

Making Everything Easier!™

Web Design

ALL-IN-ONE

FOR

DUMMIES®

5 BOOKS
IN 1

- Getting Started
- Designing for the Web
- Building Web Sites
- Web Standards and Testing
- Publishing and Site Maintenance

IN FULL COLOR!

Sue Jenkins

Concept

Design

Produce

Test

Launch



Web Design

ALL-IN-ONE

FOR

DUMMIES®

by Sue Jenkins



WILEY

Wiley Publishing, Inc.

Web Design All-in-One For Dummies®

Published by

Wiley Publishing, Inc.

111 River Street

Hoboken, NJ 07030-5774

www.wiley.com

Copyright © 2009 by Wiley Publishing, Inc., Indianapolis, Indiana

Published by Wiley Publishing, Inc., Indianapolis, Indiana

Published simultaneously in Canada

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 646-8600. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, (201) 748-6011, fax (201) 748-6008, or online at <http://www.wiley.com/go/permissions>.

Trademarks: Wiley, the Wiley Publishing logo, For Dummies, the Dummies Man logo, A Reference for the Rest of Us!, The Dummies Way, Dummies Daily, The Fun and Easy Way, Dummies.com, Making Everything Easier, and related trade dress are trademarks or registered trademarks of John Wiley & Sons, Inc. and/or its affiliates in the United States and other countries, and may not be used without written permission. All other trademarks are the property of their respective owners. Wiley Publishing, Inc., is not associated with any product or vendor mentioned in this book.

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

For general information on our other products and services, please contact our Customer Care Department within the U.S. at 877-762-2974, outside the U.S. at 317-572-3993, or fax 317-572-4002.

For technical support, please visit www.wiley.com/techsupport.

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic books.

Library of Congress Control Number: 2009924573

ISBN: 978-0-470-41796-6

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1



WILEY

About the Author

Sue Jenkins is a Web and graphic designer, illustrator, photographer, writer, award-winning teacher, and the principal and creative director of Luckychair (www.luckychair.com), a full-service design studio that has been helping businesses across the United States look better since 1997. When not designing, this Adobe Certified Expert/Adobe Certified Instructor teaches three-day courses in Dreamweaver, Illustrator, and Photoshop at Noble Desktop in New York City. In addition to this *For Dummies* book, Sue is the author of *Dreamweaver CS4 All-in-One For Dummies* (Wiley), *Dreamweaver 8 All-in-One Desk Reference For Dummies* (Wiley), *Web Design: The L-Line, The Express Line to Learning* (Wiley), and *How to Do Everything Illustrator CS4* (McGraw-Hill), and she was the technical editor of Ed Tittel's *HTML, XHTML, and CSS For Dummies*, 6th Edition (Wiley). Sue is also the software instructor in three of ClassOnDemand's (www.classondemand.com) Adobe Training DVDs, namely *Dreamweaver for Designers* (winner of a 2007 Bronze Telly Award), *Designer's Guide to Photoshop*, and *Designer's Guide to Illustrator*. Sue lives with her husband and son in Pennsylvania.

Dedication

To my father, for his love and support, for his amazing sense of design, and for teaching me the principles of honesty and hard work.

Author's Acknowledgments

As always, I'd like to thank my fantastic agent, Matt Wagner, for finding me cool projects and being such an all-around good guy; to Executive Editor Steve Hayes, for his care and assistance in crafting the tone of this book; to my project editor, Kim Darosett, who is meticulous and kind and a complete joy to work with; to John Edwards, my copy editor; and to my technical editor, Mike Lerch, for his excellent comments and suggestions — you guys rock. Thanks also to my production coordinator, Patrick Redmond, for all his hard work and to all the other folks at Wiley who were a part of this project for their fantastic work at making this color book look so fabulous. Wiley is truly a smart company, and I humbly appreciate the fine-tuned machinery. I'd also like to thank my husband, Phil, and son, Kyle, whose sweetness, love, encouragement, and humor helped me write this book.

Publisher's Acknowledgments

We're proud of this book; please send us your comments through our online registration form located at <http://dummies.custhelp.com>. For other comments, please contact our Customer Care Department within the U.S. at 877-762-2974, outside the U.S. at 317-572-3993, or fax 317-572-4002.

Some of the people who helped bring this book to market include the following:

Acquisitions, Editorial, and Media Development

Project Editor: Kim Darosett

Executive Editor: Steven Hayes

Copy Editor: John Edwards

Technical Editor: Mike Lerch

Editorial Manager: Leah Cameron

Media Development Project Manager: Laura Moss-Hollister

Media Development Assistant Project Manager: Jenny Swisher

Media Development Assistant Producers:
Angela Denny, Josh Frank, Shawn Patrick,
and Kit Malone

Editorial Assistant: Amanda Foxworth

Sr. Editorial Assistant: Cherie Case

Cartoons: Rich Tennant (www.the5thwave.com)

Composition Services

Project Coordinator: Patrick Redmond

Layout and Graphics: Samantha K. Allen,
Reuben W. Davis, Sarah Philippart

Proofreaders: Melissa D. Buddendeck,
Amanda Graham

Indexer: Christine Spina Karpeles

Publishing and Editorial for Technology Dummies

Richard Swadley, Vice President and Executive Group Publisher

Andy Cummings, Vice President and Publisher

Mary Bednarek, Executive Acquisitions Director

Mary C. Corder, Editorial Director

Publishing for Consumer Dummies

Diane Graves Steele, Vice President and Publisher

Composition Services

Gerry Fahey, Vice President of Production Services

Debbie Stailey, Director of Composition Services

Contents at a Glance

<i>Introduction</i>	1
<i>Book I: Getting Started</i>	7
Chapter 1: Starting with a Plan	9
Chapter 2: Defining the Audience	43
Chapter 3: Gathering Content	61
Chapter 4: Choosing the Right Tools	91
<i>Book II: Designing for the Web</i>	123
Chapter 1: Defining the Look and Feel	125
Chapter 2: Mocking Up the Design	153
Chapter 3: Slicing and Optimizing Web Graphics	175
<i>Book III: Building Web Sites</i>	205
Chapter 1: Adding Text, Images, and Links	207
Chapter 2: Organizing Content with Tables and Lists	249
Chapter 3: Styling with Cascading Style Sheets	271
Chapter 4: Understanding CSS Style Properties	301
Chapter 5: Creating Web Layouts	333
Chapter 6: Constructing Navigation Systems	359
Chapter 7: Designing Web Forms	391
Chapter 8: Making Your Pages Interactive	423
Chapter 9: Building Web Sites	459
<i>Book IV: Web Standards and Testing</i>	481
Chapter 1: Following Web Standards	483
Chapter 2: Testing, Accessibility, Compliance, and Validation	509
<i>Book V: Publishing and Site Maintenance</i>	543
Chapter 1: Domain Registration and Hosting	545
Chapter 2: Publishing Your Site	571
Chapter 3: Search Engine Optimization and Site Maintenance	593
<i>Index</i>	621

Table of Contents

Introduction 1

About This Book.....	1
Foolish Assumptions	2
Conventions Used in This Book	2
How This Book Is Organized.....	3
Book I: Getting Started.....	3
Book II: Designing for the Web.....	4
Book III: Building Web Sites.....	4
Book IV: Web Standards and Testing	4
Book V: Publishing and Site Maintenance.....	4
Icons Used in This Book.....	5
Where to Go from Here.....	5

Book I: Getting Started 7

Chapter 1: Starting with a Plan 9

Understanding the Different Phases of a Web Project	10
Determining the Site's Purpose	11
Checking out the competition	11
Gathering information	12
Developing a statement of purpose	13
Building a Site Image.....	15
Determining Site Content	16
Minimum requirements	16
Marketing and sales content.....	18
Diagnosing the Site's Dynamic Requirements	24
Defining Ways to Attract Visitors	27
E-newsletters.....	28
Free tips and articles	31
Blogs.....	32
Polls and calculators.....	37
Contests and sweepstakes	40

Chapter 2: Defining the Audience 43

Defining the Target Audience	44
Doing informal market research.....	44
Gathering Internet usage statistics	45
Sizing up the competition	48

Defining the Ideal Site Visitor	52
Determining Benefits to Site Visitors.....	56
Defining the true benefits	57
Taking the visitor's perspective	58

Chapter 3: Gathering Content61

Defining Site Content Requirements.....	62
Gathering content	62
Crafting the vision of the site.....	63
Building Wireframes	70
Gathering Text and Graphics.....	73
Hiring a copywriter	75
Hiring freelance artists	76
Licensing stock images.....	77
Choosing page titles and meta-tag data	80
Organizing Site Content.....	85
Building a Site Map	87

Chapter 4: Choosing the Right Tools91

Working with Web Editors.....	92
Selecting a Web editor	93
Understanding HTML and CSS structure	95
Looking at Web page structure.....	97
Building a Web page.....	98
Saving Web files	100
Choosing the Right Graphics Software.....	102
Graphics programs.....	102
Web graphic optimization programs	107
Working with Color	108
Using Web-safe colors.....	108
Using hexadecimal colors	110
Choosing a Shopping Cart.....	114
Using PayPal shopping carts	115
Checking out Google Checkout	116
Looking into third-party and Web-hosted shopping carts.....	116
Building custom shopping carts	117
Planning for secure transactions.....	118
Knowing When to Hire a Programmer.....	119
Taking a look at your dynamic content needs.....	119
Finding a good programmer.....	121

Book II: Designing for the Web..... 123

Chapter 1: Defining the Look and Feel125

Working with the Client to Make Design Choices125
 Defining a Site Theme Using Target Data126
 Making Basic Layout and Design Decisions.....129
 Choosing a size for your site.....130
 Selecting a fixed-width or flexible layout132
 Choosing a method for printing the layout.....137
 Picking a color palette138
 Choosing the right fonts140
 Selecting a Navigation System.....142
 Choosing a location and style.....145
 Determining how to handle submenus.....146
 Organizing the Site’s Look and Feel148
 Positioning the brand148
 Designing layouts on the grid148
 Making a layout checklist.....150

Chapter 2: Mocking Up the Design153

Understanding the Value of a Mock-up154
 Working from a Site Map155
 Creating the Mock-up157
 Blocking out the parts of the page157
 Designing “above the fold”160
 Unifying the layout with design elements163
 Finalizing the Mock-up.....166
 Showing the subnavigation167
 Presenting the mock-up to the client.....169
 Creating Additional Web Graphics.....171
 Header graphics171
 Rollover graphics172
 Background images173
 Other graphics.....173

Chapter 3: Slicing and Optimizing Web Graphics175

Web Graphics 101.....176
 Color mode.....177
 Color gamut warnings.....178
 Resolution180
 Unit of measure.....182
 File and page size182

Optimizing and Slicing Graphics	183
Understanding optimization	183
Choosing an optimization program	183
Optimizing using Save for Web & Devices.....	184
Slicing up graphics	186
Selecting the Right Web Format	191
Choosing Web Optimization Settings	196
GIF and PNG-8 optimization	197
PNG-24 optimization	200
JPG optimization	200
Optimization Output Options.....	201

Book III: Building Web Sites205

Chapter 1: Adding Text, Images, and Links207

Setting Up Basic HTML.....	208
Adding the title, DOCTYPE, and metadata.....	208
Adding a page title	209
Adding a DOCTYPE	209
Adding metadata	211
Coding pages by hand	214
Coding Your Pages	217
Adding Page Content	219
Inserting text.....	219
Adding graphics.....	223
Creating Hyperlinks	229
Understanding local and global links	230
Linking targets	231
Linking graphics	234
Creating other link types	237
Labeling Content for CSS Markup	243
Making Content Accessible.....	245

Chapter 2: Organizing Content with Tables and Lists249

Inserting Tables on a Page	249
Discovering what you can do with tables	250
Understanding the structure of a table	251
Adding content to table cells.....	252
Formatting Tables	253
The id attribute.....	254
Table widths and heights	254
Table and cell alignment.....	256

Table borders	257
Cellpadding and cellspacing attributes	258
Table headers	260
The nowrap attribute	260
Splitting and merging table cells	260
Background and border colors	261
Tiling background images	263
Nesting tables	264
Inserting Lists on a Page	264
Examining the two list types	264
Nesting lists	266
Adding content and formatting a list	268

Chapter 3: Styling with Cascading Style Sheets 271

Understanding CSS Basics	271
Using CSS as a Web standard	272
Taking a look at the anatomy of a style	273
Exploring inline, internal, and external CSS	274
Linking external CSS to a page	277
Setting CSS media types	279
Linking CSS with Dreamweaver	284
Working with CSS Style Selectors	286
Applying custom class styles	286
Making CSS tag redefine styles	288
Creating ID styles	290
Building compound styles	292
Creating a master CSS file	294

Chapter 4: Understanding CSS Style Properties 301

Working with the CSS Box Model	301
Exploring the Eight Style Property Categories	304
The type properties	305
The background properties	307
The block properties	310
The box properties	313
The border properties	314
The list properties	315
The positioning properties	317
The extension properties	321
Styling the Content on Your Pages	322
Styling paragraphs, headers, and footers	322
Styling lists and tables	324
Styling images and AP elements (layers)	327
Finding CSS Resources Online	329

Chapter 5: Creating Web Layouts	333
Creating Standards-Compliant, Accessible Layouts	333
Working with Layers	335
Discovering the benefits of layers-based layouts	336
Understanding what layers are.....	337
Creating a Layers-Only Layout	338
Adding a layer to a page	338
Building a CSS layers-based layout	339
Styling a CSS layers-based layout.....	344
Building an Old-School HTML Tables-Based Layout for HTML E-Mail and Newsletters.....	349
Understanding the benefits of tables-based layouts	349
Building an HTML e-mail or newsletter	351
Finding Online Resources for Layers-Based Layouts	356
Chapter 6: Constructing Navigation Systems	359
Assessing the Navigational Needs of Your Site	360
Discovering the Basic Principles of Navigation Systems	361
Wide versus deep menus	362
Single-tier menus	363
Multitier menus	363
Choosing the Right Menu for Your Site	364
Creating Text Navigation Menus	366
Exploring your layout options	366
Creating a rollover text-based navigation bar	367
Creating Rollover Button Graphic Navigation Menus	371
Understanding how to build rollovers.....	372
Outputting rollovers in Fireworks	374
Creating rollovers in Dreamweaver	377
Creating Multitier Spry Menus in Dreamweaver	379
Creating CSS List Navigation Menus	383
Chapter 7: Designing Web Forms	391
Deciding What Visitor Information to Collect	392
Encrypting and Processing Collected Form Data	394
Deciding whether to purchase an SSL digital security certificate	394
Understanding how data encryption works	396
Understanding the Structure of Web Forms	400
Creating a Web Form.....	401
Creating the structure of the form	401
Adding individual form fields	406

Validating Web Forms	412
Understanding what a validating form is	412
Adding a Validate Form behavior to a form	413
Building Spry Web Forms in Dreamweaver	416
Taking a look at the Spry validation widgets	416
Adding Spry validation fields to a form.....	417
Testing Validated Web Forms	419

Chapter 8: Making Your Pages Interactive 423

Getting to Know JavaScript.....	424
Creating Multipart Rollover Effects	426
Launching a New Browser Window	432
Deciding when to launch a new browser window.....	432
Hand-coding the script to launch a pop-up window	434
Adding a pop-up window to your page with Dreamweaver.....	436
Building Image Maps.....	439
Adding an image map to a graphic.....	440
Building complex image maps	441
Adding Multimedia Files.....	444
Adding a multimedia file to your page.....	444
Creating slide shows	447
Adding sound with Dreamweaver	450
Providing Daily Interactive Content on Your Pages.....	452
Daily tip or news item.....	452
Daily word game	457
Daily blog entries.....	457

Chapter 9: Building Web Sites 459

Building the Master Page	460
Building Web Sites with Templates.....	460
Using Dreamweaver templates	461
Preparing a page to become a template	462
Creating a Dreamweaver template	464
Creating templates with editable regions	465
Creating and editing template-based files.....	467
Working with Server-Side Includes (SSIs)	469
Understanding what SSIs are	469
Including an SSI file inside a page	469
Editing an SSI file	471
Ensuring success with SSIs	471
Creating, Including, and Testing SSIs.....	472

Editing Paths to Work with SSIs	476
Understanding the different path types	476
Adjusting paths in an SSI file from document relative to site-root relative	478
Comparing Templates and SSIs	479

Book IV: Web Standards and Testing481

Chapter 1: Following Web Standards483

Working with Web Standards.....	484
Understanding the importance of writing standards-compliant code	484
Taking a look at W3C recommendations	485
Exploring the W3C Web site.....	486
Using DOCTYPEs (DTDs)	489
Selecting a DOCTYPE	489
Adding a DOCTYPE in Dreamweaver	493
Writing Semantic HTML and XHTML Code	495
Formatting with CSS Instead of HTML.....	498
Comparing CSS and HTML formatting	499
Taking a look at the benefits of CSS	500
Exploring pages styled with CSS	501
Finding Out about Accessibility Standards	503

Chapter 2: Testing, Accessibility, Compliance, and Validation ... 509

Understanding the Process of Validating Your Code.....	510
Performing Prelaunch Testing.....	510
Creating a Web-testing checklist	511
Testing on multiple platforms, browsers, and devices	512
Cleaning Up Your Code.....	515
Finding and replacing errors.....	516
Checking spelling	517
Removing unwanted formatting	518
Applying consistent (X)HTML syntax.....	520
Applying source formatting	521
Converting syntax by DTD	521
Fixing Common Code Errors.....	524
Validating your markup	526
Checking browser compatibility	527
Verifying internal and external links	529
Generating site reports.....	530

Validating HTML and CSS Markup.....	531
Using free online validation tools.....	532
Fixing noncompliant code.....	536
Retesting and failing acceptably	538
Obtaining proof of validation.....	540

Book V: Publishing and Site Maintenance543

Chapter 1: Domain Registration and Hosting 545

Understanding How to Get Your Site Online	545
Selecting a Domain Name.....	546
Understanding what a domain name is	547
Finding a domain name for your client.....	549
Using domain name generators.....	549
Checking domain name availability	550
Registering a Domain Name.....	553
Using a domain registrar	553
Using a host provider.....	554
Activating your domain	555
Finding the Best Hosting Plan.....	555
Researching host providers	555
Evaluating hosting plan packages	557
Creating a Custom Placeholder Page.....	560
Designing a placeholder page.....	562
Uploading a placeholder page	568

Chapter 2: Publishing Your Site 571

Uploading Files with File Transfer Protocol	572
Choosing the right FTP program	572
Setting up a remote connection	575
Setting Up a Test Directory.....	579
Getting and Putting Files	581
Putting files on the remote server.....	582
Transferring files with Dreamweaver.....	582
Performing Final Site Testing.....	585
Creating Custom 401 and 404 Error Pages.....	587
Creating the error pages.....	588
Editing the .htaccess file	590
Publishing Your Site.....	591

Chapter 3: Search Engine Optimization and Site Maintenance . . . 593

Understanding Search Engine Optimization.....	594
Practicing Ethical SEO Techniques.....	595
Optimizing Your Site for Search Engines.....	597
Maximizing keywords	598
Including descriptive text and hyperlinks	600
Embedding object and image descriptions.....	600
Adding keyword and description meta tags	601
Updating bland page titles	603
Submitting a Site to Search Engines	605
Hand-submitting the URL	605
Waiting for the site to be listed.....	608
Giving Your Site an HTML Site Map	608
Deciding what to include on the HTML Site Map page.....	609
Creating a Site Map page	611
Making the site map accessible.....	613
Keeping the Site Relevant	616
Performing site maintenance.....	616
Scheduling site updates.....	617
Adding new content regularly	617
Moving on.....	619

<i>Index</i>	621
--------------------	------------

Introduction

Welcome to *Web Design All-in-One For Dummies*. This desk reference book is ideal for both the Web entrepreneur looking to design her own site and the new Web designer who plans to make a career of this exciting profession. This book uses many Adobe products to demonstrate common Web design techniques. Specifically, all of the graphic examples are done exclusively in Photoshop, and all of the Web page-building examples are done in Dreamweaver. That said, many other software programs are mentioned and recommended throughout the book, and the examples are easily adaptable to your preferred software tools.

Web design is a really unique occupation because it combines the best parts of visual creativity with modern technology. A Web designer, in essence, is a graphic designer, a creative organizer, a visual communicator, a markup language technologist, and a cutting-edge trendsetter. What sets Web design apart from other careers is that as the designer, you will play a key role in helping businesses connect with their customers in positive and meaningful ways. A good design can help attract the right target audience, sell more products and services, communicate new ideas, and change people's lives.

As a Web designer, you have the opportunity to put your visual and organizational spin on the world, taking the complex puzzle of each Web project and turning it into a visually pleasing, easy-to-navigate Web solution for your client. What's more, you will become inextricably part of the worldwide network of Web professionals who help shape the visual realm of communication in the twenty-first century.

Whether you've designed a site before or you are brand new to the world of Web design, this book takes you through all the steps of the Web design process. By the final chapter in the last minibook, you will have all the skills you need to design, build, and publish your own Web sites.

About This Book

As a reference book, you have the luxury of reading this book any way you like. You don't have to remember anything you read because the answer you are seeking will always be at your fingertips. Feel free to jump around from chapter to chapter, reading particular sections of the book as the needs arise, or go ahead and read this book from cover to cover like a sort of how-to manual to understand the craft of Web design. The book itself is divided into five minibooks, which are each divided into several self-contained chapters on a variety of particular topics.

Everything you find in this book is written simply and straightforwardly so that you can get right to the task at hand instead of having to wade through complicated technical details. When there is something of note, like the introduction of a new term, a special tip, or some geeky technical information that I think you should know, I let you know by putting an icon in the margin so that you can choose whether to read or ignore that material. Other than that, you find detailed, step-by-step instructions and easy-to-understand descriptions of the topics at hand.

Above all, this book is written to help make you comfortable with all the aspects that relate to the process of Web design. It is my sincere hope that you will use this book frequently and consider it the main go-to resource of your Web-design library.

Foolish Assumptions

This book presumes that while you may have some technical experience using computers and accessing the Internet, you might also be a newcomer to the field of Web design and the relevant ideas presented here. It is further presumed that you are a hobbyist, a do-it-yourself entrepreneur, or a person looking to become a Web professional, and that you are seeking a professional-level understanding of Web design from an experienced Web designer and software instructor. That's exactly what you'll get.

Creating Web sites, as you will soon discover, is an extremely enjoyable, challenging, and rewarding process because you can control (or help to influence, when working for someone else) which content will be displayed on the site, how it will all be organized, what the site will look like, and how the site will function. You get to engage your creativity, your knack for organization, your ability to visualize, and your artistic sensibilities all at once. Plus, if you have a flair for learning about technology, you can soon impress your friends with your vast Web vocabulary and your understanding of how Web sites work behind the scenes. Best of all, when you build a Web site, you have the unique opportunity to effectively communicate your (or your client's) ideas with the world in one of the coolest mediums available.

Conventions Used in This Book

To help you understand all the new terms and concepts that relate to Web design (and you'll find lots of them!), the following typographical rules or *conventions* are used in this book:

- ✓ **New terms:** New terms are set apart with italics. For example:

Your *meta tags* are the special lines of HTML code that you add to your Web page between the opening and closing `<head>` tags to communicate important information about the site to Web browsers.

- ✓ **Reader entry:** For times when you are instructed to enter your own content to replace sample content, those parts are listed in bold, as in

```

