the right, but the CSS code for this is not as you would expect. The CSS code actually tells you where the original circle was located; that is, 320 pixels to the left and 70 pixels upwards. The CSS code is:

```
#circle {position:relative; left:320px; top:70px;}
```

■ **Tip** To remember the code, regard the dimensions for *top* and *left* as if they are top and left margins.

An Experiment Using Relative Positioning

In this project you will learn how to relatively position a logo on the banner of a web page. Please follow these steps:

1. Use *Save As* to save a copy of *index.html* with the file name *logo.html*.
2. Open *logo.html* in the Code/Source view of your HTML editor and insert the code for the rosette logo as shown in bold type in Listing 20-1.

Listing 20-1. Add a rosette logo to the header section

```
<header>
  <h1>The Spring Garden</h1>
  
</header>
```

The source (*src*) of the rosette image is the *images* folder. The image is identified with the id “*logo*” so that the image can be styled. The *alt* and the *title* provide *tooltips* in Internet Explorer and Fire Fox respectively.

3. Now view the page in a modern browser and it should look as shown in Figure 20-3.



Figure 20-3. The rosette logo is added, but it is badly positioned. Its natural position is too high and a little to the left of the desired location

4. Minimize the page to the task bar ready for the next stage.

We will now use relative positioning to move the logo to a new position lower down and a little further to the right.

5. Anywhere in the file *style-3d-gdn.css*, add the relative positioning for the logo image as shown in bold type in the following snippet:
#logo {position:relative; top:100px; left:920px;}
6. Save the revised style sheet *style-3d-gdn.css*.
7. Click the minimized page on the taskbar and then click the refresh button to view the new logo position. It should look like Figure 20-4.



Figure 20-4. The logo image is moved to a new position

■ **Tip** You might be wondering how I chose the dimensions for positioning the logo; I might have calculated them, but I find that trial and error is much quicker.

Positioning Images Across the Wrapper Boundary

Relative positioning can be used to place an image across the boundary of the wrapper as shown in Figure 20-5. This is not always beneficial because it can spoil the focus, that is, the viewer has an additional colorful but irrelevant item to distract them from the important items. Therefore use gimmicks such as this sparingly.

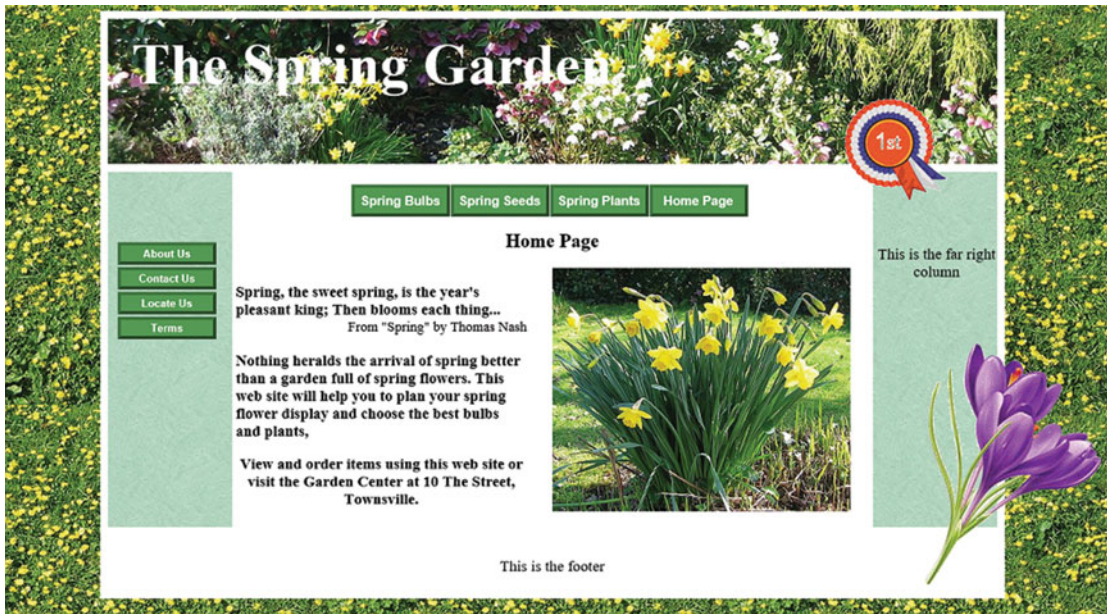


Figure 20-5. The crocus image is positioned across the wrapper boundary

Note that the crocus image distracts the user from focusing on the main elements; this exercise is provided only to demonstrate another aspect of relative positioning.

To produce the page shown in Figure 20-5, please follow these steps:

1. Open the file *logo.html* in the Code/source pane of your HTML text editor.
2. Use *Save As* to make a copy of the file and name it *crocus.html*.
3. Insert an extra internal style as shown in bold type in Listing 20-2.

Listing 20-2. Insert an additional internal style in the *crocus.html* page

```
<link href="style-3d-gdn.css" rel="stylesheet" type="text/css">
<style type="text/css">
    .small {font-size:75%; color:black;}
    #crocus {position:relative; top:60px; left:60px;}
</style>
```

4. Insert the crocus image into the *rightcol* div of the *crocus.html* page as shown in bold in Listing 20-3.

Listing 20-3. Insert the crocus image into the *rightcol* div

```
<div id="rightcol">
    <p>This is the far right column</p>
    
</div>
```

■ **Note** Images that are superimposed over background pictures must be in either the transparent *gif* or the transparent *PNG* format. The background of the image must be transparent to enable the container's background to show through. See Chapter 41 for tips on using graphics programs.

5. Save the page and view it in a modern browser.

Explanation of the Code

```
<link href="style-3d-gdn.css" rel="stylesheet" type="text/css">
  <style type="text/css">
    .small {font-size:75%; color:black;}
    #crocus {position:relative; top:60px; left:60px;}
  </style>
```

The style positions the image relative to the boundary of its div. I chose the *top* and *left* dimensions by trial and error.

```
<div id="rightcol">
  <p>This is the far right column</p>
  
</div>
```

The crocus image code is located within the boundary of the *rightcol* div, but the previous code positions it beyond the boundary and over the wrapper's white border .

■ **Caution** You cannot position *floated* elements absolutely or relatively, because *position* and *float* are mutually exclusive commands.

Positioning Images Next to Text

Images can be positioned next to text by floating the image. Items can be floated to the left or to the right. The default is `float:none`; Floated elements should always have a width (floated images already have predefined width). When an element is floated, it cannot be given a `position:absolute`; or a `position:relative`; The CSS is `float:left`; or `float:right`; or the default `float:none`;

■ **Tip** Floated elements can have positive or negative margins to assist positioning. When you have created a floated image, you can left-click it and move it up or down to position it accurately.

Figure 20-6 shows the top rosette floated left and the lower rosette floated right. I have repeated the sentences to provide a bigger block of text.

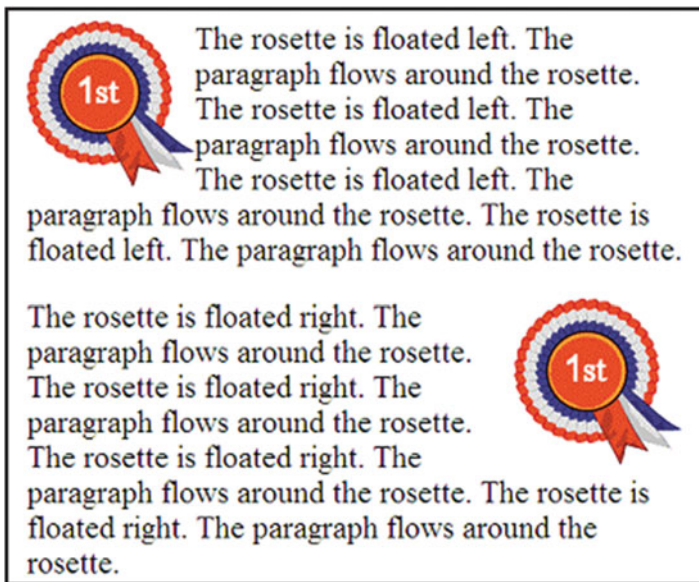


Figure 20-6. Left and right floated rosettes

Listing 20-4. Demonstrating Left and Right Float (*float-1.html*).

```
<!doctype html>
<head>
<title>float-1</title>
<meta charset=utf-8>
<style type="text/css">
  body {font-size:150%;}
  #container {width:500px; margin:auto;}
  .float-left {float:left;}
  .float-right {float:right;}
</style>
</head>
<body>
<div id="container">

<p>The rosette is floated left. The paragraph flows around the rosette. The rosette is ↵
floated left. The paragraph flows around the rosette. The rosette is floated left. The ↵
paragraph flows around the rosette. The rosette is floated left. The paragraph flows ↵
around the rosette</p>

<p>The rosette is floated right. The paragraph flows around the rosette.The rosette ↵
is floated left.The paragraph flows around the rosette.The rosette is floated left. ↵
The paragraph flows around the rosette. The rosette is floated left. ↵
The paragraph flows around the rosette</p>
</div>
</body>
</html>
```

■ **Tip** If the image is rectangular, the text will bump up against it, which is undesirable. Put a space between the side of the image and the paragraph of text by giving the image a margin on the side that abuts the text. For example, you would insert into the CSS file the following code `#container img {margin-right:5px;}` or `#container img {margin-left:5px;}`

The `<clear>` Property Applied to Floated Elements

Text flows around a floated item. Sometimes you do not want text to flow around; you would rather force the text below the floated element. This can be achieved by using the `clear` property; however, I am never able to remember what clears what, and how right and left should be applied.

The general rule is that if an element such as an image is floated left, style the text with the CSS property `clear:left`; in other words, clear the text away from the left-floated item; see Figure 20-6. Conversely, if an element is floated right and you do not want text to flow around it, then give the text the CSS property `clear:right`;

■ **Tip** The CSS command `clear` can also have the attribute both as in `clear:both`; This is extremely useful for pushing an element like a footer below the columns on a page.

The example illustrated in Figure 20-7 demonstrates the `clear:left`; and `clear:right`; properties.

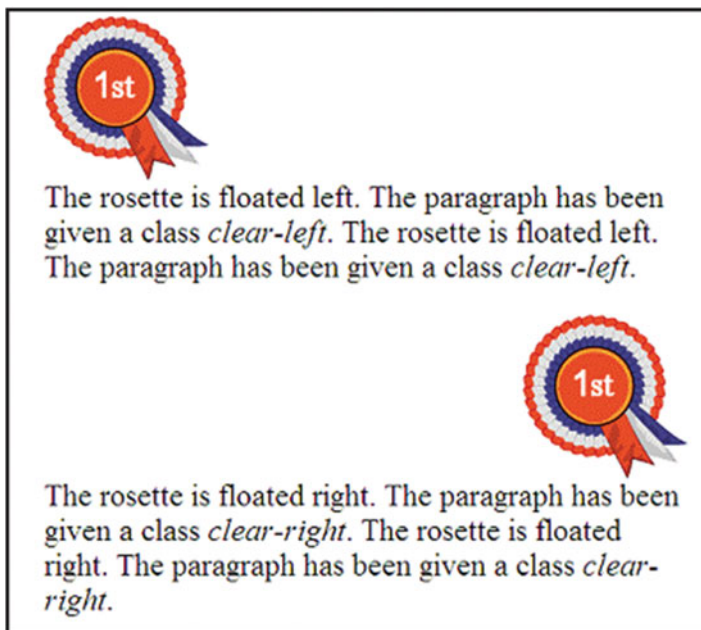


Figure 20-7. The paragraphs are styled to clear the floated images

I have repeated the sentences to provide a bigger block of text.
 The code in bold type in Listing 20-4 forces the text clear of the images.

Listing 20-4. Using the Clear Property (*float-2.html*).

```
<!doctype html>
<head>
<title>float-2</title>
<meta charset=utf-8>
<style type="text/css">
  body {font-size:150%;}
  #container {width:500px; margin:auto;}
  .clear-left {clear:left;}
  .clear-right {clear:right;}
</style>
</head>
<body>
<div id="container">

<p class="clear-left">The rosette is floated left. The paragraph has been given a ↵
class <i>clear-left</i>. The rosette is floated left. The paragraph has been given a ↵
class <i>clear-left</i>.</p>

<p class="clear-right">The rosette is floated right. The paragraph has been given a ↵
class <i>clear-right</i>. The rosette is floated right. The paragraph has been given a ↵
class <i>clear-right</i>.</p>
</div>
</body>
</html>
```

Positioning by Using the Margin Property

Elements such as images, menus headings, and blocks of text can be moved left or right by applying CSS margins. For instance, an image located within a particular <div> might have an undesirable position if no margin is applied as follows:

```
<div id="left-col">
  
</div>
```

However, the image could be pushed 15 pixels to the right by using the following CSS margin:

```
#left-col img {margin-left:15px;}
```

Alternatively it could be pulled 15 pixels further to the left by using a negative margin as follows:

```
#left-col img {margin-left:-15px;}
```

Margins can be used to position most page elements either vertically or horizontally. For example, you may want a 10 pixel gap above a wrapper's top border, in which case you would use some CSS like this:

```
#wrapper {margin-top:10px;}
```

Summary

In this chapter you learned how to use absolute and relative positioning for text and images. You also learned that images that are superimposed on backgrounds must have a transparent background and be in either the transparent *gif* or transparent PNG format. You learned how floated images can be positioned relative to a paragraph of text. Finally, you discovered that elements can be positioned by applying positive or negative margins.

In the next chapter we move into the advanced section of the book. You will learn how to create websites with many more pages and many more menu buttons, using a technique that saves time and avoids tedium. This provides a gentle introduction to a tiny piece of the PHP language. PHP is a relatively easy-to-learn interactive language that uses plain English words. Don't be alarmed at the prospect of learning something that will eliminate the tedium of amending menus, headers, and footers on larger websites.

CHAPTER 21



Save Time and Reduce Tedium

If you have worked through the projects so far, you will have experienced the tedium of modifying the menus in every page of a website. Imagine the work involved in modifying the menus in each page of a website with 20 or more pages. In this chapter you will learn a simple piece of code that will enable you to modify all the page menus in a website by amending only one file.

This chapter contains the following sections:

- The time-saving code
- A note about PHP code
- Using a server
- Download and install XAMPP for Windows
- Using the PHP *include* command
- Create the external files
- Summary

The Time-Saving Code

As the number of pages in your website grows, the task of adding the header, footer, and extra menu buttons to each new page becomes tedious. Elements that are identical on each page can be placed in single external files and included in each page by a simple PHP command. For instance, a menu can be placed in a single external file and then pulled into every web page by the following simple PHP command located in each page:

```
<?php include ('nav.html'); ?>
```

If you change the menu in the single external file *nav.html*, the change will be pulled automatically into every page of the website. Without this technique, you would have to manually amend the menu in every page in the website, then you would have to upload every page of the website to the host. With the PHP *include* file you would only have to upload the single amended menu file to the host.

Two small downsides exist: (i) Every page in the website must be given the suffix *.php*: for instance, the file *index.html* becomes *index.php*. (ii) You cannot view the result of the inclusions in a browser in the usual way. The PHP files must either be viewed by uploading them to a host or by viewing them in a server installed in your computer. The most convenient way to view the website is to install a free server like XAMPP for Windows or MAC on your computer (instructions will be given for this).

■ **Tip** The downsides are insignificant compared with the enormous advantages of using the PHP *include* command. The *include* command is not the only bit of PHP that you need to learn for working on the remaining chapters in this book. One more piece of PHP will be used in Chapter 23 to enable you to receive emails from a “Contact Us” page.

A Note About PHP Code

PHP integrates smoothly within HTML documents and allows those documents to interact with external files. PHP is a *server-side* language: that is, **it will only work when loaded into a server**. In contrast, HTML is a *client side* language that works directly in a browser without a server, but PHP offers many advantages when used within an HTML page. Because it is so versatile and reasonably easy to learn, about 82% of websites use some PHP in their pages as shown in Figure 21-1.

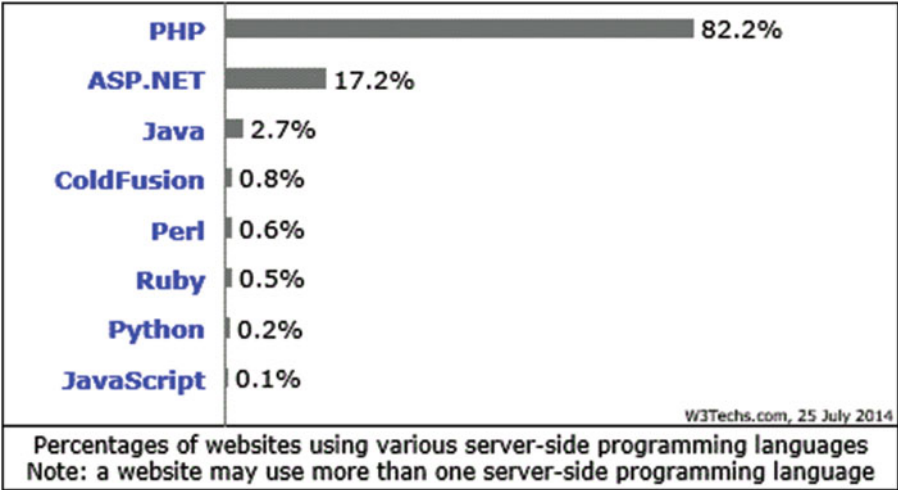


Figure 21-1. The chart of programming languages produced by www.w3techs.com

PHP code can be inserted anywhere in an HTML page. Browsers just need to know when the PHP code is present; this is declared with the tag `<?php` that signifies the start of the PHP code. The closing tag `?>` tells the browser that the PHP code has finished and that the subsequent code is HTML. In the example at the beginning of this chapter, the code between the tags is `include ('nav.html')`; this is a PHP command that instructs the browser to include and display an external file named `nav.html`. In other words the PHP code generates some HTML code on the page by pulling that code from an external HTML file. The command is closed by the important semicolon at the end of the PHP command.

You will need to check whether the firm that will host your website has the facility for presenting PHP pages to visitors. Some very basic free hosts may not have this facility, but such hosts are now extremely rare. If you have the misfortune to use one of those basic hosts, then you need to ask the host if an upgrade is available. If not, you should change host, or accept the fact that you will be unable to take advantage of this valuable time-saving feature.

Using a Server

Because PHP is a *server-side* language, you will need to access a server by one of two methods as described next.

1. Use an Existing Website Server on a Host

If you own a website and you are reading this manual to learn how to manage your site, you could download and install a File Transfer Program (FTP program). You would then be able to use your hosting company's server to test the projects in this chapter. See Chapter 40 for details about going online and using an FTP program. You would place this chapter's files in a folder, and then use your FTP program to upload the folder to the host. Say your current website is www.sams-sausages.co.uk and your new uploaded folder is named *spring-garden*. The uploaded folder will be a subfolder in your main website. To access your subfolder use this web address:

www.sams-sausages.co.uk/spring-garden/

If you intend to use this method to test your PHP files, skip the XAMPP sections and go to the section beginning "Using the PHP *include* Command."

2. Install a Server for Free on Your Computer

However, very few beginners will have a website or an FTP program to upload the files to the host; most learners and web developers prefer the convenience of using a free server installed on their own computer. My preferred home server is built into a package named XAMPP. The package is free and it contains the server called Apache; the package also includes a PHP parser (an error checker for PHP code). XAMPP contains other programs but these need not concern us at the moment. The programs are preconfigured so that the components will talk to each other. This eliminates the usual hassle involved in downloading individual components and then configuring them to work together.

■ **Important** Persons viewing your PHP website via the Internet *do not* need to have XAMPP installed on their computers. If your website is hosted, it is already on a server.

Download and Install XAMPP for Windows

MAC users can download XAMPP from: <http://www.macupdate.com/app/mac/19593/xampp>

XAMP is free and needs no configuring.

To download the package for Windows, go to:

<http://www.apachefriends.org/en/xampp-windows.html>

At the time of writing XAMPP 1.8.0 was the latest stable version. The home page varies from time to time; explore the buttons on the tool bar to load the screen shown in Figure 21-2.

XAMPP for Windows 1.8.0, 13.7.2012		
Version	Size	Content
XAMPP Windows 1.8.0		
Apache 2.4.2, MySQL 5.5.25a, PHP 5.4.4, OpenSSL 1.0.1c, phpMyAdmin 3.5.2, XAMPP Control Panel 3.0.12, Webalizer 2.23-04, Mercury Mail Transport System v4.62, FileZilla FTP Server 0.9.41, Tomcat 7.0.28 (with mod_proxy_ajp as connector), Strawberry Perl 5.16.0.1 Portable For Windows 2000, XP, Vista, 7.		
 Installer	91 MB	Installer MD5 checksum: 5de03a40fcb5865b29913affef3f115
 ZIP	166 MB	ZIP archive MD5 checksum: 90dd1e4278252520d681ff01ae00433d
 7zip	78 MB	7zip archive MD5 checksum: e45eba705f9258b543b3b19e7b7b1b0b

Figure 21-2. Installing XAMPP

Scroll right down the download page until you see the section illustrated in Figure 21-2.

Choose the *zip* version for Windows (shown ringed) in Figure 21-2. Although the *Installer* version is recommended by XAMPP, it sometimes causes an installation problem on 64-bit Windows 7 computers. The *Zip* version installed in 32-bit and 64-bit computers without any problems.

Download the file into your Downloads folder and then double-click it to unzip it into a new folder named *xampp* in the root of the hard drive. The default folder for the installation will then be C:\xampp. Do not install it in the *Programs* folder. During the installation you may experience a virus warning, but ignore it. Choose to install a desktop icon.

During the installation you may see some black screens with white text, but just keep going until the installation is completely finished. You will see a Window named XAMPP Options. Tick all the boxes except the last one labeled *Install FileZilla as service*. The installation may demand a restart. You will be asked if you want to load the XAMPP control panel; say no for the moment.

If XAMPP is running, you will see an icon in the Notification area like Figure 21-3:



Figure 21-3. The XAMPP icon

■ **Caution** The XAMPP icon is the same color and shape as the Java update icon.

If a desktop icon was not created during the installation, go to C:\xampp folder, then create a Desktop shortcut for the *xampp.exe* file. See Figure 21-4 below:

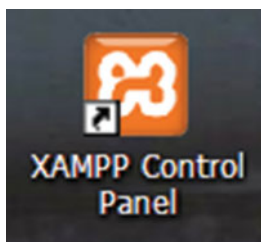


Figure 21-4. The XAMPP desktop icon

Starting XAMPP

From now onwards, to test your pages in XAMPP's Apache server, double-click the desktop icon, then minimize the XAMPP's control panel, and check that Apache has started. If it has not started, click its start button. The XAMPP control panel is shown in Figure 21-5.

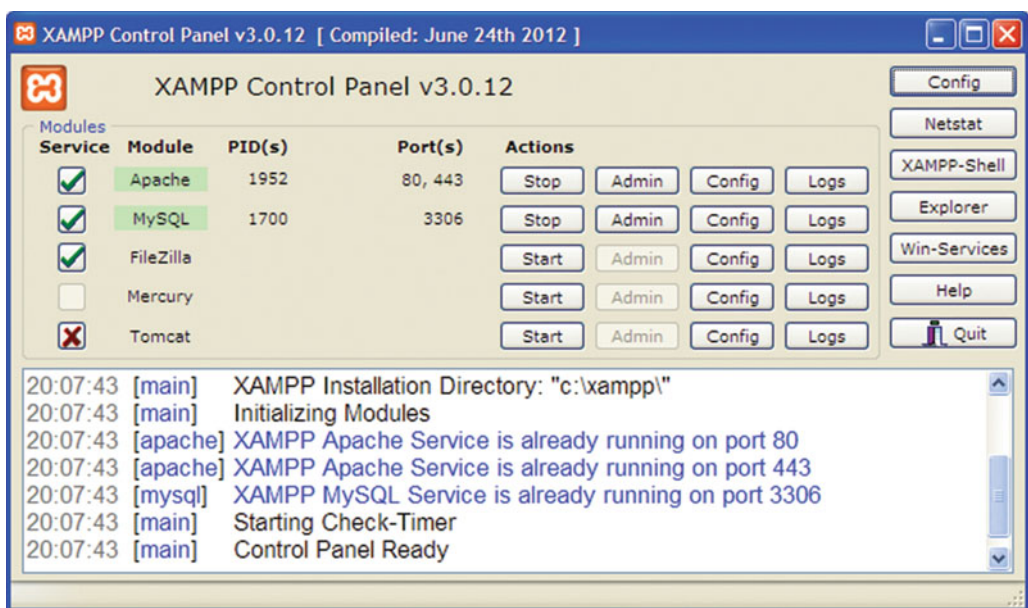


Figure 21-5. The XAMPP control panel

Apache is the server. If the Apache button says *Start*, that means it is not running and you need to click the button to start Apache.

■ **Tip** Always minimize the control panel to have a clear desktop for working on your projects.

If XAMPP/Apache fails to start, it may be because you are using Skype. XAMPP/Apache and Skype both need to run on port 80. To solve this conflict, on the Skype toolbar click Tools ► Options ► Advance ► Connections. Now uncheck the box labeled “Use port 80 and 443 as alternatives for incoming connections,” then click Save. You may need to restart Skype.

If you are not using Skype and XAMPP /Apache still refuses to start it will be because some other piece of hardware on your computer is using port 80. A search on the Internet will provide an answer to conflicts of this nature.

Closing XAMPP

Close XAMPP when you have finished testing your database/PHP files; this will free up memory for tasks other than database development. To close down, click the minimized XAMPP control panel on the task bar and then click the *Quit* button on the control panel as shown in Figure 21-6. Alternatively, you can right-click the icon in the notification area and then click *Quit*.

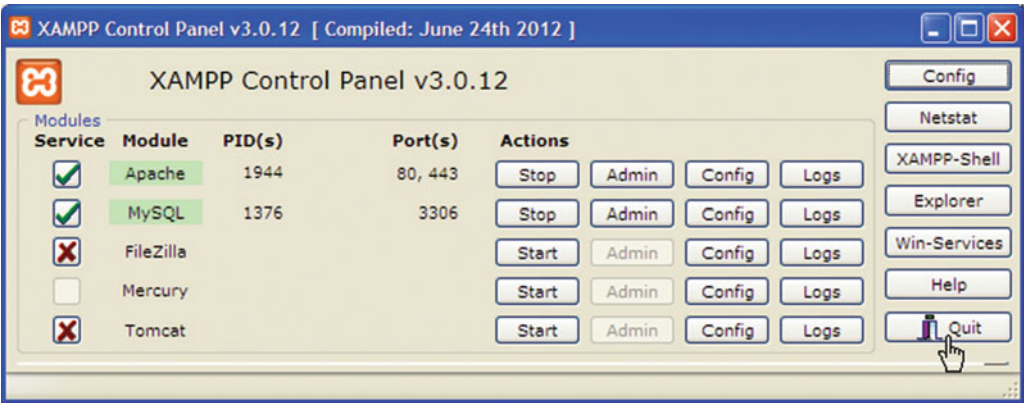


Figure 21-6. Closing down the XAMPP program

The XAMPP Security Console

The initial installation of XAMPP has the username *root* and there is no password. If you don't create a password, there is a small security risk when connected to the Internet. Also, if you work in the same room with other persons, the password will protect against interference providing the password is not divulged to the other persons. For best practice you should password protect your working environment, and XAMPP has a Security Console that simplifies this task.

Start XAMPP by double-clicking the desktop icon. Then enter the following URL in the address field of a browser:

`http://localhost/security/`

A page will appear as shown in Figure 21-7. Select your language in the left panel.

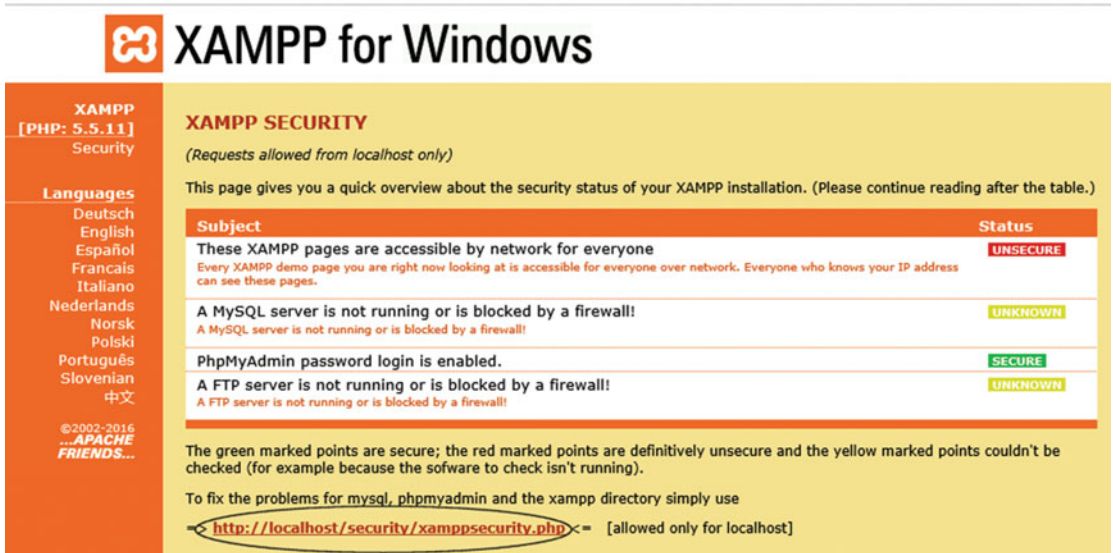


Figure 21-7. The XAMPP Security console

Any items with a red background indicate the unprotected components. Click the URL shown ringed and you will be taken to the next page shown in Figure 21-8. Only the top half of the page is shown because the password is sufficient protection; the rest of the page can be ignored. The XAMPP screens for the MAC are similar and the procedure is the same; however, you should download the manual for the MAC version of XAMPP to be sure.



Figure 21-8. The XAMPP form for entering a password

Enter a password and confirm it. Then click the *Password changing* button. **Make a careful note of the password.**

The *htdocs* Folder

Open the C:\XAMPP folder to display XAMPP's component folders; some of the the folders are shown in Figure 21-9.

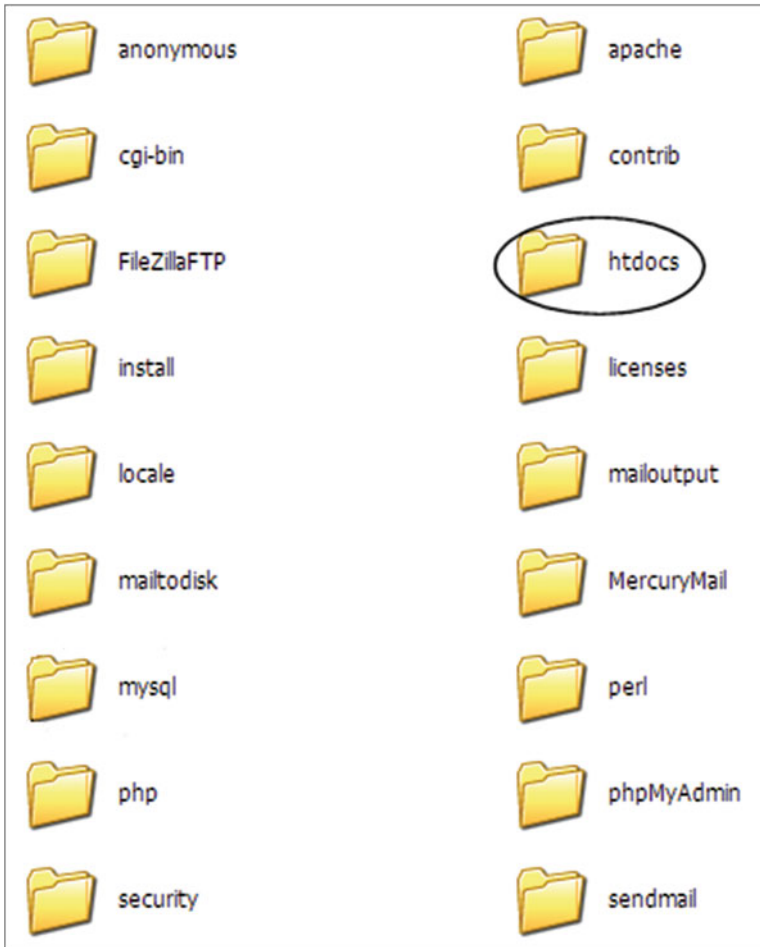


Figure 21-9. The folders in the XAMPP package

Within the XAMPP folder, the *htdocs* folder (shown circled), is the folder that you must use for this chapter; this is where you must place all your PHP and HTML pages for your website. A PHP parser in the folder named *php* runs in the background waiting to highlight any errors in your PHP code.

Create a shortcut on your Desktop for XAMPP's *htdocs* folder and place it alongside the XAMPP icon as shown in Figure 21-10. One icon starts XAMPP and the other allows you to quickly access the files in the *htdocs* folder.

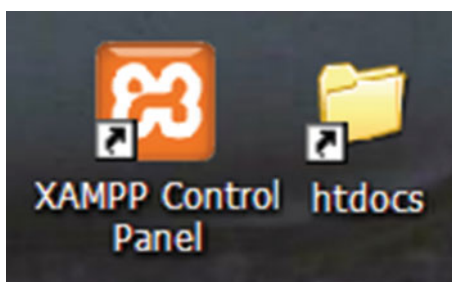


Figure 21-10. Time-saving desktop shortcuts

Testing PHP Files in XAMPP

Now that you have a server on your computer, you access it and the folder *htdocs* by using the address *http://localhost/somefolder/* Note that you must not add the *htdocs* folder name; instead you add the name of the folder within *htdocs* that contains your website pages. You can access and view your pages with a browser as follows:

Make sure that XAMPP and Apache are running, minimize the XAMPP control panel and then type the following in a browser's address field:

http://localhost/somefolder/yourfile.php OR
http://127.0.0.1/somefolder/yourfile.php

Of course, you must substitute your own folder and file name in place of *somefolder* and *yourfile*.

In this chapter, for the home page only, that is, *index.php*, you can shorten the entry to:

http://localhost/Chapter-21/

■ **Warning!** If you get a weird result when viewing your files in the Apache server, you probably entered one of these incorrect addresses; the *incorrect* items are shown in bold type:

file:///C:/xampp/htdocs/Chapter-21/somefile.php

or *http://localhost/htdocs/Chapter-21/somefile.php*

If you can view a page, but the included elements are missing, or you see a “Page not available” message, you probably forgot to start the Apache server in XAMPP.

A few students working on the project for this chapter were distracted by small problems associated with the frills added in Chapter 20, that is, the crocus and the rosette. I have therefore eliminated these so that the reader can focus entirely on the main point of this chapter, which is the PHP *include* command. This explains why this chapter is based on the Chapter 19 instead of Chapter 20. I have further simplified the project by eliminating the colored columns.

The PHP *include* command incorporates the referenced file into the main HTML page. For instance, if the include command refers to a menu file, the menu will be inserted into HTML source code just as if the menu was initially typed into the web page. The first advantage is that it is so much quicker to type the include command:

```
<?php include ("nav.html"); ?>
```

into every web page instead of typing the full menu like this:

```
<nav>
<ul>
<li class="btn"><a href="spring-bulbs.php" title="Spring garden bulbs">Spring
bulbs</a></li>
<li class="btn"><a href="spring-seeds.php" title="Spring garden seeds">Spring
Seeds</a></li>
<li class="btn"><a href="spring-plants.php" title="Spring plants">Spring Plants</a></li>
<li class="btn"><a href="index.php" title="Return to Home page">Home Page</a></li>
</ul>
</nav>
```

The second, and most important advantage, is that if at a later date you wish to change the menu, you only need to change one file, that is, the menu file. The change will cascade automatically into every page in the website.

Using the PHP *include* Command

To learn how to use the PHP *include* command, please follow these steps:

1. In the *web-tutorial* folder, create a new folder named *Chapter-21*.
2. Open the *Chapter-19* folder.
3. Use Ctrl+A then Ctrl+C to copy all the *Chapter-19* files into the computer's memory.
4. Open the new folder *Chapter-21*.
5. Paste the items from the computer's memory (use Ctrl+V) into the folder *Chapter-21*.
6. Right-click the file *index.html* and rename it as *original-index.html*. This is a precaution; if something goes wrong, you will be able to revert to the original index file and start again.
7. Open the file *original-index.html* and use *Save As* to save a copy as *index.html*. You will now have two index files.
8. Open the file *style 3d-gdn.css* in your text editor and remove the line of code shown in bold type and crossed through in the following snippet of code:


```

body {background-image:url(images/spring-tile-2.jpg);}
#wrapper {margin:auto; max-width:1110px; min-width:1045px; background:white;
    border:10px white solid; font-size:medium;}
header,nav,article,section,figure,footer {display:block;}
header {margin-top:0; border-bottom:10px white solid; height:181px;
    background-image:url('images/spring-header.jpg');}
#content {background-image:url('images/grenswrl.gif'); min-width:980px;}
h1 {position:absolute; top:-15px; margin-left:30px; font-size:450%; color:white;
    font-weight:bold;}
h2 {font-size:150%; text-align:center;}
p {font-size:120%; text-align:center;}
a {font-size:120%;}
/*add a left col*/
#leftcol {float:left; width:150px;}
#rightcol {float:right; width:150px;}

```

Deleting that line removes the last bit of decoration that bears no relation to the project.

Create the External Files

The *include* command will pull code from external files for display in browsers. We will therefore create external files for the elements that are common to every page in the website; these are as follows:

- The header
- The horizontal menu
- The vertical menu
- The footer

When you amend any of the external files, the amendments will automatically appear on every page that has a PHP *include* statement.

Create the External File for the Header

To create the external file for the header, please follow these steps:

1. Open the file *index.html* in the Code/Source view of your HTML text editor.
2. Select and copy the header code into the computer's memory.
Keep the *index.html* file visible in your text editor because we will be copying several other items from it.
3. Select and copy the header code into the computer's memory.
4. Open a new page in your HTML text editor; delete all visible code to create a blank page and paste in the code from the computer's memory as shown in Listing 21-1.

Listing 21-1. Create the external file for the header

```
<header>
    <h1>The Spring Garden</h1>
</header>
```

5. Save the file as *header.html*.

Create the External File for the Horizontal Menu

To create the external file for the horizontal menu, please follow these steps:

1. With *index.html* file on the screen in Code view, select and copy the horizontal menu's code into the computer's memory (Ctrl+C).
2. Open a new page in your HTML text editor; delete all visible code to create a blank page, then use Ctrl+V to paste in the code into your new file as shown in Listing 21-2.
3. The copied code will have an *.html* suffix in each link, change these to *.php* as shown in bold type in Listing 21-2. This is because all the main HTML web pages will be given a *.php* suffix so that they are capable of using the include command.

Listing 21-2. Create the external file for the horizontal menu

```
<nav>
<ul>
<li class="btn"><a href="spring-bulbs.php" title="Spring garden bulbs">Spring 🌷
bulbs</a></li>
<li class="btn"><a href="spring-seeds.php" title="Spring garden seeds">Spring 🌱
Seeds</a></li>
<li class="btn"><a href="spring-plants.php" title="Spring plants">Spring Plants</a></li>
<li class="btn"><a href="index.php" title="Return to Home page">Home Page</a></li>
</ul>
</nav>
```

4. Save the file as *nav.html*.

Create the External File for the Vertical Menu

To create the external file for the vertical menu, please follow these steps:

With *index.html* file on the screen, select and copy the vertical menu's code into the computer's memory.

1. Open a new page in your HTML text editor; delete all visible code to create a blank page and paste in the code from the computer's memory. Add the items shown in bold type as shown in Listing 21-3.
2. The copied code will have an *.html* suffix in each link, change these to *.php* as shown in bold type.

Listing 21-3. Create the external file for the vertical menu with links to three new pages

```
<nav id="vertical">
  <ul>
    <li class="vbtn"><a href="about.php" title="About Us">About Us</a></li>
    <li class="vbtn"><a href="contact.php" title="Contact Us">Contact Us</a></li>
    <li class="vbtn"><a href="locate.php" title="Locate Us">Locate Us</a></li>
    <li class="vbtn"><a href="terms.php" title="Terms and Conditions">Terms</a></li>
  </ul>
</nav>
```

3. Save the file as *vertnav.html*.

Create the External File for the Footer

To create the external file for the footer, please follow these steps:

1. With *index.html* file on the screen, select and copy the *footer's* code into the computer's memory.
2. Open a new page in your HTML text editor; delete all visible code to create a blank page.
3. Paste in the footer code from the computer's memory as shown in Listing 21-4.
4. Change the files' names from *.html* to *.php* as shown in bold type in Listing 21-4.

Listing 21-4. Create the external file for the footer

[illegible]

5. Save the file as *footer.html*.

We now have four external files, and each one is an element that is present in every page of the website, that is, the header, the horizontal menu, the vertical menu, and the footer.

Inserting the PHP *include* Command to Replace the Code for the Four Included Elements

Now we must make the *index.html* page pull these elements into the page using PHP function *include*.

1. With *index.html* file on the screen, amend the home page as shown in bold type in Listing 21-5.
2. Use *Save as* to save the files as *index.php*. Then delete the file *index.html*.

■ **Important** As shown in Listing 21-5, you must comment-out the code for the four elements that have been replaced by a PHP *include* command. Commenting-out the four elements is a precaution in case something goes wrong; you can then start afresh by removing the comment tags `<!--` and `-->`. Later, if all goes well, you will delete the four commented-out elements entirely, leaving only the tiny snippets of PHP *include* code.

Listing 21-5. Replace the four elements in the home page with PHP include statements, and comment-out the unwanted blocks of code for the header, horizontal menu, vertical menu, and the footer. Only the body section is shown to save space

```
<body>
  <div id="wrapper">
<!--The header that is commented-out below can now be deleted -->
<!--<header>
<h1>The Spring Garden</h1>
</header>-->
  <?php include ("header.html"); ?>
  <?php include ("nav.html"); ?>
<!--The menu that is commented-out below can now be deleted -->
<!--<nav>
<ul>
<li class="btn"><a href="spring-bulbs.html" title="Spring garden bulbs">Spring
Bulbs</a></li>
<li class="btn"><a href="spring-seeds.html" title="Spring garden seeds">Spring
Seeds</a></li>
<li class="btn"><a href="spring-plants.html" title="Spring plants">Spring Plants</a></li>
<li class="btn"><a href="index.html" title="Return to Home page">Home Page</a></li>
</ul>
</nav>-->
  <div id="content">
  <div id="leftcol">
    <?php include ("vertnav.html"); ?>
<!--The menu that is commented-out below can now be deleted -->
<!--<nav id="vertical">
<ul>
<li class="vbtn"><a href="#" title="About Us">About Us</a></li>
<li class="vbtn"><a href="contact.html" title="Contact Us">Contact Us</a></li>
<li class="vbtn"><a href="#" title="Locate Us">Locate Us</a></li>
<li class="vbtn"><a href="#" title="Terms and Conditions">Terms</a></li>
</ul>
</nav>-->
```

```
</div>
    <div id="rightcol">
        <p>This is the far right column</p>
    </div>
    <div id="midcol">
        <h2>Spring Garden Bulbs, Seeds and Plants</h2>
        <div id="midcol-left">
            <p class="left">Spring, the sweet spring, is the year's pleasant king; Then ↵
                blooms each thing...<br>
                <span class="quote">From "Spring" by Thomas Nash</span><br>
            <p class="left">Nothing heralds the arrival of spring better than a garden full ↵
                of spring flowers. This website will help you to plan your spring flower display ↵
                and choose the best spring bulbs, seeds and plants.</p>
            <p>View and order spring bulbs, seeds and plants using this website, or visit ↵
                the Spring Garden Centre at 10 The Street, Townsville.</p>
        </div>
        <div id="midcol-right">
            
        </div>
    </div>
    <div><!--content div finishes here-->
        <br class="clear">
        <?php include ("footer.html"); ?>
<!--The footer that is commented-out below can now be deleted-->
<!--<footer>
    <p>
    <a class="small" href="spring-bulbs.html" title="Spring garden bulbs">Spring Bulbs</a>
    &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~
    <a class="small" href="spring-seeds.html" title="Spring garden seeds">Spring Seeds</a>
    &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~
    <a class="small" href="spring-plants.html" title="Spring garden plants">Spring Plants</a>
    &nbsp;&nbsp;&nbsp;&nbsp;&~
    <a class="small" href="index.html" title="Return to Home page">Home Page</a>
    </p>
</footer>-->
</div><!--wrapper div finishes here-->
</body>
```

■ **Important** We want browsers to recognize that files contains some PHP; therefore use *Save As* and save the files with a *.php* suffix

View the File Using the XAMPP Server on Your Computer

The XAMPP server will only read files that are stored in the *htdocs* folder on your computer. Therefore please follow these steps:

1. Open the *web-tutorial* folder, click the folder *Chapter-21* and use Ctrl+C to copy it into the computer's memory.

- 2. Double-click the *htdocs* icon that you placed on the desktop.
- 3. When the *htdocs* folder opens, use Ctrl+V to paste the *Chapter-21* folder into the *htdocs* folder.
- 4. In the *htdocs* folder, rename the *index.html* file as *old-index.html* to prevent the server loading the *index.html* file instead of the *index.php* file.
- 5. Double-click the XAMPP icon on your desktop, start Apache and then type the following address in the address field of your browser and press Enter:

http://localhost/Chapter-21/

Caution Because we have not yet created the pages with the file endings *.php*, do not click any of the menu buttons. If you do, you will see the “Page not available” message. The new pages with file suffix *.php* will be created in the next chapter.

The home page should look exactly the same as when the page contained no PHP *include* commands; the result should be as Figure 21-11.



Figure 21-11. The home page using the PHP *include* command

To use the Server on a Remote Host

Use your FTP program and upload the *Chapter-21* folder to the root of your host (usually `public_html`).

Open a browser and access the home page *index.php* by entering the following in the Address field:

<http://mywebsite.co.uk/Chapter-21/>

Change *mywebsite.co.uk* to the name of your own website.

If the home page is satisfactory, you can delete the four commented-out elements from the home page *index.php*

■ **Caution** Because we have not yet created the other pages with the file endings *.php*, do not click any of the menu buttons. If you do, you will see the “Page not available” message. The new main pages with the suffix *.php* will be created in the next chapter.

Summary

In this chapter you learned how the PHP *include* file functions. You discovered that PHP was a server-side language and that you could only test the file by using either the server at a remote host, or a server installed on your own computer. Instructions were provided for installing the XAMPP package that includes the Apache Server and a PHP parser. You removed four elements from the home page and replaced them with PHP *include* commands. You prepared four external files so that the *include* commands could pull them into the home page. You then tested the *index.php* file in either a remote server or on the XAMPP Apache server on your own computer.

To save space and time, the project was limited to the home page, but in the next chapter you will add several other pages containing the *include* statements using files with *.php* endings. You will then be able to explore for yourself the advantage of the time-saving shortcut provided by the PHP *include* command.

CHAPTER 22



More on Using PHP *include*

The previous chapter explained the advantages of using the PHP *include* command. In this chapter the considerable advantages will be demonstrated in greater detail. In Chapter 21 we only used the *include* commands in the home page *index.php*. In this chapter we will create new pages and use the *include* commands in every page of an eight-page website. All the new pages will be adaptations of existing pages.

This chapter contains the following sections:

- Using the PHP *include* command
- Change the external footer file (*footer.html*)
- Make copies of existing pages to create new pages
- Create a template and a new page (*spring-bulbs.php*)
- Summary

We will be creating six new pages as follows:

spring-bulbs.php, *spring-seeds.php*, *spring-plants.php*, *about.php*, *location.php*, and *terms.php*

This exercise creates a simple eight-page site so that we can demonstrate the time-saving features of the PHP *include* command. The content of each page is deliberately brief to save time and space.

You will need a new folder in your *htdocs* folder to hold this project's eight-page website. To create a new folder in your *htdocs* folder, please follow these steps:

1. In the *htdocs* folder or the *web-tutorial* folder, create a new folder named *Chapter-22*.
2. Open the previous chapter, that is, the *Chapter-21* folder.
3. Use Ctrl+A then Ctrl+C to copy all the files in *Chapter-21* into the computer's memory.
4. Open the new folder *Chapter-22*.
5. Paste the items from the computer's memory (use Ctrl+V) into the folder *Chapter-22*.
6. In the *Chapter-22* folder, the following files are no longer required so delete them:
page-2.html, *page-3.html*, *page-4.html*, *form.html*, and *original-index.html*
7. Open the file *index.php* in the Code/Source view of your HTML text editor.

Because the internal style `.small {font-size:75%; color:black;}` will be used for the footer in every page of the website, we will place it in the style sheet *style-3d-gdn.css* and remove it from the web pages.

- Open *style-3d-gdn.css* in your HTML editor and insert the following anywhere in the style sheet:

```
•small {font-size:75%; color:black;}
```

■ **Note** Be aware of the full stop in front of the word *.small*; this defines the attribute *small* as a class. Also note that when a style is transferred from an HTML or PHP file into a CSS file, the two tags `<style` `type="text/css">` and `</style>` must not be included in the style sheet.

9. In the file *index.php* delete the internal style shown crossed through as follows:

```
<style type="text/css">
——.small {font-size:75%;color:black;}
</style>—
```

10. Save the file,

Using the PHP *include* Command

Because all the new pages will contain PHP *include* commands, they must be given the suffix *.php*

The new pages will require new links in the menu and in the *footer*. This website has 8 pages, including 6 modified pages. If we had not used the PHP *include* command, we would have to manually modify 3 items in each of the 2 menus in every page, plus 7 items in every footer in every page; that is a grand total of 72 manual modifications. By using the PHP *include* command as described in the previous chapter and in this chapter, we only make a total of 11 changes.

Once the include commands are in place, any changes to the header, menus, or footer can be achieved by modifying a single CSS file.

To access the four new pages *about.php*, *contact.php*, *location.php*, and *terms.php*, we will add four new links to those pages in the footer menu.

Change the External Footer File (*footer.html*)

Because we will eventually have eight web pages, we will increase the number of page links in the footer so that they link to all eight. The link to *index.php* will be moved down so that it is last in the list.

Open the file *footer.html*, then add the four new links and alter the file names in the last lines as shown in bold type:

[illegible]

[illegible]

Save the file as *footer.html*

Important: Rename *index.html* as *old-index.html*

■ **Caution** If your folder contains *index.html* as well as *index.php*, all browsers will give preference to the *index.html* file, in which case you won't see your changes.

At this point in the project, it would be wise to test the *index.php* file in the server installed on your computer. Start XAMPP and Apache, then open a browser and enter the address as follows:

http://localhost/Chapter-22/ Then press Enter

If you wish to test the *index.php* file in a website (assuming you have the FTP login details and an FTP program), upload the folder *Chapter-22* to the root folder of your website. Access <http://www.yourwebsite.co.uk/Chapter-22/> to test the pages.

The result should be as shown in Figure 22-1:



Figure 22-1. Showing the revised home page. Note the four additional links in the footer. Currently most of the links and menu buttons are dead, except for the home page and the contact page. We will eventually add new pages and make live links to them

Make Copies of Existing Pages to Create New Pages

The next section will demonstrate that web design and adding pages can be simply a matter of copying and pasting from your existing website pages. You will rarely have to start from scratch.

Create a Template and a New Page (*spring-bulbs.php*)

We will now create a template that will enable you to produce additional pages for the website. Please follow these steps:

1. With the file *index.php* open in your HTML text editor, use *Save As* to create a copy of the file with the name *template.php*. We will use this template to create several new pages.

Keep the file *template.php* open in your HTML text editor.

2. Within the head section modify the text shown in bold type in the following snippet:

```
<!DOCTYPE html>
<html>
  <head>
```

```

<title>Bulbs for the spring garden</title>
<meta charset=utf-8>
<meta name="description" content="Bulbs, crocuses, snowdrops, daffodils for the
spring garden">
<meta name="keywords" content=" bulbs, spring bulbs, spring garden bulbs">
<link href="style-3d-gdn.css" rel="stylesheet" type="text/css">

```

3. Modify the text in the *midcol* and *midcol-left* sections as shown in bold type in the next code snippet:

```

<div id="midcol">
    <h2>Spring Garden Bulbs</h2>
    <div id="midcol-left">
        <p class="left">View and order spring bulbs using this web site, or visit
        the Spring Garden Centre at 10 The Street, Townsville.</p>
    </div>

```

4. Save the page as *template.php*.
5. Use *Save As* to make a copy of *template.php* file with the name *spring-bulbs.php*. This is your first new page.

Create the Page *spring-seeds.php*

We will now create the second new page; please follow these steps:

1. With the *spring-bulbs.php* page on the HTML editor's screen, use *Save As* to save a copy with the new name *spring-seeds.php*.
2. Then modify the *head* section as shown in bold type in the following code snippet:

```

<!DOCTYPE html>
<html>
    <head>
        <title>Seeds for the spring garden.</title>
        <meta charset=utf-8>
        <meta name="description" content="Seeds for the spring garden">
        <meta name="keywords" content="Seeds, spring garden seeds">
        <link href="style-3d-gdn.css" rel="stylesheet" type="text/css">
    </head>

```

3. Modify the text in the *midcol* and *midcol-left* sections as shown in bold type in the following code snippet:

```

<div id="midcol">
    <h2>Spring Garden Seeds</h2>
    <div id="midcol-left">
        <p class="left">View and order spring seeds using this web site, or visit
        the Spring Garden Centre at 10 The Street, Townsville.</p>
    </div>

```

4. Save the file.

Create the Page *spring-plants.php*

We will now create the third new page; please follow these steps:

1. Open the *template.php* file in the Code/Source pane of your HTML text editor and use *Save As* to save a copy named *spring-plants.php*.
2. Then change the *head* section as shown in bold type in the following listing:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Plants for the spring garden.</title>
    <meta charset=utf-8>
    <meta name="description" content="Plants for the spring garden">
    <meta name="keywords" content="plants, spring garden plants">
    <link href="style-3d-gdn.css" rel="stylesheet" type="text/css">
```

3. Change the text in the *midcol* and *midcol-left* sections as shown in bold type in the following code snippet:

```
<div id="midcol">
  <h2>Spring Garden Plants</h2>
  <div id="midcol-left">
    <p class="left">View and order spring plants using this web site, or visit ↩
      the Spring Garden Centre at 10 The Street, Townsville.</p>
  </div>
```

4. Save the file.

To save time, the three new pages you just created were deliberately brief and contained the same picture; in a real-world website they would contain relevant pictures and prices.

■ **Note** If you are not working in the folder *htdocs* you will not be able to view any of the new files in your browser unless you upload the folder Chapter-22 to your remote host. If you created the folder Chapter-22 within the *htdocs* folder, you can view the files in a browser by entering:

<http://localhost/Chapter-22/filename.php> (Change the file name to an appropriate file name)

We now need to create the three new pages that can be accessed from the vertical menu.

Create the Page *about.php*

We will now create the first new page for the vertical menu; please follow these steps:

1. Open the *template.php* file in the Code/Source pane of your HTML text editor and use *Save As* to save a copy named *about.php*.
2. Then change the head section as shown in bold type in the following snippet:

```
<!DOCTYPE html>
<html>
  <head>
    <title>About The Spring Garden</title>
    <meta charset=utf-8>
    <meta name="description" content="About our bulbs, seeds and plants for the spring garden">
    <meta name="keywords" content=" bulbs, seeds, plants, spring bulbs, spring garden ←
      seeds, spring garden plants">
```

3. Change the *midcol* and *midcol-left* sections as shown in bold type in the next code snippet:

```
<div id="midcol">
  <h2>About The Spring Garden Company</h2>
  <div id="midcol-left">
    <p class="left">We were established in 1936 and have a reputation for supplying ←
      a superb range of Spring Garden bulbs, seeds and plants. Order from this ←
      website, or visit The Spring Garden Centre at 10 The Street, Townsville.
    </p>
  </div>
```

4. Save the file.

Create the Page Named *location.php*

We will now create the second new page for the vertical menu; please follow these steps:

1. Open the *template.php* file in the Code/Source pane of your HTML text editor and use *Save As* to save a copy named *location.php*.
2. Change the content of the head section as shown in bold type in the following code snippet:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Location of The Spring Garden Company</title>
    <meta charset=utf-8>
    <meta name="description" content="The location of the Spring Garden Company. Bulbs, ←
      seeds and plants for the spring garden">
    <meta name="keywords" content=" bulbs, spring bulbs, spring garden bulbs">
```

3. Change the *midcol* and *midcol-left* sections as shown in bold type in the next code snippet:

```
<div id="midcol">
  <h2>The Location of the Spring Garden centre</h2>
  <div id="midcol-left">
    <p class="left">We are located at The Spring Garden Centre,<br>at 10 The Street, ←
      Townsville.</p>
  </div>
```

4. Save the file.

Create the Page Named *terms.php*

We will now create the third new page for the vertical menu; please follow these steps:

1. Open the *template.php* file in the Code/Source pane of your HTML text editor and use *Save As* to save a copy named *terms.php*.
2. Then change the content of the head section as shown in bold type in the following snippet:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Terms and conditions</title>
    <meta charset=utf-8>
    <meta name="description" content="Terms and conditions Bulbs, seeds and plants for the
spring garden">
    <meta name="keywords" content="bulbs, spring bulbs, seeds, plants, spring garden bulbs">
```

3. Change the *midcol* and *midcol-left* sections as shown in bold type in the next code snippet:

```
<div id="midcol">
  <h2>Terms and conditions</h2>
<div id="midcol-left">
  <p class="left">Pay by debit/credit card or PayPal.<br>Free delivery normally 3 days.
  <br>Next day delivery available &pound;2 per order.
  </p>
</div>
```

4. Save the file.

A fourth page *contact.html* already exists but its suffix must be changed to *.php* and its content changed to accommodate the PHP *include* commands.

Modify the Page Named *contact.php*

The contact page contains the feedback form that we do not want to re-create from scratch. Because *form.html* already exists, we will alter it by replacing the *header*, *menus*, and *footer* with the *include* commands. To achieve this we will insert the four *include* commands, delete the header code, menu codes, and the footer code, and save the file as *contact.php*.

1. Open the *contact.html* file in the Code/Source pane of your HTML text editor and use *Save As* to save a copy named *contact.php*.
2. Ensure that the title is correct as shown in bold type in the following snippet:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Contact Us</title>
```

3. Then within the wrapper section, delete the code shown crossed through, and add the code as shown in bold type in the following snippet:

```

<body>
  <div id="wrapper">
    <header>
<h1>The Spring Garden</h1>
</header>
    <?php include ("header.html"); ?>
    <?php include ("nav.html"); ?>
  <nav>
    <ul>
      <li class="btn"><a href="page-2.html" title="Go to page 2">Go to Page 2</a></li>
      <li class="btn"><a href="page-3.html" title="Go to page 3">Go to Page 3</a></li>
      <li class="btn"><a href="page-4.html" title="Go to page 4">Go to Page 4</a></li>
      <li class="btn"><a href="index.html" title="Return to Home page">Home Page</a></li>
    </ul>
  </nav>
  <div id="content">
  <div id="leftcol">
    <?php include ("vertnav.html"); ?>
  <nav id="vertical">
    <ul>
      <li class="vbtn"><a href="#" title="About Us">About Us</a></li>
      <li class="vbtn"><a href="contact.html" title="Contact Us">Contact Us</a></li>
      <li class="vbtn"><a href="#" title="Locate Us">Locate Us</a></li>
      <li class="vbtn"><a href="#" title="Terms and Conditions">Terms</a></li>
    </ul>
  </nav>
</div>

```

4. Then go to the end of the page, change the code as shown in bold type, and delete the code shown crossed through in the following snippet:

```

</div><!--end of form-->
</div>
</div><!--content div finishes here-->
  <?php include ("footer.html"); ?>
<footer>
<br>&del>This is the footer<br><br>
</footer>
</div>
</div><!--wrapper div finishes here -->
</body>
</html>

```


5. Save the file.

Test all the pages in the website; if you have the XAMPP/Apache server installed on your computer, start XAMPP and Apache, then open a browser and enter the address as follows:

`http://localhost/Chapter-22/`

Press *Enter* then test all the menu buttons.

If you use the server on a remote host, upload the *Chapter-22* subfolder to the host and enter the appropriate web address and test all the menu buttons.

To add more pages, use the *template.php* file to create the files. Then add the new page file names to the relevant external files. As your website grows, imagine the work you will save by using the PHP *include* command; after adding each new page you only need amend one of the external menu files and the external footer file. On a large site, say 30 pages, if you did not use *include* commands, the menus and footers on 30 pages would have to be amended manually, and you would have to make a minimum of 60 amendments instead of 2.

Summary

In this chapter you learned how to take advantage of the PHP *include* command when adding new pages to a website. You produced a template that you used to create six new pages. If you installed the XAMPP/Apache server, you were able to practice testing your new PHP pages on your own computer.

In the next chapter, you will learn how emails can be sent to the website's owner from the website's *form.php* page.

CHAPTER 23



Receive Emails from a *Contact Us* Page

By using a PHP form-handler to process the feedback-form *contact.php*, we can provide a reasonably spam-free way of receiving emails. Also the information entered by the user can be filtered to remove malicious content. This exercise is not intended to create a fully worked website; its purpose is to demonstrate a PHP form-handler. Therefore the content of the pages will be deliberately brief to save time and space, and we will use all the pages from the previous chapter as a basis for this chapter.

When the user enters data into the contact form, the form posts the data to the form-handler. The form-handler then sends the information by email to the website's owner.

This chapter contains the following sections:

- Examine the code of the downloaded page *form-handler-typical.php*
- Validating user input
- Create the message pages for the project
- Create the style sheet for the Thank-you page and the Error message pages
- Test the Form-handler and the Messages
- Summary

As a reminder, Figure 23-1 shows the contact form we produced in Chapter 18:



Figure 23-1. The contact page posts the user's entries to a form-handler that sends the email

The Contact Us form requires a new page with its own CSS formatting. To create this page, please follow these steps:

1. If you have installed the XAMPP package, open the htdocs folder and create a new folder named *Chapter-23*.
2. Open the previous chapter, that is, the *Chapter-22* folder.
3. Use Ctrl+A then Ctrl+C to copy all the files and folders in *Chapter-22* into the computer's memory.

4. Open the new folder *Chapter-23* in the folder *htdocs*, or if you are not using XAMPP open the folder *Chapter-23* in the folder *web-tutorial*.
5. Paste the files and folders from the computer's memory (use Ctrl+V) into the folder *Chapter-23*.
6. Download the *Chapter-23.zip* file from your book's page at Apress.com and unzip it in your folder *Chapter-23* within the *htdocs* folder (or in the folder *Chapter-23* in the *web-tutorial* folder if you intend to upload to a website to test it).
7. If you have not installed XAMPP, and if you have an FTP program for logging-in to a remote host, you can use the remote host to test your files. In this case, place the new folder in the *web-tutorial* folder named *Chapter-23*.

■ **Note** You do not need to create the code for the form-handler; this is provided by two files in the downloaded folder. One file is named *form-handler-project.php*, and the other is named *form-handler-typical.php*. The two files are almost identical except that *form-handler-project.php* has a few amendments that will enable you to test the error messages in your XAMPP/Apache server.

You can adapt the file *form-handler-typical.php* for your own websites. You do not have to change any of the code shown in normal type in Listing 23-1; you just need to change a few elements shown in bold in Listing 23-1. Those elements are your web address, your subject, your email address, and the names of your website pages and your products. I have given an explanation of the code in case you wish to understand how it works. To adapt the file for your own use, you need to change the file name to *form-handler.html*,

Examine the Code of the Downloaded Page *form-handler-typical.php*

We will now examine the code for the feedback form. Later, when you wish to adapt the “Contact Us” form and its form-handler to your own websites, open the form-handler file in your text editor and change only those items shown bold in Listing 23-1. Change the product or service content to match your Contact Us form. For instance, instead of spring garden items, you may be promoting some other product or service.

■ **Tip** You may find the code in this section heavy going. It does not matter if you don't understand it. I have emphasized items with bold type that relate directly to the “Contact Us” form fields, and to the website's products or services. Try to discern the logic of those items in bold type. The commented-out items shown in uppercase will help you to understand the logic of the steps in the code. The explanation of the code given after the Listing will also be helpful.

Listing 23-1. The code for a typical form-handler that you can adapt for your own websites

```
<?php
//Feedback Form-handler//
// SET THE WEBSITE OWNER'S EMAIL ADDRESS
$mailto = "owners-address@example.co.uk";
// SET THE WEB DESIGNERS EMAIL ADDRESS FOR TEST PURPOSES
$mailto = "designers-address@example.co.uk";
$subject = "Feedback from The Spring Garden website";
// LIST THE PAGES THAT MAY HAVE TO BE DISPLAYED
$formurl = "contact.php" ;
$errorurl = "error.html" ;
$thankyouurl = "thankyou.html" ;
$emailerrurl = "emailerr.html" ;
$errorphoneurl = "phonerror.html" ;
$errormessageurl = "messagerror.html" ;
$errorboxurl = "boxerror.html" ;
$uself = 0;
// SET THE INFORMATION RECEIVED FROM THE FORM AS $ VALUES
$headersep = (!isset( $uself ) || ( $uself == 0 )) ? "\r\n" : "\n" ;
$username = $_POST['username'] ;
$useremail = $_POST['useremail'] ;
$userphone = $_POST['userphone'];
$bulbs = $_POST['bulbs'];
$seeds = $_POST['seeds'];
$plants = $_POST['plants'];
$contact=$_POST['contact'];
$message = $_POST['message'] ;
$http_referrer = getenv( "HTTP_REFERER" );
if (!isset($_POST['useremail'])) {
    header( "Location: $formurl" );
    exit;}
//CHECK THAT ALL THREE ESSENTIAL FIELDS ARE FILLED IN
if (empty($username) || empty($useremail) || empty($message)) {
header( "Location: $errorurl" );
    exit;}
//OPTIONAL FILTER FOR CHECK BOXES
//if ((!$bulbs and !$seeds and !$plants)) {
//header( "Location: $errorboxurl" );
//
//    exit;}

//CHECK THAT NO URLS HAVE BEEN INSERTED IN THE USERNAME TEXT AREA
if (strpos( $username, '://')||strpos($username, 'www') !==false){
    header( "Location: $errorsuggesturl" );
    exit;}
//CHECK THAT NO URLS HAVES BEEN ENTERED IN THE PHONE FIELD
if (strpos( $userphone, '://')||strpos($userphone, 'www') !==false){
    header( "Location: $errorphoneurl" );
    exit;}
// REMOVE ALL CHARACTERS FROM THE PHONE ENTRY THAT ARE NOT NUMBERS (DIGITS)
$userphone = preg_replace('/\D+/', '', ($userphone));
```

```

//CHECK THAT NO URLS HAVE BEEN INSERTED IN THE MESSAGE TEXT AREA
if (strpos($message, '://') || strpos($message, 'www') !== false) {
    header( "Location: $errormessageurl" );
    exit;}
if (preg_match( "[\r\n]", $username ) || preg_match( "[\r\n]", $useremail )) {
    header( "Location: $errorurl" );
    exit;}
//REMOVE ANY SPACES FROM BEGINNING AND END OF EMAIL ADDRESS
$useremail = trim($useremail);
//CHECK FOR PERMITTED EMAIL ADDRESS PATTERNS
$_name = "/^[-!#$%&'\*+\\.\\/0-9=?A-Z^_`{|}~]+";
$_host = "([0-9A-Z]+\.)+";
$_tlds = "([0-9A-Z]){2,4}$/i";
if(!preg_match($_name."@".$_host.$_tlds,$useremail)) {
    header( "Location: $emailerrurl" );
    exit;}
if(!$bulbs) {$bulbs = "No";}
if(!$seeds) {$seeds = "No";}
if(!$plants) {$plants = "No";}
$messageproper =
    "This message was sent from:\n" .
    "$http_referrer\n" .
    "-----\n" .
    "Name of sender: $username\n" .
    "Email of sender: $useremail\n" .
    "Telephone No: $userphone\n" .
    "bulbs?: $bulbs\n" .
    "seeds?: $seeds\n" .
    "plants?: $plants\n" .
    "Contact?:$contact\n" .
    "----- MESSAGE -----\n\n" .
    $message .
    "\n\n-----\n" ;
mail($mailto, $subject, $messageproper, "From: \"$username\" <$useremail> ");
header( "Location: $thankyouurl" );
exit;
?>

```

Explanation of the Typical Form-Handler Code

```

<?php
//Feedback Form-handler//
// Set the website owner's email address
$mailto = "owners-address@example.co.uk";
// Set the web designers email address for test purposes
// $mailto = "designers-address@example.co.uk";

```

The code begins with the PHP tag `<?php`, and this is followed by two single line PHP comments. The owner's email address and the web designer's email address appear next and are assigned to the PHP variable `$mailto`.