```

- ✓ **Code examples:** The HTML, CSS, and JavaScript code examples in this book either are listed in monospaced text within a paragraph, like this:
``, or set apart from the text, like this:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01  
    Transitional//EN" "http://www.w3.org/TR/html4/  
    loose.dtd">  
<html>  
<head>  
<meta http-equiv="Content-Type" content="text/html;  
    charset=utf-8" />  
<title>Untitled Document</title>  
</head>  
<body>  
</body>  
</html>
```

How This Book Is Organized

By design, this book enables you to get as much (or as little) information as you need at any particular moment. Need to know something fast about how to mock up a Web design before you build a site? Read the part of Book II, Chapter 2 that applies. Whenever some new question about Web design comes up, *Web Design All-in-One For Dummies* is a reference that you reach for again and again.

This book is divided into five minibooks, each of which is further divided into relevant chapters about the process of Web design, organized by topic. If you're looking for information on a specific topic, check the headings in the Contents at a Glance or skim the Table of Contents. In the following sections, you find an overview of what each minibook contains.

Book 1: Getting Started

This minibook covers all the behind-the-scenes work you need to do before you design and build a Web site. Topics include planning your site goals, building a site image, defining the target audience, building wireframes,

gathering and creating content, building a site map, and choosing the right tools for the job, including a Web editor to build your pages, a graphics program to design and optimize your Web graphics, shopping cart software (when applicable), and knowing when to hire others to assist you. By the end of this minibook, you will understand how to create a site that best projects the site's identity and attracts the ideal site visitors.

Book II: Designing for the Web

Designing for the Web is a special task that blends the visual with the technological, because your designs must conform to the rules of HTML, CSS, JavaScript, and other markup and programming guidelines. If you're looking for information about defining a site's look and feel, choosing the right layout and navigation scheme, mocking up a design, and optimizing graphics for the Web, you definitely want to read this minibook.

Book III: Building Web Sites

This minibook is all about Web site construction. Using your preferred HTML editor, here you find out how to set up a basic Web page; add text, images, hyperlinks, and multimedia files; work with semantic HTML; organize your content with tables and lists; style your pages with CSS; create layers-based layouts; build customized navigation systems; build, validate, and test Web forms; and work with templates and Server-Side Includes to build smarter, more efficient Web sites.

Book IV: Web Standards and Testing

After you've built a site, it isn't necessarily ready for publishing. Before you put your finished work online, spend some time reading the chapters in this minibook to find out about working with Web standards and making your pages accessible to the widest possible audience. You also find information here about using the proper DOCTYPE, writing semantic code, testing on multiple platforms in multiple browsers and devices, validating your code to ensure that your markup meets those Web standards, and resolving any issues that may come up during testing before you share your work with the world.

Book V: Publishing and Site Maintenance

After your site is fully built and tested, you will be ready to publish it on the Internet. This minibook teaches you about choosing and registering a domain name for your site, setting up a hosting plan, creating a custom placeholder page for your site, and publishing your site to your host server using FTP (File Transfer Protocol). Postlaunch, you may also need to make

further changes and enhancements to your site. Here you find an entire chapter devoted to enhancing your site with Search Engine Optimization techniques, performing routine site maintenance, and finding out ways to keep your site up to date so that visitors will be more likely to return to it again and again.

Icons Used in This Book

To make your experience with the book easier, you'll find the following icons in the margins to indicate particular points of interest.



Tip icons alert you to interesting techniques and hints that can save you time and effort when planning, designing, building, and publishing your Web sites.



This icon is a friendly reminder or a marker for things to keep in mind when performing certain tasks. It is also used to alert you to important facts, principles, and ideas that can help you become a better Web designer.



Watch out! This icon is the equivalent of an exclamation point. Warnings are placed next to information that can help you avoid making common mistakes. They also give you important directions to help keep you from experiencing any Web design nightmares.



Throughout the chapters, you will see this icon next to particularly technical information. While this kind of geek-talk will be interesting to some, it's not essential reading for everyone. That said, please do consider at least glancing at the text marked with the Technical Stuff icon just in case it applies to your situation.



This icon alerts you to examples that utilize sample files. You can download these files at any time from www.dummies.com/go/webdesignaio.

Where to Go from Here

While this book is written so that more experienced Web designers can skip around to the parts they need, novice users probably need to start with Book I, which gives a good foundation of building Web sites, before proceeding to the other minibooks. If you're one of those experienced designers, scour the Index for the material you need and then read those sections.

Read through the Table of Contents to find what interests you. Otherwise, consider the following jumping-off topics:

- ✔ To find out about site planning, check out Book I, Chapter 1.
- ✔ For tips on choosing the right Web editor and graphics software programs, see Book I, Chapter 4.
- ✔ For help in creating a mock-up of your Web page, see Book II, Chapter 2.
- ✔ To find out about optimizing graphics for the Web, see Book II, Chapter 3.
- ✔ For information about adding text, graphics, and links to your pages, read Book III, Chapter 1.
- ✔ To discover everything you want to know about working with Cascading Style Sheets, look at Book III, Chapter 3.
- ✔ If you want to know more about creating a layers-based layout and building a navigation system, see Book III, Chapters 4 and 5.
- ✔ To find out about Web forms, see Book III, Chapter 7.
- ✔ To get help with testing and validation, see Book IV.
- ✔ For information on publishing your site, see Book V.

Book I

Getting Started

The 5th Wave

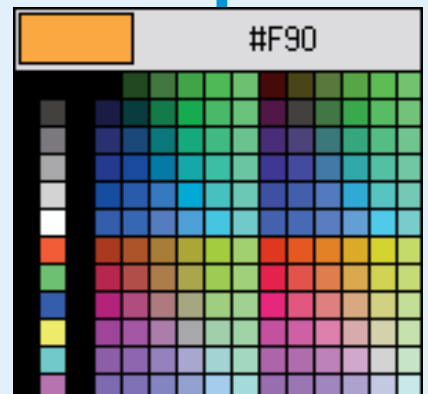
By Rich Tennant



“As a Web site designer I never thought I’d say this, but I don’t think your site has enough bells and whistles.”

While the best-laid plans might often go awry, for Web sites they don't necessarily have to if you do your homework. A good Web plan includes figuring out the site's purpose, building a site image, defining the target audience, carefully planning the content requirements, and choosing the right tools for the job.

In this minibook, you discover how easy it is to properly get started with any new Web project. Armed with the information you find in these chapters, you'll be totally ready to design and build your site.



Chapter 1: Starting with a Plan

In This Chapter

- ✓ Determining the site's purpose
- ✓ Building a site image
- ✓ Determining site content
- ✓ Diagnosing the site's dynamic requirements
- ✓ Defining ways to attract visitors

At the start of any Web site project, you — and your client, if you're designing for someone else — should probably sit down and mull over or discuss your ideas about the final product. If those ideas are vague, you need to flesh them out, and if they're specific, you need to keep them organized and understand the purpose behind them. In other words, you need a plan.

To get the project off to a good start and ensure that each of your ideas and issues get the consideration they deserve, begin by pinpointing the purpose for building the Web site. The purpose is like an arrow that points you in the right direction. Some people build Web sites to sell a product or service. Others create sites to share ideas and information. One might also create a site to promote a good cause, educate the public, or simply have a place for family and friends across the country — or across the planet — to visit and catch up with one another.

Because the answer to “Why build a site?” is largely determined by the specific needs of the Web site owner, this chapter includes a handful of brainstorming exercises that you can use as a guide to discovering why you are building any particular site. By defining the site's purpose, you develop a foundation for the rest of the site-planning process. By the end of this chapter, you should have all the tools you need to establish a plan for building almost any Web site.



Understanding the Different Phases of a Web Project

Before you begin any Web site, you must first have a good understanding of the project as a whole, as well as all the different steps or phases that you will move through during the Web-development process.

Most Web site projects have a logical flow of development, a type of evolution with distinct phases that, when followed, can streamline the entire design process. Here's the general order in which most Web site projects evolve:

- 1. Planning phase:** Define the goals and purpose of the site, construct a site identity, determine what content should go on the site, diagnose dynamic site requirements, if any, and figure out ways to attract visitors to the site after it gets published on the Web.
- 2. Contract phase:** Draft and submit a proposal to the client for the project that outlines the scope of the work in written form so that both the designer and client have a clear understanding of the expectations and outcome of the project, as well as financial agreements, time frame, and deliverables. Upon approval, the proposal gets converted into an official contract that both parties sign, and a deposit is paid to the designer to begin work.
- 3. Design phase:** Characterize a target audience; construct an identity for the ideal site visitor; gather information about the target audience's computer usage; determine the benefits to visitors; make decisions about layout, color, organization, and content; and finally mock up a design and present it to the client for approval.
- 4. Building phase:** Convert a mock-up into HTML, CSS, and JavaScript in a WYSIWYG Web editor such as Adobe Dreamweaver; organize content in visually pleasing ways; create and optimize Web graphics; add dynamic capabilities to the pages with JavaScript and other programming languages when warranted; and ensure that all the pages on the site look good and function well in a variety of browsers on both the Mac and PC as the pages are being built.
- 5. Testing phase:** Test the design on a testing server in the most popular browsers and browser versions on Mac, PC, and Linux platforms in the most popular operating systems (Windows XP, Vista, Mac OS X, and so on) at a variety of monitor resolutions; validate the code; check for spelling errors; fix coding errors; and otherwise ensure that each visitor can navigate through the site with no technical problems.
- 6. Site launch:** Secure a domain and hosting plan, upload the site's files to a host server, retest the site, and be ready to maintain the site postlaunch.

- 7. Postlaunch site maintenance:** Ensure that the site's content stays relevant and up to date by adding new and editing existing content, as well as making improvements and other enhancements to the site.

Determining the Site's Purpose

Before you begin any new Web project, it would greatly behoove you to first develop a plan. With a plan, you will know in advance what you're going to do and the order in which everything needs to be done. Furthermore, when designing sites for others, a plan can help keep both you and your client on the right track.

If you've never created a Web site plan before and aren't sure where to begin, the very first thing you should do — before you even start to think of designing the site — is to determine the ultimate purpose of the site. To do that, start by asking yourself a few simple questions and jotting down the answers:

- ✓ Why are you building this site?
- ✓ Will the site be professional, fun, silly, or informative?
- ✓ Will the site sell products, services, information, ideas, or some combination of these things?

If you think of any other questions that can help you determine the site's purpose, add those to the list. When finished, continue reading.

Checking out the competition

In the boom of the Internet revolution in the late 1990s, every big company with a brick-and-mortar store learned that having a Web site would instantly make its products and services available to millions of site visitors each day. New Web sites sprouted up daily as more and more people purchased computers, learned how to use them, and began searching, finding, and buying the products and services they needed online.

As the Internet continued growing in popularity, so did the idea that selling products and services exclusively online was a viable form of running a business, mainly because it entirely negates the need for costly store overhead, which in turn can increase profits.

Today, most businesses have their own Web site, or are in the process of creating one, or are in desperate need (but may not know it!) of having their current site redesigned and improved upon. People create Web sites to promote business services; sell products; share information; provide free resources; offer contests, coupons, tips, and advice; and more.

The bottom line here is that in this *age of the Internet*, anyone who owns a business, wants to stay competitive, and wants to be taken seriously by Web-savvy consumers needs to have a Web site. If you agree with this premise, you have to know what the competition is doing right now, both online and off.

Take a look at your competitors and make notes about what they're doing. What are they doing that works, and what are they doing that doesn't? Pay attention to color, graphics, format, layout, content, and the tone of the writing. This information can help you determine the type of content to go on your site and understand how to deliver it.

Gathering information

In your career as a Web designer, oftentimes your client will rely on you to assist with determining the site's purpose. If you or your Web client hasn't discussed this issue yet, read through the following questions and take careful note of your client's answers:

- ✓ **Will the site showcase biographies, histories, and other informational data?** Some Web sites like to show a listing of employees and board members, biographies, historical timelines, and general information about the company and its key players.
- ✓ **Will the site market services?** A company, group, or sole proprietor (such as a nonprofit arts organization, a law firm, or a marketing consultant) might want an informational or *brochureware* Web site to help spread the word about its services. How many services will be offered? Will pricing information be made available online too?
- ✓ **Will this site provide in-depth information about a particular topic?** The function of a political news blog or nonprofit organization is to share ideas and information with the public. For example, a lawn mower company might want to offer lawn-care advice in addition to selling mowers.
- ✓ **Will the site be someone's personal Web site?** Personal Web sites are just for family, friends, and schoolmates. It could be a digital family photo album, a blog, or an outlet for personal expression.
- ✓ **Will the site be someone's professional portfolio?** Professionals use portfolios to generate new business and showcase their talents. People who use portfolios include artists, illustrators, designers, writers, singers, photographers, musicians, poets, and academics.
- ✓ **Will the site sell any products, and if so, what kinds?** If the site will sell lots of products, find out how many product categories are needed and whether the products will be sold wholesale, retail, or both. Will the products be sold online or through an outside distributor?

Developing a statement of purpose

A statement of purpose, such as the example shown in Figure 1-1, is a brief summary of the goals for the site, including who the site is for, what the site hopes to accomplish, how it will look, and most importantly, how it will function. Think of the statement of purpose as a type of mission statement. Use the answers to the questions in the preceding section to begin forming a vision of how the site will look and function. For example, a realtor's Web site that markets rental properties and realty services will look and function much differently than a nonprofit site for railroad veterans or one that showcases a watercolor artist's portfolio.

After discussing these and other questions with the site owner, you can find out whether you need to design the site to attract business, share information, provide feedback and advice, be a blog with text and photographs, provide a dating service, sell moving and relocation services, provide online banking, supply wholesale products to retailers, or some combination of these and other things. After you have all that information for your site, you just need to take this information and boil it down into a statement of purpose. Table 1-1 gives examples of several types of businesses and statements of purpose that those business owners might come up with for their sites.

Statement of Purpose for a Graphic Design Studio
This site will be an online portfolio to showcase and promote the design services of CUSTOMER NAME to art directors, editors, publishing companies, and other people who regularly hire graphic designers for various projects including book jackets, brochures, letterhead, and creative packaging design. In addition to generating new business, the site will also promote CUSTOMER's service by showcasing industry-related awards, a biography, and a short history of the CUSTOMER's studio.

Figure 1-1: A good statement of purpose can help guide the Web design process.

Table 1-1 Creating a Statement of Purpose	
Type of Business	Example Statement of Purpose
Sole proprietor or entrepreneur (such as a business consultant, family therapist, or private detective)	This site will market services to a wider audience, lend a sense of legitimacy to the business, generate more clients, and allow customers to register for a monthly newsletter.
Creative entity (such as an artist, designer, illustrator, photographer, poet, actor, musician, or band)	This site will be an online portfolio/showcase for displaying and promoting work (art, music, photos) to art directors, editors, and other people in the industry. Additionally, it will help generate new business, share news and information, and sell a limited number of creative works.

continued

Table 1-1 (continued)

<i>Type of Business</i>	<i>Example Statement of Purpose</i>
Nonprofit organization	This site will promote services, provide industry-related information, educate the public, collect donations, offer public and private programs and events, list classifieds for members, and supply registration information for fund-raising events.
Personal/nonbusiness blog site (such as a blog that covers the local music or arts scene)	This blog site will report on the local scene, providing information about current and upcoming events, news, popular venues, and important people around town, as well as allow visitors to submit feedback and questions to the authors of the various posts. In addition, the blog will allow visitors to subscribe to an RSS feed, get updates by e-mail, and research topics of interest in the blog's archive.
Small- to medium-sized business (such as a greeting card company, a network backup hardware manufacturer, or an adventure tour company)	This site will be an online storefront to sell products and services, answer FAQs, have a library of information related to products and services, and allow visitors to contact the business, receive customer support via e-mail and live chat, and subscribe to a weekly newsletter.

Take a moment to think about the purpose(s) of your Web site project and record your answers on a sheet of paper or in a new word processing document using the format shown in Figure 1-2. (If you don't have a project in mind, pretend that you're planning a site for a marketing consultant who promotes art books so that you can practice generating ideas for a statement of purpose.) Whatever your answers happen to be, turn them into a statement of purpose that you can keep handy throughout Book I; the statement can help you organize your ideas and plan the best Web site for your needs.

Type of Business:
Purpose of site:
1.
2.
3.
4.
5.
6.
7.
8.
9.
10.

Figure 1-2: Write out a statement of purpose for each site you create.

Building a Site Image

The next important step to take with your project is to define and build the image that the Web site will project to the consumer. This image establishes the unspoken identity or personality of your Web site to visitors — an identity that they will (hopefully) respond to both intellectually and emotionally.

To help construct this identity, follow these steps:

- 1. Try thinking of the site as if it were your client's number one salesperson, someone who fully represents the best about the company.**
- 2. Come up with as many words as you can to describe this "person's" traits.**

Is the person professional or laid back, serious or fun, creative or traditional? If the salesperson angle is a bit awkward for you and/or your Web client, think about the ideal image you'd like the Web site to project and find adjectives that describe that ideal. Table 1-2 lists descriptive terms you can use to begin defining the Web site's image; it's by no means complete, but it should get you started.

Professional	Casual	Innovative	Creative
Traditional	Cutting-edge	Popular	Honest
Open	Fun	Witty	Intelligent
Smart	Open-minded	Supportive	Caring
Technological	Trend-setting	Urban	Cultured
Educated	Contemporary	Organized	Efficient
Cost-effective	Reliable	Trustworthy	Friendly
Talented	Confident	Capable	Established
Savvy	Respected	Clever	Solution-oriented

In addition to providing you with a strong and clear sense of what you're doing with this Web project, the identity you construct for it can help you make aesthetic and organizational decisions about the site, such as what colors and graphics to use and the best layout for the content. For example, if your Web client sells hockey equipment, you'll probably decide to use bolder masculine colors over pastels in the design. Or if your client is a medical consultant looking to advertise his services to hospitals and medical centers, you'll probably want to advise him to invest in some good royalty-free, industry-specific artwork for the site rather than display the

often-overused and amateurish-looking illustrations from the Microsoft Word Clip Art archive. You find out more about creating and licensing artwork for your site in Book 1, Chapter 3.

Right now, on a blank piece of paper or in a new word processing document, write out a list of at least ten adjectives that describe the company image for your current Web site project. These are the words that should automatically come to mind when a person visits the Web site for the first time.

Determining Site Content

By now, you should have a pretty good idea about the site you want to develop. You have identified the site's purpose and made the initial steps toward defining an identity for the site. Now you're ready to start thinking about what content needs to be presented on the Web site.

Though there is technically no such thing as an industry standard on the Web, logic should tell you that certain content should be on every Web site, regardless of the site's purpose. Beyond that, anything else that goes on the site is up to you — the designer — and your client.



The more informed you and your Web client are about the whole Web design process, the better the finished product. Even if you already know how a Web page should look or how it should function, being able to explain why can help you educate your client. The client, in turn, might also be able to give you more constructive input if he understands the *concepts* behind your design and the principles that drive content selection.

To assist you in figuring out what, at a minimum, should go on a Web site, the following general guidelines should help make your project more effective.

Minimum requirements

At a minimum, your Web project needs to supply basic information, so your job during this planning process is to decide what content you or your client will need.

The following information is commonly found in some variation on most Web sites.

Home page information

The home page is the first page on a Web site that visitors see when they type in your Web address, such as *www.yourwebsite.com*. In addition to setting the visual tone of your site through the use of graphics and Cascading Style Sheets, this page should include the company name and/or

logo, navigation to the rest of the site, and text describing the site's products or services. It is also the most important page on the site because this is where you introduce the site to visitors. For this and other reasons, the home page should contain at least a paragraph or two of descriptive, search-engine-friendly HTML text (not a graphic) that generally outlines what visitors can expect to find on the site. Whenever possible, any keywords (descriptive terms used to find information on a specific topic) in the text should be hyperlinked to other relevant pages on the site.



In the not-too-distant past, many sites used the home page as a place to play introductory flash animations, have one giant graphic with an Enter or similar hyperlink, or have a different set of graphics and layout than found on the rest of the Web site. Though these strategies may have contained a bit of the old “Wow!” factor, they never were a good idea, particularly because they lacked meaningful, searchable content (text) on the home page that can prevent the site from being fully indexed by the most popular search engines. More importantly, when visitors can't find what they're looking for by quickly scanning the home page, they leave. Therefore, make the most of the home page by including only relevant copy, links, and graphics, using the same layout found on the rest of the site. Consistency is key!

Contact information

Visitors will want to know how to get in touch with the owners of the Web site. Be ready to provide the physical address of the company, the mailing address (if different), telephone and fax numbers, and at least one contact e-mail address. You may also want to include special contact information for various employees, departments, and services, as well as local geographical area maps, transportation directions, and hours of operation. Some sites even provide a form on the contact page where visitors can submit personal information, answer survey questions, provide comments and feedback, and/or request information. If you plan to collect data from visitors on the contact page, make a list of the data you intend to collect so that you can have it handy when you build the page with the form.

Privacy policy

If you intend to collect any personal information (e-mail address, name, telephone number, and so on) from site visitors on a form, during registration, or for purposes of responding to an inquiry, the site would benefit greatly from including some kind of privacy policy that explains to visitors why their data is being taken and what the site will or will not do with that private data.



In the most general terms, a privacy policy should state how the company will care for the collected data, including any *cookies* collected from the computer used to visit the site. (Cookies contain personal data, such as name, address, phone, username, password, IP address, shopping cart contents, and so on, collected by a visited site's server and saved to the

visitor's computer so that future visits to that site will run more efficiently.) For example, if the company will share the data with or sell the data to other vendors, you need to state that expressly. Conversely, if the company plans to honor the privacy of visitors and closely guard collected information as if it were a priceless gift, state that clearly. For a clear example of a simple privacy policy, see www.bbbonline.org/Privacy/sample_privacy.asp. Alternately, FindLegalForms.com has a generic policy (Privacy Policy Agreement #28152) that you can purchase online for only \$14.95 (at the time of this writing). Or, if you want to generate a policy to match your specific business, you can use the Policy Wizard at the PrivacyAffiliates.com Web site for just \$19.95.

Site map

A site map is the often-forgotten Web page that contains a list of organized text links to all the pages on the Web site. If you want your site to be accessible to as many visitors as possible (including visitors with disabilities using assistive devices), regardless of how simple or complex the site is, include a site map page. Alternately, if your site has a lot of pages with multiple categories, consider adding key site map information to the footer area of every page.

Footer

At the bottom of every page of a site, you should include the company name, copyright information, and a series of what I call *footer links* or navigation links to the most important pages on the site. At a minimum, include links to such pages as Home, About, Services, Contact, and Privacy Policy. This information not only reminds visitors whose site they're visiting but also provides additional ways for visitors to navigate to other pages on the site.

To really harness the full power of this often-overlooked Web real estate, treat this area like a mini site map and list links to not only high-level navigation destinations but also to more detailed subnavigation category pages. Figure 1-3 shows an example of a site that includes all these basics. To find out more about how to make your sites accessible, see Book IV, Chapters 1 and 2.

Marketing and sales content

Whether the Web site you're designing is for yourself, a sole proprietor, an entrepreneur, a nonprofit organization, or a small- to medium-sized business, the rest of the site content should be geared toward promoting the business and attracting new customers from the pool of site visitors. Therefore, be sure to provide ample information about the company, organization, person, or entity and all the skills, talents, materials, services, and/or products available, plus anything else you can think of that can benefit the visitor and positively impact business.



Figure 1-3: Smart Web sites include footer links.



Developing an enthusiastic awareness of the Web site's online and offline competition can greatly assist you in making decisions about what information should (and should not) be included on a Web site. For example, if you're designing a site for a children's gymnastics program, the site should probably include the necessary information that can help visitors choose to enroll their children as students, such as a schedule, photographs of the facilities and smiling children, a teaching philosophy, student and parent testimonials, and perhaps a price list. If the school's offline competitors happen to give students free tote bags, perhaps your client's gymnastics school should offer students free T-shirts and balloons that include the company logo and Web address.

The following list of Web site sections is not meant to be comprehensive; rather, you may use it as a starting point for brainstorming about the particular content needed for each specific site.

Company information

This part of the site, usually called About Us or something to that effect, typically consists of either one page of company-related information or several pages of logically organized company details. The information here should describe the company to the visitor and include some form of the company's mission statement. In addition, this section might include a corporate history and philosophy statement, a directory of employees with bios of the management team, and/or information about company internships and careers.

Biography

Similar in scope to the Company Information section, the biography page (called About Me, Bio, or Biography) usually includes historical and other interesting information about the artist, sole proprietor, educator, or small-business owner. This page, or series of pages, should provide information to stimulate interest in the services, skills, work, products, and so on being presented on the Web site. This might also be a good place to include a résumé or curriculum vitae (an academic's work history and accomplishments).

Product/service information

Every product and service offered on a Web site should have its own detailed description. If the business is service oriented, describe what the business does, who needs this service, and how long the business has been operating. If the business sells products, the products need to be organized into logical categories and subcategories, such as Electronics↔Digital Cameras↔Nikon Digital Cameras.

In addition to a description for each main category, every individual product deserves its own description, including any information that might be interesting or necessary to purchasers, such as size, dimensions, color, weight, materials, ingredients, nutritional information, care instructions, technical specifications, country of manufacture, and warranty information. Whenever possible, also try to offer client/customer testimonials.



Keep in mind that for any copyrighted material you intend to use on the site — including intellectual property, photographs, and illustrations — you must have permission to use it. This means paying royalty fees for rights-managed work, requesting and receiving written permission for non-rights-managed work, and otherwise obtaining the right to use and display the work created by another person or entity. To find out more about copyrights and permissions, see Book I, Chapter 3.

News and press information

This section of the site typically contains current and recent press releases or newsletters, a press release/newsletter archive, articles about the business or industry, and/or any news items in the form of media coverage. This area might also include information about upcoming programs, trade shows and exhibitions, gifts and collections, relevant technology, works in progress, a historical corporate timeline, an image gallery or media library, and a listing of literary publications.

Video and podcasts

With the advent of YouTube, it's never been easier to insert video into your Web site. Video clips can be used to sell products, promote services,

showcase ideas, offer news and information, and even provide training and tutorials to interested visitors. In addition to adding video directly to the individual pages of your Web sites, many blogs now include plug-ins that allow you to insert video clips directly into your blog posts. You find out how to add multimedia files to your Web pages in Book III, Chapter 8.

By contrast, if you're looking to do a little more than post a video on your site, you might be interested in *podcasting*. A podcast is a way of transmitting any combination of audio, images, and video media to subscribers via syndicated download through Web feeds to computers and handheld devices like iPods and Blackberries. To find out more about podcasting and how to create your own podcasts, visit <http://blog.podcast.com/podcastcom-faq/> and www.how-to-podcast-tutorial.com. Or check out *Podcasting For Dummies*, 2nd Edition, by Tee Morris, Chuck Tomasi, and Evo Terra (Wiley).

Portfolio

When the site belongs to an artist, designer, or other creative professional, this area displays an online version of a portfolio, including photos and graphic examples of their work, a résumé or curriculum vitae, video clips, sound files (MP3s), and other types of media files. The online portfolio is fast becoming the best way to market services to a global audience, generate new business, and share news and industry information with the public.

Frequently Asked Questions (FAQs)

Most visitors have questions — lots of questions — and those questions need answers. Plus, by having both questions and answers online, visitors can often find what they need without having to spend extra time making a phone call or sending an e-mail — a big plus in our fast-paced world. Most FAQ pages cover information about contacting the site, searching for information on the site, customizing site preferences or membership accounts, getting more information, using the site, and accessing customer service. Alternately, the FAQ area may contain answers to common questions about products, services, and business-specific information.



If you don't have a list of information to create a FAQs page yet, start keeping track of questions the business gets asked. When a pattern begins to emerge, add those questions and answers to the FAQs page.

Site search

Though not required, providing a means for searching an entire Web site's content with keywords can improve the site's *stickiness* (the ability of a site to entice people to stay on the site longer). The most popular free search tool is Google Site Search. Get the code from Google at www.google.com/sitesearch. Google also offers a custom search engine tool at

22 *Determining Site Content*

www.google.com/coop/cse that allows site owners to customize the look of the search box as well as host the results on the same site. Two other free site-search tools worth looking at include Bravenet's Site Search (www.bravenet.com/webtools/search2) and FusionBot's Free Package (www.fusionbot.com).

Of course, as an alternative to these types of site-search tools, you can hire a PHP programmer to build a custom server-based search tool, complete with a search results page, using something like IBM's free and open-source Sphinx search engine. For further information, see www.ibm.com/developerworks/library/os-php-sphinxsearch.

Terms of service

Similar in importance to the Privacy Policy, the Terms of Service page should state how the site provides services to — and the conditions under which those services must be accepted by — visitors. This may include concepts of intellectual property rights, usage, registration, security, payment, advertising, applicable law, legal compliance, indemnification, and more. Because the Terms of Service should contain legal content specific to the Web site's offerings, the best way to create the page is to consult with a lawyer. Do-it-yourselfers can download a generic Web site Terms of Use Agreement from FindLegalForms.com for only \$14.95.

Shopping cart

Depending on your needs, several kinds of Web shopping carts are available for you to choose from. The most basic is a cart that uses PayPal to process payments. This option requires no merchant account, special software, or secure server licensing fees; however, it does require all purchasers to create their own PayPal account before their transaction can be processed. Another option is to create an online store through Yahoo! Shops, which uses Yahoo!'s proprietary shopping cart system. This is pricey and doesn't have the easiest interface to use, but it has the benefit of automatically being listed in Yahoo!'s shopping directory.

For more customized solutions that include some kind of inventory management system, you'll want something that's tailored specifically to your site's needs. With a simple search, you can find online shopping carts that are free and customizable, carts that are controlled by host providers, and carts that are powered by third-party software manufacturers.



In a 2009 Shopping Cart Software Report on TopTenReviews.com (<http://shopping-cart-review.toptenreviews.com>), the top-rated shopping cart software programs on the market included Shopsite Pro, Merchandizer Pro, and NetworkSolutions.

Whatever shopping cart software solution you decide to use, you must take extra care to ensure that visitors' personal information is safe and secure during the purchasing transaction. If your Web site will process credit card payments (instead of processing them through an outside service like PayPal), you'll need to set up a special merchant account with a qualified bank, as well as purchase an SSL (Secure Sockets Layer) digital security certificate for your domain. Your host provider should be able to assist you in both finding a bank to set up your merchant account and licensing and installing the SSL certificate. You can find more information about merchant accounts, SSL certificates, and working with the different types of e-commerce shopping carts in Book I, Chapter 4.

Customer service (Help)

Any site that plans to sell products or services must have a place for visitors to go to get more information about customer service, including how to contact you, ask questions, and resolve problems. Try not to think of this section of the site as a liability but rather as an opportunity to get to know and serve your customers better. Look to other successful Web sites to gather ideas on how to set up this valuable area of your site. Consider having sections for ordering information, privacy issues, shipping and delivery, dealing with returns or damaged items, and accessing account information, just to name a few.



The easier you make it for visitors to get answers, the more positive their experience on your site, and the better the customer satisfaction, the more likely that can quickly translate into free word-of-mouth advertising and repeat customers.

Site credits

Though this by no means needs to be included on a Web site, why not toot the horn of the designer or design team (you!) that turned a site concept into a Web reality? If you've included a clause in your client contract to do so, add a Site Credit link on the site, preferably embedded somewhere in the footer links. Otherwise, ask your client for permission to include the link. The Site Credit link itself can go directly to the Web site of the designer or open a page similar in layout to the rest of the site with designer contact information.

XHTML, HTML, CSS, and 508 compliance information

If the site has been built to be accessible to any and all Web visitors, consider proudly displaying compliance information somewhere on the site, such as in the footer, on the contact page, or in some other logical section of the site. You can find out more about how to adhere to the accessibility guidelines set by online Web standards organizations in Book IV, Chapter 1, which is entirely devoted to working with Web standards.

RSS feeds

Sites that include blogs may want to consider syndication of the blog content through an RSS feed, which can be used to automatically notify registered readers (by e-mail, Web portal, and news readers) of new posts and information. There are some really good RSS feed services such as FeedBurner.com and ListGarden on softwaregarden.com, as well as products you can use yourself, like the RSS Feed Creator application from SourceForge.net. To find out more about RSS, check out Google's help pages at www.google.com/support/feedburner/.

Diagnosing the Site's Dynamic Requirements

A dynamic Web site refers to any site that uses a programming language — such as PHP, ASP, ASP.NET, JSP, CGI, Perl, Oracle, Java, Ajax, or ColdFusion — to gather specific records of information from a database, such as Microsoft Access or MySQL, and displays that data on a Web page. A *database* is any collection of information, such as a spreadsheet, that organizes the data into categories that can be easily retrieved by a computer program or programming language on a Web site.

Many sole proprietors, small businesses, and nonprofit organizations with limited-content sites might have little (if any) need to offer a Web site with dynamic capabilities. Having dynamic content on a Web site largely depends on the goals and budget of the site's owner.

For sites with lots of content, a dynamic site with a database should be a serious consideration. By organizing and storing data in a database, the content can be selectively pulled according to different scenarios or rules set up in advance. For instance, one business might want to display the ten most recent news items about the company on a page, or perhaps a site wants to post a calendar (like the one called ConnectDaily from MHSoftware.com, as shown in Figure 1-4) so that visitors can view upcoming events. Presuming that new data is regularly being entered into the database, the programming language can be set to check article publication dates and always pull and display the ten most recent files on a particular page.

Databases should almost always be used if you are selling products on your site; however, you can also use databases to store and retrieve all kinds of information. For instance, you might decide to use a database on your (or your client's) Web site to display the following:

- ✓ Articles, papers, and documents sorted by date, author, and so on
- ✓ Lists of services and service detail information
- ✓ Calendar of events, schedules, contact information, and important dates

- ✓ Categories of products for sale and product detail information
- ✓ Customer data such as a purchase log, order records, and account information
- ✓ Customer membership information or saved shopping cart details
- ✓ Store locations, hours of operation, and contact information
- ✓ Tracking information for uploads and downloads to and from the site
- ✓ A glossary of industry-related terms or FAQs



Figure 1-4: A calendar can dynamically display events, reminders, and other important information for visitors.

In addition to dynamically accessing and using data, databases can be used to assist with adding, deleting, and editing content on a Web site. For an added fee, many programmers and host providers can now build a custom Content Management System (CMS) for a site, which allows site owners to easily manage specific parts of the site's content — without having to know any programming languages or HTML — through a customized Web interface. Depending on the size of the project and the complexity of the dynamic needs, a CMS Web site component can cost from as little as \$1,000 to as much as \$15,000 or more. This type of cost-effective tool can be extremely useful for sites that require frequent updates.

Though admittedly slick and cool, not every site needs to use a database. To determine whether the site you are building needs to use one, take a good look at the type of content you intend to display and ask yourself (or your client) these questions:

- ✓ **How often will the content need updating?** Sites with daily and weekly update requirements might benefit from a database, whereas sites requiring less frequent modifications might be better off without the added expense. Nondatabase site updates can be performed by the designer who built the site (you!) or by the client using simple Web-editing software like Adobe Contribute.
- ✓ **Are more than 20 products or services being sold?** If the site is selling only a handful of products, though time consuming, each product can have its own Web page. However, if more than 20 products will be sold or if the client anticipates increasing its product line to over 20 in the foreseeable future, using a database to dynamically create each product page is more efficient.
- ✓ **What kind of growth does the company expect to achieve in the next year, three years, or five years?** For some sites, little to no anticipated growth will be expected, and therefore you have no real cost justification for using dynamic features. On the other hand, sites that project to grow their products and services over the coming months and years might greatly benefit from building a site that can accommodate such growth.
- ✓ **Does the company need to collect and use visitor data?** E-commerce sites, like the Dummies.com Web site shown in Figure 1-5, have good reason to collect data from purchasers, to both streamline the ordering process and provide future sale and promotional information. By contrast, a small business could just as easily manage that information by using a simple HTML form and an Excel file.
- ✓ **Is there or will there soon be enough dynamic content — such as a listing of store locations or the ten most exciting daily news articles — to justify the added cost of making the site dynamic?** If you have the budget and foresee a need for dynamic content, setting up a data-driven site from the start can be more cost effective than adding one to a static

site down the road. Certainly the old adage “to make money, you need to spend money” pertains, but not everyone can afford to spend the money up front, even when he or she wants to.

Ultimately, the decision about whether to create a dynamic site should be fairly clear after answering these types of questions. If you’re still unsure about whether to use a database, get quotes from programmers or hosting companies to see how it will impact the budget for your project. Money can often be the great decider.

DUMMIES.COM The Online Resource for the Rest of Us!

Billing Information

First Name:

Last Name:

E-mail Address: (e.g. sam@company.com)

E-mail Address(again):

A verification will be sent via e-mail when your order is received and also when your order is shipped.
 I would like to receive special notices from Wiley.
 I can unsubscribe at any time, and my e-mail address will never be sold or traded.

Address:

Address (line 2):

City:

State/Province:

Zip/Postal Code:

Country:

Phone:

Select a shipping destination: Ship my order to this address
 Ship my order to a different address

[Continue Secure Checkout](#)

[About Dummies](#) | [Register for eTips](#) | [Contact](#)

Copyright © 2008 by Wiley Publishing, Inc. All rights reserved. Please read our [Privacy Policy](#).

Figure 1-5: An e-commerce Web site collects data from its visitors.

Defining Ways to Attract Visitors

One of the best ways to figure out what will attract visitors to a particular site is to think about the site from the visitor’s perspective. When most people visit a Web site, they’re typically looking for *specific* information about a *particular* product or service, such as a 16.6-cubic-foot refrigerator or a Promaster 28–210mm f3.5–5.6 MF lens for a Nikon camera. Finding that information is important — presuming the products or services are the online company’s bread and butter.

To make those customers happy and keep them coming back, a Web site should also provide other relevant information that supports the product or service, such as the answers to frequently asked questions, company information, customer support, and contact information. Beyond this kind of expected customer service content, any other information on the site is strictly optional — unless, of course, the site owners want to drive more traffic to the site, which they should.



Statistically speaking, the more traffic a site gets, the greater the likelihood that some of those visitors will either want to purchase the products and/or services being offered, or feel confident telling someone they know about the site, which in turn can increase traffic!

Fortunately, you can use lots of nice techniques to increase visitor traffic that have nothing to do with the product or service being sold. For instance, you or your client might decide to start a newsletter that offers industry-related tips, free downloads, or coupons, or the site owner (or you if she hires you to do postlaunch site maintenance) might begin to post weekly articles on a variety of topics related to products or services. Other sites might post blogs, use polls, offer free calculator tools, serve up news items through an RSS feed, or even have frequent contests with fun prizes — all designed to attract and keep visitors coming back, day after day.

In the following sections, you get a chance to look at a few of these techniques in greater detail. As you compare them and decide which one(s) you might want to include in your plan, keep the site's purpose, benefit to visitors, and image at the forefront of your (and your client's) mind. These factors should help identify the best ways to make the site attractive.

E-newsletters

E-newsletters, whether sent weekly, monthly, quarterly, or sporadically, are a fantastic way to communicate regularly with customers through e-mail. In addition to keeping the company name, brand, products, and services in customers' minds when they read it, each issue creates a new opportunity to have a positive and meaningful exchange with site visitors, who either are, or soon could become, customers or clients.

If you or your client plan on having a newsletter, make sure that you set aside space on the Web site, preferably in the same location on every page of the site, for a form that visitors can fill out to sign up for the newsletter, as in the example shown in Figure 1-6.

Most e-newsletters are graphically formatted in HTML (but they might also be plain text, or you can offer both) and typically include the following:

- ✓ Some kind of topical news

- ✓ Sale offers
- ✓ Information about new products and services
- ✓ Upcoming events listings
- ✓ Links to articles or products online
- ✓ Company information, the date, instructions on how to subscribe to and unsubscribe from the newsletter, and a few Web site links



Figure 1-6: Savvy Web sites send newsletters that get readers to visit the site.



Giving readers the choice to subscribe and unsubscribe is an important part of *netiquette* and can help the site avoid looking like a spammer. With that in mind, I highly recommend that when sending e-newsletters, you take extra care to ensure that

- ✓ You ask permission of your site visitors to add their e-mail address to your customer list *before* sending them anything.
- ✓ You include a link to your site's privacy policy so that interested visitors can learn more about how you will use their e-mail address and other personal data.
- ✓ You include, in every mailing, a simple method for visitors to unsubscribe from your list.

See the nearby sidebar for more about the art and practice of netiquette.

For exceptional information about writing, designing, and sending out e-newsletters, visit the MailChimp Resource Center, shown in Figure 1-7, at www.mailchimp.com/resources and be sure to download a free copy of MailChimp’s Email Marketing Beginners Guide.

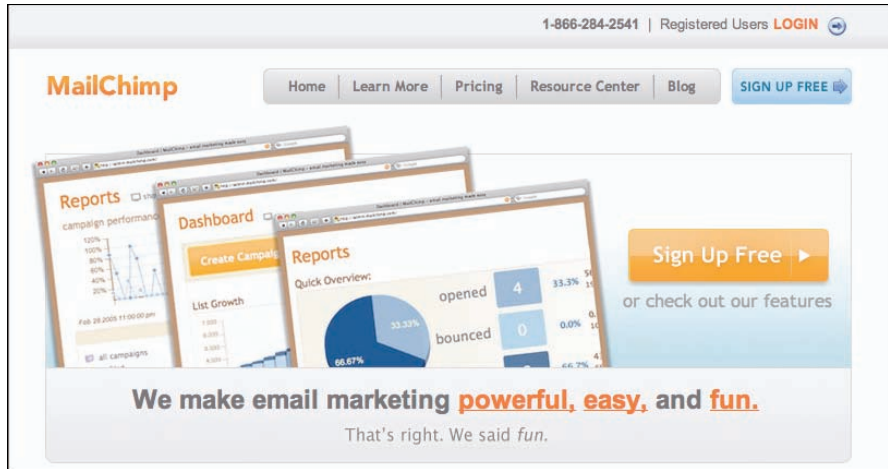


Figure 1-7: Discover free e-newsletter tips at the MailChimp Resource Center.

To send newsletters, you can choose from a variety of e-mail programs, though the best-supported applications are for PCs only and often only allow the sending of one e-mail to a maximum of 50 addresses at a time. A better solution, for both Mac and PC users, is to outsource the management of your e-mail list to one of several online services that can handle the job. Table 1-3 lists one mail program and two newsletter services that have great reputations.

Table 1-3 Third-Party E-Newsletter Services	
<i>Product</i>	<i>Web Site</i>
Direct Mail (PC & Mac)	www.ethreesoftware.com/directmail
MailChimp	www.mailchimp.com
Constant Contact	www.constantcontact.com

The importance of netiquette

Nowadays, when you purchase anything online, your e-mail address will probably be automatically added to the selling company's e-newsletter mailing list. If you enjoy learning more about towel sales, electronics equipment, or office supplies (for instance), seeing these e-mails in your inbox might be a pleasant surprise for you each time they arrive. But when unwanted newsletters arrive — especially when you didn't expressly authorize the enrollment to the e-mail list — these kinds of missives can seem downright spammy.

When sending e-mails and otherwise communicating over the Internet, do you always use

your best online manners? Network etiquette, or *netiquette*, is the set of unspoken rules that everyone online should follow, whether sending personal or professional messages. Think of it as the art of being respectful on the Internet. Each online interaction should be polite, courteous, kind, and considerate — using a sort of “do unto others” set of e-ethics to guide all your online correspondence and transactions.

To find out how your Internet manners rate, take the Netiquette Quiz at www.albion.com/netiquette/netiquiz.html.

Free tips and articles

If marketing products or services is the driving force of a Web site, e-mailing industry-related tips to subscribed members and publishing regular articles on the site are both smart ways to provide tangible benefits and build a positive relationship with visitors. And remember, the more positive contact a site has with its audience, the greater the likelihood that audience will want the site's products or services.

Finding ideas for these tips and articles is quite easy. Just think of all the details you know about the business that could help visitors and then jot them down. For example, if you're designing a site for a dog-grooming business, the tips might include the following:

- ✓ How to choose a dog-grooming brush
- ✓ A review of the best dog shampoos
- ✓ How to keep a dog's teeth clean
- ✓ Exercise tips that keep dogs fit

Quick tips within the e-mail can also help bring visitors to your site to read more detailed tips online, as well as find out more about and potentially purchase the site's products and services. Take the CliffsNotes Web site (www.cliffsnotes.com) for example, shown in Figure 1-8. Visitors can sign up for newsletters; browse for literature, test prep guides, and other titles; and get free advice on studying and student life — all for free.

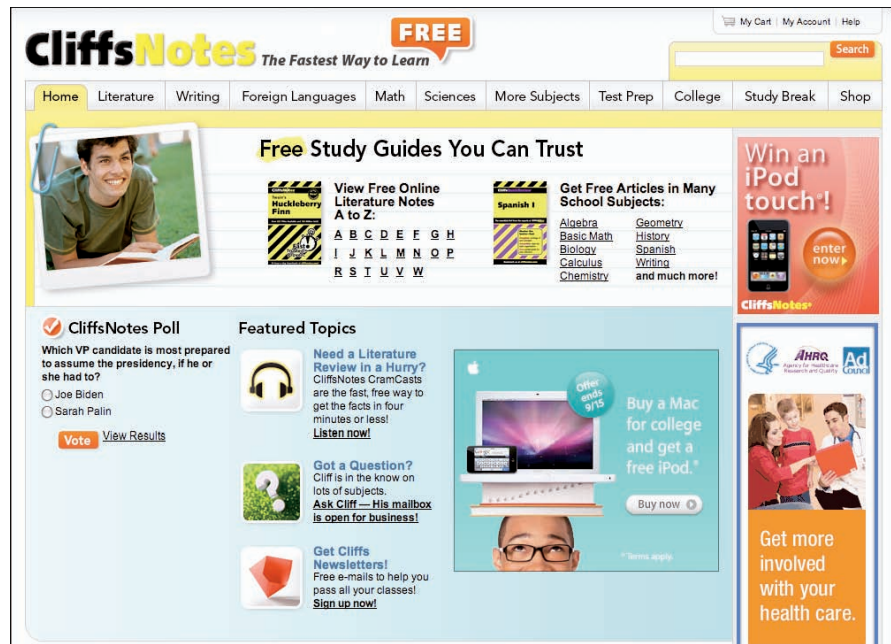


Figure 1-8: Savvy Web sites send e-mails to subscribers that encourage readers to visit the site.



To help generate more traffic to your Web site, be sure to regularly update the content on the site. Fresh content often translates into more visitors — and hopefully more sales. For a fast approach, sit down now and come up with at least 12 ideas that you could conceivably use for tips or articles over the next 12 months. Then all you have to do is write them out and send them at the appropriate time. Keep in mind that if you intend to archive the old news items as you update information on your site, you may want to seriously consider using a database to both manage that content and help you to display the new data dynamically on your site.

Blogs

Blogging is now the single best way for Internet readers to find out about and provide feedback on nearly every topic of interest. That’s because blog news travels fast. Blogs encourage instant feedback from readers and are the place for readers to share information and experience. Best of all, participation in the blogosphere provides instant cachet in the Internet world. When combined with business goals (such as increasing Web traffic and online sales), blogs provide business owners with the opportunity to communicate directly with their target audience.

Understanding what a blog is

So what is a *blog*? The name is short for *Web log*, and it typically refers to a Web site that publishes or *posts* articles (or just a few paragraphs with a photo or other graphic) of information related to particular products, services, news, careers, hobbies, thoughts, opinions, beliefs, images, movies, impressions, or ideas on a topic (or series of topics along a central theme) and solicits feedback from visitors. These articles tend to be published on a regular basis (daily is most popular, although weekly, biweekly, or monthly is okay too) and listed on the site in reverse chronological order, with the newest articles at the top of the page and older articles below. In addition to newer articles, most blogs contain archived articles, pictures, and links to other sites and blogs of interest in the same or related fields.

A person who has his own blog or writes for one is called a *blogger*. Thus, *to blog* means to either post entries on your own blog or to respond to an entry on another person or company's blog, and the *blogosphere* refers to the entire movement of blogging and all things pertaining to the world of blogs.

One popular feature of blogs is the ability to allow visitors to e-mail the author and/or respond directly to any given article by posting their comments to it, thereby creating a forum for online exchanges between the blogger and the blog's audience. Having a blog that offers advice and feedback from other consumers can be a very effective site tool for attracting and keeping visitors.

Using a blog on your site

As you consider whether to include a blog on the site you're designing, here are some important points to keep in mind:

- ✓ A successful blog requires one or more people who can add new content on a regular basis. To maintain visitor attractiveness, or *stickiness*, your site must be able to post new and interesting content frequently. It's what keeps people coming back to your site. Some blogs post one or more short articles per day, while others post content a few times per week.
- ✓ To spread out the responsibility of authorship, blogs can be set up for groups or businesses, where there can be multiple blog authors in addition to full participation in commenting and feedback.
- ✓ Be sure that you understand some basic blogging rules and authoring styles prior to starting your own blog. One great resource is *Blogging For Dummies*, 2nd Edition, by Susannah Gardner and Shane Birley (Wiley), which takes you through the steps of setting up a blog, explains how to generate revenue and build an audience, and even covers podcasting and videoblogging.

- ✓ A visually appealing blog makes a good impression. Besides the content, the look of the blog is all important, because a visually captivating blog is more welcoming than one that obviously took no care in its design. Fortunately, sites like WordPress.com, Blogger.com, and Eblogtemplates.com contain several useful free layouts, scripts, graphics, and more to assist you with the blog look and feel.
- ✓ In addition to the overall look of the blog, adding pictures and other graphics to individual posts adds appeal, too. Though having images on a blog isn't a requirement, it is a nice feature, and most blogs allow you to quickly upload photos to each post as a way to enhance or editorialize an article. Some even let you upload photos and text straight from your mobile camera phone or Flickr account.

Figure 1-9 shows how the folks at CHOW.com incorporate several blogs (The Grinder, Chow Pick, and so on) right into the main content on the site.



Be realistic about the reach of the blog, because audiences currently comprise only about 20 percent of the total Internet population. (That's worldwide, with most of the audience living in the United States.)

Choosing a blogging tool

Before you add your blog, take some time to decide which blogging tools you'll use. When you add a blog to your site, it can either be off your main URL on a different domain, hosted by a special blog-hosting service (such as www.blogger.com), or on your main URL by using special blog software like the free WordPress software on the server used to host your site (www.yourcompany.com/blog). The four most popular offsite blog hosts are Blogger.com, Multiply.com, WordPress.com, and LiveJournal.com. For a complete listing of the Best Blogging Hosts, check out the Blogger's Choice Awards at www.bloggerschoiceawards.com.



Of all the blog tools that I've used, my favorite, by a long shot, is WordPress, shown in Figure 1-10. WordPress is free, easy to set up and configure, provides hundreds of free templates, provides scads of user support and other technical information, and is easily customizable should you decide you want to make the blog match your Web site's design.

Be aware that some of the free blog tools offer enhanced blogging services for a fee. For example, TypePad.com has five tiers of pricing, from a simple Basic one blog per author (around \$4.95/month) to a Business Class that is robust enough to handle unlimited blogs and authors (approximately \$89.95/month), and BlogHarbor.com offers full features at different pricing tiers (from about \$8.95/month to \$14.95/month) that vary by bandwidth (5GB to 40GB) and hard drive space (2GB to 10GB).

On TechRepublic: Are you too old for IT?

CHOW
FOOD. DRINK. FUN.

Login | Sign Up | Newsletters
Sign up for CHOW newsletters!

SEARCH

Recipes ▾ Chowhound ▾ Places ▾ Stories ▾ Blogs ▾ Videos ▾ my CHOW

theme ▾

Kitchen Timers
Keep food from burning, in style
BY MICHELE FOLEY | view»

Wanted: Kitchen Timers

A DOZEN WAYS TO BREAK 'EM.
AN EGG RECIPE CHALLENGE
CHOW FOOD FIGHT ENTER

RECIPES

FOOD FIGHT
Eggs ENTER NOW!

NEW! Member Recipes Publish a Recipe

Now you can Add Your Own recipes to CHOW!

Recently Added:

- Vegan Filet Mignon by Veggie Queen
- Deconstructed Egg with Chipotle Cream Sauce by plumblossom

See all...

SOUP/SALAD/SANDWICH
Three-Bean Salad
A perennial picnic favorite

DRINK
Arnold Palmer
An unbeatable combination of iced tea and lemonade

BREAKFAST/BRUNCH
Chile-Cilantro Hash Browns
Crisped potatoes with a Southwest flair

STORIES

SUPERTASTER
The Kitchn Value of Jimmy Dean
Sweet & Sour Shrimp, and Pancakes and Sausage Minis

PROJECT
Make Your Own Jerky
Homemade jerky takes little more than patience

THE JUICE
Slow Spirits
If booze is to be slow, it has to be right

PROJECT
Shockingly Tasty Fruitcakes
Start now, finish aging by Christmas

TABLE MANNERS
Unforbidden Fruit
How to help yourself from your neighbors' trees

BLOGS

THE GRINDER Our Food Media Blog
The Not-So-Supermarket
Shrinking grocery stores cater to customers who want quick, focused, and already prepared. [More...](#)

Traci Vogel
today at 10:58am | [Comments \(0\)](#)

CHOW P/CK Our Favorite Things
Coke Farm's Meyer Lemon Syrup Tart, lemony, and balanced. [More...](#)

Roxanne Webber
last Friday at 3:25pm | [Comments \(0\)](#)

THE DIGEST Chowhound's Daily Roundup
Armenian Bazaar Specialties
Festivals offer authentic dishes. [More...](#)

Cicely Wedgeworth
last Friday at 1:34pm | [Comments \(0\)](#)

TASTINGnotes Our Wine Blog
Surf, Wine, and Money
My Hawaiian vacation fantasy. [More...](#)

Figure 1-9: The CHOW.com Web site uses a blog to allow its writers to communicate directly with visitors.

Adding a profile

To help draw visitors to a new or existing blog, add a profile. When the blog is live on the Internet, you can create a profile (such as an About page on a Web site) for the blog's author or authoring group. The profile identifies the blog/blogger's interests by category (such as worldwide volunteerism, CSS hacks for Web designers, or organic foods and recipes), by statistics, and by outside blog links. This profile helps people with common interests find the blog.

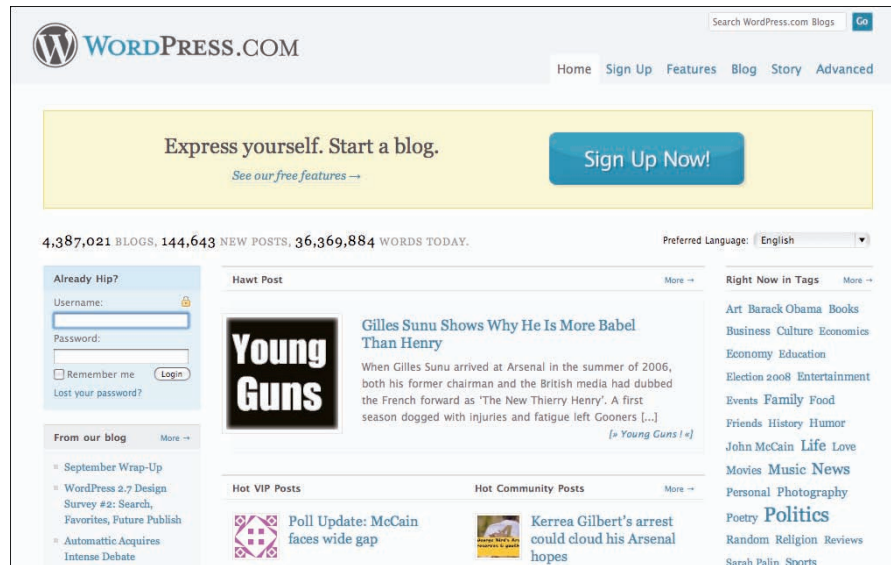


Figure 1-10: Use WordPress if you plan to host your own blog.

Profiling is key so that the audience visiting the site belongs to the demographic group the blog is concerned about. For example, if you create a blog for your client on the topic of fundraising for education, the target audience might not necessarily include train hobbyists unless one or more of the posts on the blog relates to using hobby trains as a means to raise funds for education.

Examining the pros and cons

On the plus side, with very little overhead, blog posts can and often do spread news and information faster than most traditional media sources, such as TV, newspapers, and radio. Similar to some forms of guerilla marketing, where information is passed through word of mouth, blogs far surpass traditional marketing avenues because they're global and typically reach an audience who's interested and takes active participation in getting the news rather than passively having advertising and ideas presented to them.

On the minus side, regularly posting to blogs can be a very time-consuming task, so you'll need to consider in advance what kind of posting schedule to maintain — be it daily, a few times a week, or weekly — and then be ready to stick to it. Remember your site's statement of purpose? If you're planning on blogging as a way to keep your site relevant and interesting to visitors, make sure your commitment to keep the blog content current is part of your mission statement.



An abandoned or unkempt blog can be more injurious to a business identity than no blog at all. A blog with little to no content — as well as one with little to no feedback — can give visitors the impression that the blogger doesn't care about visitors, which in turn can make visitors not care about visiting. And, if no one cares, why bother reading posts, exploring the adjoining Web site, and possibly using the site's products and services?

In addition to posting relevant articles, news, and information, other elements can be added to a blog to enhance the visitor experience and generate revenue for the blog owner. For example, you can use Google AdSense, a tool that automatically places content-relevant ads on registered users' blogs. Each click on an ad by a visitor earns money for the blog owner. You can also globalize content through blog syndication (news feeds with Atom or RSS), whereby the blog host generates machine-readable versions of the blog for display on special newsreaders, handheld devices, and Web sites. Bloggers might also benefit from enrollment in blog services (such as Bloglines.com or Technorati.com) that allow enhanced blog searches and shared news feeds, among other things.



The bottom line for blogging is that because the blogosphere is rife with illiterate, uninteresting, and infrequently updated or abandoned blogs, adding a blog to your site is only a good idea if someone is willing to devote time to adding to and improving the blog, to implementing ways to drive relevant traffic to it, and ultimately to both saying something interesting and saying it well.

Polls and calculators

Because people love to give their opinions as well as learn about what other people think, polls are great tools to add to Web sites where opinions matter. For example, folks who are crazy about *American Idol* can visit the entertainment section of America Online to (unofficially) vote for their favorite idol. Likewise, moviegoers who want to give their opinion about whether a book was better than the movie version can sound off with a poll at MoviePhone.com.

Polls generate buzz at the water cooler, and that kind of talk might generate more business. Like blogs, polls can be hosted remotely or added to a site by installing poll software on the server. Addpoll.com, Bravenet.com, Basicpoll.com, and Sparklit.com, among others, offer free or subscription polling services. Or, if you're more technologically minded, visit www.javascriptkit.com/howto/polls.shtml to find information about installing a polling program on your site by using PHP, CGI, or ASP.

As for calculators, depending on the type of business you're designing a Web site for, having a calculator somewhere on the site could help increase traffic from the target demographic. For instance, if the Web site offers mortgage loans, like the one shown in Figure 1-11, having a mortgage calculator

on the site that crunches different monthly payments and interest rates for prospective clients can provide a big incentive for visitors to check out a new site. Or, if you're designing a site for a travel company, adding currency and temperature calculators can be an unexpected and appreciated special aid to travelers.

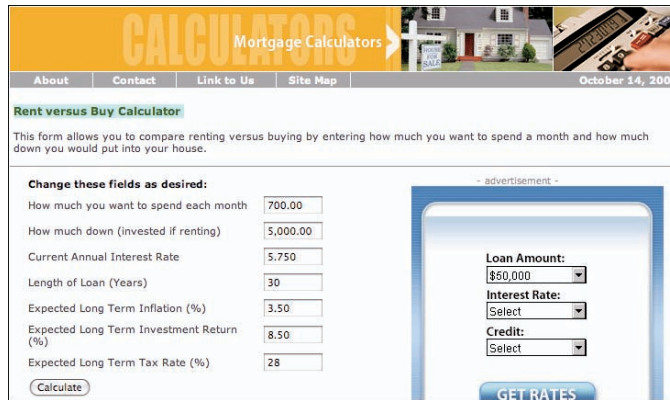


Figure 1-11: A calculator can help visitors figure out their monthly payment.

Because calculators are a great value-added feature for Web sites, and the JavaScript code for many online calculators can be found and used for free, the following example walks you through the steps you'd take to find and use a calculator on any of your Web site projects. For your own projects, I highly recommend that you first visit www.calculator.com to get a general overview of the kinds of online calculators that exist, and then spend some time searching for free calculators to find Web sites that offer free JavaScript code, such as <http://javascriptkit.com>.

To see how easy it is to add a JavaScript calculator to your site, follow these steps to add a Body Mass Index (BMI) calculator to a sample HTML page:

1. Open a blank Web page in your preferred HTML code editor.

Or, if you're using a simple text editor such as Notepad or TextEdit, open a new blank document and type in the following basic Web page code:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"
/>
<title>Untitled Document</title>
</head>
<body>
<!-- Insert javascript below this line -->
</body>
</html>
```