■ **Tip** Think of PHP variables as little boxes in the computer's memory that can store temporary information. PHP variables always start with a dollar sign, for example, `$formurl`

The variables in the form-handler code store the information posted from the Contact Us form.

In your own adaption of the form, the dummy addresses and the subject in bold type would be replaced by your own details. By moving the comment-out marks from the designer's address to the owner's address, the designer can test the form-handler when it is uploaded to a host. If the email is sent, and you receive it successfully, the comment-out tags can be switched back so that the owner will receive the emails.

■ **Important** To be able to test the error messages in the XAMPP/Apache server, you need to use `http://localhost` instead of an actual web URL; that is why the above section is different for the file *form-handler-project.php*. The code for the project on your own Apache server is as follows:

```
// List the pages to be displayed,
$formurl = "http://localhost/Chapter-23/contact.php";
$errorurl = "http://localhost/Chapter-23/error.html";
$thankyouurl = "http://localhost/Chapter-23/thankyou.html";
$emailerrurl = "http://localhost/Chapter-23/emailerr.html";
$errormessageurl = " http://localhost/Chapter-23/messagerror.html" ;
$errorboxurl = "http://www.localhost/Chapter-23/boxerror.html";
```

The contact page directive is `<form action="form-handler.php" method="post">` , therefore you must change the name of the handler file to *form-handler.php*

```
$subject = "Feedback from The Spring Garden website";
```

The subject of the email is assigned to a PHP variable named ***\$subject***.

```
// List the pages that may have to be displayed, insert your own website address
$formurl = "contact.php";
$errorurl = "error.html";
$thankyouurl = "thankyou.html";
$emailerrurl = "emailerr.html";
$errormessageurl = "messagerror.html" ;
$errorboxurl = "boxerror.html";
```

This code assigns PHP names to the fall-back and other types of message.

The second line assigns the address (URL) of the *contact.php* form to a PHP variable.

The third line relates to a *thank-you* page that notifies the user if the email was successfully sent.

If the user enters unsatisfactory content, an explanatory error message page will appear and the email will not be sent. The various error message pages are assigned to PHP variables. The message pages are described later in this chapter.

```

$uself = 0;
// Set a simplified version of the information received from the form
$headersep = (!isset( $uself ) || ($uself == 0)) ? "\r\n" : "\n";
$username = $_POST['username'];
$useremail = $_POST['useremail'];
$userphone = $_POST['userphone'];
$bulbs = $_POST['bulbs'];
$seeds = $_POST['seeds'];
$plants = $_POST['plants'];
$contact = $_POST['contact'];
$message = $_POST['message'];

```

From the fourth line to the end of this block of code, the user's entries that are posted from the *contact.php* form are assigned to PHP variables.

```

$http_referrer = getenv( "HTTP_REFERER" );
if (!isset($_POST['useremail'])) {
    header("Location: $formurl");
    exit;}

```

The code in the line beginning `if (!isset` means if the code in the variable (`$_POST['useremail']`) is not set (i.e., does not exist). The exclamation mark makes the attribute **isset** a negative, that is, NOT `isset`. In other words, if the user has not entered an email address, the code in the third and fourth lines is executed. If the user has not entered an email address there is no point continuing; therefore the feedback form *contact.php* is reloaded awaiting an entry from the user. The code `header("Location: $formurl");` and `exit;` is an instruction to reload *contact.php*. The address (URL) of *contact.php* is stored in the variable `$formurl`.

The command `exit;` terminates the PHP script and allows the user to correct the fault.

```

//Check that all three essential fields are filled in
if (empty($username) || empty($useremail) || empty($userphone)) {
header("Location: $errorurl");
    exit; }

```

This is a fall-back message for browsers that do not have built-in validation. The second line looks at the three required variables to ensure they have been filled in. If one or more of the variables is empty, the third line of code displays a page with an error message. The page also gives instructions so that the user can correct the omission. The `||` symbol means *or*

```

//OPTIONAL FILTER INCLUDED IN THE DOWNLOADED FILE
//Check that at least one box has been ticked
if (($bulbs and !$seeds and !$plants)) {
header("Location: $errorboxurl");
    exit;}

```

This filter code is optional and is included in the project file. If you wish to use it in your own adaptation, you need to mark the heading for the check boxes with a red asterisk; this indicates that one or more boxes must be ticked. The second line of code checks the content of the check box variables to determine that at least one box has been ticked. If all three box variables are empty, the third line of code displays a page with an error message; the page also gives instructions so that the user can correct the omission. As before, the `!` symbol means *not*.


```
//check that no urls have been inserted in the username text area
if (strpos($username, '://')||strpos($username, 'www') !==false){
    header("Location: errormessageurl" );
    exit ;}
```

Hackers may insert malicious web addresses (URLs) into form fields; these must be filtered out. The second line of code checks the *username* variable for the web address symbols // and www; if one or more of these is present, the third line of code displays a page with an error message. The page also gives instructions so that the user can correct the problem, (or in the case of a hacker, he will realize he has been found out and he will abandon any further attempt to send malicious links).

The function *strpos* examines the content of the variable and checks the position of the undesirable symbols if they exist. If no undesirable symbols are present, the function gives a value of *false*, in other words, as no symbols were found, their position cannot be determined. However, if one or more symbols exist, the function gives a value of *not false* (*!==false*), in which case an error message page is triggered.

```
//Check that no urls have been entered in the phone field
if (strpos($userphone, '://')||strpos($userphone, 'www') !==false){
    header( "Location: $errormessageurl" );
    exit ; }
```

The code checks the *\$userphone* variable to see if the user has entered a website address. If so, the error message appears and the form-handler will stop running until the offending item is removed by the user. If the user has not entered a web address the *else* clause is invoked. The *if...else* is a common PHP pattern and it simply means this: *if* the first test detects no problem, then *else* executes the next statement.

```
else
// Remove all characters from the phone entry that are not numbers (digits)
$userphone = preg_replace('/\D+/', '', ($userphone));
```

Users enter phone numbers in a variety of patterns, for example:

01111 222 333 or (01111)222333 or 01111-222-333

When gathering users' phone numbers, this code gives a uniform presentation of the numbers, and this is good practice. The code removes all unwanted characters in the *\$userphone* variable. For example, if the user enters (01111)-222-333, the code removes unwanted characters giving 01111222333. This does not show up on the contact form, but it does clear out the unwanted characters in the email message.

```
if (preg_match( "[\r\n]", $username ) || preg_match( "[\r\n]", $useremail )) {
    header( "Location: $errorurl" );
    exit ; }
```

This code prevents hackers using header injections that are dangerous; any further explanation is beyond the scope of this book.

```
//remove spaces from the beginning and end of the email address and check the emailformat
$useremail = trim($useremail);
//Check for permitted email address patterns
$_name = "/^[-!#$%&'*+\\.\\0-9=?A-Z^_`{|}~]+";
$_host = "([-0-9A-Z]+\\.)+";
$_tlds = "([0-9A-Z]){2,4}$i";
if(!preg_match($_name."@".$_host.$_tlds,$useremail)) {
    header( "Location: $emailerrurl" );
    exit ; }
```

This is a fall-back message for browsers that do not have built-in email validation. The complicated piece of regex code checks that the email address has been entered in the correct format. If the address is unacceptable, an error message page is displayed that allows the user to try again. The regex code is beyond the scope of this book, but it is part of the downloadable file *form-handler-typical.php* and requires no adaption to suit your own website. There is no need to understand it; you can be confident that email addresses will be tested for an acceptable pattern.

```
if(!$bulbs) {$bulbs = "No";}
if(!$seeds) {$seeds = "No";}
if(!$plants) {$plants = "No";}
```

This code reports which check boxes have not been ticked and assigns to them a value *No*, and this value is printed in the email (see Figure 23-2). The code `if(!$bulbs)` means; *if the box labeled bulbs has not been ticked*. The default value of each check box was originally set in the *contact.php* form as “Yes,” but if a box has *not* been ticked, the value is changed to “No” by the code `if(!$bulbs) {$bulbs = "No";}`

```
//check that no urls have been inserted in the message text area
if (strpos ($message, '://')||strpos($message, 'www') !==false) {
    header( "Location: $scriptmessageurl" );
    exit;}
```

This code forbids URLs and scripts in the *\$message* variable. Malicious persons can insert dangerous web addresses. URLs begin with either `http://` or `www` and they will be detected by the above code.

```
$messageproper =
    "This message was sent from:\n" .
    "$http_referrer\n" .
    "-----\n" .
    "Name of sender: $username\n" .
    "Email of sender: $useremail\n" .
    "Telephone No: $userphone\n" .
    "Bulbs brochure?:$bulbs\n" .
    "Seeds brochure?:$seeds\n" .
    "Plants brochure?:$plants\n" .
    "Contact me by?:$contact\n" .
    "----- MESSAGE ----- \n\n" .
    $message .
    "\n\n-----\n" ;
```

This code collects together and formats the email content. The symbol `\n` moves the next piece of content down one line. Note carefully that there must be a space and a full stop at the end of each line. Let’s look at a typical line of this code, “Name of sender: \$username\n” . The first part is printed in the email and this is followed by the content of the variable (see the result in Figure 23-2).

```
mail($mailto, $subject, $messageproper, "From: \"$username\" <$useremail>");
    header("Location: $thankyouurl");
    exit;
?>
```

The email is sent by the PHP *mail* function and a “thank-you” page is displayed.

The `?>` tag closes the PHP code.

Figure 23-2 shows a typical reply from the contact form and form-handler.

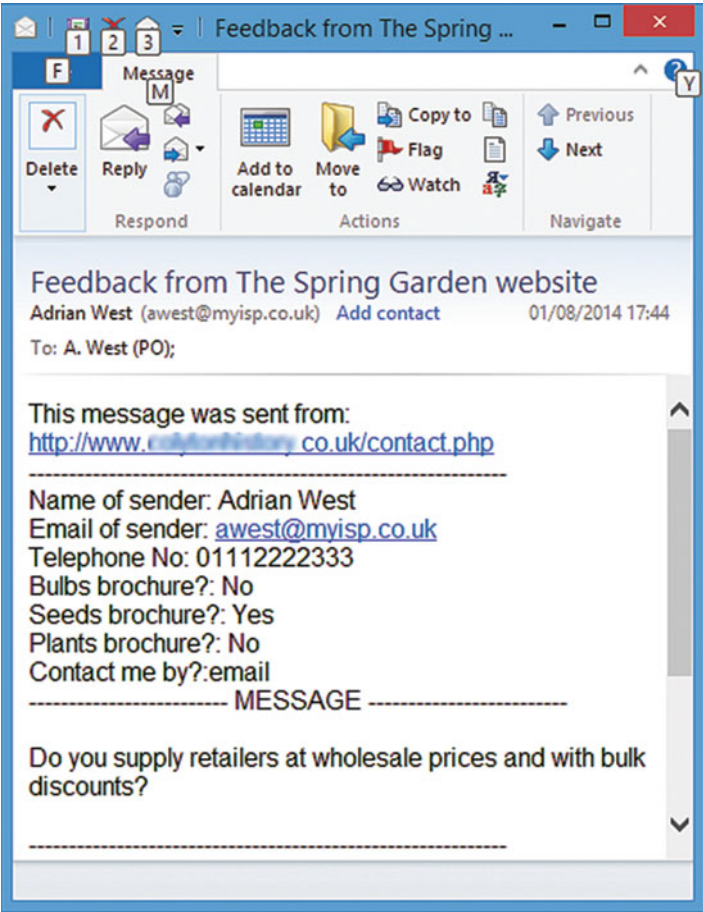


Figure 23-2. A typical email received from the filtered contact form (I have blurred the “from” address and shown a dummy email address and phone number for security)

Figure 23-3 shows a typical hacked email received from an unfiltered contact page.



Figure 23-3. A typical hacked feedback message; note the links to suspicious website addresses

The above example was taken from an actual hacked feedback page, but I have changed the suspicious URLs and email addresses for security.

Validating User Input

The latest browsers contain some built-in validation of the user's input, but at the time of writing the browsers differ slightly; that is, they do not all check the same unacceptable user input. Also, because older browsers will still be used for a while, it is better to accept that browsers vary and therefore we should provide a fall-back to cover as many cases as possible. When the *submit* button is clicked, several modern browsers will halt the form and provide a brief pop-up message. They also add a distinctive red border to a field that is empty or wrongly filled in. Figure 23-4 shows the pop-up and the red borders around three *required* fields that are empty. Different browsers may show a different style of pop-up.

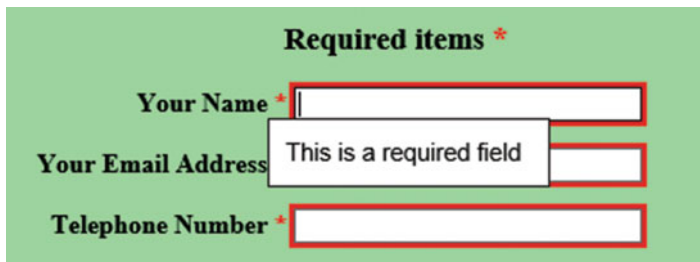


Figure 23-4. Three empty required fields are highlighted in red, and a pop-up appears on the first field

To cater for those modern browsers that have built-in validation, please follow these steps:

1. Open the file *contact.php* in your text editor.
2. Scroll down to the *form* section and add the word *required* as shown in bold in Listing 23-2.

Listing 23-2. Specifying the required fields for the attention of most modern browsers

```
<form action="form-handler.php" method="post">
<div>
<label class="label" for="username"><strong>Your Name</strong> ⚡
<span class="red">*</span>
<input id="username" name="username" size="30" required></label>
</div><br><br>
<div>
<label class="label" for="useremail"><strong>Your Email Address</strong> ⚡
<span class="red">*</span>
<input id="useremail" name="useremail" size="30" required></label>
</div><br><br>
<div>
<label class="label" for="userphone"><strong>Telephone Number </strong> ⚡
<span class="red">*</span>
<input id="userphone" name="userphone" size="30" required></label>
</div><br>
```

3. Save the *contact.php* file

The next section provides some fall-back messages for browsers that don't recognize the *required* attribute. It also includes some security checks and other messages that are not provided by either old or modern browsers.

Create the Message Pages for the Project

Use your text editor to create the following style sheet and message pages and save them in the folder *Chapter-23* either within the *htdocs* folder, or within the *web-tutorial* folder.

Create the Style Sheet for the *thank-you* Page and All the *error message* Pages

Create the style sheet as given in Listing 23-2. It styles all five message pages.

Save it as (*messages.css*)

Listing 23-2. Style the messages (*messages.css*)

```
body {text-align:center; font-size: large; font-weight:bold; background-color:#c9eCa0;}
span.red {color:red; font-size:xlarge; font-weight:bold;}
#back-button {margin:auto; text-align:center; width:200px; height:25px; padding:5px ~
background-color:red; color:white; font-size:105%; font-weight:bold;}
#back-button a {text-decoration:none; color:white;}
#back-button a:hover {color:aqua;}
```

Create the Thank-You Page

The “*thank-you*” page confirms to the user that the email was sent successfully. Try to include your navigation menu in the “*thank-you*” page (or at least a *Go Back* button). It would be a pity to lose your visitor. It would also be helpful to add a *Go Back* button to each error page. This page has a *Go Back* button with the text—“Return to Home Page”—but you can replace this button with your main navigation menu by using PHP *include* for the menus. The styling for the button is provided in the style sheet *messages.css*. The *thank-you* page is shown in Figure 23-5 and the code is in Listing 23-3.

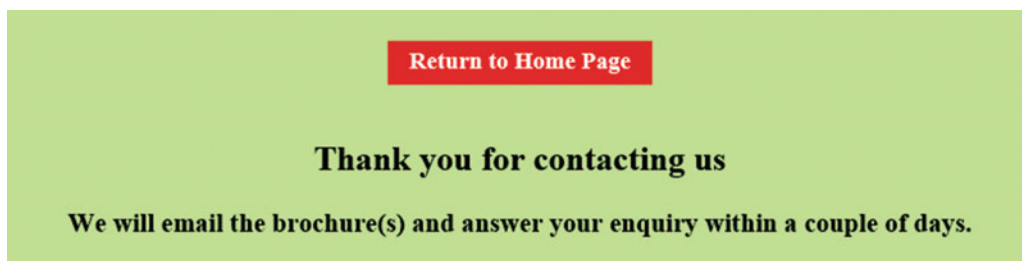


Figure 23-5. The *thank-you* page

Create the thank-you page (*thankyou.html*) using the code in Listing 23-3.

Listing 23-3. Create the *thankyou.html* page

```
<!doctype html>
<html>
<head>
<title>Thank-you for your message</title>
<meta charset=utf-8>
<link href="messages.css" rel="stylesheet" type="text/css">
</head>
<body>
<a id="top"></a><br>
<div id="back-button"><a title="Return to the Home page" href="index.php">Return to Home Page</a>
</div>
<div><br>
<h2>Thank-you for contacting us</h2>
<h3>We will email the brochure(s) and answer your enquiry within a couple of days.</h3>
</div>
</body>
</html>
```

Create the Missing Essentials Error Message (*error.html*) Using the Code in Listing 23-4

Listing 23-4. The missing essentials error message page

```
<!doctype html>
<html>
<head>
<title>Error message. Missing essentials</title>
<meta charset=utf-8>
<link href="messages.css" rel="stylesheet" type="text/css">
</head>
<body>
<p>One or more of the essential items in the form has not been filled in.</p>
<p>Essential items have a red asterisk like this <span class="red">*</span></p>
<p>Please click the Back button on your internet browser and then supply the
missing information.</p>
</body>
</html>
```

Create the Email Error Message Using the Code in Listing 23-5

Listing 23-5. Error message about unacceptable email pattern (*emailerr.html*)

```
<!doctype html>
<html>
<head>
<title>Email error message</title>
```

```

<meta charset=utf-8>
<link href="messages.css" rel="stylesheet" type="text/css">
</head>
<body>
<p>Your email address has an incorrect format.</p>
<p>Please click the Back button on your internet browser<br> and then correct ↩
your email address.</p>
</body>
</html>

```

Create a Check Box Error Message (*boxerror.html*) Using Code in Listing 23-6

I have included this in case you want the user to click at least one of the check boxes in your own adaption of the form and form-handler.

■ **Note** You will need to add a red asterisk to the contact form as shown bold in the following line of code:

```

<h3>Please email me the following brochure(s)<b><span class="red">*</span></b></h3>

```

Listing 23-6. Optional error message for check boxes

```

<!doctype html>
<html>
<head>
<title>Box error</title>
<meta charset=utf-8>
<link href="messages.css" rel="stylesheet" type="text/css">
</head>
<body>
<p>&nbsp;</p>
<p>Please tick one of the boxes to say which brochure or brochures you would like.</p>
<p>Essential items have a red asterisk like this<b><span class="red">*</span></b></p>
<p>Please click the Back button on your Internet browser and then supply the missing information.</p>
</body>
</html>

```

Create the Error Message to Prevent URLs Being Entered

Listing 23-7 provides the code for prevent the entering of URLs.

Listing 23-7. (*messagerror.html*)

```

<!doctype html>
<html>
<head>
<title>Error message. Do not enter URLs or scripts</title>
<meta charset=utf-8>

```

```
<link href="messages.css" rel="stylesheet" type="text/css">
</head>
<body>
Website addresses and scripts are not allowed</h2><br>This prevents low-life ↵
cyber vandals from inserting links which lead to dodgy websites.
<p>Please click the Back button on your internet browser<br>and then remove all ↵
website addresses and scripts from the form.</p>
</body>
</html>
```

Testing the Form-Handler and the Messages

If you are using the Apache server on your computer, access the *Chapter-23* folder **within the htdocs folder** and rename *form-handler-project.php* as *form-handler.php*. Start XAMPP and Apache and then access the contact form by entering <http://localhost/Chapter-23/contact.php> in the address field of a browser.

Then test the filters by entering unacceptable information ONE FIELD AT A TIME and then click the *Submit* button. Add a web address in the text fields and the message area. Leave an essential field empty. Enter an incorrectly formatted email address in the email field. Lastly, enter correct information in all the fields and click the *Submit* button.

You should see the *thank-you* page, but your email won't be sent because that is not possible using the server on your computer unless you configure the Mercury email client that comes with XAMPP.

To use the server on a remote host, open the *web-tutorial* folder and then open the folder *Chapter-23*. Rename *form-handler-typical.php* as *form-handler.php*. Open *form-handler.php* and change the initial blocks of code to match your website address and your email address. Upload the *Chapter-23* folder to the host. Access the *contact.php* file in a browser using your website address and the file *contact.php* (example: <http://www.mywebsite/Chapter-23/contact.php>). Click *Submit* to send yourself an email. Then test the filters by entering unacceptable information ONE FIELD AT A TIME and then click the *Submit* button. Leave an essential field empty. Add a web address in the text fields and the message area. Enter an incorrectly formatted email address in the email field. Lastly, enter correct information in all the fields and click the *Submit* button. You should see the *thank-you* page and receive the email via your email program.

■ **Tip** If you see a “Page not available” 404 message, the most likely reason is that you probably did not upload the *Thank-you* page.

Summary

In this chapter, you learned how emails can be sent from your website's *contact.php* page by means of a PHP form-handler. The form-handler code was explained and then you created a series of message pages. If the form was filled in correctly a *thank-you* page appeared and the email was sent to the owner of the website. If any fields were filled in incorrectly or omitted, an error message was displayed. The security filters in the form-handler checked the fields for malicious content and displayed a message to say that the malicious content would not be accepted. You were able to test the messages by entering completely correct or faulty information in the fields; however, you were unable to actually send the email because the form and the file-handler page must be located on a remote server to achieve this. XAMPP does contain a program called Mercury that would enable you to send and receive emails on your own computer, but that is beyond the scope of this book for absolute beginners.

In the next chapter you will learn how to add slide shows and videos to a web page.

CHAPTER 24



Add Slideshows and Videos

The first part of this chapter deals with the use and misuse of video and slideshows. The second part describes how to add videos and slideshows to a web page. The chapter does not contain a project, but it gives a full description of how to embed slideshows and videos within a web page. I have also provided examples so that you can view a video and a slideshow. The download folder for this chapter contains a fully worked website with 11 pages. Since the advent of HTML5, the task of incorporating videos has become much easier.

The chapter contains the following sections:

- Using other people's multimedia clips
- The benefits of using a slideshow or a video
- Good housekeeping tips
- Add a slideshow
- Adding video to a web page
- How to embed the HTML5 video code in a web page
- Using YouTube
- Summary

Using Other People's Multimedia Clips

Don't steal multimedia clips. You will need a genuine license (or permission from a non-commercial source) to incorporate other people's clips on your website. Owning a video tape, a CD, or DVD does not give you the right to use that music or video on your website. Downloaded clips cannot be used unless they are offered for free. So-called royalty-free clips sometimes have to be paid for.

Caution Some multimedia clips work on the Internet, but they fail or misbehave when tested on your own computer. Always try to upload and test on the Internet if you can.

The Benefits of Using a Slideshow or a Video

Great care should be taken not to use multimedia (slideshows, audio and video) gratuitously. Some website designers insert multimedia just because they can, and because it is fun. Creating and embedding multimedia can enliven a boring day for the web designer, but it may not please users or the client who commissioned the website. The following are some valid reasons for including multimedia in a website:

- A multimedia clip might explain something much better than text or images, provided there is no background music drowning the speaker's commentary.
- If your page provides instructions for writing code, or making, fitting, or repairing something, then a multimedia clip could be helpful. However, most users will not remember the instructions on the clip when they come to put them into practice (unless they have two computers: one to show the clip and the other to carry out the instructions). Preferably, provide a printable set of instructions in addition to the video.
- Multimedia can present a product or service, but be warned: a poorly produced clip will have a negative effect. Avoid inappropriate background music like the plague, as this is the biggest turn off. Why video and documentary makers so often drown the presenter's commentary is a mystery.
- A slideshow or a video is ideal for displaying hotel accommodations, real estate, or tourist attractions.
- Pop groups, folk bands, and choirs looking for fixtures can give a brief sample of their stage presentation and music style by using a video clip.
- As an example of the helpful use of multimedia, a website for bird watchers could incorporate a slideshow or video of birds and bird song.

What to Avoid

Videos can be intensely annoying if used at the wrong time or in the wrong place.

- **Strongly resist the temptation to put multimedia on the home page.** It will dictate the focus and distract the viewer's attention away from the more important items on the page, such as your navigation menu. A multimedia clip might cause a MIME type problem or a coding error that could prevent complete access to your home page.
- **Do not use auto start.** Auto start causes the clip to automatically begin when the page is loaded. It is particularly bad practice on the home page and a bad idea on any page. The sudden burst of sound can startle users, especially blind ones. Users will either immediately switch away from your website, or frantically search for a way to turn off the clip. They will probably never return to your site again.
- **Multimedia clips must have user controls.** Make sure the clip can only begin when the user clicks the start button on a control console. At the first hint of a thumping, disco-type background "music," the great majority of users will instantly abandon the site or hit the mute button.
- **Do not use onmouseover sounds or videos.** A blind or partially sighted person will receive a fright if she inadvertently mouses over an onmouseover video link. Even sighted persons can accidentally mouse over a sound link and they won't be pleased with the sudden burst of sound. For the user's sake, always use onmousedown or, preferably, create a control console for multimedia clips.

Good Housekeeping Tips

In previous chapters, when using PHP *includes*, the included files were in the same folder as the main files. This made it rather difficult to find files; in a larger website with many more page files the problem is magnified. In earlier chapters we put all the pictures in a folder called *images*. In this chapter, I have removed all the *include* files from the main folder and placed them in a folder named *includes*. I have put the video and slide files in their own folders named *video* and *slides* respectively.

In all the main pages, the code for the included files has changed from the following:

```
<?php include ("header.html"); ?>
<?php include ("nav.html"); ?>
<?php include ("vertnav.html"); ?>
<?php include ("footer.html"); ?>
```

to a new format in which the folder containing the file is specified (shown in **bold**) as follows:

```
<?php include ("includes/header.html"); ?>
<?php include ("includes/nav.html"); ?>
<?php include ("includes/vertnav.html"); ?>
<?php include ("includes/footer.html"); ?>
```

Add a Slideshow

Slideshows can sometimes be more informative than videos because the user is in total control of each step of the display. A slideshow with a control console can be paused to enable a slide containing text or a diagram to be examined for as long as the user wishes.

This section describes the simplest and most straightforward method from BarelyFitz Designs for creating web-embedded slideshows. I have added a control console to the original code. A search on the Internet will reveal several other methods, but they require a working knowledge of JavaScript or JQuery, or they involve an extra step using YouTube. Many tutorials fail to describe how and where to embed the code into a web page; our slideshow is already embedded in a web page.

The BarelyFitz Designs Slideshow

Patrick Fitzgerald of BarelyFitz Designs produced an excellent piece of free, open source JavaScript for slideshows. He also provides a very useful tutorial on his website:

<http://www.barelyfitz.com/projects/slideshow/>

■ **Caution** Do NOT download the JavaScript file from the BarelyFitz website. At the time of writing the download arrived together with dozens of Trojans. In the download for this chapter, I have provided a completely safe zipped copy of Patrick Fitzgerald's JavaScript file *slideshow.js*.

Following the usual procedure for this book, I would normally ask you to copy the files from the previous chapter and paste them into a folder named Chapter-24. However, because I have introduced so many new items in Chapter 24, you would be required to work on several lengthy projects. Therefore, to save you time and frustration, there are no projects; I ask you instead to please follow these steps:

1. If you have installed the XAMPP package, open the *htdocs* folder and create a new folder named *Chapter-24*.
2. If you have not installed XAMPP, and if you have an FTP program for logging-in to a remote host, you can use the remote host to test your files. In this case, create a folder in the *web-tutorial* folder named *Chapter-24*.
3. Download this chapter's zip folder *Chapter-24.zip* from the book's page at *apress.com*.
4. Unzip it in the folder *Chapter-24* within the *htdocs* folder (or in the folder *Chapter-24* in the *web-tutorial* folder if you intend to upload to a website).

Figure 24-1 shows the slideshow page.

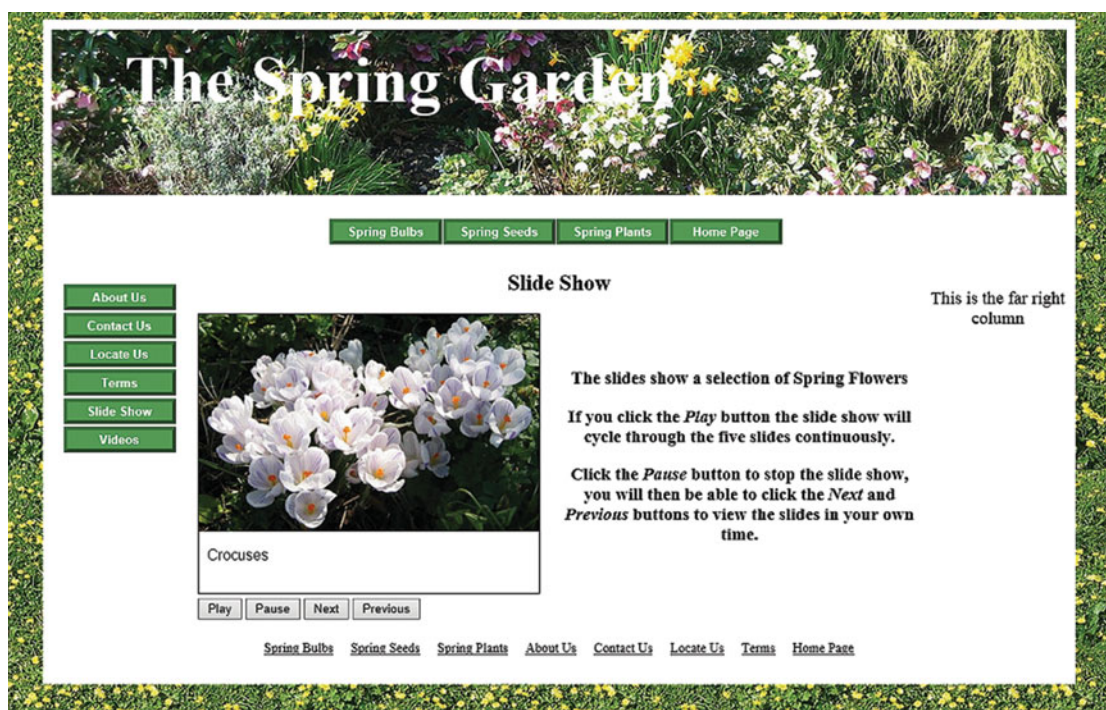


Figure 24-1. The slideshow. Note the console that gives the user complete control of the slideshow.

Before creating a slideshow you must produce a set of images (slides) identical in size and optimized so that they load quickly. The images in this slideshow are each 372 pixels wide by 304 pixels high and they have been optimized to a file size ranging from 53k to 77k.

The slideshow code is embedded in a new Spring Garden page named *slideshow.php*; the code is given in Listing 24-1. The file is a slideshow template that you can adapt for your own websites by changing the items in bold type.

Listing 24-1. The code for the page *slideshow.php*

```

<!DOCTYPE html>
<html>
  <head>
    <title>Spring Garden slideshow</title>
    <meta charset=utf-8>
    <meta name="description" content="Bulbs, crocuses, snowdrops, daffodils for the spring garden">

    <meta name="keywords" content=" bulbs, spring bulbs, spring garden bulbs">
    <link href="style-3d-gdn.css" rel="stylesheet" type="text/css">
      <!--[if lte IE 8]>
    <script src="html5.js">
    </script>
      <![endif]-->
    <!--THE JAVASCRIPT FILE FOR THE SLIDESHOW IS LOADED-->
      <script type="text/javascript" src="slideshow.js">
    </script>
    <script type="text/javascript">
      SLIDES = new slideshow("SLIDES");
    <!--FOR YOUR OWN SLIDESHOW , REPLACE THE SLIDE SHOWN IN BOLD TYPE WITH YOUR OWN SLIDE-->
    s = new slide();
    s.src = "slides/crocus-372-304.jpg";
    s.text = "Open link in same window";
    SLIDES.add_slide(s);
    <!--FOR YOUR OWN SLIDESHOW , REPLACE THE SLIDE SHOWN IN BOLD TYPE WITH YOUR OWN SLIDE-->
    s = new slide();
    s.src = "slides/daffs-372-304.jpg";
    s.text = "Open link in same window";
    SLIDES.add_slide(s);
    <!--FOR YOUR OWN SLIDESHOW , REPLACE THE SLIDE SHOWN IN BOLD TYPE WITH YOUR OWN SLIDE-->
    s = new slide();
    s.src = "slides/pansies-372-304.jpg";
    s.text = "Open link in same window";
    SLIDES.add_slide(s);
    <!--FOR YOUR OWN SLIDESHOW , REPLACE THE SLIDE SHOWN IN BOLD TYPE WITH YOUR OWN SLIDE-->
    s = new slide();
    s.src = "slides/bergenia-372-304.jpg";
    s.text = "Open link in same window";
    SLIDES.add_slide(s);
    <!--FOR YOUR OWN SLIDESHOW , REPLACE THE SLIDE SHOWN IN BOLD TYPE WITH YOUR OWN SLIDE-->
    s = new slide();
    s.src = "slides/hellebore-372-304.jpg";
    s.text = "Open link in same window";
    SLIDES.add_slide(s);
    </script>
  </head>
  <!--NEXT, THE SLIDESHOW IS SHOWN AND PAUSED AS SOON AS THE PAGE IS DISPLAYED-->
  <body onLoad="SLIDES.pause()">
    <!--THE FOLLOWING CODE IS COMMON TO MOST PAGES IN THE SPRING GARDEN WESITE-->
    <div id="wrapper">
      <?php include ("header.html"); ?>

```

```

<?php include ("nav.html"); ?>
<div id="content">
<div id="leftcol">
  <?php include ("vertnav.html"); ?>
</div>
<div id="rightcol">
  <p>This is the far right column</p>
</div>
<div id="midcol">
  <h2>Slideshow</h2>
<!--LOCATE THE SLIDESHOW IN THE MIDCOL-LEFT COLUMN. ENTER THE WIDTH AND HEIGHT-->
<div id="midcol-left">
<a href="javascript:SLIDES.hotlink()">

</a>
<script type="text/javascript">
</script>
<!--THE CONTROL CONSOLE IS NEXT. THE JAVASCRIPT COMMAND onClick IS INVOKED-->
<form>
<input type="button" value="Play" onClick="SLIDES.play()">
<input type="button" value="Pause" onClick="SLIDES.pause()">
<input type="button" value="Next" onClick="SLIDES.next()">
<input type="button" value="Previous" onClick="SLIDES.previous()">
</form>
<script type="text/javascript">
if (document.images)
{ SLIDES.set_image(document.images.SLIDESIMG);}
</script>
</div>
<div id="midcol-right">
<h3>&nbsp;</h3>
  <p>The slides show a selection of Spring Flowers</p>
<p>If you click the <em>Play</em> button the slideshow will cycle through the five ↩
  slides continuously./p>
<p>Click the <em>Pause</em> button to stop the slideshow, you will then be able to ↩
  click the <em>Next</em> and <em>Previous</em> buttons to view the slides in your own time</p>
</div>
</div><!--content div finishes here-->
  <br class="clear">
  <?php include ("footer.html"); ?>
</div>
</div><!--wrapper div finishes here-->
</body>
</html>

```

Test the Slideshow

If you are using XAMPP, start XAMPP and Apache, then test the slideshow by entering the following in the address field of a browser:

`http://localhost/Chapter-24/slide-show.php`

If you are using a remote host, upload the folder *Chapter-24* to your host, then test the slideshow by accessing the file *yourwebsite/slideshow.php*.

Other Ways of Creating a Slideshow

- Use the Animoto online slideshow creator
- Convert a PowerPoint slideshow into a video
- Use Windows Live Movie Maker
- Investigate the many alternative methods on the Internet

Adding Video to a Web Page

This section gives some background to the past difficulties of displaying videos on websites. It then explains how HTML5 has greatly simplified the process.

Yesterday's Video Formats

Until the advent of HTML5, video was a web designer's nightmare. With 14 video file formats, 4 popular media players, and 5 main browsers with 4 plug-ins, the best description would be "a dog's breakfast." The following list describes a small selection of the common video formats that were used before the advent of the HTML5 video tag:

- The *.flv* format is a Flash file from Adobe. It must be located within an *.swf* container. *swf* stands for Shockwave Flash.
- The *.mov* format developed by Apple is known as a QuickTime movie. The free QuickTime player has to be installed to play these movies on a Windows computer.
- The *.rm* or *.ram* format is produced by RealMedia for streaming online radio and Internet TV. Its low bandwidth requirement is good, but not top quality.
- The *.wma* format from Microsoft is very versatile and can be highly compressed. It can be tailored to any download speed. It can be any size and is used for streaming radio and video.
- The *.wmv* format from Microsoft is popular, but can only be played on Windows computers using the Windows Media Player.

Because of the many different file formats, the user needed several different video players. HTML5 has introduced a solution that eliminates the need for multiple players and the solution is acceptable to all modern browsers.

Today's Video Formats

The HTML5 video tag `<video>` uses only three video formats: a welcome reduction from 14 formats. However, if you analyze Table 24-1 you will see that all modern browsers will play videos if you use only two video formats, MP4 and Ogg.

Table 24-1.

Browser	MP4	WebM	Ogg
I.E. 9 and later	YES	NO	NO
Chrome	YES	YES	YES
Firefox	YES	YES	YES
Safari	YES	NO	NO
Opera	NO	YES	YES

The new `<video>` tag can be used right now provided you ignore Windows XP users who might be using Internet Explorer 8. When users eventually cease using IE 8, the HTML5 video tag will make life so much easier for users and web designers. The next snippet of code in Listing 24-2 demonstrates the simplicity of the markup when using the HTML5 video tag without a fall-back for IE 8. The video files are assumed to be in a folder named *video*. Each browser provides controls that are familiar to the user.

Listing 24-2. The basic HTML5 video code

```
<video width="420" height="315" controls="controls">
  <source src=video/somevideo.mp4 type='video/mp4;'>
  <source src=video/somevideo.ogv type='video/ogg;'>
  Your browser does not support the video tag.
</video>
```

Note that in the third line, the video is .ogv and the type is .ogg
Because the webM format is open source, it might eventually be adopted more widely; therefore you can insert the following optional line just below the mp4 line:

```
<source src=video/somevideo.webm type='video/webm;'>
```

■ **Note** In Listing 24-2, the line `<video width="420" height="315" controls="controls">` is important. When you produce the video files in various formats, they must have the same dimensions, and those dimensions must be included in the code; otherwise the videos may not play or they will appear distorted.

Explanation of the Code

```
<video width="420" height="315" controls="controls">
  <source src=video/somevideo.mp4 type='video/mp4;'>
  <source src=video/somevideo.ogv type='video/ogg;'>
  Your browser does not support the video tag.
</video>
```

The video will have a certain width and height; these details must be inserted in the first line.
The user’s browser reads the lines of code one-by-one until it comes to a line it understands, then it will then play the video file; the other lines will be ignored by the browser. Because Internet Explorer 8 can’t understand any of the lines, it will display the error message, “Your browser does not support the video tag.”
HTML5 video code is remarkably simple, but you will need some video file conversion programs to convert your video to the mp4 and ogg formats. Here are some suggestions:

Converting File Formats

- To convert between .flv and .swf, download the free FoxTab video converter from <http://www.foxtab.com> This is my favorite converter because it can convert almost any format.
- To convert files to .webm, .ogv, and .mp4, download the free Miro Video Converter at <http://mirovideoconverter.com>

Miro may eventually offer an iPad version of their converter.

The wagon video clips used in the following projects were kindly provided by the artist Roger Laughton and his daughter Helena of the Dolphin House Gallery, Colyton, Devon, United Kingdom. Their website is at <http://www.dolphinhousegallery.co.uk>

How to Embed the HTML5 Video Code in a Web Page

HTML5 introduced a much simplified way of displaying videos. Figure 24-2 shows a screenshot of a video using the new `<video>` tag embedded in a web page. The code given is in Listing 24-3.



Figure 24-2. Showing the HTML5 `<video>` tag displaying a video in Mozilla Firefox

The HTML5 simplified code for Figure 24-2 is embedded in a file named *video-page.php*

■ **Note** This file is included in the downloadable zip file; there is no need to create it.

Listing 24-3. The code for the file *video-page.php* with a YouTube fall-back for IE 8

```
<!DOCTYPE html>
<html>
  <head>
    <title>Video page. Spring Garden</title>
    <meta charset=utf-8>
    <meta name="description" content="Terms and conditions. Bulbs, seeds and plants for
the spring garden">
    <meta name="keywords" content="bulbs, spring bulbs, seeds, plants, spring garden bulbs">
    <link href="style-3d-gdn.css" rel="stylesheet" type="text/css">
    <style type="text/css">
      video {margin-left:200px;}
    </style>
    <!--[if lte IE 8]>
      <script src="html5.js">
      </script>
    <![endif]-->
  </head>
  <body>
    <div id="wrapper">
      <?php include ("includes/header.html"); ?>
      <?php include ("includes/nav.html"); ?>
    <div id="content">
    <div id="leftcol">
      <?php include ("includes/vertnav.html"); ?>
    </div>
      <div id="rightcol">
        <p>This is the far right column</p>
      </div>
    <div id="midcol">
      <h2>Demonstrating the HTML5 video tag combined with a YouTube fall-back</h2>
      <p>&nbsp;</p>
      <video width="420" height="315" controls="controls">
        <source src=" video/wagon3.mp4" type="video/mp4">
        <source src=" video/wagon3.webm" type="video/webm">
        <source src=" video/wagon3.ogv" type="video/ogg">
        <iframe width="420" height="315" src="http://www.youtube.com/embed/cUrSf-YDdyU"
frameborder="0" allowfullscreen></iframe>
      </video>
    </div>
    </div><!--content div finishes here-->
    <br class="clear">
    <?php include ("includes/footer.html"); ?>
  </div><!--wrapper div finishes here-->
</body>
</html>
```

■ **Tip** I recommend that you turn your speakers down before playing this file.

To play this file in a modern browser, enter the following URL in the address field:

`http://localhost/Chapter-24/video-page.php`

To embed a video into any website page, you simply place one of the following example codes within the body of the page. Substitute your own video files, which should be stored within a folder named *video*.

Listing 24-4. A version to cover all modern browsers, plus a YouTube fall-back for IE 8

```
<video width="420" height="315" controls="controls">
  <source src=" video/wagon3.mp4" type="video/mp4">
  <source src=" video/wagon3.webm" type="video/webm">
  <source src=" video/wagon3.ogv" type="video/ogg">
  <iframe width="420" height="315" src="http://www.youtube.com/embed/cUrSf-YDdyU"↵
    frameborder="0" allowfullscreen></iframe>
</video>
```

Listing 24-5. A version for all modern browsers with a warning for users of IE 8

```
<video width="420" height="315" controls="controls">
  <source src=" video/wagon3.mp4" type="video/mp4">
  <source src=" video/wagon3.ogv" type="video/ogg">
  your browser does not support the video tag.
</video>
```

Listing 24-6. A universal version for using YouTube only. This will work in all browsers

```
<iframe width="420" height="315" src="http://www.youtube.com/embed/cUrSf-YDdyU"↵
  frameborder="0" allowfullscreen>
</iframe>
```

Using YouTube

You can place almost any video format with YouTube, but you need to set up an account and then follow these instructions.

Signing up for a YouTube Account

If you have a Google account, use your Google details to log in (Google owns YouTube). Otherwise, on the YouTube home page, click *Create an Account* and fill in the details required. Give the email address of your client (or yourself if the video is for your website) so that your client (or you) can receive the email verifying the account details. You will be asked to invent a username and password. When you receive the email, click the verification link to activate your new account.

■ **Caution** You cannot upload a duplicate of something you have already uploaded; this applies even if you change the file name. At the time of writing, Google has made what was formerly a user-friendly site into a magical mystery tour. Also be aware that Google/YouTube tends to change things occasionally, so the following procedure might go out of date.

At the time of writing, the most irritating omission is that all the configuration tricks have vanished from the YouTube website. This omission may now have been rectified.

To Host a Video with YouTube

I have already uploaded a video to YouTube for this project. The following instruction will enable you to upload a video of your own at some time in the future.

Log in at <http://www.youtube.com>, click the *Upload* item on the top menu. After clicking *Upload* on the top menu bar, click the button labeled *Select files* from your computer. A small window will allow you to navigate to where your video is stored on your hard drive. Click the video and then click the *Open* button. On the right of the next screen, select *Unlisted*, anyone with the link can view. After a while the video will have finished uploading and you will see a message saying that the upload of your video is complete. Click the *Embed* button. A small window will show the code highlighted in blue. Copy and paste the code into your web page.

A client may ask you to embed a video that he or she has chosen. The client will give you the URL for that video. Enter the URL in the address field of a browser, and when the video is displayed, click the word *Share*. Below that you will see the word *Embed*, click that word and you will see the code for pasting into a web page. Copy that code (Ctrl+C) and paste it into your web page. The code places the video's URL inside `<iframe>` tags and it will have the following format:

```
<iframe width="420" height="315" src="http://www.youtube.com/embed/cUrSf-YDdyU" ↵
frameborder="0" allowfullscreen></iframe>
```

To embed it into a page, see Listing 24-4 and Listing 24-6. The `<iframe>` tag specifies an inline frame, which is the only type of frame allowed in HTML5. *iframes* are used to embed another document or video within the current HTML document. An *iframe* behaves as if it is another browser window embedded inside your web page. The URL inside the `iframe` links to an external source such as the YouTube server. You can include many *iframes* in an HTML5 page.

Use a CSS style to position the video on the page; in our example the style is this:

```
<style type="text/css">
  video {margin-left:200px;}
</style>
```

■ **Caution** YouTube code sometimes omits the item `http://`, or `https://` be sure to add it to the embedded code.

Summary

In this chapter you learned how to add slideshows and videos to a web page. A fall-back was described that enabled videos to be viewed in Internet Explorer 8. In the next chapter you will learn how to create a tabbed navigation system.

CHAPTER 25



Create a Tab Menu

Tab menus are sometimes used in smaller websites with few pages, say a maximum of 12 pages. Tab menus are not particularly suitable for large sites because of the limited number of tabs that can be placed across a page. In this chapter you will view a ready-made website with six tabs so that you can see how the tabs work. Because people learn by doing and not by reading, you will be asked to add two more pages and two more tabs.

The tab menu in this tutorial is my adaptation of Joshua Kaufman's CSS tab menus at:

http://unraveled.com/publications/assets/css_tabs/

The menus are licensed under <http://creativecommons.org/licenses/by/3.0/> and are free to use and to adapt for your own websites.

This chapter contains the following sections:

- Examine the code for the home page
- Examine the code for the CSS file
- Create two new tabs
- Create two new pages
- Summary

Note I have greatly simplified the project so that the technique for tab menus is clearly demonstrated without surrounding clutter. No PHP *includes* are used and the pages are very basic.

We will need a new folder to contain this chapter, so please follow these steps:

1. We will not be using PHP *includes* in this chapter so do not create the new folder in the *htdocs* folder. Create a folder in the *web-tutorial* folder named *Chapter-25*.
2. Download and unzip the *Chapter-25.zip* file in your new folder *Chapter-25*.
3. Right-click the *index.html* file and choose to open it with a browser.
4. Click each tab in turn to see how the menu tab appears to become part of the selected page.
5. Note how the tab widths vary automatically to accommodate the number of characters in the labels.

The home page is shown in Figure 25-1.



Figure 25-1. The selected tab indicates clearly which page is currently being viewed

The selected tab blends in with the page content and appears to be part of it. Note how the *Page Three* tab has automatically stretched to accommodate the extra characters. The six tabs link to the website’s six pages; later you will be asked to increase the website to eight pages and the menu to eight tabs.

Figure 25-2 shows that page four has been selected.

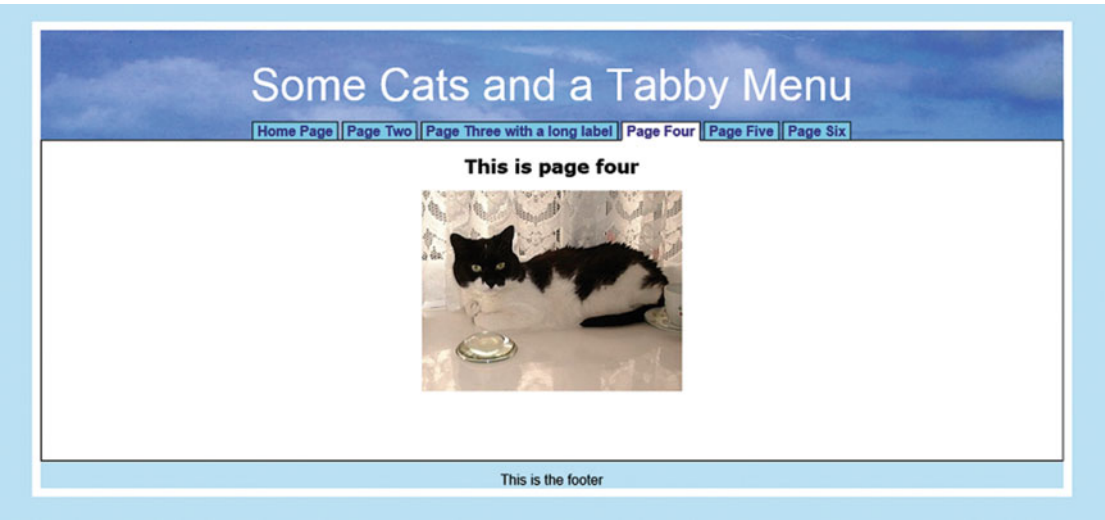


Figure 25-2. Page four has been selected

Examine the Code for the Home Page

The code for the downloaded home page is given in Listing 25-1.

Listing 25-1. The code for a page with a six-tab menu (*index.html*)

```
<!doctype html>
<html>
<head>
<title>The home page. Square tabs</title>
<meta charset=utf-8>
<link rel="stylesheet" type="text/css" href="tabs.css">
<!--Add conditional JavaScript for IE8-->
<!--[if lte IE 8]>
<script src="html5.js" type="text/javascript">
</script>
<![endif]-->
</head>
<body>
<div id="wrapper">
<div id="container">
<header>
    <h1>Some Cats and a Tabby Menu</h1>
</header>
<nav>
    <ul>
        <li><a class="active" href="index.html">Home Page</a></li>
        <li><a href="page2.html">Page Two</a></li>
        <li><a href="page3.html">Page Three with a long label</a></li>
        <li><a href="page4.html">Page Four</a></li>
        <li><a href="page5.html">Page Five</a></li>
        <li><a href="page6.html">Page Six</a></li>
    </ul>
</nav>
<div id="content">
<h2>This is the home page</h2>
    <p>&nbsp;</p>
<p class="img"></p><p>&nbsp;</p><p>&nbsp;</p><p>&nbsp;</p><p>&nbsp;</p>
</div>
<footer>
This is the footer
</footer>
</div></div>
</body>
</html>
```

Explanation of the Code

The majority of the code will be familiar to you, but the menu code for the pages and tabs contains a new attribute **id="active"** shown in bold type in the following menu snippet:

```
<nav>
  <ul>
    <li><a id="active" href="index.html">Home Page</a></li>
    <li><a href="page2.html">Page Two</a></li>
    <li><a href="page3.html">Page Three with a long label</a></li>
    <li><a href="page4.html">Page Four</a></li>
    <li><a href="page5.html">Page Five</a></li>
    <li><a href="page6.html">Page Six</a></li>
  </ul>
</nav>
```

Clicking the first tab loads the page *index.html*. Note that the first tab is set to *active*. The associated CSS file works the magic that causes the tab to blend into the content of the home page. We will examine the CSS file later. Meanwhile we will discuss the menu on page four to see the change that causes the page four tab to blend with the content of *page4.html*.

Listing 25-2 shows in bold type how the menu differs on page four.

Listing 25-2. The menu in page four

```
<nav>
  <ul>
    <li><a href="index.html">Home Page</a></li>
    <li><a href="page2.html">Page Two</a></li>
    <li><a href="page3.html">Page Three with a long label</a></li>
    <li><a id="active" href="page4.html">Page Four</a></li>
    <li><a href="page5.html">Page Five</a></li>
    <li><a href="page6.html">Page Six</a></li>
  </ul>
</nav>
```

Note that the *active* identity now relates to the link to *page4.html*. That is how we apply the tab mechanism to each page; on page four the active *id* is applied to the link to page four. On page two the active class is applied to the link to page two and so on.

We cannot use the PHP *include* for the menus because the menu is slightly different on every page. However, this is not a serious disadvantage because tab menus usually apply to websites with few pages.

Other Items Also Change from Page to Page

Obviously each page will have its own content and images; also the title must match the page. For instance, in *page4.html* the title, heading, and content are also related to the page as shown in bold type in the following two extracts from page four's code.

Listing 25-3.

```

<!doctype html>
<html>
<head>
<title>Page four. Square tabs</title>

...

    <nav>
<ul>
    <li><a href="index.html">Home Page</a></li>
    <li><a href="page2.html">Page Two</a></li>
    <li><a href="page3.html">Page Three with a long label</a></li>
    <li><a id="active" href="page4.html">Page Four</a></li>
    <li><a href="page5.html">Page Five</a></li>
    <li><a href="page6.html">Page Six</a></li>
</ul>
</nav>
<div id="content">
<h2>This is page four</h2>
<br>
<p class="img">
</p>

```

To summarize, the following items are unique to each page: the HTML *page title*, the *page heading*, the *active menu link*, and the *picture*.

The menu is displayed as a row of active tabs by the CSS file provided in the downloadable zip file. The code is given in Listing 25-4.

Examine the Code for the CSS File

Listing 25-4. The CSS file for creating and positioning a row of tabs (*tabs.css*)

```

/* SET STYLES TO EQUALISE BROWSER RENDITION*/
html, body, h1, h2, h3, h4, h5, h6, p, ol, ul, li {padding:0; margin:0;}
header, footer, section, article, nav {display:block;}
/*SET THE GENERAL STYLES FOR THE PAGES*/
body {text-align:center; margin:20px; background:#bee8ff; font:medium arial;}
#wrapper {margin:auto; max-width:1180px; min-width:960px; border:10px white solid;}
#container {max-width:1180px; min-width:960px; margin:-10px auto 0 auto;}
h2 {text-align:center;}
p,img {text-align:center;}
header {margin-top:10px; height:146px; width:100%; ↵
background-image:url('images/bluepan-2.jpg'); background-repeat:no-repeat; ↵
                background-position:center; border-bottom:0;}
h1 {font-family:Arial; font-size:320%; color:white; font-weight:normal; ↵
    text-align:center; padding-top:35px; margin-bottom:-10px; height: 94px; margin:auto;}
/*POSITION THE MENU ON THE PAGE*/
nav {position:relative; top:-13px; border-bottom:1px solid black; margin:-25px 0 0 0;}

```

```

/* USE DISPLAY:INLINE TO DISPLAY THE MENU AS A ROW OF TABS*/
nav li {display: inline; overflow: hidden; list-style-type: none;}
/*SET THE COLOURS FOR THE NON-ACTIVE TABS. SPECIFY THE FONT USED IN THE TABS*/
nav a, a#active {color:navy; background:#81d9f6; font:bold 1em Arial; ↵
    border:1px solid black; padding:2px 5px 0px 5px; margin:0; text-decoration:none;}
/*CREATE A 3PX WIDE BOTTOM BORDER ON TAB THE SAME COLOUR AS THE CONTENT PANEL*/
nav a#active {background:white; color:navy; border-bottom:3px solid white}
/*HOVERING OVER A NON-ACTIVE TAB CREATES AN AQUA TAB*/
nav a:hover {background:aqua;}
/*MOVE THE WHITE CONTENT PANEL CLOSE UP TO THE ROW OF TABS*/
#content {margin-top:0; font:arial; position:relative; top:-13px; text-align:left; ↵
    background:white; padding:20px; border:1px solid black; border-top:none;}

```

Explanation of the Code

You will have encountered most of the code in earlier chapters. Also, explanations are provided by the comments in the above Listing. The following explanation will therefore be confined to the last five blocks of code that deal with the tab menu.

```

/* USE DISPLAY:INLINE TO DISPLAY THE MENU AS A ROW OF TABS*/
nav li {display:inline; overflow:hidden; list-style-type:none;}

```

Without this line the tabs would display in a block stacked on top of each other.

```

/*SET THE COLOURS FOR THE NON-ACTIVE TABS. SPECIFY THE FONT USED IN THE TABS*/
nav a {color:navy; background:#81d9f6; font:bold 1em Arial;
border:1px solid black; padding:2px 5px 0px 5px; margin:0; text-decoration:none;}

```

The tabs are set to a mid-blue background with a black border and Arial font.

```

/*CREATE A 3 PIXEL WIDE BOTTOM BORDER ON TAB THE SAME COLOUR AS THE CONTENT PANEL*/
nav a#active {background:white; color:navy; border-bottom:3px solid white}

```

This causes the active tab to appear attached to the content panel of the page.

```

/*HOVERING OVER A NON-ACTIVE TAB CREATES AN AQUA TAB*/
nav a:hover {background:aqua;}

```

The color of the non-active tabs changes to Aqua when the cursor hovers over them.

```

/*MOVE THE WHITE CONTENT PANEL CLOSE UP TO THE ROW OF TABS*/
#content {margin-top:0; font:arial; position:relative; top:-13px; text-align:left; ↵
    background:white; padding:20px; border:1px solid black; border-top:none;}

```

This code ensures that the white content panel butts up against the menu.

Create Two New Tabs

To create the two new tabs please follow these steps:

1. Open *index.html* in your text editor.
2. Add additional links to pages 7 and 8 to the *end* of the menu links as shown in bold type in the following snippet of code:

```
<nav>
  <ul>
    <li><a id="active" href="index.html">Home Page</a></li>
    <li><a href="page2.html">Page Two</a></li>
    <li><a href="page3.html">Page Three with a long label</a></li>
    <li><a href="page4.html">Page Four</a></li>
    <li><a href="page5.html">Page Five</a></li>
    <li><a href="page6.html">Page Six</a></li>
    <li><a href="page7.html">Page Seven</a></li>
    <li><a href="page8.html">Page Eight</a></li>
  </ul>
</nav>
```

3. Save the file.
4. In your HTML text editor, add the code for the two additional links to the menus in each of the other five pages.
5. Save each changed page.

Create Two New Pages

We will follow our usual practice for creating additional pages; we will use *Save as* to save an existing page with a new name.

Open *index.html* in your HTML text editor. Use *Save as* to make a copy named *page7.html*.

Then using *Save As*, make a copy of *page7.html* and name it *page8.html*

You now have two new pages, but they each have four items that need changing to match the fact they are new pages:

Please follow these steps:

1. In *page7.html*, remove the **id="active"** from the first menu link, then add the **id="active"** attribute to the page seven link as follows: **< a id="active" href="page7.html">Page Seven**.
2. Change the page title to `<title>Page 7</title>`.
3. Change the `<h2>` heading to `<h2>Page Seven</h2>`.
4. Change the pictures file name to *cat7.jpg*.
5. In *page8.html*, remove the **id="active"** from the first menu link, then add the **id="active"** attribute to the page eight link as follows: **< a id="active" href="page8.html">Page Eight**.

6. Change the page title to <title>Page 8</title>.
7. Change the <h2> heading to <h2>Page Eight</h2>.
8. Change the pictures file name to *cat8.jpg*.

In your Chapter 25 folder, double-click the *index.html* file and view all eight pages, you should see eight active tabs in all the pages. Three of the pages are shown in Figures 25-3, 25-4, and 25-5.



Figure 25-3. The home page with eight tabs



Figure 25-4. The Page Seven tab has been clicked

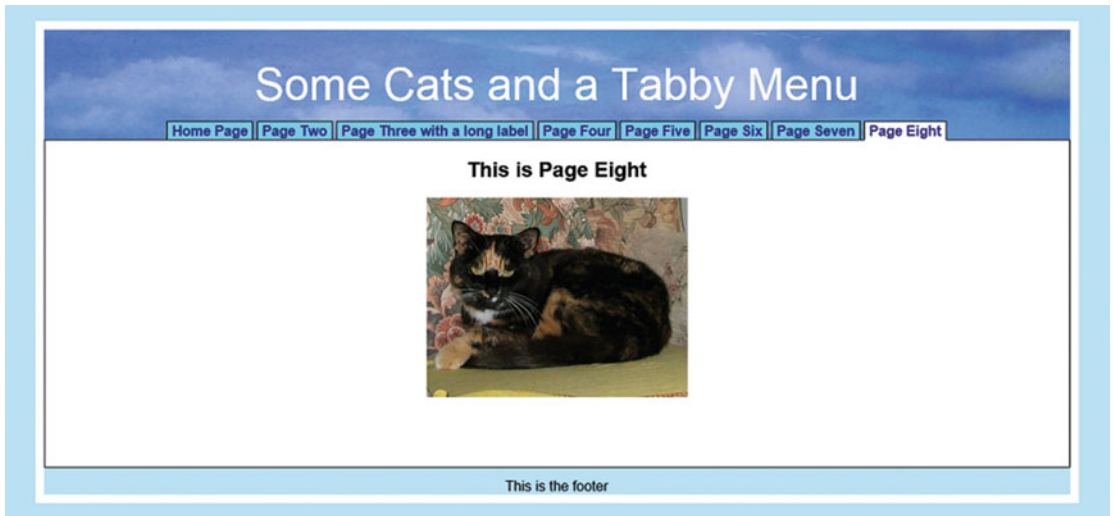


Figure 25-5. *The Page Eight tab has been clicked*

Summary

In this chapter you learned how to create a tab menu and how to add extra tabs and additional pages.

In the next chapter you will learn how to create a tab menu with drop-down submenus. The problems inherent in this method of navigation will be explained and I will describe a circumstance in which a drop-down tab menu might be less daunting to users.

CHAPTER 26



Designing a Drop-Down Menu

In the previous chapter we explored tab menus; tabs can also be created as drop-down menus. However, in the past, I would never use drop-down menus in my own websites and I used to advise my clients to avoid them. I could not see the point of hiding page links behind drop-down menus. Since the advent of mobile phones capable of viewing websites, I have been forced to change my mind because drop-down menus take up less space on the tiny screens of mobile devices. Drop-down menus for mobile devices are described in Chapters 35 and 38 of this book.

This chapter contains the following sections:

- Why drop-down menus may not suit all types of websites
- Planning the menu
- Showing and hiding the drop-down submenus
- Create a new page to see how the drop-down menu works
- Create the publications page
- Summary

Why Drop-Down Menus may not Suit all Types of Websites

My reasons for not using drop-down menus are as follows:

- Hiding navigation from the user is unfriendly and unhelpful, and it contravenes one of the cardinal rules for good website design, that is, navigation must be prominent and self-explanatory. Users should not have to hunt for a way to obtain information from the site.
- When teaching IT to people who had used Word for a year or two, I was surprised that the majority had never explored the drop-down menu bar at the top of the interface. Instead, they always used the icons on the Toolbar. Software producers now recognize this, and drop-down menus are being rapidly replaced by icons in ribbon menus; as a result, drop-down menus are increasingly regarded as old fashioned.
- If a JavaScript drop-down menu is used, the JavaScript prevents search engines from detecting pages other than the home page. The search engine thinks there is only one page on the site and so gives the site a low rating. Most paint-by-numbers programs such as WordPress use JavaScript drop-down menus. The drop-down menu in this chapter does not use JavaScript.

- Drop-down menus are a problem for the visually impaired.
- People with a hand tremor or a physical disability have difficulty using a mouse to locate items on the submenus.

However, if the target audience is computer ‘savvy’ they will not be afraid of exploring and using drop-down menus. Therefore drop-down menus would be acceptable for websites on technical matters such as programming techniques, and for smartphones where menus must occupy the minimum of screen space.

The project in this chapter is suitable for a website that teaches basic website design. So that search engines can locate all the website’s pages, the technique described in this chapter does not employ JavaScript; the project uses only HTML and CSS, so therefore it is search engine friendly.

The tabs and the drop-down lists must be carefully planned. The items in the drop-down list must be closely related to the name on the tab. This is important; users can be very annoyed by drop-down menus that put sub-list items under inappropriate tabs.

Planning the Menu

Planning means deciding which items will be top-level (the tab labels) and which items you wish to hide from the user. The main items will form the visible top-level list (the tabs in this project). The hidden items will be sublists. This is demonstrated in the following plan for this chapter’s project:

- **1st visible item (the tab labeled *About Us*)**
 - Hidden item 1 (*About Us*)
 - Hidden item 2 (*Meet the Team*)
- **2nd visible item (the tab labeled *General*)**
 - Hidden item 1 (*Structure*)
 - Hidden item 2 (*File Format*)
 - Hidden item 3 (*Browser*)
- **3rd visible item (the tab labeled *HTML & CSS*)**
 - Hidden item 1 (*Hyperlinks*)
 - Hidden item 2 (*CSS*)
 - Hidden item 3 (*HTML Examples*)
 - Hidden item 4 (*Link CSS to the HTML*)
 - Hidden item 5 (*CSS Examples*)
- **4th visible item (the tab labeled *PHP*)**
 - Hidden item 1 (*What is PHP*)
 - Hidden item 2 (*PHP ‘includes’*)
 - Hidden item 3 (*PHP for Databases*)
- **5th visible item (the tab labeled *Resources*), and so on**

Having planned the list of items we can then use HTML and CSS to produce the pages.

The technique described in this section was published by Harry Roberts who kindly gave me permission to adapt his method for this project. His website <http://csswizardry.com> is well worth visiting as it contains a wealth of useful resources.

In this project the items are real-world examples for creating a menu, as shown in Figure 26-1.

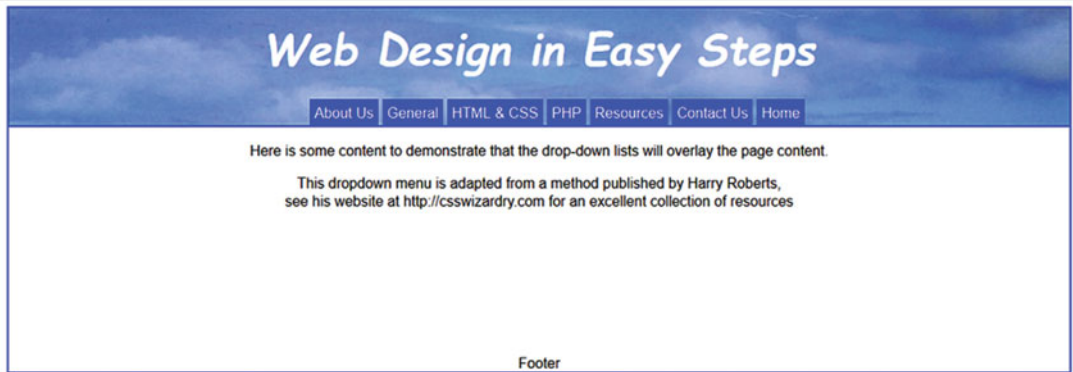


Figure 26-1. The home page showing the drop-down menu tabs used for this project

The visible top-level items provide the labels for the tabs.