2. **Point your browser to** <http://javascript.about.com/library/blbmi2.htm>.

This page contains the form part of a free script that can calculate the BMI, or Body Mass Index, based on the input weight and height.

3. **In the scrollable text area below the description, click the Highlight All button to select all the code and press Ctrl+C (Windows) or ⌘+C (Mac) to copy it to your computer's Clipboard.**
4. **Paste the copied calculator code anywhere between the opening and closing <body> tags on your blank Web page.**

If you created your own page by using the code from Step 1, paste the calculator script below the comment line that says Insert javascript below this line.

5. **Type the following line of code into the head of your Web page to link the form to the JavaScript that will process the form input data:**

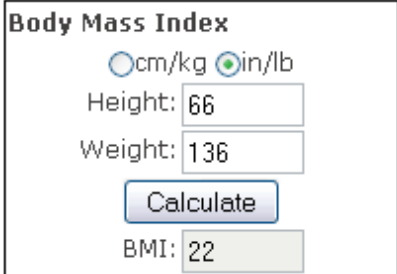
```
<script type="text/javascript" src="bmicalc.js"></script>
```

6. **Choose File⇨Save As, give your Web page a filename such as bmicalculator, and save the page with the .html or .htm file extension.**
7. **Point your browser to** <http://javascript.about.com/library/blbmi3.htm>.
This page has the JavaScript that can make the form code function.
8. **In the scrollable text area below the description, click the Highlight All button to select all the code and press Ctrl+C (Windows) or ⌘+C (Mac) to copy it to your computer's Clipboard.**
9. **Open a new blank document and paste the copied code into it. Then save this file as bmicalc.js in the same location as your bmicalculator.html file.**
10. **View the calculator in action by opening the saved HTML file in a browser window.**

To open the page in a browser, either double-click the file to launch it in a browser or drag and drop the file by its icon into any open browser window, either with or without a live Internet connection.

11. To test the calculator script, enter some test data.

For instance, you might choose in/lb (inches/pounds) and enter **66** for the height (the equivalent of 5 foot 6 inches) and **136** for the weight to get a BMI of 22, as shown in Figure 1-12.



Body Mass Index

cm/kg in/lb

Height:

Weight:

BMI:

Figure 1-12: Adding a free calculator to your site, like this one for BMI, helps attract site visitors.

Contests and sweepstakes

Having a contest, sweepstakes, or some kind of giveaway on a site is a great way to attract visitors and keep them coming back month after month. For example, Cookinglight.com (shown in Figure 1-13), offers several different contests and sweepstakes to visitors each month.

Contests can be for nearly anything you can think up. Raffle off a car. Give away a free computer class. Send winners on an all-expenses-paid vacation. Affiliate your company with a worthy cause and offer cash to winners while increasing awareness about an important issue. Sponsors of events will often provide valuable prizes for relevant and worthwhile contests at no cost in exchange for the free publicity, such as offering winners a \$500 T-Mobile gift card, 10 guest passes to the latest IMAX movie, or 100 free Betty Crocker cookbooks. Enrollment in the contest can be at the visitor's discretion through an online contest entry form, or it can happen automatically after a visitor signs up for an e-newsletter or registers for membership.

Most contests and sweepstakes need the following:

- ✓ A set of rules and regulations outlining who can and cannot participate in the contest
- ✓ A defined contest time frame, with entry dates and award dates
- ✓ A list of prizes and the odds of winning
- ✓ A way, both online and off, for visitors to enter the contest
- ✓ An objective third-party administrator
- ✓ Proper insurance



If you decide to have an online contest or sweepstakes on your site, be sure to follow the strict federal and state legal guidelines to ensure that your contest is fair. Read the article “Online Contest or Illegal Lottery?” by Ira Rothkin on the legality of contests and the pitfalls of illegal lotteries at

http://findarticles.com/p/articles/mi_m1563/is_/ai_n27524204 as a starting point to find out more about what legal rules to follow. You might also want to seriously consider hiring an outside firm, like Nationalsweeps.com or Oddsonpromotions.com, to organize and administer the contest for you.

Cooking Light Perks Special offers and promotions from *Cooking Light* and its advertisers

Home | **CONTESTS/SWEEPSTAKES** | Supper Clubs | Contests/Sweepstakes | Getaways | Events | Special Promotions | Marketplace | Advertise With Us

Cooking Light Ultimate Reader Recipe Contest

Thank you to all of those of you who entered our fourth annual *Cooking Light* Ultimate Reader Recipe Contest and check out our [12 finalists and winners](#).

OREGON BOUNTY

Foodies, Rejoice!
Discover Oregon's Bounty. You're invited to get away and sample artisan wines, spirits, brews, and a plethora of fresh, local cuisine during our annual gastronomic extravaganza, Oregon Bounty. Come meet the vintners, chefs, brewers, innkeepers, and farmers who make Oregon taste like Oregon. Enter for a chance to win the ultimate Oregon Bounty escape.

[Enter Now»](#)

visitnc.com

Explore North Carolina
Golden leaves, green golf courses, red wine, and the Blue Ridge Mountains. In North Carolina, we've got spectacular fall colors that will take you places. Enter at to win an amazing three-day getaway to the Great Smoky Mountains in North Carolina and get some well-deserved Blue Ridge color therapy!

[Enter Now»](#)

NIGHTS SAVOR

Win a Weekend Trip for Four to Napa Valley
Enter the Nights To Savor Sweepstakes for your chance to win a weekend trip for four to Napa Valley, including airfare, accommodations, and more. You can even double your chances to win! Simply fill in the UPC code online from your favorite Sutter Home wine or Colavita product to be entered twice. *Ends 12/31/08*

[Enter Now»](#)

FREE NEWSLETTERS

- » [Cooking Light](#)
- » [Supper Club Dish](#)

Figure 1-13: Contests are a great way to bring in visitors to a site time and again.

Chapter 2: Defining the Audience

In This Chapter

- ✓ Performing market research
- ✓ Gathering information on the target audience's computer use
- ✓ Assessing a site's competition
- ✓ Understanding how to characterize a target audience
- ✓ Determining benefits to site visitors

After the planning phase of your Web project, as explained in Book I, Chapter 1, you should have enough information to successfully move into the contract phase, where you estimate fees for the project and then draft and submit a proposal to the client.



When the client gives verbal approval of the proposal, you can write up an official contract and present it to the client for signatures. If you're in need of a sample contract, you might want to start with the example shown at www.premiumwebdesign.com/contract.htm. After you create your own copy, you can modify it as needed to suit your individual projects.

At the same time as you receive the signed contract from your client, you can also collect financial retainers or deposits along with any content or materials needed to begin development of the site. Try to get at least a 25 percent deposit from the client before you do any work. This shows good faith on your part for doing the work and good faith on the client's part that she is serious about having you do the work and is willing to pay part of the fees to retain your services.

After you get the contract and deposit, you can safely enter the design phase, which is where this chapter begins. Identifying the target audience (the first part of the design phase) is an information-gathering process that helps you make a Web site's design effective.

In this chapter, you define your target audience by finding out everything you can about the target audience members — what their computer usage and Internet surfing habits are, where they fall demographically, what

	IE7	IE6	IE5	Fx	Moz	S
December	21.0%	33.2%	1.7%	36.3%	1.4%	1
November	20.8%	33.6%	1.6%	36.3%	1.2%	1
October	20.7%	34.5%	1.5%	36.0%	1.3%	1
September	20.8%	34.9%	1.5%	35.4%	1.2%	1
August	20.5%	35.7%	1.5%	34.9%	1.3%	1
July	20.1%	36.9%	1.5%	34.5%	1.4%	1
June	19.7%	37.3%	1.5%	34.0%	1.4%	1
May	19.2%	38.1%	1.6%	33.7%	1.3%	1
April	19.1%	38.4%	1.7%	32.9%	1.3%	1
March	18.0%	38.7%	2.0%	31.8%	1.3%	1
February	16.4%	39.8%	2.5%	31.2%	1.4%	1
January	13.3%	42.3%	3.0%	31.0%	1.5%	1

their buying preferences are, and what special interests they may have. You also spend a little time doing your own informal market research by taking a good look at the competition to give your Web project an extra edge.

Defining the Target Audience

The *target audience* for a Web site is the ideal group of visitors site owners hope to attract in an effort to increase Web traffic and thereby improve sales. In other words, they are the intended visitors of a Web site, as defined by their common interests, habits, and demographics. This ideal group might be of a certain age or gender; come from a particular part of a neighborhood, city, county, state, region, or country; and have very specific interests as well as particular likes and dislikes.

To determine these and other characteristics, you and your client should perform certain tasks, including performing a bit of informal (or formal) market research, gathering Internet usage statistics, and taking a look at what the competition is doing so that you can create a more attractive design for your site and ensure that the site includes the relevant content that the target audience is likely to seek.

Doing informal market research

Market research is a type of research performed when information about a particular group of people needs to be gathered as an aid to making strategic marketing decisions. Whether performed in a formal or informal manner, market research is one of the best ways to begin the design phase of any Web site project.

With your particular Web design project in mind, be sure to complete the following three tasks, from the checklist shown in Figure 2-1, to make the most of your market research time:

- ✓ **Gather general information about the computer usage and Internet browsing habits of Internet users.** This knowledge about the people using the Internet can greatly help you make important decisions about the site's design measurements, organization, layout, color palette, image usage, navigation scheme, and accessibility features.
- ✓ **Find out what other businesses in the same field have already done with their Web sites.** By looking at competitors' Web sites, you can quickly assess what was already done poorly and take steps to avoid those same mistakes. In addition, you can find out a lot

✓	Investigate computer stats of potential visitors.
✓	Check out the competition.
✓	Define the ideal site visitor.

Figure 2-1: Create your own marketing research checklist to gather information about your project.

from what the competition has already done successfully and make plans to implement similar, though not identical (otherwise, that would be plagiarism or copyright infringement), ideas in your Web project.

- ✓ **Define the ideal site visitor based on the demographic and other statistical information you gather from your own informal marketing research.** The clearer your understanding of your target audience — their ages, income levels, buying habits, and interests — the easier it is to customize your site design to their tastes. (See Book II, Chapters 1 and 2, for more on implementing your design.)



If you don't know much about market research, take a look at the KnowThis.com Web site, which bills itself as the Marketing Virtual Library — a place where visitors can find out about market research, marketing, advertising, and more. Besides the general marketing information, you can also find many useful articles, tutorials, and even free research reports there. Another great resource on marketing and market research is the Marketing section on About.com (<http://marketing.about.com>).

Gathering Internet usage statistics

Before you gather any specific demographic information about your target audience (which you define later in this chapter), do research and gather some general information about what folks are on the Internet and what kinds of computers, operating systems, and browsers they are using.

Finding statistics online

You can easily find out more about Internet users by reviewing the latest statistics regarding their computer usage and Internet-browsing habits. Here are some guidelines to get you started:

- ✓ **Visit one or more online resource Web sites that offer information on computer usage and browsing.** One of the best is www.w3schools.com, shown in Figure 2-2, where you can find a list of detailed, long-running (since January 2002), up-to-date browser, operating system (OS), and computer usage statistics.

For additional, comparative statistical data on general Internet usage and browsing habits, visit these sites:

- www.thecounter.com/stats
- www.upsdell.com/BrowserNews.index.htm
- www.webreference.com/stats/browser.html
- www.ews.uiuc.edu/bstats/latest.html

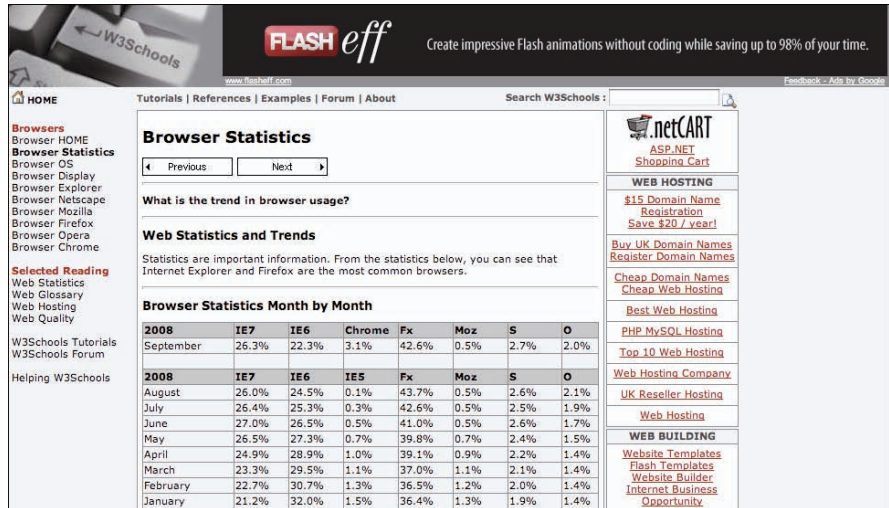


Figure 2-2: Gather Internet usage statistics from reputable sites like this one.

- ✓ **Jot down the information you find out about browser usage.** Look specifically at the percentage of users on the most popular browsers, including Internet Explorer (IE), Firefox, Safari, Opera, and Mozilla.
- ✓ **Find out which operating systems (Windows, Mac, Linux) are the most popular among people accessing the Web.** You might be surprised at how few Internet users are on a Mac versus a PC.
- ✓ **Find out the monitor resolutions and bit depths of Internet users.** This data is often referred to as *Browser Display Statistics*.

The *bit depth* is the grayscale or color depth of an individual pixel of an image displayed on a monitor. An 8-bit-depth monitor can display only 256 colors, whereas a 24-bit color monitor displays 16.7 million colors!

Display resolution refers to the number of pixels, organized horizontally and vertically, that are used by a computer monitor to set the screen size of the monitor. The most common factory default size is 1,024 x 768 or higher. When the resolution size is set to a particular size, the monitor uses bits — actual tiny squares — of color to create the on-screen display. The more bits that are used, the clearer the image and the more colors that can be displayed. Higher resolutions result in more pixels being used on-screen. In other words, you can fit more on your screen at a higher resolution, though everything will be displayed proportionally smaller.



✓ **Note what percentage of Internet users (95 percent as of January 2008) leave JavaScript enabled in their browsers.** JavaScript is an easy-to-use, easy-to-learn, simple scripting language that, when added to a Web page’s HTML code, can perform dynamic and interactive functions in a browser window, such as opening new browser windows, changing an image when a visitor moves the cursor over a graphic (*rollover buttons*), and displaying the current date.



Though the percentage is small, some site visitors either don’t have browsers or other Web-reading devices that support JavaScript, or have chosen to disable it. To make a site accessible to all possible visitors, the site should be accessible even when JavaScript is turned off.

Interpreting statistics

Going by the W3Schools.com analytic data, as of December 2008 nearly 20 percent of all computers surfed the Internet with the Internet Explorer 6 browser; 26 percent were on Internet Explorer 7; 44 percent were using Firefox; 3 percent were using Google’s new Chrome browser; only 2.7 percent were using Safari; and another 2.4 percent were using Opera. A whopping 71.4 percent of all Internet consumers are PC users running the Windows XP OS, while only 15.6 percent have migrated to Vista; only 5.3 percent are using the Mac. And, because new monitors come with factory preset resolutions of at least 1,024 x 768 pixels (48 percent) or higher (38 percent), nearly all of these computer users who go online leave their monitors’ resolution at the factory setting.

This kind of information clearly tells you that, at a minimum, you need to test Web page development in both IE 6 and IE 7 JavaScript-enabled browsers on a PC with a 24-bit-depth monitor with a resolution set to 1,024 x 768 or higher prior to site launch, because these are the facts about most of the online population that is likely to visit the Web site. In other words, though your ideal site visitor will have certain characteristics, the majority of visitors will have certain characteristics in common, as listed in Figure 2-3. In addition, to make the site accessible to the widest possible audience, you should also test in all the operating systems and in the different browsers on both the Mac and PC.

Browser Statistics of the Typical Site Visitor	
✓	Uses a PC running WinXP or Vista
✓	Uses Internet Explorer 6 or 7
✓	Has JavaScript enabled
✓	Has a 24-bit-depth monitor with a resolution set to 1,024 x 768 or higher

Figure 2-3: A majority of Internet users have several things in common.



After a Web project is completed and published on the Internet, the site can harness the power of real-time Internet traffic reports to gather statistical data about actual site visitors, including geographic origin, search term usage, entry and exit pages, and more. Three well-respected, fee-based services that provide Web statistics and analysis are Opentracker.net, Omniture.com, and ClickTracks.com. Although costly, this kind of data can help site owners identify marketing strategies that aren't working so that they can make site improvements based on actual visitor preferences such as search term usage (which search term led the user to your site), entry page, exit page, and time spent on the site. For a free alternative, consider using Google Analytics at www.google.com/analytics to learn more about your visitors and how they interact with your site and to learn ways you can improve your site to attract more visitors and convert them into paying customers.

Sizing up the competition

No matter what business your Web project happens to be in, you're bound to find several, if not hundreds, of other Web sites that represent competitors in the same field that can be evaluated by you and your client for their apparent successes and failures. The beauty of this market research strategy is that it's free, and as long as you have access to a computer, you can do this kind of business investigation online anytime, 24 hours a day, 7 days a week.

Performing keyword searches for similar companies

Begin your research by doing several keyword search-engine searches for similar businesses in the local, regional, state, national, and global arenas, regardless of your Web project's anticipated marketing scope. For example, if you're working on a Web project for a local catering company that does business only within a 100-mile radius of its offices, don't limit your searches to local competition only. Instead, make sure that you also search for catering company Web sites across the country. If you do this, you're more likely to find some great catering Web sites in other geographic regions that can be used for design inspiration as well as a springboard for defining and refining your project's site content and design requirements.

Keywords — if you didn't already know this — are any specific words or phrases that define the object, person, or place a site visitor is searching for in a search engine, database, or catalog. For example, if you want to find a new hairdresser, you might do a search-engine search for the keywords "hair salon" along with the name of your city and state, such as "hair salon, San Francisco, CA." After submitting the keywords to the search engine, the browser then displays a search results Web page that lists all the results that contain those same keywords. The results are culled from all the Web sites on the Internet that include those keywords in their page content, page titles, headings, and metadata, all of which were purposefully placed there to drive additional traffic to a Web site.

How do you know which keywords to use to find these competitors' Web sites? The answer is simple: Just think of the words you would use to find your client's business online.



Another way to find relevant keywords for your site is to take a peek at the competition's keywords, which are typically placed in the `Keywords` and `Description` meta tags inside the HTML code of the Web site's home page. Not all sites use them, but the good ones do. To locate these keywords, open a competitor's Web site in a browser window and choose `View` → `Page Source` from the browser's main menu. The browser then opens a separate window that contains the HTML code of the page displayed in the browser. Scroll down just a little from the top until you see the `title` and the meta tags, and review the `Keywords` and `Description`. The tags should look something like this example from www.rockwoodandperry.com:

```
<title>Rockwood & Perry offers fine wines, advice, accessories, direct
imports, 1982 Bordeaux, etc.</title>
<meta name="Keywords" content="Fine wine, Fine Bordeaux, Fine Burgundy, Italian
Wine, Rhone Valley Wine, California Cabernet Sauvignon, 1982 Bordeaux, old
and rare wines, vintage Port, California Chardonnay, Montrachet, Wine
Glasses, Wine Accessories, Westchester Wine Store, Riedel Glasses, small
production wines, low yield wines, high quality producers">
<meta name="Description" content="Offerings of fine vintage wine: Bordeaux,
Cabernet, Barolo, Burgundy, Chardonnay, Merlot, Chateaneuf du Pape, Cotes
du Rhone, Brunello di Montalcino, Syrah. Advice on collecting, cellaring,
drinking and storing French, Italian, California, Spanish, Australian wines.
Corkscrews, accessories.">
```



You learn more about working with meta tags in Book I, Chapter 3, but if you're eager to find out more about them now, read Google's Webmaster Central blog post on meta tags at <http://googlewebmastercentral.blogspot.com/2007/12/answering-more-popular-picks-meta-tags.html>.

To illustrate how to perform keyword searches at the local, state, and global market levels, I'll presume that you have a new Web client who owns and operates an antique shop in Hartford, Connecticut, that specializes in Victorian furniture, and you want to assess the competition. Here's what you should do:

- 1. In the search field of your favorite search engine (such as Google, Yahoo!, MSN, Ask, or AOL), type the following local search keywords, including the commas and spaces, as illustrated in Figure 2-4: antique furniture, victorian, hartford, connecticut.**

The local search includes the name of the client's city and state, which can help the search engine narrow the focus of the returned results. Results should include a listing of other antique stores in the same town that also specialize in antique Victorian furniture. Take a look at the top 10 or 20 links and review each site, making notes about the content and layout.

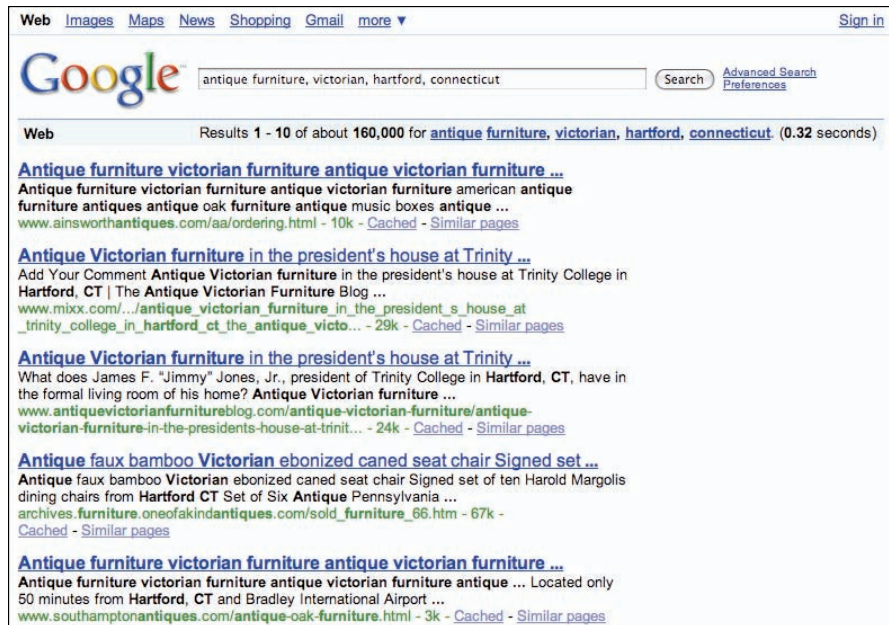


Figure 2-4: Use your favorite search engine to find competition at the local, state, and global market levels.

- 2. For the next search, expand the results listing to a statewide search by entering these keywords into the browser search field: antique furniture, victorian, connecticut.**

This search doesn't include the city of Hartford, so the results include antique furniture stores throughout the state of Connecticut. The results are now geared more toward antique businesses within the state; dealers, buyers, and collectors; antique search directories; and antiquing blogs. Again, click through to the top 10 to 20 links in the search results listings and note any interesting content and layout features that appeal to you and might be useful to mention to your Hartford antique shop Web client.

- 3. For the last search, which is national or global in nature, omit any geography in the keywords. Search for antique furniture, victorian.**

This last search provides links to other antique furniture businesses across the country, as well as to sites that sell other kinds of antiques besides Victorian furniture, antique buying guides, home furnishing stores, appraisers, reproduction services, and showrooms. In this search results listing, you might want to explore sites deeper than 20 entries to find other relevant competitors' sites, because this search result includes a wider variety of results that may or may not suit your needs.

As you can see, the results listings for each of these keyword searches are geared to the specific geographic areas contained in the keyword string, providing you with more insight to the world of antique Victorian furniture across the city, state, and country. By doing wider searches like this, you're likely to find that companies doing national and even global business tend to have the better-looking, better-functioning sites when compared to their state and local counterparts.

Evaluating competitors' sites

As you evaluate the sites in the search results listings, look closely at the content presented and work with your client to ensure that his or her site includes content that visitors will want to read. In the case of BBQ catering, that might mean including sections on the site like Company Information, Menu, Directions to the Restaurant (if the catering is part of the main business of running a restaurant), Press Releases and Awards, Visitor Comments, Specialty Sauces for Sale, and possibly Dine-In Coupons.

Here are some questions you might want to ask yourself when reviewing competitors' sites:

- ✓ Can you determine where the target audience lives: in the suburbs, in the countryside, in the city, or all over the world?
- ✓ What kinds of photographic and graphic images appeal to the target audience?
- ✓ Do competitors' sites use any particular colors repeatedly in an attempt to appeal to their target audience?
- ✓ What is the target audience's age group: Teens? Kids? Seniors?
- ✓ Does the target audience have any particular hobbies or group interests, like being an avid reader, sports fan, cook, artist, or gamer?
- ✓ Can you guess what kinds of values (religious, ethical, political, and so on) the audience might have?
- ✓ Can you identify any specific needs of the target audience?

Pay particular attention to Web site details, such as colors, shapes, fonts, photographs, and other design elements that are consistently used on competitors' sites. When you know what everyone else is doing, you have the opportunity to decide whether to follow suit or break the mold in a unique and interesting way. If you're developing a new site for a company in the finance industry, for instance, and you notice that almost all the competitors use navy blue as the primary color and burgundy as the secondary color, consider using two different but similar colors for your design, such as a light blue and a rusty tan.

Also pay attention to the marketing messages other sites use to sell their products and services. These messages often shed revealing light on the target audience's buying preferences, habits, likes, and dislikes.

Even when your client provides you with all the content before you begin the design phase of the project, you should still perform a keyword review of the competition to ensure that the client hasn't forgotten anything that might make his or her Web site more attractive to visitors. This research can also help you to make important decisions in regard to layout and design so that the site will stand out from the competition.



Other good resource areas for gathering market research on a particular topic or business are the industry-related associations and organizations, as well as government agencies. You can often find Web sites for these groups by doing a search in your favorite search engine using terms like “*Industryname* Association of America,” “American *Industryname* Association,” or “Association of *Industryname* of America.”

Summarizing your results

When you're done with your research, organize the information into logical categories and summarize the details into a few definition-packed sentences. For example, if you perform keyword searches about a new Web site you'll be designing for an interior design company based in San Francisco, you might discover that many other interior design Web sites favor using black, white, gray, and earth tones in their Web site designs; display lots of photographic examples of their work; and target their services to an audience who is well-educated, cultured, and moneyed.

With this information, you can write out a summary of facts such as

“Black, white, gray, and earth tones; clean, linear layouts; and a sophisticated audience that needs to be wowed with photographic examples of the client's work.”

You can then use this summary to assist you in both defining the ideal site visitor and then coming up with a design that will appeal visually to that target audience. In fact, if you want to take things a step further, you can quickly mock up a visual profile of the target audience, like the one shown in Figure 2-5.

Defining the Ideal Site Visitor

Having a clear understanding of the ideal site visitor is an essential element of the design phase because that description guides you in making important decisions about the site's design. For instance, if you're developing a

health care Web site for an audience composed mostly of seniors, you might deduce that a larger font size is an important issue and thus choose to make special modifications to the site's Cascading Style Sheets (by using percentages or other scalable measurement units for the font size rather than pixels, which are fixed in size) that can allow those visitors to adjust font sizes in their browsers. This also provides you with the opportunity to ensure that the site layout looks good both when the fonts are "normal size" and when they're increased in the browser.

At this stage, after gathering general computer usage information and doing some informal research about the competition (as described earlier in the chapter), it's time to figure out who the ideal visitor is for your Web site. If you're lucky, the company hiring you as a designer will have already done some of its own market research and can quickly tell you the detailed demographic information about the people using its products or services. Your role here is one of gathering and distilling what the client gives you into an *identity description* of the ideal site visitor.

Start by asking your client for a demographic profile of his target audience. If that information isn't available, you can do what I call *research by proxy*, which is essentially harvesting information about the target audience by looking at competitors' Web sites and other industry-related sites. Perhaps you already got a sense of that audience as you performed your keyword searches (as described earlier in "Sizing up the competition"). Or, if you haven't done any keyword searches on your own yet, do them now.

To assist you, use the following questions to help define the ideal site visitor for your Web project. To illustrate how you might answer each question, I'll use the example of a client seeking a Web site to sell his designer, screen-printed, organic cotton, men's and women's T-shirts.

✓ **Is the ideal visitor a man or woman, or does that matter?**

A Web site for a mostly male audience might look very different from one that has a mostly female audience, whereas a site that should appeal to all visitors, both male and female, can (and probably should) use more gender-neutral colors in its design.

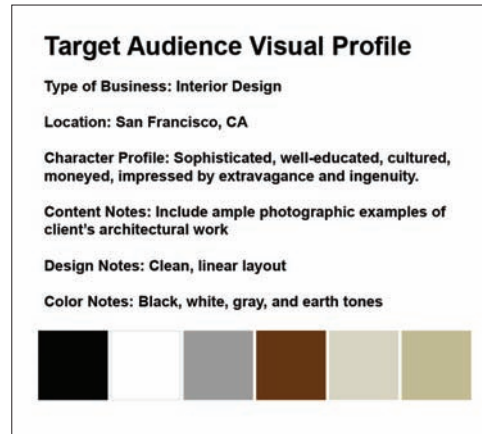


Figure 2-5: A quick visual profile of the target audience can help you make important decisions about a site's content, layout, and design.

T-shirt site: Because the client is selling both men's and women's T-shirts, the audience must include both men and women. However, because women shop more often than men, the ideal visitor is probably female.

✓ **Is the ideal visitor young, old, or somewhere in between? What age range does he or she fall in?**

Having an age range for the ideal site visitor can assist you with making artistic and accessibility decisions. For example, a site for mostly college-aged student visitors can be more alternative in design layout, color, and font usage than a site that needs to appeal to white-collar businesspeople in their 40s or 50s.

T-shirt site: Designer T-shirt wearers tend to be in the 12- to 42-year-old age range, depending on the grade of cotton, the sophistication of the design, and the intended retail outlets, if any. Presuming that the client wants starlets like Paris Hilton and Lindsay Lohan wearing them, I'll narrow the age range to 18- to 30-year-olds.

✓ **Answering the next set of questions, which might rely solely on your imagination, can assist you in making other design and layout decisions. Does the ideal visitor**

- Smoke or drink alcohol?
- Attend a place of worship?
- Eat organically?
- Participate in sports?
- Clip coupons?
- Watch TV?
- Read newspapers?
- Vote and get involved with politics?
- Own any pets?

Really get down to particulars and describe your ideal site visitor as clearly and vividly as you would a person sitting right next to you. The more you understand who will be visiting a site, the better that site can look and function as well as provide the information the visitor seeks.

T-shirt site: Because trendy people often break the rules, presume that the ideal visitor is a female who goes to parties, attends church on holidays, occasionally eats organically, doesn't play team sports but does yoga and Pilates, snowboards in the winter, barely watches TV or reads newspapers, spends most of her free time with friends, and has (or really wants to get) a tiny pet dog.

- ✓ **Using adjectives or descriptive statements, create a list of ten or more identity traits that define the ideal site visitor, similar to the exercise you perform in constructing an image for the Web site in Book I, Chapter 1.**

Is the ideal site visitor smart or of average intelligence? Is he or she urban or suburban? Is he or she organized or messy, confident or timid, silly or serious? Who is this person?

T-shirt site: Single, urban, confident, a little irresponsible, fashionable, outgoing, makes good grades, image-conscious, sassy, and fun to be around.

- ✓ **Using the ten or so adjectives you generated in the previous point, write an identity statement for your project’s ideal site visitor.**

The identity statement becomes your guiding statement of who to design the Web site for.

T-shirt site: A single, 22-year-old female who is an urban, confident, sometimes irresponsible, fashionable, outgoing person who overspends on fashion; likes to dance, have fun, meet new people, and try new things; goes to parties; drives a fun car; and lives with roommates.

When you are finished, you may want to round out your visual profile of the ideal site visitor with a photo, like the one shown in Figure 2-6.

Table 2-1 offers some additional examples of identity descriptions for a variety of businesses. Each description is unique to the type of business it represents, and each speaks to the creation of a different look and feel for the site’s design. Remember, the specific account of the ideal site visitor for your project can help you make intuitive, informed decisions about the site’s colors, fonts, navigation, images, and more when you start working on the site’s design.

Target Audience Visual Profile

Type of Business:
T-Shirt Designer/Manufacturer

Location:
Brooklyn, NY

Character Profile:
Ashley is a single, 22-year-old female who is an urban, confident, sometimes irresponsible, fashionable, outgoing person that overspends on fashion, likes to dance, have fun, meet new people, and try new things, goes to parties, drives a fun car, and lives with roommates.

Content Notes:
Include photos of cool young people wearing merchandise

Design Notes:
Simple design with areas of flat color punctuated with sample tshirt designs

Color Notes:
purples, teals, black, and white



Figure 2-6: Add a photo to your target audience visual profile to put a sample face on the ideal site visitor.

Table 2-1		Sample Identity Descriptions	
<i>Type of Business</i>	<i>Description of the Ideal Site Visitor</i>		
Life coach	Committed, somewhat educated, lower- to middle-class male or female who is or will soon be making a major life change and needs assistance getting organized, making decisions, meeting deadlines, and setting and achieving goals.		
Alternative rock band	Urban, hip, trendsetting, open-minded, cutting-edge, casual, friendly 20- or 30-something person who wants to hear new music in clubs, download music, and/or write/create positive reviews about us for blogs, podcasts, YouTube videos, e-zines, magazines, and newspapers. Also professional, honest, intelligent, open-minded, efficient, friendly, capable, and respected A&R music executive who is looking to sign record contracts with new alternative rock bands.		
Global warming awareness nonprofit organization	Smart, informed, educated, caring, concerned, active, supportive, reliable, and solution-oriented person (of all ages, races, sexes, and religions) who will take steps in his or her life to reduce global warming, as well as take an active role in educating others about this serious issue.		
Start-up greeting card company	Fun, open, honest, witty, creative, professional, 20- to 80-year-old woman owner of retail gift store who is looking to buy new greeting card lines and establish long-term wholesale buying relationships with a start-up greeting card company.		

Determining Benefits to Site Visitors

Now you can put all the pieces together. Taking the site's general purpose, which you create in Book I, Chapter 1, and combining it with the keyword market research you did on the competition and the identity description you just created for the ideal site visitor, you can begin to construct ideas about the tangible benefits to visitors.

Benefits can help persuade visitors to purchase products, use services, tell all their friends about it, and return to the site often. To really understand what those benefits are, put yourself in the shoes of consumers and look at the Web site from their perspective.

Before you design and build any Web site, you should always determine what that site's visitors can gain from visiting that particular site and hopefully purchasing its particular products or services. The benefits of doing this are what can set your site apart from your competitors'. For example, when you're designing a site for a fine-wine and liquor company, if you know that its distinguishing benefits are (a) the number of years it has been in

business, (b) the quality of its products, and (c) its reputation for expertly rating and evaluating the wines it sells, you can highlight those details in the design for the company's site, as illustrated in Figure 2-7. If you don't have this knowledge at the onset of the project, you might encounter design revision setbacks further down the line.



Figure 2-7: Understanding a company's benefits to visitors can help you with the site's design and layout.

Defining the true benefits

A *benefit* is something that is useful, helpful, or advantageous and enhances or promotes health, happiness, and prosperity.

Whereas opinions won't necessarily provide any tangible benefits to the customer, benefits can sway a buyer to favor one product over another. For example, every pizza parlor across the country will tell you it has the best pizza. And to stay competitive in business, each parlor probably has a legion of regulars who will swear up and down that the pizza there really is the best in their neighborhood, town, state, or country. To claim that the pizza is the best, however, is only an opinion.

Having the best pizza in town, while true, might benefit the consumers only if their lives are improved by eating it. Therefore, rather than boasting on a Web site to have the best pizza in town, it makes more sense to market verifiable facts about the parlor — and build those elements into the design — like that it uses the best reduced-fat mozzarella, makes its own low-cholesterol pizza sauce from tomatoes grown fresh at local farms, and is rated number 1 in the ZAGAT survey, and that two slices of its famous salad pizza contain only 390 calories.

What are some of the tangible benefits your Web project's target audience might get from visiting that site? What will make them happy, prosperous, and healthy? What can they find there that is useful, helpful, or advantageous? What would be important to you if you were that visitor? Write it all down.

Taking the visitor's perspective

To give you the experience of taking the visitor's perspective to come up with a solid list of benefits to the visitor, try this next example in which you take a look at one type of business to see how you can convert someone's skills into benefits and clearly state why visitors should want to use the business's products or services.

Here's the scenario: Suppose that your client is a professional digital photographer looking to increase business by putting a portfolio Web site online. She has extensive studio and field experience, has won some important industry awards, has done a lot of fashion shoots around the world, and is willing to travel for the right project.

What's in it for me?

When you purchase a product or service online, the benefits you hope to receive from the item are part of what makes you decide to make the purchase in the first place. Good online marketers know that those benefits need to appear front and center so that you can decide quickly whether a product or service is right for you.

Take a few minutes to visit the following sites to see whether you can quickly identify at least two product or service benefits:

✔ Dreamweaver CS4: www.adobe.com/products/dreamweaver

✔ Firefox: www.mozilla.com/firefox

✔ Epson: www.epson.com (In the Products area, note how features and benefits are highlighted for individual products.)

✔ The Nature Conservancy: www.nature.org

Hint: Benefit statements often begin with action verbs such as *create*, *manage*, and *develop*.

The benefits to those visitors making their way to this client's Web site might include the following:

- ✓ **This photographer is Equipped.** She owns her own studio and digital photographic equipment, so there will be no hidden equipment fees if a site visitor hires her.
- ✓ **This photographer is Accomplished.** Hiring this award-winning photographer means that visitors can feel confident that their projects will have quality results when using this photographer's services.
- ✓ **This photographer is Experienced.** With over ten years' experience in the fashion industry, visitors can rely on this photographer's skills, talent, and professionalism.
- ✓ **This photographer is Global.** She has traveled around the world in the past with *Elle*, *Vogue*, and *Sports Illustrated*, and she is willing to travel anywhere in the world. Visitors can rest assured that this photographer will probably go anywhere for the right assignment.

To discover some benefits that your particular Web site project can offer to its ideal site visitors, try asking yourself what you would want to know if you wanted to do the following tasks:

- ✓ Hire someone who does what your client does. (For instance, your client might be an artist who paints faux finishes for home interiors, a clown who specializes in children's birthday parties, or a private marketing consultant for the knitwear industry.)
- ✓ Find a company that sells what your client's company sells.
- ✓ Find a business that provides services like your client's company.
- ✓ Find an artist with your client's particular skills and experience.
- ✓ Get information about a nonprofit agency like your client's organization.

You can then easily convert the answers to these types of questions into a list of site benefits or a more formal benefit statement. In addition, think about why visitors might want to use the products or services on your client's (or your own) Web site and add those to your list. The better you can predict what will appeal to your target audience, the more you can cater to that audience's wants and needs.

Chapter 3: Gathering Content

In This Chapter

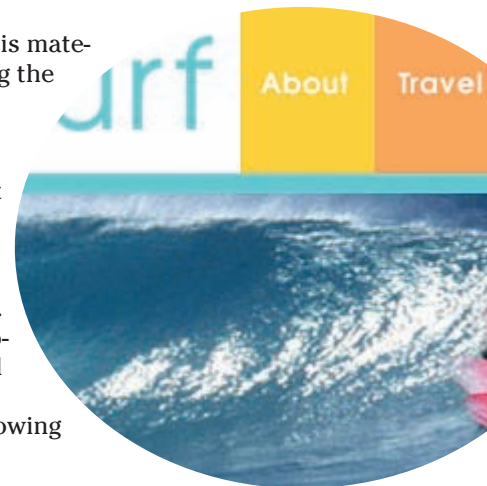
- ✓ **Determining a site's content needs**
- ✓ **Gathering existing content and obtaining new content**
- ✓ **Organizing site content**
- ✓ **Creating a visual site map**

At this point in the process, you've created an identity for the site, have a clear understanding of the target audience, know who the ideal site visitor is, and recognize the benefits that can be offered to site visitors. You're now at the stage where you can begin guiding the Web client in the task of gathering and organizing content for the site in a useful and meaningful way. *Content* includes any text, logos and branding, graphics, photos, illustrations, Flash movies, MP3s, QuickTime videos, plug-ins, and so forth that will appear on the site.

Why would you (and your client) want to determine the site's content needs before you begin working on the design? Because the content can help determine the design, organization, navigation, and layout of the site. You need this information now, before you start the design.

With luck, the client has already prepared a lot of this material for you, and your work is just a matter of helping the client organize that information in a way that's best suited for the Web environment. What's more likely, however, is that the client has only a vague, fuzzy idea of what should go on the site, how that content should be organized, and what a visitor's experience on the site might be like.

Please try not to feel too overwhelmed by this task. Yes, a lot of content may be available, but this chapter is designed to give you the techniques you need to gather the right content for your site. In fact, to make it even easier, I'll start right now with the following guidelines:



- ✓ The client should be responsible for gathering and providing all the Web site content to you. If the client wants your help with gathering and/or generating content, he needs to pay you additional fees for these services.
- ✓ You should have no redundancy in the content on the site.
- ✓ Everything should be logically placed into hierarchical categories.
- ✓ The client must create original work or else obtain the rights for any images, graphics, and photos used on the site.
- ✓ If it doesn't make sense or isn't necessary, don't use it.

To assist you with this large and what some think of as a tiresome task, I've created some helpful tools and techniques for you in this chapter. For starters, you find a series of questions you can ask your client (or yourself, if designing your own site) to help the client generate ideas for content, page order, metadata, and site navigation. Then you discover information and suggestions about where to obtain help with copywriting and editing and where to license or purchase photographs and illustrations needed for the site. The latter part of the chapter deals with a Web site architectural technique called wireframing, and it ends with a few sections on content organization and the creation of a graphical site map of the entire project.

Defining Site Content Requirements

Before you design or build any Web pages, you must first gather everything you can from the client to really lock down what content will go on the site. The reason for this is twofold:

- ✓ You want to know what will go on the site so that you can create a design tailored specifically to the site's content.
- ✓ You need to set tangible boundaries for your client (for example, gather all the content now, and after that, no new pages can be added to the site without incurring additional costs) so that the project can move ahead successfully and stay on schedule through completion.

Gathering content

The content-gathering process includes getting all the text for the pages, logos and branding graphics, photographs (either in electronic format or as images that need to be scanned), illustrations, Flash movies, MP3s, QuickTime videos, PDFs, and anything else that will appear on, or can be downloaded from, the site. Some of these items will already be prepared by the client, but other items might need to be created, licensed, and otherwise obtained. Most of this content-gathering stuff is really the client's responsibility, not yours. However, because many Web clients don't yet

know what they're supposed to do (having never created their own Web site before) and may be unsure of how to do it, your role as the designer might include educating and guiding them through the content-gathering process. Of course, if you're working on a Web redesign project rather than on a brand-new Web site, most (if not all) of the content has already been gathered for you. Still, a redesign is a good time for a virtual spring cleaning, so be sure to check with your client about any content edits, additions, and deletions.



If you're billing your client by the hour, be sure that you include any time spent on these content-gathering tasks as part of your billable services. However, if you're working for a flat rate, make sure that you don't overdo it with the time you put into this part of the process. A few hours here and there to oversee the client should be part of your overall rate. But if the client somehow expects you to do more of or all the legwork, renegotiate your contract with an addendum so that you get paid for any extra time you put in.

Crafting the vision of the site

To begin the process of gathering content, set up an in-person hour-or-so meeting with the client (or schedule a telephone conversation if you can't meet with the client in person) so that you have a formal time in which you can question the client about his or her vision for the site. During this conversation, take careful notes of the client's answers.

The following sections include some of the questions that you might find helpful to ask your client during this client meeting.

In addition to a home page, what other main pages do you want on the site?

Most sites have, at a minimum, both About and Contact pages. After that, the remaining pages depend on the site's purpose and goals.



Because part of the content-gathering task includes helping the client make decisions about what to put on the new site, try to get the conversation started by suggesting ideas, such as which words might be beneficial to use for the labels on all the main navigation buttons. You can then move on to ideas about how to identify each Web page with elements like the page header, the page filename, and the page title. For example, if the client owns a hair salon and wants to have a section on her new Web site for "hair styling trends for long hair and summertime events," you might want to guide her to shortening the main navigation button to something like Styling Trends and then suggest including several subpages under Styling Trends for each trend she'd like to discuss on the site, such as Men's Summer Styles, Women's Summer Styles, and Summer Weddings.



Pay close attention to what the client says because he or she might not realize the long-term implications of the choices that are made now. In the case of the hair salon example, be sure to ask the client whether she really meant that she wanted to offer only summer trends or if, in fact, she wants to display new trends each season, which would mean the navigation for those subpages would need to be updated quarterly. If she wants to display only summer trend information, she needs to be okay with the idea that for a substantial part of the year, that summer trend information will be irrelevant. With that in mind, the client can then decide whether she truly wants to update this section four times a year, for each of the seasons, or change the navigation and subnavigation sections to simply Trends for Men, Trends for Women, and Wedding Trends, leaving off any reference to the season. As you can see, the latter option is more generally labeled and would still allow the client to update the page content to match the season without having to edit the navigation. Alternatively, updates like these might help you decide to build the site dynamically, using a database or some type of Content Management System (CMS).

Do you want an About page or section to provide company information?

Find out what the client would like to call this page/section — About, About Us, or About Our Company? What subpages, if any, might go in this section? How about a page for listing a Board of Directors, Donors, or Sponsors; a Staff Directory; a Mission Statement; a Corporate History; Testimonials; or some other information?

What do you want to include on the Contact page?

This presumes, of course, that the client wants a Contact page. While it is highly recommended, some of your clients may want to break the rules by including contact information in the footer of all the pages rather than dedicate an entire page to listing an address, e-mail, and phone number, as shown in Figure 3-1.

However, for everyone else, the Contact page might list a physical address, telephone and fax numbers, and a single contact e-mail address or several department-specific e-mail addresses. In addition, this page may have a contact form to collect contact information, comments, questions, and feedback from visitors. If you have a lot of contact information to impart to visitors, this page may warrant having subpages for details like transportation information, directions, maps, or other facts, as well as retail and wholesale buyer information and sales representative contact details.



Figure 3-1: Forgoing the traditional contact page and putting contact information in the page footer.

Do you want to have a Clients page that lists past and current clients?

A Clients page might contain a list of past and current clients, including links to recent projects and links to client Web sites, and all this data can be listed in alphabetical order, by vertical industry, or by project type. Some client pages also include client case studies, and if so, you want to find out how many case studies will appear in this area and whether they'll be bunched together on a single page or displayed on separate pages. Before posting any client information, be sure to get approval from each client. You might get the occasional “No, thanks,” but most clients say yes in exchange for the free publicity.

If you're selling products, do you want to have a Products page?

When considering this page, discuss whether the products will be merely described or both described and sold on this site. Be sure to find out how many products will be displayed and whether multiple product categories will be shown, because the decision to use a database may have an impact on how the pages need to be designed. For instance, having a small number of products gives you the freedom to design each page in a different way, whereas having a large number of products often begs for a data-driven template.

The Products page may include subpages, such as a product detail page and technical details page for each product. All these factors impact the content you need to get from the client. They also help determine (some-what) how the pages need to be designed.

Will you have an e-commerce component on the site and, if so, what kind of shopping cart will be used?

Will the cart be included in the hosting plan, need to be purchased as a third-party software application, use PayPal to avoid using a merchant account, or be custom-built by a programmer specifically for the client's products? Knowing this information in advance can help you organize and design a site that's optimally suited to the products or services being sold.

Do you need a Services page to list all the services your company provides?

How many services will there be? Does each service warrant its own subpage, or can all the services be listed together on a single page? Do any of the services require diagrams or other graphics to support them? Will you list the pricing of these services on the site? Can services be purchased online?

Do you want a page that describes what the company produces?

For example, this page might be titled Our Work, What We Do, or Portfolio. You can break down this page into categories that match the client's particular offerings, such as Planning & Urban Design, Landscape Architecture, and Interior Design; or Illustrations, Paintings, and Sculpture. Find out how many sample images, if any, will be shown in each category and how the client wants these images to be displayed — in a slide show, as thumbnails that link to larger close-up images, or in some other format.

How about a page for news or press releases?

How many news items and/or press releases will you have, and how often will they be updated? How will the list of news items or press releases be displayed? By date, by topic, or by title? Will they need to be sortable by the site visitor (which means that you'll need a database)? If a lot of news items will be posted regularly, does the client need a news archive?

What about an Events page?

What kinds of events will there be? How often will the Events page be updated? Enough to justify having subpages in this section? Will events need to be presented by separating them into current/upcoming and past

categories, or will only upcoming events be listed? Does the site warrant having some kind of interactive calendar?

What about other industry-specific pages?

Do other pages on the site call for having their own sections, with or without subsections? In other words, are any pages or sections so important that they should become part of the main navigation, or can they be logically tucked into some other section of the site?

What about including an RSS feed?

Do you want to allow visitors to subscribe to an RSS feed so that they can be notified automatically of new events, news items, products, and press releases? Setting up an RSS feed is quite easy to do. Most news-related sites, blogs, and ezines often syndicate their site content through an RSS feed, but any site that has relevant news and content to offer visitors can create one.

Will the site need any other pages?

Some pages don't need main navigation links but should still be included on the site, which means that they'll need to be accessible through footer links on every page, through the site map page, or through regular text links on other pages throughout the site. Examples might include pages for Articles, Links, a Résumé, Partners, Affiliates, Terms of Service, Privacy Policies, FAQs, a Site Map, or a page or link that links to an external blog.

Does the text on the site need to appear in multiple languages?

If so, how many other languages do you need and what are they? What technology will be used to create these sister Web sites? Does the client need to purchase translation software or hire a translation service? If you expect a lot of pages, will each site be separate or does the client prefer that the content is organized with a database and displays dynamically based on a visitor's language preference? How can visitors switch between languages on the site? One idea, as illustrated in Figure 3-2, is to allow visitors to switch between languages at will using a clearly labeled "View this site in Spanish" or "Lea este sitio en inglés" graphic link.



Finding a good translation service may be difficult because services vary in quality, ability, and price, and because the translation service providers' sites, although helpful, aren't always the best indication of quality. Though I have no personal experience directly hiring a translation service, three services that seem to have a good reputation, competitive pricing, and nice Web sites (in my opinion, a big factor in determining whether to do business with that company) are Transware.com, FreeTranslation.com, and VerbatimSolutions.com.

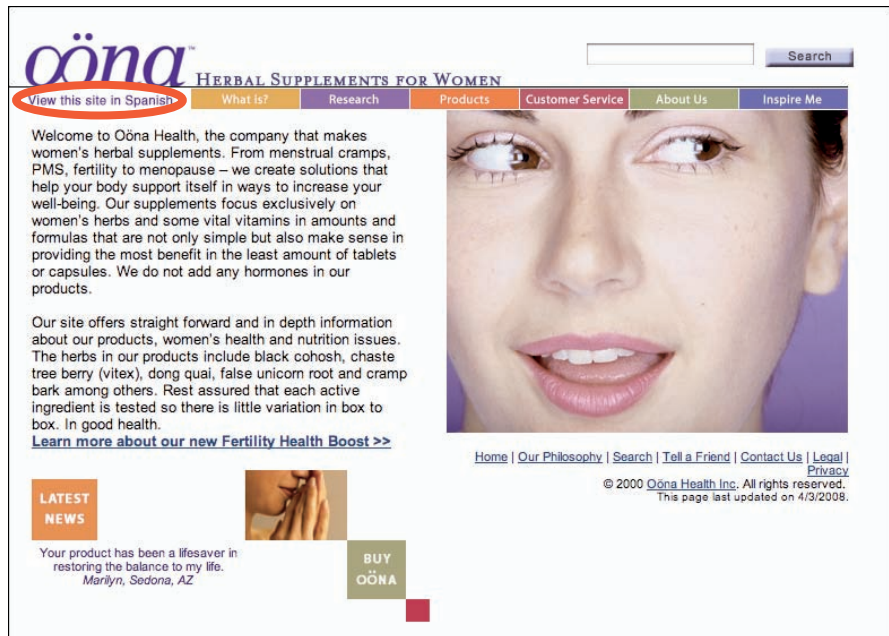


Figure 3-2: When you present a site in multiple languages, give visitors an easy way to switch between languages.

What elements — text and graphical — should appear on every page?

Definitely include any branding information, including logos, tag lines, and the main navigation links in the same location on every page. However, what other things should be accessible from anywhere on the site? How about Join Our Mailing List, Location/Store Finder, Site Search, Request a Brochure, Help, Customer Service, My Account, Login, Online Chat, an e-mail address, or a toll-free telephone number? These items can be graphics or text links anywhere on the page, such as at the upper-right corner of the page or down in the footer, depending on their importance.

If the site’s design includes one or two sidebars on most of or all the pages, elements like these can be placed there. Sidebars refer to a section of a Web page, usually along the left or right margin, that contains content separate from the main body text of a page, often used for subnavigation, advertising, feature articles, and other site content that the site owners want you to pay particular attention to. The sidebars typically have different background colors, borders, text treatments, and graphics to visually distinguish them from the main content area. The repeating elements on the CouponConnector.com Web site, shown in Figure 3-3, include a logo and tagline, a site search form, and navigation and footer areas.

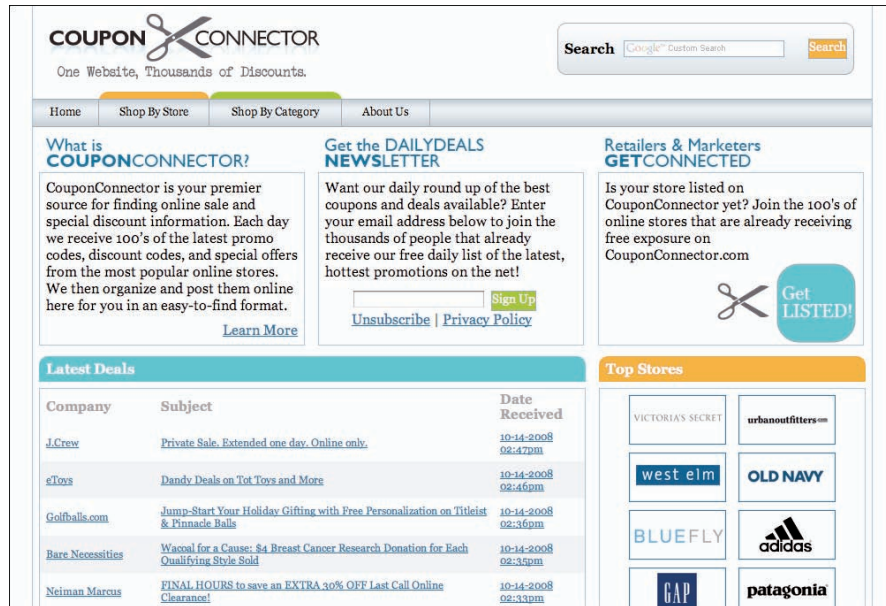


Figure 3-3: Repeating elements, like the logo and a search form, should be in the same location on every page.

Who will provide all the graphics, photos, and illustrations for the site as well as branding information?

The branding information includes logos and other brand-identity graphics. If the answer to this question isn't "the client," you should definitely find out about some of the resources discussed in the "Hiring freelance artists" and "Licensing stock images" sections, later in this chapter.

Does anything else need to go on the site?

Make sure that you've covered everything the client or group you're working with has in mind for the site. Be clear with your client that there should be no last-minute surprises; otherwise that could create both timeline delays and additional charges. Other content might include a photo gallery, special password-protected members-only section, bookstore, class listing, or registration form. Every detail must be thought out in advance and planned for.

At the end of the conversation, thank the client for his time and let him know that there is still much to do during this stage of the Web design process with regard to gathering content, the responsibility of which mostly falls on the client's shoulders before you begin the actual design.

Concurrent with gathering the content for the site — which may include generating new content as well as culling content from existing company sources and/or hiring a copywriter — the client needs to produce, acquire, and/or license photographs and illustrations for the site.

In addition, after the content areas have been discovered and outlined, all that information must be organized to assist with creating a layout for the site. This can be done through the creation of *wireframes*, as you read about in the next section, to help identify what content is needed for each page on the site. Oftentimes, putting everything down on paper helps the client better envision what content to display and how the site will function.

Building Wireframes

Creating *wireframes* is an interesting technique that helps many clients when they're trying to determine what content to put on each of the individual pages of their Web site. In addition to understanding what each of the pages should contain, a wireframe helps site owners plan how their sites will function and gives a general overview of what the layout might look like.

As you can see in Figure 3-4, a wireframe is a text-only diagram, or blueprint, of a particular page in the Web site, and it typically includes the following elements:

- ✓ **General site navigation:** Using text only, the wireframe diagrams the navigation and subnavigation elements, buttons, links, form fields, and other functional elements that show or describe how the visitor can enter and exit the page.
- ✓ **Content that appears on every page:** In addition to the navigation tools, the site may include components such as corporate identity and branding, search boxes, mailing list or newsletter signup buttons, links, or form fields, page headers, page footers, and other page elements.



A *page header* is a word (such as Contact or Contact Us), phrase (such as About Our Services), or short line of text (such as Sign In to Use the Control Panel) that identifies the content that can be found on the page and is placed above or away from the main body of text on the page. Page header text can be formatted with CSS using the `<h1>` through `<h6>` tags, or it can be a graphic that includes text and is inserted onto a Web page in roughly the same location to serve the same purpose.

- ✓ **Interactive components:** This can include hyperlinks, rollover buttons, navigation menus, hidden layers, games, rotating banner ads, rotating graphics, photo galleries, slide shows, animations, video clips, movies, and other multimedia files.

- ✓ **Dynamic functionality:** Any data, images, or other site content that will be dynamically pulled from a database and displayed on the Web page should be indicated on the wireframe to assist with the site layout and programming.
- ✓ **Content for a particular page on a Web site:** For each individual wireframe, include the page title, placeholder graphics, and text for that page (or if text is unavailable, “dummy” or “greeking” placeholder text can be used instead). Greeking text looks similar to Latin (refer to Figure 3-4) and is often used to show how text in a particular font will fill the desired space within the layout. It is also sometimes used as placeholder text in graphic layouts until the real content becomes available. Greeking text usually begins with the famous “Lorem ipsum dolor sit amet . . .” and looks similar to English in its word size and distribution within sentences.

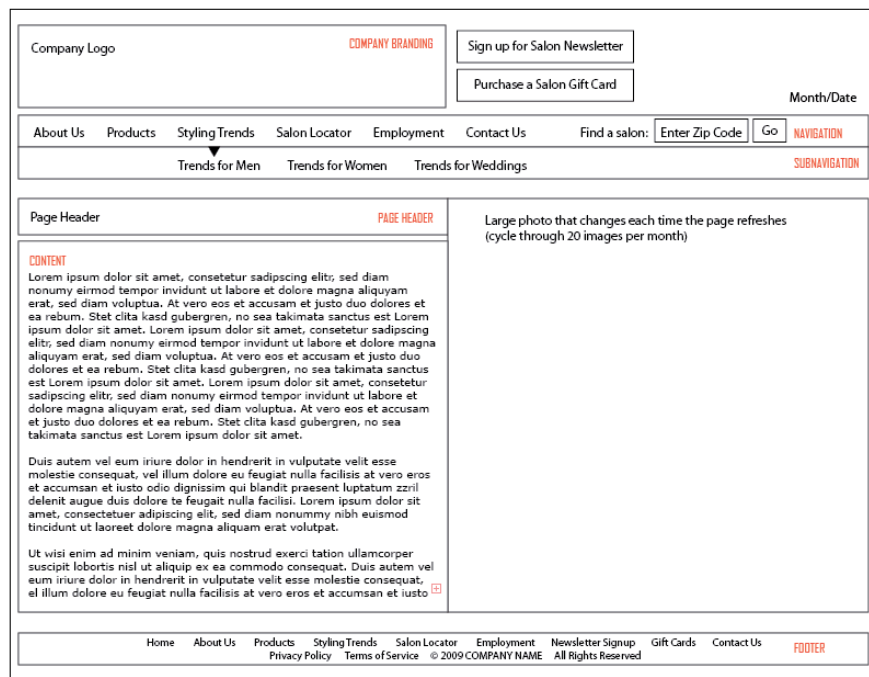


Figure 3-4: Wireframes are a text-only depiction of a Web site’s navigation, interactive features, and content areas.

All these elements will be replaced with actual graphics and text on the Web page. For example, your client might include notations such as “About Us page content” and “Boardroom Photo” within the wireframe, as shown in Figure 3-5.



Figure 3-5: Greeking text and dummy graphics can be used in a wireframe.

Because the making of wireframes is primarily the responsibility of the client, and most of your clients will have never heard of wireframes before, creating them often gets overlooked at this stage of the Web site’s development. Make it part of your job to educate your client about how she can use wireframes as a time-saving, cost-effective, content discovery tool, and consider creating a sample wireframe for the client’s home page as your special Web design value-added service. Who knows? The client might appreciate it so much that she’ll consider paying you a little extra to help create additional wireframes for the rest of the pages on the site.

Because the wireframe typically contains only text, one of the key benefits of making wireframes for all the pages on the site (or at least the most important ones) is that the client can focus on user experience and content without having to think about design and layout decisions or coding and development issues. That stuff comes later. It is only after the content has been gathered and the wireframes created that the client should tackle what actual text the visitors will see on each page.

To create the wireframes, your client (or you) can draw them by hand on plain or graph paper, mock them up in a graphics program like Adobe

Illustrator (as was done in Figure 3-5), or use any other application that allows the insertion and arrangement of text. For instance, you might consider using one of the new interesting online collaborative wireframing tools, such as ProtoShare.com and SmartDraw.com, which range in price from \$25 to \$500. You could also consider purchasing one of the software applications on the market, such as Microsoft Visio or Axure RP Pro, that are specifically designed for creating HTML wireframes. These programs contain Weblike buttons, form fields, menus, and other page elements that can easily be dragged onto the wireframe diagram, making it easier for the client to foresee the content and its functionality predesign. Or, if your client would rather go low-tech due to budgetary or time constraints, he or she can certainly create wireframes in a simple word processing or spreadsheet program such as Microsoft Word or Excel, or simply draw the wireframe by hand on a whiteboard and then take digital photos of it.



You don't need to create wireframes for every page on a Web site unless this is something the client expressly asks for and is amenable to compensating you for. A more likely scenario is that you or your client will create a wireframe for the home page and possibly one or two additional wireframes for any subpages with distinct content and layout requirements, such as a shopping cart page, a product detail page, or a search results listing.



The wireframing technique is meant to assist the client with any content-gathering issues he might have for the site. Similar in concept to a book's table of contents, the wireframe merely outlines or hints at the content in a particular section, not how it is supposed to look. With the wireframe, the client can focus on the general rather than the specific by noting what information should be displayed, even when the information (such as text and graphics) isn't available or hasn't been created yet. For example, if your client will be selling products on his Web site, he should plan to have at least a main products listing page and a product details page, both of which can be wireframed to show how the individual products and product details will appear in the browser. In the absence of real content, the details page wireframe can include information placeholders like the company logo, content, and page title areas.

Gathering Text and Graphics

If you're lucky, your client is likely to have some or perhaps all the textual content for his Web site already created in the form of existing newsletters, press releases, brochures, and other marketing collateral. Ask the client to start pulling all these materials together for the site, as well as to generate any new content that is needed and to rewrite any existing copy that needs updating to fit into the specific pages you discussed in the content-gathering meeting.

When the site will use a database and have some kind of dynamic functionality, the client would also be wise to begin entering that data into an Excel spreadsheet for later importing into a database of some kind. Because each dynamic site uses a unique combination of coding, it is very important to know in advance how the data should be entered into the spreadsheet (how files are named and numbered), because that may depend on the programming language (such as ASP, JSP, PHP, or CFML) chosen to make the site dynamic, and this decision will most likely be made by the person who is programming the site, whether that's you or a contracted programmer. For instance, to ensure that the wines are listed appropriately on their site, as shown in Figure 3-6, the folks at RockwoodandPerry.com must have the name, manufacturer, size, bottle and case price, and description for each wine in their rotating inventory. Making this decision now can save everyone time later on when you need to import data from the database into the Web pages.