For simplicity and to show the drop-down technique without complication, no PHP *includes* will be used. Therefore the project can be loaded into a chapter within the *web-tutorial* folder and not into the *htdocs* folder. The code for the home page and the CSS file is provided in the downloadable zip file.

This website works well in all browsers including Internet Explorer 8.

To create a folder for this chapter's downloadable files, please follow these steps:

1. In the *web-tutorial* folder (not the *htdocs* folder), create a folder named *Chapter-26*.
2. Download the *Chapter-26.zip* file and unzip it in your folder *Chapter-26*.
3. In the folder *Chapter-26*, right-click the file *index.html* and try hovering over the tabs, but don't click any of the submenus because, at the moment, we have no pages that link to the items on the submenus.

Figure 26-2 shows what happens when the cursor is hovered over the *General* tab and then moved down to hover on one of the submenus.



Figure 26-2. Hovering over a tab (or clicking it) reveals its drop-down list

Note how the drop-down overlays the page content. Hovering over a particular submenu item on the drop-down list changes that item to a different color. Clicking on that item would access a page if it existed. Later, I will ask you to create a page that can be accessed from an item on the resources drop-down submenu. In our project only the home page currently exists. Six tabs have drop-down menus but the Home tab does not. This is because it relates to one page only; it is the only tab that accesses a page directly.

Nested unordered list tags `` and `` are used to provide the submenus. Semantic tags have not been used to avoid the need for the JavaScript fix for IE8. You can of course include the *html5.js* script if you prefer to use HTML5 semantic tags. The HTML5 code for Figures 26-1 and 26-2 is shown in Listing 26-1. Naturally, in the real world the `href="#"` tags would be replaced by the URLs for the various pages.

Listing 26-1. Creating the drop-down menu (*index.html*).

```
<!doctype html>
<html>
<head>
<title>Home page for a CSS drop-down menu</title>
<meta charset=utf-8>
<link rel="stylesheet" type="text/css" href="dd-style.css">
</head>
<body>
<div id="container">
<div id="header">
<h1>Web Design in Easy Steps</h1>
  <ul id="nav">
    <li>
      <a href="#" title="About Us">About Us</a>
      <ul>
        <li><a href="#">About Us</a></li>
        <li><a href="#">Meet Our Staff</a></li>
      </ul>
    </li>
    <li>
      <a href="#" title="General introduction">General</a>
      <ul>
        <li><a href="#">Structure</a></li>
        <li><a href="#">File Format</a></li>
        <li><a href="#">Browser</a></li>
      </ul>
    </li>
    <li>
      <a href="#" title="HTML & CSS">HTML & CSS</a>
      <ul>
        <li><a href="#">HTML Code</a></li>
        <li><a href="#">Hyperlinks</a></li>
        <li><a href="#">CSS</a></li>
        <li><a href="#">HTML Examples</a></li>
        <li><a href="#">CSS Examples</a></li>
        <li><a href="#">Link CSS & HTML</a></li>
      </ul>
    </li>
    <li>
      <a href="#" title="PHP">PHP</a>
```

```

        <ul>
            <li><a href="#">What is PHP</a></li>
            <li><a href="#">PHP "includes"</a></li>
            <li><a href="#">Interactive PHP</a></li>
            <li><a href="#">PHP for Databases</a></li>
        </ul>
    </li>
    <li>
        <a href="#" title="Resources">Resources</a>
        <ul>
            <li><a href="#">Free Tools</a></li>
            <li><a href="#">Publications</a></li>
            <li><a href="#">Websites</a></li>
            <li><a href="#">Templates</a></li>
            <li><a href="#">Forums</a></li>
        </ul>
    </li>
    <li>
        <a href="#" title="Contact us">Contact Us</a>
        <ul>
            <li><a href="#">Contact details</a></li>
            <li><a href="#">Find us</a></li>
        </ul>
    </li>
    <li>
        <a href="index.html" title="Return to Home Page">Home</a>
    </li>
</ul>
</div>
<p>Here is some content to demonstrate that the drop-down lists will overlay
the page content.</p>
<p>This drop-down menu is adapted from a method published by Harry Roberts,
<br>see his website at http://csswizardry.com for an excellent collection of
resources</p>
<p>&nbsp;</p><p>&nbsp;</p><p>&nbsp;</p><p>&nbsp;</p><p>&nbsp;</p>
<div id="footer">
Footer
</div>
</div>
</body>
</html>

```

Explanation of the Code

You have used most of this code before, but I will explain the new drop-down feature:

```

<h1>Web Design in Easy Steps</h1>
<ul id="nav">
    <li>
        <a href="#" title="About Us">About Us</a>
    </li>

```

```

        <li><a href="#">About Us</a></li>
        <li><a href="#">Meet Our Staff</a></li>
    </ul>
</li>
<!--The six other menu tabs follow the same pattern as this-->
...
</ul>

```

The line `About Us` displays the label on the tab. In this example the topic *About Us* has been split between two pages; these are *About Us* and *Meet our Staff*. The **link href="#"** is a deliberate dead link, so that, if the user clicks the tab, the dead link prevents the user from going directly to the *About Us* page. Instead, if the user hovers on the tab or clicks it, the drop-down list will appear allowing the user to choose which page to access.

I have used the dead link code **href="#"** for the two pages *About Us* and *Meet our Staff* because, apart from the Home page we have no other pages to link to.

Note how the `` `` sub link tags are nested within the surrounding `` `` tags.

The entire tab menu is surrounded by a pair of `` `` tags.

The sublinks are enclosed by their own `` `` tags.

This drop-down menu technique relies on the nesting of menu items within other menu items.

The next snippet provides the code that links to the only page that currently exists.

```

<li>
    <a href="index.html" title="Return to Home Page">Home</a>
</li>

```

The Home page tab is a straightforward tab with no drop-down submenu for the obvious reason that there is only one Home page. Therefore that tab links directly to the Home page.

Showing and Hiding the Drop-Down Submenus

To show or hide the submenu items, we use the CSS file *dd-style.css*, which is included in the downloaded zip file. The code for this is given in Listing 26-2.

Listing 26-2. The CSS style sheet code (*dd-style.css*).

```

/*Set the general style for all the pages*/
body{text-align:center; font-family:Arial; font-size:medium; font-weight:normal;
background:white; color:black;}
#container {max-width:1180px; min-width:960px; margin:auto; background-color:white;
color:black;border:2px blue solid; text-align:center;}
#header {padding-top:10px; width:100%; text-align:center; height:120px; margin:auto;
background-image:url('images/bluepan-2.jpg'); background-repeat:no-repeat;
position:relative;border-bottom:2px blue solid;}
h1 {color:white; font-family: cursive; font-style: italic; font-size:300%;
text-align:center; margin:0 auto 0 auto;}
#footer {clear:both; text-align:center; font-size:small;}

/*Position the tab menu on the pages and set the general tab style*/
#nav {position:absolute; top:86px; left:23%; list-style:none; font-weight:normal;
width:700px; margin-bottom:0;text-align:left;}
#nav li {float:left; margin-right:5px; position:relative;}

```

```
#nav a {display:block; padding:5px; color:#fff; background:blue; text-decoration:none;}
#nav a:hover {color:#fff; background:#060; text-decoration:none;}

/*Set the drop-down styles*/
/*The background is made transparent by the final zero, otherwise a white patch spoils ↵
the drop-down effect. The code left:-9999px; hides the drop-down lists off-screen when ↵
it is not needed (alternatively you could use display:none;)*
#nav ul {background:rgba(255,255,255,0); list-style:none; position:absolute; left:-9999px;}
#nav ul li {border-top:1px blue solid; float:none;}
/*Prevent text from wrapping*/
#nav ul a {white-space:nowrap;}
/*Display the drop-down list when hovering over a tab and bring back on-screen when needed*/
#nav li:hover ul {padding-left:0; padding-right:0; left:0;}
/*Make the top-most link stay 'hovered' even when the cursor moves down the list*/
#nav li:hover a {background:#7adff8; color:black; text-decoration:none;}
/*Specify what happens when the user hovers over each individual link in the drop-down lists*/
#nav li:hover ul li a:hover {background:#868; color:white;}
```

Explanation of the Code

The first block of code defines the general style of the website's pages, that is, color, font, header, and footer.

The second block positions the menu on the page and sets the general tab style.

The code **#fff**; is the hex code for white; you could use `color:white`; Similarly, you could use the code `background:aqua`; instead of `background:#060`;

The other blocks of code have been explained by comments in the listing.

Create a New Page to See how the Drop-Down Menu Works

We will now modify the home page so that we can test the drop-down tabs; please follow these steps:

1. Open the folder *Chapter-26*.
2. Open the file *index.html* in the Code/Source pane of your HTML text editor.
3. Locate the following lines:

```
<li>
  <a href="#" title="Resources">Resources</a>
  <ul>
    <li><a href="#">Free Tools</a></li>
    <bli><a href="#">Publications</a></b>
    <li><a href="#">Websites</a></li>
    <li><a href="#">Templates</a></li>
    <li><a href="#">Forums</a></li>
  </ul>
</li>
```

4. Replace the hash (#) in the line `Publications` with a link to the publications page shown in bold type as follows:

```
<li><a href="publications.html">Publications</a></li>
```

5. Save the file.
6. Use *Save As* to save the file *index.html* with the new name *publications.html*. You now have a publications page that you can access from a drop-down menu. However it is a replica of the home page so we will add some more appropriate content to the publications page. Leave the page *publications.html* on the screen of your text editor ready for the next steps.

Create the Publications Page

We will now modify the publications page so that we can test the drop-down tabs; please follow these steps:

1. We will now create two columns in the content area of the page *publications.html*.

Find the following lines:

```
<li>
  <a href="index.html" title="Return to Home Page">Home</a>
</li>
</ul>
</div>
```

Immediately below the `</div>` insert the following code:

```
<div id="left-col">
  <br>Practical HTML5 Projects by Adrian W West<br>Published by Apress.com <br><br>
  <br><br>
  <p class="lft">The title is rather misleading, the book is actually a collection of ↵
  useful website tips and tricks that a designers can adapt for their own use</p>
</div>
<div id="right-col">
  <br>Practical PHP and MySQL Website Databases<br>Published by Apress.com ↵
  December 2013<br><br>
  
<br><br>
  <p class="lft">A project based book that demystifies interactive database-driven ↵
  websites. In the first two chapters you will set up your free development and ↵
  testing environment and build your first PHP and MySQL database-driven website.
</div>
```

2. Save the file.

To view the publications page: In the folder *Chapter-26*, view the *index.html* page in a browser and hover over the *Resources* tab, then slide the cursor down the drop-down menu and hover over the *Publications* item. Click the *Publications* item and you should see the page illustrated in Figure 26-3.



Figure 26-3. The publications page

Summary

In this chapter you learned how to create a drop-down tab menu without using complex JavaScript. The method described used only HTML5 and CSS. You created a new page so that you could test the drop-down tab menu. In the next chapter you will learn how to create drop shadows using CSS.

CHAPTER 27



Drop Shadows

Drop shadows can be created in many ways. Most graphics programs can create drop shadows for images, but with the advent of CSS3, the process is greatly simplified.

My book *Practical HTML5 Projects* (apress.com) covers over a dozen techniques for shadows around images and web pages, and many of those will work with Internet Explorer 8. This chapter will deal only with drop shadows for modern versions of Internet Explorer (IE 9 or later), and of course with all the other modern browsers that are compatible with CSS3.

CSS3 introduced many useful features and is backwardly compatible with CSS2. It does not replace CSS2 but it supplements CSS2 by adding several new modules.

This chapter contains the following sections:

- Drop shadows for images and websites with sharp corners
- Drop shadows for a website with rounded corners
- Add a drop shadow and a white border to an image
- Add a drop shadow to text
- Shadows on four sides of an HTML element
- Create a four-sided drop shadow
- Summary

■ **Caution** CSS3 uses the name *box-shadow* for *drop shadow*; you might forget this and enter the more familiar term *drop shadow* into a style sheet and then wonder why it doesn't work.

CSS3 drop shadows (applied to the right and bottom edges of a picture or wrapper) are used in the first section of this chapter. These are shown in Figures [27-1](#) and [27-2](#).



Figure 27-1. Showing a drop shadow on the image and the white border of the web page

Drop Shadows for Images and Websites with Sharp Corners

The website for this project contains no PHP code so we will not be using the *htdocs* folder.

Please follow these steps:

1. Download the Chapter 27 zip file into the *web-tutorial* folder and unzip it.
2. You should now have a folder named *Chapter-27* within your *web-tutorial* folder.
3. In the folder *Chapter-27* you will see three sub-folders: *square-corners*, *round-corners*, and *Completed-files*.
4. Open the *square-corners* folder and open the file *index.html* in your HTML editor.
5. Find the following lines and change them as shown in bold type in the next snippet of code.

```
<div id="midcol-right">

</div>
```

The image has been given a class **class="shadow"** so that CSS3 can add a drop shadow.

6. Save the file.

7. Now open the file *style-3d.css* and add the following two lines to the end of the existing code:


```
#wrapper {box-shadow:10px 10px 20px #505050;}
.shadow {box-shadow:10px 10px 15px #505050; height:290px; width:348px;}
```
8. Save the file.
9. When you have completed these changes and saved them, view the *index.html* file in a browser.
10. Click the menu buttons to the other three pages and you will see that the drop shadow fits nicely around any size of wrapper.

Explanation of the Code

```
#wrapper {box-shadow:10px 10px 20px #505050;}
.shadow {box-shadow:10px 10px 15px #505050; height:290px; width:348px;}
```

This piece of code attaches a shadow to two sides of an element. It has three dimensions in pixels and it also specifies a color for the shadow. The first dimension is the horizontal offset (i.e., how much the shadow protrudes from the right of the element). The second dimension is the vertical offset from the element (i.e., how much the shadow protrudes below the element). The third dimension defines the width of the shadow and how much to blur the shadow; in other words, the shadow has a gradient so that it is dark close to the element, and it fades as it moves further from the element. In this example, I have assumed that we don't want the same size of drop shadow on both the wrapper and the image. I have made the box-shadow a little different for the image, the blur radius is 15 pixels, whereas the blur radius for the wrapper is 20 pixels. Note that the dimensions of the image have been included to ensure that the shadow fits snugly around the image. The wrapper has a variable height and width; therefore we cannot include its dimensions.

The CSS3 command *box-shadow* is applied to the *wrapper*'s identity by using the CSS code `#wrapper`.

The *box-shadow* command is applied to the image that has the attribute **class="shadow,"** The corresponding CSS code is `.shadow`.

Let's examine the CSS for the `#wrapper`; the box-shadow components are as follows:

```
{box-shadow:10px 10px 20px #505050;}
```

The code is specifying a horizontal offset of 10 pixels, a vertical offset of 10 pixels, a blur of 20 pixels, and a color `#505050`;

The horizontal offset specifies the amount that the shadow protrudes to the right of the *wrapper*.

The vertical offset specifies the amount that the shadow protrudes below the *wrapper*.

The blur also sets the radius of the blurred shadow; if this is zero the shadow will have sharp corners.

The color of the shadow is set as `#505050`; this is suitable for most web pages. However, it can be varied to suit different background colors.

The shadow attributes were determined by trial and error, but the attributes in this example provide a good starting point for setting your own drop shadows.

■ **Note** The wrapper is identified by an **id**, but a **class** identifies the image. You will recall from earlier chapters that a particular **id** can only be used once per web page, whereas the same **class** may be used any number of times on a page. This is because there is only one wrapper per page, but you might have a whole gallery of images on a page.

The CSS3 drop shadow will also fit around a web page with rounded corners; in fact both rounded corners and drop shadows are features that were made possible by CSS3.

Drop Shadows for a Website with Rounded Corners

This project should produce a display as shown in Figure 27-2.



Figure 27-2. A drop shadow applied to the wrapper's white border

The website for this project contains no PHP code so we will use the *web-tutorial* folder instead of the *htdocs* folder. Please follow these steps:

1. In the folder *Chapter-27* open the *round-corners* folder and then open the file *index.html* in your HTML editor. Find the following lines and change them as shown in bold type in the next snippet of code.

```
<div id="midcol-right">

</div>
```

The image has been given an identity **class="shadow"** so that CSS3 can give it a drop shadow.

2. Save the file.
3. Now open the file *style-3d.css* and add the following two lines at the end of the existing code:


```
#wrapper {box-shadow:10px 10px 20px #505050;}
.shadow {box-shadow:10px 10px 15px #505050; height:290px; width:348px;}
```
4. Save the file.
5. When you have completed these changes and saved them, view the *index.html* file in a modern browser (CSS3 drop shadows will not work in Internet Explorer 8).

Note how the shadow follows the rounded corner at the bottom right of the wrapper.
6. Click the menu buttons to the other three pages and you will see that the drop shadow fits nicely around curved corners for any size of wrapper.

Add a Drop Shadow and a White Border to an Image

CSS3 allows us to add a white border and a drop shadow to an image. The next project illustrates this feature and the result should be as shown in Figure 27-3.



Figure 27-3. Shows a picture with a CSS3 white border and a drop shadow

Please follow these steps:

1. In the folder *round-corners*, open the file *index.html* in your HTML editor.
2. Use *Save As* to save the file with the new name *index-new.html*.
3. Change the file as shown in bold type in the following snippet of code:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Home Page</title>
    <meta charset=utf-8>
    <link href="style-3d-white-border.css" rel="stylesheet"
      type="text/css">
```

4. The picture you have used so far has a wooden frame and would look very odd if we added a white border around a wooden frame; therefore we will use the picture named *art3.jpg*, as this has no wooden frame. In *index-new.html*, find the code for the image and change it as shown in bold type in the next snippet:

```
<div id="midcol-right">  </div>
```

We now need to provide a revised style sheet named *style-3d-white-border.css*

5. Open the style sheet *style-3d.css* in your HTML text editor
6. Use *Save As* to save it with the new name *style-3d-white-border.css*.
7. Change the last of line of the two lines of code as shown in the next snippet:

```
#wrapper {box-shadow:10px 10px 20px #505050;}
.shadow-bdr {background-color:white; padding:10px; box-shadow:10px
10px 15px #505050; ⬅
height:270px; width:327px;}
```

8. Save the file.
9. Open the *index-new.html* file in a modern browser, and it should display as shown in Figure 27-3.

Don't click the Home Page button because it will take you to the original home page *index.html*.

Explanation of the Code

```
.shadow-bdr {background-color:white; padding:10px; box-shadow:10px 10px 15px #505050; ⬅
height:270px; width:327px;}
```

The class **.shadow-bdr** has two new attributes, and the first one adds a white background color behind the image. The second attribute is **padding:10px;** and adds 10 pixels of padding around the image and this padding reveals the white background as a border 10 pixels wide.

■ **Tip** A drop shadow can be added around any HTML box such as a paragraph `<p></p>`, or heading `<h1>`, `<h2>`, `<h3>`, etc.

Add a Drop Shadow to Text

Texts with drop shadows have traditionally been created by using graphics software. Search engines cannot read images; therefore header text created as an image would be wasted as far as search engines are concerned. Because header text is regarded as very important to search engines it should always be real text and not an image. CSS3 can add a drop shadow to real text so that we can have the best of both worlds. Figure 27-4 show the left-upper quarter of the page so that you can see the drop shadow clearly.



Figure 27-4. *Heading text with a CSS3 drop shadow*

Create the page shown in Figure 27-4 by following these steps:

1. In the folder *round-corners*, open *index-new.html* in your HTML editor.
2. Use *Save As* to save the file with the new name *index-text.html*.
3. Change the file as shown in bold type in the following snippet of code:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Home Page</title>
    <meta charset=utf-8>
    <link href="style-3d-text.css" rel="stylesheet" type="text/css">
```

4. Open the file *style-3d-white-border.css* in your HTML text editor and save it with new name *style-3d-text.css*

5. Change the file *style-3d-text.css* as shown in bold type in the following snippet of code:

```
header h1 {position:absolute; top:45px; margin-left:30px;
font-size:350%; color:white; ↵
font-weight:bold; text-shadow:2px 2px 4px black;}
```

6. Save the file.
7. View the file in a modern browser, but do not click the Home Page button because it will take you back to the previous *index.html* page.

Explanation of the Code

The changes were the following:

font-size:350%; color:white; font-weight:bold; **text-shadow:2px 2px 4px black;**

To demonstrate the feature clearly the font size was increased and its color was changed to white. Then the text shadow was added using the code **text-shadow:2px 2px 4px black;** The code is very similar to the box-shadow code but the term *text-shadow* is used.

Shadows on Four Sides of an HTML Element

Four-sided shadows are shown in Figure 27-5.



Figure 27-5. The picture and the wrapper have been given a drop shadow on all four sides

Create a Four-Sided Drop Shadow

To create a four-sided shadow, please follow these steps:

1. In the folder *round-corners*, open the *index-new.html* file in your text editor and then use *Save As* to save it with the new name *index-4shadows.html*.
2. Change the link to the style sheet as shown in bold type in the following snippet:

```
<link href="style-3d-4shadows.css" rel="stylesheet"
type="text/css">
```

3. Open the file *style-3d-white-border.css* with your text editor.
4. Use *Save As* to save copy with the new name *style-3d-4shadows.css*.
5. Look for the shadow styles and change them as shown in bold type in the following snippet:

```
#wrapper {box-shadow:0 0 30px black;}
.shadow-bdr {background-color:white; padding:10px; border:1px
black solid; ↵
box-shadow:0 0 20px black; height:290px; width:348px;}
```

Note the two zeros that replace the offset amounts. The zeros set no horizontal or vertical offset; therefore the shadow appears on all four sides of the image and the wrapper.

6. View *index-4shadows.html* in a modern browser, and it should look like Figure 27-5. Don't click the Home Page button because it will take you back to the previous *index.html* page.

■ **Tip** How can you add drop shadows to several images on a page if the images are not the same size?

The various sizes of image would need their own identity such as **class="shadow-bdr1"**, **class="shadow-bdr2"**. The CSS would match this with the following:

```
.shadow-bdr1 {background-color:white; padding:10px; border:1px black solid; ↵
box-shadow:0 0 20px black; height:290px; width:348px;}
```

and:

```
.shadow-bdr2 {background-color:white; padding:10px; border:1px black solid; ↵
box-shadow:0 0 20px black; height:300px; width:360px;}
```

This is also valid for images of different sizes with two drop shadows (with or without the white border). Having said that, when designing a picture gallery it is good practice to make the pictures the same size for a neat display; this automatically solves the drop shadow problem.

Summary

In this chapter you learned how to add a CSS3 drop shadow to a website. You found that the drop shadow automatically adjusts to accommodate any size of wrapper; but when adding a drop shadow to an image, the dimensions of the image must be stated in the CSS code. You learned how the CSS3 drop shadow can also be used for a web page's *wrapper* with rounded corners. You then discovered that you can use CSS3 to add a white border around an image and that a drop shadow can be applied to that white border. You learned how to add a drop shadow to text, and finally you discovered how to add four shadows around an image and a wrapper. In the next chapter you will learn how to create a user name and password for a page that is only accessible to a limited group of people.



User Name and Password for a Member's Page

This project is suitable for a small society or club. It hides certain information from the general public but makes it available to members (or a restricted group of members such as a committee). We achieve this by means of a user name and password.

The only drawback is that each member must be given the same password and user name. However, the user name and password can be changed at any time if security suddenly becomes an issue.

If a larger society or club prefers a unique identification for each member, the members would register their user names and passwords via a registration page; their identification details would be stored in a database. Databases are beyond the scope of this book but are described in my recent book *Practical PHP and MySQL Website Databases – A Simplified Approach* (Apress.com).

This chapter contains the following sections:

- Create the login page
- Explanation of the form code
- Create a members' page
- Create a login-handler
- Summary

Create the Login Page

The login page is shown in Figure 28-1.



Figure 28-1. The login page

To create the login page please follow these steps:

We will be using a PHP file in this project; therefore we will work within the XAMPP *htdocs* folder.

1. Download the zip file named *Chapter-28.zip* and put it in the *htdocs* folder.
2. Unzip the zip file *Chapter-28.zip*.
3. Find the file named *template.php* and open it in the Source/Code view of your HTML text editor.
4. Use *Save As* to save a copy of the file with the name *login-form.php*.
5. Change the **<title>** from *Template* to *Login Form*.
6. Add the link to the login style sheet.
7. Scroll down and change the **<h2>** heading from *Template* to *LogIn*.
8. Create some line spaces (use the *Enter* key) immediately below the line **<h2>LogIn</h2>** and in the spaces insert the form code shown in bold type in the Listing 28-1.

Listing 28-1. Create the Login Page.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Login form</title>
    <meta charset=utf-8>
    <link href="style-3d-gdn.css" rel="stylesheet" type="text/css">
    <link href="login-style.css" rel="stylesheet" type="text/css">
    <!--[if lte IE 8]>
      <script src="html5.js"></script>
    <![endif]-->
  </head>
  <body>
    <div id="wrapper">
      <header>
        <h1>The Spring Garden Club</h1>
      </header>
      <?php include ('includes/hmenu.html'); ?>
    <div id="content">
    <div id="leftcol">
      <?php include ('includes/vmenu.html'); ?>
    </div>
    <div id="rightcol">
      <p>This is the far right column</p>
    </div>
    <div id="midcol">
      <h2>Log In</h2>
      <p>The members' page will not appear if you enter incorrect information</p>
      <form action="login-handler.php" method="post">
        <h3>Required items <span class="red">*</span></h3>
        <label class="label" for="username"><strong>User&nbsp;Name</strong> ~
        <span class="red">*</span>
        <input id="username" name="username" size="30"></label><br><br>

        <label class="label" for="password"><strong>Password</strong><span class="red">*</span>
        <input id="password" type="password" name="password" size="30"></label><br><br>
        <div id="submit">
          <input id="submit" value="Submit" title="Submit" type="submit">
        </div>
      </form><!--end of form-->
    </div>
  </div>
  <footer>
    <br>This is the footer<br><br>
  </footer>
</div><!--content div finishes here-->
</div><!--wrapper div finishes here-->
</body>
</html>
```

Explanation of the Form Code

Most of the form code was explained in Chapters 18 and 23. However, some new features will be described as follows:

```
<form action="login-handler.php" method="post">
```

In Chapter 23 the information entered by the user was sent (posted) to a handler that sent an email. The line of code above sends the user name and password to a much simpler handler that checks the user name and password; if these are satisfactory it opens the members' page.

```
<label class="label" for="password"><strong>Password</strong>
  <span class="red">*</span>
<input id="password" type="password" name="password" size="30"></label><br><br>
```

In the second line of this code you will see this: **type="password."** This provides the row of asterisks to hide the password from anyone watching the user.

Styling the Login Page

The CSS style sheet for the login page *login-style.css* is included in the download files and the code is given in Listing 28-2.

Listing 28-2. The code that styles the form elements (*login-style.css*).

```
h3 {text-align:center;}
/*Position the form elements on the page*/
form {width:500px; margin:auto; text-align:center;}
.label {float:left; width:400px; text-align:right; clear:left;}
#submit { text-align:center;}
.red {color:red; font-weight:bold;}
```

Create a Members' Page

A successful login will display the members' page. However, an unauthorized user could access the members' page by trial and error. For instance, he could enter this:

<http://www.thespring-gardenclub.co.uk/members.php>

To prevent this, use an obscure file name for the members' page. I therefore ask you to use the name *tuliptime5.php*.

Please follow these steps:

1. Find the file named *template.php* and open it in the Source/Code view of your HTML text editor.
2. Use *Save As* to save a copy of the file with the name *tuliptime5.php*.
3. As shown in bold type in Listing 28-2, change the title to *Members' Page*.
4. In the <head> section, add the internal style for the agenda and the h2 tag.
5. Change the <h2> heading from *Template* to *Members' Page*.

6. Immediately below that line `<h2>Members' Page</h2>`, create some line-spaces (use the *Enter* key).
7. In the spaces, insert the agenda text shown in bold type in the Listing 28-3.

Listing 28-3. Create a members' page (tuliptime5.php).

```
<!DOCTYPE html>
<html>
  <head>
    <title>Members' Page</title>
    <meta charset=utf-8>
    <link href="style-3d-gdn.css" rel="stylesheet" type="text/css">
    <style type="text/css">
      p#agenda {margin-left:220px; text-align:left; }
      h3 {text-align:center;}
    </style>
    <!--[if lte IE 8]>
      <script src="html5.js"></script>
    <![endif]-->
  </head>
  <body>
    <div id="wrapper">
      <header>
        <h1>The Spring Garden Club</h1>
      </header>
      <?php include ('includes/hmenu.html'); ?>
      <div id="content">
        <div id="leftcol">
          <?php include ('includes/vmenu.html'); ?>
        </div>
        <div id="rightcol">
          <p>This is the far right column</p>
        </div>
        <div id="midcol">
          <h2>Members' Page</h2>
          <h3>The Agenda for the Meeting of 17th December 2016 7.00pm</h3>
          <p id="agenda">Apologies for absence<br>Minutes of previous meeting<br>
            Approval of minutes<br>Financial report for 2015<br>Approval of ↵
            accounts<br>
            Plans for 2017<br>Report of club visit to Chelsea Flower Show <br>
            Any other business<br>Date of next committee meeting</p>
        </div>
        <footer>
          <p>This is the footer</p>
        </footer>
      <br>
    </div><!--content div finishes here-->
  </div><!--wrapper div finishes here -->
</body>
</html>
```

Start XAMPP and Apache and use `http://localhost/Chapter-28/tuliptime5.php` to view the file in a modern browser. It should look like Figure 28-2.



Figure 28-2. The members' page

Now that we have a *login* page and a *members'* page, we need a form-handler to create a link between the two. When the *Submit* button is clicked on the login page, the user name and password are sent (posted) to the form-handler file. This is a much simplified version of the contact form-handler in Chapter 23. To differentiate it from the previous *form-handler.php*, I have named it *login-handler.php*. Error checking has not been included; if the user enters incorrect information, the form remains on the page ready for another attempt to log in.

Create a Login-Handler

To create the *login-handler.php* page please follow these steps:

1. Open your HTML text editor in Source/Code view.
2. Erase all existing text in the Source/Code view and type in the code shown in Listing 28-4. In this example the username is `picklejar` and the password is `optimum37`.

Listing 28-4. Create the *login-handler.php* file.

```
<?php
// list the pages to be displayed,
$formurl = "login-form.php" ;
$members = "tuliptime5.php" ;
```

```
// ----- Set the information received from the form as $ values -----
$username = $_POST['username'] ;
$password = $_POST['password'] ;
//Check that the two essential fields are filled in
if (empty($username) || empty($password) ) {
header( "Location:$formurl" );
    exit() ; }
#remove any spaces from beginning and end of password
$username = trim($username);
$password = trim($password);
if ($username=="picklejar" AND $password="optimum37") {
header( "Location:$members" ) ;
exit();
} else {
    header( "Location:$formurl" );
}
?>
```

3. Save the file as *login-handler.php*.
4. Start XAMPP and Apache and use *http://localhost/Chapter-28/login-form.php* to load the file in a modern browser. Test the login process using the correct user name and password. Then try deliberately entering incorrect details, or omit the user name and password altogether.

Explanation of the Code

```
<?php
```

This tag tells the browsers that the code is PHP and not HTML:

```
// list the pages to be displayed,
$formurl = "login-form.php" ;
$members = "tuliptime5.php" ;
```

The two main pages are given shortened names by assigning them to PHP variables.

```
$username = $_POST['username'] ;
$password = $_POST['password'] ;
```

The user name and password that are posted from the login form are given shorter names by assigning them to the PHP variables *\$username* and *\$members*.

```
//Check that the two essential fields are filled in
if (empty($username) || empty($password) ) {
header( "Location: $formurl" );
    exit ;}
```

If the user has failed to enter either the username or the password (or both), the handler stops working and the login form is displayed ready for another attempt. To simplify the form and its handler, no error traps or error messages are included. Instead, the login form displays the following statement:

The members' page will not appear if you enter incorrect information

```
$username = trim($username);
$password = trim($password);
```

Any spaces are removed from the beginning and end of the username and password.

```
if ($username == "picklejar" AND $password="optimum37") {
header("Location: $members" );
exit;
```

If the user has entered the username *pickle jar* and the password *optimum37*, the form-handler displays the members' page. Note the double equals symbol; this is very important. It means *exactly equal*; if you replace the double equals symbol with a single one, the handler will allow incorrect entries and absent entries to access the members' page. Also, because the double equals symbol means *exactly equal*, the input is case sensitive so that entering Picklejar instead of picklejar would not permit the user to access the members' page.

```
} else {
    header( "Location:$formurl" );
}
```

The *else* keyword tells the browser that if the user name and password are not present and correct, the login form will continue to display and the user will not access the members' form.

```
?>
```

This tag signifies the end of the PHP code.

Summary

In this chapter you learned how to create a simple login form, a members' page, and a form-handler. In the next chapter, you will learn how to create a printable order form that can be filled in on the computer screen.

CHAPTER 29



Create a Printable Order Form

Selling goods or services through a website requires a payment method; PayPal and its credit/debit card process are described briefly in Chapter 17 of this book. However, some people prefer to pay by check because either they don't have a PayPal account, or they feel uncomfortable about divulging their debit/credit card details over the Internet. To avoid losing sales, we should make provision for payments by PayPal, card, and check. The traditional way of achieving this is to have an ordering page that presents the user with a choice of payment methods. If users choose to pay by PayPal/card they will be presented with an appropriate page where they can select the goods and see the total cost before clicking the "Buy Now" button. This is beyond the scope of this beginners' book because it requires a database; a fully worked example is given in Chapter 11 of my recent book *Practical PHP and MySQL Website Databases: A simplified approach* (Apress.com).

This chapter describes a simple printable form. The technique can be applied with any web page; it does not need to be an order form. I have chosen to use an example that will enable users to order goods or services and pay by check. The form has the advantage that the user can fill in part of it on the screen using the keyboard and mouse. Of course this does not apply to the user's signature; this must be written on the printed form. Also, the user must choose the goods and total the cost after the form has been printed out. The form prints out using only black ink to save the user's expensive color cartridge. Some elements on the order form (that are displayed on the screen) are automatically omitted from the printout because they are not necessary (e.g., the header picture, the print button, the footer, and the up-arrow).

This chapter has the following sections:

- Add an order form button to the vertical menu
- View and modify the order form
- Create the style for the printed version of the form (*print-order.css*)
- Summary

Add an Order Form Button to the Vertical Menu

Please follow these steps to create a folder for Chapter 29 and its contents:

1. Open your *htdocs* folder.
2. Download the zip file *Chapter-29.zip* and place it in *htdocs* folder, then unzip it.
3. Open the *includes* folder.

4. Open *vmenu.html* in the Code/Source view of your HTML text editor and add a new button below the *Terms* button as shown in bold type in the next snippet of code:

```
<nav id="vertical">
  <ul>
    <li class="vbtn"><a href="#" title="About Us">About Us</a></li>
    <li class="vbtn"><a href="contact.php" title="Contact Us">Contact Us</a></li>
    <li class="vbtn"><a href="#" title="Locate Us">Locate Us</a></li>
    <li class="vbtn"><a href="#" title="Terms and Conditions">Terms</a></li>
    <li class="vbtn"><a href="order-form.php" title="Order Form"> Order Form</a></li>
    <li class="vbtn"><a href="login-form.php" title="Log In">Log In</a></li>
  </ul>
</nav>
```


5. Save the file.

We now have a menu button that will allow us to access the order form.

View and Modify the Order Form

We will now examine the file *order-form.php* (included in the downloaded zip file; therefore there is no need to create it from scratch).

Figure 29-1 shows the order form.



ORDER FORM

[Return to Home Page](#)

Type your details on-screen using your mouse and keyboard.

Data protection: Your name and address will not be stored or shared

You will find a "print" button half way down the form.

Print two copies, then sign and date them. Keep one copy and post one to:
 The Spring Garden Club, The Garden Centre, 10 The Street, Townsville.
 and enclose a cheque made payable to The Spring Garden Club

Required *

Name*

Your Email Address*

Your Phone Number

House and street*

Address

Town*

Post Code*

When you have printed the form, please write your purchase details
 in the table below

[Click to automatically print form in black and white](#)

Please send me the following items:


Qty	Item	Price	Total
	Pack of 10 Tulip bulbs (purple white)	£4.00 +£1.00 p&p	
	Pack of 20 Crocus bulbs (yellow)	£4.00 +£1.00 p&p	
	Pack of 20 Crocus bulbs (purple)	£4.00 +£1.00 p&p	
	Pack of 20 Snowdrop bulbs	£4.00 +£1.00 p&p	
	Pack of 10 King Alfred Daffodil bulbs	£5.00 +£1.00 p&p	
		Total	

Note: For multiple orders: please use the Contact Us button on the home page for a quote for the P&P

☐ Please tick the box to receive emailed newsletters

Your email address will not be shared with any other organization

Signed _____ Date _____


[Go to Top](#)

This is the footer

Figure 29-1. The printable order form

The code for the order form is given in Listing 29-1; some items in bold type are omitted from the downloaded file because you will be asked to enter them. Note that menus are excluded in order to provide the maximum space for the form elements. This avoids problems such as displaced labels that are caused by a lack of horizontal space. However, a button has been added so that the user can return to the home page.

Please follow these steps:

1. Open the file *order-form.php* with your HTML text editor.
2. In the Code/Source view insert the code shown in bold type in Listing 29-1.
3. Save the file.

Because of the inclusion of a lengthy table, the listing for the order form is rather long. I have broken Listing 29-1 into two blocks for clarity; the two blocks are Listing 29-1a and Listing 29-1b:

The items in bold type prepare the form so that we can add the ability to print the form in black ink. The items in bold also ensure that the printed version is devoid of redundant features.

Listing 29-1a. How the printable order form was created (first block of code)

```
<!DOCTYPE html>
<html>
<head>
<title>The Spring Garden Club order form</title>
<meta charset="utf-8">
  <link href="style-3d-gdn.css" rel="stylesheet" type="text/css">
  <link href="order.css" rel="stylesheet" type="text/css">
  <link href="print-order.css" rel="stylesheet" type="text/css" media="print">
  <!--[if lte IE 8]>
    <script src="html5.js"></script>
  <![endif]-->
</head>
<body>
<a id="top"></a>
<div id="wrapper">
<!--header begins now-->
<header>
  <h1>The Spring Garden Club</h1>
</header>
<div id="midcol">
  <h2>ORDER FORM</h2><br>
  <div id="back-button"><a title="Return to the Home page" href="index.php">Return
    to Home Page</a>
  </div><br>
<h2 class="none"><span class="red">Type your details on-screen using your mouse and
  keyboard.</span> </h2>
<p class="none">Data protection: Your name and address will not be stored or
  shared</p>
<h3 class="none">You will find a "print" button half way down the
  form.</h3>
<h3>Print two copies, then sign and date them. Keep one copy and post one to: <br>
  The Spring Garden Club, The Garden Centre, 10 The Street, Townsville.<br>
  and enclose a cheque made payable to The Spring Garden Club</h3>
<h3>Required <span class="large-red">*</span></h3>
</div>
</div>
```

```

<label class="label" for="firstname">Name<span class="large-red">*</span>
  <input id="name" name="name" size="35"></label><br>
<label class="label" for="useremail">Your Email Address<span
  class="large-red">*</span>
  <input id="useremail" name="useremail" size="35"></label><br>
<label class="label" for="phone">Your Phone Number
  <input id="phone" name="phone" size="35"></label><br>
<label class="label" for="address1">House and street<span class="large-red">*</span>
  <input id="address1" name="address1" size="35"></label><br>
<label class="label" for="address3">Address&nbsp;  
  <input id="address3" name="address3" size="35"></label><br>
<label class="label" for="town">Town<span class="large-red">*</span>
  <input id="town" name="town" size="35"></label><br>
<label class="label" for="postcode">Post Code<span class="large-red">*</span>
  <input id="postcode" name="postcode" size="35"></label>
</form>

```

Listing 29-1b. This block of code follows on from Listing 29-1a, and it contains the order form's table

```

<br><br class="clear">
<div id="purchases">
  <h3>When you have printed the form, please write your purchase details<br>in the
    table below</h3>
  <div id="button">
    <input type="button" value="Click to automatically print form in black and white"
    onclick="window.print()" title="Print this Form"><br>
  </div>
  <h3>Please send me the following items:</h3>
  <div >
    <table style="width:770px;">
      <tr class="row1">
        <td style="width: 56px">Qty</td>
        <td style="width: 298px">Item</td>
        <td style="width: 191px">Price</td>
        <td style="width: 110px">Total</td>
      </tr>
      <tr class="30">
        <td>&nbsp;  </td>
        <td>Pack of 10 Tulip bulbs (purple/white)<br></td>
        <td>&pound;4.00&nbsp;  &pound;1.00 p&amp;p</td>
        <td class="td4">&nbsp;  </td>
      </tr>
      <tr class="30">
        <td>&nbsp;  </td>
        <td>Pack of 20 Crocus bulbs (yellow)<br></td>
        <td>&pound;4.00 +&pound;1.00 p&amp;p</td>
        <td class="td4">&nbsp;  </td>
      </tr>
    </table>
  </div>

```

[illegible]

Now I will explain the new code that you just entered; later I will explain the code in Listing 29-1 that you will not have seen before.

Explanation of the New Code

```
<link href="style-3d-gdn.css" rel="stylesheet" type="text/css">
<link href="order.css" rel="stylesheet" type="text/css">
<link href="print-order.css" rel="stylesheet" type="text/css" media="print">
```

The first two lines, by default, dictate how the form will be displayed on the screen. They can also be written like this:

```
<link href="style-3d-gdn.css" rel="stylesheet" type="text/css" media="screen">
<link href="order.css" rel="stylesheet" type="text/css" media="screen">
```

```
<link href="print-order.css" rel="stylesheet" type="text/css" media="print">
```

This is called a media query, and it is only invoked if the user chooses to print the form. In the third line, the attribute *media="print"* tells the browser to use the style sheet *print-order.css* for printing the form. The new style sheet *print-order.css* contains special instructions that cause the printer to print selected items only. The print-order style sheet will be described later in this chapter.

```
<div id="button">
<input type="button" value="Click to automatically print form in black and white"
onclick="window.print()" title="Print this Form"><br>
</div>
```

When clicked, this button will print the filled-in form. The item **onclick="window.print()"** is a standard function that all browsers understand.

Explanation of the Unfamiliar Code in Listing 29-1

```
<body>
<a id="top"></a>
```

The link `` marks the top of the page, so that when the up-arrow at the bottom of the page is clicked, the display moves to this link at the top of the page.

```
<h2>ORDER FORM</h2><br>
<div id="back-button"><a title="Return to the Home page" href="index.php">Return to Home
Page</a>
</div><br>
```

This code displays a back button that will return the user to the home page. This compensates for the lack of menus. The color, style, and position of the back button are set by the style sheet *order.css*

■ **Tip** The form code that follows the back button code is similar to the form described in Chapter 23 and needs no further explanation.

<h3>Please send me the following items:</h3>

<div>

```

    <table style="width:770px;">
      <tr class="row1">
        <td style="width: 56px">Qty</td>
        <td style="width: 298px" >Item</td>
        <td style="width: 191px">Price</td>
        <td style="width: 110px">Total</td>
      </tr>
      <tr class="30">
        <td>&nbsp;</td>
        <td>Pack of 10 Tulip bulbs (purple/white)<br></td>
        <td>&pound;4.00&nbsp;&pound;+&pound;1.00 p&amp;p</td>
        <td class="td4">&nbsp;</td>
      </tr>
      <tr class="30">
        <td>&nbsp;</td>
        <td>Pack of 20 Crocus bulbs (yellow)<br> </td>
        <td>&pound;4.00 +&pound;1.00 p&amp;p</td>
        <td class="td4">&nbsp;</td>
      </tr>
      <tr class="30">
        <td>&nbsp;</td>
        <td>Pack of 20 Crocus bulbs (purple)<br></td>
        <td>&pound;4.00 +&pound;1.00 p&amp;p</td>
        <td class="td4">&nbsp;</td>
      </tr>
      <tr class="30">
        <td>&nbsp;</td>
        <td>Pack of 20 Snowdrop bulbs<br></td>
        <td>&pound;4.00 +&pound;1.00 p&amp;p</td>
        <td class="td4">&nbsp;</td>
      </tr>
      <tr class="30">
        <td>&nbsp;</td>
        <td>Pack of 10 King Alfred Daffodil bulbs<br></td>
        <td>&pound;5.00 +&pound;1.00 p&amp;p</td>
        <td class="td4">&nbsp;</td>
      </tr>
      <tr class="30">
        <td>&nbsp;</td>
        <td><br></td>
        <td class="total">Total</td>
        <td class="td4">&nbsp;</td>
      </tr>
    </table>
  </div>
</div>

```


This code displays a table. You may recall from Chapter 16 that the `<tr>` tags display rows and the `<td>` tags display cells within the rows. The table is set to a width of 770 pixels; the rows have been given classes that refer to the height of the rows; these heights are set by the style sheet *order.css* file.

```
<tr class="row1">
  <td style="width:56px">Qty</td>
  <td style="width:298px">Item</td>
  <td style="width:191px">Price</td>
  <td style="width:110px">Total</td>
</tr>
```

The four cells in the first row are given specific widths; these automatically set the widths of the columns, so this means that the widths need not be repeated for each cell in the subsequent rows.

`<h4>Note: For multiple orders please use the Contact Us button on the home page for a quote for the P&P </h4>`

The post and packing item P&P displays and prints as P&P, if we did not use the entity *&*; the ampersand would display as a strange symbol.

`<p>□Please tick the box to receive emailed newsletters
Your emailaddress willnot be shared with any other organization</p>`

Standard check boxes tend to print too small; therefore the entity *□* is used here and is styled by the style sheet *order.css* to give a larger box. The style uses a class named *.boxstyle*.

`<p class="uparrow">

Go to Top</p>`

Clicking the up-arrow will move the display to the top of the page. The first line gives the paragraph a class "uparrow" that is used to style and position the up-arrow on the page.

Examine the Style Sheet for the Order Form (*order.css*)

This style sheet given in Listing 29-2 is included in the downloadable files; therefore you do not need to create it.

Listing 29-2. The order form's style sheet (*order.css*)

```
/*Position the form elements on the page*/
#form {margin:30px;}
.label {float:left; margin-left:0; margin-bottom:10px; width:500px; text-align:right;}
.input {font-size:12pt;}
/*centre the back button on the order form*/
#back-button {margin:auto; text-align:center; width:200px; height:18px; padding:5px;
background-color:purple; color:white; font-size:90%; font-weight:bold;}
#back-button a {text-decoration:none; color:white;}
#back-button a:hover {color:red;}
#submit {text-align:center;}
.red {color:red;}
```

```
span.large-red {font-size:large; color:red; font-weight:bold;}
#midcol {width:750px; margin-left:155px;}
#midcol h2 {margin:auto;}
#midcol h3 {text-align:center;}
#purchases {width:100%;}
h4 {text-align:center;}
table {width:770px; border:1px black solid; border-collapse:collapse; margin:0;}
tr.row1 {font-weight:bold; text-align:center;}
tr.30 {height:30px;}
tr.40 {height:40px;}
.td4 {height:40px;}
td {border:1px black solid; border-collapse:collapse; padding-left:5px; text-align:left;
font-size:110%}
td.total {text-align:right; font-weight:bold; padding-right:5px; }
.boxstyle {font:xx-large bold; font-family:"Lucida Sans Unicode"}
#button {margin-left:210px;}
p.sign {margin-top:10px;}
img {border:none; }
```

Note that the check box is styled using the following code:

```
.boxstyle {font:xx-large bold; font-family:"Lucida Sans Unicode"}
```

The rest of the code has been used in previous chapters and needs no further explanation.

Create the Style for the Printed Version of the Form (*print-order.css*)

A supplementary style sheet called a *media query* is used to create the printed version of the order form. This style is only invoked when the user chooses to print the form. Please follow these steps:

1. In your HTML editor please create the following CSS style sheet:

Listing 29-3. Styles for a printed form that (i) uses black ink only and (ii) eliminates redundant items

```
/*SELECT ITEMS THAT YOU DO NOT WANT TO PRINT*/
img, #button, #back-button, p.uparrow, img, .screen, footer {display:none;}
#wrapper {border:none;}
body {color:black !important;}
#form {font-size:115%; !important}
#form {margin-left:60px; margin-right:0;}
.label {float:left; margin-left:50px; margin-bottom:10px; width:500px; text-align:right;
font-size:12pt;}
input {font-size:12pt;}
table {width:1100px; margin-left:-60px;}
table, td {border:1px black solid;}
p.sign {margin-bottom:-20px;}
```

2. Save the file as *print-order.css*.

Figure 29-2 shows the printed version of the order form.

The Spring Garden Club order form
Page 1 of 1

The Spring Garden Club

ORDER FORM

**Print two copies, then sign and date them. Keep one copy and post one to:
The Spring Garden Club, The Garden Centre, 10 The Street, Townsville.
and enclose a cheque made payable to The Spring Garden Club**

Required *

Name*

Your Email Address*

Your Phone Number

House and street*

Address

Town*

Post Code*

**When you have printed the form, please write your purchase details
in the table below**

Please send me the following items:

Qty	Item	Price	Total
	Pack of 10 Tulip bulbs (purple/white)	£4.00 +£1.00 p&p	
	Pack of 20 Crocus bulbs (yellow)	£4.00 +£1.00 p&p	
	Pack of 20 Crocus bulbs (purple)	£4.00 +£1.00 p&p	
	Pack of 20 Snowdrop bulbs	£4.00 +£1.00 p&p	
	Pack of 10 King Alfred Daffodil bulbs	£5.00 +£1.00 p&p	
		Total	

**Note: For multiple orders please use the Contact Us button on the home page for a
quote for the P&P**

☐ Please tick the box to receive emailed newsletters
Your email address will not be shared with any other organization

Signed _____ Date _____

<http://localhost/Chapter-29/order-form.php>
11/10/2014

Figure 29-2. The printed version of the order form

Note that the unwanted items are not displayed on the printed copy, for example, the header image, the colored items, the print button, the back button, the up-arrow, and the footer.

With XAMPP and Apache running, open the file *order-form.php* in a browser using the address <http://localhost/Chapter-29/order-form.php>

Fill in your personal details in the top half of the form and then click the button that will automatically print a paper copy of the form in black ink. Alternatively you could click *File* on the browser menu and select *Print Preview*; however, this may not accurately depict what will be printed.

Explanation of the Styling Code for the Printed Version of the Order Form

```
/*SELECT ITEMS THAT YOU DO NOT WANT TO PRINT*/
img, #button, #back-button, p.uparrow, img, .screen, footer {display:none;}
```

The attribute *display:none*; prevents the listed items from printing.

```
#wrapper {border:none; float:left; margin-right:100px;}
```

We do not wish to waste the user's colored ink for printing the wrapper's colored border. It is therefore styled as *border:none*;

```
body {color:black !important;}
```

All text is set to print in black ink only. The attribute *!important* forces this style to override all other color styles.

```
#form {font-size:115%; !important}
#form {margin-left:60px; margin-right:0;}
.label {float:left; margin-left:50px; margin-bottom:10px; width:500px; text-align:right;
        font-size:12pt;}
input {font-size:12pt;}
```

This code sets the styles for the form. I determined these by trial and error.

```
table {width:1100px; margin-left:-60px;}
table, td {border:1px black solid;}
```

The table is styled, and again I determined this by trial and error.

Summary

In this chapter you learned how to create an order form that could be filled in on the screen using a keyboard and mouse. The filled-in form could then be printed. You inserted code that linked to a style sheet for printing the form; this style sheet ensured the most economical use of paper and ink.

In the next chapter you will learn how to add a search field to your websites.

CHAPTER 30



Add a Search Field to Your Website

A search field can increase a website's usefulness, or it can draw people away from your site, depending on how it is implemented. If you choose to provide search capability on *a specific* website, this presents no problem. The search results will concentrate on that particular website, but occasionally one or two other websites might show up among the search results.

A search field is unnecessary for small websites that have a good navigation menu. For a large site with more than 12 pages, a search field can be very useful

■ **Note** The search facilities offered in this chapter do not take you straight to the page that relates to your search word or phrase. Because it is operated by an external search engine, you will see a page of search results just as if you have entered the keyword or phrase into a search engine. Fortunately, these results will relate to the specific website. Even better, the topic the user is searching for will appear at the top, or very near the top, of the first page of results (see Figure 30-5).

The method described in this chapter is a useful compromise. To make a search go straight to the relevant page on the specific website, you would need a database. Databases are beyond the scope of this book for beginners, but my recent book, *Practical PHP and MySQL Website Databases: A Simplified Approach* (Apress.com) gives instructions for building a searchable database.

In this chapter, I will describe the code for search fields provided by the three dominant search engines: Bing, Yahoo!, and Google. This chapter contains the following sections:

- Bing search field
- Yahoo! search field
- Google search field
- Testing the Google search field on one of my websites
- Summary

You will need to create a folder for this chapter, so please follow these steps:

1. Within your *web tutorial* folder (not in *htdocs*), create a new folder named *Chapter-30*.
2. Download the Chapter-30 zip file and unzip it in your *Chapter-30* folder.

Bing Search Field

Add a Bing search field that will search only *your* website, as shown in Figure 30-1.

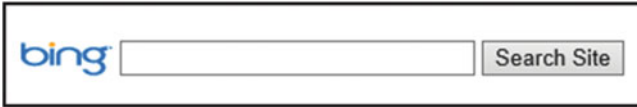


Figure 30-1. Add a Bing search field

The main search engines provide code snippets for search fields, but Bing is the easiest to install. Listing 30-1 shows how to include a Bing search field in a web page.

Listing 30-1. Creating the file *bing-search.html*

```
<!doctype html>
<head>
<meta charset=utf-8>
<title>Bing search field</title>
  <style type="text/css">
    #bingfield img {margin-bottom:-7px; border:none;}
  </style>
</head>
<body>
<div id="bingfield"><br>
  <p>
    <form method="get" action="http://www.bing.com/search">
      <input type="hidden" name="cp" value="utf-8">
      <input type="hidden" name="FORM" value="FREESS">
      <a href="http://www.bing.com/">
        <img rc="http://www.microsoft.com/presspass/_resources/images/
          img_windowsBingLogo.png" alt="Bing"></a>
      <input type="text" name="q" size="30">
      <input type="submit" value="Search Site">
      <input type="hidden" name="q1" value="site:www.yoursite.com">
    </form>
  </p>
</div>
</body>
</html>
```

The following notes refer to the bold items in Listing 30-1 above:

- The internal style in the head section removes the border from the Bing logo and pushes it down in line with the field.
- The **<p> ...</p>** tags surround the form section. Bing uses a table here, but because tables are deprecated, I changed the Bing markup to a paragraph using **<p>** tags.
- Change the charset **<meta charset="utf-8">** to match the charset your website is written in. For example, the charset will be 1252 if your website has **<charset:windows-1252>** in the head section. The number will be utf-8 if your website has **<charset:utf-8>** in the head section.

- The URL is not simple, such as **value="http://www.yourwebsite.com">**, it must be preceded by the word "site:" as follows:

```
<input type="hidden" name="q1" value="http://site:www.yourwebsite.com">
```

Don't forget to replace www.yourwebsite.com with your own website address.

Although Bing is my favorite search engine, I was rather disappointed with this search field because the page I was searching for was not at the top of the first page of results. It was on the first page of results but in sixth place. Searches using the Yahoo! and Google search fields did show the required page in first position on the first page of results.

Yahoo! Search Field

Some large organizations use Yahoo! on their websites as the default search engine; therefore the Yahoo! search field is included in this section. For the Yahoo! search field code, go to Yahoo! at http://search.yahoo.com/info/ysearchfield_instructions.html or use Listing 30-2.

Figure 30-2 shows an enlarged view of one of the four available Yahoo! search fields.

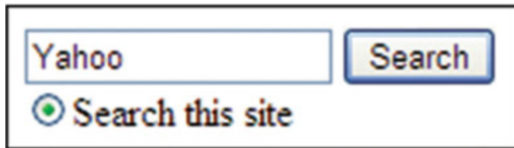


Figure 30-2. The Yahoo! search field

Four formats are available, but the instructions are rather confusing. I have simplified the Yahoo! Code as shown in Listing 30-2.

Listing 30-2. Include Yahoo! Search Box on a Web Page (*yahoo-search.html*)

```
<!doctype html>
<html>
<head>
<title>Search with Yahoo</title>
<meta charset=utf-8>
</head>
<body>
<p class="search">
<form method=get action="http://search.yahoo.com/search">
<a href="http://search.yahoo.com/"></a>
<input type="text" name="p" value="Yahoo" size=15>
<input type="hidden" name="fr" value="yscpb">
<input type="submit" value="Search"><br>
<input type="radio" name="vs" value="http://www.yourwebsite.com" checked="checked"> ↵
    Search this site
</form>
</p>
</body>
</html>
```

Insert your own website's URL in place of <http://www.yourwebsite.com>.

Google Search Field

Figure 30-3 shows one version of the various downloadable Google search fields suitable for a website.

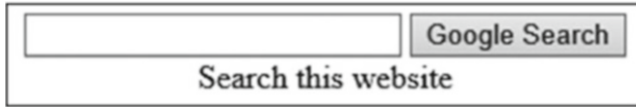


Figure 30-3. Add a Google search field

Google requires that you sign up to an account and then it presents you with a rather baffling form to fill in. To avoid all this hassle, Dave Taylor has simplified and improved the Google code. See his website at www.askdaveytaylor.com/how_can_i_add_a_google_search_field_to_my_website.html.

Listing 30-3 is an adaptation of Dave Taylor's code. An internal style sheet is used for illustration purposes only. No Google account is needed. It shows how to install a Google search field in a web page.

Listing 30-3. Inserting a Google Search Box in a Web Page (*google-search-1.html*)

```
<!doctype html>
<html>
<head>
<title>Google search field by Ask Dave Taylor</title>
<meta charset=utf-8>
  <style type="text/css">
    body #outerbox {font-size:100%;}
    #outerbox {border:1px solid black; padding:4px; width:300px; height:40px;
text-align:center; font-size:75%;}
    .googlebox {display:inline;}
    .hidden {display:none;}
  </style>
</head>
<body>
<div id="outerbox">
  <p class="googlebox">
    <form method="get" action="http://www.google.com/search" >
      <input type="text" name="q" size="25" maxlength="255" value="">
      <input type="submit" value="Google Search"><br>
      <input class="hidden" type="checkbox" name="sitesearch"
value="http://www.mywebsite.com" checked="checked">Search this website
    </form>
  </p>
</div>
</body>
</html>
```

For your own website, change the URL shown in bold type in Listing 30-3.

Figure 30-4 shows the header on my website. It contains the Google search field using my simplified version of Dave Taylor's code (see Listing 30-4 above).



Figure 30-4. I have simplified the Google search for the website; no Google account is needed for this

In Listing 30-4, I have embedded the Google search field in a page heading and I have further simplified Dave Taylor's code. As previously mentioned, no Google account is needed.

Listing 30-4. Embedding a Google Search Box in a Web Page (*google-search-2.html*)

```
<!doctype html>
<head>
<meta charset=utf-8>
<title>Google search field adapted from the 'Ask Dave Taylor website'</title>
<link rel="stylesheet" type="text/css" href="search.css">
</head>
<body >
<div id="hdr">
<h1 >Coly Computer Help</h1>
<div class="search">
<form method="get" action="http://www.google.com/search">
  <input type="text" name="q" size="15" maxlength="255" value="">
  <input type="submit" value="Find">
  <input type="hidden" name="sitesearch" value="http://www.colycomputerhelp.co.uk">
  <br>Search this website
</form>
</div>
</div>
</body>
</html>
```

For your own website, change the URL shown in bold type.

Listing 30-5 is the CSS code for styling the file *google-search-2.html*.

Listing 30-5. This Listing provides the code for the style sheet (*search.css*)

```
/*reset browsers for cross-client consistency*/
html,body,h1,h2,h3,h4,h4,h5,h6,p {margin:0; padding:0 }
img {border-style:none; float:none; margin-left:0; margin-right:0;}
/* position content at horizontal center of screen*/
body {text-align:center; background-color:#D7FFEB; color:black; ↵
  font-family:"times new roman"; max-width:1024px; min-width:800px;
font-size:medium; color:#000000; margin:auto; width:95%;}
/* set header height and background image */
#hdr{ margin-top:0; margin-bottom:0; background-position:45% top; ↵
  background-image:url('images/compbkgcrop.jpg'); background-repeat:no-repeat; ↵
```

```

height:160px; padding-bottom:0;}
h1 {padding:110px 0 0 12%; margin-bottom:0; font-family:"times new roman"; ↵
font-size:250%; color:#0080a0; font-weight:bold;}
#hdr {position:relative;}
.search {position:absolute; top:10px; right:40px; height:56px;}

```

Testing the Google Search Field on One of My Websites

In the file *google-search-2.html* (given in Listing 30-4), I used the URL of one of my own websites:

<http://www.colycomputerhelp.co.uk>

I opened the page in a browser and entered the key phrase *Windows 10* in the search field. The page of search results that appeared is shown in Figure 30-5.

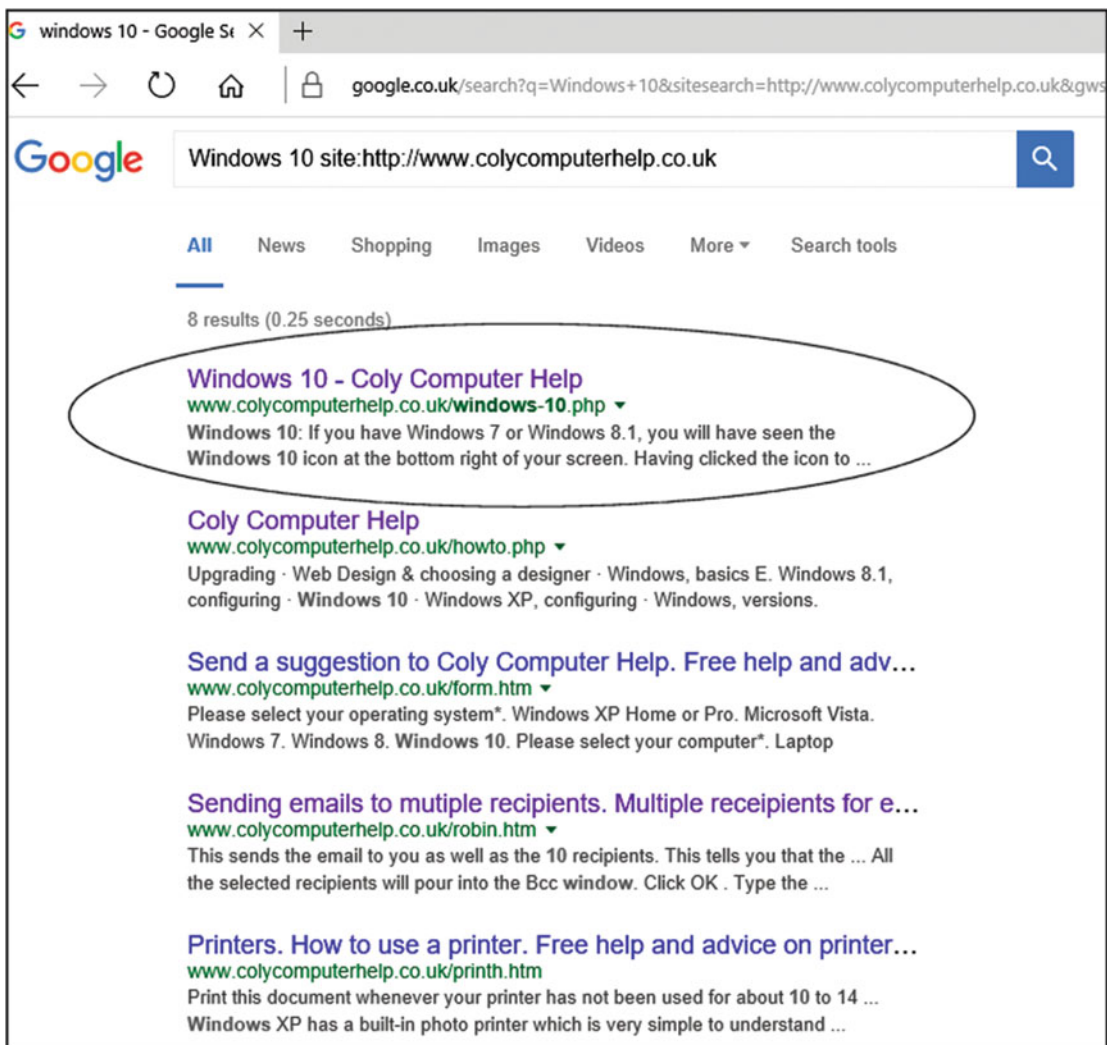


Figure 30-5. Shows the Google search results obtained by entering the key phrase *Windows 10* into the search field of my website <http://www.colycomputerhelp.co.uk>

Note the relevant web page for Windows 10 that I have shown circled at the top of the first page of search results.

You can test this yourself; open *google-search-2.html* in a browser and enter a key word/phrase such as *Windows 10*. Of course you must be connected to the Internet because the code accesses the search facility at *Google.com*.

Summary

This chapter described the search fields provided by Bing, Yahoo!, and Google. These can be embedded in your own web pages. You were then given a working example that you could test and then view the page of search results.

In the next chapter you will learn how to style bullets.

CHAPTER 31



Styled Bullet Points

The appearance and positioning of bullet points can be modified using CSS. This overcomes a problem associated with the default bullets in multicolumn text pages. This chapter contains the following sections to enable you to customize the bullets and to improve the layout of bulleted text:

- The problem with default bullets
- Create a test page
- Reduce the gaps caused by the default bullets
- Add bold headings to the bulleted paragraphs
- More bullet styles using CSS
- Summary

The Problem with Default Bullets

In multicolumn pages, the text in each column can appear to be rather narrow because the bullets and texts are indented to the right. The gap between the bullet and the text may also be larger than desirable. I have put a border around a column to illustrate the gap between the bullet and the text and the gap between the edge of the column and the bullet; these are shown circled in Figure 31-1.

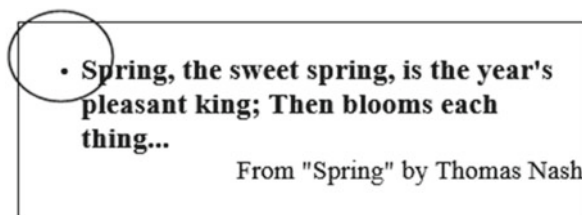


Figure 31-1. Showing the spacing of the default bulleted text

The default indent can also make the margin between the columns wider than intended as shown circled in Figure 31-2.

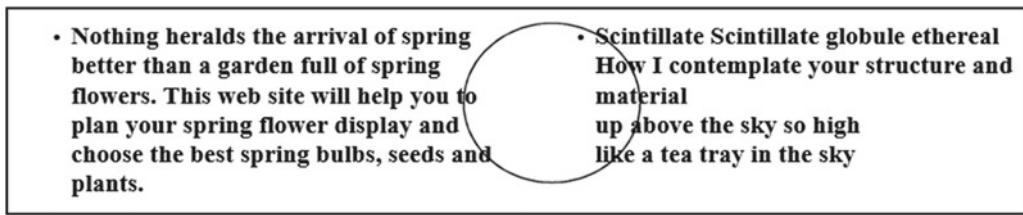


Figure 31-2. The margin between the columns appears greater than intended

The bullets and the gap between the bullets and the text are set by the browser; this may vary slightly according to which browser you use. By default, unordered bullets are black disks. The HTML tag for both an unordered list is ``, and the actual list sits between those tags. Ordered bullets are either numbers or letters, and their tag is ``. The bullets and the formatting can be changed using CSS. We will create a test page to demonstrate this.

Create a Test Page

You will need to create a new folder to hold the test page for this project. We will use the Spring Garden home page again so please follow these steps:

1. In the *htdocs* folder, create a new folder named *Chapter-31*.
2. Open the folder named *Chapter-22* and use Ctrl+A, then Ctrl+C to copy all the files into the computer's memory.
3. Open the new folder *Chapter-31*.
4. Use Ctrl+V to paste the items from the computer's memory into the folder *Chapter-31*.
5. Open the file *index.html* in your HTML text editor and save a copy as *bullet-test.html*.
6. Change the title in the header section to *bullet-test*.

■ **Tip** If you are using a WYSIWYG editor you can use the Design/WYSIWYG view; click the picture and use the delete key to delete it; then type the additional text directly into the Design/WYSIWYG view. Then you can select the text in a column and click the bullet icon on the toolbar to apply bullets.

7. We will also put a temporary border around the *midcol-left* and *midcol-right* elements so that you can see what is happening when you style the bulleted items. Find the `<style>` section in the `<head>` section and add the code shown in bold type:

```
<style type="text/css">
    .small {font-size:75%; color:black;}
    #midcol-left, #midcol-right {border:1px black solid;}
</style>
```

8. Scroll down to the section `<div id="midcol-left">` and add the un-ordered bullet tags as shown in bold type in Listing 31-1.

Listing 31-1. Add the bullets and replace the daffodil picture with some text (shown bold).