ROCKWOOD & PERRY
Fine Wine & Spirits
Since 1982

Cellarers, Direct Importers & Vendors of Fine Wine and Spirituous Liquors

VINTAGE: (ALL) REGION: (ALL) FIND Search by Item/keyword FIND

FIND & BUY WINE ACCESSORIES WINE OFFERINGS ABOUT US QUESTIONS: 800-281-0260

ROCKWOOD & PERRY
STAFF RECOMMENDATIONS

Staff Recommends

At any given moment in the wine flow there is a small, select group of wines that deliver way more quality for their price than is normal. Our staff knows where and who they are because they are tasting, talking, enjoying and generally over-involved in wine. So here and now we propose a list of these wines from our personal experience. You will find them to be more intense, complex and interesting than the general run of wines at the same price point.

Sort by: Select One Page: 1

Luigi Bosca 2002 Cabernet Sauvignon Reserva, Mendoza, Argentina
Scents of blackcurrant, blueberry and graphite...notes of chocolate and white truffle...blackfruit flavors...heated by a good dollop of alcohol...unusually long.
Rating: 90 R&P | Size: 750ML
Bottle Price: ~~\$18.99~~ \$15.99
Case Price: ~~\$205.09~~ \$186.96
Qty: 1 Bottles Cases
ADD TO CART
.: IN STOCK.:
[LEARN MORE](#)

Ch. Hauchat La Rose 2003 Fronsac
Woodsmoke, mineral earthiness...slowly evolving cassis, cherry kirsch scents...challenging but approachable tannins...blueberry, blackcurrant fruits.
Rating: 90 R&P | Size: 750ML
Bottle Price: ~~\$28.99~~ \$23.99
Case Price: ~~\$313.09~~ \$279.96
Qty: 1 Bottles Cases
ADD TO CART
.: IN STOCK.:
[LEARN MORE](#)

JOIN OUR E-MAIL INSIDER LIST
Get first access to hard-to-find, cheap high-rated wines...
[SIGN UP NOW](#)

ASK OUR EXPERTS
What is the best wine to serve with lamb?
Get advice from our experts about buying, serving, storing and drinking fine wine...
[ASK A QUESTION NOW](#)

Figure 3-6: Some database-driven Web sites use spreadsheets to update their inventories of products.

Likewise, if you know that you will need to pull together text and graphics from someone other than your client, now is the time to look into hiring a copywriter and freelance artists, licensing stock images, and generating appropriate page titles and metadata.

Hiring a copywriter

While you may have some of your client's Web site content a little ways into the process, some other items for the site might simply not exist yet. In such cases, the client needs to decide whether to write the missing information or to hire someone to do it. If you're confident in your copywriting skills, you might want to offer your services for an additional fee. However, if writing isn't your thing (or if the client rejects your offer to do copywriting and the client isn't capable of it), he needs to hire an outside service, either through an agency or online through a job-listing site or writing organization.

The more promising-looking national copywriting services include Freelancewriting.com, ScribeGroup.com, Guru.com, Elance.com, and WriterFind.com; however, you (or your client) should unquestionably still search online to see whether you can find someone to do the work locally. For example, if you are looking for a Web copy editor in a particular geographical region within the United States, visit www.freelancedesigners.com/dir/writers/ (shown in Figure 3-7) where you can find regional listings of copywriters, many of whom have Web copyediting experience.



Figure 3-7: Locate a Web copy editor in your area by checking an online directory.



Above all in your search, be skeptical: Not everyone who thinks he can write can really do a good job. Ask for writing samples, see whether you like the way they communicate in print, and compare fees to ensure that you get the best service for the price.

Hiring freelance artists

Despite how creative noncreative people may think they are, you should, without hesitation, insist that your Web client use professional artists, illustrators, and photographers to create any custom visual graphics and photographs that will appear on their site. The only exception is if the client is a graphic designer, illustrator, photographer, or artist, in which case you may welcome their graphics and other artwork.

Ask the client whether he intends to use established contacts for the graphic art that will go on his site. Some clients already have an established relationship with other graphic designers, artists, illustrators, and photographers and will gladly get any needed site graphics, such as icons and illustrations, directly from those people. Other clients, however, will look to you for direction on these things. If you are an artist, illustrator, or photographer as well as a Web designer, definitely offer your services to the client for an additional fee. Otherwise, be ready to suggest some local artists, illustrators, and photographers who would be happy to create the needed art and photos.

Of course, you could leave the hiring of such people solely to the client, but many times the client will want or need you to take on this responsibility because you are the one with the vision for how the site will look and function. Fortunately, your role in this part of the content-gathering process can be as big or as small as you feel comfortable with. For instance, you might suggest that the client buy a digital camera and create her own photographs. Likewise, you may recommend that your client purchase or license stock art (see the next section in this chapter) for the site. Or, you may simply want to steer your client toward a particular handful of artists you have worked with before and are confident in their services.

Whatever the client's preference turns out to be, create a list of freelance artists, illustrators, and photographers that you like and keep this list handy. You can then choose to selectively contact people from the list on a project-by-project basis, or simply turn your list over to your client and let her vet, select, and hire someone from your list.

To generate this list, you need to do some homework by doing any of the following:

- ✓ Post ads looking for artists on Craigslist.com or some other local online community Web site that has a job board.
- ✓ Search for specific artists on sites like Directory of Illustration (<http://directoryofillustration.com>).
- ✓ Join the local chapter of the Graphic Artists Guild (<http://gag.org>).
- ✓ Join a MAC user group (www.mugcenter.com or www.apple.com/usergroups).
- ✓ Join some other local arts organization where you can network with other artists, illustrators, photographers, and designers.

You may even find some up-and-coming artist types through a local university; lots of student artists are eager to work for a reduced fee — sometimes even for free — while they build their portfolio.

Above all, remember that each project is unique, so be open to making suggestions and let the client decide what works best for her, given her project's time frame, needs, and budget.

Licensing stock images

For your fixed-budget and quick-turnaround-time projects, a good alternative to hiring a freelance artist is to either license or purchase stock imagery directly from an online service. You have two general options with stock images:

- ✓ **Royalty-Free (RF):** Royalty-free photographs, illustrations, and other artwork are images that you purchase, usually for a one-time fee, to use for a given project, as long as that project does not include the reselling or redistribution of the image in some other form. Some stock art services charge different rates for royalty-free art depending on its size, quality, and intended usage. With all royalty-free artwork, the image being licensed still belongs to the creator of it under U.S. property and copyright laws, which means that if you were to use the image in some unlawful way, the creator of that image could sue you and you could be fined, jailed, or otherwise punished for any improper usage.
- ✓ **Rights-Managed (RM):** Rights-managed images are licensed photographs and other artwork that you can purchase from an online service or agency (or directly from the creator), with the exclusive usage rights for a given period of time. While it is a much more expensive option than royalty-free stock art, rights-managed work cannot be used by anyone else, competitors included, until the contract period ends.

Respecting the copyright

Before you search for images to use on a Web site, you should know a little about copyright protection laws. All the images that you see online — whether they're on a royalty-free image site, in an online store, or in somebody's blog or Flickr account — belong to somebody. Someone took the time to take that photograph, draw that illustration, build that animation, and design that icon. Savvy art makers have registered their own custom artwork, illustrations, and photographs with the U.S. Copyright Office. Likewise, smart Web site owners make the necessary arrangements to legally license or purchase the work of others. For the rest of the world, what are the implications if you or someone else copies an image from another site, without permission or payment, and then uses that image for another project? What rights might have been violated? The answers depend on your usage and intent.

Suppose that you go to the Disney Web site and see a picture of Goofy that you love and want to turn it into your desktop wallpaper so that you can look at it every day. In this case, your usage

intent is for private use and isn't for profit; therefore, your use of that Goofy image isn't harming anyone or earning you any income. Disney knows that many visitors might use its graphics for personal use, so it includes a clause in its Privacy Policy and Terms to allow that, but it forbids visitors from taking images from its site to use in any way for profit. Of course, it has no way of tracking this information because no reliable way of accurately monitoring images that are copied from the Web site exists. Nonetheless, by stating its rights on the Web site, Disney is protecting those rights to the images in case it needs to file suit against someone who knowingly violates the copyright.

Therefore, if you need an image for your site — say a photograph of a sunset on Waikiki beach that you use as part of the home page design for a luau event and catering company based in Los Angeles, California — the right thing to do is to create it yourself, hire someone to create it for you, or license the image from one of the many stock-art or clip-art Web sites.

Both royalty-free and rights-managed images can be purchased online. For projects that require several images along a particular subject, consider purchasing an entire CD of royalty-free images on a particular theme, such as “diet & exercise,” “architectural marvels,” or “growth economy.” Most CDs cost between \$299 and \$799 for anywhere from 50 and 100 images.

For projects where you only need a handful of images on a particular theme, single images can be licensed from several royalty-free stock art Web sites for as little as \$1 per image, depending on the image size and quality, license time frame, and usage parameters. Corbis.com is the leading provider of rights-managed and royalty-free stock photography and illustration. Eyewire.com, Veer.com, and iStockphoto.com are also reputable services with similar price structuring. For those looking for a less expensive, purchase-what-you-need option, check out two of my personal

favorites, Fotolia.com and Dreamstime.com, shown in Figure 3-8, as well as ClipArt.com, FotoSearch.com, Comstock.com, GettyImages.com, RoyaltyFreeArt.com, Shutterstock.com, and Inmagine.com.

Figure 3-8: Find interesting and affordable stock art at sites like these.

Another useful source of stock art comes from regular folks from around the world who have decided to license their work through Creative Commons, a nonprofit organization (<http://creativecommons.org>) that offers an “alternative to full copyright.” For instance, if you visit www.flickr.com/creativecommons, you can find graphics for your projects that have Attribution, Noncommercial, No Derivative Works, and Share Alike copyright assignments.

Another great alternative is for you or your client to subscribe to one of the royalty-free online image services, such as AbleStock.com, JupiterImages.com, and PhotoObjects.net, where, for a flat rate, you can download an unlimited number of images on a monthly, biannual, or annual basis.



The only real drawback to using online royalty-free images is that they carry a high likelihood of other businesses — including potential competitors — having also selected and licensed the same images for their projects.

If you want to go the super DIY route (which means, of course, that an even higher likelihood exists that someone else might use the same images), consider buying a box of royalty-free clip art on CD. You can find something like Nova’s Art Explosion 300,000 Clip Art and Nova’s Art Explosion 800,000 Clip Art on Amazon.com for anywhere from \$29.99 to \$109.99. If you decide to do this, just be sure what you’re purchasing will work on your computer and operating system, because some clip-art CDs are PC-only for Windows NT/95/98/2000/Me/XP, but not for Vista.



Wherever the client decides to get images for his site, be sure that you, as the designer, add only legally licensed artwork to your design for the site. Ethically, you should also be sure that the images provided by the client are also legally licensed. In other words, don’t make yourself liable to any copyright infringements made by your client. In fact, you might want to add to your contract some kind of Permissions and Releases regarding Copyrights and Trademarks clause. This clause could state that the client agrees that she either own the rights to all images provided to you, or that she shall be responsible for obtaining all permissions and rights for the lawful usage of images created by others for use on her site. The clause should also clearly state that you, the designer, will be held harmless for using any and all client-provided images in your design, should they be found to be unlawful. For more information about copyrights, trademarks, and fair use, visit the Web sites of the U.S. Copyright Office (www.copyright.gov) and the U.S. Patent and Trademark Office (www.uspto.gov).

Choosing page titles and meta-tag data

In addition to all the text, photos, illustrations, and other graphics that will appear directly on the pages of your Web site, you also need to acquire or create two other important bits of information, namely unique page titles and meta-tag data, which go into the HTML code of the site’s pages.

If you're creating a site for someone else, ask your client to provide the titles and metadata to you. Otherwise, if you're designing a site for yourself, you can easily create this content.

Page titles

Page titles are pretty self-explanatory. Each page on a Web site needs its own unique title, which after being placed appropriately in the head area of the HTML code, will appear in the browser window's title bar. Page titles are set between opening and closing `<title>` tags in the HTML code of a Web page. Each page on a Web site should have its own unique title because this information assists Web crawlers in indexing pages and entire sites. For example, if you go to Amazon.com, the title of the home page is "Amazon.com: Online Shopping for Electronics, Apparel, Computers, Books, DVDs & more." As you can see in Figure 3-9, the title appears at the top of the browser window.

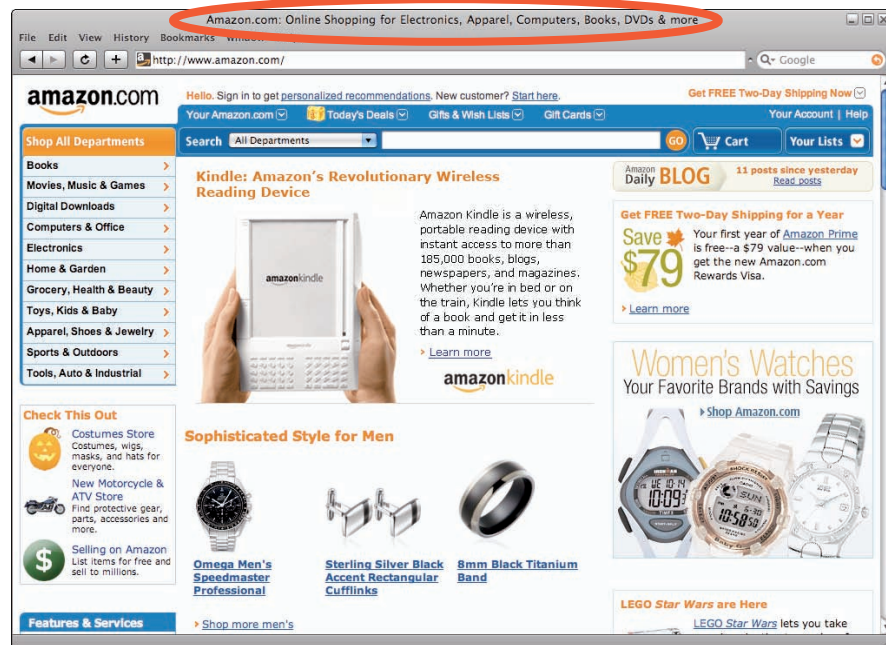


Figure 3-9: The title appears in the browser's title bar at the top of the browser window.

Titles need to identify the site as well as the content displayed on each page by using a maximum of 70 to 80 characters, including any letters, numbers, symbols, spaces, punctuation, or other text entities, such as A, 23, ©, %, and #. For example, "NY Bests: Best Breakfasts for Under \$6 in New York City"

would be a good title for a Web page called NY Bests that had a page with reviews of restaurants serving cheap breakfasts. Titles longer than 70 characters might get truncated, or cut off, by the browser, so if titles are a bit longer, put the most important words in the first 70 characters. When writing your page titles, a good rule is to pack each one of them full of “key-words” that visitors are likely to use when searching for pages that contain desired information.

Meta tags

Meta tags are special HTML tags that also go inside the head area of the HTML code on a Web page, but rather than appear somewhere on or in the browser window, this information is strictly used by search engines, indexing spiders, and robots, and is hidden from the visitor’s view. Specifically, these tags provide search engines and Web browsers with informational content about the client’s company and code settings, which helps those search engines rank the site in search results listings and helps those browsers accurately display the code inside the browser window.

Keep in mind that meta tags aren’t magical search engine ranking-improvers; on the contrary, they’re merely assistive code bits that can help people find your site. Truth be told, Google and some other search engines don’t use this kind of information as much as some of the other search engines, spiders, and robots do. Nonetheless, even though they aren’t as important as they once were, using them is still worthwhile.



You may be wondering why it might be important to ask for this data from the client at this stage of the site development. The answer is because the client is focused right now on getting the content to you, and he will be more likely to comply with your request now rather than later. In my experience, if you wait until you start building the site to ask for titles and meta tags — even though realistically it should only take your client about ten or so minutes to generate them — the chances of you receiving them are slim.

While a lot of different kinds of meta tags can be added to the HTML of the pages, only two tags should be placed on every page of a site — the meta description and the meta keywords:

- ✓ **Description:** The description of the site is a single sentence that concisely describes what can be found on the site. Normally it should be written as a single sentence or short phrase and should contain no more than 150 characters, including spaces and punctuation. This information is critical, because this exact copy is often (though not in all search engines) what appears in a search engine results listing when the site is found after a keyword search. As a matter of fact, you may even want to

create unique descriptions for each page of your site to further assist visitors in finding relevant information on the site's different pages.

- ✓ **Keywords:** While many Web crawlers ignore this meta tag, some still do use this information, so you may as well plan to include keywords in the code of your pages. Keywords should include the words or short phrases that you feel would be helpful when searching for this site. Keywords (and/or key phrases) should be separated by a comma either with or without a space, such as "bread,cakes,pies." or "bread, cakes, pies." Place the seven most important words first, in order of importance. Any more than seven words will probably be ignored.

If you'll be coding the meta-tag information yourself, be sure that you understand the proper use of syntax, because if you code it incorrectly, it won't be read by the search engine crawlers and spiders. To illustrate proper coding, here's an example of the meta-tag HTML code used by Oceana Surf, shown in Figure 3-10, a private surf instruction and surfing retreat company based in Santa Monica, California:

```
<HEAD>
<title>Oceana Surf: Lessons, Surf Camps, Women's Travel, Retreats</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<META NAME="keywords" CONTENT="surf lessons, learn to surf, santa monica,
  malibu, surf retreat, surf travel...">
<META NAME="description" CONTENT="Beginning through intermediate year-round surf
  instruction, from Santa Monica to Hawaii, Costa Rica to Indonesia. Private
  one-on-one training. Includes equipment. Los Angeles: 310-392-9112 info at
  oceanasurf.com">
<META NAME="Publisher" CONTENT="Oceana Surf">
<META NAME="Copyright" CONTENT="Copyright, Oceana Surf. All Rights Reserved.">
<META NAME="Author" CONTENT="oceanasurf.com">
<META NAME="Language" CONTENT="en-US">
<META NAME="Robots" CONTENT="All">
</HEAD>
```



The limited power of keywords

If you do any online research about meta tags and their usage, you'll quickly discover that keywords are much less useful on a site than the description. This is because, for a time a few years ago, keywords were widely abused by unethical site owners looking to improve their Web site rankings and increase site traffic by padding their keyword meta tags with irrelevant search terms. Soon after this abuse was discovered by the major search engines, the

importance of keywords quickly diminished. Today, only a couple of search engines still use a database that supports the usage of keywords as a ranking tool, but in my opinion, even one is enough to justify keeping the keywords listed in the code. As an alternative to listing the seven most important keywords in the meta keywords tag, some designers simply duplicate the meta description content there instead.



Figure 3-10: A good meta description should help search engines properly index the page for potential site visitors.

As you can see, many different kinds of meta tags can be added to the head of a Web page, but these are less critical to the function of indexing the site on a search engine and instead are used more for informational purposes should anyone decide to look at the code. In addition to the tags shown here, many others can be used for a variety of purposes. For instance, one meta tag can forward a page to another URL, and another can prevent the browser from caching the content on the page. You find out more about meta tags in Book III, Chapter 1, including how and where to add them to your Web pages.

Organizing Site Content

Depending on how organized you are, you can use several methods to arrange the content — the text, logo, branding graphics, photos, illustrations, Flash movies, MP3s, QuickTime videos, plug-ins, and so forth that will appear on the site — in a meaningful and useful way. The best practice, of course, is to have the client begin to organize the content for you before he hands it over to you. Notice I say *begin to*. That's because the client isn't always the best judge of how to effectively organize everything. The client might, for example, want to have a staff contact page on the site that lists the names, telephone numbers, and e-mail addresses of everyone in the company and might think this information should go on the site as part of a Company History page under the About Us section. That just doesn't make sense. You have to step in and think about the site from a visitor's perspective. Where would you expect to find this information? A more logical location for a staff directory would be as a separate subpage under either the About Us or the Contact Us page.

To begin your part of the content-organization process, follow these general steps:

1. Write down all the names of the main pages across the top of a sheet of paper.

These represent the main navigation buttons or links that will enable visitors to navigate their way through the site. For example, the main pages might include Home, About, Services, Testimonials, and Contact.



If the site you're working on is fairly large in scope and breadth, another useful technique for content organization is to write all the page names on individual index cards and then, on a table or other flat surface, order the index cards into the main navigation and subnavigation categories. The final order of the index cards can then be transcribed on a single sheet of paper.

2. Look at all the other pages of content that the client has provided to you, and decide where each page would best be placed relative to the main pages.

This way, you can both establish logical categories and subcategories of information, and create the navigation scheme that you will use when building the site.



During this process, take extra care not to make too many levels of subnavigation in your page organization. Three levels should be your maximum depth (that is, main, subnavigation, sub-subnavigation). In fact, an extremely important Web usability principle states that it's better to have a wide navigation than a deep navigation. In other words, rather than embed subnavigation that has subnavigation on top of subnavigation with even

more subnavigation (deep), it is more user-friendly to site visitors to keep most of the content close to the top level (wide) so that *all the content on a site is accessible in no more than three clicks from the home page*. This technique works great for most sites with less than ten main categories of information. Over ten and the site might need to use a directory type of navigation system, where users can click a category (think the Yahoo! home page) with hypertext links to narrow what they're looking for and can then be taken directly to that section of a larger site.

For your smaller Web site projects, organizing the content is much simpler and quicker because most sites have at least three of the main pages in common: Home, About, and Contact. However, for your medium-sized sites — over 15 pages, say — exactly where in the site architecture each of the main page and subpages should go becomes a matter of logical organization. Try to arrange the content from the top down, starting at the home page; then move on to the main navigation categories and finally the subpages. If you happen to run across a page of content that doesn't seem to have a logical place to fit on the site, ask the client where he thinks it should go and be prepared to make a suggestion to the client about where it might fit in. If there truly is no logical place for the content, the site might benefit from combining that content with the information on another page that already does fit within the site architecture. Otherwise, that content may need to be removed from the site. Just because the client thinks something might be a good idea doesn't necessarily make it so. What matters is the visitor's experience on the site. The site should be well organized, easy to navigate, and only contain information that is interesting and meaningful to site visitors.

As a little practice test, take a few minutes to try assembling the following pages into a logical layout for a Web site using the paper or index card methods described in the preceding paragraphs:

- Contact Information
- About Us
- Our History
- What We Do
- Our Goals and Values
- Publications
- Upcoming Events
- Home
- Links
- Events
- Donate

Our Mission Statement

Staff Directory

To begin the page-ordering process, follow these steps:

- 1. Select which pages will be the main navigation pages.**
- 2. Pick the pages that will become subpages.**

Did you find any pages that didn't quite fit anywhere? If so, what would you recommend that the client do with the page(s)?

When you're finished with the organization, compare your solutions with the following potential answers for each step:

- ✓ For the main pages, include Home, About Us, Events, Donate, Publications, and Contact Information.
- ✓ For the subpages, under the About Us page, include What We Do, Our Mission Statement, Our History, and Our Goals and Values. Place Staff Directory as a subpage under the Contact Information page, and place Upcoming Events as a subpage under the main Events page.
- ✓ The Links page doesn't quite fit in anywhere and may not be necessary to site visitors. Recommend that the client either remove it from the site, add it as a subpage of the About Us section, or add it as a new main page called Resources, which sounds better than Links.

Having something like this written out on paper is a good first step in preparation for creating a design mockup for the site. However, if you really want to impress your client, consider building a site map.

Building a Site Map

A *site map* is a visual illustration or representation of a Web site's architecture or structural design. By reducing the site to its most important components, such as pages and page names, you can pay special attention to the placement and ordering of the pages relative to each other. In addition, you can add other relevant information to the site map, such as notes regarding the dynamic functionality of the site and which fonts and colors will be used in the site's design.

The site map serves a totally different function than the wireframes that you or your client may have created for the site to outline and organize the content for the pages. Instead, the site map becomes your guide to designing the mock-up for the site's layout and look and feel, as well as your road map for building the Web pages as you construct the entire Web site. Therefore, the site map should be created by you, the designer.

Site maps can be sketched by hand or produced in any software program that allows you to add text and draw rectangles and lines. Acceptable-looking site maps can be created by the nondesigner in Microsoft Word, Excel, PowerPoint, and Visio. Those with more design experience can create a site map in Illustrator or Photoshop by using the various line, shape, and text tools. Your site map can be as simple or elaborate as you want, as long as its main function to diagram the site is accurate and complete.

When designing client site maps, my preference is to use the drawing tools in Adobe Illustrator. This is because Illustrator allows me to create more elegant site maps by using unique shapes, like rounded rectangles, ovals, and polygons, as well as to display the specific fonts and colors that will actually be used in the site's design, giving me the opportunity to customize each site map to the client's project. Figure 3-11 shows an example of a site map I created for Evergreen Printing & Graphics, an environmentally responsible graphic design, offset and digital printing, and mailing and fulfillment company in New Jersey, which hired me to create its Web site in June 2007.

When you are ready with a list of information about your particular site, use the following steps to guide you in creating your site map:

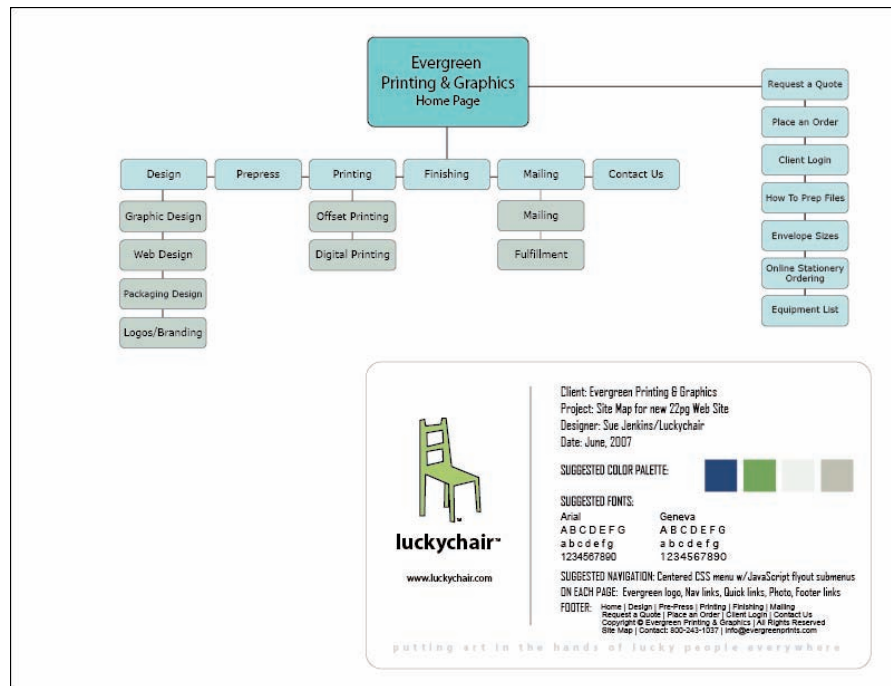


Figure 3-11: This visual site map was created in Adobe Illustrator.

- 1. Using a standard 8.5-x-11-inch letter-size page with a landscape layout, create and center a 2-inch-wide rectangle at the top of the page and label it “Home page.”**

A good site map shows how each of the pages on a site is connected from the home page as well as through the site’s navigation and subnavigation menus.

- 2. Directly below the Home page rectangle, draw a row of equally sized rectangles to represent all the main pages on the site, and label each of them according to your notes about the site.**

For example, your main pages might include About Us, Events, Donate, Publications, and Contact, or in the case of the Evergreen Printing & Graphics Web site, the main pages are Design, Prepress, Printing, Finishing, Mailing, and Contact Us.



Make sure that these main pages are listed in the exact order you intend them to appear on the Web site — from left to right for the main navigation and from top to bottom for any pages accessed through subnavigation links. This order can assist you in the site’s design and building phases.

- 3. Connect all the “page” rectangles by drawing straight connector lines. One line should flow horizontally behind each of the main page rectangles, and another line should connect the Home page to the horizontal line.**

Connecting the “pages” in this manner can aid you in identifying to the client how a visitor will interact with the navigation and experience visiting the site.

- 4. Add the subpages and subpage connector lines.**

Add additional rectangles and text labels to represent each of the subpages that falls under each of the main pages on your site. Then draw a vertical line behind each column of subpages to connect them to the main page that they fall under, as well as connect them to the main horizontal line that links the main pages (refer to Figure 3-11).

The line behind the subpages shows that the subpages will be accessible through the main navigation buttons. If any sub-subpages exist in your site, use this same technique to add them to your site map.

- 5. Most sites have at least one or more special pages, such as a Terms & Conditions, Privacy Policy, or Site Map, that are not accessible through the main navigation but that need to be accessible through footer links and/or regular hypertext links. If this applies to your site, add additional page rectangles to your layout, perhaps off to the side of the site map page, and label them accordingly.**

Though you could if you wanted to, you don’t need lines to connect these odd pages to the rest of the site map; they are indicated merely to

show you and the client that these pages exist and are accessible through means other than the main navigation.

- 6. In a blank area of the site map page, type in any special notations you may have about site details that should be appearing on every page, such as the logo, company slogan, toll-free telephone number, footer links, site search features, and so on.**

If desired, you may also want to include design information about what colors, fonts, font sizes, layout attributes, and page dimensions to use.

- 7. On another part of the site map page, preferably on the lower-left or -right side, add the date the site map was created along with your company name, logo, and contact information.**

This information clearly identifies you, the designer, as the author of the site map for this project.

- 8. Save the file, and if you own a copy of Adobe Acrobat Professional, convert the page into an Adobe Acrobat PDF document, which you can use to present the site map to the client.**

PDFs are wonderful tools to use because they don't require the client to own or purchase any design software to view them. If anything, the client may need to download and install the latest version of the free Adobe Acrobat Reader software, which is readily available online at Adobe.com.

If you don't own Acrobat Professional, consider using a free online PDF file-conversion tool, such as the one found at www.pdfonline.com.



Each time you make contact with your clients — whether by e-mail, in person, or by phone — you have the opportunity to make another positive impression on them about you and the quality of your Web design services. Friendly banter and a professional work ethic definitely go a long way toward fostering a good rapport with your clients, but so does the unspoken look of all your Web-related paperwork, e-mails, PDFs, and other correspondence. Therefore, make the most out of every nonverbal communication opportunity you have by consistently adding your name, company logo, and contact information, including telephone numbers and e-mail address, to every document you present to your clients.

When you have the chance, the completed site map should be reviewed by the client to make sure that it meets his or her expectations and that it includes and diagrams all the pages of the site in a logical manner. If any changes need to be made to the architecture of the site, making those changes now can save you valuable time after you begin creating the site design. After the site map is reviewed and approved by the client, get approval in writing before you proceed to the next step in the design phase — creating the design.

Chapter 4: Choosing the Right Tools

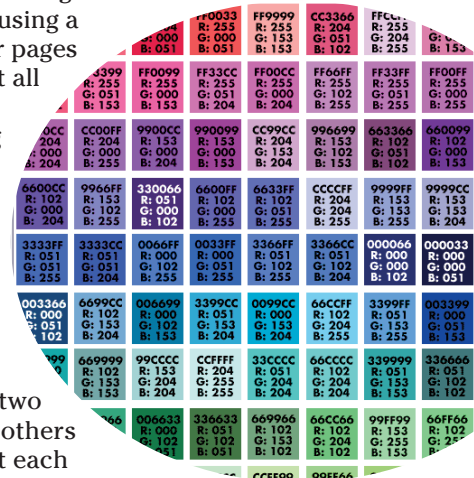
In This Chapter

- ✓ Using a Web editor (HTML versus WYSIWYG)
- ✓ Choosing the right graphics software
- ✓ Understanding HTML basics and code structure
- ✓ Using color effectively on the Web
- ✓ Choosing the right shopping cart for your e-commerce site
- ✓ Determining when to hire a programmer

By now you should have pulled together and organized all (or most of) the content required for the site, and you're nearly ready to begin working on the site design. Before you can do that, however, you must first make a few important decisions about which development tools to use. After that, you need to be sure that you understand some fundamental techniques for using those tools.

You begin this chapter by discovering the basics of coding and working with HTML. You can hand-code your pages using a simple text editor, or even better, you can build your pages with some kind of HTML or Web editor. Although not all Web-editing programs insist that you know HTML before you use them, having a simple understanding of HTML's structure and syntax can definitely help you build the pages for the site more quickly and efficiently. It's important, too, that you understand how to save your HTML files using the proper naming conventions and correct file extensions.

After you've selected your Web-editing tool, you need to look into and purchase (or download a free copy of an open-source application) at least one or two graphics programs. Some programs are better than others at certain tasks, so having an understanding of what each program can do can help you in selecting the right tools for the job. I highly recommend having at least one vector and one raster graphics program to create the Web site design and optimization of all the images



that will appear on the Web site. While you are creating your Web graphics, it is critical that you know how to work in the RGB color space, use Web-safe color palettes, and understand hexadecimal color values. This attention to color inside the program you choose to work with can help ensure that the colors in your design look just right.

In addition to finding a good Web editor and the right graphics software programs, when you are creating a site that has an e-commerce component or needs a way to process credit card payments, you also need to look into finding the right shopping cart solution. Your options range from a simple PayPal setup to third-party software or host-provided carts to custom-built shopping carts. All of these options — with the exception of PayPal, Google Checkout, or some other outside credit card processing service — require the site owner to get a merchant account from his bank and an SSL (Secure Sockets Layer) certificate from the host provider to securely process online payments.

Finally, although you may want to design and build the site entirely by yourself, some sites have such customized and complex data-processing needs that you'll want or need the assistance of a professional Web programmer. For example, if your client wants a custom-built shopping cart that caters specifically to her products, hiring a professional programmer is a very good idea. The last part of this chapter examines some criteria that can help you determine when it is time to call on a programmer and offers some suggestions on good places to look online to find one.

Working with Web Editors

HyperText Markup Language, or HTML, is the foundation code of any Web page. The code is comprised of a simple tag-based *markup language* for the World Wide Web that communicates information about how the hypertext links, text and other content formatting, and page structure of a document should be viewed in a Web browser. Almost anyone can learn HTML in a short period of time because its rules are reasonably straightforward, it isn't a full-scale programming language, and its structure is fairly uncomplicated.

The HTML code for any Web page can either be typed out using a plain-text editor, such as Notepad for the PC or TextEdit for the Mac, or written out using the specialized tools in a dedicated HTML (code) or Web (visual) editor. An editor is really the better choice for creating Web pages because an editor can help you create the HTML quickly as well as integrate other important technologies with the pages, such as adding CSS (Cascading Style Sheets), JavaScript, and other programming languages to the code.

Selecting a Web editor

There are two basic types of Web-editing programs — code editors and visual editors — that you can choose from to build your Web pages in HTML:

- ✓ **Code editors:** These kinds of HTML editors are perfect for people who already know some HTML and prefer to hand-code their HTML pages. Code editors can be as simple as the text-editing program that comes with your computer — whether that's Notepad on a PC or TextEdit on a Mac — or as complex as a program dedicated to writing HTML, such as BBEdit, shown in Figure 4-1, or HomeSite (which is a former Macromedia product and is now owned by Adobe), which has special tools, buttons, and other code helpers to assist with the tasks of coding your pages. If you're looking for a free coding editor, you might enjoy using the Bare Bones TextWrangler or the CoffeeCup free HTML editor. Any of these and other editors you might find online are fine to use, as long as the editor you choose assists you in writing HTML 4.01- or XHTML 1.0-compliant code.



Figure 4-1: For an HTML-only editor, use a robust tool like the Bare Bones BBEdit software.



Do not use a word processing application like WordPad or Microsoft Word as a code editor. These programs tend to add extra characters and unnecessary markup to the HTML code that can drastically increase the file size of the saved .html document.

- ✓ **Visual editors:** Also sometimes called *design, drag and drop*, and *Web* editors, these types of HTML editors, like Adobe Dreamweaver shown in Figure 4-2, allow designers to build pages by using a friendly WYSIWYG (pronounced “wizzy-wig”), or *What You See Is What You Get*, user interface. Visual editors provide you with easy-to-use buttons, tools, special shortcuts, and drag-and-drop features that allow you to add text, graphics, and other content to the Web page. This means that you don’t need to know much HTML to use them (though the more you know, the faster you can code and edit, and the better designer you’ll ultimately be). Besides their user-friendly interfaces, visual editors have working offline design views that closely mirror what the page will look like when displayed in a browser window (either with or without a live Internet connection), making the Web page–building process much more fun and easy to do.