```
<div id="midcol-left">
  <ul>
    <li><p class="left">Spring, the sweet spring, is the year's pleasant king; Then ↵
      blooms each thing...<br><span class="quote">From "Spring" by Thomas Nash</span> ↵
    </li><br><br>
    <li><p class="left">Nothing heralds the arrival of spring better than a garden ↵
      full of spring flowers. This web site will help you to plan your spring flower ↵
      display and choose the best spring bulbs, seeds and plants.</p>
    </li>
    <li><p class="left">View and order spring bulbs, seeds and plants using this ↵
      website, or visit the Spring Garden Centre at 10 The Street, Townsville.</p>
    </li>
  </ul>
</div>
```

9. Scroll down to the section headed `<div id="midcol-right">` then comment-out the image and substitute some text as shown in bold type:

```
<div id="midcol-right">
  <!---->
  <ul>
    <li><p class="left">Mary had a little lamb<br> it walked into some soot,<br> ↵ then
      everywhere that Mary went<br>his sooty foot he put.</p></li>
    <li><p class="left">Scintillate Scintillate globule ethereal<br> ↵
      How I contemplate you structure and material<br> up above the sky so high ↵
      <br> like a tea tray in the sky.</p></li>
  </ul>
</div>
</div>
</div><!--content div finishes here-->
```

10. Save the page.

With XAMPP and Apache running, use `http://localhost/Chapter-31/bullet-test.php` to view the page in a modern browser. You will now have a two-column page with default bullets; the columns have been given a temporary black border so that you can see how the default bullets affect the layout of the text. This is illustrated in Figure 31-3.

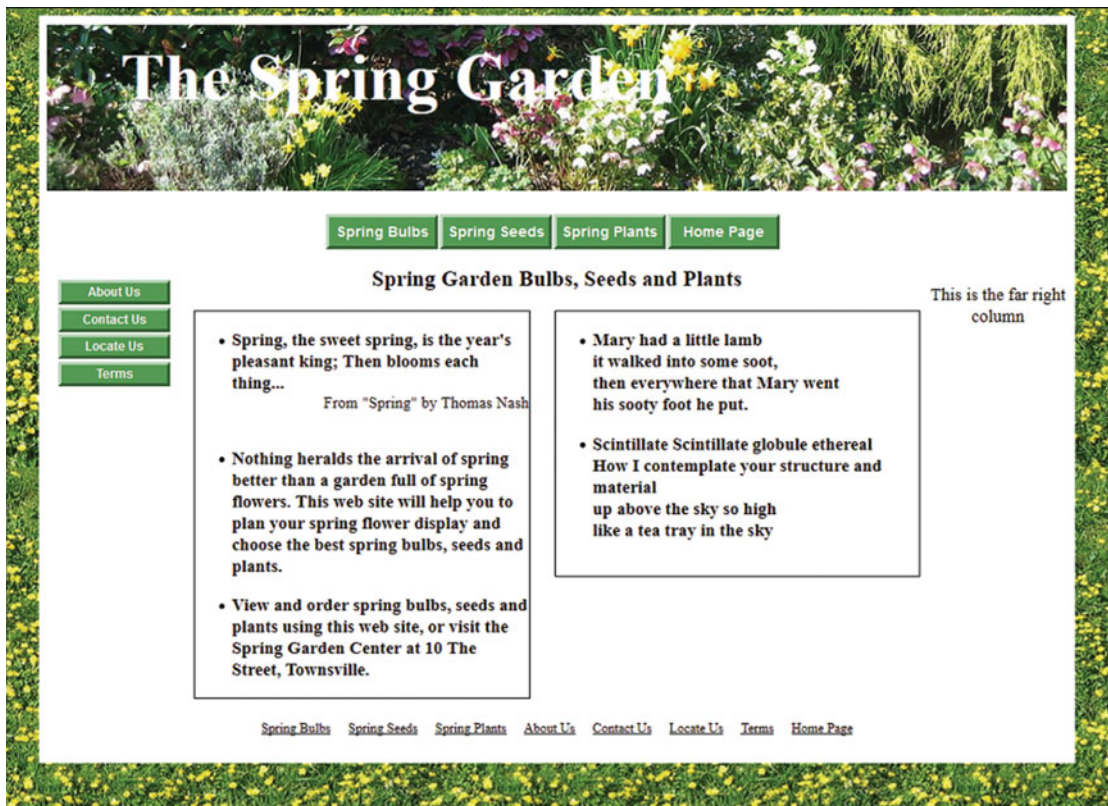


Figure 31-3. The temporary black border shows how the default bullets affect the layout of the text

With the page on the screen, use a ruler to measure the distance from the left border to the start of the word *Spring* in the first paragraph of the left column. Make a note of the distance and compare it with the distance after you have completed the next exercise.

Reduce the Gaps Caused by the Default Bullets

In code/Source view locate the internal style in the <head> section. Then add the lines shown bold in the next snippet of code:

```
<style type="text/css">
    .small {font-size:75%; color:black;}
    #midcol-left, #midcol-right {border:1px black solid;}
    #midcol-left li, #midcol-right li {padding-left:0; margin-left:-15px;}
</style>
```

Save the file.

With XAMPP and Apache running, use <http://localhost/Chapter-31/bullet-test.php> to view it in a modern browser; the result should be as shown in Figure 31-4. For clarity, only the central content of the web page is shown.

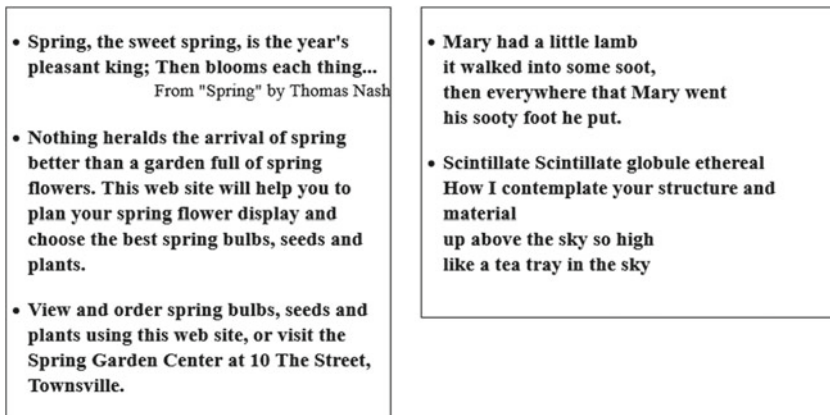


Figure 31-4. Showing the reduced margins and gaps

Explanation of the Code

#midcol-left li, #midcol-right li {padding-left:0; margin-left:-15px;}

The styling is applied to the `` tags within the columns.

We cannot not use **li {padding-left:0; margin-left:-15px;}** because it would have affected the menus that also contain `` tags. Therefore we must be very specific and apply the style only to the `` tags that are contained within the midcol-left and midcol-right sections.

Browsers automatically apply padding between the bullet and the text; by setting the padding to zero we can reduce the gap between the bullet and the text. Note that you cannot have negative padding.

We then used negative left margins **margin-left:-15px;** to pull the bullets and text to the left.

Bulleted paragraphs are usually preceded by a bold heading; we will now add the headings.

Add Bold Headings to the Bulleted Paragraphs

Beginners are sometimes unsure where headings should be located in the HTML structure. This exercise emphasizes that headings should be outside the list, otherwise the headings will also be prefixed by bullets. To demonstrate this, please follow these steps:

1. Open *bullet-test.php* in your text editor, just beneath the line `<div id="midcol-left">` insert the line shown in bold in the next code snippet.

```
<div id="midcol-left">
    <h2 class="left"><strong>The Joy of Spring</strong></h2>
</div>
```

Note that the heading must be located outside the `` tag. If it is placed within the `` tag, the heading will have a bullet and it will be indented. Also, the page will not validate because headings are not permitted within `` tags.

2. We will now put a heading in the midcol-right column. Just below the line `<div id="midcol-right">` enter the line shown in bold:


```
<div id="midcol-right">
    <h2 class="left"><strong>Poets' Corner</strong></h2>
</div>
```

3. Save the page.
4. With XAMPP and Apache running, use `http://localhost/Chapter-31/bullet-test.php` to view it in a modern browser, the result should be as shown in Figure 31-5. For clarity, only the central content of the web page is shown.

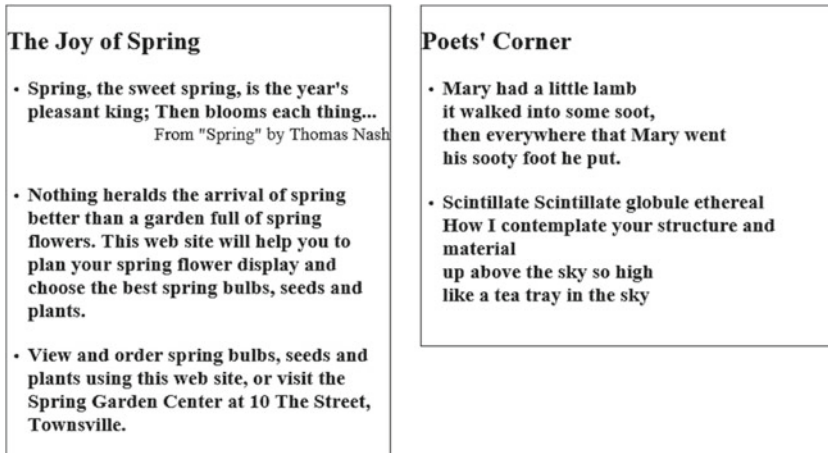


Figure 31-5. Displaying a bold heading at the top of each column

If you wish to view the modified bulleted paragraphs without the temporary border, open the file `bullet-test.php` in your text editor, and in the internal style within the head section, comment-out the borders as shown in the next snippet of code. Then view the modified page.

```
<style type="text/css">
    .small {font-size:75%; color:black;}
    <!-- #midcol-left, #midcol-right {border:1px black solid;}-->
    #midcol-left li, #midcol-right li {padding-left:0; margin-left:-15px;}
</style>
```

More Bullet Styles Using CSS

The default bullet is a solid round bullet known as a disc. With CSS you can create various bullets as shown in Figure 31-6.

Unordered bullets

Disc	Circle	Square
<ul style="list-style-type: none"> List item 1 List item 2 List item 3 	<ul style="list-style-type: none"> List item 1 List item 2 List item 3 	<ul style="list-style-type: none"> List item 1 List item 2 List item 3

Ordered bullets

Decimal	Upper Alpha	Lower alpha
<ol style="list-style-type: none"> List item 1 List item 2 List item 3 	<ol style="list-style-type: none"> List item 1 List item 2 List item 3 	<ol style="list-style-type: none"> List item 1 List item 2 List item 3

Figure 31-6. The six most popular bullets

The default bullet is a disc and this is produced by the following code:

```
<ul>
  <li>List item 1</li>
  <li>List item 2</li>
  <li>List item 3</li>
</ul>
```

No CSS is required for the default disc bullet.

Unordered lists: To produce the other unordered lists you must insert a *class* into the `` tag as follows:

```
<ul class="circle">
  <li>List item 1</li>
  <li>List item 2</li>
  <li>List item 3</li>
</ul><
```

The style must be specified either in the style sheet or in an internal style as follows:

```
.circle {list-style-type:circle;}

<ul class="square">
  <li>List item 1</li>
  <li>List item 2</li>
  <li>List item 3</li>
</ul>
```

The corresponding CSS style will be `.square {list-style-type:square;}`

Ordered lists: To use ordered lists the `` tag must be replaced by `` tags.

To style the ordered bullets you must insert a *class* into the `` tag as follow:

```
<ol class="decimal">
  <li>List item 1</li>
  <li>List item 2</li>
  <li>List item 3</li>
</ol>
```

The corresponding CSS style will be `.decimal {list-style-type:decimal;}`

```
<ol class="upper-alpha">
  <li>List item 1</li>
  <li>List item 2</li>
  <li>List item 3</li>
</ol>
```

The corresponding CSS style will be `.upper-alpha {list-style-type:upper-alpha;}`

```
<ol class="lower-alpha">
  <li>List item 1</li>
  <li>List item 2</li>
  <li>List item 3</li>
</ol>
```

The corresponding style will be `.lower-alpha {list-style-type:lower-alpha;}`

Several other styles are available, and it is even possible to use an image for the bullets, but these are beyond the scope of this book for beginners.

■ **Note** You will see from the above snippets of code that a style applied to the `` or `` tags will affect all the items in the list. It is not necessary to add a class to each of the listed items; it is sufficient to add the class to the `` or `` tag like this `<ul class="square">`.

Summary

You created a test page to see how bullets can be styled by using CSS. You were given instructions on how to style the bullets so that you can create different bullet forms such as circle, square, upper alpha, and lower alpha. In the next chapter you will learn how to indicate which menu button has been clicked.

CHAPTER 32



Indicating Which Horizontal Menu Button Has Been Clicked

Users should be able to navigate to another page from anywhere within the site; this is also vital for search engine optimization. It is also important that users should know exactly which page they are currently viewing. The technique described in this chapter is suitable for a small website (say up to 12 pages). For a large website you could generate the menu on each page using PHP *includes* and then use an interactive PHP technique to highlight the clicked menu button. That PHP technique is beyond the scope of this book for absolute beginners. There are two indicators that will tell the user which page they are on, and which menu button has been clicked:

1. A clear title at the top of each page.
2. An indication of which menu button was clicked to access the page.

We can assume that web designers normally use the first indicator; therefore, in this chapter you will be shown how to add the second indicator. This chapter contains the following sections:

- View the pages in the downloaded folder
- Tweak the menu so that it can be styled to indicate the selected menu button
- Make a small change to the home page styling to match the revised menu
- Add the indicator style to the other pages
- Emphasize the page indicator
- Summary

View the Pages in the Downloaded Folder

The downloadable folder for this chapter contains the files and images required for this project. Please follow these steps:

1. Download the folder named *Chapter-32* and unzip it in your *htdocs* folder.

You will see another folder within the *Chapter-32* folder named *Completed-files*; ignore it for the moment, but you may wish to examine it and compare the pages with your own completed project.

2. Start XAMP and Apache, then open the *index.php* file using <http://localhost/Chapter-32>

Click each menu button and you will see that the title on each page fulfils the requirement for the first indicator. However, there is nothing to indicate that a particular menu button has been clicked. See Figure 32-1.



Figure 32-1. The page heading is displayed but there is no indication of which menu button has been clicked

Mr. Jeff Eastwood kindly gave me permission to adapt his pictures and his website that I produced for him. You can view his website at <http://www.clean-living-sw.co.uk/>

Tweak the Menu So That It Can Be Styled to Indicate the Selected Menu Button

The menu must be tweaked to identify each button; by doing this, an additional internal style on each page can underline the button that has been clicked. To change the menu, please follow these steps:

1. Open the folder *Chapter-32*.
2. Open the *includes* folder.
3. Open the file *menu.html* file in your text editor.
4. Insert the code shown in bold type in the next snippet of code:

```
<nav>
  <ul>
    <li class="btn tab4"><a href="carpet-cleaning-other.php">Other Services</a></li>
    <li class="btn tab3"><a href="carpet-cleaning-contact.php">Contact Us</a></li>
```

```

    <li class="btn tab2"><a href="carpet-cleaning-commend.php">Commendations</a></li>
    <li class="btn tab1"><a href="index.php">Home Page</a></li>
</ul>
</nav>

```

By giving each menu button a *class* we can single out each button for individual treatment.

■ **Note** The above code illustrates that you can give an element two class attributes: in this example the attributes are *btn* and *tab1*, *btn* and *tab2*, etc. Make sure you have a space between each attribute.

5. Save the file *menu.html*.

Next, we need to make a small change to home page *index.html*

Make a Small Change to the Home Page Styling to Match the Revised Menu

We will now tweak the internal style in the home page by inserting an indicator style. This will add a black underline to the home page button, which was given the class **tab1**).

Please follow these steps:

1. In your folder *Chapter-32*, open the file *index.php* in your text editor. In the `<head>` section add the line of code shown bold in the next snippet of code.

```

<style type="text/css">
  li {margin-bottom:10px;}
  #mid-left-col {margin-left:0; width:42%;}
  #mid-right-col {float:left; margin-left:0; width:50%;}
  #mid-right-col ul {font-size:110%;}
  .btn.tab1 {border-bottom:4px black solid; border-top:none; border-left:none; ↵
border-right:none;}
</style>

```

This small change will add a black underline to the home page menu button.

2. Save the file,

To view the effect of these changes to the home page, with XAMPP and Apache running type the following URL in the address field of a modern browser:

<http://localhost/Chapter-23/>

The home page should look like Figure 32-2,



Figure 32-2. A black underline indicates that the Home page button has been clicked

We must now insert a new line of code into the internal style of the other pages.

Add the Indicator Style to the Other Pages

In the folder *Chapter-32*, open the file *carpet-cleaning-commend.php* in code/source view in your text editor. In the `<head>` section add the line of code shown bold in the next snippet.

```
<style type="text/css">
  li {margin-bottom:10px;}
  #midcol {margin-left:150px; margin-right:220px; text-align:left;}
  h3 {text-align:center; font-weight:bold;}
  .btn.tab2 {border-bottom:4px black solid; border-top:none; border-left:none; ↵
border-right:none;}
</style>
```

■ **Note** In each page the style refers to a different tab: in the home page it was tab1, in this page it is tab2. Therefore you could copy and paste the code into other pages, provided you remember to change the tab number to match the page.

Save the file *carpet-cleaning-commend.php*

In the folder *Chapter-32*, open the file *carpet-cleaning-contact.php* in code/source view in your text editor. In the <head> section add the line of code shown bold in the next snippet.

```
<style type="text/css">
  l#midcol {margin-left:220px; margin-right:150px;}
  h3 {text-align:center; font-weight:bold;}
  .btn.tab3 {border-bottom:4px black solid; border-top:none; border-left:none; ↵
    border-right:none;}
</style>
```

Save the file *carpet-cleaning-commend.php*

In your folder *Chapter-32*, open the file *carpet-cleaning-other.php* in code/source view in your text editor. In the <head> section add the line of code shown bold in the next snippet.

```
<style type="text/css">
  #midcol {margin-left:250px; margin-right:350px; text-align:left;}
  h3 {text-align:center; font-weight:bold;}
  .btn.tab4 {border-bottom:4px black solid; border-top:none; border-left:none; ↵
    border-right:none;}
</style>
```

Save the file *carpet-cleaning-other.php*

In your folder *Chapter-32*, open the file *carpet-cleaning-contact.php* in code/source view in your text editor. In the <head> section add the line of code shown bold in the next snippet.

```
<style type="text/css">
#midcol {margin-left:220px; margin-right:150px;}
h3 {text-align:center; font-weight:bold;}
.btn.tab3 {border-bottom:4px black solid; border-top:none; border-left:none; ↵
  border-right:none;}
</style>
```

Save the file *carpet-cleaning-contact.php*

When you have changed the internal style in all four pages, start XAMPP and Apache then test them in your browser using <http://localhost/Chapter-32/>

Now click any of the other buttons to see how the underline appears on the clicked button. The indicator on the clicked button together with the page title will leave the user in no doubt as to which page he or she is viewing.

Figure 32-3 shows the underline on the “Other Services” page.

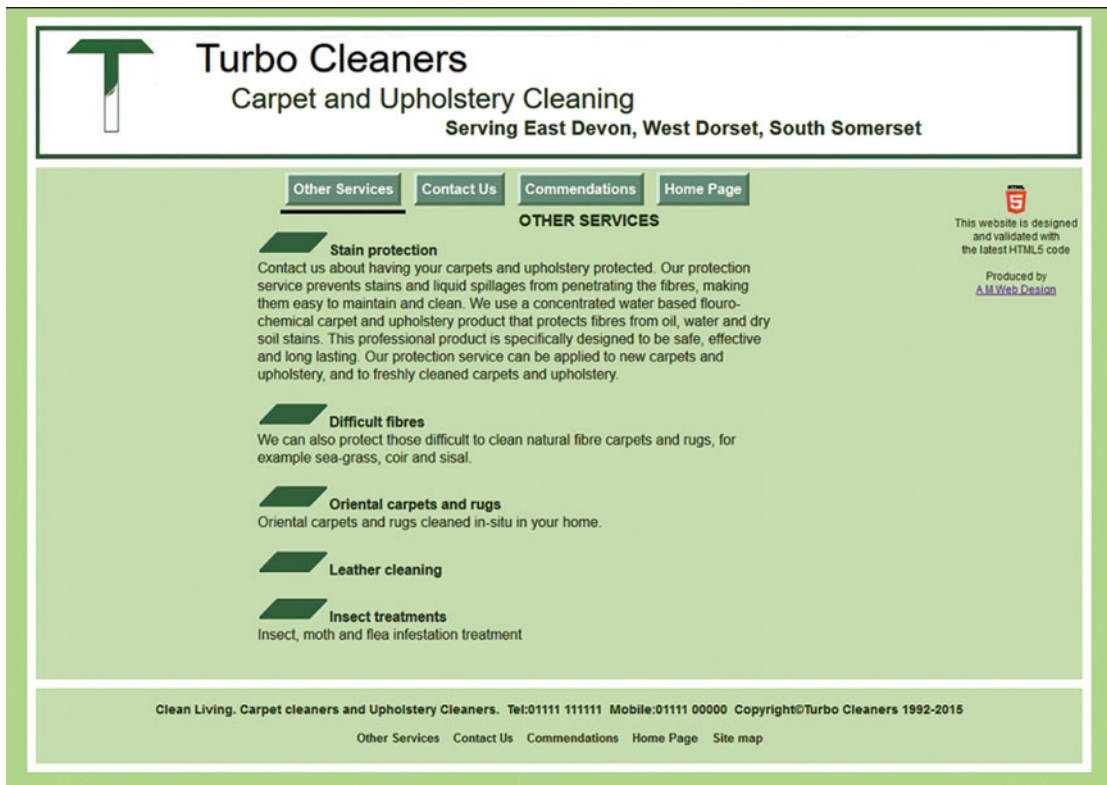


Figure 32-3. The Other Services page

The next section briefly introduces an alternative or additional way of emphasizing the page indicator.

Emphasize the Page Indicator

The page indicator can be made even more distinctive by changing the color of the button for the selected page. When combined with an underline, the page indicator becomes even more obvious to the user. To achieve this, you would simply add an extra line of code in the internal style on each page.

To illustrate this, try adding the new line of code to the home page, but please take the following steps:

1. Open the home page *index.html* in your text editor.
2. Use *Save as* to make a copy named *index-test.html*
3. In the <head> section of *index-test.html* insert the line shown in bold in the next snippet:

```
<style type="text/css">
  li {margin-bottom:10px;}
  #mid-left-col {margin-left:0; width:42%;}
  #mid-right-col {float:left; margin-left:0; width:50%;}
  #mid-right-col ul {font-size:110%;}
  .btn.tab1 a {background:olive; border:4px outset #a2d0a2;}
</style>
```

4. Save the file.

Start XAMPP and Apache and view the page by entering: `http://localhost/Chapter-32/index-test.php` in the address field of your browser.

The page should look like Figure 32-4.



Figure 32-4. The emphasized page indicator

You would need to add the new line of code in each page of the website; remember to change the tab number each time to match the page.

If you prefer the color change without an underline, just delete or comment-out the following line:

```
.btn.tab1 {border-bottom:4px black solid; border-top:none; border-left:none; ↵
border-right:none;}
```

Treat the files in the *Completed-files* folder as new templates that you can adapt for your own use for almost any kind of service product.

Summary

In this chapter you discovered how to use CSS styling to indicate which page the user has selected. You inserted a line of code that provided an indicator on the menu, and this indicator style added a black line beneath the button relating to that page. You also tested an alternative or additional way of emphasizing the page indicator by using a different color for the clicked button.

In the next chapter you will learn how to apply these techniques to a similar website with vertical menu buttons.

CHAPTER 33



Indicating Which Vertical Menu Button Has Been Clicked

The previous chapter described how to provide a page indicator on a horizontal menu button. This chapter uses a vertical menu, which presents us with a problem; the user might be uncertain whether a black line refers to the menu button above the line, or the menu button below the line. The solution is to add a line around the entire clicked button.

For this chapter, Mr. Jeff Eastwood kindly gave me permission to adapt his pictures and his website. I produced his website and you can view it at <http://www.clean-living-sw.co.uk>

This chapter is similar in many ways to the previous chapter and it contains the following sections:

- View the pages in the downloaded files
- Make minor changes to the menu to indicate that a particular button has been clicked
- Change the menu on the other pages
- Emphasize the menu's page indicator
- Summary

View the Pages in the Downloaded Files

To view the pages in the downloadable folder and use them in this chapter's project, please follow these steps:

1. Download the zip folder named *Chapter-33* and unzip it in your *htdocs* folder.

You will see another folder within the *Chapter-33* folder named *Completed-files*; ignore it for the moment, but you may wish to examine it when you have finished this exercise and then compare the results.

2. Start XAMPP and Apache, then open the *index.php* file using <http://localhost/Chapter-33>
3. Click each menu button and you will see *no* indication that a particular button has been clicked.

The only indicator is the title on the page. See Figure [33-1](#).

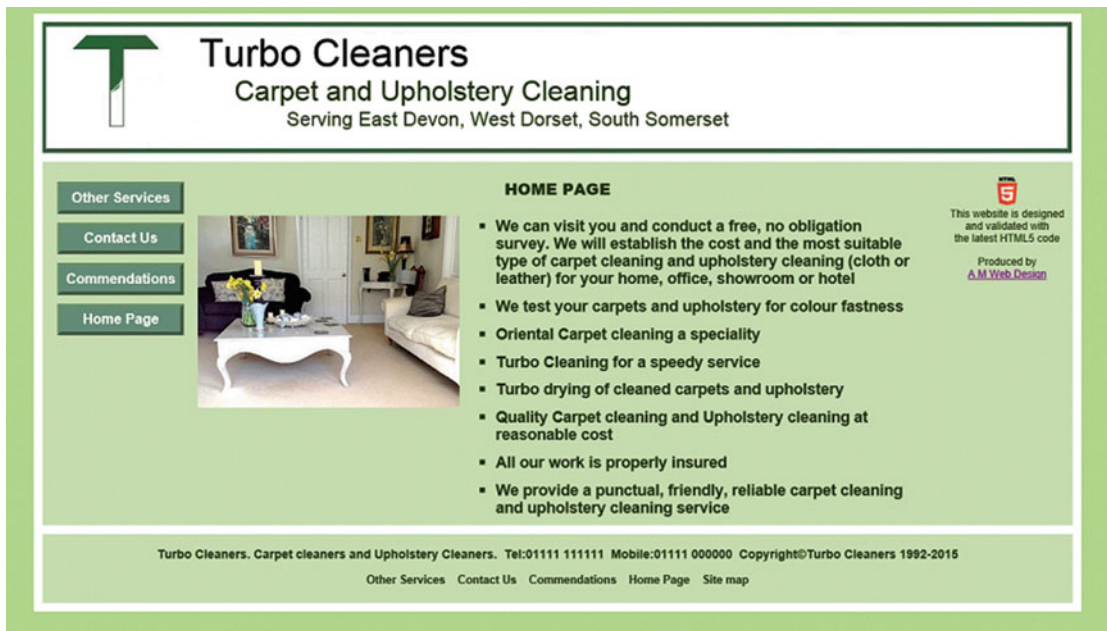


Figure 33-1. Showing no indication as to which menu button has been clicked

The code for the home page is given in Listing 33-1.

Listing 33-1. The home page (*index.php*).

```
<!DOCTYPE html>
<html>
<head>
  <title>Carpet cleaning, Carpet cleaners, Upholstery cleaning, Upholstery cleaners,
    Devon, Dorset, Somerset </title>
  <meta name="description" content="Carpet cleaning, Carpet cleaners, Upholstery cleaning,
    Upholstery cleaners, Devon, Dorset, Somerset">
  <meta name="keywords" content="Carpet cleaning, Carpet cleaners, Upholstery cleaning,
    Upholstery cleaners, Devon, Dorset, Somerset">
  <meta charset="utf-8">
  <link rel="stylesheet" type="text/css" href="carpet-cleaners.css">
  <style type="text/css">
    li {margin-bottom:10px;}
    #mid-left-col {margin-left:20px; width:305px;}
    #mid-right-col {float:left; margin-left:0; width:60%;}
    #mid-right-col ul {font-size:110%;}
  </style>
</head>
```

```

<body>
<div id="wrapper">
    <?php include 'includes/header.inc'; ?>
    <div id="mainpanel">
        <br>
        <div id="leftcol">
            <?php include 'includes/vmenu.html'; ?>
        </div>
        <div id="rightcol">
            <?php include 'includes/infocol.inc'; ?>
        </div>
        <div id="midcol">
            <h2><strong>HOME PAGE</strong></h2><br>
            <div id="mid-left-col">
                <p class="cntr">
                    <br>
            </div>
            <div id="mid-right-col">
                <ul>
                    <li><strong>We can visit you and conduct a free, no obligation survey. We will
                        establish the cost and the most suitable type of carpet cleaning and upholstery
                        cleaning (cloth or leather) for your home, office, showroom or hotel</strong></li>
                    <li><strong>We test your carpets and upholstery for colour fastness</strong></li>
                    <li><strong>Oriental Carpet cleaning a speciality</strong></li>
                    <li><strong>Turbo Cleaning for a speedy service</strong></li>
                    <li><strong>Turbo drying of cleaned carpets and upholstery</strong></li>
                    <li><strong>Quality Carpet cleaning and Upholstery cleaning at reasonable
                        cost</strong></li>
                    <li><strong>All our work is properly insured</strong></li>
                    <li><strong>We provide a punctual, friendly, reliable carpet cleaning and
                        upholstery cleaning service</strong></li>
                </ul>
            </div><br>
        </div><br><br>
        <footer>
            <?php include 'includes/footer.html'; ?>
        </footer><br></div>
    </div><br>
    <br class="clear">
</body>
</html>

```

Make Minor Changes to the Menu to Indicate That a Particular Button Has Been Clicked

We need to tweak the menu so that each button can be identified as a target for styling; please follow these steps:

1. In your folder *Chapter-33*, open the *includes* folder, then open the *vmenu.html* file and add the code shown in bold type in the next snippet of code:

```
<nav>
  <ul>
    <li class="btn tab4"><a href="carpet-cleaning-other.php">Other Services</a></li>
    <li class="btn tab3"><a href="carpet-cleaning-contact.php">Contact Us</a></li>
    <li class="btn tab2"><a href="carpet-cleaning-commend.php">Commendations</a></li>
    <li class="btn tab1"><a href="index.php">Home Page</a></li>
  </ul>
</nav>
```

By giving the menu buttons an additional *class* we can single them out for special treatment.

■ **Note** This snippet above illustrates how you can give *two* class attributes to an element. In this example the two attributes are *btn* and *tab1*, *btn* and *tab2*, etc. Make sure you have a space between each attribute as shown in the above snippet.

2. In your folder *Chapter-33*, open the file *index.php* in your text editor. In the `<head>` section add the line of code shown bold in the next snippet of code.

```
<style type="text/css">
  li {margin-bottom:10px;}
  #mid-left-col {margin-left:0; width:42%;}
  #mid-right-col {float:left; margin-left:0; width:50%;}
  #mid-right-col ul {font-size:110%;}
  .btn.tab1 {border-top:3px black solid; border-right:3px black solid; ↵
border-bottom:3px black solid;border-left:3px black solid;}
</style>
```

By adding one line of code we have persuaded the home page button to display a black border all around the menu button as shown in Figure 33-2.

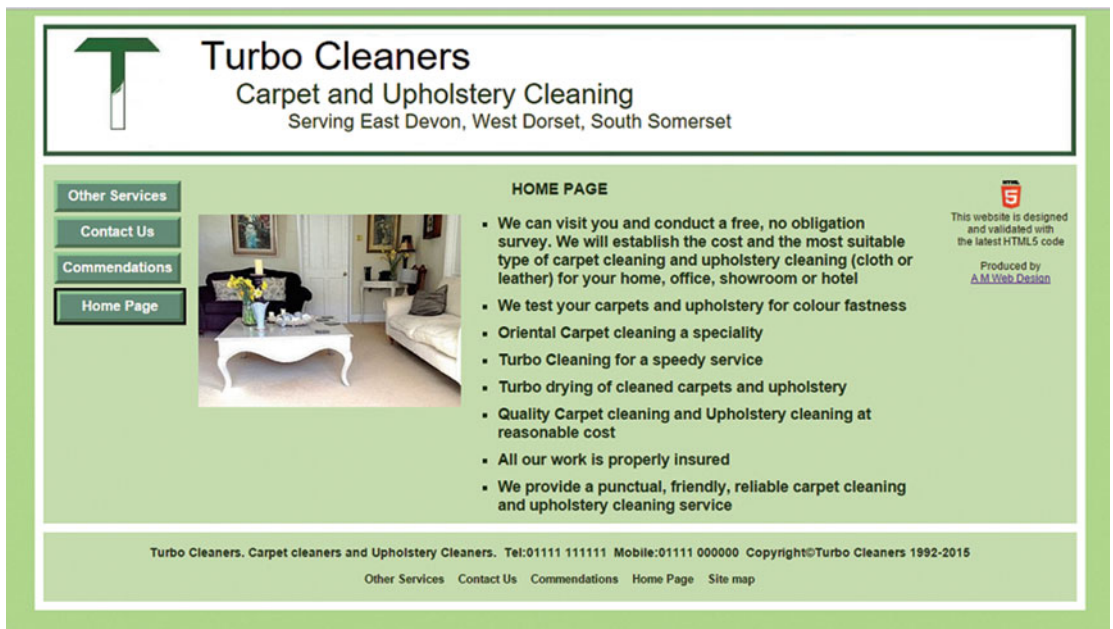


Figure 33-2. Showing the black border around the home page button

We must now repeat this exercise for each page.

Change the Menu on the Other Pages

In your folder *Chapter-33*, open the file *carpet-cleaning-commend.php* in code/source view in your text editor. In the <head> section add the line of code shown bold in the next snippet. Note that the button tab is now **tab2**.

```
<style type="text/css">
  li {margin-bottom:10px;}
  #midcol {margin-left:150px; margin-right:220px; text-align:left;}
  h3 {text-align:center; font-weight:bold;}
  .btn.tab2 {border-top:3px black solid; border-right:3px black solid; ↵
border-bottom:3px black solid; border-left:3px black solid;}
</style>
```

In your folder *Chapter-33*, open the file *carpet-cleaning-contact.php* in code/source view in your text editor. In the <head> section add the line of code shown bold in the next snippet. Note that the tab is now **tab3**.

```
<style type="text/css">
  l#midcol {margin-left:220px; margin-right:150px;}
  h3 {text-align:center; font-weight:bold;}
  .btn.tab3 {border-top:3px black solid; border-right:3px black solid; ↵
border-bottom:3px black solid; border-left:3px black solid;}
</style>
```

In your folder *Chapter-33*, open the file *carpet-cleaning-other.php* in code/source view in your text editor. In the <head> section add the line of code shown bold in the next snippet. Note that the tab is now tab4.

```
<style type="text/css">
#midcol {margin-left:250px; margin-right:350px; text-align:left;}
h3 {text-align:center; font-weight:bold;}
.btn.tab4 {border-top:3px black solid; border-right:3px black solid; ↵
border-bottom:3px black solid; border-left:3px black solid;}
</style>
```

When you have changed the style section in all four pages, start XAMPP and Apache, then test them in your browser using <http://localhost/Chapter-33/>

Emphasize the Menu's Page Indicator

As described in the previous chapter, the page indicator can be made even more distinctive by changing the color of the button for the selected page. When combined with a black line on four sides of the button, the page indicator becomes even more obvious to the user. To achieve this, you would simply add an extra line of code in the internal style on each page.

To illustrate this, try adding the new line of code to the home page using the following steps:

1. Open the home page *index.html* in your text editor.
2. Use *Save as* to make a copy named *index-test.html*.
3. In the <head> section of *index-test.html* insert the line shown in bold in the next snippet:

```
<style type="text/css">
li {margin-bottom:10px;}
#mid-left-col {margin-left:0; width:42%;}
#mid-right-col {float:left; margin-left:0; width:50%;}
#mid-right-col ul {font-size:110%;}
.btn.tab1 {border-top:3px black solid; border-right:3px black solid; ↵
border-bottom:3px black solid; border-left:3px black solid;}
.btn.tab1 a {background:olive; border:4px outset #a2d0a2;}
</style>
```

4. Save the file.

Start XAMPP and Apache and view the page by entering this:

<http://localhost/Chapter-32/index-test.php> in the address field of your browser.

The page should look like Figure 33-3.



Figure 33-3. The home page showing additional emphasis on the home page menu button

If you wish to use the color change, add the extra line of code to the internal style in the file *index.php* and also in the other three pages. You may prefer to use the color change and dispense with the black lines around the button; to achieve this, just delete or comment-out the line of code that creates the black border.

Summary

You learned how to indicate which page the user is viewing by means of a black line around the menu button that was clicked to access that page. You also learned how to give additional emphasis to the clicked menu button by using a background color change. In the next chapter you will learn how to create a double row of menu buttons for a website with many more pages.

CHAPTER 34



Creating Multi-row Menus and Picture Galleries

In this chapter you will create a menu with two rows, and a picture gallery with four rows of pictures. The chapter consists mainly of description and explanation with very little project work.

This chapter contains the following sections:

- Create a double row of menu buttons
- To accommodate an odd number of buttons
- Create a gallery with four rows of pictures
- Summary

The chapter uses PHP *includes*, and the downloadable folder must be unzipped into your *htdocs* folder. Please follow these steps:

1. In your *htdocs* folder create a new chapter named *Chapter-34*.
2. Download the zip file for *Chapter-34* from the book's page at Apress.com.
3. In your *Chapter-34* folder, unzip the folder. The unzipped folder contains a website with a double-row menu consisting of five buttons per row.
4. Use `http://localhost/Chapter-34/` to view the home page in a modern browser.

You will see the double-row menu as shown in Figure 34-1.

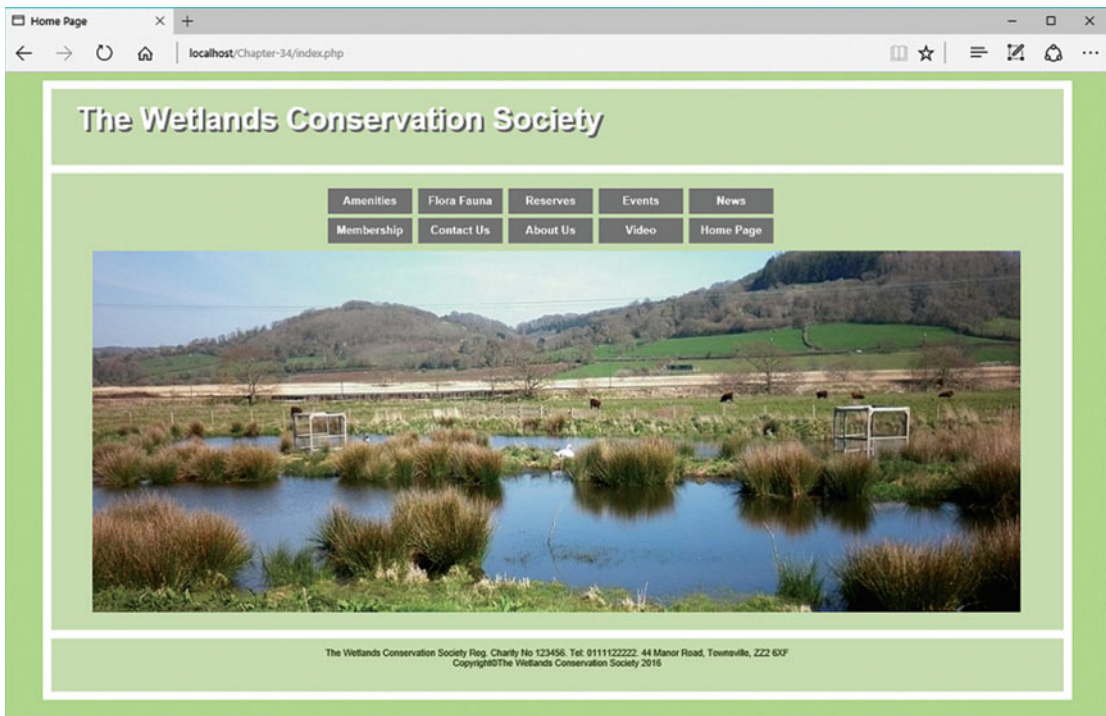


Figure 34-1. Showing a double row of menu buttons in the home page

Create a Double Row of Menu Buttons

A single row of horizontal menu buttons can limit the number of buttons to about 8 or 10. A double row of menu buttons can solve this problem, giving up to 20 buttons. This avoids the need to employ a less user-friendly drop-down menu. A 10-button menu is used in this chapter, but this is sufficient to demonstrate the double-row technique.

The code for this home page is surprisingly short and simple; the two rows of buttons are created by the code shown bold in Listing 34-1.

Listing 34-1. The code for the home page.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Home Page</title>
    <meta charset=utf-8>
    <link href="style-roll-2row.css" rel="stylesheet" type="text/css">
    <style>
      img {margin-top:-15px;}
    </style>
  <!--[if lte IE 8]>
    <script src="html5.js">
    </script>
  <![endif]-->
</head>
```

```

<body>
<div id="wrapper">
    <?php include 'includes/header.inc'; ?>
<div id="mainpanel">
    <?php include("includes/menu-row1.inc"); ?>
    <?php include("includes/menu-row2.inc"); ?>
<p>&nbsp;  </p>
    <br>
</div>
    <br>
    <div id="midcol">
        <p></a></p>
    </div>
<footer>
    <?php include 'includes/footer.html'; ?>
</footer><br>
</div><!--wrapper div finishes here -->
</body>
</html>

```

Explanation of the Code

```

<?php include("includes/menu-row1.inc"); ?>
<?php include("includes/menu-row2.inc"); ?>

```

The double menu is created by including two separate menu files that are located in the *includes* folder. We need nothing very special to position the rows on the page because they are already structured one above the other in the home page code. Be very careful to indicate where the external files are stored; in this case they are in a folder called *includes*.

The code for the included menus is given in Listings 34-2 and 34-3. Only five pages are provided for this website to save space and time, the other pages have dead links, that is, href="#".

Listing 34-2. The top-row menu

```

<nav>
    <ul>
        <li class="btn"><a href="wetlands-amenities.php">Amenities</a></li>
        <li class="btn"><a href="#">Flora Fauna</a></li>
        <li class="btn"><a href="#">Reserves</a></li>
        <li class="btn"><a href="wetlands-events.php">Events</a></li>
        <li class="btn"><a href="#">News</a></li>
    </ul>
</nav>

```

Listing 34-3. The bottom-row menu.

```

<nav>
    <ul>
        <li class="btn"><a href="wetlands-membership.php">Membership</a></li>
        <li class="btn"><a href="wetlands-contact.php">Contact Us</a></li>
    </ul>

```

```

    <li class="btn"><a href="#">About Us</a></li>
    <li class="btn"><a href="#">Video</a></li>
    <li class="btn"><a href="index.php">Home Page</a></li>
</ul>
</nav>

```

The CSS Code

The menu styling is shown in bold in the main style sheet given in Listing 34-4.

Listing 34-4. Styling the home page.

```

body {background:#88CCFF;}
#wrapper {margin:auto; max-width:1200px; min-width:1100px; background:#A6CCFF;
    border:10px white solid; font-size:110%; background-color:#CEE1BA;}
header,nav,article,section,footer {display:block;}
header {margin-top:0; border-bottom:10px white solid; height:90px;
background-color:#CEE1BA;}
h1 {position:absolute; top:10px; margin-left:30px; font-size:250%; color:white;
    font-weight:bold; text-shadow:3px 3px #707070;}
h2 {font-size:150%; text-align:center;}
p {font-size:120%; text-align:center;}
a {font-size:120%;}
figure {float:left; margin-left:15px;}
/*add a left col*/
#leftcol {float:left; width:15px;}
#rightcol {float:right; width:150px; height:15px;}
#midcol {margin-left:18px; margin-right:20px;}
br.clear {clear:both}
#midcol-left {float:left; width:46%; font-weight:bold;}
#midcol-right {float:right; width:50%; font-weight:bold;}
.lft {text-align:left;}
/*set position of horizontal button menus*/
nav ul {float:left; width:750px; height:30px; margin-left:18%; text-align:center;
    padding-left:0; list-style-type:none; margin-top:0; margin-bottom:5px;}
/*set general horizontal menu button styles*/
nav ul li {display:inline-block;}
/* set general side button styles */
li.btn {width:100px; line-height:30px; margin-bottom:0; text-align:center;
    margin-left:3px;}
/* set general access (anchor) styles */
li.btn a {display:block; width:100px; color:white; background:gray; font-family:arial;
    font-size: small; font-weight:bold; text-decoration:none;}
/* mouseover */
li.btn a:hover {background:green;}
/* mousedown */
li.btn a:active {background:green;}
img {border:none;}
body {background:#BBD999; font-family:arial; font-size:small; color:#2B4103; margin:auto;
    padding:7px;}
footer {font-size:60%; border-top:10px white solid;}

```

Explanation of the Code

You have used most of the code before, but the style for the menus needs some explanation.

```
/*set position of horizontal button menus*/
nav ul {float:left; width:750px; height:30px; margin-left:18%; text-align:center; ↵
padding-left:0; list-style-type:none; margin-top:0; margin-bottom:5px;}
```

The menus are positioned horizontally on the page and their widths are set to accommodate the five buttons per row. To accommodate more buttons per row, increase the width and adjust the left margin.

The rows in the home page have an equal number of menu buttons. What would happen if we had an odd number of buttons? The next section describes a double-row menu with an unequal number of buttons.

To Accommodate an Odd Number of Buttons

We will now remove the video button to demonstrate the effect of an odd number of buttons.

Please follow these steps.

1. Open the *includes* folder.
2. Open the file *menu-row2.html* in your text editor.
3. Comment-out the line for the video button shown in bold in Listing 34-5.

Listing 34-5. Comment-out the video button.

```
<nav>
  <ul>
    <li class="btn"><a href="wetlands-membership.php">Membership</a></li>
    <li class="btn"><a href="wetlands-contact.php">Contact Us</a></li>
    <li class="btn"><a href="#">About Us</a></li>
    <!--<li class="btn"><a href="#">Video</a></li>-->
    <li class="btn"><a href="index.php">Home Page</a></li>
  </ul>
</nav>
```

4. Save the file.
5. Use *http://localhost/Chapter-34/* to view the home page in a modern browser.
6. Figure 34-2 shows the menus with an odd number of button.

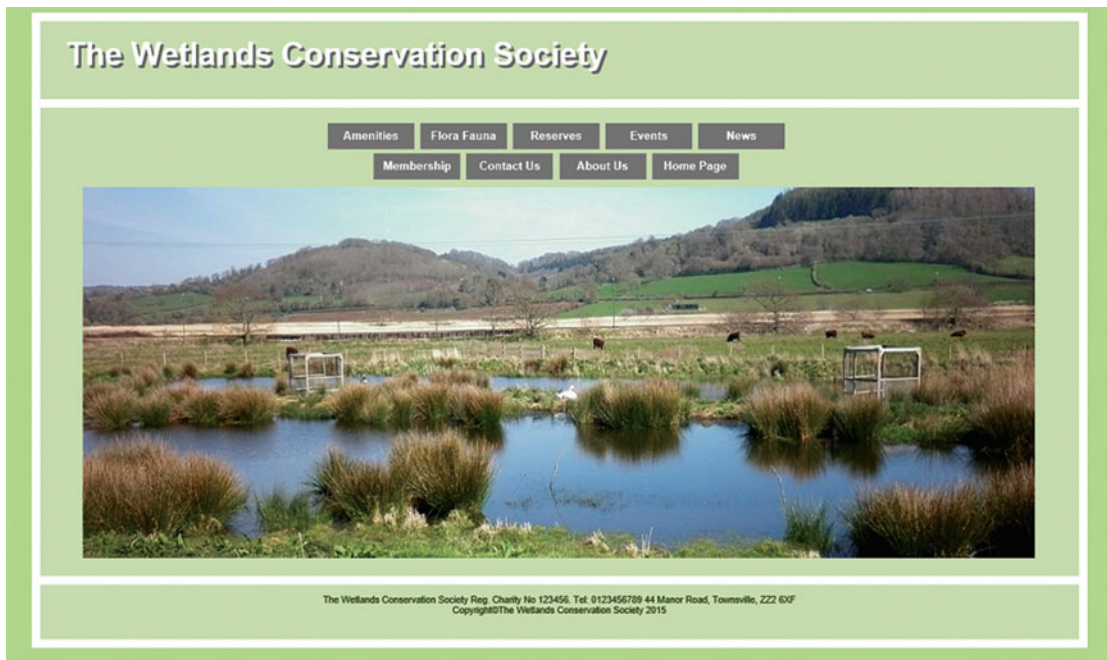


Figure 34-2. A double-row menu with an odd number of buttons

You can revert to an even number of buttons by removing the comment-out tags in the second row.

Create a Gallery with Four Rows of Pictures

Chapter 11 described a gallery with two rows of pictures; the downloaded folder for the current chapter contains a gallery with four rows of pictures. We will now examine the code that created this gallery.

Listing 34-6. Creating a four-row gallery.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Amenities in the Wetlands nature reserve</title>
    <meta charset=utf-8>
    <link href="style-roll-2row.css" rel="stylesheet" type="text/css">
    <style>
      img {margin-top:-15px;}
      #midcol-left, #midcol-mid, #midcol-right {width:345px; float:left; font-
        weight:normal;}
      #midcol-left {margin-left:60px;}
    </style>
    <!--[if lte IE 8]>
      <script src="html5.js">
      </script>
    <![endif]-->
  </head>
```

```

<body>
  <a id="top"></a>
<div id="wrapper">
  <?php include 'includes/header.inc'; ?>
  <div id="mainpanel">
    <?php include("includes/menu-row1.inc"); ?>
    <?php include("includes/menu-row2.inc"); ?>
    <p>&nbsp;</p>
    <br>
    <div id="midcol">
      <h2>The Wetlands Amenities</h2>

      <!--Insert four picture in the first column-->
      <div id="midcol-left">
        <p>
          <br>The Visitors' Centre has a range of helpful leaflets and
          souvenirs.<br><br></p>
        <p>
          <br>The Car park<br><br></p>
        <p>
          <br>The Classroom<br><br></p>
        <p>
          <br>Plenty of Sign Posts<br><br></p>
      </div>

      <!--Insert four picture in the second column-->
      <div id="midcol-mid">
        <p>
          <br>Flat dry paths lead to the various<br>bird hides.<br><br></p>
        <p>
          <br>The Toilets<br><br></p>
        <p>
          <br>Inside one of the Bird Hides<br><br></p>
        <p>
          <br>One of the Viewing Platforms<br><br></p>
      </div>

```



```

<!--Insert four picture in the third column-->
<div id="midcol-right">
  <p>
    <br>View of a pond from one of the hides.<br>(Overlooks the tram
      track).<br><br></p>
  <p>
    <br>Inside the Visitors' Centre (1)<br><br></p>
  <p>
    <br>Inside the Visitors' Centre (2)<br><br></p>
  <p>
    <br>Star Gazing Area<br><br></p>
</div>
</div>
</div><!--content div finishes here-->
<br class="clear">
<footer>
  <?php include 'includes/footer.html'; ?>
</footer>
<br>
</div><!--wrapper div finishes here -->
<br>
</body>
</html>

```

Explanation of the Code

Chapter 11 described how to produce a picture gallery with two rows by creating the rows and then inserting four pictures in each row. This chapter describes a different way of constructing a gallery that is better suited to a gallery with more than two rows. This technique creates three columns, and then it inserts four pictures vertically into each column. In Listing 34-6 you will see that there are three columns, and each column is preceded by a comment such as `<!--Insert four pictures in the first column-->`. The columns are identified with the names *midcol-left*, *midcol-mid*, and *midcol-right*. These are styled by the internal style shown in Listing 34-7.

Listing 34-7. The internal style that formats the columns.

```

<style>
  img {margin-top:-15px;}
  #midcol-left, #midcol-mid, #midcol-right {width:345px; float:left; font-weight:normal;}
  #midcol-left {margin-left:60px;}
</style>

```

You may recall that Chapter 11 stated that gallery pictures must be a uniform size, and this also applies to the method described in this chapter. The pictures are all 320 pixels wide, but the height is not quite as critical using the new method; the height varies slightly from 339 pixels to 344 pixels.

The gallery page is illustrated in Figure 34-3.

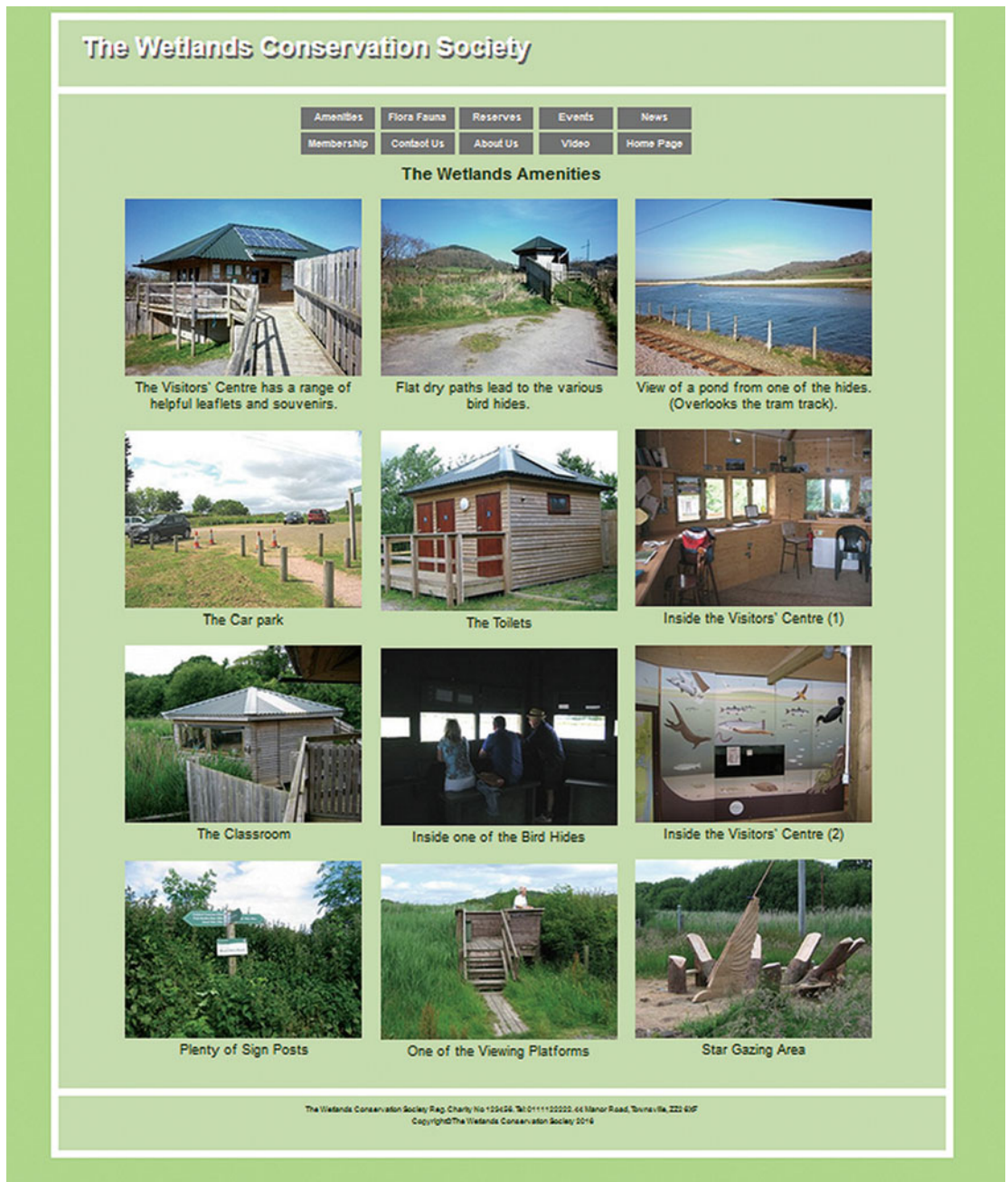


Figure 34-3. The four-row gallery

Summary

In this chapter you discovered how to create a web page with a double-row menu. You then experimented with a double-row menu with an odd number of buttons. A four-row picture gallery was described that used a technique that differed from the method used in [Chapter 11](#).

In the next chapter you will learn how to create websites for mobile devices such as tablets and smartphones.

CHAPTER 35



Building Responsive Websites for Mobile Devices Part 1

The great majority of websites designed for desktop computers can be viewed on a tablet and a smartphone; however, problems may be revealed when the site is displayed on the smaller devices.

Figure 35-1 shows a desktop website on an Android smartphone in landscape mode.

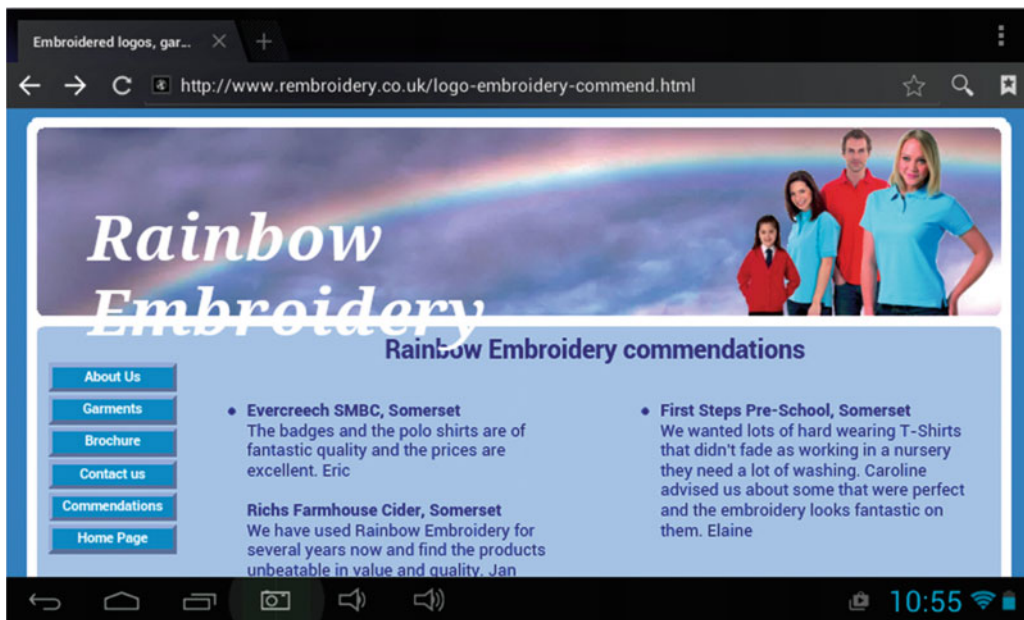


Figure 35-1. The heading text is too big for a smart phone, so the word Embroidery drops down a line

The appearance may be unsatisfactory on a smartphone for the following reasons:

- The pages may be too wide, causing the user to scroll horizontally.
- Text may be tiny and difficult to read or too large and out of position.

- Menu buttons may be too small and too close together, making it difficult to touch them accurately.
- The mobile device's screen may appear to be cluttered with unnecessary elements.

Many of these problems can be solved if viewers use the scroll, zoom, and landscape facility on their phones. However, the basic problems still remain because smartphone users don't like horizontal scrolling, zooming, and switching to landscape view; we should make the users' experience as easy and pleasurable as possible. This chapter describes a remedy for these problems.

■ **Note** To simplify the code and to save space, no Internet Explorer 8 conditionals are included in this chapter or in Chapters 36, 37, and 38. Therefore, the examples in these chapters will not work in Internet Explorer 8 or in earlier versions of Internet Explorer.

This chapter has the following sections:

- Definitions
- Is it worth all the effort?
- Different websites? Or different style sheets?
- The responsive solution
- The viewport statement
- Media queries and the responsive CSS style sheet
- Responsive websites use proportional dimensions
- Using proportional dimensions for the elements on RWD pages
- Responsive images
- The responsive navigation menu
- View the other responsive pages
- Testing responsive websites
- Summary

Definitions

Mobile: A tablet or smartphone capable of viewing web pages.

Device: Any piece of equipment for viewing websites.

Viewport: The size of a browser's window (within the borders and scroll bars) on a mobile device.

Screen size: The size of a device's screen in pixels.

Breakpoint: The viewport width at which the appearance of a web page changes.

Responsive web design: The means by which a single purpose website can be made to look good in a wide range of viewport sizes.

A media query: An essential tool in responsive web design; it replaces alternative style sheets.

RWD: The acronym for Responsive Web Design

Is It Worth All the Effort?

Table 35-1 shows four factors that will help us to decide whether to make a website responsive to different viewports:

Table 35-1. *Viewing websites on smart phones has increased greatly since 2011*

	Year 2011	Year 2013	Year 2015
Percentage of potential viewers owning smartphones	40%	50%	80%
Is the website likely to be of interest to smart phone users?	Unlikely	Depends on the subject matter	Depends on the subject matter
What percentage of mobile users view websites on their smartphones?	Very few	30%	80%
What is the average viewport (portrait mode) width on the current range of smartphones?	190 pixels	190 to 360 pixels	320 to 768 pixels

Conclusion: If your website is likely to be viewed on a smartphone, you should seriously consider making it responsive.

■ **Caution** Some manuals on Responsive Website Design repeat the mantra “Forget design. Content is everything.” This of course is nonsense; a poorly designed web page packed with content will be a confusing mess that benefits nobody, least of all the user. Good design and relevant content are equally important. However, the smartphone version of a website is usually a stripped-down version of a desktop website; that is, many of the visual enhancements in a desktop viewport can cause unnecessary clutter in a smartphone display.

Different Websites? Or Different Style Sheets?

Different websites for mobiles and desktops are a nuisance to users and programmers. You may be thinking that using different style sheets for each screen size or device might solve the problem.

For example:

```
<link href="desktop.css" rel="stylesheet" type="text/css" media="screen">
<link href="phone.css" rel="stylesheet" type="text/css" media="handheld">
```

Sadly this will not work because mobile browsers will not respond to this approach.

The special browsers on mobiles have built-in code for adjusting a web page to suit the browser’s *viewport* (window size). The mobile browsers will therefore adjust your hand-held styles and give unexpected results.

■ **Note** In a responsive website the ‘different style sheet’ solution is replaced by *media queries* located within the main CSS file of the website. *Media queries* will be explained shortly.

The Responsive Solution

The responsive solution will provide one single website that can cope with smartphones, tablets, laptops, and desktop computers. The solution in this chapter relies entirely on HTML and CSS to eliminate the need for JavaScript. JavaScript will cause pages to load slowly on a smartphone.

■ **Warning** Because responsive websites are customized to suit a wide range of viewports, always let your clients know that a responsive version of their website will not look the same on a smartphone or tablet as it does on a desktop screen. They may not want this, in which case, you should tell the client that users viewing the website on a smartphone will have to scroll and zoom. Note that in Figure 35-4 the display in the smartphone version has been deliberately simplified. All the desktop decoration has been removed from the smartphone display for clarity and fast loading.

Now for some good news, apart from the statement in bold type below, responsive websites have exactly the same structure as any other website except for a viewport statement shown bold in Listing 35-1:

Listing 35-1. The structure for an RWD web page

```
<!DOCTYPE html>
<html>
  <head>
    <meta content="text/html; charset=utf-8" http-equiv="Content-Type">
    <title>Wetlands Conservation Trust</title>
    <link rel="stylesheet" href="style.css">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
  </head>
  <body>
    <div id="wrapper">
      Header
      Main navigation
      Content
      Footer
    </div><!--end of wrapper-->
  </body>
</html>
```

The Viewport Statement

Responsive web design requires a generic *viewport* statement in the `<head>` section of each web page. This will control the mobile browser's viewport.

The viewport information is shown in bold in Listing 35-1, and it is repeated below.

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

The viewport statement sets the web page to 100% of the width of the device's browser viewport. This cancels the default viewport setting in the browsers of various devices and avoids any built-in zoom that would spoil the display of our design. Users will still be able to zoom, but the initial display will be standardized on all devices.

The code: **content="width=device-width"**, sets the width of the page to the screen width of the device. To ensure that your layout will be displayed as you intended, we also set the zoom level.

For example the code: **initial scale=1.0** ensures that your layout will be displayed properly at 1:1 scale. No automatic zooming will be applied by the device.

The viewport statement is the only new feature in a responsive HTML file. All the other responsive features occur in the CSS style sheet as described in the next section.

Media Queries and the Responsive CSS Style Sheet

In RWD we replace the technique of using different style sheets with a device called *media queries*. You encountered one media query in Chapter 29 for printing pages in a stripped-down form. The media query in Chapter 29 was used in a <link> statement, but in responsive web design a media query in best located in the main CSS style sheet. We use media queries in the style sheet to set the break points for desktop, tablet, and smartphone.

A file in the downloadable code will demonstrate this on your desktop or laptop computer. Please follow these steps:

1. Create a folder labeled *Chapter-35*.
2. Download the zip file for Chapter 35 into the folder *Chapter-35* and unpack it.
3. On your desktop or laptop computer, open the file *media-query-demo.html* in Mozilla Firefox, Chrome, Safari, Internet Explorer 10 or 11, but not Edge because it does not shrink sufficiently to illustrate the full responsive effect.
4. Shrink the browser's viewport by dragging the browser's right edge *very slowly* to the left. You will see the effect of the three *breakpoints*.

Listing 35-2 gives the code for the file *media-query-demo.html*

Listing 35-2. The Media Query Demo code

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>Media Query Demo</title>
<style type="text/css">
    .viewing-area span {color:#000; display:none;}

    /* max-width. The break point for a smart phone */
    @media screen and (max-width:640px) {
        span.lesst640 {display:inline-block;}
    }
    /* min-width. The break point for a desktop computer */
    @media screen and (min-width:960px) {
        span.grtr960 {display:inline-block;}
        /*create a border around the wrapper in the desktop display*/
        #wrapper {border:solid 1px #000; padding:5px 10px; margin:1px;}
    }
</style>
</head>
</html>
```



```

/* min-width and max-width. Break points for tablets */
@media screen and (min-width:641px) and (max-width:959px) {
    span.betw641-959 {display:inline-block;}
}
</style>
</head>
<body>
<div id="wrapper">
    <p>&nbsp;</p>
    <p>&nbsp;</p>
    <div>
        <p class="viewing-area"><strong>Your current viewing area width is:</strong>
            <span class="lesst640">less than 640px</span>
            <span class="betw641-959">between 641 and 959px</span>
            <span class="grtr960">960px or greater</span>
        </p>
        <p>RWD demo by Adrian W West. Adapted from a demo by: webdesignerwall.com</p>
    </div>
</div>
</body>
</html>

```

■ **Note** Some older RWD tutorials included the word “only” or “all” within the media queries; with modern browsers this is no longer necessary. Also, you can if you wish, omit the word “and” immediately after the word “screen.” Be aware that the breakpoints used in this chapter are not definitive, and they may need to be tweaked as the screen resolutions of smartphones and tablets change.