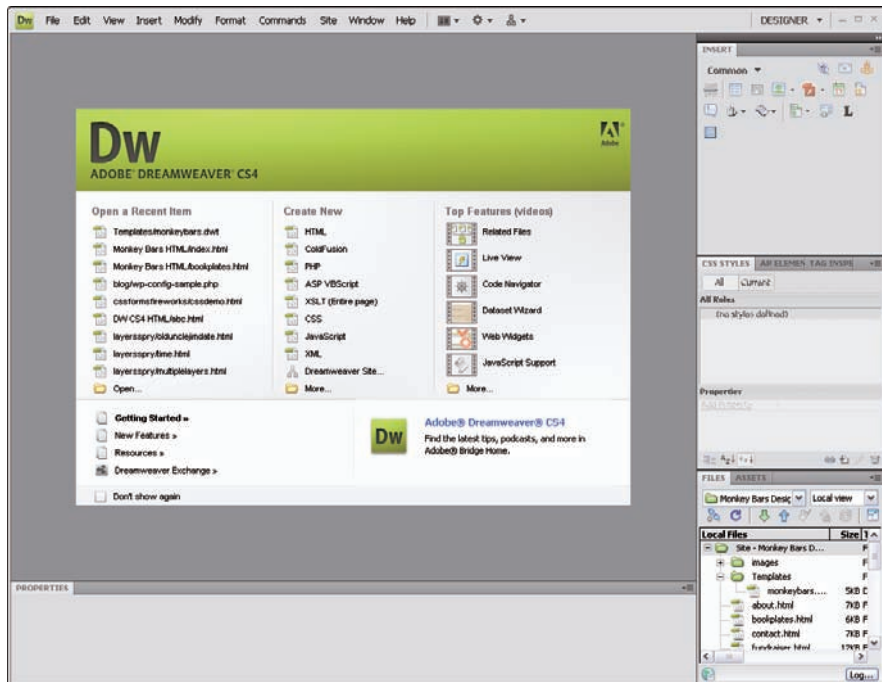


Figure 4-2: If you’re looking for the best WYSIWYG editor, get Adobe Dreamweaver.

Another great benefit to using a visual editor over a code editor is that most visual editors have built-in coding editors, so users can easily switch back and forth between the code and visual editing modes, or in the case of Dreamweaver, work in both modes simultaneously using a Split Code display that shows both the code and the design views of a page.

The most popular visual editors being used by professional designers today are Adobe Dreamweaver, CoffeeCup, Microsoft Expression Web, and Nvu (pronounced “n-view”). Of course, other Web editors are out there for you to choose from if none of the options presented here suit your fancy. Freeware seekers might like FormBreeze’s PageBreeze application or OpenOffice’s BlueFish editor.



If you have heard about an editor called GoLive, you can now forget about using it because that software was discontinued by Adobe in April 2008 so that Adobe could focus on the development and sales of Dreamweaver. To that end, Adobe is encouraging existing GoLive users to migrate to Dreamweaver by offering a special \$199 upgrade offer to GoLive CS, CS2, and 9 users.

You may also have heard about the FrontPage editor from Microsoft. This software was discontinued by Microsoft and replaced with Expression Web as of December 2006. Unlike FrontPage, which added a lot of unnecessary code to your pages, Expression Web is an appropriate tool to build standards-compliant Web sites.

This book uses Dreamweaver and hand-coding methods exclusively as the basis for providing instructional steps in later chapters on how to build your Web pages. You also find instructions for using Photoshop and a few other software tools. No matter which software programs you ultimately choose to use, all the instructions are easily adaptable so that you can follow along with the examples provided.

Understanding HTML and CSS structure

Even if you are somewhat uncomfortable right now with the idea of working with HTML code, as a Web designer you can benefit from having a basic knowledge and understanding of HTML code. Then when you get the hang of the basics, understanding the rest of the syntax rules should be a snap.

For starters, you should understand the difference between HTML and XHTML. HTML 4.01 code is the current standard coding markup language used for Web pages. Coding rules are somewhat forgiving with HTML should you make some organizational mistakes or code omissions, and the syntax is fairly straightforward to comprehend. By contrast, XHTML, which stands

for eXtensible HyperText Markup Language, is a stricter version of HTML that allows the data on a Web page to be used as an application of XML. Mistakes made in XHTML coding can throw off the display of content on a page in a browser and otherwise make the page nonfunctioning. Therefore, you should begin your journey into the world of Web coding with HTML and advance into XHTML when you have a clear understanding of coding fundamentals.

HTML code uses small components called *tags* to mark up your text and graphic content for presentation in a Web browser. Tags are essentially predefined words or acronyms written in all lowercase letters and surrounded by left (<) and right (>) angle brackets, as in the tag <html>. Understanding a few basics about HTML tags can help you understand much of the HTML code that you'll see as you begin creating Web pages.

Most HTML tags come in tag *sets*, or *pairs*, to mark the start and end of a text block or other object, such as an image, table, or layer, that appears on a Web page. Think of each tag pair as a kind of container that can hold certain kinds of content as well as inform a browser how to format and display that content. Closing tags look identical to the opening tags with the exception of having an added forward slash directly before the tag name after the opening bracket. For example, a sentence might be marked up with opening and closing paragraph, or <p>, tags like this:

```
<p>HTML tags make your documents viewable on the Internet.</p>
```

In HTML, the opening/closing tag rule has a few exceptions. These exceptions include the use of meta tags, line breaks
, image tags , and a few assorted others, like the tags used to add form fields and embed media files on a Web page. Because these exceptions can be somewhat confusing to people who are new to HTML, unclosed HTML tags can be closed out either by closing all unclosed tags with a closed tag, thereby creating a tag pair, like this:

```
<br></br>
```

or by borrowing from one of the syntax rules used in XHTML, namely, adding a space and a forward slash before the ending right angle bracket of the opening tag in question, as in this example:

```
<br />
```

When formatting needs to be applied to the contents of a tag or tag pair, the opening tag can also contain elements or attributes that tell the browser how the tag or the contents between the opening and closing tags appears. Formatting attributes can include such things as color, alignment, width and

height, and style. Attributes appear only in opening tags (never in the closing tags) using the syntax $x="y"$, where x is the attribute and y is the value of that attribute, as in the following example:

```
<hr width="500">  
<p align="center">This sentence will be centered on the page.</p>
```

If desired, you can list multiple attributes in the opening tag should the need arise:

```
<div id="main" align="center">
```

Many HTML tags deal with semantically identifying the different parts of the Web page content. You can use the basic tags like `<p>` for paragraphs and `<h1>` through `<h6>` for headings; character formatting tags, like `` for bold and `` for italics to add emphasis; and then all the rest of the tags — including tags for inserting links, images, lists, tables, frames, and forms — for adding graphics, objects, programming, styles, and metadata to the page.



For a complete listing and description of all the HTML tags you can add to your Web pages, visit the following sites:

```
www.w3schools.com/tags/ref_byfunc.asp  
www.webmonkey.com/webmonkey/reference/html_cheatsheet  
www.html.net  
http://reference.sitepoint.com/html
```

Looking at Web page structure

Every Web page uses a similar fundamental structure. All tags are hierarchically nested between opening and closing `<html>` tags, and all the code in the page falls between either the `<head>` or the `<body>` tags:

```
<html>  
<head>  
<title>The page title goes here and displays in the browser's  
  title bar.</title>  
</head>  
<body>  
This part of the Web page contains all the content that will  
appear in the browser window, including text and graphics  
marked up by HTML tags with attributes.  
</body>  
</html>
```

Figure 4-3 shows how this HTML code appears in a browser. The head area of a page contains certain information about the Web page that is interpreted by the browser, such as the page title (which appears in the

browser's title bar), meta tags used for page indexing by a search engine, style definitions in the form of CSS (Cascading Style Sheets) to control how content appears on the page, and JavaScript to manage simple site interactivity, such as rollover buttons and form processing. All the head data, with the exception of the title definition between the `<title>` tags, is hidden from view in the browser window and does not appear anywhere on the finished Web page. Rather, this information is used strictly by search engine spiders, robots, and crawlers.

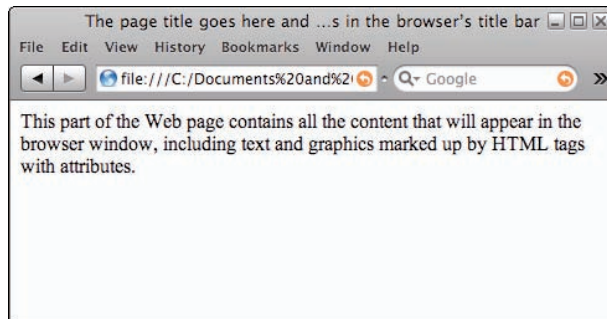


Figure 4-3: The content that appears on a Web page is determined by the markup positioned between the opening and closing `<body>` tags.

By contrast, the body area of the code is where all the page's text, graphics, and other objects go. All content included between the opening and closing `<body>` tags — unless marked up with special comment tags — appears as content on the page in the browser window. Therefore everything in the body of the page should be marked up with additional HTML tags to specify layout, style, and position.

Building a Web page

Building a simple Web page is easy. To prove it, you're going to create your first Web page by hand-coding it in HTML.

Follow these steps to see how easy HTML is to understand and use:

- 1. Open a new document in your computer's text-editing program.**

On a PC, choose Start → All Programs → Accessories → Notepad.

On a Mac, launch your Applications folder and double-click the TextEdit icon.

A new, untitled document should open automatically. If that doesn't happen, choose File⇒New to open a new file.

2. Type the following HTML tags into the document window:

```
<html>
<head>
<title></title>
</head>
<body>
</body>
</html>
```

3. Between the opening and closing <title> tags, type My First Web Page.

Your code should now look like this:

```
<html>
<head>
<title>My First Web Page</title>
</head>
<body>
</body>
</html>
```

4. Between the opening and closing <body> tags, type Hello world.

Now your code should look like this:

```
<html>
<head>
<title>My First Web Page</title>
</head>
<body>Hello world.
</body>
</html>
```

5. Choose File⇒Save to open the Save As dialog box.

6. Set the Save In location to your computer's desktop.

7. In the File Name field, type index.html.

8. In the Save as Type field, select All Files.

9. Click the Save button to save the file with the settings you just entered and close the file.

10. Launch your favorite browser and open the window to roughly half the size of your desktop.

11. Drag and drop the icon of your new index.html page into the open browser window.

Your new Web page appears with the words “Hello world.” in the body of the page, and the title, My First Web Page, appears in the browser’s title bar. Figure 4-4 shows a side-by-side comparison of your HTML code file and the browser window that displays your page.

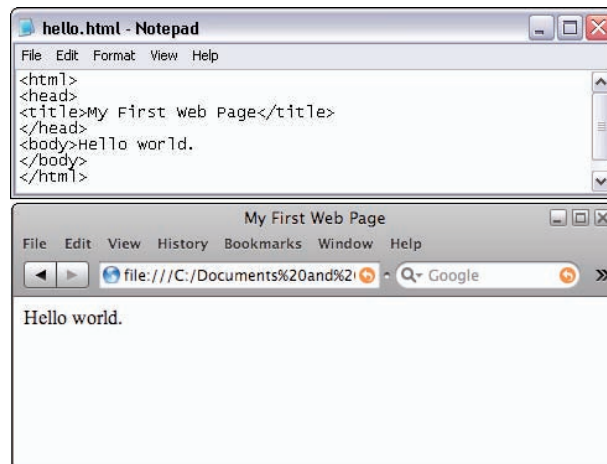


Figure 4-4: A side-by-side comparison of your code and Web page can help you visualize how your code works.

Congratulations! You’ve just created your first Web page. Of course, you already know that you can do a lot more to a page than include text, which you find out how to do in future chapters. For now, feel free to experiment by adding more text between the body tags or changing the title.

Saving Web files

I need to add a quick word about saving Web files that you may not know about. When saving your files, you can save them with any name you like. Filenames can be any length, but it’s more common to keep page names fairly short and succinct, and to avoid using spaces or odd characters with the exception of the underscore (`_`) and hyphen (`-`). For example, a page filename may be something like `about` or `email-signup`.

In addition to choosing a good descriptive filename, you should save all your Web page files with the appropriate file extension. It’s important to remember to do this because the extension informs the browser about the type of code used in the file so that the browser knows how to process and display the file’s contents. Acceptable extensions for HTML are `.html` and `.htm`. Either extension works; however, I strongly recommend that you

select a single extension (I use `.html`) and use it exclusively for all the pages on a single site.

Though most of your pages can be named anything you like, your home page must be deliberately named so that it appears as the default page when visitors go to the site. In most cases, you should save the home page as either `index.html` or `index.htm`. Because the filename `index` is the default page name in a Web directory on most Web servers, visitors do not need to type in the filename and extension when they want to access the first (home) page on a site. For example, if I want to see the home page of the Museum of Modern Art, rather than typing the URL and the filename plus the `.html` extension into my browser's address bar — **`www.moma.org/index.html`** — I only need to type **`www.moma.org`** (or simply **`moma.org`** if I felt particularly lazy) to get there.



Though it's almost always the case that your home page should be named `index.html`, some Web sites might be hosted on different types of Web servers that require the home page to be named either `default.html` or `default.htm` instead. In my experience, this happens so infrequently, though, that I can usually figure this out by testing the `index.html` naming convention with a test page on the live server. If `index.html` doesn't work, `default.html` should. To save time, however, if you're unsure which filename to use, check with your (or your client's) host provider or system administrator.

When your pages use programming languages or Server-Side Includes (a page containing HTML code that's embedded inside another HTML file), other file extensions need to be used instead of `.html`. For example, `articles.shtml`, `contact_us.asp`, `products.jsp`, and `services.cfm` are all acceptable filename+file extension combos. Table 4-1 outlines the file extensions you can use for different types of pages.

Table 4-1 Saving with the Correct File Extension

<i>Type of Page</i>	<i>Extensions</i>
HTML/XHTML	<code>.html</code> , <code>.htm</code>
HTML/XHTML pages with Server-Side Includes	<code>.shtml</code> , <code>.shtm</code>
Microsoft Active Server page	<code>.asp</code> , <code>.aspx</code>
JavaServer pages	<code>.jsp</code>
PHP script pages	<code>.php</code>
ColdFusion Markup Language pages	<code>.cfml</code> , <code>.cfm</code>

Choosing the Right Graphics Software

One of the nice things about being a designer, in my opinion, is the fact that you get to make lots of important decisions throughout the Web design process, including what software applications to use to create the design mock-up, optimize all the graphics, and build all the pages on the site. Whether they're amateurs or looking to become professionals, all Web designers need a few of the right software tools to successfully design and build Web sites. In the following sections, you find out about how to choose the right graphics software applications.

Graphics programs

To work with and create images for the Web, designers can freely use both raster and vector programs:

- ✓ **Raster:** A raster or bitmap program, like Adobe Photoshop, shown in Figure 4-5, uses pixels to represent each bit of an image at the size used to create the image. Raster/bitmap programs are great for photographic retouching and for building Web site mock-ups. While it is okay to scale down (make smaller) a raster image, you should definitely try to avoid enlarging a raster file because the larger version needs to use a computerized guessing technique called *resampling* and tends to look, well, pixelated. Alternately, Web designers on a tight budget might enjoy working with Adobe Fireworks, a dedicated Web graphic image editor.
- ✓ **Vector:** By contrast, a good vector program, like Adobe Illustrator, shown in Figure 4-6, uses mathematical algorithms to draw shapes and paths. This means that vector images can be scaled up and down a zillion times and still retain the sharpness and clarity of their lines and shape at any size. Vector programs should always be used to create branding and corporate identity such as logos and other artwork because the work can be scaled with no loss of resolution. In addition, vector graphics often have smaller file sizes than their raster-based bitmap counterparts.

By far, the most popular raster graphics software application in use today is Adobe Photoshop (CS4, CS3, and Elements 6 and 7), which, while primarily being a raster (or bitmap) application, can also create vector shapes and incorporate placed vector graphics or SmartObjects from vector programs like Adobe Illustrator. Originally developed as a digital photo retouching and image-editing program, Photoshop has since evolved its capabilities for use as a nice Web graphics design and optimization tool. To get the most from Photoshop, use the full version (such as CS3 or CS4). Alternatively, if you're on a budget or you already own a copy of Photoshop Elements, it

should suffice. For vector art, most designers use Adobe Illustrator because it's a feature-rich drawing program that allows you to draw just about anything for use in print, Web, animation, mobile devices, and video.

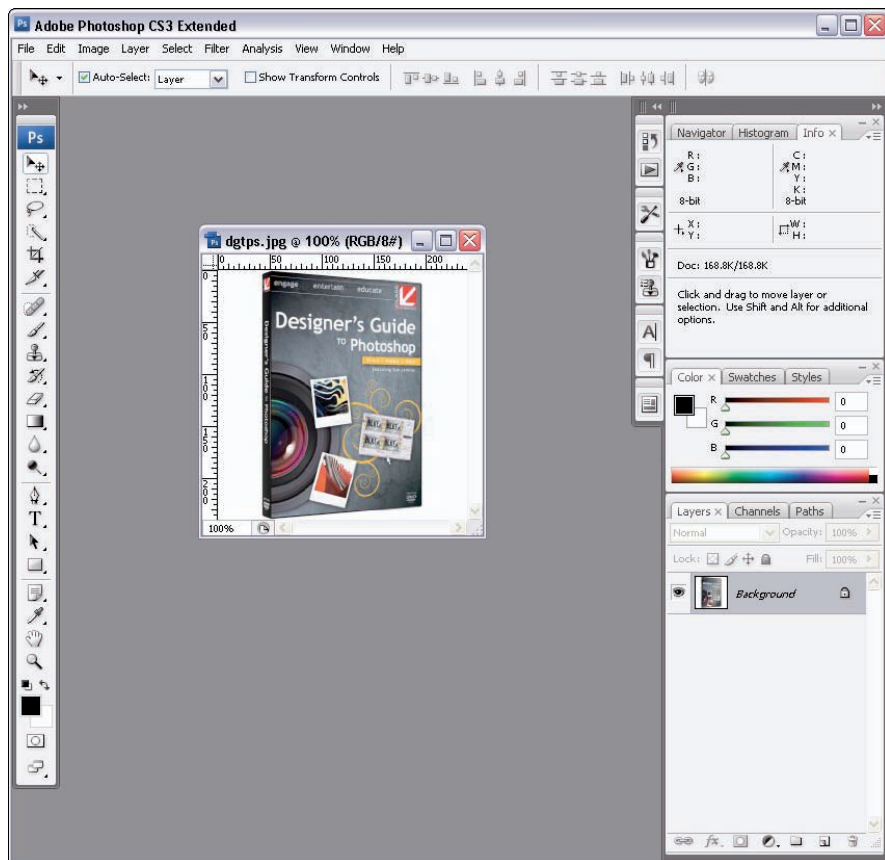


Figure 4-5: A good program like Adobe Photoshop is perfect for creating raster graphics and editing digital photographs.



Though some graphic designers use Photoshop and Illustrator as page layout programs for print projects like postcards, brochures, and business cards, most tend to use software programs designed specifically for page layout, such as Adobe InDesign or QuarkXPress. *Do not* use a page layout program for your Web layouts! Page layout programs were designed for

print, not the Web, and therefore don't necessarily have all the same features you'd currently find in Photoshop, Illustrator, and Fireworks. To be fair, Quark 7 now allows designers to export graphics into HTML as well as perform a few other optimization-like features similar to the things you can do with graphics in ImageReady and Fireworks. InDesign, however, currently offers nothing in the way of exporting HTML or graphics optimization. What it can do is allow designers (since the CS version) to export only an InDesign-tagged XML file. That file can then be imported into a pre-designed Adobe Dreamweaver template, which inserts tagged data into preset areas on a Web template and doesn't allow designers to take advantage of their own design layout.

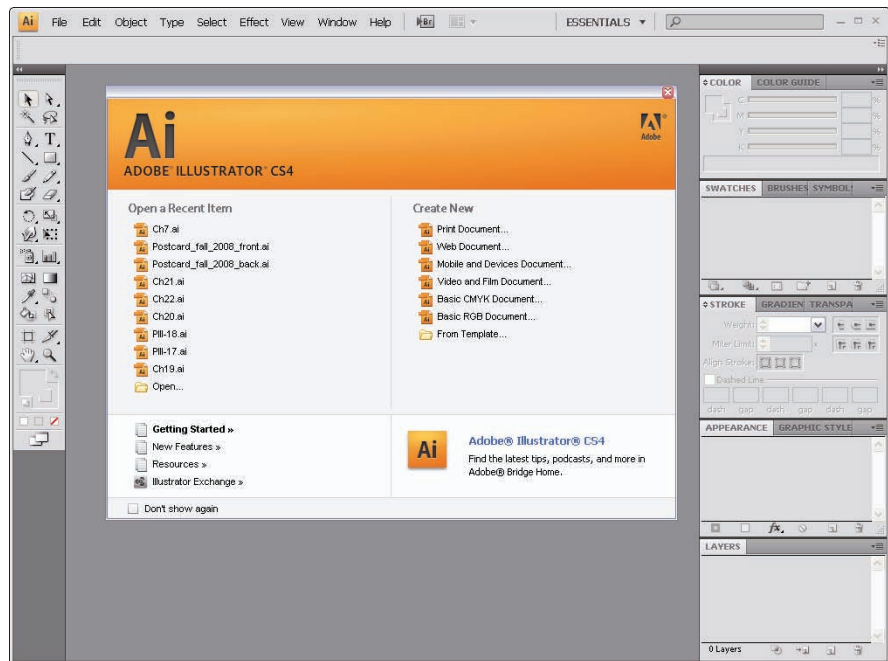


Figure 4-6: Use a program like Adobe Illustrator to create your vector illustrations and graphics.

At a minimum, I suggest that you have access to at least one good raster and one good vector art program, preferably Photoshop and Illustrator, respectively. Use the raster program to lay out all the parts of the Web page in the mock-up (which you find out more about in Book II, Chapter 2), and if the design requires any special logos, shapes, drawings, or illustrations, you can create them in the vector art program and then copy and paste them as SmartObjects (graphics that retain their vector scalability while

sitting as an object inside a raster program) into your raster mock-up. You could also use the vector application exclusively for the layout if desired, but many designers find the filters and effects in a vector program somewhat limiting compared to the special effects that can be applied to your work in raster programs.

You can, of course, use other vector and raster image-editing programs to create your Web graphics:

- ✓ Adobe Fireworks is an excellent all-in-one Web image-editing and image-optimization tool that integrates beautifully with Adobe Dreamweaver.
- ✓ Paint.NET is a free, open-source, easy-to-use, powerful, bitmap-image and photo-editing software program for PCs running Windows, available for download from www.getpaint.net.
- ✓ DrawPlus from FreeSerifSoftware.com is a somewhat new tool for vector drawing, editing, and text manipulation.
- ✓ For a super low-end, cookie-cutter graphics option, check out NetStudio's Easy Web Graphics program or Xara's Webstyle software.
- ✓ If you have no budget for purchasing software, you can use several free online applications, such as the image editor at www.myimager.com. You can also find some good freeware (like the Gnu Image Manipulation Program, or GIMP), shareware, and 30-day trials of image-editing software programs (like CorelDraw) through download Web services like www.download.com and www.zdnet.com.

That said, I highly recommend that you make the investment and buy Photoshop and Illustrator if you don't already own them, or update them to the newest version if you own old copies. These programs really are the best tools available right now, and the quality of your Web site may be relative to the quality of the graphics you can create for it.



If you don't own either of these programs yet or if you have old versions of them and are in need of an upgrade, a smart choice would be to buy the Adobe CS4 Production Premium, Master Collection, Design Premium, or Web Premium bundle directly from Adobe. Alternately, students who can prove they are enrolled in a K-12 school, privately licensed training center, community college, or four-year college or university can qualify to purchase this software at a discounted rate. For example, the full version of the Adobe CS4 Web Premium bundle costs around \$1,699, whereas the educational version of the same bundle costs only \$549. Table 4-2 shows a cost comparison of the bundled Adobe product, stand-alone versions, and some truly useful, free, open-source products.

Table 4-2 Side-By-Side Software Cost Comparison

<i>Product Name</i>	<i>Product Type</i>	<i>Cost</i>	<i>Company</i>
Adobe CS4 Web Premium comes with CS4 versions of: Dreamweaver, Flash, Photoshop, Illustrator, Fireworks, Acrobat, Soundbooth, Contribute, Bridge, Device Central, and Version Cue.	Full suite of products to help design mock-ups create optimized graphics, create vector-based animations, convert files to PDFs, and build Web sites.	\$1,699 or upgrade from \$599	adobe.com
Dreamweaver CS4	WYSIWYG HTML Web Editor	\$399 or upgrade from \$199	adobe.com
Photoshop CS4	Raster/Bitmap Graphics Editor	\$699 or upgrade from \$199	adobe.com
Photoshop Elements 7	Raster/Bitmap Graphics Editor	\$139 or upgrade from \$119	adobe.com
Illustrator CS4	Vector Graphics Editor	\$599 or upgrade from \$199	adobe.com
Fireworks CS4	Web Graphics Editor	\$299 or upgrade from \$149	adobe.com
CorelDraw	Graphic Design and Illustration Vector/Bitmap Software	\$389 or upgrade from \$199	corel.com
Microsoft Expression Web 2	WYSIWYG HTML Web Editor	\$299 or upgrade from \$99	microsoft.com/expression/
BBEdit	HTML Web Editor	\$125 or upgrade from \$30	barebones.com
TextWrangler	HTML Web Editor	Free	barebones.com
CoffeeCup	WYSIWYG HTML Web Editor	Free	coffeecup.com/free-editor/
PageBreeze	Open-Source Web Authoring System	Free	pagebreeze.com
KompoZer	Open-Source Web Authoring System	Free	kompozer.net

<i>Product Name</i>	<i>Product Type</i>	<i>Cost</i>	<i>Company</i>
NVU	Open-Source Web Authoring System	Free	http://net2.com/nvu/
BlueFish	Open-Source Web Authoring System	Free	http://bluefish.openoffice.nl
Paint.NET	Open-Source Digital Photo Retouching and Image Editor	Free	getpaint.net
InkScape	Open-Source Vector Graphics Editor	Free	inkscape.org
DrawPlus	Open Source Graphics Editor	Free	freeserifsoftware.com
PhotoPlus	Open Source Digital Photo Retouching and Image Editor	Free	freeserifsoftware.com

Web graphic optimization programs

After you've finished creating your mock-up for the site design, you need to *optimize* your graphics (which is covered in detail in Book II, Chapter 3). Optimization means compressing the graphics into acceptable file formats, such as .gif, .jpg, and .png, that are small enough to be displayed on the Web without losing too much image quality.

In the past, the application most frequently used for image optimization was Adobe ImageReady, which used to come bundled with Photoshop. Today the ImageReady optimization engine has been integrated into both Photoshop and Illustrator (using the File⇨Save for Web and Devices command), eliminating the need for a separate program for the optimization process. If you happen to have an old copy of ImageReady, you can certainly still use it, but if you've already migrated to CS3 or CS4 in Photoshop or Illustrator, you don't need to.

Another option is to design and optimize or just optimize your Web graphics with Adobe Fireworks or any number of the free or affordable optimization programs for a PC or Mac, like DeBabelizer Pro and Ulead's SmartSaver. What makes the Save for Web and Devices dialog box and Fireworks applications so good, though, is that they can both optimize graphics and export tables- and layers-based HTML files with CSS. This can greatly reduce the time it takes to build a Web site using an HTML or Web-editing program like Adobe Dreamweaver. For non-Adobe alternatives, including some truly fantastic, free, open-source image-editing software programs, refer to Table 4-2.

Working with Color

In the following sections, you find out how to work with RGB color on the Web using the Web-safe palette and hexadecimal values.

Using Web-safe colors

Of necessity, the Web-safe color palette was born in the early days of the Internet (coined by Lynda Weinman of Lynda.com) because computer monitors at the time were only capable of an 8-bit display, which meant they could only handle showing a maximum of 256 colors on-screen.

Unfortunately, the 256 colors that were viewable in an Internet Explorer, Netscape, or Mosaic browser on a PC running Windows were somewhat different from the 256 colors viewable in an Internet Explorer, Netscape, or Mosaic browser on an 8-bit monitor connected to an Apple computer running the Mac OS. As it turned out, a total of 40 nonoverlapping colors out of the total possible 256 colors were visible on both platforms. This left a total of 216 colors, shown in Figure 4-7, that would render uniformly on an 8-bit monitor in those early Internet browsers on both Mac and PC platforms.

Thus, to help ensure that visitors on PCs and Macs would have uniform visual experiences when they visited a particular Web site, Web designers used to limit their Web design colors to this Web-safe or browser-safe color palette. As you can imagine, this Web-safe color palette was somewhat restrictive in scope. And to make matters worse, the palette didn't necessarily have a wide range of appealing colors in it, having been developed mathematically by programmers, not aesthetically by artists.

But, one must work with the tools and technology available at any given time and make the best of it. And thankfully, these color limitations were somewhat short-lived because of improvements made to monitor resolutions and browser capabilities. Today, monitors capable of 16-, 24-, and 36-bit displays can render millions of colors on-screen, which means that the Web-safe palette is now pretty much a thing of history.

Nonetheless, for some Web projects, it is still suggested that designers create designs for (or at least consider) viewers of the lowest common denominator — the audience members still using old computers, old monitors, and old versions of old and perhaps now-extinct browsers — and thus choose colors from the Web-safe palette for their Web designs. At the time of this writing, Internet users with monitors having an 8-bit display or using handheld computers with 8-bit displays represent less than 2 percent of the installed base. In most cases these days, the Web-safe palette simply isn't an issue, especially with products that use more standardized computer equipment, such as when making a site for a corporate intranet.

Using hexadecimal colors

When you're creating graphics for the Web in your preferred vector or raster software program, you need to do two things to ensure that your color looks good on the Web. The first thing you should do is check to see that you're working in the RGB (red-green-blue) color mode (not CMYK!). The second thing to do is ensure that you specify all the color in your mock-ups and other graphic files using hexadecimal values.

As you'll soon discover, when you're working in an HTML code or Web page editor, all colors — Web-safe or otherwise — must be specified in hexadecimal values preceded by the number symbol (#) to display properly on a Web page. The hexadecimal numbers, often called simply “hex colors,” “hex numbers,” or “hex values,” refer to the six-letter/number combination RGB (red-green-blue, the additive colors used to display color on a computer monitor) values written in three pairs of numbers (from 0 to 9) or letters (from *a* to *f*), as shown in Table 4-3.

Table 4-3		Sample Hexadecimal Values		
<i>Color</i>	<i>Hexadecimal Value</i>	<i>Red</i>	<i>Green</i>	<i>Blue</i>
Black	#000000	0	0	0
White	#ffffff	255	255	255
Gray	#c0c0c0	192	192	192
Red	#ff0000	255	0	0
Green	#00ff00	0	255	0
Yellow	#ffff00	255	255	0
Blue	#0000ff	0	0	255



If you like gadgets, you might enjoy playing around with the online Color Mixer at MathIsFun.com, shown in Figure 4-8. As you move each of the sliders, you can see the RGB and hex equivalents.

Of the millions of monitor colors that have hex values, only a handful of them have named equivalents that are supported by most Web browsers and can be used in HTML and CSS code instead of the hex values, such as LightCoral for #f08080 and DarkSeaGreen for #8fbc8f.

To find the hexadecimal value of any RGB color or the hexadecimal equivalent of a CMYK (cyan-magenta-yellow-black, the subtractive ink colors used in four-color process printing) color, you must use a software program that has a value conversion tool installed. Photoshop, Illustrator, Fireworks, and Dreamweaver all include Web-safe palettes and color tools that display and

can convert RGB, CMYK, and Pantone (a color-matching system used by designers and printers; see www.pantone.com) colors into hexadecimal values.

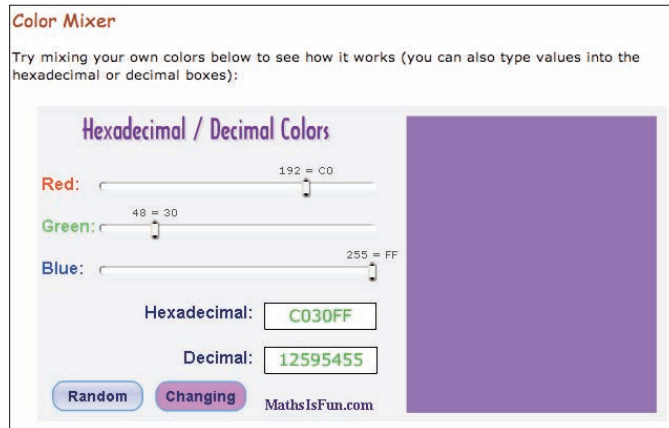


Figure 4-8: Use the hexadecimal color value when specifying a color on the Web.

If you want to use a stand-alone color-conversion tool, you can find several freeware applications by searching online for the term *free color picker*. ColorPic, Color Cop, Color Spy, Huey, and ColorPickerPro all provide excellent RGB, CMYK, and hex conversions. You can also find a fine online color-conversion tool at <http://web.forret.com/tools/color.asp>.

When building Web sites for others, you may often encounter clients who provide you with Pantone colors instead of CMYK or RGB values. In those cases you will need to find the RGB equivalents to use within your Web designs. If you are using an Adobe product, the Pantone matching system is built-in to most programs like Photoshop and Illustrator. Non-Adobe users, however, need the special Pantone.com program myPANTONE palettes. Although this software is free, users also need to purchase and download palette libraries from Pantone, which typically cost between \$10 and \$20 apiece.

To find the hexadecimal equivalent of a Pantone color provided by your client using Photoshop, follow these steps:

- 1. Launch the program and choose File⇨New to create a new document.**

The New dialog box opens. Here you can enter the appropriate settings to set up your document.

You don't necessarily need a new document to use the color-picker tool, but in this exercise, you will so that you can save the swatches for future reference.

- 2. In the New Document dialog box, enter the following settings for the new document:**

Enter **760 x 420** pixels for the Web page width and height dimensions and **72** pixels/inch for the Resolution. For the Color Mode setting, choose RGB, **8** bit, and for the Background contents, set that to **White**.

These settings are typical for Web design, and you can use them for most of your Web page mock-ups.

- 3. Click the OK button to close the dialog box and return to the new blank document.**
- 4. With the rectangular marquee tool selected in Photoshop, click and drag inside the top-left edge of the document window to create a small rectangular shape (about 1 x 2 inches) and release the mouse button.**

The rectangular marquee tool creates a selected area that you can fill with any color.

- 5. Click the Foreground Color icon at the bottom of the Tool palette.**

The foreground/background color selector icons are the overlapping squares. The foreground is on the upper left, and the background is on the lower right.

This step opens the Color Picker dialog box, shown in Figure 4-9, which has several options for selecting and viewing color.

- 6. Select the Only Web Colors check box at the bottom of the dialog box.**

This shifts the look of the colors that display in the left side of the dialog box from showing millions of colors to showing only the 216 Web-safe colors. Because you're going to be looking at RGB, Pantone, and CMYK colors to find their hexadecimal equivalents, you don't need to be restricted to the Web-safe palette.

- 7. Deselect the Only Web Colors check box.**

You want to see millions of colors.

- 8. To look up an RGB color, enter the following values in the R, G, and B fields: 246, 191, and 0, respectively.**

As you enter the numbers into the R, G, and B fields, notice how the color shifts inside the Color Picker.

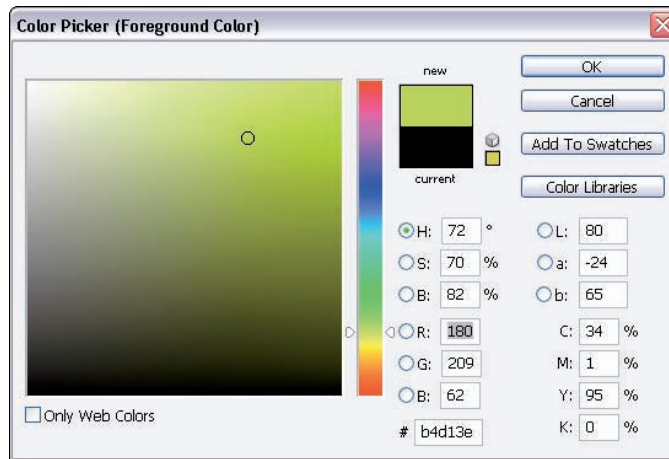


Figure 4-9: The Photoshop Color Picker displays colors in RGB, CMYK, Pantone, and hexadecimal values.

The hexadecimal value of your entered color appears as a combination of six letters and numbers in the hex field at the bottom of the Color Picker, beginning with a # symbol.

The hexadecimal value of this RGB color is #f6bf00.

9. Click the OK button to close the Color Picker dialog box.

The rectangular shape is still selected. Next you fill it with the new hex color you just specified the RGB values for.

10. Choose Edit→Fill to open the Fill dialog box and, from the Use menu, select Foreground Color. Then click the OK button.

The yellow color fills the selected rectangle.

11. To the right of the yellow rectangle in your document, repeat Steps 4 and 5 to create another rectangular shape and open the Color Picker dialog box.

You use the same process to look up the hex value of a CMYK color.

12. Look up the CMYK color by entering the following values in the C, M, Y, and K fields: 52, 15, 98, 1, respectively.

The hexadecimal value of this CMYK color is #89ac40.

13. Repeat Steps 9 and 10 to fill the second rectangle with the new hex color.

- 14. To the right of the green rectangle, repeat Steps 4 and 5 to create a third rectangular shape and open the Color Picker dialog box.**

You use the same process to look up the hex value of a Pantone color.

- 15. To look up a Pantone color, click the Color Libraries button.**

This changes the layout of the Color Picker dialog box.

- 16. Choose Book→Pantone Solid Coated, and then type a Pantone color number: 297.**

The color field automatically displays a range of blue color swatches with the Pantone 297C (the C stands for coated) color highlighted.

- 17. To convert the Pantone color to a regular process color and find its hex equivalent, click the Picker button.**

This changes the layout of the Color Picker dialog box back to the normal Color Picker mode. The same color blue is selected, but now instead of seeing a Pantone swatch, you can see that color's RGB, CMYK, and hexadecimal equivalents!