Explanation of the Demonstration Media Query CSS Code

The demo code uses an internal style for simplicity. In a real-world website, the styles would be located in an external style sheet.

The three *media queries* set the *breakpoints* at which the display changes to match the *viewport* widths. The style section contains three *media queries*, one for each typical *viewport* width. When a particular *viewport* width is detected, a message is displayed by means of the `` classes. There is no need to memorise this `` class code because you will probably never use it again. Try to understand the logic behind the *media queries* and how they set the *breakpoints* for various *viewports*. Also be sure to grasp the fact that *media queries* behave like conditional style sheets.

■ **Note** A border has been specified only for the desktop viewport to demonstrate that, with media queries, we can enhance the appearance of the desktop display without affecting the tablet or smartphone display.

Responsive Websites Use Proportional Dimensions

This means using percentages instead of pixels for sizing the elements of a web page. This initiates float-drop causing horizontal elements to stack vertically in small viewports as shown in Figure 35-2.



Figure 35-2. Horizontal desktop elements (left) stack vertically upon one another, in smaller viewports (right)

RWDs require a website with liquid (percentage) dimensions; a device's browser can then rearrange the page elements to fit its viewport.

RWD sites employ float-drop so that horizontally oriented page elements on a desktop screen drop-down and become vertically oriented elements when squeezed into a smaller viewport. After spending many years battling to prevent float-drop in fixed or semi-liquid designs, I approached RWD with some trepidation. For desktop screens I always ensured that the content on the right of a page was clipped as the viewport shrank; suddenly I had to accept that float-drop was an important requisite of RWD, because clipping is not acceptable on tablets or smart phones.

Using Proportional Dimensions for the Elements on RWD Pages

The <body></body> tags. The body needs no width adjustment because its width is not usually specified. However, you may wish to include a background image or color for the body.

The <wrapper>. The best solution for a wrapper is to provide a percentage width together with a maximum width. This prevents problems on very wide screens with a huge number of pixels; otherwise the wrapper web would stretch too much and spoil the desktop layout. For instance, we might specify a wrapper width of 100% together with a maximum width of 1,400 pixels.

The CSS could be as follows:

```
#wrapper {max-width:1400px; width:100%; margin:auto;}
```

This would limit the width to 1,400 pixels on a desktop screen, but ensure that the wrapper filled the viewport of a tablet or smart phone.

Column widths. In RWD we must use percentage widths instead of pixels; for instance three equal-width columns might have widths of 33.32999% each. Four equal-width columns might have widths of 24.99999% each. The percentage widths should add up to a total just below 100%. You will find that 99.9% is almost always enough, but add all the element widths together to determine the maximum width for your layout. Using five figures after the decimal point is normal practice in RWD.

In the downloadable file *index-35.html*, I created two columns with the styles `width:47.99999%;` and `float:left;` These are the containers for the picture and the textual content.

The header (banner): A large number of words in a header will break up into several lines when displayed in narrow viewports. The header then occupies too much vertical space. Therefore you should create the briefest possible header text and use the minimum amount of decoration in the header.

Responsive Images

In previous chapters, we followed the normal practice of including a fixed height and width in image tags. In RWD, image tags no longer include fixed dimensions. Instead, pictures are allowed to grow and shrink to match the viewport size without loss of quality. To achieve this, the following lines need to be inserted into the CSS file.

```
/*MAKE IMAGES RESPONSIVE*/
img {max-width:100%;}
```

This section demonstrates how you can create flexible pictures that will adapt to various viewport widths. In the following example the desktop version of home page has two equal-width columns, with a picture in the left column and some text in the right column (see Figure 35-3). That means two columns with proportional widths of approximately 50% each.

Because the wrapper surrounds all the elements on the web page, it makes sense to choose something slightly less than 50% of the wrapper width for each column. We should set a maximum width for the wrapper to prevent problems on very wide desktop viewports. In this case the maximum width will be set to 1,400 pixels. A picture must fit into the maximum column within which it is located. I chose 50% of the maximum wrapper width to determine the responsive picture's width. However, a responsive picture can be a little wider than its intended container (i.e., the maximum column width). The CSS code for the wrapper was set as follows:

```
#wrapper {max-width:1400px; width:100%; margin:auto;}
```

I prepared an optimized picture *wetlands-comp.jpg* for you in the downloadable zip file for this chapter. Its width is 50% of the maximum wrapper width, which is 700 pixels (50% of 1,400 pixels). Always start with a large, good quality picture and resize it to fit its maximum container width. In this exercise I chose a picture that was 2,000 pixels wide, cropped it, and shrank it in a graphics program to 700 pixels wide. In theory you could use the 2,000 pixels wide picture because, being responsive it will shrink automatically to 700 pixels or less; that would be a bad idea because smartphones are not blessed with much memory or band width and the huge picture would download very slowly.

As well as shrinking the picture to a width of 700 pixels, I used my graphics program to compress it to give the fastest possible loading speed. The file size of my original 2,000 pixel picture was 3.1 MB. By cropping, resizing, and compressing it, the file size shrank to 69Kb.

If we have a picture 700 pixels wide, surely that is a fixed width? Yes that is true, but the picture's dimensions will be deliberately omitted in the code that inserts it into the HTML page. Also, I have made it responsive by means of a simple piece of CSS code; all will be revealed shortly.

The Responsive Navigation Menu

Clear navigation menus are essential on desktops, laptops, and tablets. On a smartphone they are also essential but they must not dominate the small viewport. Vertical sidebar navigation menus are not really suitable for smartphones, so we will therefore use a horizontal menu.

To design the menu for this chapter, I followed the guidelines set out in Chapter 17. I planned the number of pages using the rule “one page per topic.” Then I planned a logical layout of the menu buttons so that the user could easily access the pages. I used variable width menu buttons so that buttons with short labels took up less room in the smart phone viewport. I juggled with the order of the menu items to ensure that smart phones displayed two rows of buttons, rather than three rows; this left more room below the menu for content.

We need a way of ensuring that the menu is not only clearly presented, but also takes up the least possible amount of the smartphone’s screen. This chapter describes one practical solution as follows:

- Use a horizontal menu with a limited number of buttons. The most commonly recommended maximum number of buttons for responsive situations is between five and seven depending on the length of the button labels. We don’t want the viewport of the smartphone to be filled with navigation links to the exclusion of informative content.

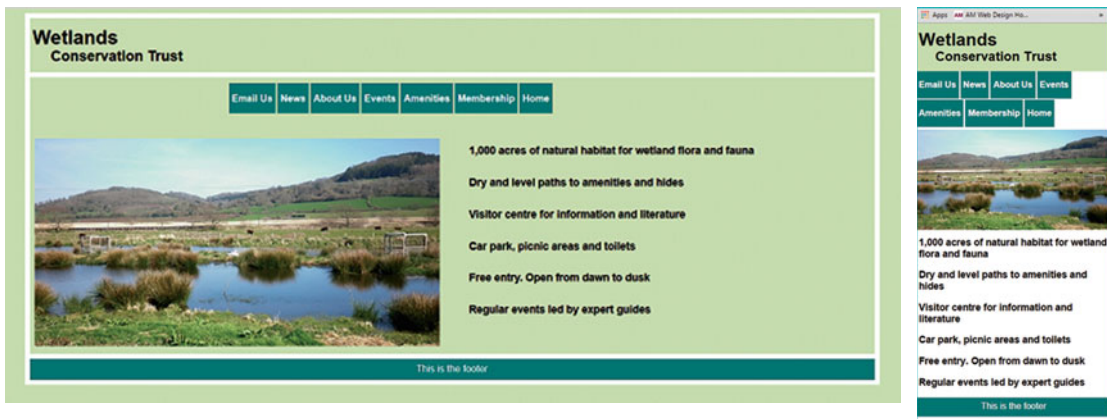
Many responsive websites collapse the navigation menu into a single button or icon for the smartphone version. The single menu button or icon can be clicked to expose the other navigation buttons. This technique allows plenty of room for content in the smartphone’s viewport. This advanced technique is covered in Chapters 36 and 37.

■ **Tip** Most tutorials and manuals recommend that you design the mobile version first, and then use media queries to enhance the desktop version of the pages. This means that you should bear in mind the mobile’s narrow viewport and *think Responsive* right from the start of the design of a website. We will follow this advice in this chapter and in Chapters 36 and 37. Technically it doesn’t matter if you start from a fixed width/semi-liquid website, or you start with the smart phone viewport; but you will find it easier to start with a mobile responsive website in mind.

The downloadable folder for this chapter contains Responsive Web Design files that you can view to see the effect of various viewports. To view them please follow these steps:

1. Open the downloaded file *index.html* in Firefox, Chrome, Internet Explorer 10 or 11, but not Edge because it does not shrink sufficiently to illustrate the full responsive effect.
2. Shrink the browser’s viewport by dragging the browser’s right edge *very slowly* to the left.
3. Particularly note how the menu changes to match the viewport width.
4. Click the Amenities button to see how the horizontally oriented pictures become vertically oriented in the smart phone viewport.

Figures 35-3 and 35-4 show the home page in a desktop viewport and the smartphone viewport (not to scale).



Figures 35-3 and 35-4. A responsive web page on a desktop screen and a smartphone

The HTML code for the home page is given in Listing 35-3.

Listing 35-3. The home page (*index.html*)

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>Wetlands C T. Home page</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
<div id="wrapper">
<header>
  <h1>Wetlands</h1>
  <h2>Conservation Trust</h2>
</header>
<div id="menu">
  <ul>
    <li><a href="form.html">Email Us</a></li>
    <li><a href="#">News</a></li>
    <li><a href="#">About Us</a></li>
    <li><a href="events.html">Events</a></li>
    <li><a href="amenities.html">Amenities</a></li>
    <li><a href="#">Membership</a></li>
    <li><a href="index.html">Home</a></li>
  </ul>
</div>
  <p>&nbsp;</p>
  <p>&nbsp;</p>
<div id="col-1">

```

```

</div>
<div id="col-2">
    <p>1,000 acres of natural habitat for wetland flora and fauna</p>
    <p>Paths to amenities and hides, are sign posted, dry and level</p>
    <p>Visitor centre for information and literature</p>
    <p>Car park, picnic areas and toilets</p>
    <p>Free entry. Open from dawn to dusk</p>
    <p>Regular events led by expert guides</p>
</div>
<br class="clear">
<footer>
This is the footer
</footer>
</div>
</body>
</html>

```

■ **Note** In this website example, to save space only four pages are used, the other pages in the menu therefore have dead links, that is, **<a href="#"**

The HTML page Listing 35-3 contains only two surprises: the *viewport* statement and the *image* source, and both are shown in bold. As stated earlier, images must NOT specify their width and height dimensions.

The responsive aspect of the page is dictated by the code in the CSS file given in Listing 35-4.

Listing 35-4. The CSS style for the responsive website (*style.css*)

```

/*MAKE IMAGES RESPONSIVE*/
img {max-width:100%;}
/*SET BASIC STYLES FOR ALL VIEWPORTS*/
body {background-color:white;}
#wrapper {font-family:Arial, Helvetica, sans-serif; line-height:2.0;}
header {background-color:#CEE1BA; height:80px; padding:5px;}
header h1 {margin-bottom:-45px; margin-top:-5px; min-width:320px;}
header h2 {margin-left:30px; min-width:320px;}
#col-2 {float:right; text-align:left; font-size:110%; font-weight:bold;}
.clear {clear:both;}
/*STYLE THE HORIZONTAL MENU*/
ul {list-style-type:none; margin:0; padding:0; margin-left:30px;}
ul li a {padding-left:5px; padding-right:5px; margin-left:0; margin-right:0;}
/*STYLE THE MENU BUTTONS*/
li {display:inline-block; float:left; margin-right:3px;}
li a {display:block; height:50px; text-align:center; line-height:50px; ↵
    padding:0 5px 0 5px; font-family:Arial, sans-serif; color:#FFF; font-weight:bold; ↵
    background:teal; text-decoration:none;}
/*HOVER STATE FOR MENU BUTTONS*/
li:hover a {background:#19C589;}
/*STYLE THE FOOTER*/
footer {text-align:center; font-family:Arial, Helvetica, sans-serif; ↵
    background-color:teal; color:white; height:35px; line-height:2.0;}
/*STYLE THE "Email Us" FORM*/

```

```

#feedback form {margin:auto;}
#feedback form #message-field {font-family:Arial, Helvetica, sans-serif; font-size:1.4em; ↵
    font-weight:normal}
#feedback h2 {font-weight:bold; padding:0; margin:0;}
#feedback h3 {font-weight:bold; padding:0; margin:-10px 0 -10px 0;}
label {font-size:1.4em; font-weight:200; padding-top:12px; display:block; min-width:320px;}
#caption {width:300px;}
input {min-width:265px; font-size:1.4em; font-weight:200; padding:10px;}
input[type=checkbox] {transform:scale(2); margin:auto;}
input[type=submit] {margin:0 auto 0 auto; background-color:green; color:white; ↵
    padding:5px; border:none; font-size:1.4em;}
.red {color:red; font-weight:bold; font-size:110%;}
/* STYLE TABLES */
table {background-color:#FFF; border-collapse:collapse; border-spacing:0; width:95%; ↵
    margin:auto; border:1px #000 solid;}
th,td {border:1px #000 solid; text-align:left; padding:5px; line-height:1.5; ↵
font-size:115%;}
.figure {margin:0 1% 10px 1%; display:inline-block;}
/*STYLE PICTURE GALLERY*/
.figure p {text-align:center; margin:5px 0 0 0;}
#gallery {margin:auto;}
#content h2 {text-align:center;}

/* THE RESPONSIVE STYLES */
/*STYLE THE ELEMENTS FOR THE NARROW SMART PHONE VIEWPORT*/
@media screen and (max-width:380px){
    .figure {width:93%; margin-left:10px;}
    #feedback input {min-width:255px;}
}
@media screen and (max-width:768px){
/*STYLE THE ELEMENTS FOR THE WIDER SMART PHONE VIEWPORT*/
    #wrapper {border:none; padding:0; margin:0 -10px 0 -10px;}
    /*STYLE THE COLUMNS AND PULL THE PICTURE UP CLOSER TO THE MENU*/
    #col-1 {margin-top:-65px;}
    #col-2 {padding:0 5px 0 5px; line-height:1.2; margin-top:-15px; float:left;}
    /*STYLE THE OTHER SMART PHONE ELEMENTS*/
    header {margin-top:-10px; padding:5px;}
    ul {margin-left:0px;}
    /*CREATE A MARGIN BELOW THE BUTTONS*/
    li {margin-bottom:2px;}
    table {margin-top:-60px; padding:0;}
    #feedback form {margin-left:30%; margin-top:-20px; padding:20px 5%;}
}

@media screen and (min-width:769px) and (max-width:959px) {
/*STYLE THE ELEMENTS FOR THE IN-BETWEEN SIZE VIEWPORTS*/
#col-1 {margin-top:-65px;}
    #col-2 {float:left;}
    #feedback form {margin-left:30%; margin-top:20px;}
}

```

```

@media screen and (min-width:960px) {
/*STYLE THE ELEMENTS FOR THE DESKTOP VIEWPORT*/
    body {background-color:#CEE1BA;}
    #wrapper {max-width:1400px; border:solid 8px #FFF; padding:0; margin:auto;}
    footer {border-top:8px white solid;}
    /*ADD A WHITE BOTTOM BORDER*/
    header {margin-bottom:10px; border-bottom:8px white solid;}
    /*SET STYLE FOR TWO COLUMN HOME PAGE*/
    #col-1 {width:47.99999%; float:left; margin-top:-25px;}
    #col-2 {width:47.99999%; float:left; margin-top:-40px; margin-left:10px;}
    #menu {width:800px; margin:auto;}
    /*STYLE THE PICTURE GALLERY*/
    #gallery .figure {float:left; margin:0 1% 0 1%; width:22.99999%;}
    .drop-shadow {box-shadow:5px 5px 5px rgba(0, 0, 0, 0.5);}
    #gallery {width:95%;}
    /*STYLE THE FORM*/
    #feedback form {margin-left:30%; padding:20px 5%;}
    #message-field {width:380px; margin-left:-60px;}
}

```

Explanation of the Styles (*style.css*)

```
/*MAKE IMAGES RESPONSIVE*/
```

```
img {max-width:100%;}
```

This code states that an image must fill its containing element; in our home page the container for the picture is the left column (col-1). The image will shrink to match smaller viewport widths.

■ **Important** In the HTML page the image tag must not specify width and height dimensions; otherwise the image will not shrink. It should be as follows:

```



```

The main block of code before the line `/*THE RESPONSIVE STYLES */` provides the basic styles common to all viewports. The salient points in these common styles are now described.

```

header {background-color:#CEE1BA; height:80px; padding:5px;}
header h1 {margin-bottom:-45px; margin-top:-5px; min-width:320px;}
header h2 {margin-left:30px; min-width:320px;}

```

This code ensures that, in a smartphone, the header text occupies the least vertical space consistent with a clear heading.

```

input[type=checkbox] {transform:scale(2); margin:auto;}
input[type=submit] {margin:0 auto 0 auto; background-color:green; color:white;
padding:5px; border:none; font-size:1.4em;}

```


Note the square brackets around the attributes `[type=checkbox]` and `[type= submit]`; this useful device enables us to define an element's style by using its attribute.

The default check box size in all current browsers is rather small, and the first line of code uses the style `transform:scale(2)`; to give the check box a reasonable size.

The form's *Submit* button is styled using the attribute `[type=submit]`

The responsive styles have explanatory comments; therefore no additional explanation is required.

■ **Note** Blue Griffon and Microsoft Expression Web 4 are not able to display a true view of RWD pages in the WYSIWYG/Design view. In fact the display will be a mess with elements superimposed on top of each other.

This does not affect the source/code view of either text editor nor will it affect the display in a browser.

View the Other Responsive Pages

Open *index.html* in a modern browser and use the menu to view the other pages. Expand and shrink the viewport to view the responsive effect on each page. Use your HTML text editor to examine the code in these three pages.

Testing Responsive Websites

So far you have tested responsive pages by *slowly* shrinking your browser horizontally to the left. However, this is not a trustworthy substitute for testing on actual mobile devices. Mobile devices have browsers with built-in quirks that your desktop computer cannot anticipate. Although the browsers for mobiles may have the same names as desktop browsers, they have differences to adapt them to the smaller viewports and to speed up their performance. Your website might give a satisfactory display on a tablet in landscape, but you need to make sure that when users turn their tablets around to portrait mode, they will still receive an acceptable display.

I have tried using Android and iOS emulators but these are difficult to understand. If, like me, you cannot afford to buy several smartphones and tablets, you could rely on friends and colleagues to test your websites for you. Alternatively, your public library may have Wi-Fi, in which case you could visit the library and politely ask smartphone and tablet users if they would view your website while you look over their shoulder.

For a low-cost device try the iPod Touch for testing Apple's iOS and Safari; this is not a phone but it has the dimensions of a small smart phone and it will connect to your router via Wi-Fi. You can then view your hosted websites. The battery is the main drawback, as it is not replaceable and it does not hold its charge for long. Also if you don't charge it frequently, it eventually refuses to charge. No warning is provided with the device.

Google provides a very basic online test at

<http://www.google.com/webmasters/tools/mobile-friendly/>

You need to upload your web page to a host server, then enter its URL in the Google tester.

A non-responsive website will give the following report:

"Page appears not mobile-friendly. Text too small to read. Links too close together. Content wider than screen. Mobile viewport not set"

The online tester at <http://mobiletest.me/> is an attractive alternative solution, and you can choose to display the website in several viewport sizes. You must upload the site to a host server first. Such testers are good, but they are not as reliable as testing on actual mobile devices.

In my opinion the best online tester is <http://quirktools.com/screenfly/>

You must upload your site to a host server first. The tester is easy to use and the results are as good as testing on actual mobile devices. The toolbar has icons that show the three types of devices. When these are clicked, a drop-down menu will allow you to choose the particular brand of device. Another icon allows you to flip between landscape and portrait view.

Summary

We examined the four basic essentials for a responsive website design: (i) place a viewport statement in the head section of each page, (ii) use percentage widths instead of pixels, (iii) add media queries in the style sheet for setting breakpoints, and (iv) use responsive images. You experimented with a file that demonstrated breakpoints, and then you investigated a basic navigation menu designed for responsive websites. In the next chapter you will learn how to add a drop-down feature to the RWD menu to accommodate more pages.



Building Responsive Websites for Mobile Devices Part 2

This chapter will show you how to add extra features to the responsive website design that was described in Chapter 35. The menu will be rearranged and augmented by means of a basic drop-down menu. This will allow several new pages to be added to the website. The previous menu used large buttons suitable for touch screens on smartphones. You might be tempted to use smaller buttons to allow more to be squeezed into a smartphone screen, but this would make them too fiddly to operate easily. Instead, this chapter continues to use the same large buttons as Chapter 35.

The pages used in the previous chapter will be augmented by five extra pages, but to save space and time only two of these will be described and included in the downloadable folder. The pages in the previous chapter used a single column per page (except for the amenities page that had several responsive images set by percentages). This chapter will introduce you to responsive two-column, three-column and four-column pages. The chapter is purely descriptive and contains no projects.

This chapter contains the following sections:

- Adding a drop-down feature to the menu buttons
- Creating a two-column responsive web page
- Creating a three-column responsive web page
- Creating a four-column responsive page
- Summary

Adding a Drop-Down Feature to the Menu Buttons

The menu is designed on the principle outlined in the previous chapter, that is, work with the smartphone layout first, then use that to develop the larger viewport layouts.

In Chapter 35 you were introduced to Responsive Web Design (RWD) and you examined a basic RWD website and the code that produced it. The viewport of a narrow smartphone limited the number of menu buttons to seven and these were located in two rows (see Figure 36-1). Limiting the number of rows to two rows is good practice because we don't want to fill the smartphone's viewport with menu buttons, leaving less room for content. In this chapter the menu is reorganized (see Figure 36-2) and provided with drop-down menu buttons so that a larger number of web pages can be accessed. To achieve this, the drop-down technique described in Chapter 26 is added to the responsive menu in this chapter.

Note that two of the reorganize buttons have down-pointing arrows indicating that they are drop-down menu buttons.

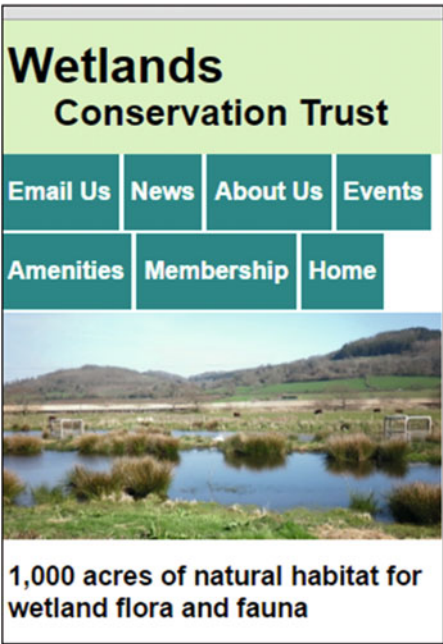


Figure 36-1. The menu from Chapter 35



Figure 36-2. The reorganized menu

The code for the drop-down menu buttons is embedded in each page of the website. The code is given in Listing 36-1.

Listing 36-1. The drop-down menus

```
<div id="drop-menu">
  <ul id="menu">
    <li><a href="index.html">Home</a></li>
    <li><a href="form.html">Contact Us</a></li>
    <li><a href="#">3 pages &#65516;</a>
      <ul class="hidden">
        <li><a href="3-col.html">3 Cols</a></li>
        <li><a href="4-col.html">4 Cols</a></li>
        <li><a href="#">Volunteer</a></li>
      </ul>
    </li>
    <li><a href="#">Reviews</a></li>
    <li><a href="#">Information &#65516;</a>
      <ul class="hidden">
        <li><a href="amenities.html">Amenities</a></li>
        <li><a href="events.html">Events</a></li>
        <li><a href="#">Location</a></li>
        <li><a href="#">About Us</a></li>
        <li><a href="#">Join</a></li>
        <li><a href="#">News</a></li>
      </ul>
    </li>
  </ul>
</div>
```

```
        </li>
      </ul>
</div>
```

Explanation of the Code

```
<li><a href="#">3 pages &#65516;</a>
  <ul class="hidden">
    <li><a href="3-col.html">3 Cols</a></li>
    <li><a href="4-col.html">4 Cols</a></li>
    <li><a href="#">Volunteer</a></li>
  </ul>
</li>
```

The drop-down links as shown in bold type are nested within `` tags. The first line of the `` tag contains an entity `￬` which provides the down-pointing arrow. The second line begins the new unordered list that contains the drop-down links. This has the `class="hidden"` that hides the drop-down links until the cursor hovers over the button.

The menus could have been inserted into each page using PHP *includes*, but in this instance that method was not used so that you can view the pages by simply right-clicking the file and choosing to load it in your favorite modern browser. View the file in Firefox, Chrome, Safari, or Opera and drag the right edge leftwards to view the effect in various viewports. At the time of writing, the Microsoft Edge browser did not shrink enough to simulate a narrow smartphone.

The drop-down links are shown in Figures 36-3 and 36-4.

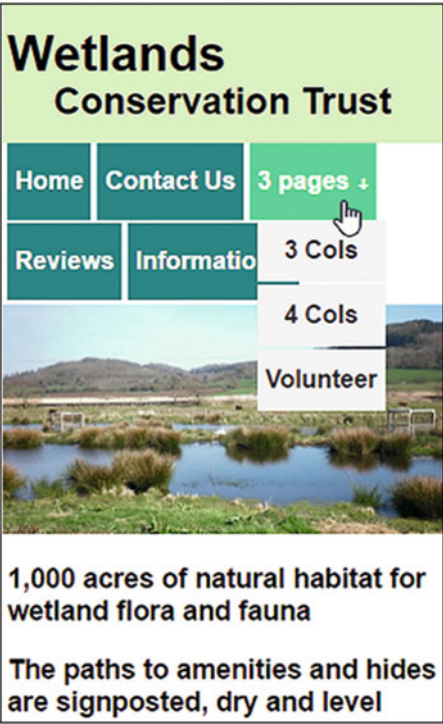


Figure 36-3. The first drop-down list



Figure 36-4. The second drop-down list

The drop-down list in Figure 36-3 has two links that need some explanation because they do not fit the theme of the website. The links *3 cols* and *4 cols* are present only to demonstrate the appearance of the three-column and four-column pages. In your own responsive websites you would change the button label and the pages to something more appropriate. For example, in this wetlands website, the button might be labeled Flora & Fauna, or Wildlife; the three pages might be Birds, Plants, and Insects.

To save space, no pages have been provided for the following links:

Volunteer, Reviews, Location, About Us, Join, and News. The menus have dead links to these non-existent pages.

The live linked pages are Home, Contact Us, 3-cols, 4-cols, Amenities, and Events.

The link for the two-column page is the button labeled *Home*. We will now consider the responsive two-column home page.

Creating a Two-Column Responsive Web Page

The revised home page has two columns: the left column contains a picture, and the right column contains text as shown in Figure 36-5.

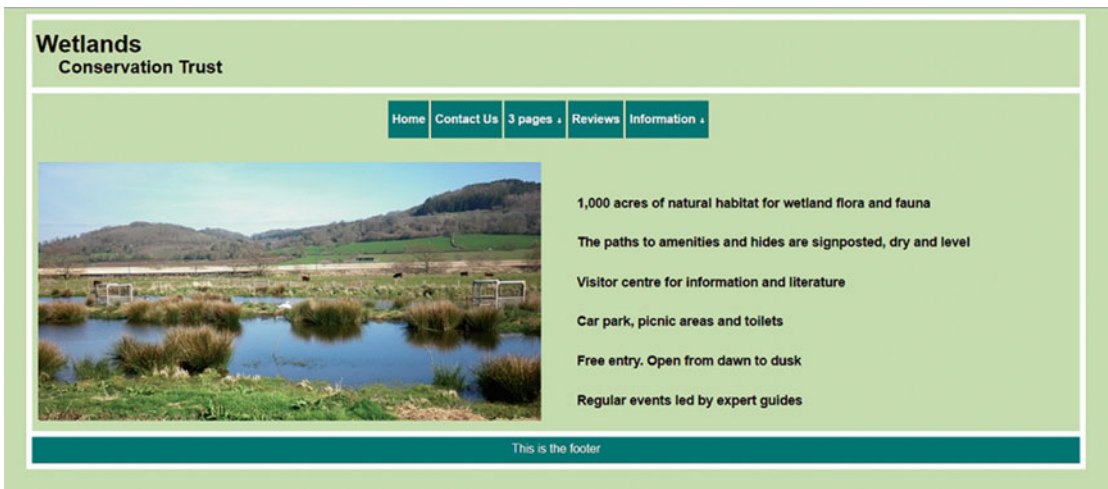


Figure 36-5. Showing the two-column layout

To view the page in various viewports please follow these steps:

1. Download the *Chapter 36.zip* folder and unzip it in your *Web Tutorial* folder (not in the *htdocs* folder) .
2. Open the *index.html* file in a modern browser, and you will see the two columns.
3. Shrink the browser viewport by dragging the right edge slowly to the left. You will see that in the smartphone viewport, the two columns become one column with the elements stacked one above the other.

The code that sets the two columns is shown in bold type in the home page Listing 36-2.

Listing 36-2. The code for the two-column home page

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>Wetlands C T. Home page</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
<div id="wrapper">
<header>
    <h1>Wetlands</h1>
    <h2>Conservation Trust</h2>
</header>
<div id="drop-menu">
    <ul id="menu">
        <li><a href="index.html">Home</a></li>
        <li><a href="form.html">Contact Us</a></li>
        <li><a href="#">3 pages &#65516;</a>
            <ul class="hidden">
                <li><a href="3-col.html">3 Cols</a></li>
                <li><a href="4-col.html">4 Cols</a></li>
                <li><a href="#">Volunteer</a></li>
            </ul>
        </li>
        <li><a href="#">Reviews</a></li>
        <li><a href="#">Information &#65516;</a>
            <ul class="hidden">
                <li><a href="amenities.html">Amenities</a></li>
                <li><a href="events.html">Events</a></li>
                <li><a href="#">Location</a></li>
                <li><a href="#">About Us</a></li>
                <li><a href="#">Join</a></li>
                <li><a href="#">News</a></li>
            </ul>
        </li>
    </ul>
</div>
    <p>&nbsp;</p>
    <p>&nbsp;</p>
<div id="col-1">
    
</div>
<div id="col-2">
<p>1,000 acres of natural habitat for wetland flora and fauna</p>
    <p>The paths to amenities and hides are signposted, dry and level</p>
    <p>Visitor centre for information and literature</p>
    <p>Car park, picnic areas and toilets</p>
```

```

        <p>Free entry. Open from dawn to dusk</p>
        <p>Regular events led by expert guides</p>
</div>
<br class="clear">
<footer>
This is the footer
</footer>
</div>
</body>
</html>

```

The CSS code for styling the page and menu is given in Listing 36-3.

Listing 36-3. The main style sheet for all the pages in the website (*style.css*)

```

body {background-color:white;}
#wrapper {font-family:Arial, Helvetica, sans-serif; line-height:2.0;}
header {background-color:#CEE1BA; height:100px; padding:5px;}
header h1 {margin-bottom:-45px; margin-top:-5px;}
header h2 {margin-left:30px; }
/*Make images responsive*/
img {max-width:100%;}
img.home {margin-top:-20px;}
#col-2 {float:right; text-align:left; font-size:110%; font-weight:bold;}
.clear {clear:both;}
/*Strip padding and list styling from the menu*/
ul {list-style-type:none; margin:0; padding:0; position:absolute;}
/*Create a horizontal list with spacing*/
li {display:inline-block; float:left; margin-right:3px;}
/*Style the menu buttons*/
li a {display:block; height:50px; text-align:center; line-height:50px; ↵
padding:0 5px 0 5px; font-family:Arial, sans-serif; color:#fff; font-weight:bold; ↵
background:teal; text-decoration:none;}
/*Hover state for top level links*/
li:hover a {background:#19c589;}
/*Style for dropdown links*/
li:hover ul a {background:#f3f3f3; color:#2f3036; height:40px; line-height:40px;}
/*Hover state for dropdown links*/
li:hover ul a:hover {background:#19c589; color:#fff;}
/*Hide dropdown links until they are needed*/
li ul {display:none;}
/*Place smartphone dropdown links in a vertical block*/
li ul li {display:block; float:none;}
/*Prevent text wrapping
li ul li a {padding: 0 20px; margin-right:2%;}*/
/*Display the smartphone dropdown on hover*/
ul li a:hover + .hidden, .hidden:hover {display:block;}
footer {text-align:center; font-family:Arial, Helvetica, sans-serif; ↵
background-color:teal; color:white; height:35px; line-height:2.0;}

```



```

/*RESPONSIVE STYLES*/
@media screen and (max-width:400px){
#content h2 {margin-top:60px;}
#col-1 img {margin-top:3px;}
}

@media screen and (max-width:760px){
  #wrapper {border:none; padding:0; margin:0 -10px 0 -10px;}
  /*pull the picture up closer to the menu*/
  #col-1 {margin-top:-10px;}
  #col-2 {padding:0 5px 0 5px; line-height:1.2;}
  /*Insert a line for the smart phone header*/
  header {background-color:#CEE1BA; height:80px; margin-top:0; padding:5px;}
  /*Create vertical spacing in drop-down links*/
  li {margin-bottom:2px;}
  ul li, li a {padding-left:5px; padding-right:5px; margin-left:0; margin-right:-3px;}
  footer {text-align:center; font-family:Arial, Helvetica, sans-serif; ↵
  background-color:teal; color:white; height:35px; line-height:2.0;}
}

@media screen and (max-width:959px){
  footer {text-align:center; font-family:Arial, Helvetica, sans-serif; ↵
  background-color:teal; color:white; height:35px; line-height:2.0;}
  #col-2 {margin-top:-10px; float:left; margin-left:0;}
}

@media screen and (min-width:960px) {
  body {background-color:#CEE1BA;}
  #wrapper {max-width:1400px; border:solid 8px #fff; padding:0; margin:auto;}
  /*Insert a line for the desktop header*/
  header {background-color:#CEE1BA; height:80px; margin-bottom:10px; ↵
  padding:5px; border-bottom:8px white solid;}
  img.home {margin-left:8px; margin-bottom:3px;}
  #col-1 {width:47.99999%; float:left;}
  #col-2 {width:47.99999%; float:right;}
  ul {list-style-type:none; margin:0; padding:0;}
  #drop-menu {width:800px; margin-left:34%; }
  li a {padding-left:5px; padding-right:5px; margin-left:0; margin-right:0;}
  footer {border-top:8px white solid;}
  footer {text-align:center; font-family:Arial, Helvetica, sans-serif;
  background-color:teal; color:white; height:35px; line-height:2.0;}
}

```

Explanation of the Code

Examine each block of code in Listing 36-3 to find the styles in bold type applied to the two columns *col-1* and *col-2*. In the last block of code you will see the style that sets equal-width columns on a desktop or laptop screen. An automatic gap between the columns was set by making the sum of the widths less than 100% and by floating the columns left and right. The columns can also be given unequal widths; for instance they could be 40.79999% and 55.19999%.

We will now examine the code for a page with three equal-width columns.

Creating a Three-Column Responsive Web Page

The responsive three-column page again uses percentages to set the width of the columns. In this case there are three equal-width columns as shown in Figure 36-6.

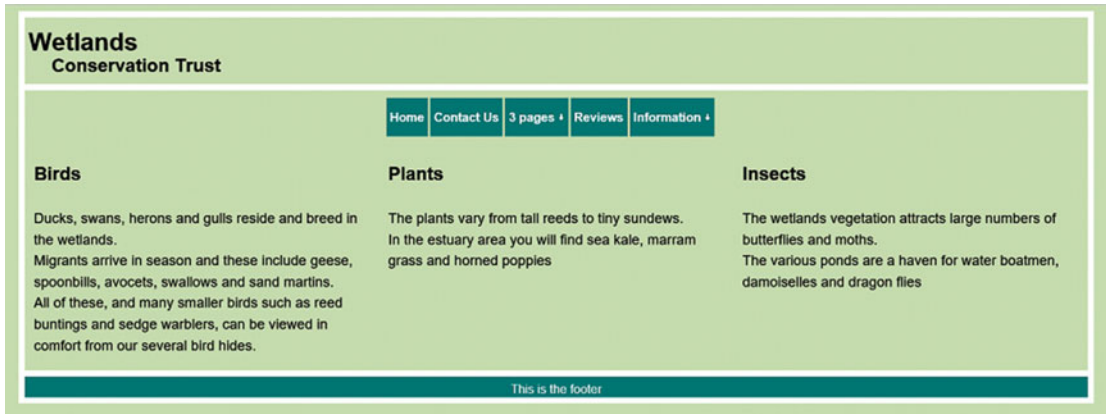


Figure 36-6. A three-column page

The HTML code for setting the three columns is given in Listing 36-4. The salient points are shown in bold type.

Listing 36-4. Creating a page with three responsive columns

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>Three columns</title>
  <link rel="stylesheet" href="style.css">
  <link rel="stylesheet" href="cols-table.css">
</head>
<body>
<div id="wrapper">
<header>
  <h1>Wetlands</h1>
  <h2>Conservation Trust</h2>
</header>
<div id="drop-menu">
  <ul id="menu">
    <li><a href="index.html">Home</a></li>
    <li><a href="form.html">Contact Us</a></li>
    <li><a href="#">3 pages &#65516;</a>
      <ul class="hidden">
        <li><a href="3-col.html">3 Cols</a></li>
        <li><a href="4-col.html">4 Cols</a></li>
        <li><a href="#">Volunteer</a></li>
```

```

        </ul>
    </li>
    <li><a href="#">Reviews</a></li>
    <li><a href="#">Information &#65516;</a>
        <ul class="hidden">
            <li><a href="amenities.html">Amenities</a></li>
            <li><a href="events.html">Events</a></li>
            <li><a href="#">Location</a></li>
            <li><a href="#">About Us</a></li>
            <li><a href="#">Join</a></li>
            <li><a href="#">News</a></li>
        </ul>
    </li>
</ul>
</div>
<p>&nbsp;</p>
<div id="content">
<div class="third col">
<h2>Birds</h2>
<p class="close-gap">Ducks, swans, herons and gulls reside and breed in the wetlands.<br>
Migrants arrive in season and these include geese, spoonbills, avocets, swallows and
    sand martins.<br>
All of these, and many smaller birds such as reed buntings and sedge warblers, can be
viewed in comfort from our several bird hides.</p>
</div>
<div class="third col">
<h2>Plants</h2>
<p class="close-gap">The plants vary from tall reeds to tiny sundews.<br>
In the estuary area you will find sea kale, marram grass and horned poppies</p>
</div>
<div class="third col">
<h2>Insects</h2>
<p>The wetlands vegetation attracts large numbers of butterflies and moths.<br>
The various ponds are a haven for water boatmen, damoiselles and dragon flies<br>
</div>
</div><!--The content div finishes here-->
<br class="clear">
<footer>
This is the footer
</footer>
</div>
</body>

```

Explanation of the Code

<link rel="stylesheet" href="cols-table.css">

The additional style sheet **cols-table.css** formats the columns and tables.

<div class="third col">

Each column is enclosed within a `<div>` that has the two classes *third* and *col*

The additional link to *cols-table.css* provides the column styles. This style sheet is separate from the main style sheet so that you can see clearly how the CSS provides the three columns; normally this would be rolled into the main style sheet. The code also sets the format for data tables, but don't attempt to use tables to set columns. This is not only deprecated but it won't work in a responsive website. Tables will work responsively only when used for presenting data.

The code is given in Listing 36-5.

Listing 36-5. The CSS code for the column styles (*cols-table.css*)

```
*{box-sizing:border-box}
header {height:95px;}
#content p {line-height:1.5; font-size:115%;}
/* TABLES */
table {background-color:#fff; border-collapse:collapse;border-spacing:0;width:95%;
margin:auto; border:1px #000 solid;}
th,td {border:1px #000 solid; text-align:left; padding:5px; line-height:1.5;
font-size:115%;}
/* COLUMNS */
.col,.half,.third,.twothird,.quarter{float:left; width:100%}
.col {padding:1px 12px}
/*RESPONSIVE STYLES */
@media screen and (max-width:480px){
#events table {margin-top:20px;}
div #content h2 {margin-top:45px;}
div #content p.close-gap {margin-bottom:-35px;}
}

@media screen and (max-width:760px){
table {margin-top:-60px; padding:0;}
div #content {margin-top:0; padding:0;}
#content h2 {margin-bottom:-10px; margin-top:0;}
}

@media screen and (min-width:960px){
.quarter {width:24.99999%}
.third {width:33.33333%}
}
```

Explanation of the Code

***{box-sizing:border-box}**

The elements on a web page are all boxes. If you add borders or padding to a box it becomes wider than the width you set. This can cause some puzzling problems, particularly with columns on a responsive website. By using ***{box-sizing:border-box}** the content, padding and borders will all sit within the width you set; this will eliminate any baffling width problems (See Chapter 15 for box model).

```
/* COLUMNS */
.col,.half,.third,.twothird,.quarter{float:left; width:100%}
.col {padding:1px 12px}
```

This code provides the general setting for various column classes that might be included in a website.

```
@media screen and (min-width:960px){
.quarter {width:24.99999%}
.third {width:33.33333%}
```

This code sets the percentages for four columns and three columns on viewports larger than 960 pixels. We will now investigate the responsive four-column page.

Creating a Four-Column Responsive Web Page

The responsive four-column page again uses percentages to set the width of the columns. In this case there are four equal-width columns as shown in Figure 36-7.



Figure 36-7. *The four-column page*

The new code is given in the partial Listing 36-6.

Partial Listing 36-6. The new code for four columns

```
<div id="content">
<div class="quarter col">
<h2>Birds</h2>
<p class="close-gap">Ducks, swans, herons and gulls reside and breed in the wetlands.<br>
Migrants arrive in season and these include geese, spoonbills, avocets, swallows and ↵
  sand martins.<br>
All of these and many smaller birds such as reed buntings and sedge warbler can be viewed ↵
  in comfort from our several bird hides.</p>
</div>
<div class="quarter col">
<h2>Plants</h2>
<p class="close-gap">The plants vary from tall reeds to tiny sundews.<br>
In the estuary area you will find sea kale, marram grass, sea purslane and horned poppies</p>
</div>
```

```

<div class="quarter col">
<h2>Insects</h2>
<p class="close-gap">The wetlands vegetation attracts large numbers of butterflies and
    moths.<br>The various ponds are a haven for water boatmen,
    damoiselles and dragon flies</p>
</div>
<div class="quarter col">
<h2>Terrain</h2>
<p>The wetlands contain ponds, lakes, marshes and a sandy estuary.<br>
    They can all be accessed by dry and level paths.
<br>Electric buggies are available for hire so that disabled vistors may view
    the various attractions.</p>
</div>
</div><!--content div ends here-->

```

The fourth column begins with the line shown in bold type.

All the columns have been given two classes, quarter and col, that is, **<div class="quarter col">**

You can view the complete code and the finished pages using the downloaded folder for Chapter 36.

Shrink the viewport to see how the content of each column stacks one upon another in smaller viewports.

Explore the drop-down menu in various viewports.

Summary

You discovered how a drop-down feature can be added to a responsive menu. The code for two-column, three-column, and four-column pages was provided and explained.

In the next chapter you will learn how to compress a responsive menu into a single button that expands to reveal more buttons when clicked.

CHAPTER 37



Building Responsive Websites for Mobile Devices Part 3

Many responsive websites collapse the navigation menu into a single button or icon for narrow viewports. The single menu button or icon can be clicked to expose the other navigation buttons. This technique allows plenty of room on the smartphone's viewport for content. When this technique is combined with a drop-down menu, a website can have additional pages. This chapter and Chapter 38 cover this advanced technique. The first part of this chapter is purely descriptive and the second part contains a small project. The downloadable folder for Chapter 37 will enable you to test the collapsible responsive menu and its drop-down features.

This chapter has the following sections:

- A collapsible responsive menu
- Adding a header
- Summary

A Collapsible Responsive Menu

The home page in this chapter is loosely based on the website in Chapter 36. However, because the collapsible menu technique involves learning some new code and new processes, the website will be stripped of many of the sophisticated items in the previous chapter. This will allow you to concentrate entirely on the collapsible menu technique.

The desktop display of the collapsible responsive menu is shown in Figure 37-1.

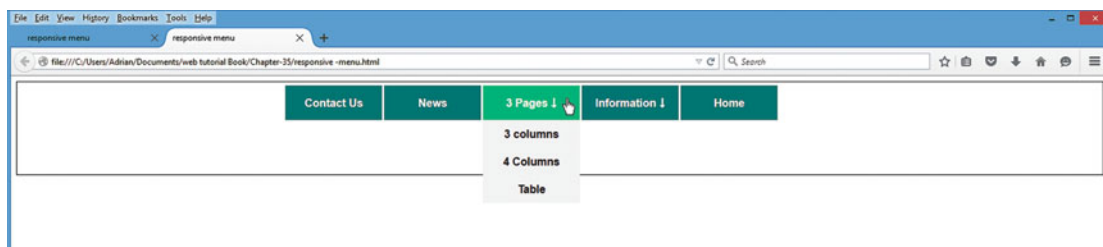


Figure 37-1. The collapsible responsive menu as it appears on a desktop or laptop computer screen

The collapsible menu is expanded when the user taps the hamburger symbol on the right of the Show/Hide Menu label.

Figures 37-2 and 37-3 show the menu collapsed and expanded in a smartphone viewport. These pictures were produced by horizontally shrinking the browser in a desktop or laptop computer.

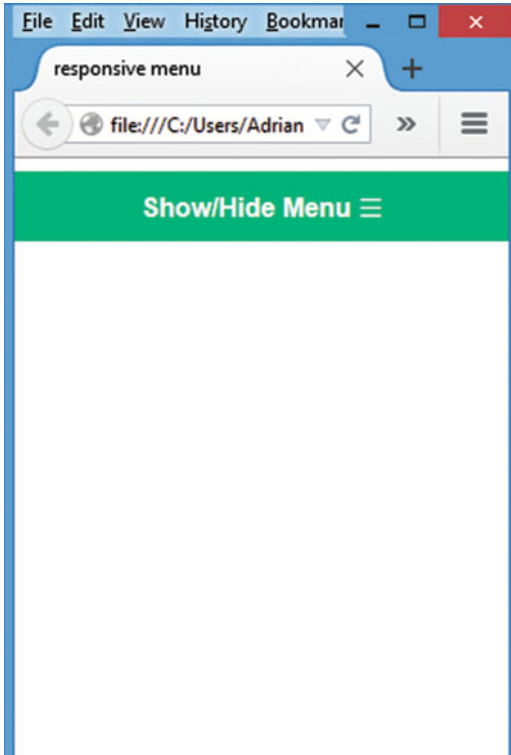


Figure 37-2. The collapsed menu

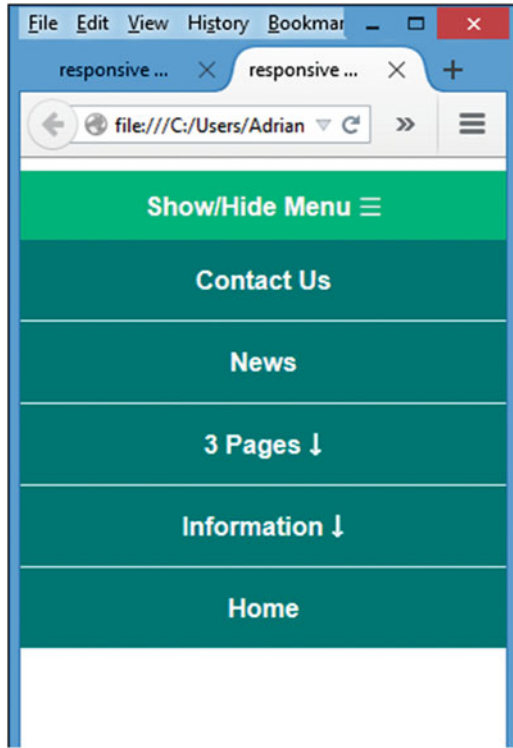


Figure 37-3. The menu revealed

The code for the collapsible responsive menu is given in Listing 37-1.

Listing 37-1. The code for creating a collapsible responsive menu (*responsive-menu.htm*). (Loosely adapted from a tutorial by Tony Thomas at medialoot.com)

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>Collapsible responsive menu</title>
<link rel="stylesheet" href="style.css">
</head>
<body>
<div id="wrapper">
<div id="drop-menu">
<label for="show-menu" class="show-menu"><strong>Show/Hide Menu &#9776;</strong></label>
<input type="checkbox" id="show-menu" role="button">
```



```

<ul id="menu">
  <li><a href="#">Contact Us</a></li>
  <li><a href="#">News</a></li>
  <li><a href="#">3 Pages &#65516;</a>
    <ul class="hidden">
      <li><a href="#">3 columns</a></li>
      <li><a href="#">4 Columns</a></li>
      <li><a href="#">Table</a></li>
    </ul>
  </li>

  <li><a href="#">Information &#65516;</a>
    <ul class="hidden">
      <li><a href="#">Amenities</a></li>
      <li><a href="#">About Us</a></li>
    </ul>
  </li>
  <li><a href="#">Membership</a></li>
  <li><a href="#">Home</a></li>
</ul>
</div>
<p>&nbsp;</p><p>&nbsp;</p><p>&nbsp;</p><p>&nbsp;</p>
</div>
</body>
</html>

```

Explanation of the HTML Code

```
<div id="drop-menu">
```

The *id* in this line provides a target so that the CSS can position the menu on the web page.

```

<label for="show-menu" class="show-menu"><strong>Show/Hide Menu &#9776;</strong></label>
  <input type="checkbox" id="show-menu" role="button">

```

This code is not easy to understand, so just try to grasp *what* it does rather than *how* it does it. In a narrow viewport the code creates a single button at the top of the display. The code does not apply to wider viewports such as a desktop or a laptop screen. In a narrow viewport, when the hamburger symbol is tapped, the menu expands and becomes several buttons. If the symbol is tapped again, the menu shrinks to a single button.

A hidden checkbox is the trigger that expands or contracts the menu. The entity `☰` displays the hamburger symbol for a menu. The item `role="button"` is not essential for this exercise; it is an aid for partially sighted or blind persons who use a screen reader. You should include it (and other “landmark roles”) in real-world websites. For information on “landmark roles” visit:

http://www.w3.org/TR/wai-aria/roles#role_definitions

```

<ul id="menu">
  <li><a href="#">Contact Us</a></li>
  <li><a href="#">News</a></li>

```

```

    <li><a href="#">3 Pages &#65516;</a>
      <ul class="hidden">
        <li><a href="#">3 columns</a></li>
        <li><a href="#">4 Columns</a></li>
        <li><a href="#">Table</a></li>
      </ul>
    </li>

```

This code begins with two normal links. These are followed by a drop-down link named *3 Pages* that contains a list of three drop-down links. The character entity *￬* displays a down-arrow.

I have used dead links (*href="#"*) to prevent you seeing the 404 “page not available” message if you accidentally click a link.

```

<li><a href="#">Information &#65516;</a>
  <ul class="hidden">
    <li><a href="#">Amenities</a></li>
    <li><a href="#">About Us</a></li>
    <li><a href="#">Membership</a></li>
  </ul>
</li>

```

This code provides another drop-down menu button leading to three more pages. The CSS code is given in Listing 37-2.

Listing 37-2. Creating the styles for the collapsible responsive menu (*style.css*).

```

/*Adapted from the tutorial by Tony Thomas at medialoot.com*/
/*Strip the menu of padding and list styling*/
ul {list-style-type:none; margin:0; padding:0; position:absolute;}

/*Create a horizontal list with spacing*/
li {display:inline-block; float:left; margin-right:3px;}

/*Style for menu buttons*/
li a {display:block; min-width:140px; height:50px; text-align:center; line-height:50px;
      font-family:Arial, sans-serif; color:#fff; font-weight:bold; background:teal;
      text-decoration:none;}

/*Hover state for top level links*/
li:~hover a {background:#19c589;}

/*Style for drop-down links*/
li:~hover ul a {background: #f3f3f3; color:#2f3037; height:40px; line-height:40px;}

/*Hover state for drop-down links*/
li:~hover ul a:~hover {background:#19c589; color:#fff;}

/*Hide drop-down links until they are needed*/
li ul {display:none;}

/*Place the smart phone drop-down links in a vertical block*/
li ul li {display:block; float:none;}

```

```

/*Prevent text wrapping*/
li ul li a {width:auto; min-width:100px; padding: 0 20px;}

/*Display the smart phone drop-down on hover*/
ul li a:hover + .hidden, .hidden:hover {display:block;}
/*Style the 'show menu' label button and hide it by default*/
.show-menu {font-family: Arial, sans-serif; text-decoration:none; color:#fff; ↵
    background:#19c589; text-align:center; padding:10px 0; display:none;}

/*Create an invisible check box which acts as a trigger to show/hide the menu in ↵
    the smart phone*/
/*Hide checkbox*/
input[type=checkbox]{display:none; }

/*Show menu when invisible checkbox is checked*/
input[type=checkbox]:checked ~ #menu{ display:block;}

/*Responsive Styles*/

@media screen and (max-width:760px){
    /*Make drop-down links appear inline*/
    ul {position:static; display:none;}
    /*Create vertical spacing*/
    li {margin-bottom:1px;}
    /*Make all smart phone menu links full width*/
    ul li, li a {width:100%; padding-left:0; padding-right:0; }
    /*create a class to display the 'show menu' link*/
    .show-menu {display:block;}
}
@media screen and (min-width:960px) {
    #wrapper {border:solid 1px #000; padding: 5px 10px; margin:auto;}
    ul {list-style-type:none; margin:0; padding:0; position:absolute; }
    #drop-menu {width:800px; position:absolute; left:25%;}
}

```

Explanation of the CSS Code

The main block of code sets general styles applicable to both desktop and smartphone viewports.

These general styles are followed by two blocks of styling that apply particularly to specific viewports (their media query headings are shown in bold type).

```

/*Strip the padding and list styling from the menu */
ul {list-style-type:none; margin:0; padding:0; position:absolute;}

```

This code removes any differences in the default margins and paddings of various browsers.

The list-style-type bullets are removed to provide clean links. The absolute position ensures that the menu's position is not influenced by other elements in the viewport.

```

/*Style for menu buttons*/
li a {display:block; min-width:140px; height:50px; text-align:center; line-height:50px; ↵
    font-family:Arial, sans-serif; color:#fff; font-weight:bold; background:teal; ↵
    text-decoration:none;}

```

This code specifies the dimensions, fonts, and colors for the menu buttons. Underlines are removed from the link labels.

```
/*Hover state for top level links*/
li:hover a {background:#19c589;}
/*Style for drop-down links*/
li:hover ul a {background: #f3f3f3; color:#2f3037; height:40px; line-height:40px;}
/*Hover state for drop-down links*/
li:hover ul a:hover {background:#19c589; color:#fff;}
```

This code specifies various hover states.

```
/*Hide the drop-down links until they are needed*/
li ul {display:none;}
```

The drop-down links are hidden until triggered by a click on a desktop/laptop or a finger tap on a mobile device.

```
/*Place the drop-down links in a vertical block*/
li ul li {display:block; float:none;}
/*Prevent text wrapping*/
li ul li a {width:auto; min-width:100px; padding: 0 20px;}
```

The comments provide sufficient explanation for these lines.

```
/*Display the drop-down on hover*/
ul li a:hover + .hidden, .hidden:hover {display:block;}
```

The plus sign here is a piece of advanced CSS and I suggest you accept it rather than try to understand it. But if you are curious and would like to learn about sibling combinators, then put an icepack on your head and prepare to wrestle with the description given at the following site:

<http://www.w3.org/TR/css3-selectors/#sibling-combinators>

```
/*Style the 'show menu' label button and hide it by default*/
.show-menu {font-family: Arial, sans-serif; text-decoration:none; color:#fff; ↵
    background:#19c589; text-align:center; padding:10px 0; display:none;}
```

The class *show-menu* is declared here because it has the attribute `display:none`; it lurks in the background waiting to be triggered.

```
/*Create an invisible check box which acts as a trigger to show/hide the menu on ↵
    narrowviewports*/
/*Hide checkbox*/
input[type=checkbox]{display:none; }
/*Show menu when invisible checkbox is checked*/
input[type=checkbox]:checked ~ #menu{ display:block;}
```

The symbol ~ (tilde) is another sibling combinator that will ensure that the drop-down effect works properly.

```
/*The Responsive Styles*/
/*Set the styles for smart phones*/
@media screen and (max-width;768px){
```

This code sets the breakpoint for smaller viewports. The following CSS applies from here onwards until a subsequent breakpoint is declared.

```
/*negative margins remove the wrapper side margins from the smart phone display*/
#wrapper {border:none; padding:0; margin:0 -10px 0 -10px;}
```

The comment is sufficient explanation.

```
/*Make the drop-down links appear in vertical blocks*/
ul {position:static; display:none;}
/*Create vertical spacing*/
li {margin-bottom:1px;}
/*Make all smart phone menu links full width*/
ul li, li a {width:100%; padding-left:0; padding-right:0; }
```

The comments are sufficient explanation.

```
/*change the .show-menu attribute so that it displays the 'show menu' link on a smart
phone*/
.show-menu {display:block;}
}
```

Because this line resides in the media query code block related to the smaller viewports, it changes from the default *display:none*; to *display:block*; The “Show/Hide Menu” link will then be visible in a smaller viewport.

```
@media screen and (min-width:960px) {
#wrapper {border:solid 1px #000; padding: 5px 10px; margin:auto;}
ul {list-style-type:none; margin:0; padding:0; position:absolute;}
#drop-menu {width:800px; position:absolute; left:25%;}
}
```

A new breakpoint is set for the desktop/laptop viewport. When the viewport is 960 pixels wide or more, then the different block of CSS code applies. A border around the wrapper is displayed on desktop viewports, and the horizontal menu is positioned.

The drop-down effect in the desktop viewport is shown at the start of this chapter in Figure 37-1.

■ **Note** To make the collapsible menu technique as clear as possible, I have deliberately omitted the home page text and the home page picture. Also I describe only a single page. In Chapter 38 I will reintroduce the more attractive desktop home page and web pages from Chapter 36 together. Meanwhile we will add one more enhancement to the current page.

The header from the pages in the previous chapter has been omitted in the downloaded folder for Chapter 37. We will now add the header so that users know what they are viewing.

Adding a Header

As stated in Chapter 35, a header with a large number of words will break up into several lines in narrow viewports. The header will then occupy too much vertical space in the smartphone viewport; therefore we will now create the briefest possible header text.

Please follow these steps:

1. Download the Chapter 37 zip folder from the book's page at [apress.com](#).
2. Unzip it in your *Web tutorial* folder (not in the *htdocs* folder).
3. Save a copy of the file *responsive-menu.html* with the new name *responsive-index.html*.
4. Insert and change the code shown in bold type in the following code snippet:

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>Compressible responsive menu</title>
  <link rel="stylesheet" href="style-2.css">
</head>
<body>
<div id="wrapper">
  <header>
    <h1>Wetlands</h1>
    <h2>Conservation Trust</h2>
  </header>
```

5. Save the file as *responsive-index.html*.
6. Open the file *style.css* in your HTML editor and anywhere near the top of the code insert the snippet of code shown next:

```
header {background-color:#CEE1BA; height:100px; padding:5px;}
header h1 {margin-bottom:-25px; margin-top:5px;}
header h2 {margin-left:30px;}
```

7. Save the changed file as *style-2.css*.
8. View the file *responsive-index.html* in a modern browser (Firefox, Safari, Opera, or Chrome; but not Edge because it may not shrink sufficiently to demonstrate a narrow smartphone display). You should now see the header as well as the menu. Expand and contract the browser viewport to see the effect.

The header and collapsed menu are shown in Figure 37-4.

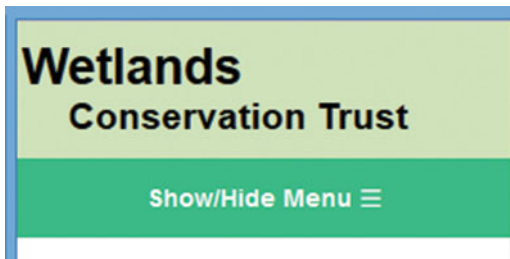


Figure 37-4. The header and collapsed menu on a phone

The user will now know the name of the website and that it has a menu that can be expanded. Try opening the collapsed menu and explore the drop-down features. Because the drop-down submenus have dead links, you will receive no response if you tap or click them.

The smartphone view should look like Figures 37-5 and 37-6.

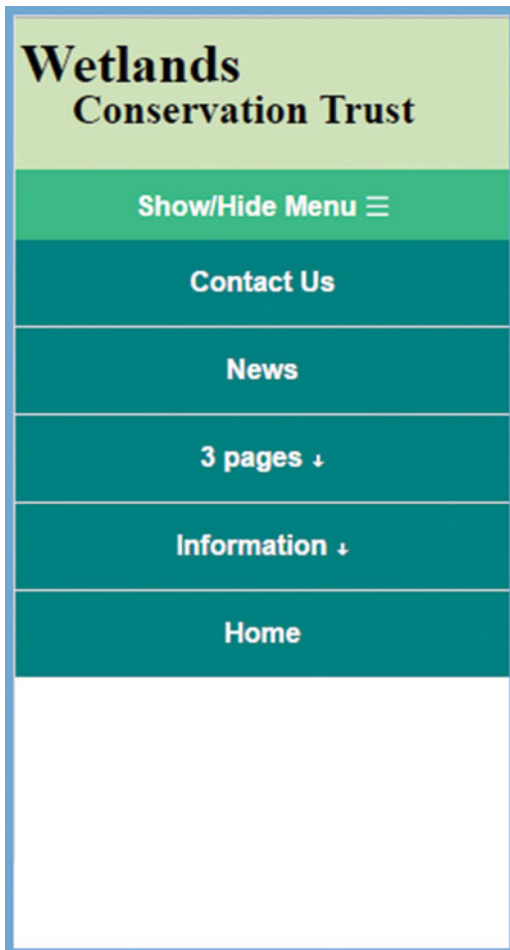


Figure 37-5. The expanded menu

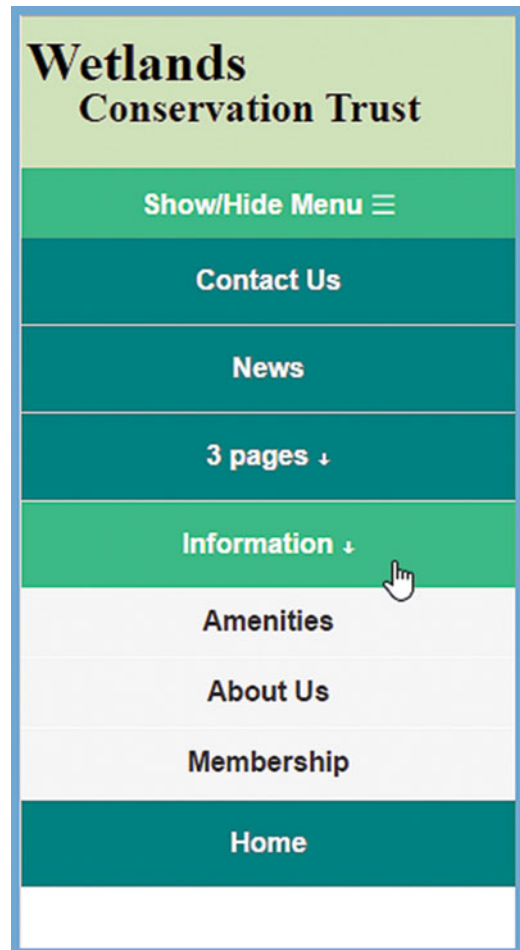


Figure 37-6. A drop-down submenu

Summary

You learned how to produce a collapsible navigation menu specially designed for responsive websites. You added a header to the responsive web page and tested it. In the next chapter you will add the pictures and some of the pages from Chapters [35](#) and [36](#) so that you can test the compressed and expanded menu to access the additional pages.

CHAPTER 38



Building Responsive Websites for Mobile Devices Part 4

This chapter begins with a reminder and summary of media query formats for a responsive website.

The chapter will combine the web pages from Chapter 36 with the compressible menu from Chapter 37.

This chapter has the following sections:

- Summarizing the use of media queries in CSS style sheets
- View the responsive pages in this chapter's website
- Brief descriptions of the new pages
- Adding new menu buttons and extra drop-down items
- Bullet points and the unordered list problem
- Responsive videos
- Additional notes on responsive websites
- Summary

Summarizing the Use of Media Queries in CSS Style Sheets

This section summarizes the description of media queries given in previous chapters.

Media queries in a CSS file may use the following typical format:

Start with the styles that are common to all viewports

Then introduce the styles for specific viewports

```
@media screen and (max-width:320px) {  
    Styles for a narrow smart phone  
}
```

```
@media screen and (max-width:768px) {  
    Styles for a wide smart phone  
}
```

```

@media screen and (min-width: 769px) and (max-width: 959px) {
    Styles for small tablets
}

@media screen and (max-width:960px) {
    Styles for a desktop or laptop computer screen
}

```

In practice, the style for very narrow smartphones is often covered by the style for a wider smartphone. The second and fourth blocks of code are often sufficient and should be used when beginning to produce a responsive website. Introduce the first and fourth blocks only if the second and fourth blocks prove insufficient during testing. The styles for the various viewports consist of small changes to the common styles.

View the Responsive Pages in This Chapter's Website

To view the pages, please follow these steps:

1. Create a folder named *Chapter-38* in your Web Tutorial folder (not the *htdocs* folder).
2. Download the file *Chapter-38.zip* and unzip it in your *Chapter-38* folder.
3. The file *index.html* is an enhanced version of the file responsive home page from Chapter-35.
4. Open the file *index.html* in Firefox, Opera, Safari, or Chrome and view it at desktop size, and then shrink it to the width of a narrow smartphone.

The home page in desktop and smartphone viewports (not to scale) is shown in Figures 38-1 and 38-2.

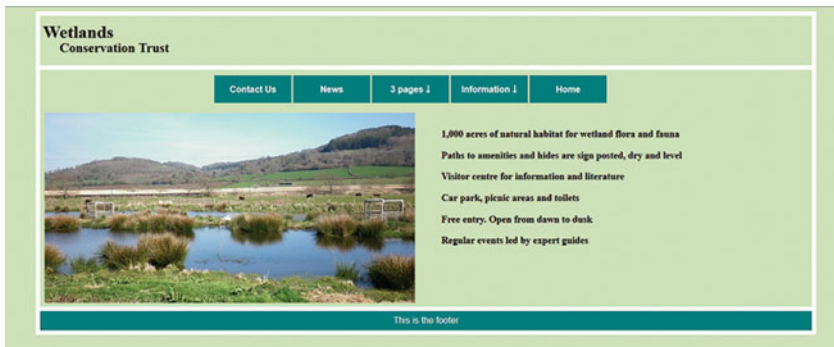


Figure 38-1. The desktop viewport



Figure 38-2. Phone

The code for the home page and the CSS code are the same as the home page in Chapter 37.

Brief Descriptions of the New Pages

To view the entire code for this chapter, download the folder from the book's page at Apress.com and load the pages into your HTML text editor (and print them for convenience if you wish). The downloadable folder contains three new pages. Space limitations prevent me from giving the full code for each page here; therefore we will only discuss the main features.

Each new HTML page contains the collapsible menu instruction as shown in bold in the next snippet:

```
</head>
<div id="drop-menu">
<label for="show-menu" class="show-menu"><strong>Show/Hide Menu &#9776;</strong></label>
  <input type="checkbox" id="show-menu">
```

The website is set up for nine pages; however, the three new pages are not fully worked. They are copies of the previous chapter's *three-column* page, and the columns are filled with faux Latin. In your own adaption of those pages you would replace the faux Latin with your own content. What is faux Latin?

Faux Latin is a common device for padding out the layout of a page prior to adding meaningful content. It is dummy text and it functions as a filler to occupy the space that will later be filled with 'real' content. This is useful when the final text and pictures are not yet available. For more information about faux Latin visit the website: <http://lipsum.com/>

A paragraph of faux Latin is included as a text file in the download for this chapter. Here is a sample of faux Latin:

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

The three pages containing faux Latin are the following: *about.html*, *members.html*, and *news.html*.

The Supplementary Style Sheet Links

I have used supplementary style sheets for the new pages. These are accessed by a supplementary link on each page. This is so that you can see clearly how the main CSS style is augmented to match different page content. In a real-world website you would try to place all the styles within the main style sheet for the website. The supplementary links are as follows:

The *Amenities* page contains a gallery of pictures; the supplementary style sheet link is the same as the previous chapter and is shown bold in the following snippet:

```
<title>Wetlands Amenities</title>
  <link rel="stylesheet" href="style.css">
  <link rel="stylesheet" href="style-gallery.css">
</head>
```

The *three-column*, *four-column*, and *table* pages used in the previous chapter had a common supplementary style sheet named *style-cols.css*. This style sheet is used again in the three new pages *about.html*, *members.html*, and *news.html*. The supplementary style sheet link is shown bold in the following snippet:

```
<title>Three columns</title>
  <link rel="stylesheet" href="style.css">
  <link rel="stylesheet" href="style-cols.css">
</head>
```

The *Contact us* page has the same supplementary style sheet link as the previous chapter and is shown bold in the following snippet:

```
<title>Wetlands C T. Contact Us form</title>
  <link rel="stylesheet" href="style.css">
  <link rel="stylesheet" href="style-form.css">
</head>
```

You can examine the supplementary style sheets in the downloadable folder for this chapter.

The Supplementary Style Sheets

The Supplementary Style Sheet for the amenities page is given in Listing 38-1.

Listing 38-1. The supplementary CSS style sheet for the amenities page (*style-gallery.css*).