The hexadecimal value of this Pantone color is #78c7eb.

- 18. Repeat Steps 9 and 10 to fill the third rectangle with the new hex color.**

- 19. Choose File→Save to save the document.**

Select a location on your computer to save your new file, name your file `hexswatches.psd`, select the Photoshop format, and click the Save button.

Choosing a Shopping Cart

When the site you are building will be selling any products or services online, the site needs some kind of shopping cart or payment-processing system for payments coming from the purchasers. Payments can be processed in many ways, depending on the needs and budget of the site owner.

The most basic kind of shopping cart uses a payment-processing service such as PayPal. The next tier of service is to create an online store using a specialized shopping cart service's proprietary software, like Yahoo! Merchant Solutions. Some of these carts, however, must reside on the service's server away from the site owner's main URL, which can at times make purchasers feel uneasy, unless the carts are hosted by a reputable service like Yahoo!, which can elicit instant trust from purchasers. Other shopping cart services include building more customized carts from either Web-host-provided services or out-of-the-box shopping carts from third-party software

manufacturers. While fancier, these solutions do have limitations and can be very frustrating to customize, even for the most experienced designers.

For clients looking for something really slick, the best option is to have a shopping cart custom built so that it's tailored specifically to the site's needs. This, however, can cost significantly more money and take a lot more time to build than the other shopping carts.

In the following sections, I give you the lowdown on each of these shopping cart services.

Using PayPal shopping carts

One of the great things about PayPal is that it offers several payment solutions for individuals and businesses. The PayPal shopping cart is great for everything from sites working within a budget or selling only a handful of products to sites with existing shopping carts and sites where a customized cart will be integrated with the PayPal payment system. What's more, PayPal accounts are easy to set up and easy to use.

To use any of PayPal's services, follow these steps:

1. The client needs to set up a Merchant PayPal account linked to the bank she'd like to use to accept credit card and other online payments.
2. The client must also input data about all the products she'd like to sell on the site so that PayPal can generate "Add to Cart" or "Buy Now" PayPal buttons and HTML code for each product.
3. When that's done, as the designer, you need to copy and paste the code for each of those buttons from the PayPal site into the HTML pages that list the products on the client's site.

That's all there is to it.

To use the PayPal shopping cart online, visitors click the PayPal button of the item they want to purchase, as illustrated in Figure 4-10, and after clicking, a special PayPal payment-processing page opens. Registered PayPal users can simply enter their payment information to complete the transaction. Unregistered visitors can either choose to set up a PayPal account or simply enter their credit card information to proceed with the transaction without creating a PayPal account. After the transaction is complete, PayPal sends an automated e-mail to the site owner about the purchase. The site owner then has the responsibility of processing the order for the customer. PayPal processes Visa, MasterCard, American Express, Discover, eChecks, and PayPal payments.






















Age/ Day & Time	MON	TUES	WED	THURS
8 - 12 mos		TUES 12:30 		THUR 12:30 
12 - 24 mos	MON 10:15 	TUE 9:30 	WED 10:15 	THUR 9:30 
	MON 1:00 	TUE 2:30 	WED 1:00 	THUR 2:30 
24 - 36 mos	MON 11:15 	TUE 10:30 	WED 11:15 	THUR 10:30 
		TUE 3:30 		THUR 3:30 
3 - 4 yrs	MON 9:15 	TUE 11:30 	WED 9:15 	THUR 1:30 
	MON 3:30 			THUR 4:30 
4 - 6 yrs	MON 4:30 	TUES 4:30 	WED 4:00 	

Figure 4-10: PayPal offers free Add to Cart and Buy Now buttons for e-commerce sites that want to process payments off-site.

Checking out Google Checkout

If you already advertise with Google AdWords (or plan to), you might be interested in Google Checkout, a PayPal-like button-activated checkout payment-processing system that can process credit and debit cards free of charge. The system does require visitors to have their own Google account to complete the transaction. To find out more about Google Checkout, visit <https://checkout.google.com/sell>.

Looking into third-party and Web-hosted shopping carts

Two additional payment-processing alternatives for e-commerce Web sites include using a host-provided shopping cart system or purchasing a third-party software solution.

Most reputable Web-hosting companies offer some kind of shopping cart service in one or more of their hosting plans. Host e-commerce plans often use their own specialized, custom-built applications that allow site owners to add all the required products to their site and manage those products through some kind of Web interface.

These kinds of systems are typically fairly easy to set up, configure, and use; however, site owners should be aware of the greater fees associated with these types of e-commerce hosted plans. In addition to the monthly or annual hosting fee, site owners also need to get a merchant account to process credit cards through their shopping carts, as well as pay for an SSL certificate to ensure that online transactions through their Web sites are secure. These extra fees and capabilities can increase the total cost of a hosting plan considerably (say, from \$25 per month to upwards of \$200 per month). Because host-provided carts can vary drastically in functionality and visual appeal, be sure to test-drive any of the carts you're considering before committing to using the service.

For those looking for a more customized but still reasonably inexpensive e-commerce solution, a third-party software application could be the answer. However, finding a good software package might prove challenging. In the "2009 Shopping Cart Software Report" on TopTenReviews.com (www.shopping-cart-review.toptenreviews.com), ShopSite Pro, MerchandiZer Pro, and NetworkSolutions were rated the best shopping cart software programs on the market, but be sure that you take these recommendations with a grain of salt.



I've personally used one of the programs that TopTenReviews.com rated highly on its top-ten list, but I'd never recommend it, even to my worst enemy. For that reason alone, I strongly urge you to stay far away from packaged solutions and instead go either the PayPal or customized cart routes.

Believe it or not, some free and inexpensive shopping cart software applications are fantastic and ready for downloading. And though they require a bit of coding to configure the cart, a few of them are quite reputable. Check out Zencart.com, 1stshoppingcart.com, Litecommerce.com, Interspire.com, and Xcart.com. Definitely do a Web search to find out more about your shopping cart options if you decide to go the third-party route.

Building custom shopping carts

My favorite way to add a shopping cart to a site is to enlist the help of a programmer to build a custom cart that meets the client's exact specifications. With a custom-designed shopping cart, every aspect of the ordering and checkout process can be customized, from the color scheme and user interface to the checkout confirmation page. This kind of detailed customization is possible because a tailored database is typically developed and integrated into the site as part of the e-commerce solution while the site is initially being built. All the e-commerce page layouts can be customized to the products being sold and have the same navigation, look, and feel as the rest of the site. Not only that, but using the customized database means being able to customize the payment-processing tasks, including sending e-mail receipts to the purchasers and creating administrative reports for the client.

If you don't have a background in database development and integration, you could either figure out how to do these things yourself or hire a programmer to do them for you. If you have good split-brain power and are motivated to learn, go for it. Personally, I'd much rather have someone who enjoys this kind of left-brain data manipulation do the work for me than to spend my own right-brain power trying to figure out how to do it myself. You discover more about hiring a programmer later in this chapter.

Planning for secure transactions

Whichever solution you decide to use to process payments online, take extra time and care to ensure that the site visitors' personal information is safe and secure during the purchasing transaction. If credit card payments are going to be processed on the Web site (instead of through an outside service like PayPal or Google Checkout), the site owner needs to set up a business merchant account through his bank as well as purchase and install an SSL certificate from his host provider for the domain.

A *merchant account* is a special bank account that handles the processing of credit card transactions. In fact, oftentimes, a bank will require business owners with both online and "brick and mortar" stores to have two separate merchant accounts: one for online sales and another for in-store, phone, and fax orders. If you or your client only plan to sell online, you only need the one merchant account. Either way, the merchant account collects payments electronically from the purchaser and then transfers the funds into the business owner's local business checking account. If you or your client decides to set up a merchant account, keep the following facts in mind:

- ✓ **Setup and application fees:** Merchant accounts can be set up through local banks, through host providers, and through billing software Web sites such as the Intuit site for QuickBooks. Most accounts cost \$25–\$250 in setup and application fees.
- ✓ **Monthly processing fees:** Fees for merchant services can add up quickly! Most merchant accounts charge a minimum each month, in the \$15–\$30 range (this is sometimes called a gateway or statement fee) when the number of transactions processed is below a preset minimum, plus a nominal charge, \$0.20–\$0.30, on all transactions processed. If the monthly minimum is met in transaction fees, the monthly minimum fee is often waived. Thirty dollars might be merely a token savings if you do a lot of business online. For instance, at \$0.30 per transaction, if you sell \$10,000 in products a month, you'd pay \$300.00 in transaction fees.

SSL stands for Secure Sockets Layer and refers to the digital security Web certificate that needs to be purchased from the host provider by the site owner and configured for the domain by the host provider. A valid SSL certificate guarantees that the site uses 128-bit or higher encryption methods

to keep both visitors' personal information and credit card numbers secure and to protect the Web site from hackers and credit card thieves.

The leading brand of SSL certificate is VeriSign, but several others are available at varying costs. Annual rates vary from vendor to vendor depending on the bit encryption rate. Here are the current rates for the three most popular SSL brands:

- ✓ VeriSign, \$695–\$2695
- ✓ Thawte, \$399–\$1899
- ✓ GeoTrust, \$249–\$995

All certificate issuers charge a one-time or annual fee, and the host provider might charge about \$50 as a setup fee to obtain the certificate for you, though sometimes with specials, host providers will waive the setup fee. The first year always costs more than the renewal rates for subsequent years.

Knowing When to Hire a Programmer

In Book I, Chapter 1, you discover a little bit about diagnosing a site's dynamic needs. Adding dynamic features to a site adds cost and time to the project, and not all sites truly need it. Budget is often the primary factor for determining whether to add dynamic functionality to a site. Other considerations include expected growth of the site, the projected schedule for making updates, and the amount of data to be served on the site.

Taking a look at your dynamic content needs

You might need to hire a programmer if you

- ✓ Have information stored in a database (or intend to) and want to have that data dynamically served on the pages of the Web site.
- ✓ Want a site-search feature on the Web site that accesses the site's database and returns search results based on selected search criteria, such as an alphabetized listing of store locations.
- ✓ Want a custom-built shopping cart for the site's products or services that allows you to customize the user's experience throughout the entire purchasing process from product selection to payment confirmation and order tracking.
- ✓ Would like a Content Management System (CMS) built for the site so that you (or your client) can manage content on the site through an easy-to-use Web interface.

- ✓ Need to create an area of the site that requires a username and password for secure login to a database of accessible records or to other password-protected and members-only areas of the site.
- ✓ Want to allow visitors to choose how data will be displayed and sorted on a page by clicking the category heading of a table of data to re-sort the records, as with the Sort by Price shopping feature found on most e-commerce sites.
- ✓ Want to collect information from site visitors who have completed an online form, add that data to a database, and use that data to generate newsletters and e-mail blasts.
- ✓ Would like to dynamically display information, such as product descriptions, course listings, job opportunities, and real estate listings.

Dynamic sites are truly useful, and some things on the Web just can't be done without a database and some programming. Keep in mind that a matter of degree exists here. For instance, if you're truly completely rolling your own site Search and Content Management System (CMS), you need one sort of programmer and will probably pay a big ticket. But you can hire a totally different type of programmer and pay a much smaller ticket to have someone get Google's custom search integrated with your site or get WordPress up and running for you. Similarly, in advertising campaigns, if you're going to roll your own advertising solution, that's one thing, but getting someone to integrate Google AdWords or Microsoft ads into your site, while possibly still requiring outsourcing, is a different type of resource.

Fortunately, even a relatively inexperienced designer can learn and handle the simpler programming-like tasks. For example, a novice can easily figure out how to hard-code hypertext links to individual pages and add the appropriate actions, hidden fields, and script-configuration settings to a Web form so that a Perl or CGI script provided by a Web host can process the data entered into the form by a site visitor.



Perl and CGI scripts are often used to process data collected in online forms. The scripts themselves must typically be placed inside the CGI (Common Gateway Interface) or cgi-bin folders at the root level of the Web host server to function properly. You discover more about forms and form processing in Book III, Chapter 7.

The two most common types of databases are created with MySQL and Microsoft Access, both fairly easy programs to learn. After the database has been constructed to meet the needs of the Web site, you can easily figure out the appropriate code to add to the HTML of a Web page so that the browser can pull the right data from a database on the fly.

On the other hand, if programming is something that doesn't interest you, you might want to hire a programmer to do the work for you. Finding a good programmer may take a little time, so be sure to start the search well before you actually need a programmer's services.

Finding a good programmer

Here are some suggested steps you can follow to help find a good programmer for your project:

- 1. Write out a “programmer wanted” ad. In your ad, be as clear as possible about your exact programming needs and how you'd like applicants to respond.**

For instance, if you know you need to collect e-mail addresses so that the site owner can generate and mail monthly e-newsletters, state that. Also, if you know the language you want the programmer to work in, whether it's .NET, JSP, PHP, ColdFusion, or something else, specify it.

Similarly, if you really need someone who can come to your office and work with you side by side, specify in your job posting that the applicant must live in your town and clearly state that you're seeking on-site help only. If you want to hire only seasoned programmers, request to see evidence of the programmer's portfolio and references. Always ask for references from everybody, even the student who might be charging next to nothing to build his or her portfolio.

- 2. Post your ad. Fortunately, some of the best places to look for a programmer are online.**

You can find programmers through an agency like Monster.com or the less-formal job-listing sites like Elance.com and GetACoder.com. Other, perhaps smarter, places to look are Craigslist, local programming school job boards, programmer blogs, and forums. Also consider word of mouth in programmer chat rooms.

- 3. Ask each of the applicants who responds to your ad (who, by the way, will be responding from around the globe) as many questions as you want about their experience, fees, and the time frame it may take them to complete your project.**

Use your e-mails with applicants as a way to weed out the less-capable candidates. A good way to screen candidates is to see how they respond to questions. Applicants with good answers should stay in the running, while applicants with bad or incomplete answers get tossed. It's truly fascinating to see how some people completely ignore your specific requests and try to hound you into hiring them. Stay away from those types.

4. When you've whittled your list down to two or three promising programmers, try them out, if you can, on a small test project (different from the reason you're hiring them) to see whether they're reliable, friendly, hard working, accountable, responsible, and capable.

If you're lucky, you'll end up with at least one, but hopefully two, programmers who you can begin working with on all your dynamic projects.

Book II

Designing for the Web

The 5th Wave

By Rich Tennant



“What you want to do, is balance the image of the pick-up truck sittin’ behind your home page, with a busted washing machine in the foreground.”

One of the funnest things about being a Web designer is actually creating the Web site. But a lot more goes into this process than meets the eye. To best design a Web site, you must first choose the right layout and navigation scheme, mock up a design for your site that will best promote the company while attracting the ideal target audience, and optimize all the graphics so that you can add them to your Web pages.

This minibook helps make sure that you know everything you need to know about defining the look and feel of your site, mocking up a Web design, and creating and optimizing your Web graphics.



Chapter 1: Defining the Look and Feel

In This Chapter

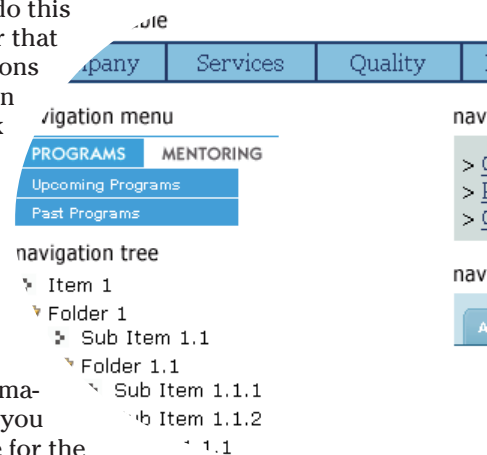
- ✓ Generating design ideas based on client and target audience profiles
- ✓ Making design size and layout orientation decisions
- ✓ Choosing appropriate site colors, fonts, and other design elements
- ✓ Selecting the best navigation system for the site's design

In this chapter, you find out how to develop the site's look and feel based on the information you gathered about the client's target audience profile and site identity. You also use the information you gathered from your client about his or her preferences for fonts, colors, layout, size, orientation, and other design elements, as well as the site's navigation.

Working with the Client to Make Design Choices

By making important design and layout decisions with the client now, before you begin your design work for the site's mock-up, you can save yourself and the client valuable time. In fact, if you do this step with all your projects, you will quickly discover that generating a *design theme* based on your conversations with the client can significantly jump-start the design process when you get to the mock-up phase in Book II, Chapter 2. Furthermore, by involving the client in finding the answers to these fundamental design questions, the issue of the site's design becomes a cocreative effort that can both enhance your relationship with the client and strengthen your role as designer.

Besides being very useful, this process should also yield some interesting and sometimes unusual information about the client's preferences, which until now you would not have known, such as a particular distaste for the color red or a penchant for center alignment. While some of these preferences can be catered to, others may need to be avoided from an



aesthetic point of view and to ensure that the site appeals more to the target visitors than any one person in charge of a company. In other words, even though your client is the Owner, Vice President of Marketing, or the de facto person of the department in charge of the new Web site, your “client” is a mere human with his own particular aesthetic preferences, and though he might need to have you follow some company-set design initiatives to keep the new site in line with the company’s other marketing materials, this person’s subjective tastes might also strongly influence what will and won’t be done with the site’s design and layout.

To illustrate, a few years ago I designed a site for a large traditional bakery that provides breakfast pastries to the hotels, retail food stores, and corporate dining rooms in the New York metropolitan area. At the look-and-feel defining stage, the owner said that although he didn’t have any specific ideas of how his new site should look, he did want the site to have a slick, clean edge to it, similar to a fine luxury car Web site like the ones BMW and Mercedes-Benz had at the time. Armed with this specific knowledge, I had a clearly defined starting point for the design and layout: Create a linear, modern design with the use of fine lines within a fixed-width layout and include crisp photos throughout the site. I then combined those ideas with a sophisticated yet neutral color palette based on collected data from the target audience profile to create an elegant site that the client absolutely loved and is still using to this day, nearly seven years later.



Occasionally you might be hired to create a site’s design around a client’s preexisting marketing materials, which means you don’t need to develop a new look and feel. If that’s the case with your current project, read through this chapter’s sections on layout and design decisions, selecting appropriate fonts and choosing the right navigation scheme. Then continue on to mocking up the design as outlined in Book II, Chapter 2, at which time you can get the specific color values (Pantone or CMYK, for example) from your client along with the logo and any other photos and graphics you may need to use.

The choices you help your client make now will dictate much of what the client sees when you’re finished creating your design mock-up. Therefore, before you begin your “look and feel” conversation with your client, read through the different sections in this chapter to discover the specific questions you’ll need to ask and to understand how you can use the answers to assist the client in making the best decisions about the site’s look and feel.

Defining a Site Theme Using Target Data

Now, finally, it is time to really use the target audience information and ideal site visitor profile you gathered and created, as described in Book I, Chapters 1 and 2. As the designer, you can take this information and convert it into a visual theme. You can use this information to anticipate the

preferences and needs of the ideal visitor, and to make design and layout decisions specific to those needs. Everything about the ideal site visitor can influence the decisions you make about layout, navigation, color, image usage, and even reading level. It's like you have a secret design assistant right there in the site's profiles.

In a way, these thematic decisions are fairly straightforward to make if you take a little time to think logically about what might appeal to the target audience. Wouldn't you agree that the navigation, layout, and color scheme for a business coach's Web site should be very visually different from that of a hard-core heavy metal rock band's site? Of course you would. Both businesses conjure up totally different mental images, and it is the ideas that those images bring to mind, plus the ones that come to mind when you reread the target audience description, that you want to use for your design inspiration.

Take, for instance, the following information about this ideal site visitor from a target audience profile:

- ✓ Male, aged 20–60
- ✓ Annual income from \$30,000 to \$65,000
- ✓ Outdoorsy, active, and an avid fisherman

For this ideal site visitor, you might choose to make the following thematic design and layout decisions:

- ✓ Keep the navigation simple and easy to use.
- ✓ Ask the client to set the writing tone of the text at around a high school reading level (suggested by the average income level).
- ✓ Use bold, woodsy colors like hunter green, brown, and navy blue, contrasted with white and neutral colors, and have lots of photographs of trees, rivers, sunsets, camping, and fishing scenes throughout the site (the ideal visitor is active and outdoorsy and likes to fish, so nature photos and this rich color palette will be appealing).

A helpful example of a site that already uses a similar theme for its ideal visitor is www.orvis.com, depicted in Figure 1-1.

As the designer, try not to influence the look and feel of the site too much with your own aesthetic. The idea here is to let the target audience profile determine how the site should look before you apply your own sense of order and style to it. On the other hand, you might begin to develop your own aesthetic style and want to include certain features in all the sites you build, such as making them all center aligned and fixed width. Nonetheless, some sites must be built to specifications that fall beyond the bounds of your own preferences, and you need to stay flexible enough to be able to

build a site that has elements in it that you might not choose. Remember, too, that the client might have very specific and unwavering needs that must be catered to.

The client is not always right

You may have heard the familiar phrase, “The customer is always right.” In Web design, however, that is not always the truth. Yes, the client is paying you for your design services, and she may have specific requirements for the Web site, such as making sure that you include certain graphics and text, but how those items look and function should really be up to you, the one with the Web design experience.

Now, in a perfect world, you’d have 100 percent authority to make all the aesthetic and design decisions, but because you’re a hired hand, you don’t. This doesn’t mean, however, that you need to bend to every one of your client’s whims. On the contrary, you should use your expertise as both a designer and Internet user to guide your client into making the best design decisions for her site.

For instance, your client may be the kind of person who is still for some reason wowed by cheesy Flash intros and primitive GIF animations. Clients like this will have definite ideas about what they think the site should look like and may even make some pretty awful suggestions to you, such as telling you they want “a big, spinning globe on the home page and text that flies in from the left and right that says something like Professional . . . Reliable . . . Fast . . . Affordable . . . and then everything fades into a big photograph of the company’s president.”

If your client makes a strange suggestion, let her know in a kind and gentle way why the suggestion is not a good idea from the visitor’s perspective (especially because Flash animations are neither search engine–friendly nor accessible to visitors with disabilities), and then be ready to make suggestions about what will

appeal to the target audience. As long as you frame your comments around promoting the company in the best way possible to visitors, your criticism should be well received.

In this example, I’d suggest to the client that the home page needs to be a place where visitors can find what they are looking for without having to wade through unnecessary information. I might even go so far as to say, “Please do not put a spinning globe on your home page. It will look cheesy, old-fashioned, and unprofessional.” Animations may be interesting to some people, but they often tend to be more sparkle than substance, which can detract from the overall goals of the site and take up valuable visual real estate that could have been better used to promote the site’s products or services. Instead, the home page should have clearly defined areas for company branding and navigation, descriptive text about what can be found elsewhere on the site that can be read by both visitors and search engine crawlers, and lead articles or teaser introductions to other information the site visitors may want to find out more about.

If your client is still a little resistant, tell her that you’ve done extensive research and have consulted with seasoned professionals (like me) who have years of experience with creating sites that appeal to the target visitors their company wants to attract. You might also say that you want to build a standards-compliant site that appeals to the target audience *and* conforms to best practices outlined by the W3C. Hopefully, both you and your client can come to an agreement that will satisfy the goals for the site and the needs of the target audience.



Figure 1-1: Use the ideal site visitor profile to generate a design and layout theme that appeals to the target audience, as this site does for its audience.

In the following sections, you discover more about how to use the target audience information to make design and layout decisions with your clients for upcoming Web projects. Specifically, you need to discuss colors, fonts, navigation, layout, size, orientation, and graphics. At the end of this chapter, you have an opportunity to put all your newly acquired skills to good use by filling out a simple Layout Checklist for your project based on the ideal site visitor and target audience profile.

Making Basic Layout and Design Decisions

You and your client definitely need to make a few decisions about the site's look and feel well before the design gets under way. In particular, you should determine how the Web site layout will be positioned inside the browser and how it will fill the browser space. That means deciding whether the layout will be fixed in width (like Yahoo!'s home page) or expandable (like

Amazon.com), what its orientation will be relative to the top-left edge of the browser window, making the site either centered, left, or right aligned, and whether the pages will be printer friendly on their own or whether you'll need to build a second CSS file to handle how the pages will print. In the following sections, each of these layout issues is addressed, starting with the layout size, and each of them both forms a general framework for creating the mock-up and provides a glimpse at how the site might potentially be built in HTML and CSS.

Some designers help their clients make decisions about the layout based on foreknowledge of HTML, CSS, JavaScript, and the site-building process. This is something I do quite often because my first-hand knowledge about building navigation systems can greatly help the client envision how some parts of the site will look before the design is even started. For instance, you can have the client look at select Web sites to preview how different navigation systems and layouts look and function and assist him in making the right decisions for his site.

Even if you have little to no experience with building Web sites, you can still help your clients make informed decisions about their sites at this stage in the process. Specifically, you should try to get them to make decisions about the layout width, expandability, orientation, and printability, as well as help them choose appropriate color and font palettes that will be appealing to the target audiences.

Choosing a size for your site

As you probably found out from your market research, most computer monitors come with a factory preset resolution of 1024 x 768 pixels or higher. Though most people never adjust this setting, some do increase or decrease the monitor resolution from as low as 640 x 480 to as high as 2560 x 1600 with some of the new widescreen LCD monitors.

Here are some examples of typical resolutions for LCD monitors:

- ✓ **14–15-inch:** 1024 x 768 (XGA)
- ✓ **17–19-inch:** 1280 x 1024 (SXGA)
- ✓ **20-inch+:** 1600 x 1200 (UXGA)
- ✓ **17-inch:** 1280 x 800 (WXGA, Widescreen)
- ✓ **19-inch:** 1440 x 900 (WXGA+, Widescreen)
- ✓ **20-inch:** 1680 x 1050 (WSXGA+, Widescreen)
- ✓ **24-inch:** 1920 x 1200 (WUXGA, Widescreen)
- ✓ **30-inch:** 2560 x 1600 (Widescreen)

With no control over this visitor variable, how does a Web designer select a layout width for a client's Web site without alienating or infuriating some of the visiting audience? Follow these tips:

- ✓ For a general audience on the Internet (which is a public network, accessible to anyone with a computer, browser, and Internet connection), the answer to the layout width-versus-resolution question in most cases is pretty simple: Design for a size that can display readable text to all viewers, regardless of monitor resolution. By creating a design for monitors set to an 800 x 600 resolution and making columns of text (with font sizes set to 10 pixels or larger) no wider than 500 pixels if you can help it, everyone should be able to read content on the site. Furthermore, when you take into consideration that most of the *browser chrome* (those browser elements like scroll bars, the status bar, the navigation bar, and the Favorites bar) takes up some of that design space, the actual safe design size for a monitor set at 800 x 600 becomes more like 760 x 420.

Figure 1-2 shows a screen shot of my Web site, which has an overall width of 720 pixels and two columns of content with widths set to roughly 360 and 280 pixels, leaving a little room for spacing between the design's edges and the two columns of content.

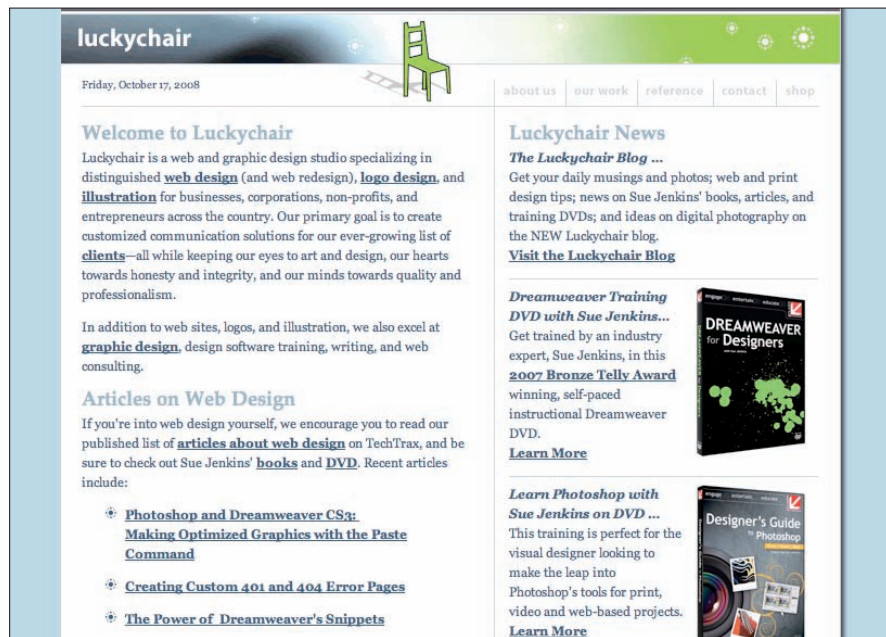


Figure 1-2: Fixed-width layouts designed for 800 x 600 resolution can range as wide as 760 pixels and can contain one to three columns of content.

- ✓ For clients who have a limited or very specific audience, you may need to gather more information. Picture, for example, that you've been contracted to design a Web *intranet* (a private network of interconnected local- and wide-area networks using Web protocols to share information among the members of an organization) for the human resources department of a large corporation. You might learn from the client that the target audience consists of only PC users with desktop monitors set to 1024 x 768, and all the users access the intranet site using only Internet Explorer 7.0. Armed with these details, you may help the client choose to create a much wider design with multiple columns that maximize the use of the known usable browser space for the employees' PC monitors in IE 7 — about 1000 x 580 pixels.



When in doubt, go for the 760 x 420 design size. For a more expansive discussion on monitor resolution and actual design space, turn to Book II, Chapter 2.

Selecting a fixed-width or flexible layout

Next, consider whether to make the site's design using a fixed-width or an expandable design layout:

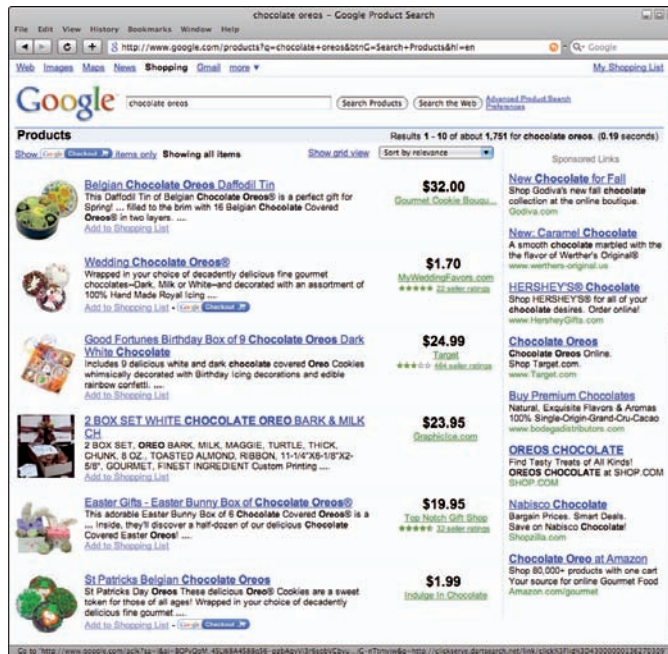
- ✓ **Fixed-width layout:** A fixed-width design means that the content on the Web page will remain fixed within a predetermined content area and that any overflow content will expand the page vertically rather than horizontally, like on the Twitter.com Web site shown in Figure 1-3. Everything outside the fixed-width layout is considered part of the background design of the page, which can include color and graphics.
- ✓ **Expandable layout:** By contrast, an expandable-width design is one in which the Web site layout spans the full breadth of the browser window and includes one or more columns of information that can expand and contract with the width of the browser window, displaying the page as you'd see on the Google Shopping search results listings, like the one shown in Figure 1-4. The expandable layout uses percentages relative to the browser window's width. Some designers refer to this technique as *liquid design* or *fluid design*.

The expandability or lack thereof of a Web site design helps determine which techniques you can and should use to build the site in HTML. For example, a fixed-width site with a left/top browser orientation can easily use absolutely positioned layers in the layout, whereas a fixed-width design with a center alignment or a site with a liquid design cannot. Table 1-1 outlines the pros and cons of each format.



Figure 1-3: Fixed-width layouts can expand vertically when you have more content than can fit inside an expanded browser window.

Minimized



Maximized

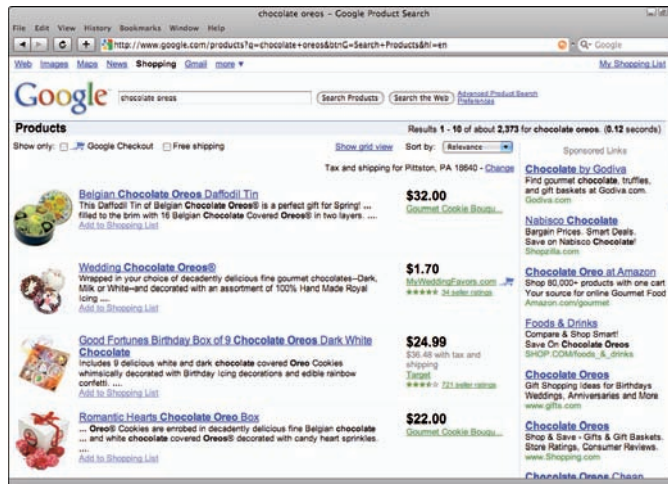


Figure 1-4: Fluid layouts expand horizontally to fill the entire browser window width as well as expand vertically to handle content that overflows past the browser window's height.

Table 1-1 Fixed-Width versus Expandable-Width Designs

Type of Design	Pros	Cons
Fixed-width	The designer can predict how content will look on a Web page before the site gets built in HTML.	Visitors with larger monitor resolutions will see more blank space surrounding the fixed-width design than those with smaller monitor resolutions. Not a drawback, but a design consideration that can be creatively utilized.
Expandable-width	The site will always fill the entire browser window, regardless of the visitors' monitor resolution, and whether the browser is minimized or maximized.	The text in the main area of the page can become so wide and extended (set to 100% of the browser window) that it's difficult to read.



Although fluid design was a more popular design solution in the early days of the Internet, it's less popular now as a layout technique for many small and medium-size businesses, artist portfolio sites, and blogs. The exception to this trend would be for sites like online e-mail accounts (Gmail, for example) and search engine results listings, where extensibility is important because it allows those sites to display more content when the browser is maximized. For everyone else, the shift away from fluid designs likely has more to do with page readability and printability (see the next section for more on this) than with *accessibility* (the ease with which visitors with disabilities and nonhuman devices can access and navigate through a site). Think about your own preferences. Wouldn't you rather read a long, narrow page than a wide, short one? And how important is it to you that a site's design extends the full width of your browser window at any monitor resolution? It probably doesn't matter that much to you, which means that the average Internet visitor for any one site never considers the difference between a site with an expanding layout versus one that has a fixed width.

If you and your client choose the fixed-width design size, the next aesthetic design issue that you need to decide upon is the orientation of the page relative to the browser window. Will the design begin fixed to the upper-left corner of the browser window, leaving empty space to the right of the design, like HGTV.com, or will it be anchored to the top of the page but aligned to the center of the browser window with empty space to both the left and right of the fixed-width design, like HomeDepot.com? Figure 1-5 illustrates the general differences between each layout.

136 Making Basic Layout and Design Decisions

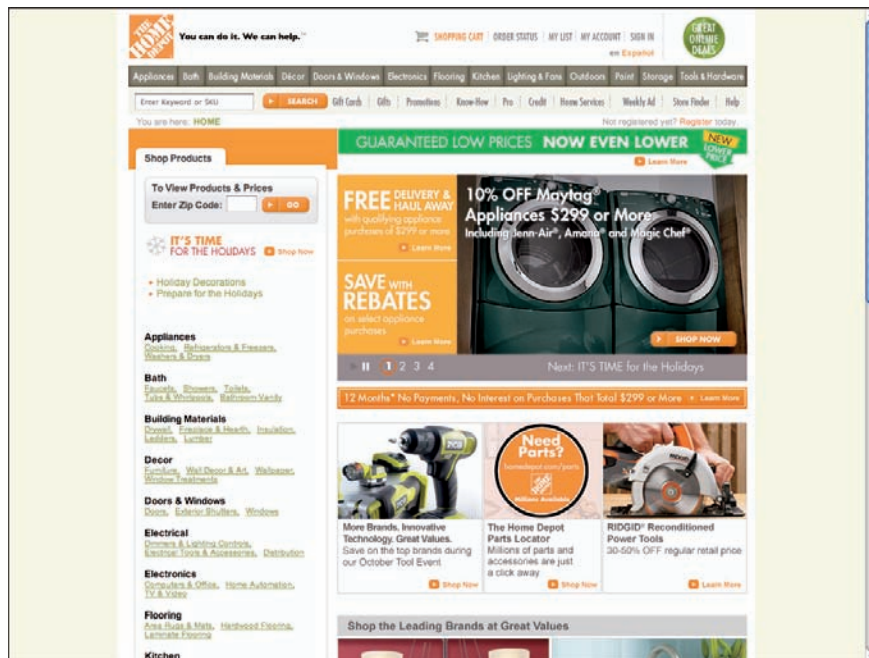


Figure 1-5: The top site has a fixed-width layout with a center browser alignment, and the bottom site has a fixed-width layout with a left browser alignment.

Neither solution is better than the other, so choosing the right one is simply a matter of taste. In recent years in the United States, the prevailing trend has been to create fixed-width designs that are center aligned to the browser. As you know, however, trends change. Ultimately, you should choose a design layout that will suit the client's needs and the Web site's content and, above all, cater directly to the target audience's preferences.