```
*{box-sizing:border-box}
img {max-width:100%;}
header {height:100px;}
.figure {margin:0 1% 10px 1%; display:inline-block;}
.figure p {text-align:center; margin:5px 0 0 0;}
#gallery {margin:auto;}
#content h2 {text-align:center;}

/* RESPONSIVE */
@media screen and (max-width:380px){
  .figure {width:93%; margin-left:10px;}
}
@media screen and (max-width:760px){
  #gallery {width:95%;}
  #gallery.figure {float:left; width:49.99999%}
  .drop-shadow {box-shadow:none;}
}
@media screen and (min-width:960px){
  #gallery .figure {float:left; margin:0 1% 0 1%; width:22.99999%;}
  .drop-shadow {box-shadow:5px 5px 5px rgba(0, 0, 0, 0.5);}
  #gallery {width:95%;}
}
```

The Supplementary Style Sheet for the *Contact Us* page is given in Listing 38-2.

Listing 38-2. The supplementary style sheet for the *Contact us* page (*style-form.css*).

```
form {font-family:Arial, Helvetica, sans-serif; font-size:1em;}
form #message-field {font-family:Arial, Helvetica, sans-serif; font-size:1.3em; ↵
  font-weight:normal}
#feedback h2 {font-weight:bold; padding:0; margin:0;}
#feedback h3 {font-weight:bold; padding:0; margin:-10px 0 -10px 0;}
#feedback label {font-size:1.4em; font-weight:200; padding-top:12px; display:block; ↵
  min-width:320px;}
#caption {width:300px;}
```

```
#feedback input {min-width:265px; font-size:1.4em; font-weight:200; padding: 10px; ↵
    box-sizing:border-box;}
#caption #checkbox1 {font-size:100%; font-weight:bold; display:block;}
#message-field {font-size:1.4em; font-weight:bold; }
#feedback input[type=submit] {margin:0 auto 0 auto; background-color:green; ↵
    color:white; padding:5px; border:none; font-size:1.4em;}
#feedback input {width:180px;}
.red {color:red; font-weight:bold; font-size:110%;}

@media screen and (max-width:390px) {
    #feedback input {width:180px;}
    #feedback {margin-left:-50px; margin-top:-20px; max-width:180px;}
    h2 {min-width:320px;}
    h3 {min-width:320px;}
}
@media screen and (max-width:370px) {
    #feedback input {width:180px;;}
    #feedback form {margin-left:10%; margin-top:-20px;}
}
@media screen and (max-width:780px) {
    #feedback input {width:180px;;}
    #feedback form {margin-left:30%; margin-top:-20px; padding:20px 5%;}
}
@media screen and (min-width:761px) and (max-width:959px) {
    #feedback {margin-left:30%; margin-top:20px;}
}
@media screen and (min-width:960px) {
    #feedback form {margin-left:35%; padding:20px 5%;}
    #message-field {width:380px; margin-left:-60px;}
}
```

Listing 38-3 gives the code for the supplementary style sheet for the three-column, four-column, and table pages.

Listing 38-3. The supplementary style sheet for the columns and tables.

```
*{box-sizing:border-box}
header {height:95px;}
#content p {line-height:1.5; font-size:115%;}
/* TABLES */
table {background-color:#fff; border-collapse:collapse;border-spacing:0;width:95%; ↵
    margin:auto; border:1px #000 solid;}
th,td {border:1px #000 solid; text-align:left; padding:5px; line-height:1.5; ↵
font-size:115%;}
/*COLUMNS*/
.col,.half,.third,.twothird,.quarter{float:left; width:100%}
.col {padding:1px 12px}
/*RESPONSIVE*/
@media screen and (max-width:760px){
table {margin-top:-60px; padding:0;}
#content {margin-top:-50px; padding:0;}
#content h2 {margin-bottom:-10px; margin-top:0;}
}
```

```
@media screen and (min-width:960px){
  .quarter {width:24.99999%}
  .third {width:33.33333%}
  .half {width:49.99999%;}
  .twothird {width:66.66666%}
}
```

As time passes, websites inevitably need extra pages and extra menu buttons. This is achieved simply and quickly by inserting the new items into the drop-down menu. The next section shows you how to do this by copying and pasting.

Adding New Menu Buttons and Extra Drop-Down Items

Let's assume that the Wetlands Conservation Trust has decided that it would like a section on the website for the management committee. We will add a menu button labeled *Committee* and a drop-down list revealing the links to three new pages, that is, a page with a list of the officers, a page for agendas, and a page for recording the minutes of meetings. Listing 38-4 shows the new items in bold type.

Listing 38-4. Inserting a new menu button and three new drop-down items.

```
<li><a href="#">Information &#65516;</a>
    <ul class="hidden">
      <li><a href="amenities.html">Amenities</a></li>
      <li><a href="about.html">About Us</a></li>
      <li><a href="members.html">Membership</a></li>
    </ul>
</li>

    <li><a href="#">Committee &#65516;</a>
      <ul class="hidden">
        <li><a href="officers.html">Officers</a></li>
        <li><a href="agendas.html">Agendas</a></li>
        <li><a href="minutes.html">Minutes</a></li>
      </ul>
    </li>
```

The change was achieved in less than five minutes by copying the block of code for the Information button and pasting it into a space below that block. Then the labels and file names were changed to match the extra pages. I saved the changed file as *index-committee.html* so that you can explore it. In a real-world website the new block of code would be copied and pasted into the menu in each page of the website.

The result is shown in Figures 38-3 and 38-4 (not to scale).

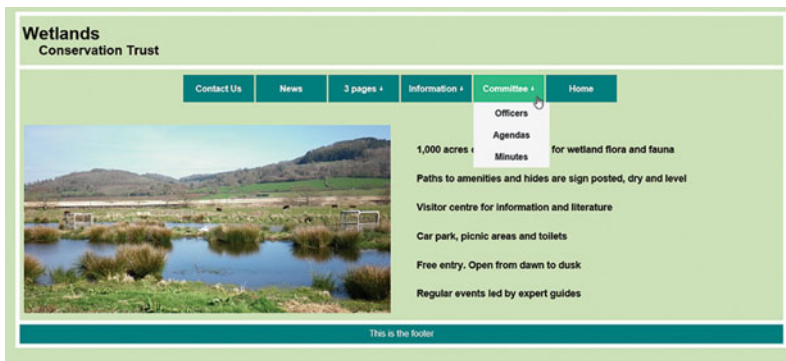


Figure 38-3. The new *Committee* menu button and its drop-down links

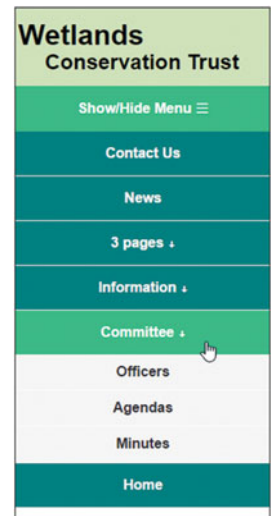


Figure 38-4. Smartphone

Note The three new *Committee* pages are not included in the downloadable folder for this chapter. Therefore don't try clicking the *Committee* button's drop-down links. If you do click the links, you will see the "Page not available error message."

We will now examine an unordered bullet problem that often puzzles beginners.

Bullet Points and the Unordered List Problem

Unordered list tags are used for bullet points as well as for links. Our new style sheet targets the 14 unordered list tags for the menu; the menu has the tags *ul*, *li* or both, or a complex combination of both. The menu style will interfere with any unordered bullets applied to paragraphs of text.

Let's suppose that you wished to use bullet points to emphasize the items on the home page. You might assume that using an unordered list using `` and `` tags would suffice. However, it does not work like that because those tags are also used in the drop-down menu, and the tags are styled to suit the menu. What suits the menu won't suit anything else that uses `` and `` tags.

We could use a special class for the unordered bullets for text, but this is tricky because the styles for the menu tend to override any attempt to create bullet points for text. I did manage to use a class successfully several times, but it took a great deal of frustrating trial and error.

Fortunately there is a simple workaround. Bullet points can be created for text by abandoning the unordered list, and using paragraph tags with an entity that will create the bullet points.

The code that does *not* work is shown in Listing 38-5.

Listing 38-5

```
<div id="col-2">
  <ul>
    <li>1,000 acres of natural habitat for wetland flora and fauna</li>
    <li>All paths to amenities and hides are sign posted, dry and level</li>
```

```

    <li>Visitor centre for information and literature</li>
    <li>Car park, picnic areas and toilets</li>
    <li>Free entry. Open from dawn to dusk</li>
    <li>Regular events led by expert guides</li>
  </ul>
</div>
```

Compare that listing with the code that does work in Listing 38-6.

Listing 38-6

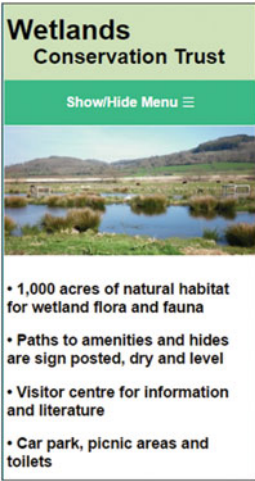
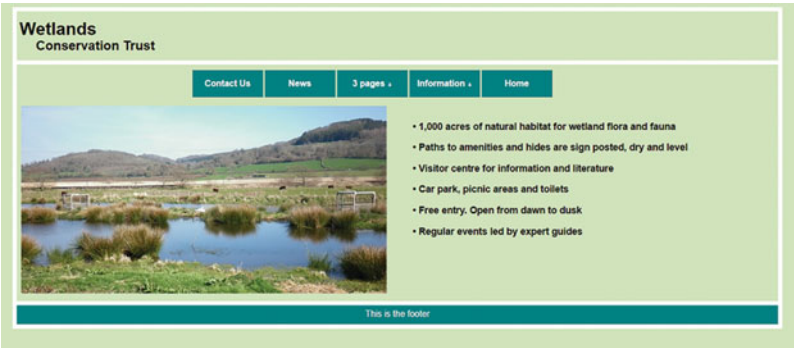
```

<div id="col-2">
  <p>&bull; 1,000 acres of natural habitat for wetland flora and fauna</p>
  <p>&bull; Paths to amenities and hides are sign posted, dry and level</p>
  <p>&bull; Visitor centre for information and literature</p>
  <p>&bull; Car park, picnic areas and toilets</p>
  <p>&bull; Free entry. Open from dawn to dusk</p>
  <p>&bull; Regular events led by expert guides</p>
</div>
```

By using paragraph tags and the entity for a bullet (•) we can easily add bullet points to text. To create the bullet points on the home page, please follow these steps:

- 1. Open *index.html* in your text editor.
- 2. Use *Save As* to save a copy of *index.html* as *index-bullets.html*.
- 3. In the file *index-bullets.html*, add the bullet entities as shown in bold in Listing 38-6.
- 4. Save the file.

The text with bullets is shown in Figures 38-5 and 38-6 (not to scale).



Figures 38-5 and 38-6. The text now displays bullets in a desktop viewport and a smartphone viewport

Responsive Videos

You may remember from Chapter 24 that, with HTML5 the format for loading a video into a web page is as simple as loading an image. Here is an example:

```
<video width="420" height="315" controls="controls">
  <source src=video/somevideo.mp4 type='video/mp4;'>
  <source src=video/somevideo.ogv type='video/ogg;'>
  <source src=video/somevideo.webm type='video/webm;'>
  Your browser does not support the video tag.
</video>
```

Note that in the third line, the video is .ogv and the type is .ogg

A responsive image must not specify the dimensions of the image, and the same rule applies to a responsive video. In the next snippet the dimensions have been removed from the first line:

```
<video controls="controls">
  <source src=video/somevideo.mp4 type='video/mp4;'>
  <source src=video/somevideo.ogv type='video/ogg;'>
  <source src=video/somevideo.webm type='video/webm;'>
  Your browser does not support the video tag.
</video>
```

For responsive images the CSS is as follows:

```
img {max-width:100%; height:auto;}
```

Similarly the CSS for a responsive video would be as follows:

```
video {max-width:100%; height:auto;}
```

Listing 38-7 gives the code for *butterfly.html*, which is in the downloadable folder for Chapter 38: The page contains an internal style to position the video on a desktop screen.

Listing 38-7. The code for the file *butterfly.html*.

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width,initial-scale=1">
<meta content="text/html; charset=utf-8" http-equiv="Content-Type">
<title>Butterfly</title>
  <style>
    video {max-width:100%; height:auto;}
    @media screen and (min-width:960px) {
      #content {margin-left:26%;}
    }
  </style>
</head>
<body>
<div id="content">
<h2>Demonstration of Responsive video</h2>
```

```

<video controls="controls">
  <source src=videos/butterfly-movie.mp4 type='video/mp4;'>
  <source src=videos/butterfly-movie.ogv type='video/ogg;'>
  <source src=videos/butterfly-movie.webm type='video/webm;'>
  Your browser does not support the video tag.
</video>
<h3>This video uses the new HTML5 video tag</h3>
</div>
</body>
</html>

```

The result of this responsive video page is shown in Figures 38-7 and 38-8 (not to scale).



Figure 38-7. The desktop view

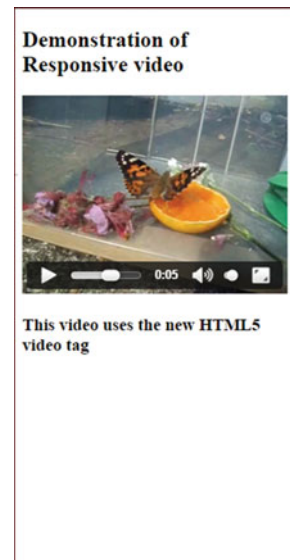


Figure 38-8. Phone

Nothing could be easier, but what about YouTube or Vimeo videos that always specify a width and height?

Responsive YouTube/Vimeo Videos

You have already learned that responsive elements must not have width and height dimensions; otherwise they cannot shrink and grow to match various viewports. Unfortunately, YouTube and Vimeo videos do specify fixed dimensions as shown in the following cut and paste code for a YouTube video:

```

<iframe src="https://www.youtube.com/embed/wS0hqBJasgw" width="560" height="315" ↵
frameborder="0" allowfullscreen></iframe>

```

I have tried many HTML5/CSS workarounds but the process was very fiddly. The result would not cover all viewports and was particularly disappointing for viewing on tablets.

I recommend that you try to obtain the original video clip that was used to produce the YouTube/Vimeo video and then use HTML5 and CSS as shown in the previous section titled “Responsive Videos.” If the original video clip is not available and you must use the YouTube/Vimeo clip, try the JavaScript/jQuery method *fitvids*.

The file *fitvids.js* must be in your main folder. In the downloadable folder, it is in a subfolder called *js*.

The key to this technique is to wrap the video within a container that is made responsive.

The file *fitvids.js* makes the container for the video responsive; in this example the container is a `<div>` that has been given the class **.video-container**.

The file *fitvids-test.html* is in the downloadable folder for Chapter 38 and the code for the file is given in Listing 38-8.

Listing 38-8. The code for the responsive YouTube video (*fitvids-test.html*)

```
<!DOCTYPE html>
<html>
  <head>
    <meta name="viewport" content="width=device-width,initial-scale=1">
    <title>Responsive Video using fitvids</title>
    <style>
      body {background:#FFF;}
      video {max-width:100%; height:auto;}
      iframe, embed, object {max-width:100%;}
      .video-container {width:65%;margin:0px auto; background:#FFF;}
      @media screen and (max-width:780px) {
        .video-container {width:95%;margin:0px auto; background:#FFF;}
      }
    </style>
  </head>
  <body>
    <div class="video-container">
      <h2>A Responsive YouTube Video using <em>fitsvid.js</em></h2>
      <iframe width="425" height="349" src="http://www.youtube.com/embed/cUrSf-YDdyU" ↵
        frameborder="0" allowfullscreen></iframe>
      <h3>The YouTube video made responsive by using <em>fitvids.js</em></h3>
    </div>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.2/jquery.min.js"></script>
    <script src="js/jquery.fitvids.js"></script>
    <script>
      $(".video-container").fitVids();
    </script>
  </body>
</html>
```

The technique also works for Vimeo videos. The URL link for Vimeo is a great deal longer than for YouTube, and it usually has the following format:

```
<iframe src="https://player.vimeo.com/video/XXXXXXXX" width="640" height="480"
frameborder="0" webkitallowfullscreen mozallowfullscreen allowfullscreen></iframe><p><a
href="https://vimeo.com/XXXXXXXX">Untitled</a> from <a href="https://vimeo.com/
userXXXXXXXX">Adrian West</a> on <a href="https://vimeo.com">Vimeo</a>.</p>
```

Explanation of the Code for *fitvids-test.html*

```
<head>
  <meta name="viewport" content="width=device-width,initial-scale=1">
  <title>Responsive Video using fitvids</title>
  <style>
    body {background:#FFF;}
    video {max-width:100%; height:auto;}
    iframe, embed, object {max-width:100%;}
    .video-container {width:65%;margin:0px auto; background:#FFF;}
    @media screen and (max-width:780px) {
      .video-container {width:95%;margin:0px auto; background:#FFF;}
    }
  </style>
</head>
```

The viewport statement is placed just below the opening **<head>** tag. The class *.video-container* is set to 65% of the desktop viewport width. If it was set to 100% the video would fill the viewport and it would be too fuzzy (pixilated). The media query sets the width of the video to 95% of the viewport's width for viewports less than 780 pixels wide.

```
<body>
  <div class="video-container">
    <h2>A Responsive YouTube Video using <em>fitsvid.js</em></h2>
    <iframe width="425" height="349" src="http://www.youtube.com/embed/cUrSf-YDdyU"
      frameborder="0" allowfullscreen></iframe>
    <h3>The YouTube video made responsive by using <em>fitvids.js</em></h3>
  </div>
```

The code provided by YouTube is copied and embedded within the video container.

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.2/jquery.min.js"></script>
<script src="js/jquery.fitvids.js"></script>
  <script>
    $(".video-container").fitVids();
  </script>
</body>
</html>
```

The first line is the URL that fetches the latest script library from Google's content delivery network. To find the URL for the latest version of the script visit:

<https://developers.google.com/speed/libraries/#libraries>

The second line of code loads the *jquery.fitvids.js* file from the *js* folder.

Note that all the JavaScript is placed just before the final body tag. Traditionally, JavaScript was placed in the **<head>** section. Technically, it can be placed anywhere in the HTML page; however, for mobile devices it is better to place it as far down the page as possible. This ensures that the page content is displayed before the JavaScript activates. In other words, the user has something to look at while the JavaScript lumbers into action. If the JavaScript is in the **<head>** section, the user could be staring at a blank page until the JavaScript has fully loaded and finished executing.

Additional Notes on Responsive Websites

This section contains some additional information and definitions that apply particularly to responsive websites for mobile devices.

Content Strategy

Content strategy is a buzzword you will find in several Responsive Web Design books and tutorials. Strategy is always the result of planning: no sensible web producer will start a website without planning the content, that is, the number of pages, their topics, and the navigation that will access those pages. The resulting plan is the content strategy. The importance of content strategy increases when designing a Responsive Website. When you have outlined your strategy you can select the content that a mobile user would consider essential; in other words think mobile content first. You would then select an appropriate design/style for the content. Without content, design/style is mere decoration; a good design/style will support and enhance the content.

My content strategy for the project in this chapter deviates from the above paragraph because I have included two pages purely for teaching purposes, that is, the three-column and four-column pages would not be essential for a mobile user who was viewing the Wetlands Conservation website. However, those pages could be renamed and revised to make them relevant to the website.

Fonts

The rules for fast loading fonts are reasonably simple. There are 14 web-friendly fonts that are common to Windows and Apple's iOS; these are called system fonts and are very fast loading. Of these the five best fonts for responsive websites are Arial, Courier New, Tahoma, Times New Roman, and Verdana.

Although Times New Roman is the easiest font to read for printed matter, Sans Serif fonts such as Arial and Verdana look cleaner on web pages, especially on mobiles. Courier New should only be used for small paragraphs such as quotations from printed documents or other websites.

CSS3 allows the use of some fancy online fonts but these have to be fetched from other Internet sources every time a page loads; this action slows down the delivery of your content so it is best avoided.

Some proprietary fonts can be hosted locally in your main folder, which means they don't have to be fetched from an external website; however, recent tests have shown that this gives very little speed advantage.

See this website for a useful speed comparison:

<https://www.keycdn.com/blog/web-font-performance/>

Avoid using pixels for font sizes; use percentages or ems. The general equation in terms of baseline font sizes in CSS is as follows:

$$100\% = 1 \text{ em} = \text{approximately } 16\text{px} = \text{approximately } 14\text{pt}$$

Usability experts recommend a font size of at least 1 em (100%). Also, consider the line height of your text; this does not affect the width, but it does affect the page height. Most experts recommend a line height of 1.2 to 1.45 em for good readability. Web designers should consider the most important aspect relating to responsive fonts, that is, what does the text look like in various viewports. In the smartphone viewport, is the font too tiny, or is the font so large that the smartphone column contains only two or three words? If so, adjust the font size in a media query for the smartphone viewport. In the desktop viewport, does the text stretch right across the screen; if so, break it down into columns; the accepted maximum width of a column is 5.25 inches (133 mm). A website for children should have an even smaller column width.

Always view your text in various viewports by using actual devices or the online tester <http://quirktools.com/screenfly/>. The reason for this recommendation is that recent mobile devices report that their viewport is smaller than their actual viewport; this compensates for their high pixel densities and can cause the text to be much larger than we intended.

Accelerated Mobile Pages (AMP)

What problem is AMP trying to solve?

Some websites can be slow to load on smartphones and tablets and this problem is increasing. Viewing websites on mobiles has become common practice and web designers are producing websites to match the demand. Web designers should be aware of the reasons why pages load slowly. Some are directly under the control of designers as follows:

JavaScript

Many big-name publishers including Twitter and the BBC are working with Google to develop an open source scheme named Accelerated Mobile Pages (AMP). The intention is to deliver streamlined pages stripped of most JavaScript elements.

Many web designers wanting a quick way of creating a responsive solution are resorting to ready-made frameworks and online conversion programs; others are using CMS programs. These are packed with JavaScript and they contain many functions that may not be wanted on the website. The solution is to avoid using frameworks, CMS, and online solutions for creating a responsive website.

Other Avoidable Features That Cause Slow Loading

Items that are not directly related to the purpose of the website can cause slow loading.

These are the following: fancy fonts, Flash, social media buttons for expressing “likes” or “share” buttons, advertisements that are generated on the page by links to the advertisers, and Facebook's Instant articles. In other words, any item that automatically links to an external website and fetches something from that website can cause slow loading. Stripping out the extra web page elements will speed up the delivery of your content. If you must use advertisements fetched from other firms, try to write the page so that the advertisement loads last, after your own content has loaded. Links to advertisers and trackers can cause the delivery of content to hang if the advertisers' or trackers' networks are having problems.

Many websites are riddled with items that run in the background causing slow delivery and using up bandwidth; often they are not related to the theme of the website. These can be trackers that watch what users are doing. They not only contain huge amounts of JavaScript, but they send out dozens of calls to advertisers in response to the user's searches and clicks. By writing your own HTML5 and CSS code you can avoid these unnecessary elements and ensure that your website will load quickly.

Optimize all images so that they have the smallest possible file size consistent with quality. Auto start video and audio elements are notorious for slowing down the delivery of your content. Ensure that the video or audio must be clicked by the viewer to start them running.

Items That Are Not Under the Control of the Designer

Two items that cause slow delivery are not under the control of the designer as follows: (i) Mobile connections are inherently slower than the average home network and (ii) Wi-Fi will be slow if many others are using the connection.

Summary

This chapter started by summarizing the use of media queries in CSS style sheets. You were then able to view the responsive pages that are included in the downloadable folder for the chapter. You were given brief descriptions of the new pages, and you learned how to add new menu buttons and extra drop-down items to a responsive website. You learned how to overcome a problem with bullet points and unordered lists, and how to create responsive videos. The chapter provided some additional notes to help you avoid creating responsive websites that load slowly.

In the next chapter you will find advice on using Content Management Systems.

CHAPTER 39



Avoiding Some of the Pitfalls of a CMS Website

DIY (CMS) programs have helped millions of unskilled people to produce millions of dreadful websites.

Content management systems (CMS) such as WordPress, Joomla, etc., are not in themselves bad; in fact they are excellent tools that allow untrained persons to produce websites. It is the untrained persons who are responsible for the large number of dreadful websites. The CMS programs must also share some of the blame because they provide little or no guidance on good design and web-user psychology.

This chapter contains the following sections:

- Pitfalls that can be avoided
- An example of an acceptable CMS design
- CMS pitfalls that are very difficult or impossible to avoid
- Summary

The first primitive websites had only one page and the page was often extremely long. The big breakthrough came when Sir Tim Berners-Leigh invented hyperlinks and the World Wide Web. As a result, multipage sites could then be accessed from a short home page by means of menu buttons called hyperlinks. These imitate a book's index or contents list. The home page can now be short and clean with the focus directed to the menu buttons and to an explanation of the main purpose of the site.

Pitfalls That Can Be Avoided

A mile-long home page that lacks focus. DIY home pages are frequently a very long, bewildering, spotty mess. The ideal home page should be the height of one desktop screen and it should have the absolute minimum of text describing what the site is about. This stimulates the user's curiosity so that they will then focus on the navigation menu and hopefully explore other pages in your site.

When a user looks at your home page they will ask, "Is this what I am looking for?" The answer to the user's question must be *immediately* visible on the page, not buried within a jumbled mass of items. If users have to search and scroll around to find the information they are seeking, then you have failed.

Unclear navigation. The means by which a user navigates from page to page must be absolutely obvious. On most DIY sites, the navigation is swamped within a great deal of clutter. Often the clutter consists of multiple ways of accessing the same pages, but this is crazy because it defeats the whole point of hyperlinks. The ideal solution is to have a clear and prominent menu. The home page must be designed to focus the user's attention on the navigation menu. Each page of the web site must cover only one topic, and these pages must also focus the user's attention on the navigation menu. Never leave the visitor trapped and bewildered.

Inappropriate color choice. This indicates that the producer not only lacks training but also lacks common sense. Examples of inappropriate colors I have encountered: (i) a carpet cleaner's DIY site that was predominately black and gray, (ii) a wedding photographer's DIY site that was predominately black, (iii) a nature conservation site that was orange shading to peach.

Amateurs using DIY programs often splatter the page with patches of color just because they can. Don't do it.

Large areas of garish colors should be avoided, as they can be repellent. Pastel colors are attractive, and because they are restful, the page will retain the user's interest. Very dark backgrounds are unattractive, and black is suitable only for a funeral parlor or a witches' coven. White or colored text bleeds into a black background.

Always use a strongly contrasting color for text. Black text on white background gives the maximum contrast. So many poor DIY sites use pale text on a pale background, making the text difficult to read.

Excessive use of text and poor presentation of text. Pages crammed with text are a real turn off. When persons visit websites they are looking for information, and they want it quickly. Nobody searches websites for a good read. Textual information must be presented *concisely* and in a format that is easy to read. Reading web pages is much more tiring than reading a book or magazine. The real killers are the following (i) verbose text, (ii) small fonts, (iii) justified paragraphs, (iv) lines that go right across a page (always use columns like a newspaper or magazine), (v) paragraphs of italic text, (vi) fancy fonts such as comic sans or imitation handwriting.

Do not underline text. For emphasis, use a larger font or bold text. Underlining is a throwback to the ancient mechanical typewriter and is now reserved for links to email addresses or external websites.

DIY sites are often inward looking. They are no more than scrapbooks for members of the organization. This conflicts with the main purpose of a website, which is to publicize the organization to the world outside the organization (hence the World Wide Web). Why go to all that effort to please a handful of society members when there are billions of viewers out there?

The first questions to ask yourself when designing a website are these: "What is the target audience and what should the website achieve when it is up and running?"

Video/slide shows should not be used on the home page. In fact, anything that flashes, moves, flickers, or blares out "music" should never be used on the home page; this shifts the focus from the essentials and usually turns users away.

Videos/slideshows and audio on other pages. These should never be self-starting, and always provide the users with controls so that they can choose whether to run the media.

CMS programs do not provide the following warnings. Search Engines cannot read text embedded in pictures. Never insert the main heading text or any other important text as part of a picture; always overlay the image with HTML text and use text sizes that imitate the hierarchy of newspaper headings.

An Example of an Acceptable CMS Design

Figure 39-1 shows an attractive and useful home page produced with the WordPress CMS program.

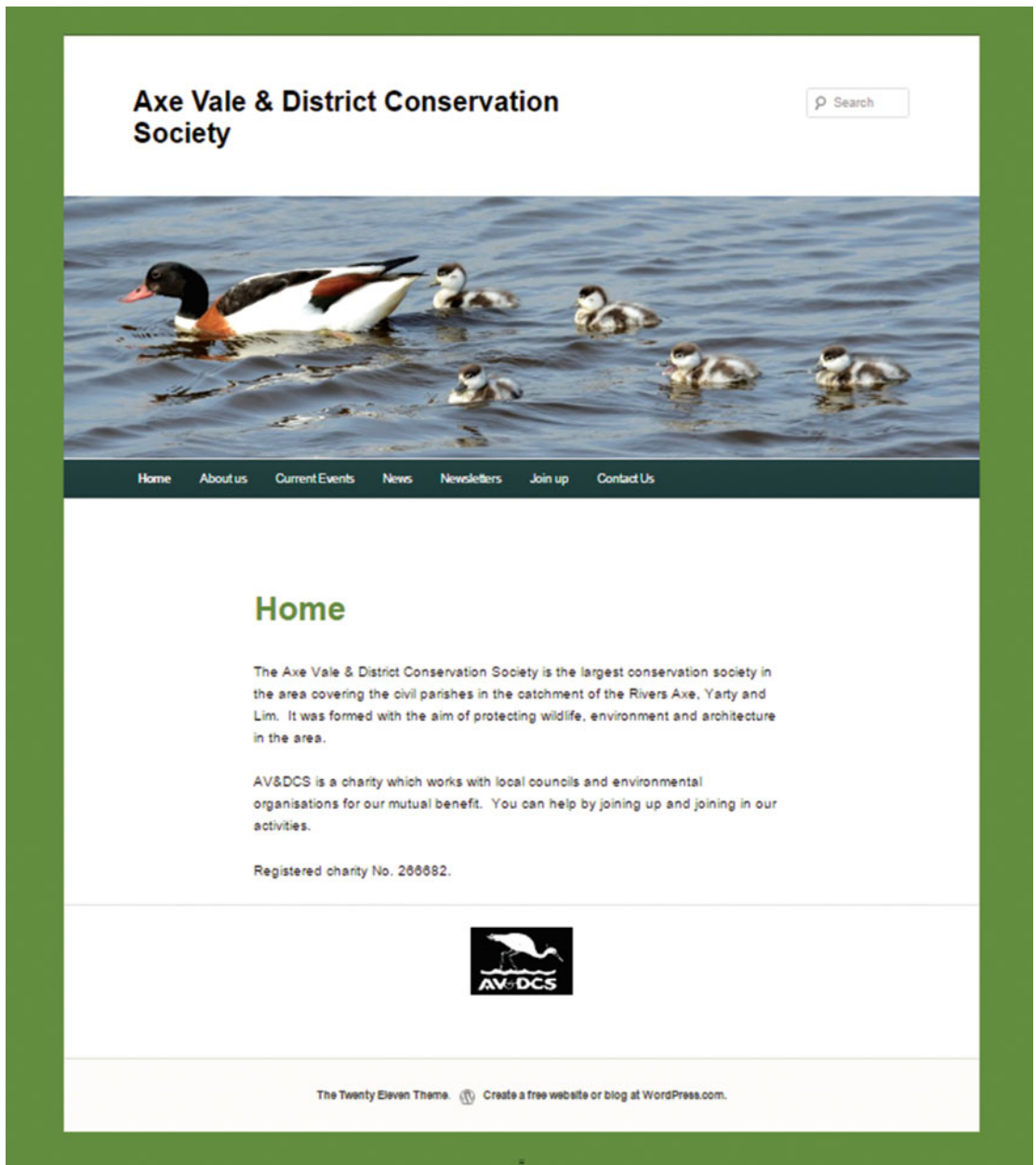


Figure 39-1. The home page of the Axe Vale and District Conservation Society

I am grateful to the Axe Vale and District Conservation Society for allowing me to use Figure 39-1.

The Axe Vale and District Conservation Society was given a copy of this chapter and it followed most of the advice contained within the chapter. The home page is uncluttered, short, and attractive. It focuses the user's attention on the two vital elements, that is, it concisely describes what the website is about, and it has a prominent menu that clearly points to the other pages on the website.

The website was produced on the latest version of WordPress; therefore the drop-down menu buttons do not use the usual CMS JavaScript menus. This means that search engines can find all the pages on the site. Earlier versions of WordPress had JavaScript menus, but these prevented search engines from penetrating past the home page.

The text would be easier to read if it was darker and perhaps bold so that it contrasts more strongly with the white background. The lines of text are rather long and would be improved if they were arranged in two columns. Despite these minor criticisms, the website is a pleasure to use. You can view it at the following address:

<https://avdcs.wordpress.com/>

The domain name is not very memorable, but despite this, search engines quickly find the website when the key phrase *axe valley conservation society* is used because it is unique.

CMS Pitfalls That Are Very Difficult or Impossible to Avoid

With training, some CMS pitfalls can be avoided as discussed in the previous section. Others cannot be avoided because they are inherent in the CMS programs.

Poor code that won't validate. Even though the CMS program will convert the amateur's input into HTML and CSS, the generated code may not validate. Sometimes this is due to the CMS robot generating its own idiosyncratic version of HTML, or it can be that the amateur "designer" does not understand the importance of validating the code.

Slow-loading pages. CMS websites load slowly because each page contains 6 to 10 times more code than an equivalent professionally coded website. This extra code is there to convert the user's text and pictures into HTML and CSS. All CMS sites load an enormous amount of baggage into your root folder; this consists of page after page of JavaScript and PHP code. For example, a basic 5-page website using HTML5 and CSS3 will need only 2 folders and 6 files. Using a CMS package for the same website results in 17 folders with an average of 30 files in each, plus dozens of PHP and JavaScript files and several additional folders so that the CMS programs can administer the website.

Some CMS programs put a page or two of CSS code on EVERY page of the website. This makes a mockery of CSS, which was designed for speed and convenience, that is, there should be a single CSS style sheet that loads once and is retained in the memory of the browser for instant application to every page in the website. Some special pages may need an internal or supplementary style, but often these can be incorporated into the main style sheet.

If the CMS program automatically adds the style sheet to every page, there is nothing you can do about it, because the CMS program is in control. By using HTML and CMS you can gain control over the website and you can decide where the style sheet will be located.

Difficulty in fine-tuning the site and pages. The "designer" has very little control over the way the website behaves. With CMS kits, if you want to tweak the code, or you wish to customize the pages, you need to learn HTML, CSS, and perhaps a little bit of PHP and JavaScript. This defeats the object of CMS programs. Even if you learn HTML and CSS you may be prevented from tweaking the code because some CMS programs use an idiosyncratic version of HTML, CSS, and PHP that does not conform to the official standard.

DIY sites tend to look alike. When using a CMS program, untrained users tend to use the templates that tens of thousands of others are using. This is despite the fact that CMS sites provide a range of templates. It is easier to choose from the first few templates rather than explore the available range. By using HTML and CSS code, you can produce unique websites exactly to your own specification.

DIY websites are vulnerable to spam and hacking. WordPress recently issued a warning that hackers had accessed thousands of its websites. If the CMS site displays email addresses, they are probably not protected against spam spiders. A professional web designer using HTML and a tiny piece of JavaScript will encode all the email addresses for security, and to prevent spam.

Summary

In this chapter you learned that some of the pitfalls associated with producing websites with CMS programs can be avoided, although you can never be in full control of the websites. You discovered that most of the problems are caused by lack of training in design and web-user psychology. You learned that by acquiring some knowledge of design and web-user psychology, and by paying attention to careful planning, it is possible to produce CMS websites that are attractive and useful. An example of an acceptable CMS website was provided and explained. Examples of pitfalls that were impossible or difficult to avoid were also described and explained.

In the next chapter you will learn how to go live with your websites, and you will discover how to validate your code using the online *w3.org* validator.

CHAPTER 40



Go Live and Validate Your Website

You will require a domain name, a host, and a file transfer program to launch your website so that the whole world can view it. This chapter gives guidance on choosing a domain name and a host. Instructions are provided for installing and using a free file transfer program that will enable you to place a copy of your website in the host's folder. The section on validating your web pages will help you to troubleshoot your code and ensure that your code conforms to the W3C standard.

This chapter contains the following sections:

- Choosing a domain name
- Choosing a host
- Download your free FTP client
- Getting to know your FTP client
- Validate your web pages
- Validate the CSS
- Summary

Choosing a Domain Name

Choose a domain name carefully. An overseas suffix will cost more to register than a local one, for instance, in the United Kingdom, using the American suffix *.com* will be more expensive compared with using *.co.uk*, or *.uk* or *.org.uk*. If possible, make the domain name match the content of the website.

Short names are best and try to avoid a hyphenated name; for instance, www.smallgarden.co.uk is better than www.the-small-garden.co.uk. Sometimes a hyphenated name is necessary to avoid confusion, for example, *spring-garden* looks less confusing than *springgarden*.

Type the entire proposed name into the address field of a browser; this will tell you whether the name is already in use. Then type part of the name (say *small garden*) into a search engine: if a large number of similar names appear in the search results, your website is likely to be confused with those already in use, so you will need to think again. I thought my computer help website would look good with the domain name www.computerhelp.co.uk; however, typing *computer problems* or *computer help* into a search engine resulted in several million results. I also found that a website named *computerhelp.com* already existed. Because I live in an area in Devon, UK, called The Coly Valley, I adopted the name www.colycomputerhelp.co.uk and this proved to be unique. Although the name might seem odd, I advertised the website in my monthly magazine articles and it soon became popular.

As a final check, visit a website that offers hosting and enter the name you wish to register. The site will run a search and tell you whether your domain name is available or not. If it is not available, the site will usually suggest other possibilities. You are not committed until you pay for the domain name.

Choosing a Host

The whole world can view your website once it is located on a host. The host is a remote server where your website will be housed. If you type “website hosting” into the address bar of your browser you will be presented with 143 million results. Narrow the search by selecting hosts based in your territory because you will need support at some time; telephoning an overseas help line is seldom cheap. Also, you may pay more because of the commission on exchange rates converting from one currency to another.

Comparison sites can be confusing and you cannot be sure that they are truly impartial. I found that several hosts that advertise extensively in computer magazines and on TV, these hosts were generally expensive and did not offer all the features that you might eventually need when your skills improve. Many of these advertised sites offer a tempting low monthly cost for three or six months, but ignore that. Base your decision on the price for subsequent months. Cost is not the only criterion; other factors include the following: reliability, customer service, the ability to host PHP pages, the possibility of hosting at least one MySQL database, and the number of free email accounts.

Most hosts offer free domain name registration for the first year; the majority offer a substantial reduction for hosting if you pay upfront for two years or more.

For UK readers I recommend that you try Bargain Host; it is a reasonably priced host with all the features you could possibly want and the customer service is superb.

See <http://www.bargainhost.co.uk>

■ **Note** The next section assumes that you have registered your Domain name and paid for hosting.

Download Your Free FTP Client

To transfer web pages to the host you will need an FTP client (File Transfer Protocol client). Several free FTP clients are available. FileZilla is probably the most popular free, open source FTP client; download it from here:

<http://filezilla-project.org/download.php>

At the time of writing the latest version was FileZilla 3.19. The installation process is quite straightforward; just follow the instructions on the screen.

The border color can vary with each new version; therefore the interface borders in this chapter’s screen shots may not be the same in the latest versions.

When you have downloaded and installed your FTP client you need to configure your computer to allow the FTP client to connect to a host over the Internet.

Allow the FTP Client to Connect to a Host

The Windows Firewall will block any attempt to connect directly to a remote host because that is what firewalls are designed to do. Fortunately, the firewall can be configured to allow the FTP client to connect to a host. Note that in Windows 8.1 and 10, the firewall configuration interface refers to programs as Apps.

The configuration steps for Windows 8, 8.1, and 10 are as follows:

1. In Windows 8, 8.1 and 10 open the search field then type **firewall**.
2. In the search results, click *Allow an App through the Windows firewall*. See Figure 40-1.

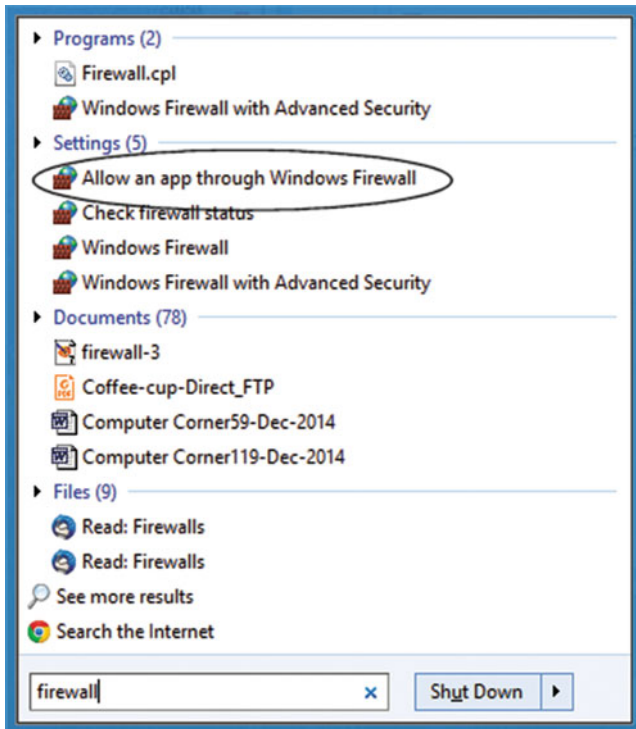


Figure 40-1. Begin to configure the Firewall

3. In the next window named **Allowed Applications**, click the **Change settings** button. See Figure 40-2.

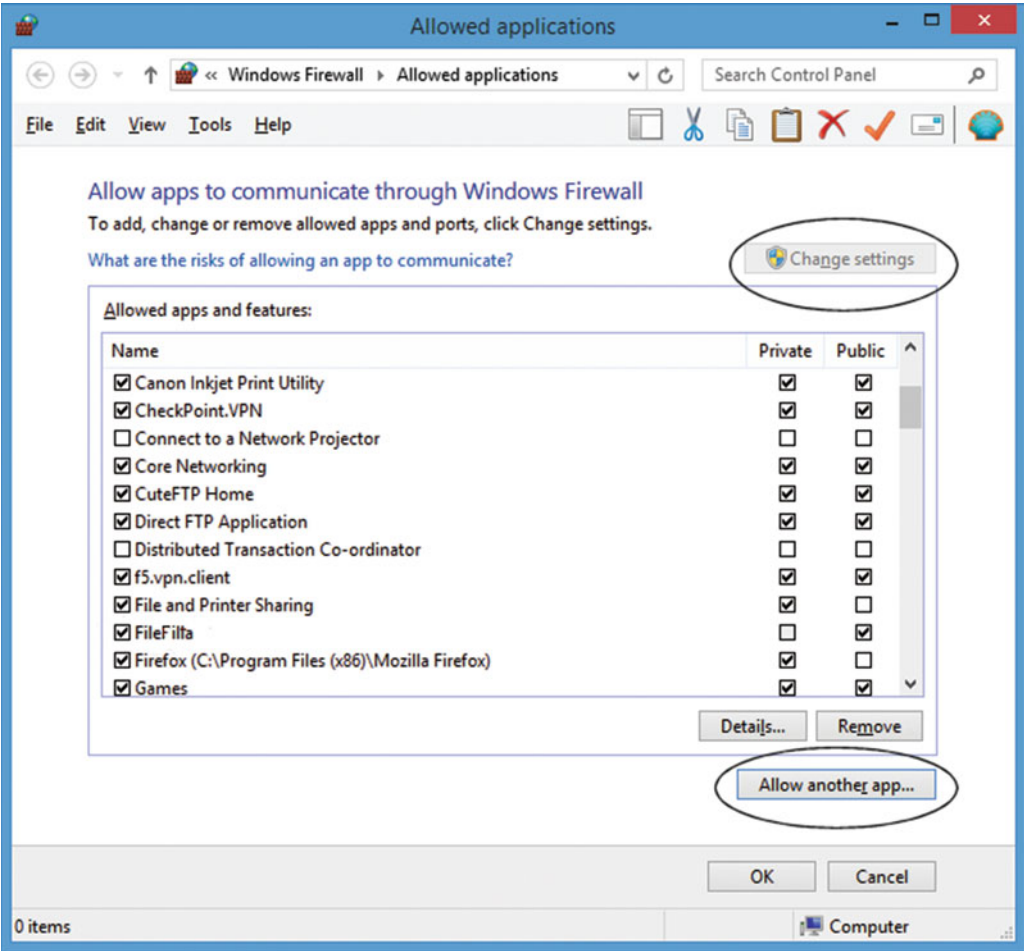


Figure 40-2. Click *Change settings*, then click *Allow another app*

4. This will activate the *Allow another app...* button near the bottom; click that button.
5. On the next pane, a new window will appear named *Add an app*. This displays a list of the programs on your computer. See Figure 40-3.

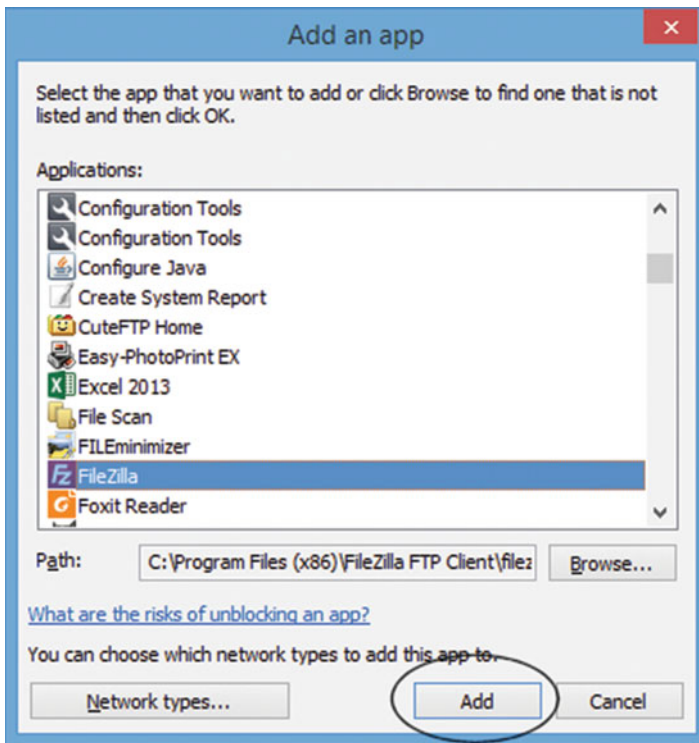


Figure 40-3. Add the FTP client

6. Scroll down to find your FTP client and click it. Then click the *Add* button.
7. The *Allowed applications* screen will reappear as shown in Figure 40-4.

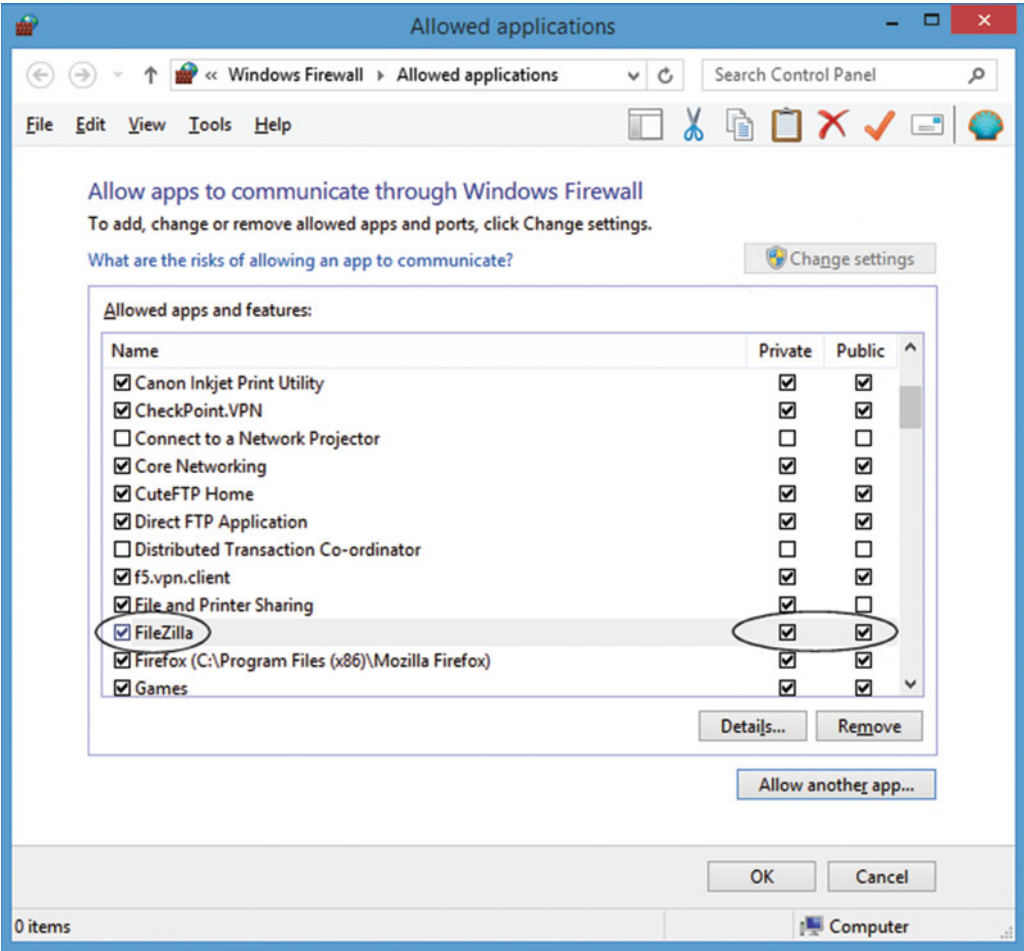


Figure 40-4. Check FileZilla and its private and public check boxes

- 8. Find the client that you just added and ensure that the *Private* and *Public* boxes are checked.
- 9. Click OK.

The great majority of PC owners use the Windows firewall, but some may install a different firewall. Lack of space prevents me from describing the configuration of these, but you should read your firewall's manual to discover how you can allow the FTP client to connect to host.

Configuring the Firewall in Windows 7 is a little different. The configuration steps are as follows:

- 1. Click on the *Start* menu (bottom left hand corner).
- 2. Click *Control Panel*.
- 3. Click *System and Security*.
- 4. Then under *Windows Firewall* click '*allow a program through Windows Firewall*'.
- 5. Now follow step 3 onwards in the Windows 8–10 configuration steps given earlier—see Figure 40-2.

Getting to Know Your FTP Client

Now that the firewall is configured you can begin to explore FileZilla. Open the program and you will see five panes and a Quickconnect bar at the top (shown ringed) in Figure 40-5.

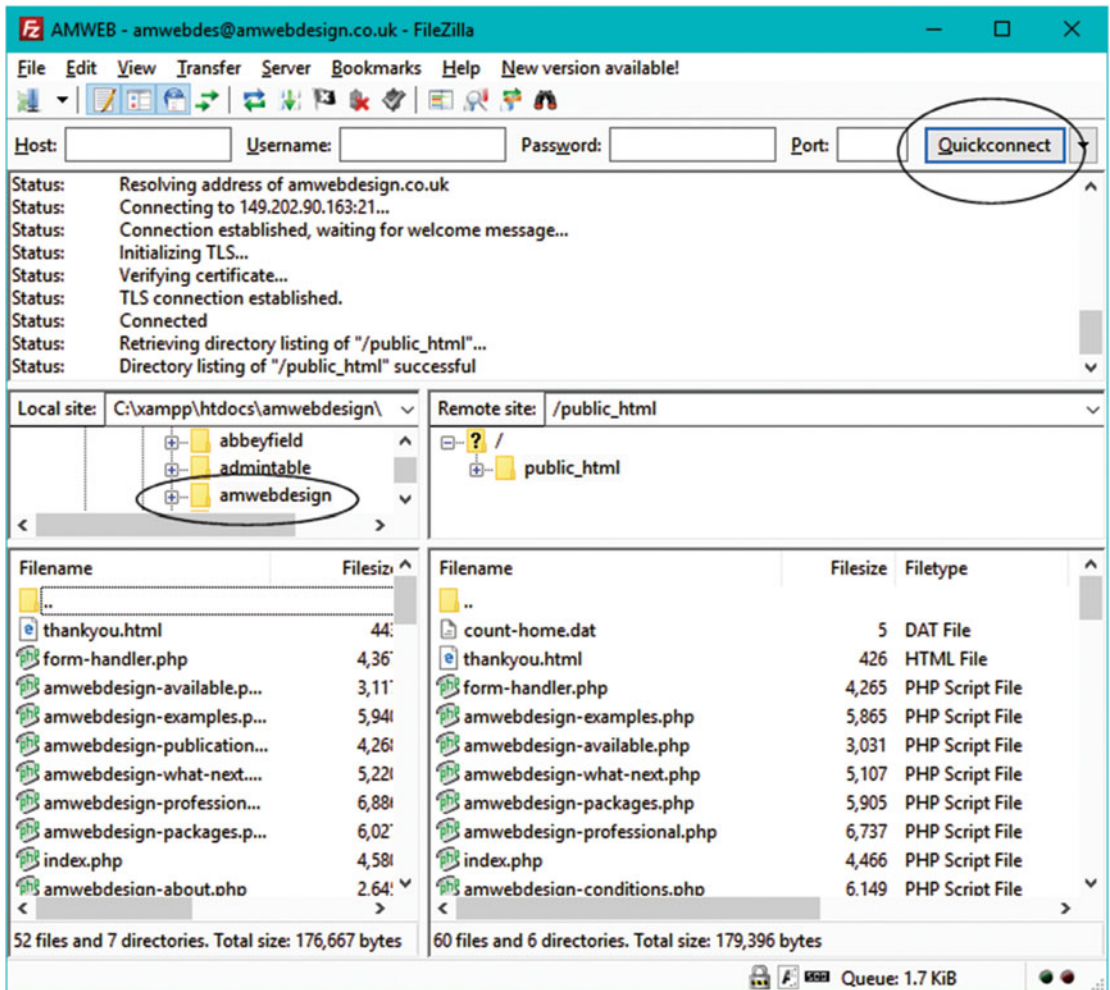


Figure 40-5. The main panes of the FileZilla interface

The full width pane just below the Quickconnect bar is the Log-display, and this describes the progress of a connection when you log on to a website at the host. If you meet a problem when trying to connect to a host, the log will let you know the cause of the problem.

The two panes below the Log-pane display folders. The left pane displays the folders on your computer. The right pane displays the folders on the host's server when you connect to a host.

The two large panes below the folder panes show files as well as folders. The left pane displays the content of a website folder on your computer. By clicking a web folder in the folder pane (shown circled), the folders and files within the web folder will appear below in the larger left pane. For instance, in Figure 40-5, the *amwebdesign* folder (circled) was clicked and its contents are shown in the larger left pane. The panes on the right will remain empty until a connection is made to a remote server.

When you connect to the host, the large pane on the right will display the files and folders within the remote server.

You will see another pane below the main panes; this shows the progress of an upload session.

Connect to the Host with Your Free FTP Client

The program is not particularly intuitive, so you will need the following instructions, however, once you understand the basic principles you will find it easy to use. The FTP client can connect to a website at the host's server by means of the Quickconnect bar as follows.

The Quickconnect Bar

Figure 40-6 shows the Quickconnect bar.

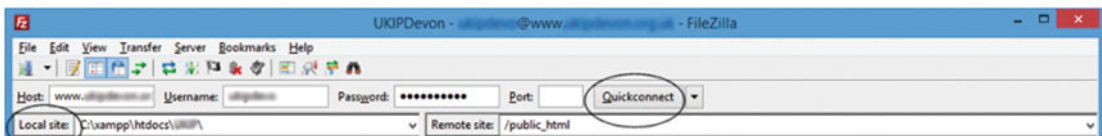


Figure 40-6. The Quickconnect bar

Enter the File Transfer Protocol (FTP) details provided by your host, that is, enter the host, the username, and the password. You can normally ignore the *Port* field because this is usually 21 and FileZilla uses this port by default. Click the *Quickconnect* button and in the right panes you should see some folders in the space you reserved in the host server. You may have to use trial and error in the host field; some hosts will ask you to enter the host using the format www.mydomain.co.uk, while others will require the host name in this format *ftp.mydomainname.co.uk*

You could also try this format: *mydomainname.co.uk*.

Obviously you won't want to enter this information every time you connect to a host, but fortunately FileZilla provides a way of storing the FTP details for each site.

Storing the FTP Details

FileZilla offers two methods for storing connection details, and one uses the Quickconnect bar as follows.

When you have entered the FTP details in the Quickconnect bar, if it has successfully connected to the remote computer (the host) you can take the following steps to store the FTP details:

1. Click *File* in the menu.
2. In the drop-down panel, click *Copy current connection to site manager*.
3. The login details will be copied into the file named **New site**.
4. Click the icon just below the word *File* to view the saved login details. Then click the *New site* to view the details in the *Site Manager*. See Figure 40-7.

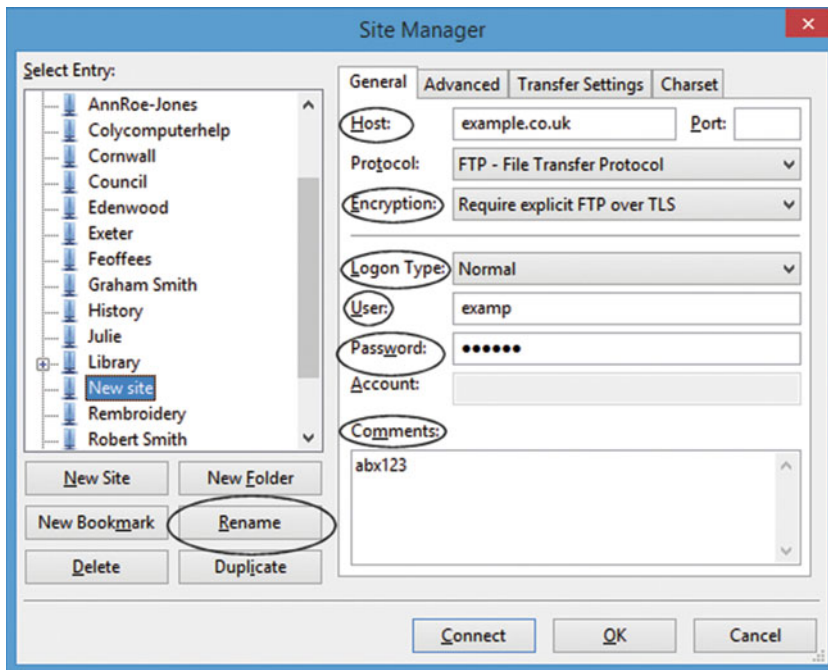


Figure 40-7. The login details for a fictitious website called *example.co.uk*

Note the folder in the left pane named *New site*, but you can change that to a more meaningful name later.

If you used step 2 in the above instructions to copy the quick connection to the site manager, the host will already be entered. Otherwise enter the Host name. The default protocol should not be changed. In the Encryption field select *Require Explicit FTP over TLS*. In the Logon Type field select *Normal*. The User and Password may already be present, however, I prefer to re-enter the User and Password just in case. The password, as usual is a row of asterisks, therefore it is rather easy to make a mistake.

To avoid this, I enter the password in the Comments box and double-check it against the details provided by the host. I then copy and paste from the Comments box into the password field. When you are satisfied, you can delete the password from the Comments box if you wish. Next, click the *Rename* button and enter an appropriate name for identifying the website.

The second method for storing the connection details uses the same dialog box (shown in Figure 40-7).

The dialog box appears when you click the first icon (far left) on the toolbar as shown circled in Figure 40-8. Click the *New site* button in the dialog box, give the new site an appropriate name, then fill in the connection details.

You can view a list of your stored websites and also connect to their hosts by clicking the first icon on the left of the toolbar shown circled in Figure 40-8.

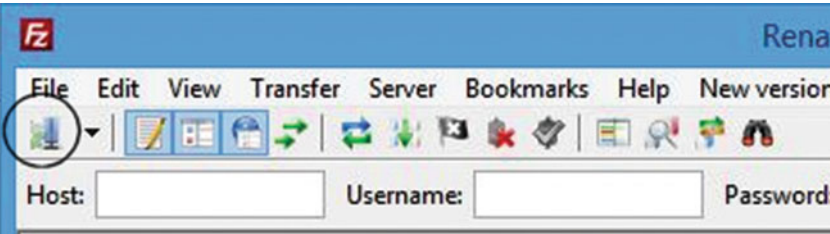


Figure 40-8. Click the icon shown circled for a list of your stored websites and their connection details

Using the Advanced Tab to Create Automatic Connections

Each time you log into a website you don't want the chore of having to navigate to the folder on your computer and then navigating to the folder on the host server. The *Advanced* tab enables you to open and display the appropriate folders the instant you log on. To complete the login details, click the *Advanced* tab and fill in the folder details as shown in Figure 40-9.

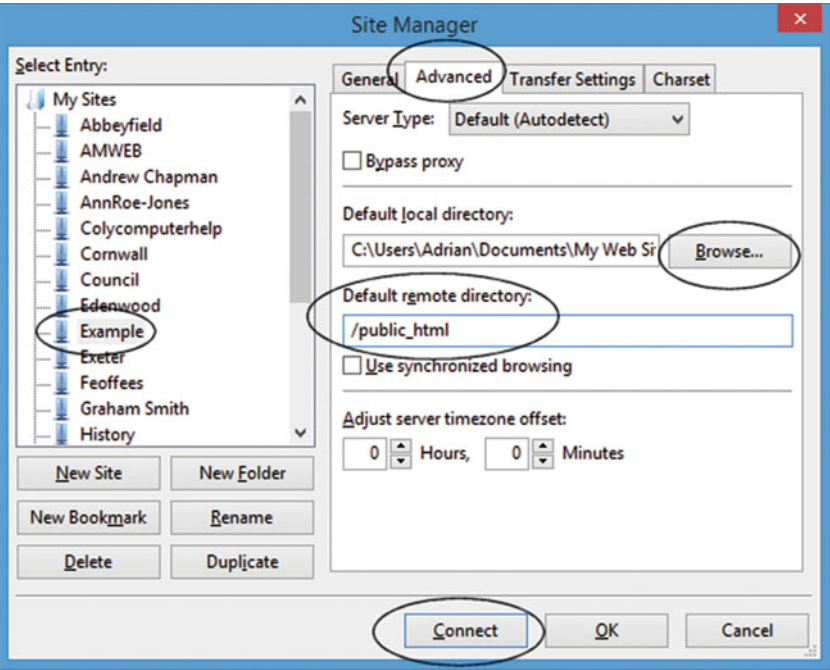


Figure 40-9. Storing folder details with the *Advanced* tab

Note the *New Site* has been renamed *Example* in the left pane by using the *Rename* button of the dialog box. To navigate to the local folder where your website is stored on your computer, click the *Browse* button. When you find the appropriate folder click it, and then click the *Select folder* button. Next enter the destination of the remote folder on the host's server. For the majority of hosts, you will be uploading folders and files to the folder named *public_html* as shown circled in Figure 40-9. Some hosts do not have a *public_html* folder, in which case you will upload items into the host's root folder, in which case, the field labeled *Default remote directory* must be empty. A few hosts will request that you enter something like this: [/www.example.co.uk/web](http://www.example.co.uk/web)

Always use the FTP details that the host sent to when you signed up.

Next, click the OK button then test your settings by clicking the *Connect* button.

Set the Date Format

The following steps will allow you to set the date format used by your territory.

1. On the FileZilla top menu click Edit, then Settings. Click Interface, then Date/time format.

You will now see the *Settings* dialog box as shown in Figure 40-10.

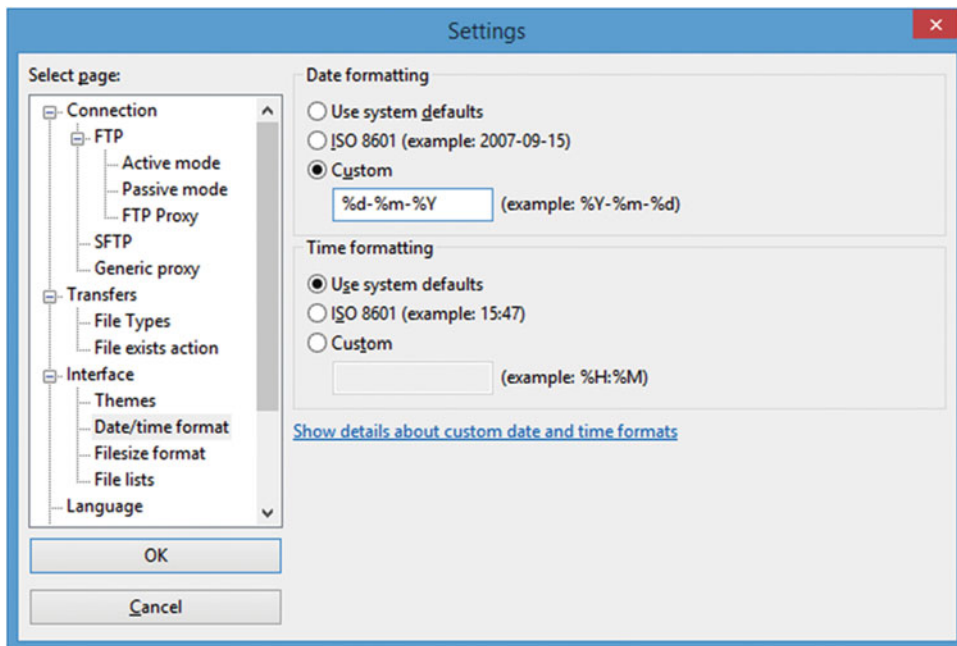


Figure 40-10. The Settings interface

2. The USA uses the default setting; otherwise select the Custom radio button then enter your territory's date format; for instance, in UK the date format is %d-%m-%Y.

3. Close down FileZilla and reopen it to see the reformatted dates.

4. Set the Recently Modified Files to Appear at the Top of the List.

The most convenient file date format puts the latest files at the top of the list.

To achieve this, see Figure 40-11.

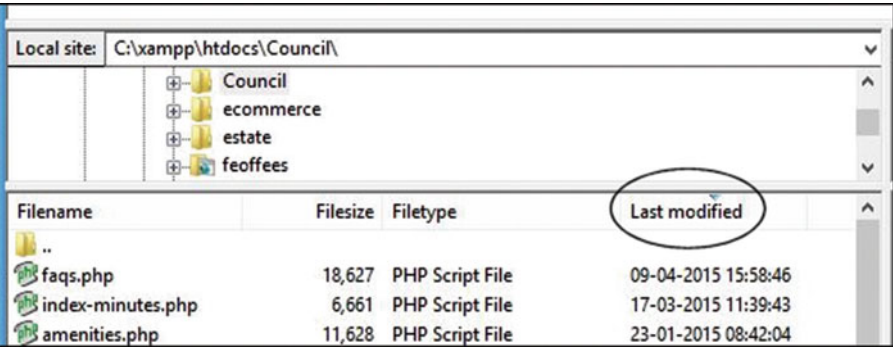


Figure 40-11. If the files are not listed in descending date order (latest at the top), click the tiny down-arrow above the word *Last modified* until the files are listed with the latest at the top in both panes

Uploading Folders and Files to the Remote Host

Open FileZilla, and click the first icon on the far left of the *Toolbar* to open the *File manager*. Select the appropriate website in the left panel of the file manager. Then click the *Connect* button and you will see the two main panes populated with the website's files and folders. In the left pane select the folders and files you wish to upload to the host. You can select multiple folders and files in the usual way by holding down the *Shift* key or the *Ctrl* key as you select. The selected items will turn blue.

To upload the item or the group of items, right-click one of the selected items and you will see the pop-up menu as shown in [Figure 40-12](#).

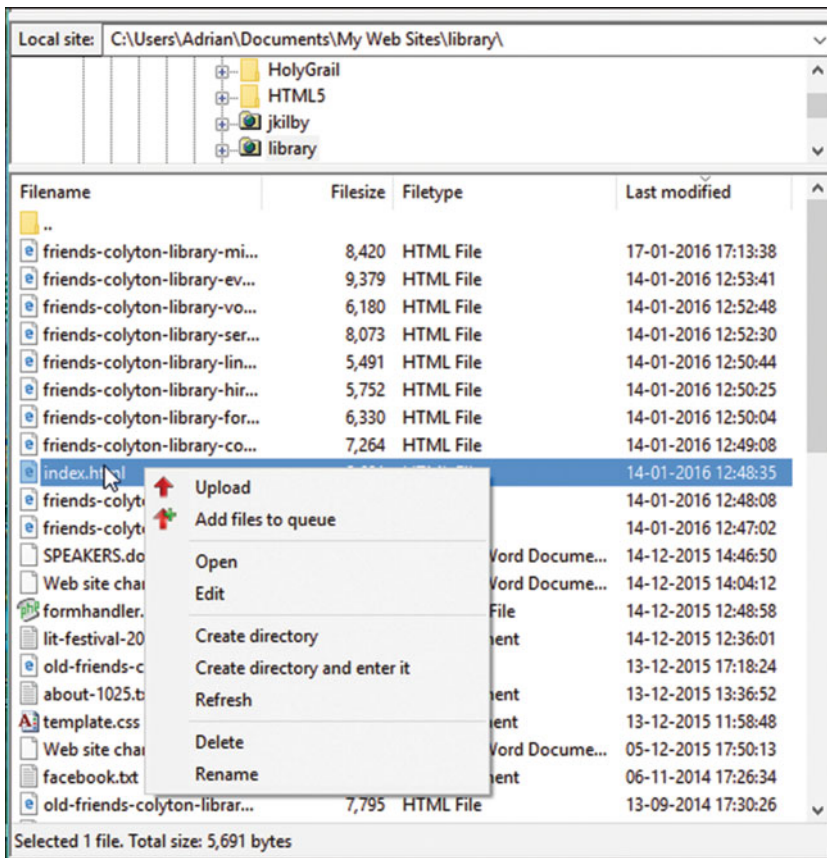


Figure 40-12. Uploading to the remote host

Left-click the *Upload* item on the pop-up menu, and a copy of all the selected items will be uploaded to the remote folder. Alternatively, you can double-click the file or folder to send a copy of the selected item(s) to the remote folder (the host). If you are replacing an existing file or folder at the host, you will see a warning; then you can then go ahead with the transfer or cancel it.

■ **Caution** Browsers will always choose to load the *index.html* file if it is present. If at an earlier date you uploaded a home page file named *index.html* and you later change the home page to a PHP file *index.php*, the PHP file must **replace** the *html* file in the host server. Right-click the *index.html* file in the right panel and rename the *index.html* file in the host as *old-index.html*, or delete it. Browsers will then be unable to find the *html* version of the home page and they will automatically load the *php* version.

Downloading Folders and Files from the Host to Your Computer

This is the reverse of uploading. In the right panel, right-click the folder or file that you wish to download and select *Download* on the pop-up menu.

Working within Folders

To move up a level in the folder hierarchy, double-click the folder shown circled in Figure 40-13.

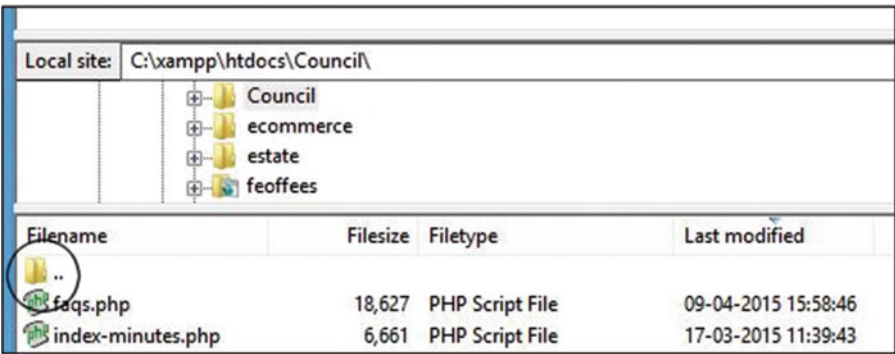


Figure 40-13. Double-click the folder shown circled to move up a level in the folder hierarchy

■ **Note** Subfolders such as the *images* or *includes* folders are always shown below the list of files in the FileZilla transfer panes.

Validate your Web Pages

Valid pages are important for the following reasons:

- Clients ask me to add a W3C validation logo to their website pages to indicate that they have employed a competent web designer.
- Validating is an excellent aid when troubleshooting obscure problems. The designer can save time and avoid frustration by validating before testing a page in various browsers.
- Validation ensures that your pages are correctly coded so that browsers do not display odd results.
- Validation ensures that search engines will not be blocked by your minor coding errors.
- Disabled users will be able to access your pages. Small coding errors can baffle an automated screen reader.
- Mobile devices have reduced versions of their browsers; they cannot cope with the errors in non-validated sites.

The DOCTYPE

To validate an HTML5 page, the DOCTYPE for an HTML5 the page must be set as follows:

```
<!doctype html>
<html>
<head>
<title>HTML5 test page</title>
<meta charset=utf-8>
</head>
```

Accessing the W3C Validator

The W3C validator can be accessed at <http://validator.w3.org>

Note that there is no letter “c” in this URL.

Using the W3C Validator

Figure 40-14 shows the interface of the W3C validator, which is accessed at <http://validator.w3.org>.

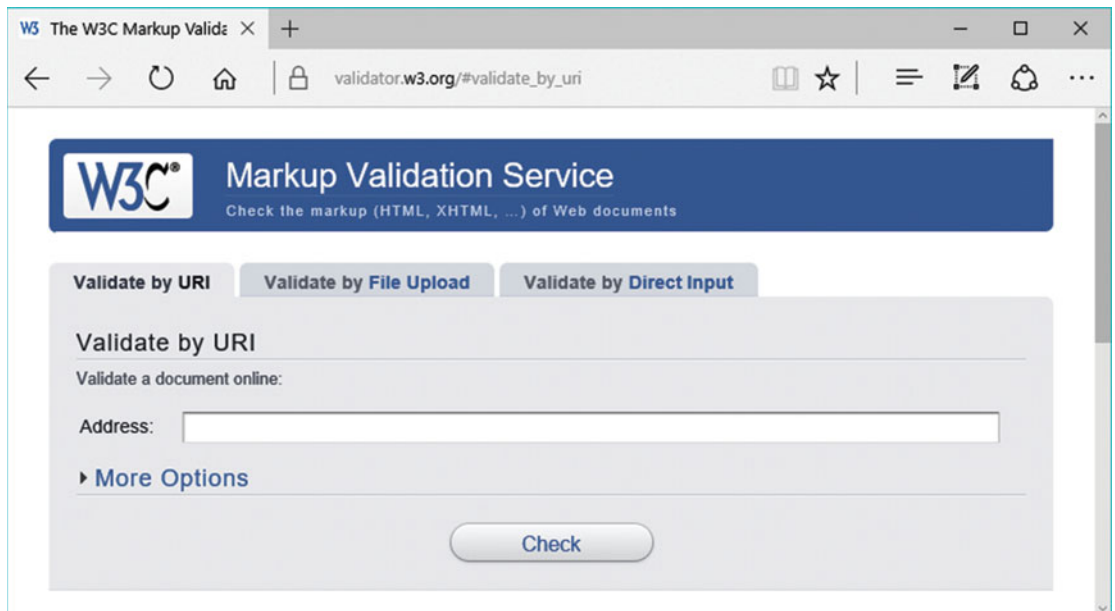


Figure 40-14. The W3C validator’s interface

Note the three tabs for entering the web page’s code in three ways. You can choose whether to validate by *URL* (i.e., the hosted web page’s address), *File Upload* (navigate to the file on your computer and click it), or by *Direct Input* (copy the code from your text editor and paste it into the text area on the validator interface.). The most useful feature is that you can validate a web page before you upload it to the host. Note that W3C.org uses the acronym URI instead of the more common URL. If your web page has been hosted, type or paste the page’s URL into the Address field in this format:

<http://www.mywebsite.co.uk/index.html>.

Click the *Check* button and wait for a few seconds. Eventually, a report will appear. Figure 40-15 shows the validator reporting one error and one warning.

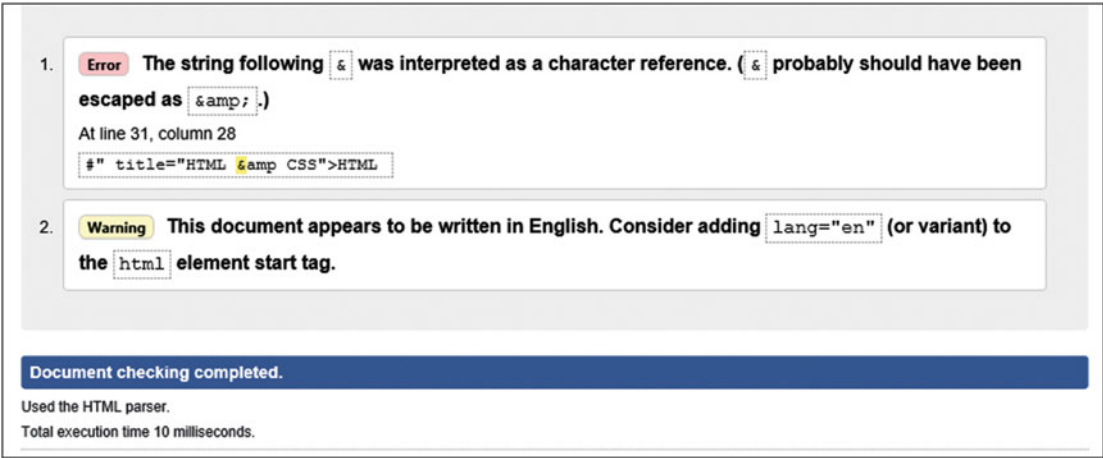
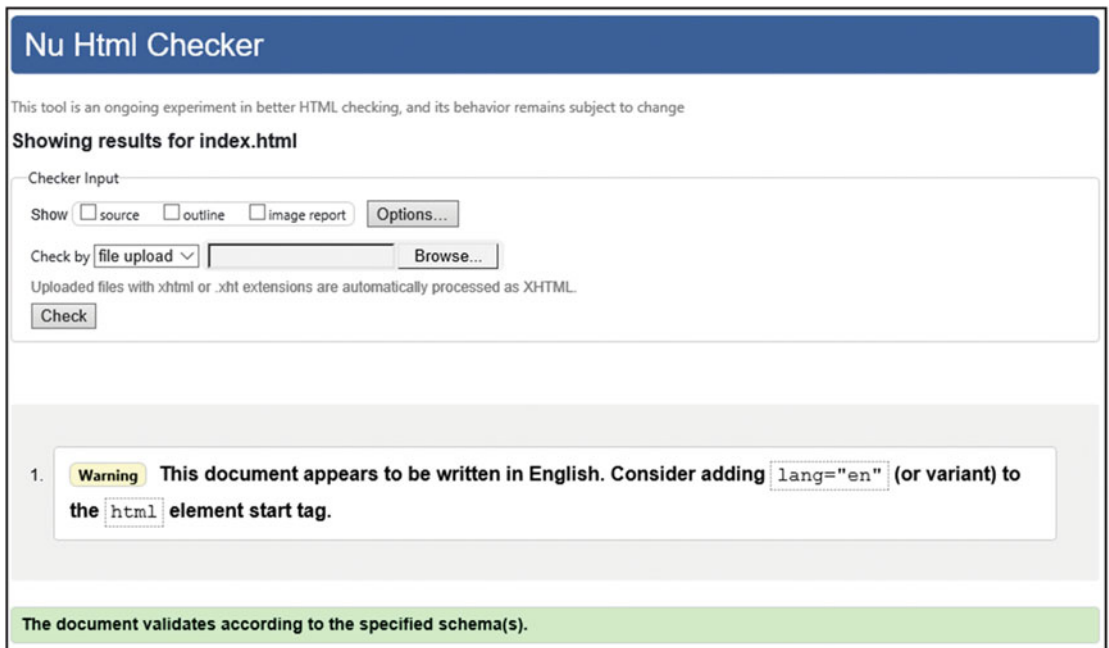


Figure 40-15. A validation report showing an error and a warning

The error was a semicolon missing from the end of the entity *&*, it should have been *&*. If the validation report gives one or two warnings, they can usually be ignored. Some warnings may refer to the fact that the validator is experimental or that utf-8 is assumed automatically. When the error was corrected the page was run through the validator once more, and the resulting report is shown in Figure 40-16.



Nu Html Checker

This tool is an ongoing experiment in better HTML checking, and its behavior remains subject to change

Showing results for index.html

Checker Input

Show ☐ source ☐ outline ☐ image report

Check by

Uploaded files with xhtml or .xht extensions are automatically processed as XHTML.

1. **Warning** This document appears to be written in English. Consider adding `lang="en"` (or variant) to the `html` element start tag.

The document validates according to the specified schema(s).

Figure 40-16. A validation report stating that the file contains no errors and one warning

Adding the semicolon to the entity `&` removed the error. The warning is still there but it is not very important. The report at the bottom of the screen shown in Figure 40-16 states that the page validated and it refers to *the specified Schema*, which is HTML5. The schema was specified by the DOCTYPE in the web page's first line of code.

■ **Tip** For maximum convenience, use a text editor that indicates the line numbers and print the page markup. Put the printout beside the computer so that you can refer to the line numbers provided by the validation report.

The HTML5 validator is only concerned with markup errors. Because HTML5 is backwardly compatible, it will accept most of the earlier tags and attributes, and the validator behaves accordingly. For instance, the HTML5 validator will accept `<meta charset=utf-8>`, `<META charset="utf-8">`, `<meta CHARSET=UTF-8 />`, or any combination of those styles. However, it will not validate `<meta charset=utf8>` because there is a missing hyphen between the *f* and the *8*.

■ **Caution** When validating the code in a page, the validation reports sometimes quote the wrong line number. This is normal; look for the error in a line near the one indicated, especially in earlier lines.

Some Typical Reports on HTML5 Errors and How to Fix Them

The HTML5 validator only looks for bad markup; it does not check code style.

If the header of the validator's report is red, the list of problems and suggested solutions will appear below the red header.

You may be alarmed to see a large number of errors listed in a report. Don't worry, the validator reports many errors when only a few exist; in reality many of the errors reported are duplicates.

The following sections describe some typical validation reports and the appropriate fixes.

A W3C Validation Report Found the Following Three Errors

The following is a snippet of the HTML5 markup that was submitted to the validator. I have added line numbers in parentheses for instructional purposes only.

```
(10) <!--[if lte IE 8]><!-- conditional Javascript added -->
(11) <script src="html5.js" type="text/javascript">
(12) </script>
(13) <![endif]-->
```

The first part of the validation report was as follows:

Line 10, Column 23: Consecutive hyphens did not terminate a comment. -- is not permitted inside a comment, but e.g. - - is permitted.

```
<!--[if lte IE 8]><!--conditional Javascript added-->
```

Line 10, Column 23: The document is not mappable to XML 1.0 due to two consecutive hyphens in a comment.

```
<!--[if lte IE 8]><!--conditional Javascript added-->
```

Line 13, Column 3: Bogus comment.

```
<![endif]-->
```

Line 13, Column 11: The document is not mappable to XML 1.0 due to two consecutive hyphens in a comment.

```
<![endif]-->
```

Line 13, Column 12: The document is not mappable to XML 1.0 due to a trailing hyphen in a comment.

```
<![endif]-->
```

Don't let the report baffle you, but start to clear away the warnings and some of the duplication as described in the following four steps.

■ **Tip** Copy the report to Notepad++ or some other text editor that displays line numbers. Then clear out the warnings. Delete the lines referring to XML because these are variations of already reported lines. This leaves a clearer field to work with (as shown next).

1. All warnings were removed from the copy of the report because they generally apply to the validator, not to the document.

2. I removed all references to XML because they duplicated the previous error messages. The following is what remained of the first section of the report:

```
Line 10, Column 23: Consecutive hyphens did not terminate a
comment. -- is not permitted inside a comment.
<!--[if lte IE 8]><!--conditional Javascript added-->
Line 13, Column 3: Bogus comment.
<![endif]-->
```

The report stated that consecutive hyphens are not permitted inside a comment. We know that consecutive hyphens in a comment are correct, so what was the problem? The validator sees the conditional block of code as a comment. This is correct because it begins with `<!--` and ends with `-->`. The report should have said you cannot have comments *within* comments.

3. So, I moved the comment outside and above the conditional comment and tried validating again. The amended markup is shown next.

```
<!-- conditional Javascript added -->
<!--[if lte IE 8]>
<script src="html5.js" type="text/javascript">
</script>
<![endif]-->
```

This validated successfully leaving only error number 3 to be corrected, as follows:

Line 38, Column 42: Bad value 88px for attribute width on element img: Expected a digit but saw p instead.

title="Valid CSS!" alt="Valid CSS!" />Syntax of non-negative integer: One or more digits (0-9). For example: 42 and 0 are valid, but -273 is not.

Looking at the markup, it is clear that a silly mistake had been made, as shown in bold type:

```
<p><a href="http://jigsaw.w3.org/css-validator/"></a>
<span class="small"><br><br>Markup Validated by the World Wide Web Consortium</span></p>
```

4. The last section of the report said it saw a letter “p” instead of a number. The 88px would be fine in a CSS style sheet, but it was bad markup when included in an HTML page. After removing the *px*, the whole page validated successfully.

Byte Order Mark Found

Microsoft Expression Web and Widows Notepad can creates markup that produces the mysterious warning, “byte order mark found.”

A byte order mark (BOM) is a sequence of bytes embedded in utf-16 pages. It makes sure that your utf-16 documents are read correctly by web browsers. Unfortunately, MS Expression Web, by default, adds a BOM to utf-8 pages or on PHP pages where a BOM is unwanted.

In MS Expression Web, to prevent a BOM being included in new pages, follow these steps:

1. Click *Tools* and select the *Page Editor Options* dialog box.
2. On the *Authoring* tab, look under New Documents.
3. Then under *Add a byte order mark (BOM)* to new utf-8 documents with these file extensions you will see a list of file extensions.
4. Clear the box next to each file extension that you would prefer not to have a BOM.

To remove a BOM from an existing page, do the following:
Open the document in Expression Web.

1. In Code view, right-click anywhere, and then click *Encoding*.
2. In the *Text File Encoding* dialog box, clear the box labeled “*Include a byte order mark (BOM)*.”
3. Save the page and upload it; then re-validate it.

■ **Note** The free Notepad ++ program has a BOM removal tool. Load the page, click *Format* on the menu, and select *Convert to UTF Without BOM*. Notepad ++ also provides line numbering, color coding, and many other great features for editing any type of web document.

Rare or Unregistered Character Encoding Detected

If you see that error message, the *meta* tag containing the text encoding has an error in it. The markup should read:

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

Most likely you omitted the hyphen in utf-8 and typed utf8.

Only a small number of validation errors are mentioned in this chapter because space is limited and there are 447 known errors. Visit the w3.org web site in the following tip to see some known error messages.

■ **Tip** For a list of 10 common validation errors and their solutions, try the websites at <http://line25.com/articles/10-common-validation-errors-and-how-to-fix-them> and <http://validator.w3.org/docs/errors.html>.

You may be alarmed to see a large number of errors reported. The validator finds an error at the top of the page, then it cascades down the report repeating the same error report many times, sometimes using different words. For instance, when validating a page that has errors, these errors may be repeated lower down in the report as XML parsing errors. Just ignore them and they will go away when you correct the errors nearest the top of the report.

Some of the Most Common Validation Error Messages

The following are some of the most common validation error messages:

OMITTAG NO: This precedes a common error. Look for elements that are not closed properly.

Line 78, Column 9: end tag for "html" omitted, but OMITTAG NO was not specified.

The end tag `</html>` for the opening `<html>` should appear on the last line of the page. I had omitted the end tag.

The `<p>` tag: The most common error is a `<p>` with a missing closing tag `</p>`.

Closing tags: The tags `
`, `<meta>` and `` are called self-closing tags. The report states closure errors like this:

Line 38, Column 10: end tag for "ul" omitted but OMITTAG NO was not specified.

That means that the closing tag `` or the tag's angle bracket `>` was omitted

When closure errors are reported, several errors will cascade through the report even though there is only one closure error. Fix the error and the cascade will vanish.

An element that is not open: The opposite of a closure error.

Line 87, Column 12: end tag for element "SPAN", which is not open.

This may mean that you removed an inline tag such as `` earlier in the sentence or document, but forgot to delete its closing element `` lower down in the page.

Incorrect nesting: Some tag errors may be described as being in a place that is not allowed. You may have inserted `` `` elements within table elements `<tr>` `<td>`. The error message will say something like this:

Line 33, Column 32: document type does not allow element "li" here.

Nesting with inline elements: In the next example, the validator is saying that block element tags like `` cannot be enclosed within a `` tag.

Line 52, Column 4: document type does not allow element "ul" here.

```
<span class="list">
<ul>
<li>The Haven provides sheltered housing companionship in
retirement</li>
<li>Thirteen purpose built units for single or double occupancy</li>
</ul>
</span>
```

The `` tags should be eliminated and the class should be inserted into the `` tag.

Other *not allowed here* errors are the result of surrounding a block element with an inline tag. For instance, surrounding an unordered list with `` tags, `` tags, or `` tags.

Deprecated tags and attributes: The error report will find deprecated elements that are no longer acceptable in HTML4 or HTML5; for example:

Line 32, Column 7: there is no attribute "CENTER".

This means that *center* is no longer acceptable in HTML markup; a style sheet must be used to center an element.

Examples of the W3C validator's more helpful explanations:

Line 137, Column 19: value of attribute "ALIGN" cannot be "ABSBOTTOM"; must be one of "TOP", "MIDDLE", "BOTTOM", "LEFT", "RIGHT". `align="absbottom" width="199" height="231"`

Some attributes such as `align` are no longer valid for lining up text with images; only top, middle, bottom, left, and right are acceptable.

Alts: The validator will report "alts" missing from images.

Some reports are not clear; for example:

Line 25, Column 39: document type does not allow element "li" here; missing one of "ul", "ol", "menu", "div" start-tag . `Home page`.

The web designer forgot to put a forward slash in the closing tag like this: ``.

Some misreporting occurs like the following example:

Elements not allowed here

Line 23, Column 9: document type does not allow element "h2" here; missing one of "object", "applet", "map", "iframe", "button", "ins", "del" start-tag . `<h2>some heading text was here</h2>`

I looked at line 23 and could not find the error; the `h2` element was properly closed. I then looked at nearby lines and found that on line 22, there was no closing `</p>` tag for a `<p>` element. Errors that are wrongly reported can usually be fixed by looking above and below the reported line for an unclosed `<p>`.

■ **Tip** You will have concluded that reported errors are not always what they seem. You will need to work out what the report should have said by looking at lines before or after the error line. Rather like cryptic clues in a crossword puzzle, with practice you will recognize the patterns and know how to correct coding errors.

Validate the CSS

Go to <http://jigsaw.w3.org/css-validator/>, and there you will find an interface very similar to the HTML validator.

- **If your CSS page has been uploaded to your web host**, type in the URL of the CSS style sheet you want to validate using this format: <http://www.mywebsite.co.uk/mystylesheet.css>.
- Click the Check button and wait for a few seconds. Eventually, a report will appear.
- **If the CSS page is not yet uploaded to your web host**, click the File Upload tab, browse to the file for the page on your computer, and load it. Then click the Check button.

Or click the Validate by Direct Input tab and paste the CSS page markup in the field provided. Then click the Check button.

Figure 40-17 shows a result of a successful validation.

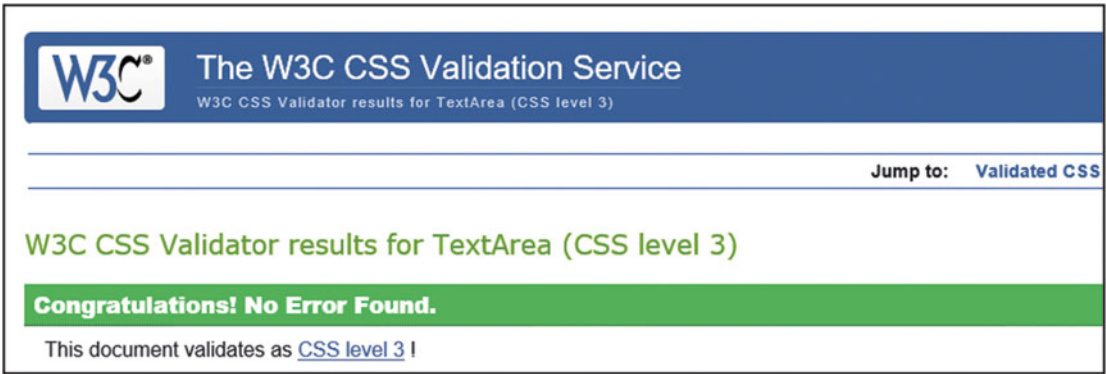


Figure 40-17. The W3C validator for CSS showing a successful validation

If the CSS markup fails to validate, you will be given hints about the errors. The following validation warning about the CSS code for a menu's 3D buttons can be puzzling:

29	li.btn a	Same color for background-color and border-left-color
30	li.btn a:hover	Same color for background-color and border-top-color
30	li.btn a:hover	Same color for background-color and border-left-color
30	li.btn a:hover	Same color for background-color and border-right-color
32	li.btn a:active	Same color for background-color and border-top-color
32	li.btn a:active	Same color for background-color and border-left-color
32	li.btn a:active	Same color for background-color and border-right-color

The outset border color on the menu's 3D buttons is the same as the background color; this works fine in most cases, but the CSS validator thinks it is not so good. These warnings can be safely ignored, but if you wish, the error can be eliminated by a tiny change to the outset border color to make it slightly different from the background color.

Here is another example of a failed CSS validation:

Sorry! We found the following errors (1)

```
URI : TextArea
2 td Parse Error ;0 5px }
```

Warnings (1)

```
URI : TextArea
```

5 You have no color set (or color is set to transparent) but you have set a background-color. Make sure that cascading of colors keeps the text reasonably legible.

I examined the CSS code that was submitted to the validator, and found the error and the warning, as shown in bold text in the following:

```
(1) table {width:500px; border:1px black solid; border-collapse:collapse;}
(2) td {border:1px black solid; padding:0 5px;0 5px}
(3) th {border:1px black solid;}
(4) caption {font-weight:bold; }
(5) table tr {background-color:#C8F0F0;}
(6) .right {text-align:right;}
```

In line 2, the semicolon after the 5px should have appeared at the end of the line as follows:

```
td {border:1px black solid; padding:0 5px 0 5px;}

```

The warning about the fifth line recommends that a text color should always be specified, as well as a background color. The line should have been:

```
table tr {background-color:#C8F0F0; color:black;}

```

Vendor-Specific CSS Errors

The CSS validator will report errors for any vendor-specific items in the style sheet; that is, items such as `-mozkit-` and `-webkit-`. This is what you would expect because the vendor-specific items are not W3C recommended; the validator's role is to check for conformance with W3C recommendations.

Embed the HTML5 Logo

Figure 40-18 shows the HTML5 logo that was released in January 2011.



Figure 40-18. The HTML5 validation logo

Currently the HTML5 logo does not indicate that W3C is the authority providing the validation. You will need to state this on your page.

The HTML5 logo is available in several sizes and several configurations. The W3C logo web site is pretty but confusing. It is found at <http://www.w3.org/html/logo/>

■ **Note** There is no letter “c” in the W3C URL; it is w3.org not w3c.org. The logo is licensed under creative Commons Attribution 3.0 Unported. You may change its color and size.

The Solution for a Verifiable HTML5 Logo

At the time of writing, the HTML5 logo did not have this safeguard; a webmaster could cheat and embed the HTML5 logo to pretend that the page was valid.

Meanwhile, I have devised some code to overcome this deficiency. The code can either eliminate the need to dynamically embed the HTML5 logo, or you can call it from your images folder. Enter the code shown in bold in Listing 40-1 at an appropriate place on the web page; you do not need my permission to use it.

■ **Note** A page must be uploaded to the host to enable my logo code to produce a verification report when clicked. If you do not, an error message will tell you that the validator was unable to locate the referrer.

Listing 40-1. Inserting a verifiable Logo on a Web Page.

```
<p>
<a href="http://validator.w3.org/check?uri=referer">
</a>
</p>
```

If you would prefer to load the image from an images folder in your root folder instead of from the W3C web site, download the 64-pixel logo into your images folder, and then use the following alternative code snippet:

```
<p>
<a href="http://validator.w3.org/check?uri=referer">
</a>
</p>
```

If you prefer a smaller logo, change every instance of 64 to 32 in the code snippets. You will need to download the 32-pixel version of the logo to enable the second code snippet to show the small logo.

Summary

In this chapter you were given advice on choosing a domain name and host for your website. Instructions were provided for downloading, installing, and using a free FTP client. You discovered how useful validation is for troubleshooting and for ensuring that your code will function properly on all types of devices. The chapter explained that the reports received from the validator can be rather cryptic and that some practice is necessary in order to become familiar with their meanings and the solutions. A code snippet was provided to embed an HTML5 logo that enables users to check that the page has been genuinely validated.

In the next chapter you will find a quick reference, a list of resources, and some advice on installing and using graphics programs.

CHAPTER 41



Quick Reference: Graphics Programs: Resources

This chapter is for reference only and contains no projects. Its sections are as follows:

- A quick reference for HTML, CSS, and PHP
- Graphics programs for optimizing pictures
- Resources
- Summary

A Quick Reference for HTML, CSS, and PHP

This section will save you having to trawl through the book to find that particular piece of basic HTML, CSS, or PHP code. It also refers to the relevant chapters when a more complex piece of HTML and CSS is required for a particular feature.

Web Page Structure

Every page on every website has a similar structure; this can be summarized as follows:

Doctype

Head section

Body section

Wrapper

Header or banner

Content

Footer

Wrapper finishes here

Body finishes here

HTML Tags

The structure is specified by means of HTML tags; these are English words, or abbreviations of English words, enclosed by angle brackets.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Some file name</title>
    <meta charset=utf-8>
  </head>
  <body>
    <div id="wrapper">
      <header>
      </header>
      <div id="content">
      </div>
      <footer>
      </footer>
    </div>
  </body>
</html>
```

The **Doctype** tag is `<!DOCTYPE html>` or `<!doctype html>` tells the browser that the code in the page is HTML5. The Doctype is normally followed by the tag `<html>` like this:

```
<!DOCTYPE html>
<html>
```

The **head** section is enclosed in the tags `<head></head>` and the content of the head section gives the browser important information as follows:

```
<head>
  <title>The Spring Garden Club order form</title>
  <meta charset="utf-8">
  <link href="style-3d-gdn.css" rel="stylesheet" type="text/css">
</head>
```

The content of the title is important for search engine optimization. The *charset* tells the browser which code set is used in the web page.

The head section can also contain *meta* details such as a *meta description* and *meta keywords* for search optimization, for example:

```
<meta name="description" content=" Spring garden supplies, spring bulbs, spring plants and
garden supplies, bulb and seed suppliers, garden supplies, garden supplies online.">
```

```
<meta name="keywords" content=" Spring garden supplies, garden supplies, garden supplies
online, plants and garden supplies, bulb and seed suppliers">
```


The *link* tag instructs the browser to load a particular style sheet and has the following format:

```
<link href="style-3d-gdn.css" rel="stylesheet" type="text/css">
```

Supplementary links add extra styling to the main style and they have the following format:

```
<link href="style-3d-gdn.css" rel="stylesheet" type="text/css">
<link href="form.css" rel="stylesheet" type="text/css">
```

href stands for hyperlink reference.

If your HTML <head> section contains one or more linked styles (and perhaps an internal style, plus a piece of Java script) they should appear in a particular order as follows:

- Link to main style sheet
- Link to supplementary style sheet
- Internal style
- Script

■ **Note** The majority of tags have closing tags, that is, the tag word is preceded by a forward slash, for example: </head>. The *meta*, *img*, *br*, and *link* tags are exceptions; they have no closing tag.

Comments

Comments are purely for the benefit of the web designer and are ignored by browsers.

Comments have the following format:

```
<!--My comment-->
```

The Main Structural Tags

The body tag <body></body> relates to the whole computer screen; for example, the CSS code could instruct the browser to fill the entire screen with a particular color.

The Wrapper surrounds the content of the page and is defined like this:

```
<div id="wrapper"></div>
```

The tag **<div** stands for a division or section of the web page, the **id="wrapper"** identifies the division so that it can be styled. The wrapper controls the boundaries of the main content and it can provide a border around the main content. Using CSS it can be given a fixed width or a maximum and minimum width to cope with various screen sizes.

The header: The header or banner usually contains a picture and some text.

The Content holds the main information that will be displayed on the user's screen; it is defined as follows:

```
<div id="content"></div>
```

ID versus class: You can use a particular id, for example, `<div id="content">` only once on a page. You can use the same class, for example, `class="btn,"` as many times as you like on a page.

The content area can contain many elements such as menus, columns, text, and pictures.

Menus: The code for a basic menu with plain colored buttons might be as follows:

```
<nav>
  <ul>
    <p>Menu</p>
    <li><a class="btn" href=page-2.html>Go to Page 2</a></li>
    <li><a class="btn" href=page-3.html>Go to Page 3</a></li>
    <li><a class="btn" href=index.html>Home Page</a></li>
  </ul>
</nav>
```

The menu's hyperlinks: such as `Home Page` allow the user to move from page to page in a website.

The `<a...>` tag stands for anchor, which is rather cryptic; it would have been easier to understand if the word “access” had been used. For clarity this book assumed that `<a` stands for the word “access.”

`href` stands for hyperlink reference, that is, it refers to the file to be accessed.

Therefore, the piece of code ` Home Page` displays a clickable hyperlink that will access the *index.html* file (which is the home page).

Note that the words *Home Page* are located between the opening tag `` and the closing tag ``. Any text between those tags is displayed on the screen.

The line break code `
` moves the next element down one line. Note that `
` tag does not need a closing tag.

The styling for a rollover menu with plain color buttons might have the following code:

```
/*set the styles for the side menu column*/
nav ul {float:left; width:150px; margin-left:10px; list-style-type:none;}

/*set the general side button styles*/
li.btn {width:115px; line-height:20px; margin-bottom:3px; text-align:center;}

/*set general access (anchor) styles*/
li.btn a {display:block; width:115px; color: white; background:gray; ↵
font-family:arial; font-size:small; font-weight:bold; text-decoration:none;}

/*mouseover */
li.btn a:hover {background: blue;}

/*mousedown (optional)*/
li.btn a:active {background:green;}
```

The style **list-style-type:none;** prevents bullets displaying, and the style **text-decoration:none;** removes underlines from the links.

For 3D menu buttons, refer to Chapter 8, and for horizontal menus refer to Chapter 12. For horizontal and vertical menus on the same page, refer to Chapter 14.

Background Color Styles for Rollover 3D Menu Buttons

Plain color buttons can be changed to 3D buttons by adding extra border code to the *mouseout* and *mouseover* settings. By changing the mouseout and mouseover styles in any of this book's 3D templates you will be able to produce a menu with buttons in a different color; some examples are given below:

Red

```
/*mouseout style*/
li.btn a {background: #D20B0D; border: 4px outset #FFAAAA; ↵
font-family:arial; font-size:100%; font-weight:bold; text-decoration:none;}
/*mouseover */
li.btn a:hover {background: maroon; border: 4px outset maroon;}
```

Light green

```
/*mouseoutstyle*/
li.btn a { background:#759885; border:4px outset #A2D0A2; ↵
font-family:arial; font-size:small; font-weight:bold; text-decoration:none;}
/*mouseover */
li.btn a:hover { background:olive; border:4px outset #a2d0a2;}
```

Pale blue

```
/* mouseout style */
li.btn a {background:#1A9CE0; border:4px outset #AABAFF;} ↵
font-family:arial; font-size:small; font-weight:bold; text-decoration:none;}
/* mouseover */
li.btn a:hover {background:#0A4ADF; border:4px outset #8ABAFF;}
```

Brown

```
/*mouseoutstyle*/
li.btn a {background: #946055; color:white; border: 4px outset #c96e6b; ↵
font-family:arial; font-size:100%; font-weight:bold; text-decoration:none;}
/*mouseover */
li.btn a:hover {background: #9F7562; color:white; border: 4px outset #C96E6B;}
```

Purple

```
/*mouseout*/
li.btn a {background:#702f84; color:white; border: 4px outset #904FA4; ↵
font-family:arial; font-size:100%; font-weight:bold; text-decoration:none;}
/*mouseover*/
li.btn a:hover {background:olive; border: 4px outset olive;}
```

Gold

```
/*mouseout*/
li.btn a {background:#BFf9002; color: white; border: 5px outset #f5D8C5; ↵
font-family:arial; font-size:100%; font-weight:bold; text-decoration:none;}
/*mouseover*/
li.btn a:hover {background:#A87848; color:navy; border: 5px outset #F5D8C5;}
```

Columns

The HTML code for a three-column layout is given in the following snippet of code:

```
<div id="content">
<h2>Home Page</h2>
  <div id="leftcol">
    <p>This is the far left column</p>
    <!--verticle menu goes here-->
  </div>
  <div id="rightcol">
    <p>This is the far right column</p>
  </div>
  <div id="midcol">
    <h2>Home Page</h2>
    <p>This is the middle column</p>
  </div>
</div><!--content div finishes here-->
<br class="clear">
```

The columns will not be evident until they are styled. The next snippet of code gives the CSS for styling the columns.

```
#leftcol {float:left; width:150px;}
#rightcol {float:right; width:150px;}
#midcol {margin-left:155px; margin-right:155px;}
```

The resulting layout of the columns is shown in Figure 41-1.

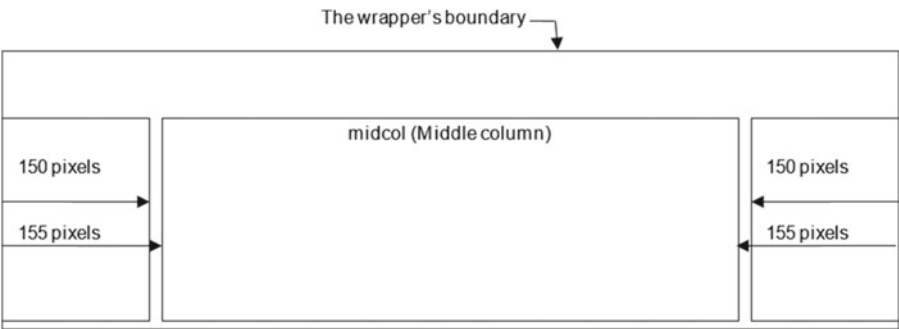


Figure 41-1. The 155 pixel margins set the middle column 5 pixels clear of the outer columns

A four-column page requires two additional columns set within the middle column. These we will name as midcol-left and midcol-right.

```
<div id="content">
  <div id="leftcol">
    <p>This is the far left column</p>
    <!--verticle menu goes here-->
  </div>
```

```

<div id="rightcol">
<p>This is the far right column</p>
</div>
  <div id="midcol">
    <h2>Home Page</h2>
    <p>This is the content</p>
    <div id="midcol-left">
      <p>This is the midcol-left</p>
      <p>Some text in the midcol-left</p>
    </div>
    <div id="midcol-right">
      <p>This is the midcol-right</p>
      <p>Some text in the midcol-right</p>
    </div>
  </div><!--the midcol div finishes here-->
</div><!--the content div finishes here-->
<br class="clear">

```

The CSS style for the four-column page will have two extra styles for the two new columns, something like this:

```

#midcol-left {float:left; width:48%;}
#midcol-right {float:right; width:48%;}

```

Elements within the Content Area

The other elements within the content area are as follows:

Heading text. Headings have tags ranging from the most prominent <h1> to the least prominent <h6>.

Example:

```
<h2>Home Page</h2>
```

Paragraphs have <p></p> tags as shown in the following example:

```
<p>This is the far left column</p>
```

Pictures are included by using the
```

**Tables** are drawn with the following typical code:

```

<table>
 <caption>Travel Methods Available</caption>
 <tr>
 <th>Destination</th>
 <th>Coach</th>
 <th>Ferry and coach</th>
 <th>Air</th>
 </tr>

```

```

 <tr>
 <td>Paris</td>
 <td>No</td>
 <td>Yes</td>
 <td>Yes</td>
 </tr>
 <tr>
 <td>Madrid</td>
 <td>No</td>
 <td>Yes</td>
 <td>Yes</td>
 </tr>
 </table>

```

The corresponding CSS is given in the next snippet of code:

```

table {width: 400px; margin:auto; border-collapse:collapse; ←
border:1px black solid;background:white;}
td, th {padding:2px; border-collapse: collapse; border:1px black solid;}
th {font-weight:bold;}
caption {font-size:150%; font-weight:bold;}

```

## An Element Below the Content Area

The **footer** is located below the content area but within the wrapper area as follows:

```

</div><!--content div finishes here-->
 <br class="clear">
 <footer>
 <p>This is the footer</p>
 </footer>
</div><!--the wrapper div finishes here -->
</body>

```

The line `<br class="clear">` pushes the footer clear of the floated columns in the wrapper and content.

Its associated CSS for the class is defined as follows: **.br {clear:both;}**

## Styling the Text with HTML

No CSS styling is required for the following two items:

**Emphasis.** The tag for bold text is `<p><strong>I am bold</strong></p>`

The tag to produce italic text is `<p><em>I am italic</em></p>`

**Character entities.** Some characters require entities so that they display correctly. `&amp;` will display the ampersand as `&`. The British pound must be `&pound;` to display as £. A medium-length hyphen is `&ndash;` and the long hyphen is `&mdash;` The copyright symbol is `&copy;`

**<span>**: This tag is used when a certain word or words are required to look different from the surrounding text. In the following example the `<span>` tag causes the words *be aware* to be displayed in a bold red font:

```
<p>Before confirming your order, be aware that sales tax ↵
will be added.</p>
```

In the main style sheet, the corresponding CSS will be `.red {color:red; font-weight:bold;}`  
Classes can be grouped as follows:

```
<h2 class="red center">Some text with two classes</p>
```

The two classes will cause the text to be red and centered, and there must be a space and no comma between the two classes. The associated CSS would be:

```
.red {color:red;}
.center {text-align:center;}
```

The next section describes the basic formats of the most common style sheet code.

## CSS Styles Sheets

The HTML tags are styled by CSS using the word or abbreviation inside the tags, for example, the `<body>` is styled by the CSS word `body`. A paragraph `<p>` is styled by the CSS word `p`. The only exception is any *id* such as `<div id="wrapper">`, which would be styled by the CSS word `#wrapper`

CSS styling instructions are contained within curly brackets like this: `p {font-size:110%;}`

The most used CSS styles are as follows:

**Borders:** `Element {border:thickness color type;}`

Example: `p {border:1px black solid;}`

**Margins:** `Element {margin:width;}`

Example: `p {margin:10px;}`

**Padding:** `Element {padding:width}`

Example: `#wrapper {padding:10px;}`

**Colors:** `Element {text color}` or `element {background-color}`

Example: `p {color:blue;}` or `p {color:#0000FF;}`

Example: `body {background-color:aqua;}` or `body {background-color:#00FFFF;}`

**Bullets:** See Chapter 31.

**Grouping selectors:** Elements requiring a common style can be grouped using separating commas, for example:

```
header,nav,article,section,footer {display:block;}
```

Elements within nested code can be targeted by using a group without commas, for example:

```
nav ul li {display:inline-block;}
```

This style targets the list items nested within the unordered listing, which is nested within the `<nav>` tags.

## PHP

The following PHP *include* code generates the menu code in a web page. It does this by including an external file named *nav.html*. In this example the *nav.html* code is located within a folder named *includes*.

```
<?php include ('includes/nav.html'); ?>
```

When a web page contains some PHP code, the suffix of the page's file name must be changed from *.html* to *.php*; for example, the page *index.html* must be changed to *index.php*.

## Graphics Programs for Optimizing Pictures

Pictures for inclusion in a website must be optimized so that they load quickly. Pictures downloaded straight from a camera or provided by a client are usually enormous. The files that can be compressed are *.jpg*, *.png*, and *.gif*. To optimize a picture, follow these steps.

Use a graphics program and follow these steps:

1. Crop the picture, that is, remove any unnecessary parts around the edges of the picture.
2. Resize the picture to fit the area reserved for it in the web page.
3. Compress the picture file.
4. Sharpen the picture if necessary.

## Precautions when Downloading Free Programs

When you download a graphics program you may be asked whether to *Run* or *Save* the downloaded file; choose *Save* and save it in your *Downloads* folder. Then double-click the file to install it. The reason for choosing *Save* is because you may one day buy a new computer, and you would then be able to transfer the program's installation file to your new computer. Some programs are tricky to install from saved files, for instance, it is easier to use *Run* instead of *Save* when installing IrfanView.

---

■ **Warning** When downloading any program, take your time and watch carefully for any pre-checked boxes. Uncheck them; otherwise you will download programs such as toolbars or search bars. Malware linked to these unwanted items has become more common. Downloaded programs from once reliable sources can no longer be trusted. Sometimes they have more than one download button: one is safe and the others are bad. Try to download from the software producer's own website rather than from mirror websites. Also if you are offered a customized installation, choose it. Don't choose the option labeled "Express install (Recommended)." Most important, take your time, and read every installation screen carefully and look for those sneaky pre-checked boxes.

---

At the time of writing, the most notorious mirror sites to be avoided are *Cnet's Download.com*, *Tucows*, *FossHob*, and *FireForum*. When there is more than one 'Download Now' button on a website, how can you know which button is safe? The best current solution is to add a browser extension named *AdBlock*, because this actually prevents the dangerous button from being displayed on the pages.

You can read about *AdBlock* by visiting <https://getadblock.com>

You must even be wary of any pre-checked boxes when installing the *AdBlock* download.



## Which Graphics Program?

If you already have one of the paid-for graphics programs, you will have access to plenty of help; so search the Internet using the *name* of the program together with keywords such as *optimize images* and *resizing images*. You will find similar help for free graphics programs such as GIMP that, in my opinion, is every bit as good as the commercial programs.

The following two smaller graphics programs are free and very good for optimizing images.

## File Minimizer Pictures

<http://www.balesio.com/corporate/eng/products.php>

Scroll to the bottom of the page and select the free version.

The interface is attractive and user friendly. I tried it to compress a 6.5Mb picture and this resulted in a file size of 610kb, which is a 90% reduction. The picture quality was impressive for such a large compression. The program is free for private use but you will need to pay for a license to use it commercially.

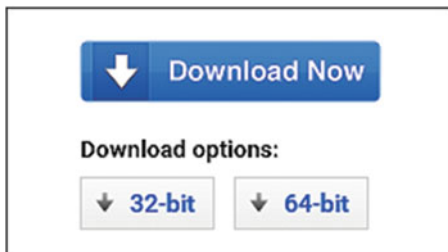
Unfortunately, File Minimizer is not able to crop or resize pictures. To prepare pictures for the web you would first need to crop and resize them using *Windows Paint* or *Paint.net*. Cropping is easier with *Windows Paint*, because after selecting an area of the picture you can adjust it. When the picture is cropped and resized, you then will be able to compress it using File Minimizer.

## IrfanView

IrfanView is able to crop, resize, and compress a picture; it can also process multiple pictures.

To download IrfanView, avoid downloading large quantities of unwanted rubbish by using the following websites. At the time of writing the following download sources were safe:

For the 32-bit version go to <http://www.techspot.com/downloads/299-irfanview.html>. Even if your computer is 64-bit, don't choose the 64-bit version, because I had extreme difficulty installing the plug-ins for the 64-bit version. Figure 41-2 shows the 32-bit button that you should click:



**Figure 41-2.** Click the 32-bit button and wait for the download bar to appear at the bottom of the screen. It will ask you if you wish to run or save the IrfanView setup file; choose *Run*

When it has finished installing, access <http://www.techspot.com/downloads/299-irfanview.html> again and scroll down to the pale blue link with the words *download IrfanView plugins*. Choose the 32-bit plug-in and wait for it to download. You must not mix plug-ins; the 32-bit IrfanView program needs the 32-bit plug-ins.

When the download bar appears at the bottom of the screen select *Run* to install it. Do not select *Save*.

**To open a picture.** Click *File* on the top menu, then click *Open*; navigate to the location of the picture, then click the picture.

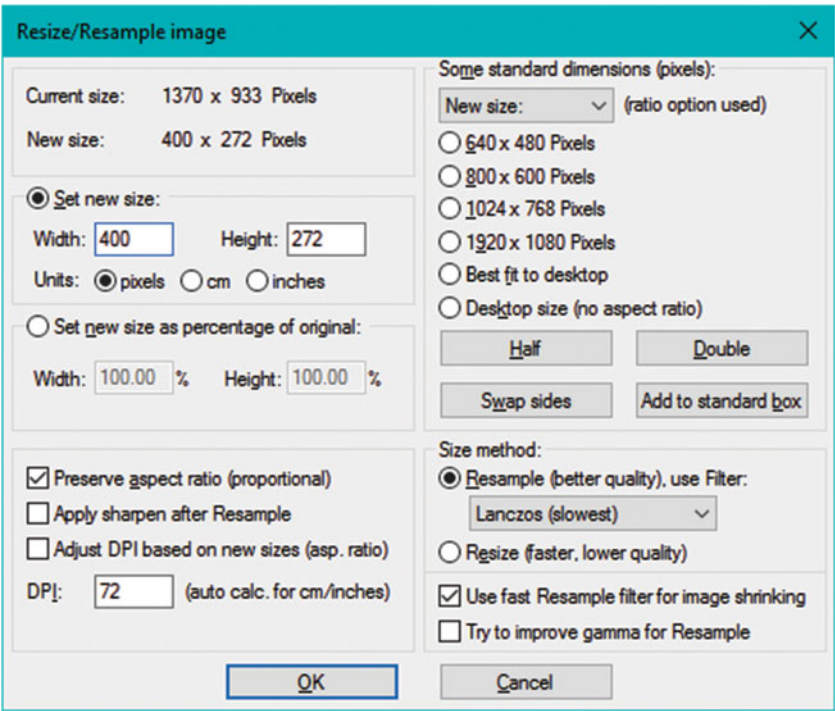
**To crop the picture (if required).** With the left mouse button, click in the picture at the top left, and with the left button held down, move diagonally to the bottom right (this called a *selection*). You then will be able to drag the borders of your selection to give the exact required area. Then click *Edit* on the menu and select *Crop to selection*.

**Resizing, that is, changing the dimensions of the picture.** When adding a picture to a website you may require specific dimensions. For example, this might be when you want a thumbnail or perhaps a gallery of the same size pictures.

With your picture open in IrfanView, click *Image* on the top menu, then select *Resize/resample*.

In the *Resize* dialog, ensure that the radio button *Set New Size* is selected and that the check box *Preserve Aspect Ratio* is checked. Enter one of the required dimensions – either the width or the height.

Now click *OK* to resize the image to the new dimensions. Figure 41-3 shows the *resizing* dialog box with some typically selected features.



**Figure 41-3.** The *resizing* dialog box. Note in the top-left corner the comparison of sizes before and after resizing

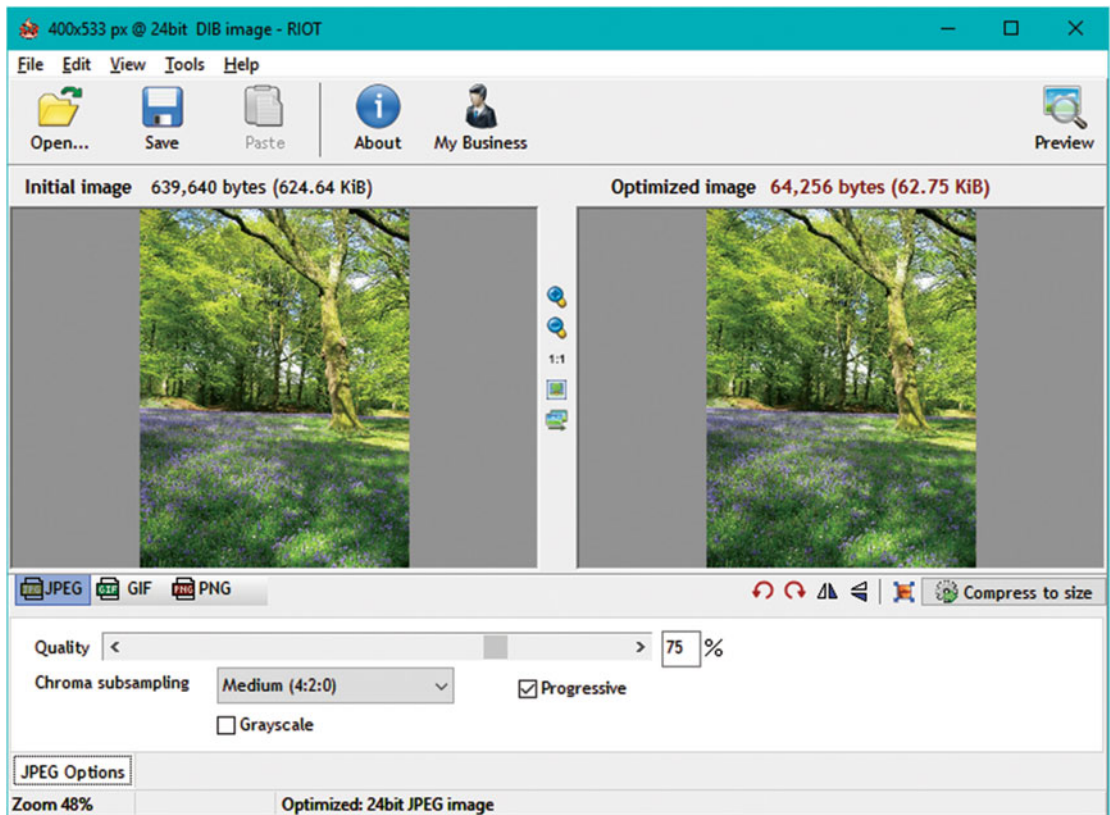
Because the aspect ratio was preserved, it may be that your resized picture will not have the height or width that you wanted. You can change the canvas size of the resized picture by adding margins.

Click *Image* on the top menu and select *Change Canvas Size*. Select the amount you wish to add to the top, bottom, left side, and right side of the picture. Lower down on the dialog box, select a color for the canvas, click *OK*, then click *OK* again.

You now have a resized picture; its file will need to be compressed to optimize it for the web.

**To compress a resized picture.** To reduce the file size of your resized picture (and leave its width and height unchanged), click *File* then click *Open*. Navigate to the location on your computer where your resized picture is saved. Click the picture to select it, and then click the *Open* button to put the picture on the IrfanView screen.

On the top menu, click *File* then click *Save for Web...(Plugin)*  
 A new panel will appear with two pictures as shown in Figure 41-4.



**Figure 41-4.** The compression dialog box

In the right pane you will see the default compressed version (75% compression), and in the left pane you will see your original picture. Above each of the pictures you will see the file sizes before and after optimizing. Click which image format you want – jpeg, gif, or png.

Towards the bottom of the dialog box you will see a slider where you can adjust the quality of your compressed picture. Note that the picture quality deteriorates as you increase the compression.

On the right of the dialog box, towards the bottom, there is a section *Compress to size* – here you can set file size in kilobytes. This is useful when you want to aim for a particular file size.

You can use either the quality slider or you can enter a new percentage in the size compression box to compress your image.

Once you are happy with your new compressed image, click the *Save* button and save it with a suitable name to the *images* folder in your website folder. I like to add *comp* to the file name when saving compressed files: for example, *bluebells-comp.jpg*.

## Resources

This section provides a few resources that will help you to expand your knowledge of website design by further reading and online tutorials.

### **Irfan Tutorials**

<http://www.techabrel.com/2014/12/how-to-compress-single-or-multiple-images-using-irfanview.html>

[https://www.montgomeryschoolsmd.org/departments/hiat/tech\\_quick\\_guides/irfanview\\_batch\\_convert.pdf](https://www.montgomeryschoolsmd.org/departments/hiat/tech_quick_guides/irfanview_batch_convert.pdf)

### **Responsive Web Design**

*Responsive Web Design with HTML5 and CSS3*. Ben Frain. PACKT Publishing. Published 2012.

ISBN 978-1-8469-318-9

*Beginning Responsive Web Design with HTML5 and CSS*. Johnathan Fielding. Friends of Ed/Apress.com. Published 2014.

ISBN 978-1-4302-6694-5

### **HTML and CSS**

*Practical HTML5 Projects*. Adrian W. West. Apress.com. Published 2012.

ISBN 978-1-4302-4275-8

*CSS: The Missing Manual 2nd edition*. David Sawyer McFarland. O'Reilly. Published 2009.

*CSS3: The Missing Manual 3rd edition*. David Sawyer McFarland (this book includes CSS2, CSS3, and many HTML5 tips). O'Reilly. Published 2013.

ISBN: 987-1-449-32594-7

### **PHP**

*PHP for the Web 4th edition*. Larry Ullman. A Visual QuickStart Guide. Peachpit Press. Published 2011.

ISBN: 978-0-321-73345-0

*Practical PHP and MySQL Website Databases: A Simplified Approach*. Adrian W. West. Zpress.com. Published 2013.

ISBN: 978-1-4302-6076-9

### **Graphics**

*Beginning Gimp 2nd Edition*. Akkana Peck. Apress.com. Published December 2008.

ISBN: 978-1-4302-1070-2

### **HTML Text Editors**

*Microsoft Expression Web 4 - Step by Step*. Chris Leeds. Microsoft Press/O'Reilly Media. Published 2010.

ISBN: 978-0-7356-3902-7

### **Browser Statistics**

Statistics for browser usage are monitored by the following websites:

<http://marketshare.hitslink.com/browser-market-share.aspx?qprid=3>

[http://www.w3schools.com/browsers/browsers\\_stats.asp](http://www.w3schools.com/browsers/browsers_stats.asp)

<http://gs.statcounter.com/>

<http://www.netmarketshare.com/browser-market-share.aspx?qprid=0&qpcustomid=0>

[http://en.wikipedia.org/wiki/Usage\\_share\\_of\\_web\\_browsers](http://en.wikipedia.org/wiki/Usage_share_of_web_browsers)

## Summary

This chapter provided a quick reference for HTML5 and CSS2/CSS3. Some free graphics programs were described to help you optimize pictures for the web. The final section contained resources that you may find useful.

The next chapter gives instructions for installing and using HTML text editors.

## CHAPTER 42



# Installing and Using Text Editors

An HTML text editor is essential for producing customized fast loading websites. Chapter 1 provided a list of free text editors, and it stated that you should refer to Chapter 42 for instructions on downloading, installing and using a text editor.

This chapter contains the following sections:

- Which free WYSIWYG program is best?
- Download, install, and configure MS Expression Web 4 (free)
- Using MS Expression Web 4
- Downloading and installing Blue Griffon (free)
- Using Blue Griffon
- Non-WYSIWYG text editors
- Summary

## Which Free WYSIWYG Program Is Best?

Both Expression Web and Blue Griffon are excellent, but each has its pros and cons. Expression Web has been abandoned by Microsoft; this means it will never be updated to keep pace with developments in HTML and browsers. However, it has more useful features than you will ever need. Blue Griffon is actively developing to keep pace with the latest web technology. It has fewer features than Expression Web, but it is more than adequate for producing websites and new features are constantly being added. Blue Griffon has a more accurate page display in WYSIWYG view than Expression Web. Blue Griffon is produced by a French organization named disruptive-innovations.com, and the program is a successor to programs such as NVU and KompoZer.

The fact that Expression Web 4 will never be updated may be an advantage because it will never change in the future, and so there will be no need to learn where Microsoft has hidden familiar items. Blue Griffon does add improvements with each new version. Blue Griffon released a new version causing me to panic when I was writing this chapter. This required a complete rewrite and all the screenshots had to be renewed. Despite this, I like both programs equally, and I use both to access the best features of both and the best option for the future. However, you will only need one for the projects in this book.

---

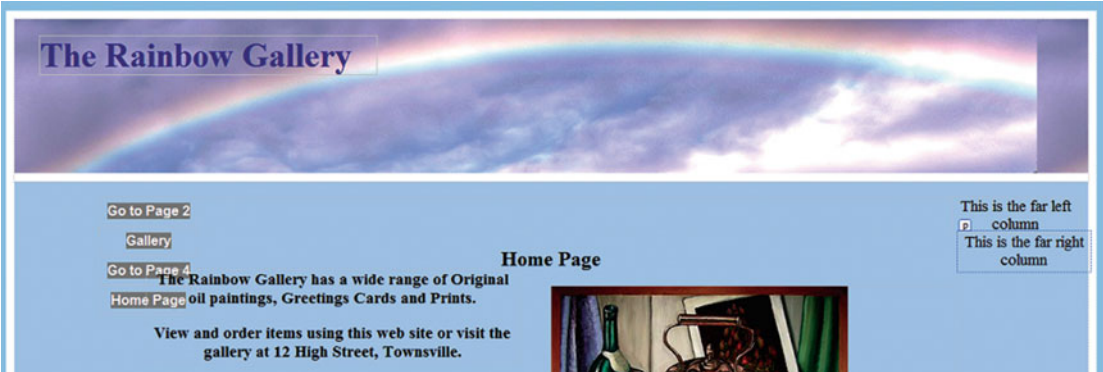
**Note** If you are using an Apple Mac computer, Expression Web 4 will not download. You will need to use Blue Griffon if you want to install a WYSIWYG editor on a Mac computer.

---

# The WYSIWYG View in Expression Web May Look Odd

Sometimes in the Expression Web 4 Design view, a web page may not look the way we intended it to. Figure 42-1 illustrates this.

Don't panic! This is normal and it won't affect the appearance when the website goes live. Also Expression Web has a simple solution to this problem; see the section "Selecting Alternative Browsers for the Design View of MS Expression Web 4." Compare Figure 42-1 with the top half of the same page displayed in Blue Griffon as shown in Figure 42-2.



**Figure 42-1.** The top of the page displayed in Expression Web 4; the horizontal menu is not as we intended



**Figure 42-2.** The top half of the same page displayed in Blue Griffon

As you can see, Blue Griffon does a better job of its WYSIWYG view. This is because Blue Griffon has strong ties with the latest version of Mozilla Firefox, whereas the Design view in Expression Web is associated with an outdated version of Internet Explorer. The section "Selecting Alternative Browsers for the Design View of MS Expression Web 4" describes a simple way of making Expression Web display an accurate page view.



Blue Griffon also has a problem that I have been unable to solve so far. After working with Blue Griffon for a while, dozens of tabs accumulate on the interface. These cause the program to load slowly and they are confusing. Hopefully an answer will be found soon.

## Be Cautious When Downloading Text Editors

When you download a program you may be asked whether to *Run* or *Save* the downloaded file, choose *Save* and save it in your *Downloads* folder. Then double-click the file to install it. The reason for choosing *Save* is because you may one day buy a new computer, and then you would then be able to transfer the program's installation file to your new computer.

---

■ **Warning** When downloading any program, take your time and watch carefully for any pre-checked boxes. Uncheck them; otherwise you will download programs such as toolbars or search bars. Malware linked to these unwanted items has become more common. Downloaded programs from once reliable sources can no longer be trusted. Sometimes they have more than one download button: one is safe and the others are bad. Try to download from the software producer's own website rather than from mirror websites. Also if you are offered a customized installation, choose it.

---

If you see more than one 'Download Now' button on a website, how can you know which button is safe? The best current solution is to add a browser extension named *AdBlock*; this actually prevents the dangerous button from being displayed on the pages.

You can read about *AdBlock* by visiting <https://getadblock.com>

You must even be wary of any pre-checked boxes when installing the *AdBlock* download.

## Download, Install, and Configure MS Expression Web 4 (Free)

To download Expression Web 4 use this website:

<http://www.microsoft.com/en-us/download/details.aspx?id=36179>

Download the program and select *Save*. When you start to download the editor, you will be told that you need to download Microsoft .NET Framework 4 (Web Installer); note that you **should** download this installer.

When you accept the installer you may be asked if you would like to download some other bits and pieces; **decline the offer** and continue with the download of Expression Web 4. The downloaded file will be named something like *Web\_Trial\_en(1).exe*, but don't be alarmed; it is no longer a trial version but a free version. Then save it in your Downloads folder and double-click the file to install it.

When the Expression Web 4 appears on the screen, click *Properties* to put a shortcut icon on the desktop. Double-click the shortcut (desktop top) icon and the HTML editor will load (slowly because it is a big program).

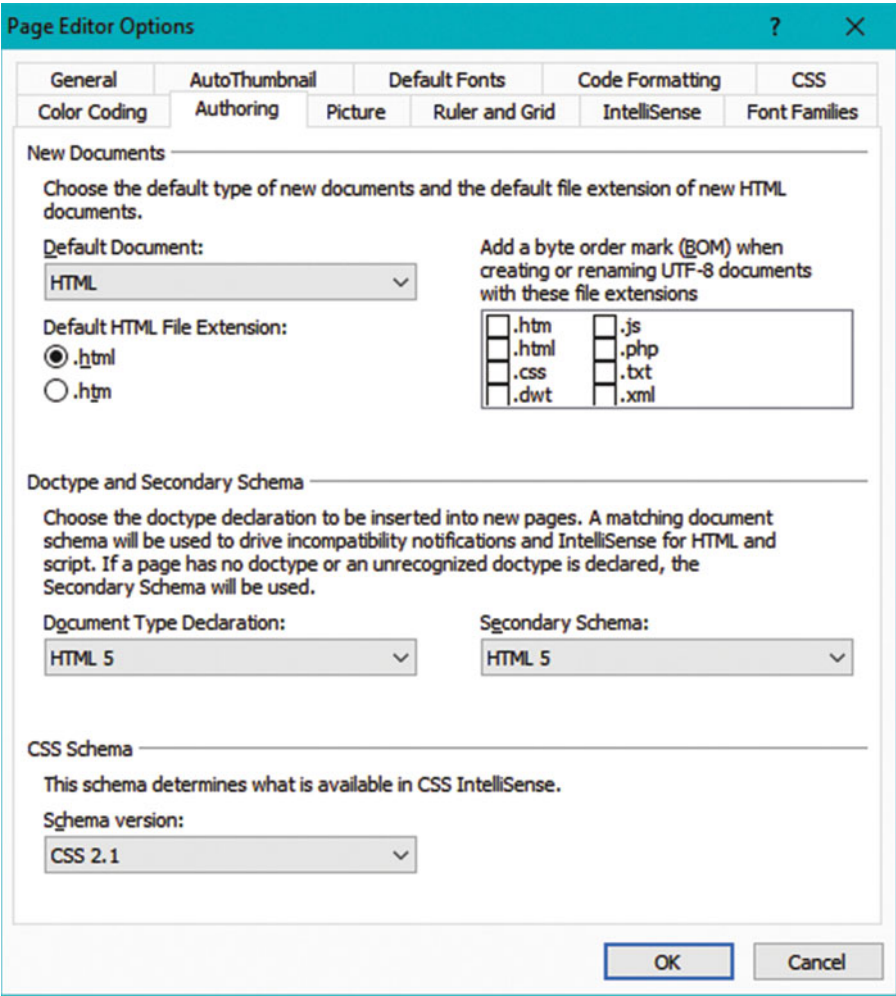
Click **Tools** then click **Page Editor Options** (the last item on the drop-down menu) and then click the following tabs:

**General tab:** Ensure that the spelling is UK or USA English.

**Default fonts tab:** The language character set should be Unicode UTF-8.

**CSS tab:** Select *Manual Style Application*.

**Authoring tab:** Set the authoring details as shown in Figure 42-3.



**Figure 42-3.** Choose your Authoring preferences

Ensure that all the check boxes relating to the various file extensions are all un-checked.

If you click the *Secondary Schema's* down-arrow you will see that the list is out of date and it will never be updated; likewise the CSS Schema version also has an out-of-date list. Fortunately this is no problem because you will always be creating code from scratch or using copy and paste; you will not be relying on an automated process.

Close that window and click *Tools* then select *Application Options*.

In the next window, click the box next to *Use your current Windows color scheme*.

That will change the sinister black theme to your default Windows theme.

## Configure the Toolbars in Expression Web

The Toolbar icons are similar to those in MS Word. The Undo feature is very important for web developers. It allows you to go back many stages to the stage when the page was good (before you messed it up). See Figure 42-4.





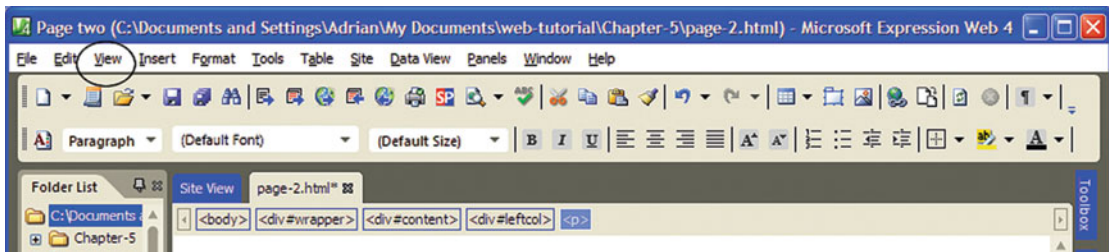
**Figure 42-4.** The Undo icon is on the left. The Redo symbol is on the right

Rest your cursor on each of the icons on the Toolbar to see the pop-up tips; many will be familiar but some relate specifically to website design.

In Design view, make sure the Toolbars for "Page" and "Quick tag selector" are displayed.

To do this, click **View** (shown circled) in the next figure, then ensure that those two Toolbars are checked.

You should then see two Toolbars as shown in Figure 42-5.



**Figure 42-5.** The Toolbar in Expression Web 4

To add the folder list, click *View* (shown circled in Figure 42-5) and on the drop-down list click *Folder List*. The folder list pane will appear on the left. In the future, whenever you access a new website or a new page on a website, that folder or web page will be accessible from the folder list.

## Selecting Alternative Browsers for the Design View of MS Expression Web 4

You will need to add your preferred browser(s) for viewing pages in the *Design* view of Expression Web.

Provided you configured the program to use various browsers, you can see how a web page looks in a modern browser simply by pressing the F12 key. If you wish to view the page in several browsers or in a particular browser, click *File* ► *Preview in browser*. A list of browsers will fly out; click the appropriate browser and the viewport size.

In the *Design* view, you will be able to preview web pages in multiple web browsers. To add new browsers to the list please follow these steps:

1. In the top menu click *File*, then click *Preview in Browser*.
2. Click *Edit Browser List*.

The browser list in my own computer is shown in Figure 42-6.

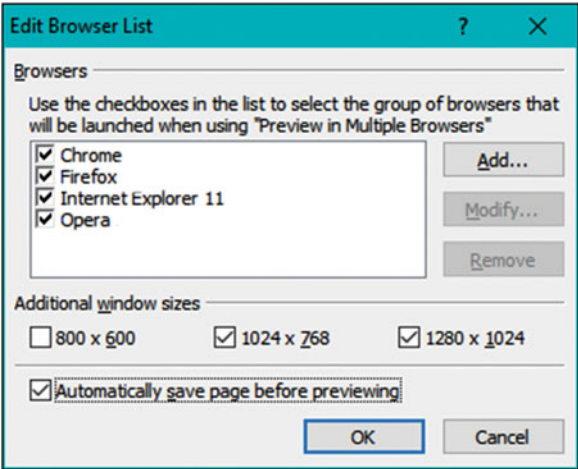


Figure 42-6. Edit the browser list

In the *Edit Browser List* panel you can change which browsers will be launched to view your web page in multiple browsers; select additional viewport sizes, and add a browser.

I decided to add Safari for Windows to the list; click the *Add* button to add a browser. You will see the dialog box shown in Figure 42-7. Enter a name for the browser and use the *Browse* button to explore the folder *Program Files (x86)*, find the *.exe* file for the browser, and select it. Figure 42-7 shows the result of selecting the *safari.exe* file.

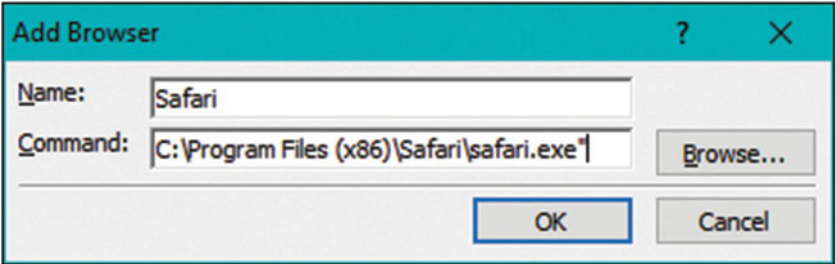
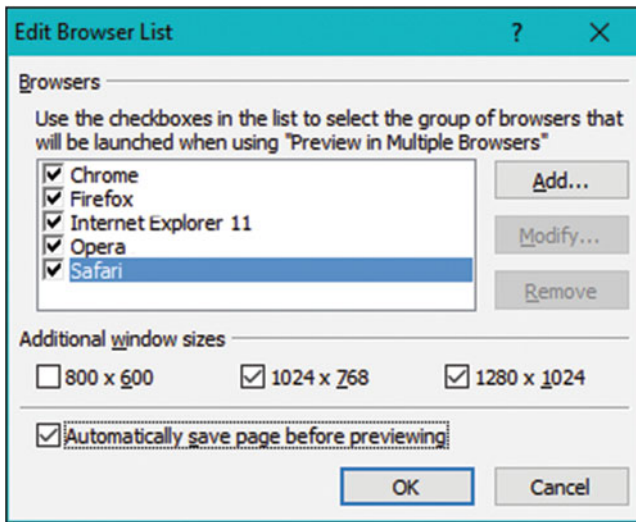


Figure 42-7. The completed “Add Browser” dialog box

Figure 42-8 shows that Safari for Windows has been added to the list.



**Figure 42-8.** Safari is now added

At the time of writing the location of the browser programs was as follows:

**Opera:** "C:\Program Files (x86)\Opera\launcher.exe"

**Firefox:** "C:\Program Files (x86)\Mozilla Firefox\firefox.exe"

**Safari:** "C:\Program Files (x86)\Safari\safari.exe"

**Internet Explorer:** "C:\Program Files (x86)\Internet Explorer\iexplore.exe"

**Chrome:** "C:\Program Files (x86)\Google\Chrome\Application\chrome.exe"

The double quotes must be included when adding the browsers to the *Add browser* dialog box.

Despite a thorough search on my computers and on the Internet, I was unable to find the location of the Application file for Microsoft Edge.

## Avoiding the Dreaded *<Span Lang*

Expression Web 4 has an irritating quirk. If you add the HTML tag that specifies the language, Expression Web 4 may clutter your code with many `<span lang="en-us">text</span>`. This not only slows down the page, but text may not line up correctly when you insert text using the WYSIWYG window.

To avoid this, do not insert the language in the `<html>` tag.

For example, the following code would cause the problem:

```
<DOCTYPE html>
<html lang=en>
```

This code will avoid the problem:

```
<DOCTYPE html>
<html>
```

As a further safeguard, click *Tools*, select the *Page Editor Options*. In the *General* tab there is a setting for 'Automatically switch keyboard to match language of surrounding text', but make sure this is not checked.

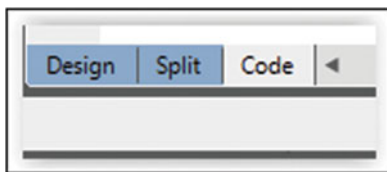
Make sure your *Default Page Language* is set to the language of your country.

If you wish, you could then use the following meta tag in the <head> section to specify the language:

```
<meta http-equiv="Content-Language" content="en-gb">
```

## Design (WYSIWYG) or Code View?

At the bottom left of the interface you will see buttons that allow you to flick instantly between the three available views as shown in Figure 42-9.



**Figure 42-9.** *Code view has been selected*

---

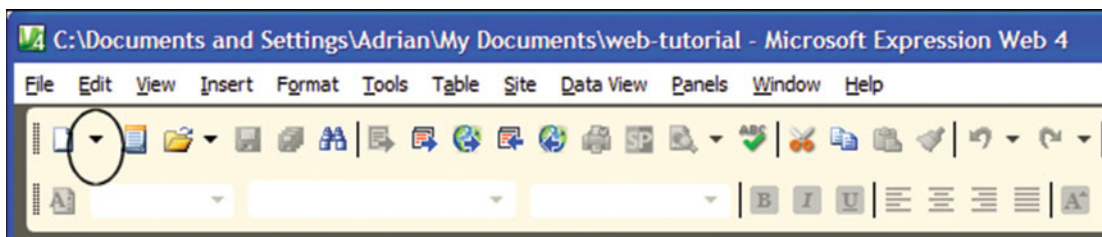
■ **Note** The Design view is WYSIWYG. You can view the page in *Code* view or *Design* view. You can alter the code in *Code* view, then view the result in *Design* view **without** saving the alterations. If the changed file is satisfactory then you can save it. After saving, as long as you have not closed the file down, you can still undo any changes using the Undo symbol on the toolbar (just like MS Word).

---

I rarely use the *Split* view because it does not display enough of either the code or the result of the code. Always preview a new file or a modified file by pressing F12 or opening it in a modern browser in the normal way.

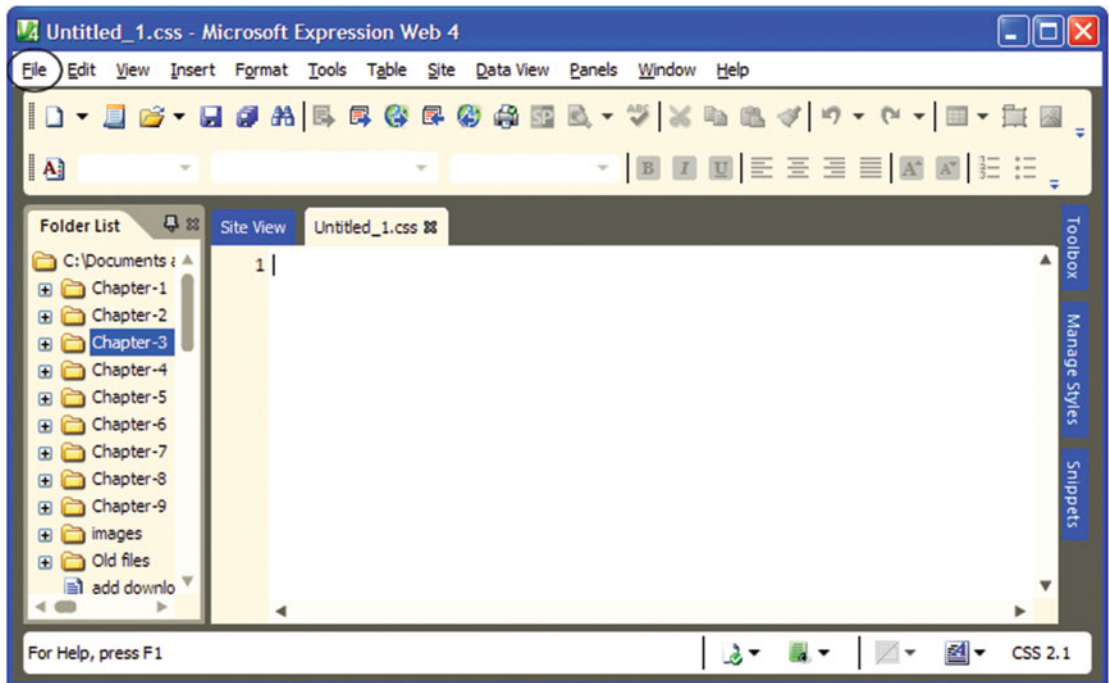
## Create an HTML Page or CSS File in MS Expression Web

Open MS Expression Web and click the down-arrow shown ringed in Figure 42-10.



**Figure 42-10.** *Click the down-arrow (shown circled)*

After clicking the down-arrow, a drop-down menu will appear; choose the type of file you wish to create by clicking either HTML or CSS in the menu. If you chose HTML you will see a pane with some basic code for a typical page. If you chose CSS, you will see an empty pane that will look like Figure 42-11.



**Figure 42-11.** To create a CSS file in Code view, type the CSS code into the empty pane in Code view

Type the CSS file code into the empty pane, then save it by clicking *File* (shown circled at top left) and then select *Save As*. Give the file the name suggested in the chapter and be sure to give it the suffix *.css.*, and make sure you have the correct folder in the top field of the *Save As* screen, then click *Save*.

---

■ **Note** With CSS files, the screen should be empty in *Design* view.

---

## Open an Existing Page or CSS file in MS Expression Web 4

In Windows Explorer, open the folder containing the website that you are working on, and right-click the file. Choose “Open with” and select Expression Web 4. You can check the box to make that the default for web pages, in which case, if you double-click any web page in the website’s folder it will load into Expression Web 4.

Alternatively, open Expression Web 4 and either click *File* then open the file; or if you have the folder list on the screen, double-click the folder containing the file you wish to load. You will see the files so that you can then click the appropriate CSS file to open it. Figure 42-12 shows the interface.

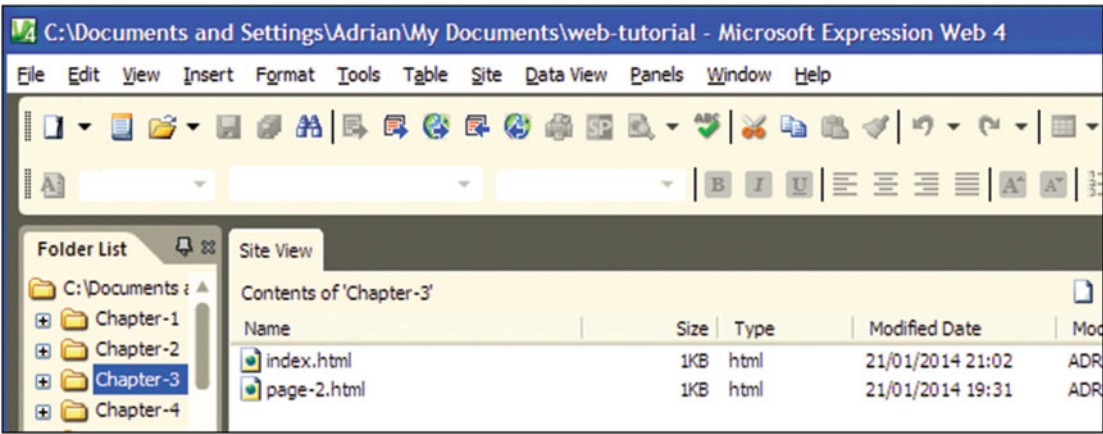


Figure 42-12. The files will be displayed

■ **Tip** For some useful tutorials try Bright Hub at: <http://www.brighthouse.com/computing/windows-platform/articles/28604.aspx>

## Download, Install, and Configure Blue Griffon (Free)

This program is available for Windows, Apple Mac, and Ubuntu computers.

Go to <http://www.bluegriffon.org/index.html>

Click Downloads on the top menu bar then click the icon that applies to your operating system.

Figure 42-13 shows the cursor poised over the icon for a Windows operating system.

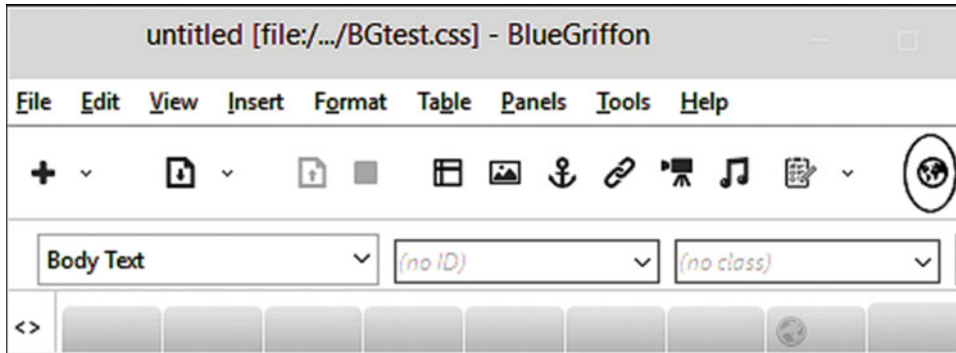


Figure 42-13. The Blue Griffon's download interface. Here you can select your operating system

At the time of writing the latest version was *bluegriffon-2.1.2*

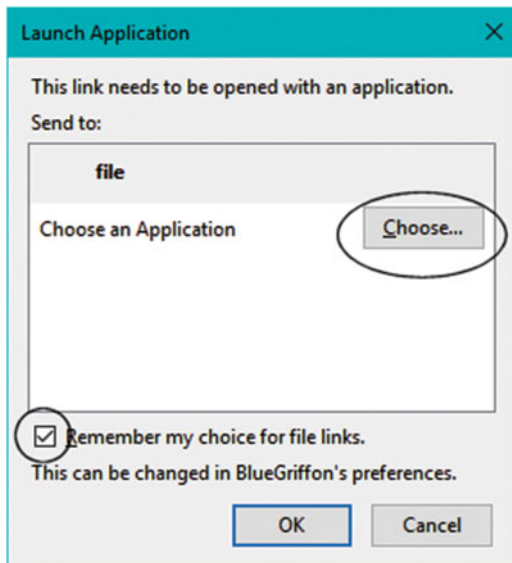
Click the icon to download the zip file to your *Downloads* folder. Then double-click the file to install Blue Griffon. The next panel will be the *Setup Wizard*; click *Next* and click *Next* again. Allow Blue Griffon to create a *Desktop icon*. On the next panel click *Install*. On the final screen click *Finish*.

Double-click the Blue Griffon icon on your desktop to check that it has installed properly. I think the funereal black and dark gray default interface is depressing, but fortunately you can choose a light scheme. Click *Tools* then select *Preferences*. With the *General* tab selected you will see a screen where you can change the interface theme from black to light. Figure 42-14 shows the toolbar in the light theme.

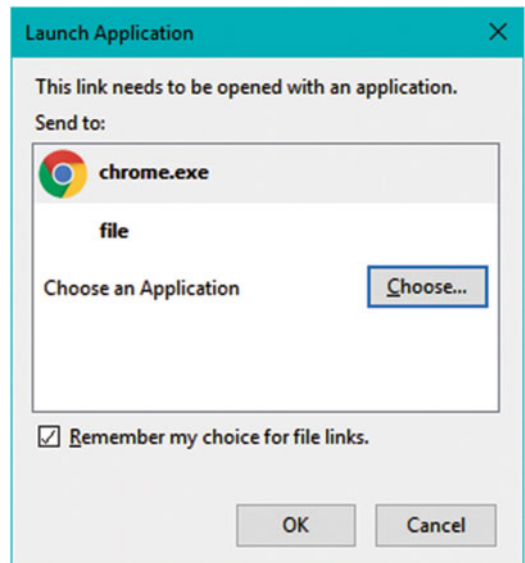


**Figure 42-14.** The Blue Griffon toolbar with the light theme

Blue Griffon is configured to show page previews in Mozilla Firefox, but you can configure it to show previews in other major browsers. To do this, click the globe icon shown circled in Figure 42-15. You will see the *Launch Application* dialog screen shown in Figure 42-16.



**Figure 42-15.**



**Figure 42-16.**

Click *Choose* on the dialog screen, and navigate to where the browser's program file is stored. For instance, if you wish to use Chrome, you would navigate to C:\Program Files (x86)\Google\Chrome\Application\chrome.exe.

Then click the .exe file icon or *chrome.exe* file. The icon for Chrome will appear as shown in Figure 42-16, then click OK.

At the time of writing the location of the browser programs was as follows:

**Opera:** "C:\Program Files (x86)\Opera\launcher.exe"

**Firefox:** "C:\Program Files (x86)\Mozilla Firefox\firefox.exe"

**Safari:** "C:\Program Files (x86)\Safari\safari.exe"

**Internet Explorer:** "C:\Program Files (x86)\Internet Explorer\iexplore.exe"

**Chrome:** "C:\Program Files (x86)\Google\Chrome\Application\chrome.exe"

The double quotes must be included when adding the browsers to the *Add browser* dialog box.

Despite a thorough search on my computers and on the Internet, I was unable to find the location of the Application file for Microsoft Edge.

## Close Down Blue Griffon

On the Toolbar, there appears to be no closing-down symbol, however, it will appear when you hover over the top-right corner as shown in Figure 42-17.



**Figure 42-17.** Hover over the top-right corner to expose the closing-down cross

If you hover a little to the left of the cross, you will see the other familiar symbols for manipulating the main window.



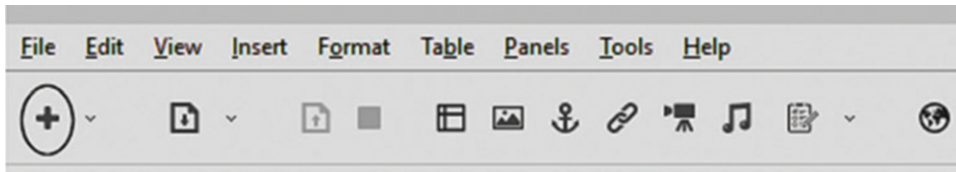
## How to Use the Blue Griffon HTML Text Editor

When you are asked to type some code into your HTML text editor, double-click the Blue Griffon icon on your desktop. When it loads, you will see the Blue Griffon interface.

The Toolbar is shown in Figure 42-18, and the globe on the far right will be grayed-out; this is normal if a file is not selected

## Creating a New HTML Page or a New CSS File in Blue Griffon

To start a new web page, click the *plus* sign on the far left (shown circled in Figure 42-18). Then the globe on the right will be colored blue and white, indicating that a sample page has been loaded.



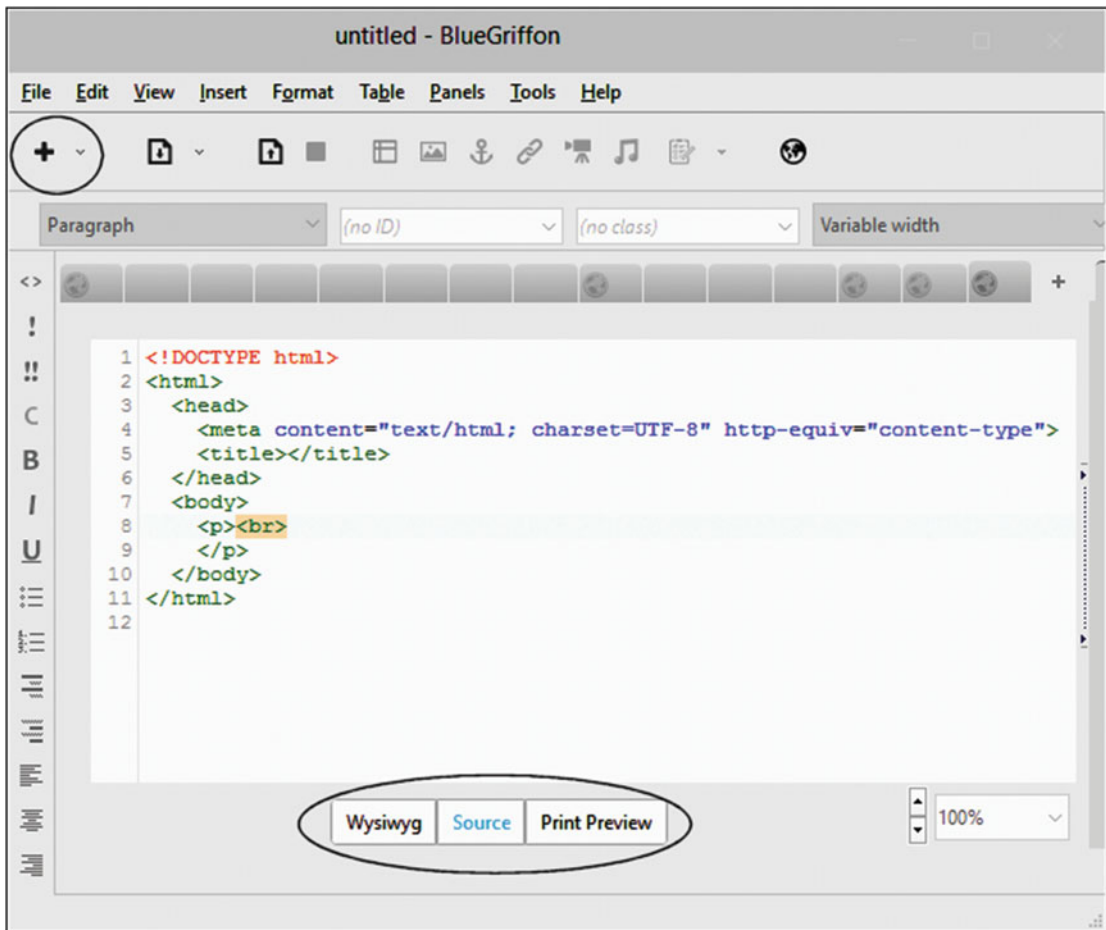
**Figure 42-18.** The BlueGriffon Tool bar. To start a new web page, click the plus sign shown circled

To enter *code*, always choose the *Source* view. Near the bottom of the interface, you will see three buttons, shown circled in Figure 42-19.



**Figure 42-19.** The three viewing buttons; *Source* view is selected. The button labels change to pale blue in the selected view

Figure 42-20 shows the default code provided when Blue Griffon is opened in Source view.

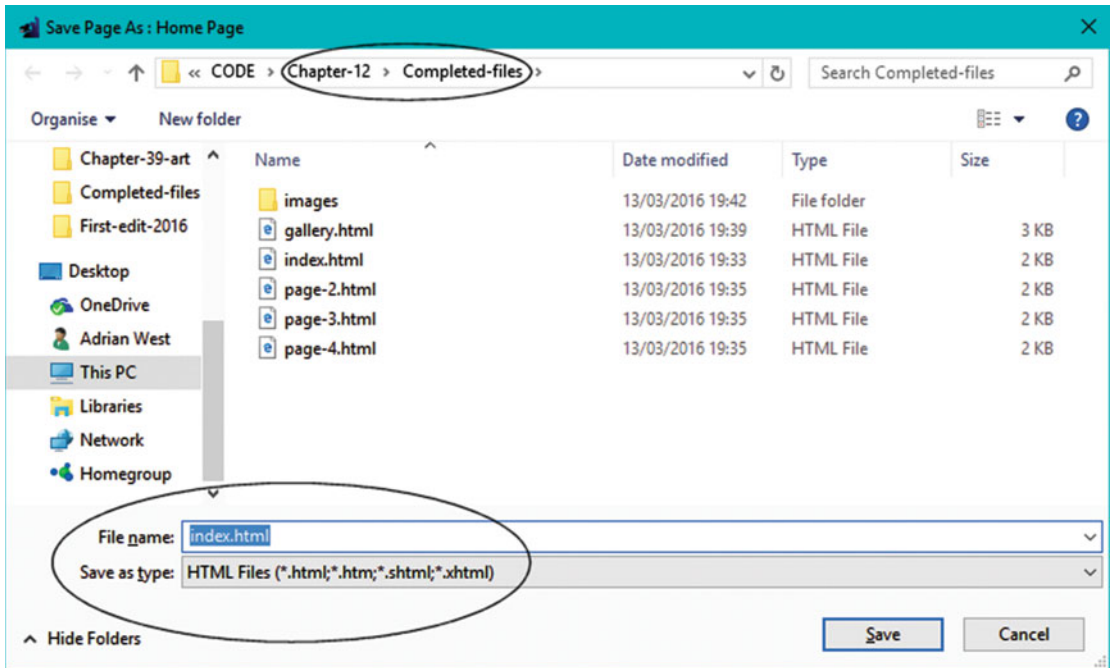


**Figure 42-20.** The interface for entering code. Note that the formatting Toolbar is in a vertical column on the left

To enter code, please follow these steps:

1. Open Blue Griffon in *Source* view.
2. Type in the HTML code as instructed by this book's chapters. You will often be asked to delete all the default code before entering the HTML code.
3. Click the WYSIWYG button to view the result.
4. Click *File* in the topmost menu, select *Save As* (in the bottom third of the drop-down list).

The *Save As* window is shown in Figure 42-21.



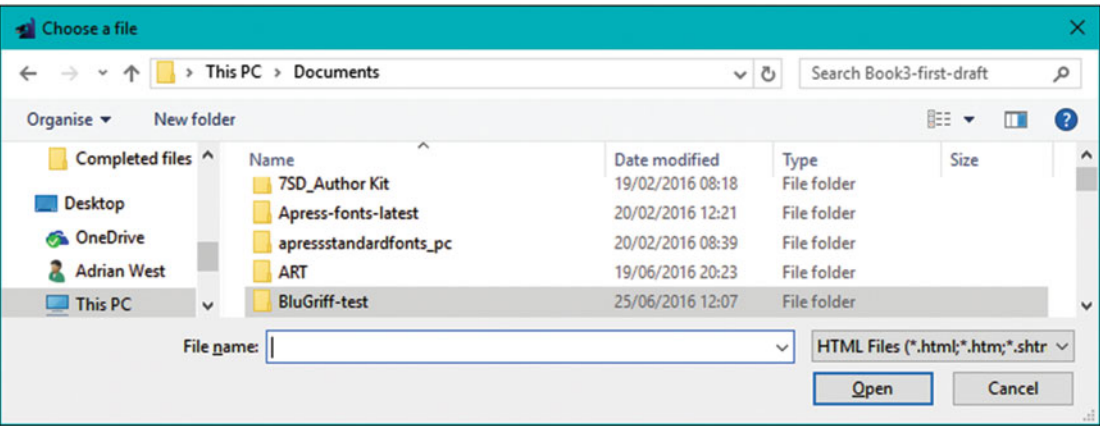
**Figure 42-21.** Saving a new file with Save As. Note that the color theme has switched from Blue Griffon's ugly black/muddy gray to the normal Window's colors

5. On this screen make sure the appropriate folder is showing in the top field (shown circled).
6. Enter the file name in the field below the main panel (shown circled) and click the *Save* button.

## When You Are Asked to Modify an HTML or CSS File in Blue Griffon

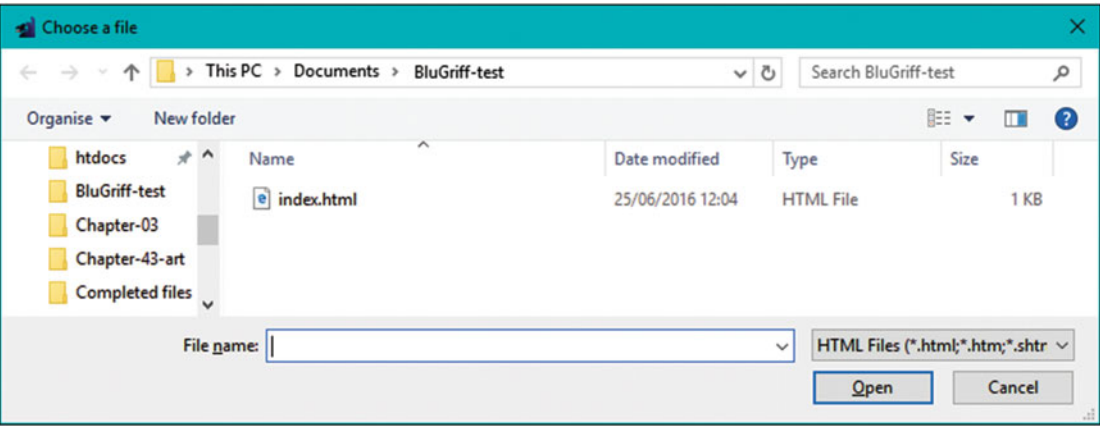
**Method 1:** Open the folder containing the file you wish to modify and RIGHT-click the file. Then select Blue Griffon from the list of Applications (programs).

**Method 2:** Open Blue Griffon and click *File* on the menu, then select *Open File*. Or use the shortcut key combination Ctrl+O to open the file that you want to modify. A new dialog box will appear as shown in Figure 42-22. Then you can navigate to the folder, for example, *web-tutorial*, and then navigate to a subfolder folder such as *Chapter-3*, or wherever your file is stored.



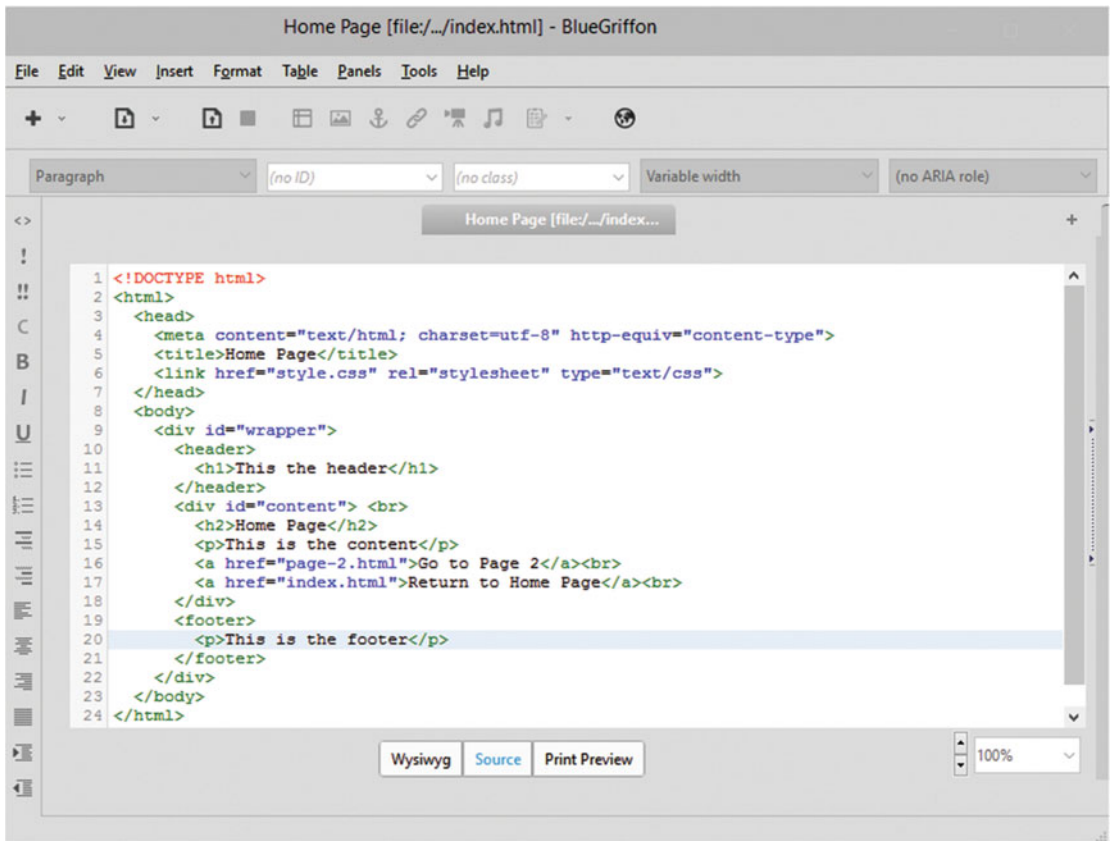
**Figure 42-22.** The dialog for opening a folder (in this case, a test folder named *BluGriff-test*)

Double-click the folder to open it and you will see the files within that folder, as in Figure 42-23.



**Figure 42-23.** Click the HTML or CSS file you wish to open

Click the file you wish to open and then click the button labeled *Open*. The result will look something like Figure 42-24 below.



**Figure 42-24.** The file name will appear in the top heading and in the tab that is open

When the file is loaded, in *source* view change the code as instructed in the chapter. The required change is indicated in bold type in each chapter's code.

## Previewing a Modified File in Blue Griffon

To see what the changed web page looks like, click the WYSIWYG button.

## Warning: When Creating or Modifying a CSS File in Blue Griffon

If you enter the following CSS code in the *Source* view and then save the file:

```

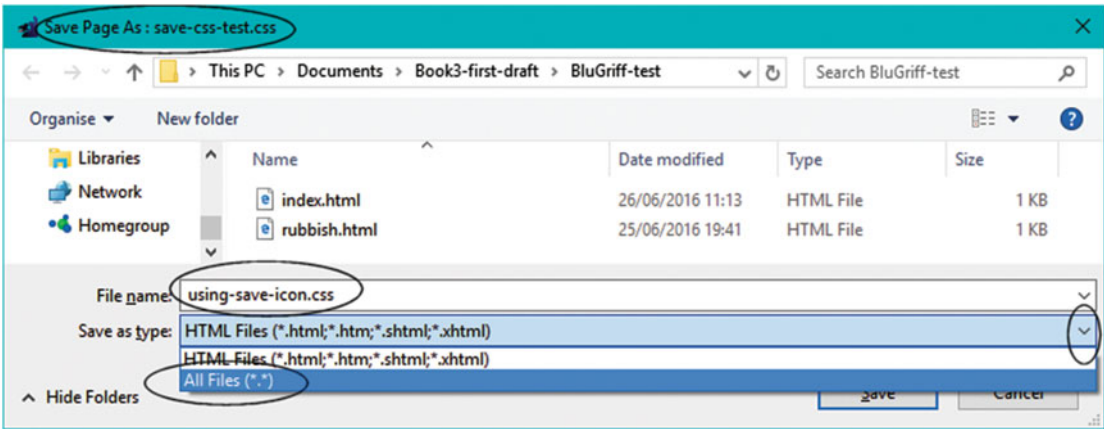
#wrapper {margin:auto; width: 1024px; border:2px blue solid;}
header {height:50px; text-align:center; color:red; background:yellow;}
h1 {font-size:22pt; font-weight:bold; text-align:center; }
h2 {font-size: 18pt; text-align:center;}
p {font-size: 14pt; text-align:center;}
a {font-size: 14pt;}

```

Blue Griffon will sometimes change the code as follows:

```
<html>
<head>
 <meta content="text/html; charset=windows-1252" http-equiv="content-type">
 <link rel="alternate stylesheet" type="text/css" Ē
 href="resource://gre-resources/plaintext.css" title="Wrap Long Lines">
</head>
<body>
 <pre>#wrapper {margin:auto; width: 1024px; border:2px blue solid;}
header {height:50px; text-align:center; color:red; background:yellow;}
h1 {font-size:22pt; font-weight:bold; text-align:center; }
h2 {font-size: 18pt; text-align:center;}
p {font-size: 14pt; text-align:center;}
a {font-size: 14pt;}
</pre>
</body>
</html>
```

To prevent this happening, read the content of the *Save* or *Save As* dialog very carefully before you save a file. The important items are shown circled in Figure 42-25.



**Figure 42-25.** Be sure to check the items shown circled before saving the CSS file

- Ensure that the correct file name appears in the top bar.
- Ensure that you have changed the file name extension to .css
- Click the down-arrow on the far right and select All Files (\*.\*)

---

■ **Tip** For \$7.50 you can download a PDF manual for the program by clicking *Help* on the top menu, then select *Buy user's manual*. The manual is geared to Windows and Macs but biased toward Macs. The manual is not essential because this chapter covers the points that you will need for all the projects in this book.

---

## Non-WYSIWYG Text Editors

The programs listed below are non-WYSIWYG, but you will be able to create websites with them by using the code and templates provided in the chapters.

---

■ **Caution** The warning given in the earlier section “Be Cautious when Downloading Text Editors” also applies to the downloading of non-WYSIWYG programs.

---

### Downloading and Installing TextEdit

TextEdit (free) is the easiest plain text editor to use and it is available for Windows or Mac.

Go to [http://download.cnet.com/windows/core-software-solutions/3260-20\\_4-6271832-1.html](http://download.cnet.com/windows/core-software-solutions/3260-20_4-6271832-1.html)

When you click the Download button the file *TESetup.exe* will be downloaded into your Downloads folder. In the Downloads folder, find and double-click the file and then click the *Next Step* button on the *Download.com* panel.

Accept the terms and conditions but click the Decline button on the next two offers. The TextEdit screen will appear, click the *Install now* button. On the *Additional tasks* panel check the box next to *Create a desktop icon* and click *Next*

You can ignore the License agreement and Register buttons and click OK.

### Downloading and Installing Komodo Edit (Free)

Go to <http://www.activestate.com/komodo-edit/downloads>

When the page opens, choose the appropriate installer for your operating system.

A form for requesting newsletters will appear, but you can ignore this and when a pop-up window appears click *Save File*, the file's name for the Windows operating system will be something like this: *Komodo-Edit-8.5.3- 14067.msi*. You will receive no indication that the file has downloaded, so go to your Downloads folder and you should see the file, double-click it to install it. On the next pop-up click *Run*. On the next pop-up click *Next*. On the next screen check the box to accept the agreement and click *Next*. On the next screen click *Next*. On the next screen click *Install*, this will take several minutes. When it has installed click the button labeled *Finish*.

You will see a *Getting started* guide and a news page. Close them both down. Click the Windows *Start* button and you will see the Komodo program. Hover your cursor over it and, holding down the right mouse button, drag the program onto the Desktop and select *Create a desktop icon*.

### Downloading and Installing NoteTab Light (Free)

<http://www.notetab.com/download-now.php>

Below the blue button you will see a pale blue button labeled *Download NoteTab light*.

Click that and choose to save the file *NoteTab\_Light\_Setup.exe* in your Downloads folder.

Go to your Downloads folder and double-click the file. On the next panel click *Run*. Click *Next* in the setup Wizard. Accept the License Agreement and click *Next*. Click *Next* again. Then accept the full installation by clicking *Next*. Check the box labeled *Don't Create a Start Menu Folder* and click *Next*. On the next screen choose to create a desktop icon, and click *Next*. On the next screen click *Install*. You will eventually be asked if you wish to try out the paid for versions, but click *No*.

When the program is loaded don't feel daunted by the many features and tabs. If your default browser is Mozilla Firefox or Chrome you can skip the next step and you are ready to use the program. Otherwise, click *Tools*, and on the drop-down menu click *Other browser*. If your default browser is Opera or Safari, navigate to the Programs Files (x86) folder, find the browser's folder, and open it. Then find and click the browser's *.exe* file.

## Downloading and Installing Notepad++ (Free)

The latest version at time of writing was 6.9.2

<http://notepad-plus-plus.org>

Click the *Download* button on the left panel. On the next screen click the green *Download* button on the left, but ignore the other buttons. You will be asked if you really want to download *npp.6.5.2 installer.exe*, and click *Save*. It will only take a few seconds to download into your Downloads folder. Find the file and double-click it. On the next screen click *Run*. A pop-up will ask which language you want; select English and click OK. Now follow the instructions as they appear. On the initial *Choose components* screen, accept the default components and click *Next*. On the next *Choose components* dialog, check the box labeled *Create shortcut on desktop*, then click *Install*. When it has installed, click *Finish*. Close the panel that describes the new features. Click the far-left icon on the toolbar to open a new page and you are ready to start entering some HTML code.

---

■ **Tip** When you have entered some HTML code you can view it in a browser by clicking *Run* on the top menu, then choose *Launch in Firefox* or whichever browser you prefer.

---

## Summary

In this chapter you learned how to download and install text editors. This chapter concludes the book. You should now be able to design websites using the code and templates provided in the book. But remember that this book cannot teach experience and expertise: this comes only with practice.



# Index

## ■ A

- Absolute positioning, 183–184
- Accelerated mobile pages (AMP), 394
- Attractive websites
  - colors, 148
  - drop-down menus, 149
  - encoded email address, 152–154
  - escramble JavaScript, 154–155
  - focus, 148
  - home page, 150
    - avoid gimmicks, 150
    - clear and concise, 150
  - navigation menu, 148–149
  - planning, 147
  - psychology
    - auto-start audios, videos
    - and slide shows, 151–152
    - slow loading websites, 151
    - user experience, 151
  - rules, 147
  - text issue, 149
  - useful, 152
- Axe vale and district conservation society, 399

## ■ B

- Bing search field, 302–303
- Blue Griffon
  - browser programs, 454
  - closing-down symbol, 454
  - CSS file/HTML page, 461–463
  - dialog screen, 459
  - download, 458
  - HTML text editor, 461
  - modification, 463, 465
  - operating system, 458
  - toolbar, 459
  - warning message, 465–466

- Box model, 136–137

- Bullet points

- bold heading
    - bullet-test.php* page, 313
    - display window, 314
  - CSS styles sheet, 314–316
  - margin and columns, 309
  - problem, 309
  - reduces gaps
    - code explanation, 313
    - code/source view, 312
    - reduced margins and gaps, 313
  - test page creation
    - bullet-test.html*, 310
    - daffodil picture, 311
    - htdocs folder, 310
    - text layout, 312
  - text editor, 309

- Byte order mark (BOM)

- in Expression web 4, 422
  - rare/unregistered character encoding text, 422
  - prevention, 421
  - validation error messages, 423–424

## ■ C

- Cascading style sheets (CSS), 4
  - color and styling, 23
  - features, 22
  - folder creation, 20–22
  - HTML code, 25–26
  - HTML pages, 19
  - need for, 25
  - overview, 19
  - reasons, 25
  - selector and declaration, 19
  - source code view, 22
  - style.css* explanation, 23–24
  - style sheet work, 20

Cascading style sheets (CSS) (*cont.*)

- three-column web page, 27
- undo and redo symbols, 26

Color

- CSS color codes, 71–72
- HTML text editor, 73–74
- pickers, 75
- schemes, 69–70
- steps of, 72
- tweaking (*see* Tweaking colors)
- web pages, 73
- Website (red, green and blue), 70–71
- WYSIWYG text editors, 75–77

Columns, 27, 35

Contact us page

- form-handler, 224
- form-handler and messages, 237
- form-handler-typical.php* page, 225–227
- message pages, 234
- steps of, 224–225
- style sheet (error message pages), 234
  - check box error message, 236
  - email error message page, 235
  - error message page, 235
  - thank-you page, 234–235
  - URLs, 236
- typical form-handler code, 227–231, 233
- user input validation
  - contact.php* page, 233–234
  - pop-up images, 233

Content management systems (CMS), 1, 2, 397

- advantage, 2
- CSS code, 400
- DIY websites, 400
- fine-tuning, 400
- home page, 398–399
- pitfalls
  - DIY sites, 398
  - home page, 397
  - inappropriate colors, 398
  - programs, 398
  - text and poor presentation, 398
  - unclear navigation, 397
  - video/slideshows, 398
- poor code, 400
- slow-loading pages, 400
- spam and hacking, 400

■ D

Domain name, 403

Drop-down menu design

- page creation, 267–268

planning

- code explanation, 265–266
- folder creation, 263
- home page, 263
- index.html*, 264–265
- tab, 263
- top-level list, 262
- publications page, 268–269
- show/hide
  - code explanation, 267
  - CSS style sheet code, 266–267
- Websites, 261

Drop shadows, 271

- four-sided shadow, 278–279
- home page, 272
- HTML element, 278
- text creation, 277–278
- website and images
  - code explanation, 273
  - htdocs folder, 272
  - rounded corners, 274–275
- white border image, 275–276

■ E

Enhancement

- font size, 57
- Internet Explorer 8 (IE8), 51–52
- JavaScript and hyperlink tags, 53
  - code explanation, 54
  - conditional code, 53
  - CSS code explanation, 56
  - semantic tags, 53
  - style sheet changes, 55
  - unordered list, 56
- semantic tags, 51

escramble JavaScript, 154–155

Excessive repetition, 173

Expression Web 4

- authoring details, 452
- code view/design (WYSIWYG), 456
- design view, 453
  - browser list, 454
  - dialog box, 454
  - Safari list, 454
  - steps, 453
- Dreaded <Span Lang, 455
- HTML page/CSS file, 456–457
- open existing
  - page/CSS file, 457–458
- toolbars, 452
- websites, 451

External links, 172

## ■ F

- Feedback form design, 157
  - code explanation, 160–161, 165–167
  - contact.html* file, 159–160
  - create, 158
  - form code (*contact.html*) page, 163–164
  - internal style, 161–163
  - vertical menu, 159
- Feedback form design feedback form, 158
- File transfer protocol (FTP) client
  - automatic connections, 412
  - connection details, 410–411
  - date format, 413
  - download (folders and files), 416
  - FileZilla, 404
  - folder hierarchy, 416
  - folders and files, 414
  - host
    - adding clients, 406–407
    - change settings, 405–406
    - configuration steps, 404
    - connection, 410
    - firewall, 405
    - private and public check boxes, 407–408
    - Windows 7 firewall, 408
  - panes, 409, 410
  - quick connect bar, 409, 410
  - stored websites, 412
  - storing folder details, 412
- fitvids-test.html* file, 392
- Fixed-width layouts, 81
- Float-drop and box model
  - appearance, 134–136
  - background websites, 131–132
  - box model, 136–137
  - distinguishable tiles, 132
  - float-drop, 136
  - pictorial tiles, 133
  - seamless tiles, 133
  - semi-seamless tiles, 132
  - title creation, 131
- form-handler-typical.php* page, 225–227
- Four-column web pages
  - code explanation, 38
  - content columns, 35
  - create new folder, 35
  - CSS code explanation, 38
  - forward planning, 40–41
  - home page, 36–37
  - hyperlink menu, 35
  - insert-style, 38
  - page creation, 36
  - two pages creation, 39–40

## ■ G

- Google search field
  - embedded code, 305–306
  - testing, 306
  - websites, 304, 305
    - google-search-1.html*, 304
- Graphics programs
  - background color styles, 433
    - brown, 433
    - columns, 434
    - content and elements, 435–436
    - four-column page, 434
    - gold, 433
    - light green, 433
    - pale blue, 433
    - purple, 433
    - red, 433
    - resulting layout, 434
  - comments, 431
  - compression dialog box, 441
  - CSS styles sheets, 437
  - definition, 439
  - downloaded file, 438
  - File Minimizer pictures, 439
  - footer, 436
  - HTML text and tags, 430, 436
  - Irfan-View, 439
  - main structural tags, 431–432
  - optimization steps, 438
  - resizing dialog box, 440
  - resources, 442
    - graphics, 442
    - HTML and CSS, 442
    - Irfan tutorials, 442
    - PHP, 442
    - responsive web design, 442
  - web page structure, 429

## ■ H

- Horizontal menu
  - 3D buttons
    - code explanation, 102–103
    - modification, 101–102
  - advantage of, 104–105
  - download and installation, 99–100
  - plain buttons, 103–104
  - templates, 101, 104
  - templates-h.zip, 99
  - tutorial section
    - Chelsea Pensioners image, 107
    - code explanation, 108
    - create gallery, 105

- delete, 107
  - insert, 106–107
  - move, 106
- vertical menu, 100–101
- Horizontal menu button
  - downloaded folder, 317–318
  - home page styling, 319–320
  - indicator style, 320–322
  - page indicator, 322–323
  - selected menu button, 318–319
- Host, 404
- Hyperlinks
  - code explanation, 18
  - code/source view, 14
  - content, 13–14
  - folder creation, 13
  - home page creation, 15
  - pages adding hyperlinks, 16–18
  - second page adding, 16
  - sections, 13
  - text displays, 15

## ■ I

Internet Explorer 8 (IE8), 51–52

## ■ J

JavaScript, 53, 394

- code explanation, 54
- conditional code, 53
- CSS code explanation, 56
- semantic tags, 53
- style sheet changes, 55
- unordered list, 56

## ■ K

Komodo edit, 467

## ■ L

Liquid layouts, 81

## ■ M

Media queries

- create folder for demo, 347
- CSS code explanation, 348
- media-query-demo.html*, 347

Member's page

- code view, 282–283
- creation, 284
  - home page, 285–286
  - tuliptime5.php*, 284–285

- form code explanation, 284
  - login page, 284
- login-handler
  - code explanation, 287–288
  - creation, 286
  - login-handler.php file, 286–287
- login page, 282
- XAMPP htdocs folder, 282
- Menu buttons
  - 3D buttons
    - amalgamated style sheet, 66–67
    - code explanation, 65–68
    - creation, 63–64
    - index.html* file, 67–68
    - insert option, 65
    - style sheet, 64
  - code explanation, 62–63
  - create folder, 59, 61
  - feature, 63–64
  - flat-colored buttons, 60
  - HTML text editor, 62
  - index-plain.css* file, 61–62
  - plain buttons, 60
- Meta tag/keywords, 171
- Mobile devices
  - bullet points, 387–388
  - collapsible responsive menu
    - code explanation, 373–375
    - collapsed menu, 372
    - CSS code explanation, 375–377
    - desktop/laptop computer screen, 371
    - medialoot.com, 372–373
    - menu collapsed, 372
    - menu revealed, 372
  - definitions, 344
  - desktop website, 343
  - factors, 345
  - four-column responsive web page
    - home page, 369
    - source code, 369–370
  - header text, 378
    - code file, 378
    - collapsed menu, 379
    - drop-down sub-menu, 379
    - expanded menu, 379
  - media queries, 381
    - create the folder, 347
    - CSS code explanation, 348
    - media-query-demo.html*, 347
  - menu buttons, 386–387
  - menu buttons (drop-down)
    - code explanation, 361–362
    - drop-down menu buttons, 360–361
    - menu option, 359
    - reorganized menu, 359–360

- navigation menus
  - CSS style website (*style.css*), 353–356
  - desktop view port, 351–352
  - downloadable folder, 351
  - home page (*index.html*), 352–353
  - responsive pages, 356
  - smart-phone, 351
  - testing responsive websites, 356–357
- overview, 359
- page descriptions
  - HTML page, 383
  - supplementary style sheet links, 383–386
- proportional dimensions, 349
  - horizontal desktop, 349
  - RWD pages, 349–350
- responsive websites
  - AMP, 394
  - content strategy, 393
  - designer control, 394
  - fonts, 393
  - images, 350
  - JavaScript, 394
  - slow loading, 394
  - solution, 346
  - websites, 382
- smartphone, 343
- three-column responsive web page
  - code explanation, 367–369
  - home page, 366
  - page creation, 366–367
- two-column responsive web page
  - code explanation, 365
  - columns layout, 362
  - home page, 362–364
  - steps to view file, 362
- unordered list tags, 387–388
- videos responsive
  - butterfly.html* file, 389–390
  - desktop view, 390
  - fitvids-test.html*, 391–392
  - HTML format, 389
  - YouTube/Vimeo, 390–391
- view port statement, 346–347
- websites/style sheets, 345
- Multi-row menu, 333
  - create folder, 333
  - double row creation
    - code explanation, 335, 337
    - CSS code explanation, 336
    - home page code, 334–335
  - home page, 334
  - odd number of buttons, 337–338

## ■ N, O

- Notepad++, 468
- NoteTab Light, 467

## ■ P, Q

- page-2.html* file, 89
- PHP *include* command, 213
  - create folder, 213
  - external footer file (*footer.html*), 214–216
  - include*, 214
  - page creation, 216
    - about.php* page, 218–219
    - contact.php* page, 220–222
    - location.php* page, 219
    - spring-plants.php*, 218
    - template creation
      - (*spring-bulbs.php*), 216–217
      - terms.php* page, 220
- Pictorial tiles, 133
- Pictures
  - CSS file-style
    - code/source view, 47–48
    - header image, 47
    - heading changes, 48
    - source code explanation, 48
  - galleries, 333
    - code explanation, 340–341
    - four-row galleries, 338–340
  - modern browser, 44
  - older browsers problem
    - home page, 45
    - unsatisfactory appearance, 44
  - plain text editor, 46–47
  - WYSIWYG editor, 45–46
- Pitfalls. *See* Content management systems (CMS)
- Positioning techniques, 183
  - absolute positioning, 183–184
  - floating image
    - float-1.html*, 190–191
    - float rosettes, 190
    - <clear> properties, 191–192
  - margin properties, 192
  - relative (*see* Relative positioning)
  - vertical position, 185
  - wrapper boundaries
    - code explanation, 189
    - crocus image, 187–188
    - internal style, 188
    - rightcol.div*, 188

Printable order form  
 order form  
   HTML editor, 298  
   printed version, 298–299  
   print-order.css, 298  
   styling code explanation, 300  
 vertical menu  
   order form button, 289  
   vmenu.html, 290  
 view and modification, 290–291  
   code explanation, 295  
   order-form.php file, 292–294  
   style sheet, 297–298  
   unfamiliar code explanation, 295–297  
 Publications page  
   create publications page, 268–269  
   home page, 269  
   *publications.html*, 268

## ■ R

Relative positioning, 185  
   code/source view, 186  
   experiment, 186  
   logo image, 187  
   rosette logo, 187  
 Responsive website design (RWD).  
   *See* Mobile devices

## ■ S

Save time and reduce tedium  
 external files  
   creation, 205  
   footer, 207  
   header, 205  
   horizontal menu, 206  
   *index.html* page, 208–209  
   remote host, 211  
   vertical menu, 206–207  
   XAMPP server, 209–210  
 PHP code, 196  
 PHP command, 204–205  
 server-side language  
   exiting server, 197  
   installation, 197  
 time-saving code, 195  
 XAMPP  
   closing down, 200  
   control panel, 199  
   desktop icon, 199  
   htdocs folder, 202  
   icon, 198  
   installation, 197–198  
   password page, 201

  security console, 200–201  
   shortcuts, 203  
   testing PHP files, 203–204  
 Screen sizes and resolutions  
   3D menu templates  
     code explanation, 83–84  
     rounded corners, 83  
     steps, 82  
   data usage, 79  
   fixed-width layouts, 81  
   handheld device, 80  
   liquid layouts, 81  
   monitor-related considerations, 80  
   plain menu buttons, 84–85  
   rounded corners, 79  
   semi-liquid layouts, 81–82  
   templates, 82  
 Seamless tiles, 133  
 Search engine (SEs) optimization  
   advertise your website, 177  
   crawler trawls, 169  
   elements, 170  
   false promises, 177  
   how search engines work, 169–170  
   keywords and phrases, 170  
     body text headings, 172  
     carpets and cleaning, 171  
     external links, 172  
     meta tag, 171  
     product/service, 171  
     title tag, 171  
     variations, 170  
     well-designed internal links, 173  
   pages of, 174  
   restrictions, 173  
   spring garden home page  
     code/source view, 178–180  
     create page for SEO, 178  
     explanation of code, 180–182  
     keywords, 177–178  
   web page, 175  
     home page.html, 176  
     SEO fault explanation, 176–177  
 Search field, 301  
   Bing search field, 302–303  
   create folder, 301  
   Google search  
     embedded code, 305–306  
     google-search-1.html, 304  
     testing, 306  
     websites, 304, 305  
   Yahoo! search field, 303–304  
 Semi-liquid layouts, 81  
 Semi-seamless tiles, 132  
 Slideshows and videos

- BarelyFitz designs, 241
  - creation, 245
  - slideshow.php* page, 243–244
  - steps of, 242
  - testing, 244
  - user complete control, 242
- benefits of, 240
- housekeeping tips, 241
- multimedia clips, 239
- overview, 239
- royalty-free clips, 239
- slideshow adding, 241
- videos, 240
- YouTube (*see* YouTube)

## ■ T

- Tables, 139
  - design/WYSIWYG view, 141
  - four-column table
    - demonstration page, 143–145
    - HTML code explanation, 145
    - internal style, 145
    - steps, 143
    - web page, 142
  - insert and delete, 141
    - default double border style, 141–142
  - two-column data table, 139
    - code explanation, 140–141
    - simple-2col-table.html*, 139
- Tab menus
  - create folder, 251
  - CSS file code
    - code explanation, 256
    - tabs.css*, 255
  - home page, 252
    - code explanation, 254
    - page to page, 254–255
    - six-tab menu (*index.html*), 253
  - new tabs creation, 257
  - page creation, 257–259
  - page four view, 252
- Text editors, 449
  - Blue Griffon
    - browser programs, 454
    - closing-down symbol, 454
    - CSS file/HTML page, 461–463
    - dialog screen, 459
    - download, 458
    - HTML text editor, 461
    - modification, 463, 465
    - operating system, 458
    - toolbar, 459
    - warning message, 465–466

- Expression Web 4
  - authoring details, 452
  - code view/design (WYSIWYG), 456
  - design view, 453–455
  - Dreaded <Span Lang>, 455
  - exist page/CSS file, 457–458
  - HTML page/CSS file, 456–457
  - toolbars, 452
  - websites, 451

### WYSIWYG

(*see* WYSIWYG program)

- Three-column web page
  - code explanation, 29
  - creation, 27
  - CSS, 27
  - id and class differences, 33
  - page structure modification, 28–29
  - style sheet creation, 30
    - code explanation, 30, 32
    - tag attribute (class), 31–32
- Time-saving code, 195
- Typical form-handler
  - code explanation, 227
  - contact.php* page, 228, 231
  - email image, 232
  - email message, 230
  - fall-back message, 229
  - feedback from, 228
  - feedback message, 232
  - form-handler-typical.php* page, 231
  - message pages, 228
  - phone field, 230
  - remove format, 230
  - source/code view, 231
  - thank-you page, 228
  - username text, 230
  - XAMPP/Apache server, 228

## ■ U

- User Experience (UX), 151

## ■ V

- Validation (web pages)
  - byte order mark (BOM)
    - error messages, 423–424
    - Expression Web, 422
    - rare/unregistered
      - character encoding text, 422
  - CSS style sheet
    - error, 426
    - HTML5 logo, 426–428
    - vendor errors, 426

Validation (web pages) (*cont.*)  
     W3C validation, 425  
     warning message, 425–426  
 DCOTYPE, 417  
 HTML5 errors and reports  
     errors, 420–421  
 valid pages, 416  
 W3C validation  
     access, 417  
     error and a warning, 418  
     interface, 417  
     report, 419  
 Vertical and  
     horizontal menus  
 changing pages, 124–125  
 code explanation, 123–124  
 home page  
     index.html, 122–123  
 page background, 122  
 create folder, 121  
 style sheet, 125  
     code explanation, 126–127  
 templates creation, 127–128  
 Vertical menu, 325  
 adaptation  
     code/source view, 88–89  
     delete, 88  
     home page, 88  
     HTML text editor, 88  
     replacement, 88  
 code explanation, 96–98  
 downloadable folder, 325–327  
 external style sheet, 98  
 gallery page  
     code/source view, 95–96  
     creation, 93  
 indicate menu pages, 330–331  
 location page, 92–93  
 menu pages, 329–330  
 minor changes, 328–329  
*page-2.html*, 89  
     code/source view, 90–91  
     left-align, 91–92  
     partial view, 89  
     WYSIWYG editors, 90  
 step of, 87  
 Videos. *See also* Slideshows  
     and videos  
     code explanation, 246  
     fall-back version, 249  
     file format conversion, 247  
     HTML5 code, 247  
     today's formats, 245–246  
     video-page.php file, 248  
     yesterday's formats, 245

## ■ W, X

Web crawler, 169  
 Web page structure, 429  
 Websites  
     browser installation, 5  
     CMS  
         advantages of, 2  
         HTML and CSS, 3  
     CSS, 4  
     definitions, 1  
     folder creation, 6  
     HTML code, 4, 11  
     HTML text editor installation, 5–6  
     images  
         appearance creation, 114–115  
         code explanation, 116–117, 119–120  
         CSS style sheet, 115–116  
         header section, 120  
         image file formats, 111, 113  
         page index.html, 118–119  
         pictures, 114  
     methods creation, 2  
     sections, 1  
     structure, 6–7  
         create the structure, 9  
         file name, 10  
         HTML code, 8  
         source code, 9  
         web pages, 7–8  
     troubleshooting, 11  
 Well-designed internal links, 173  
 WYSIWYG program  
     Blue Griffon, 449  
     Expression Web, 449  
         Blue Griffon, 450  
         design view, 450  
     non-WYSIWYG text editors  
         download and installation, 467  
         Komodo edit, 467  
         Notepad++, 468  
         NoteTab light, 467  
     text editors, 451  
         blue griffon color picker, 76–77  
         expression, 75  
         extended color picker, 75

## ■ Y, Z

Yahoo! search field, 303–304  
 YouTube, 249  
     account, 249  
     Google account, 249  
     host video, 250  
     Vimeo videos, 390