Choosing a method for printing the layout

When designing a page with content that visitors might want to print and keep, you need to think about how the design layout will print. Many Web pages contain company logos, navigation, images, banners, advertisements, and other page elements that the visitor probably doesn't need on the print-out, and you don't want visitors to use more paper (save a tree!) by printing redundant, unnecessary page elements.

To help solve this issue, do one or both of the following:

- ✓ **Make sure that all the key parts of the printable content fall within the leftmost 700 pixels of the design layout.** That should work for both fixed and flexible layouts to prevent the content from getting cut off of the right side of the printed page. Navigation buttons or other elements in the layout might get cut off on the printed page if they extend past the 700-pixel mark, but that in itself might be insignificant to the visitor if the main content gets printed intact.
- ✓ **Design a second Cascading Style Sheet for the site that can be used automatically anytime a visitor wants to print a page.** By creating and using a secondary CSS for print (that includes a custom class style with the block display set to None for the elements contained in layers that should be hidden from view on the printed page) in addition to the default media type for all media, certain page elements can be blocked from displaying on the printed page. Book III, Chapter 4 explains Cascading Style Sheets in more detail. For now, you just need to know that this is an option you can include in your design.

A third option is available, but its inefficiency should sway you against using it as a regular solution unless it only applies to a handful of pages on the site or for sites that are database driven, in which case these alternative printable HTML pages could be automatically generated on the fly. The third option would be to create an alternative printer-friendly version of the page in HTML that is set to a fixed width so that the content won't be cut off along the margins by the average home or office printer.

Picking a color palette

The colors you choose for a Web site project do more than just decorate the content. They can also communicate ideas, evoke emotions, alter moods, and convey unspoken psychological messages about the owners of the site and who they're trying to appeal to, and the professionalism and quality of any services and products being offered and/or sold.

When you pay attention to the details about the ideal site visitor, you should be able to translate that identity, that personality, along with the other demographic information you collected, into a tangible and at times obvious Web color palette. In other words, to choose an effective color palette, you need to select colors that are consistent with the target audience's cultural, social, and industry-standard preferences.

When presenting your client with a color palette, you can create a simple graphic in Photoshop, Illustrator, Fireworks, or some other graphics application and then send the client a PDF, JPG, GIF, or PNG file to preview and approve. Figure 1-6 shows an example of how you might present your client with a sample color palette so that he can decide whether it matches his vision of the ideal site visitor.

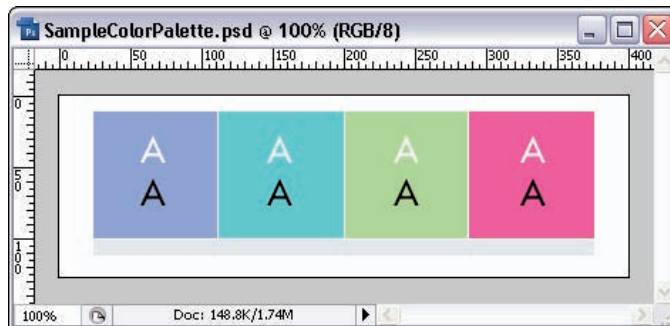


Figure 1-6: Presenting the color palette in GIF format.

The following steps outline a basic process you can follow to select an appropriate color palette for your Web project:

- 1. Refer to your notes about the site's identity and consider what colors are (or are not) compatible with that identity.**

With many businesses, the nature of the work tends to evoke consistent mental images from which an appropriate color palette can be created. For instance, if you are creating a site for a health spa and yoga retreat, you might stay away from reds (which evoke feelings of love, heat, and

excitement) and opt instead for calmer, earthier colors like brown (natural), beige (calming), light blue (truthful), and green (harmonious).

2. Refer to your market research to see whether you'd like to copy (or avoid) any industry standards.

When used effectively, colors can help sell a product and engender brand loyalty among consumers. However, when used inefficiently, color can distract from a product or service message and work against the goals of the site.

Some industries even seem to have publicly accepted color schemes that, if veered away from, might adversely affect business. Financial institutions, for example, seem to favor navy blue, red, and burgundy as their colors of choice, so unless the client is a true rebel, the idea to use pink or purple as the primary color for a bank's Web site might not be such a good idea.

3. Using the target audience data, note any cultural preferences or special considerations that the site should reflect.

Colors represent specific meanings in different cultures across the globe. For instance, black is the color of mourning in the United States, but in other cultures, mourning is represented by blue (Iran), red (South Africa), white (Japan and China), and yellow (Burma). Therefore, if your client is based in the United States but sells products or services worldwide, she might do well to use neutral or industry-standard colors on the Web site rather than select a color palette that might accidentally offend international visitors.

4. If you know your target audience's demographics, use that data to assist you with color selection, too.

The target viewer's age, sex, income, and education can provide some great cues for selecting color.

For example, younger audiences tend to like brighter colors, whereas mature audiences are more drawn to pastels and neutral palettes. Likewise, men tend to be drawn to cool colors like blues, purples, and greens, and women to warmer colors such as orange, pink, and red. If the target audience belongs to some kind of social subgroup, like motorcycle riders, organic gardeners, ham radio operators, or scrapbooking enthusiasts, you can more easily tailor the color palette to those groups' particular preferences.

5. After considering your notes and the ideal site visitor, select a primary color to dominate the site design.

The primary color of the site is the design's main color, like the wall color in a room of a house. This color typically helps delineate the site's layout against the page background, like having a swatch of color across the top of the page, behind a navigation area along the left margin, and/or behind a sidebar area along the right.

6. Based on the primary color, choose a secondary color.

The secondary color is complementary to the primary color of a Web site's design. Secondary colors can help offset content areas from navigation areas, and these colors can be used for decorative accents for things like the page background color (behind a fixed-width design), other page subdivisions, and/or page and section headings.

7. Choose a third color (the accent color) for elements such as buttons, bullets, hyperlinks, headlines, and other decorative elements.

Any more than three colors might be overkill and distract the visitors from finding the information they're looking for. However, sometimes a larger selection of colors might work well within the context of a particular site. I've seen plenty of great-looking sites that have a primary, secondary, and accent color, plus several other colors used within the navigation system to differentiate the different sections of the site or to otherwise help organize the content being displayed.



If for some reason you don't feel confident in your color-selecting abilities yet, consider using a special color selection software program that can take a more scientific approach to choosing a color palette. For example, Color Cache (www.colorcache.com) and Color Wheel Pro (www.color-wheel-pro.com) are two color programs that generate complementary accent colors based on your choice of a primary color. The more you work with color, the more you will come to develop your own internal sense of what works and what doesn't. The right colors can make or break a site, so be sure to choose your colors well!

Choosing the right fonts

For the most part, all the text that appears on a Web site needs to follow certain guidelines:

- ✓ **Text must be marked up as HTML text.** This means that a primary HTML font should be selected for the bulk of the selectable text visible on a Web site. If desired, you can use a secondary HTML font or graphics for headings, and you can use additional fonts if desired to set content in certain specific regions apart from other areas on the site, as with advertisements in a sidebar.
- ✓ **The use of graphics that contain text should essentially be limited to headlines, page headings, and pull quotes for whenever you need to use a specific font.** By contrast, when a site uses HTML text, the page loads faster in a browser window than it would if you presented the text on the pages as graphics. This includes any headlines, page headings, and pull quotes you write with HTML and style with CSS. Furthermore,

images containing text can't be "read" by search-engine crawlers or other assistive Web devices. True, image HTML tags can include attributes such as alternative text and long descriptions, but images with a lot of text should be the exception on a site rather than a rule.

- ✓ **The fonts you choose for your site need to be cross-platform (PC and Mac) fonts.** In other words, the fonts you select for your project must be preinstalled on the visitor's computer for the page to render the text in the desired font(s).

Unfortunately, all this means that unless you use advanced techniques like sIFR for your type (see the nearby sidebar about sIFR), the pool of available fonts you can use for your Web site is rather limited. Right now, the list of safe-to-use HTML fonts, as shown in Figure 1-7, includes Verdana, Arial, Helvetica, Courier, Courier New, Times, Times New Roman, Georgia, Geneva, Tahoma, Trebuchet, Comic Sans, Impact, Serif, and Sans-Serif. Other fonts that have cross-platform equivalents include Palatino Linotype, Book Antiqua, Lucida, Arial Black, Symbol, Webdings, and Wingdings.



The three most popular primary fonts in recent years have been Verdana and Arial for the sans-serif fonts (unornamented fonts) and Georgia for the serif fonts (fonts with decorative ascenders and descenders on the stems and ends of letter shapes), all displayed as black text on a white background.



Figure 1-7: Use one or two of these Web-safe fonts for all your site's HTML text.

Despite all these font limitations, you can still add quite a bit of pizzazz to your text. The different elements of the content (headings, bylines, footers) can be made larger, smaller, bolder, or italicized, and you can display them in different colors, among other things, all through the magic of CSS.

When setting up your pages in HTML and CSS, you have the option of creating *font sets*, whereby text on any Web page on your client's site is rendered in a browser window based on the font availability of the computer system viewing the page. One such typical font set is Verdana, Arial, Helvetica, and Sans-Serif. When this font set is used, Verdana would be the

preferred font to view the Web page in. If that font were missing from the visitor's computer, the text would be rendered in Arial, the next font in that set, and so on.

Sometimes, however, a client might require that a particular noncross-platform-available font be used on the site. In these cases, explain the cross-platform issues to your client and suggest a workaround that includes using HTML text for the bulk of the content and using sIFR for creating page heading graphics (About Our Services) and/or bylines (Free shipping on any order \$25 or more!) in the desired font for accents throughout the site, such as the images shown in Figure 1-8. Whenever possible, though, you should insist that your client use HTML text marked up with CSS.



Figure 1-8: You can use special fonts as graphical elements on a Web site.



You should also know that browsers running on Mac OS X render Web fonts differently than do browsers on any current PC, regardless of OS. If you were to do a side-by-side comparison, what you'd find is that the Mac browsers display true PostScript fonts with antialiasing, which makes each of the letter shapes appear sharp and smooth, especially around the curved edges. By contrast, PC browsers use a different technology to render text in a browser, and so far, it just doesn't match the quality of the Mac. Therefore, if you're developing on a Mac, be sure that you also test your pages on a PC so that you and your client aren't disappointed when you preview your Web pages using a PC browser.

Selecting a Navigation System

The navigation system on a Web page is typically the primary tool that visitors use to jump from one page of a Web site to another. The key ingredient to a good navigation scheme is *usability*. The navigation should be easy to understand and easy to use and the information on a site easy to locate, even by the most inexperienced Web visitor. This is different from *accessibility*, which has more to do with the HTML coding of the elements presented on the page rather than how the site is designed. In addition, the placement and functionality of the navigation system should be consistent throughout the entire site so that visitors aren't forced, even on one page, to guess how to access the different pages on the site.

Web sites with sIFR fonts

sIFR, which stands for Scalable Inman Flash Replacement, is a technique developed by Mike Davidson of Mike Industries that improves on the original IFR system developed by Shaun Inman, which allows designers to specify any system font as the font to display in a Web browser, regardless of whether that font resides on the visitor's computer. Simply put, sIFR is a Flash movie that uses XHTML and JavaScript to replace designated text with Flash-rendered text. Text using sIFR, which scales fonts to fit inside a defined space on a

Web page, can display beautifully in browsers with the free Flash player installed and make sure that plain text appears in its place in browsers without Flash. Best of all, the entire process is seamless to the visitor.

In the example shown here, you can compare the same page with (top) and without (bottom) sIFR font replacement. To learn more about this breakthrough process, visit www.mikeindustries.com/blog/archive/2004/08/sifr.

Book II
Chapter 1

Defining the
Look and Feel



Thankfully, most Internet visitors today are familiar with using the most common types of navigation systems. Navigation schemes usually consist of a set of text or graphical button links, with or without subnavigation menus, and visitors know they must click the links to navigate from one page to the next. Some navigation types have more intuitive interfaces than others, but they all tend to provide links to other pages on the site or external to the site. A drop-down form style “jump menu” (where the visitor selects a destination from a drop-down menu and is instantly taken there upon release of the mouse button), for instance, would arguably be easier for an Internet novice to figure out how to use than a navigation system that uses shapes, symbols, or other nontextual graphics as links to other pages on a site.

In your own Internet experience, you’ve probably already seen examples of the most popular types of navigation systems, and although they might look graphically different from one another — you find navigation tables, navigation menus, navigation trees, navigation lists, and navigation buttons — each of them functions essentially in the same way. Figure 1-9 illustrates simple examples of each of these navigation types.

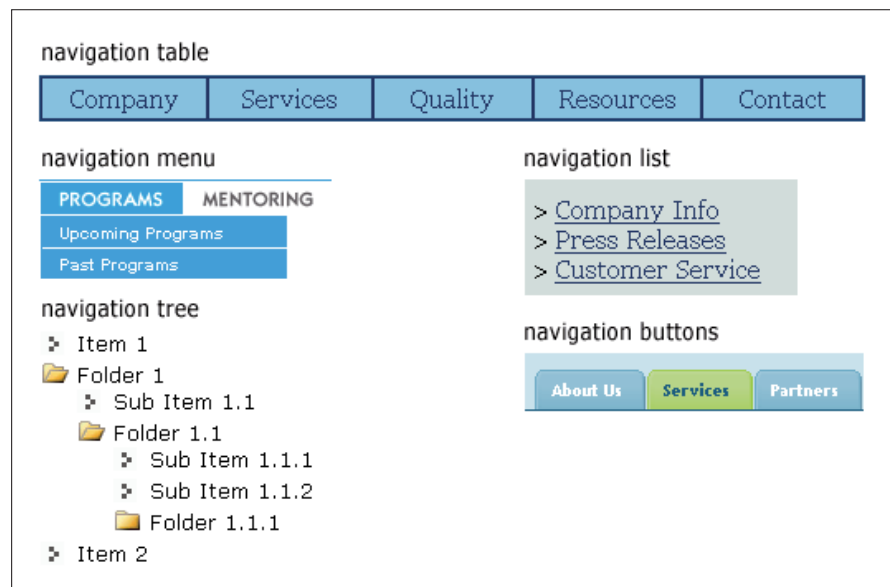


Figure 1-9: Web page navigation comes in a variety of styles.

Choosing a location and style

The placement of most Web site navigation systems tends to be either at (or near) the top of the Web page or along the left or right margin of the page layout. When a site is larger than just a handful of pages, some of or all the navigation elements might serve as both links to their respective pages and triggers that open a submenu to additional navigation links, which themselves might also have additional submenu navigation links to their respective pages.

Submenus tend to either drop down, pop up, or fly out to the left or right of the main navigation option being selected. How any one submenu system functions largely depends on the technology powering it, such as JavaScript, Java applets, Flash, DHTML, or CSS.

Talk with your client about your ideas for the navigation, and plan to have some example URLs handy that you can refer to when defining which navigation systems you think might work best for the client's site. To be sure, many clients will already have a general idea of how they'd like the navigation to function, and they may even present you with sample URLs of the type of navigation they'd like to have. Whether you can create a comparable navigation system is another question, however.

When researching someone else's navigation system, take a look at the source code to see whether you can identify whether the navigation is using CSS, JavaScript, Flash, or some kind of applet or plug-in to drive its functionality. If you can't quickly figure it out, you may need to suggest some kind of alternative to your client. Some of the fancier navigation menus you might find are often built by third-party providers using Java, a programming language (not to be confused with JavaScript, a scripting language).

Not all fancy navigation menus require programming, however. Many third-party menus these days use HTML and CSS. OpenCube.com, for example, offers QuickMenu, an affordable (\$389 for Web developers) CSS menu solution that allows you to use both graphics and text to create multilevel cross-browser-compatible menus to suit nearly any Web site. If you're looking for a free menu generator (or a more affordable developer version for just \$64) that relies a little more on JavaScript, the folks at JavaScript.CoolDev.com have developed a cool little product called COOLjsMenu that quickly generates cross-browser-compatible, customizable, drop-down menus. Of course, if you're baffled by the whole concept of menus, you can always hire a CSS whiz or a JavaScript programmer to create a navigation system for you.

Determining how to handle submenus

If you or your client is having difficulty choosing the right navigation and subnavigation system for the site, take the following steps:

1. Determine the location for the navigation on the page.

Across the top and down the left side of the page are the most common locations for the navigation. However, some sites have begun to use a combination layout where the main navigation falls across the top and the subnavigation falls along the left or right margins. Some liquid design sites have even put the entire navigation system on the right, submenus and all. When in doubt, think of the site visitor's perspective and ask yourself where it would be easiest for him or her to find it.

2. Decide how the subnavigation, when needed, will function.

When the main navigation is on the left, the subnavigation typically displays to the right of the main navigation as a fly-out menu. By contrast, when navigation is located near the top of the page, the subnavigation tends to vary quite a bit more:

- The most common top-menu subnav type is the drop-down list, but if the top menus are positioned low enough, the subnavigation menus might pop up above the navigation bar.
- The second most common submenu type tends to be a linear submenu directly below the main navigation links, usually placed atop a color bar that matches the background color of the main navigation link or button, as seen on the Barnes & Noble Web site (www.bn.com).

A newer solution that designers have been using quite a bit more in the past couple of years is to make subnavigation accessible through links along the left, but more typically the right, margin of the page.



Oftentimes, the site map can provide cues about which type of navigation is most suitable for a site, so if you're feeling stuck or undecided, look to the site map. Web sites with multiple categories and subcategories might benefit from relying on more straightforward navigation systems using drop-down lists, whereas sites with fewer pages overall can take more creative risks with how they display their navigation to the visitor.

Whichever navigation scheme you choose with your client, make sure that it makes sense within the context of the overall layout and the content being presented. Above all, navigation should be both professional looking and user friendly.

Creative navigation systems

If your Web project calls for it, don't be afraid to deviate from the standard navigation systems you're used to seeing. Each Web site is like a puzzle that must be carefully put together. The puzzle pieces are the goals of the client, the content that needs to be displayed, the preferences of the ideal site visitor, and considerations on how to best present the information to the visitor.

When famed New York fine art photographer, author, and instructor Amadou Diallo came to me to redesign his Fine Art Photography Web site (www.diallophotography.com), he knew that the design for his new site needed to have more room to present the images in his growing photo galleries than his original site did. This meant that a traditional drop-down or fly-out subnavigation menu wouldn't be appropriate for his new site's design, because he wanted nothing to obscure the experience of looking at his photographs. In addition, he also had a very clear personal philosophy upon which he wanted the new design to be based:

"I'm very much taken with a Japanese aesthetic, which places importance on the interaction of single elements as they create a unified whole. It is the idea that to truly appreciate a forest, you must understand the beauty of a single leaf on a single tree. Similarly, I strive to produce images that, taken individually, express a particular facet of human experience but, when viewed together, display a unifying element

or vision, which is my way of seeing the world. I would like visitors to come away with a sense that the site is more than a means to an end to show my work, but an integral part of their experience of the images."

Having some familiarity with the Wabi Sabi aesthetic, I immediately understood Amadou's goals and was able to develop a clean, simple, organized layout (loosely based on the Fibonacci golden section) that also contained a sense of natural imbalance to being inline with Amadou's vision. In the header of the page, I designed a navigation bar that would take up as little room as possible in the layout directly beneath the site name and logo, with the subnavigation for the gallery section on a bar directly below that. For the footer at the bottom of the page, I somewhat mirrored the design in the header to create a sense of balance in the overall layout. This left a large white canvas space between the header and footer to highlight the display of images in Amadou's five galleries as well as to place relevant content in the other pages on the site. Further, by selecting a leaflike organic color palette for the main layout and navigation elements, the rest of the design space was used to place visual emphasis on his elegant black-and-white photography. After Amadou approved the layout, I used a mirror layout concept with a black, white, and gray color palette to redesign the Digital Printmaking side of his site (www.diallophotography.com/digitalprintmaking).

Organizing the Site's Look and Feel

One of the most fun and challenging aspects of creating a Web site's look and feel is to figure out how all the pieces of the puzzle fit together. The *look and feel* is a general term that refers to the GUI (graphical user interface, pronounced *goeey*) of a Web site, which defines the overall design appearance and functionality of a site before the site is designed. Within the GUI, you have the main navigation and subnavigation, the logo and other branding, possibly a tag line, some photographs, some copy, illustrations, and other miscellaneous bits of content that need to go on the page. You also have your chosen fonts and colors, and you have made a few decisions about layout size, orientation, and other preferences of the ideal site visitor to assist you in creating a design that can project the image that the client wants while attracting the target audience that the client wants.

Fortunately, you can discover a few tricks about how to put all the pieces together in a visually pleasing way. Two factors, in particular, can greatly help get the design going in a positive direction, namely, positioning the brand and designing your site on the grid system.

Positioning the brand

The brand is the site's identity, which usually consists of a logo or logotype, a tagline or catch phrase, and occasionally some kind of photo, illustration, or other graphic treatment.

On a Web page, the most important part of the layout is the top 400 pixels. As long as you position the branding within this area, the visitor can see it when first arriving on the Web site. Most sites position the brand in the top 150 pixels, aligned either to the top left (like YouTube.com), top center (like Google.com), or top right (like AltPick.com) area of the layout, as illustrated in Figure 1-10.

No one perfect position for the branding exists. Instead, you need to find the right place for your particular logo and branding information within the context of your particular site. In essence, you have three choices, so try them all and pick the one that looks the best! On the other hand, the answer may come directly to you from your client's marketing standards, which may prescribe the exact location of the branding for you.

Designing layouts on the grid

If you've done any design work, you've probably heard about working on a grid. Each layout (and this goes for print projects as well as Web designs) has a defined space on top of which a grid can be placed to divide the space into equal parts. For example, if your design will be fixed width at 760 pixels

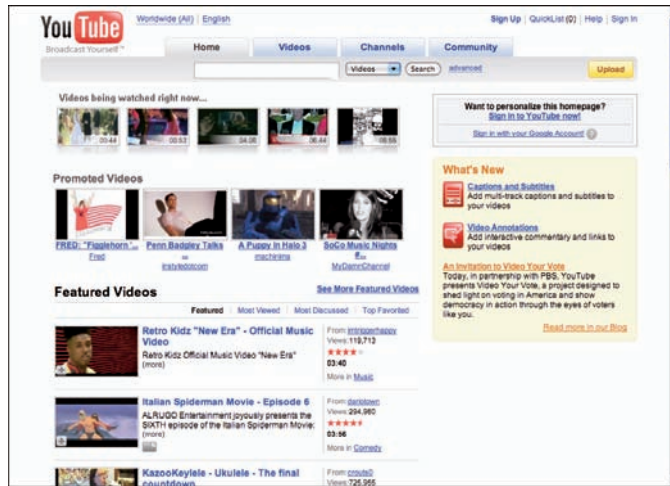


Figure 1-10: Position the brand to the left, center, or right along the top area of the site.

Book II
Chapter 1

Defining the
Look and Feel

wide and roughly 600 pixels high, you can split the workspace in half down the middle both vertically and horizontally. You can then add additional divisions between the new dividers, as shown in Figure 1-11, to further carve up the space into even increments. To vary the layout, you could also carve up the space into thirds and subdivisions of those thirds, as illustrated in Figure 1-12.

When you place your content elements within a grid system, a natural ordering of those elements begins to emerge. Navigation buttons can be equally distributed, graphics can be placed strategically in visually harmonious spots, and the whole layout can begin to take on an otherwise missing sense of cohesiveness. Of course, you don't have to design on a grid, but it can be a good jumping-off point if you're more of a rebel designer.

Making a layout checklist

To help you begin to get a sense of what is needed for each individual Web project, you may want to create a general layout checklist that you can use to guide you in the process of preparing to create your design mock-up.

Keep in mind that there are no right or wrong answers. Each project has its own set of circumstances, requirements, and guidelines, given the descriptions for company and ideal visitor.

Your checklist should look something like Figure 1-13. If you use a form like this for each project you work on, you can take each site's design ideas right into the mock-up phase. Then, as you're creating the design, you can regularly check the form to see whether the design is consistent with the site's design goals.

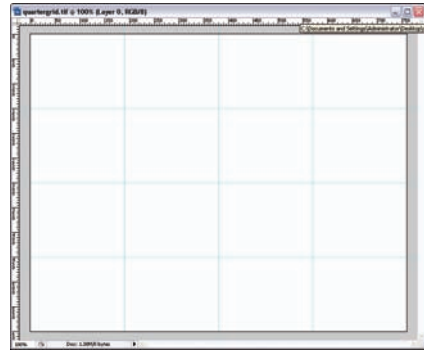


Figure 1-11: Evenly divide the space in your layout using the grid system.

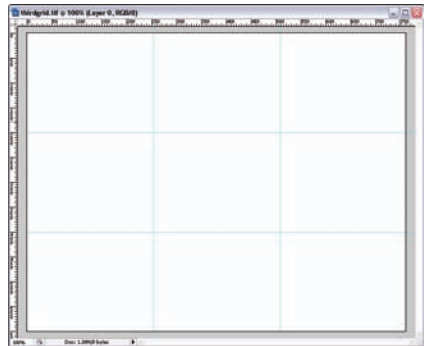


Figure 1-12: To vary your layout, divide the space using the rule of thirds.

Company name:

Description of the company image:

Description of the target audience:

Description of the ideal site visitor:

Colors: Primary: _____ Secondary: _____ Accent: _____

Fonts: Primary: _____ Secondary: _____ Accent: _____

Navigation orientation: Vertical Horizontal

Navigation alignment: Top Left Right

Navigation style: Drop-down Fly-out List Expanding Tree Table Flash
Other: _____

Branding alignment: Top Left Right

Layout size: 800 x 600 1024 x 768 Flexible Other: _____

Layout orientation within the browser: Left-aligned Center-aligned

Printability: Yes, good as is No, needs 2nd CSS for printing

Photographs, graphics, illustrations:

Cultural, social, or industry-standard preferences:

Other design ideas: Cold and edgy Warm and inviting Hard edges Soft corners
Angular Curvy Other: _____

Figure 1-13: Use a checklist to help guide you in creating the site mock-up.

Chapter 2: Mocking Up the Design

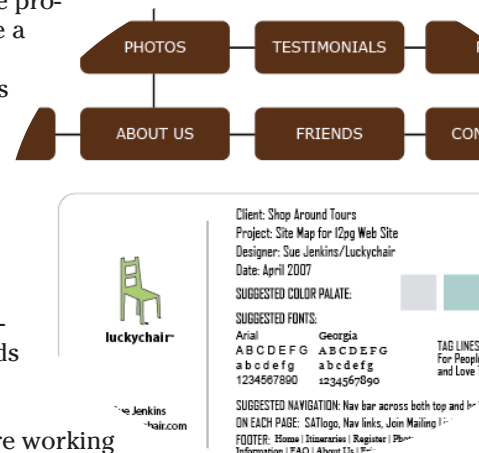
In This Chapter

- ✓ Using the visual site map as a guide for the layout
- ✓ Selecting and setting up a graphics program
- ✓ Mocking up the site design
- ✓ Strategically placing page elements in the mock-up
- ✓ Designing additional graphics for the site

If you already read the previous chapters in the book, you've covered quite a bit of ground laying the foundations for your project. With the site's purpose, a grasp of the target audience, and the definition of the ideal site visitor, you should now have a firm identity for the site that includes ideas about colors, fonts, graphics, photographs, layout, navigation, and other design-related site components. You may now take all that information, combine it with the content on the home page and your visual site map, and jump feet first into your chosen graphics software program to create the actual mock-up for the design.

In this chapter, you mosh all that information, your research, and your design decisions into a one-of-a-kind layout for your Web project. Hopefully what you'll quickly find is that by using the project's site map as your guide, you can easily generate a unique, creative, and compelling Web page mock-up that best represents the project goals and the client's vision for the site, all the while with an eye of what will best appeal to the target audience. Sounds like a tall order, but it's really not. All you must really do is put all the parts of the site together in a visually pleasing way — whatever that means to you — given your aesthetic preferences, the client's specific requests, and the visual pieces you have to work with. Think of it as a balance between your aesthetics, what the client wants, and what the site needs to be successful.

Creating a good design, no matter what project you're working on and how many design restrictions you feel you have, requires that each of the elements for the site — text, fonts, colors, design elements, and so on — is strategically placed on the page. For some sites, this can be



somewhat like solving a complicated puzzle, but if you take your time and follow the basic organizational rules outlined in this chapter, you should be able to come up with at least one, if not two or three, suitable layouts. One extremely useful rule is to include on the mock-up a graphic example of the type of navigation system the site will use, preferably with one of the navigation links shown in its “rollover state” when applicable. This can greatly help your client envision how the completed Web page will look and function when converted into HTML.



If you'd like to find out about some of the more popular navigation options before building the mock-up, turn to Book III, Chapter 6. After you have chosen the best navigation system for your project, return to this chapter to complete your mock-up.

Understanding the Value of a Mock-up

There are several very valuable reasons why you should create a mock-up (or mock-ups, if you've contracted to create more than one for your client to choose from) of a Web site in a graphics program before you build it in HTML and CSS. Here are the top four reasons:

- ✓ **Visual representation:** First and foremost, the mock-up is meant to provide your Web client with a visual representation of how the completed site will look in a browser window before you actually spend any time generating the graphics or building the Web pages in HTML, CSS, JavaScript, and any programming language you may decide to use. In other words, the mock-up becomes a kind of blueprint that both the designer and the client can refer to when communicating about the specifics of how the site will look and function. It's also a great way to communicate dynamic needs to programmers and other Web developers who may be assisting you with the construction of your site.
- ✓ **Easy modification:** Should the design require any adjustments (which it inevitably will), you can more easily modify a single graphic mock-up than rebuild or modify the code on all the pages on a Web site. Most Web clients do like to have some say in the design process. Allowing for client feedback during the design phase is a nice way to share the decision-making power and arrive at the best possible final design. Of course, you only want to show your client your very best work so that the choices he can assist you in making will be ones that are acceptable to you from a design perspective.
- ✓ **Design unification:** Ultimately, the mock-up allows you to put all your design ideas in one place, providing a single, unified vision of the site's look and feel that you can constantly refer to as you build the site. You will also use the layout as a guide from which you can generate all the additional necessary graphics for the remaining pages on the site.

- ✓ **Satisfaction:** For many Web clients, the site mock-up has an emotional component. Not only is an approved mock-up a clearly definable milestone within the Web design process, but it also provides the client with a great sense of accomplishment toward the finished project.

You should expect, after presenting the initial design to the client, to go through at least one to three rounds of revisions before the client approves the design. Two rounds are often sufficient, but given the fact that many designers now communicate with their clients exclusively through e-mail and voice mail, three rounds allow you to resolve any possible miscommunication that might naturally occur.

Whether you'll be designing one, two, or possibly more mock-ups for your client's Web project, be sure to limit the number of revisions the client can make to her preferred design, and clearly remind your client of this limit so that the project stays on track within the predefined site budget and time frame. Some designers allow unlimited changes until the client is satisfied. However, in my experience, limiting the number of revisions to three or less (or a maximum of five in special circumstances) helps keep the project moving forward.



If you include some kind of clause in your design contract that states the maximum number of revisions to the design before any additional fees kick in, you can inform the client of her responsibilities and your expectations in advance. For example, you might want to state that the contract “allows a maximum of three rounds of revisions to the initial design and that any additional work beyond this maximum shall be automatically billed at \$X/hour.” This simple clause can greatly help you prevent the more aggressive clients from asking more from you than they've agreed and contracted you to pay for.

In addition to using the mock-up as the foundational graphics you need to build the site, the mock-up also provides you with all the necessary design elements you would need for creating other graphics for the site, as well as for other projects the client may hire you to perform. For instance, your client may decide that he wants to place some banner ads on some other Web sites and decides to hire you to create those graphics. Because you already have a style guide embodied in your mock-up, generating these new banner ads should be a piece of cake.

Working from a Site Map

When you are staring at an empty canvas (or blank page or whatever you want to call the new file you are working with), the task of putting all the elements together can seem so overwhelming that you don't know where to

begin. To help you get started, go to the visual site map you create (as described in Book I, Chapter 3) and use it to guide you in creating your site mock-up. Remember, the site map is an agreed-upon, client-approved document about the site, and as you can see in Figure 2-1, it also contains at-a-glance information about all the navigation links, subnavigation elements, the page order, and all the other bits and pieces of content that need to be added to the mock-up. This content includes things like the logo and other branding or company details, footer links, a copyright notice, and certain site-wide dynamic functionality, such as a site-search feature or e-mail sign-up form.

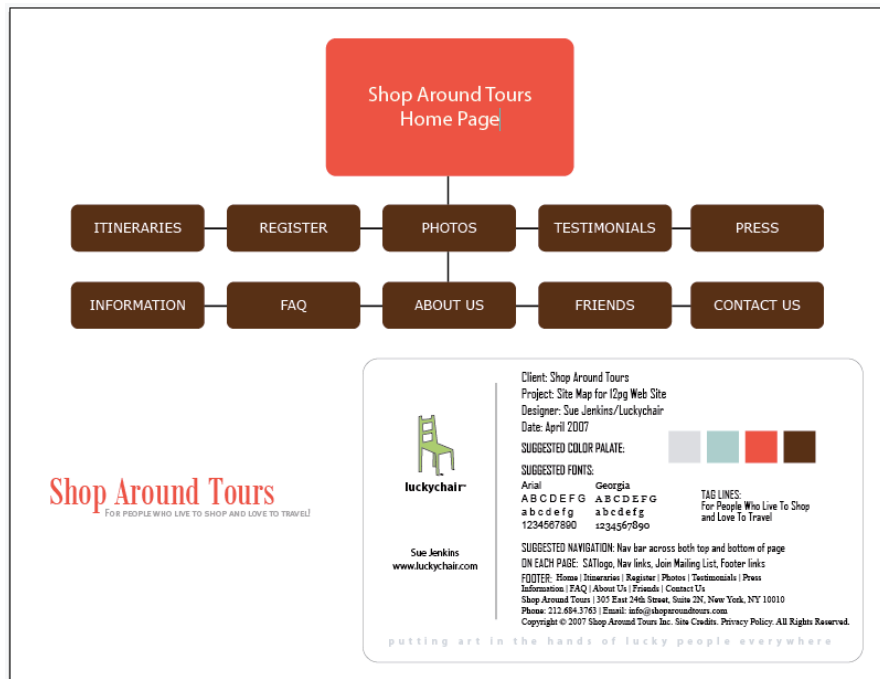


Figure 2-1: Use your site map as a starting point for designing your site mock-up.

As long as you start by putting all the necessary content onto the design canvas before you start thinking about the layout and design, you should have an easier time arranging all the elements when you start to move them into the position consistent with the design decisions you made about the page layout, size, orientation, color panel, and font selection. In the following sections, you find out how to translate the information in the site map into an organized graphic mock-up worthy of presentation to the client.

Creating the Mock-up

A thousand times better than any sketch, your graphic mock-up is the single best way to show the client what the finished Web site will look like. The *mock-up* is the graphical representation of the Web site layout used to communicate the look, feel, and functionality of a site before the graphics are optimized and the site gets constructed using HTML, CSS, and other development techniques.

Of course, if the idea appeals to you, feel free to sketch a mock-up by hand before generating the graphical version of it in your preferred graphics program. However, do not use a sketch in lieu of the graphical mock-up, because sharing a sketch with a client creates too many opportunities for miscommunication.

What makes the graphical mock-up so powerful is that it can clearly demonstrate a unified vision that pulls together all the site's elements, including precise color selection, accurate representations of fonts and font sizes, and precise placement of the navigation system and textual and graphic content on the site. Furthermore, this precision assists with discussions between designer and client about revisions to the design should any need to be made. With luck, your client will like what you have done and will only have a handful of minor suggestions to improve the layout.

In the sections that follow, you find some important suggestions about creating the mock-up, including how to block out different parts of the page, strategically place your most important design elements “above the fold,” and unify your design with graphic elements.

Blocking out the parts of the page

Because the most important page of any Web site is the home page, you should plan to mock up that page for the client (or for yourself, if you're designing your own site). If you and your client agree in advance that you, the designer, will mock up a different internal page (that is, any pages on the site other than the home page) instead of or in addition to the home page, doing so can be helpful if the budget and time frame allow this. Generally, though, the home page sets the standard for the design of the rest of the site, because it includes most if not all the repeating graphical elements that will appear in the same location on the other pages throughout the site.

As you discover in Book I, Chapter 4, you can use any of several graphic applications (Photoshop, Illustrator, Fireworks, and so on) to create the Web mock-up, depending on your specific needs, budget, and personal preferences. Begin the mock-up process by following these steps in your graphic application:

1. Add, in any order you like, all the site elements to a blank, appropriately sized document in your chosen graphics editor.

For example, if you have decided to work in Photoshop (like I do), you will have created a new, blank RGB, 8 bit, 72-ppi document in the agreed-upon size, such as 760 x 420 pixels.

2. After you have added all the elements to the page, begin to reposition each of them around the document window into a visually pleasing order consistent with the agreed-upon design directives.

Use rulers, grids, guides, and any other software-specific tools to assist you in organizing and aligning your content, and using consistent spacing between elements.

Figure 2-2 shows an example of how a budding mock-up might look.

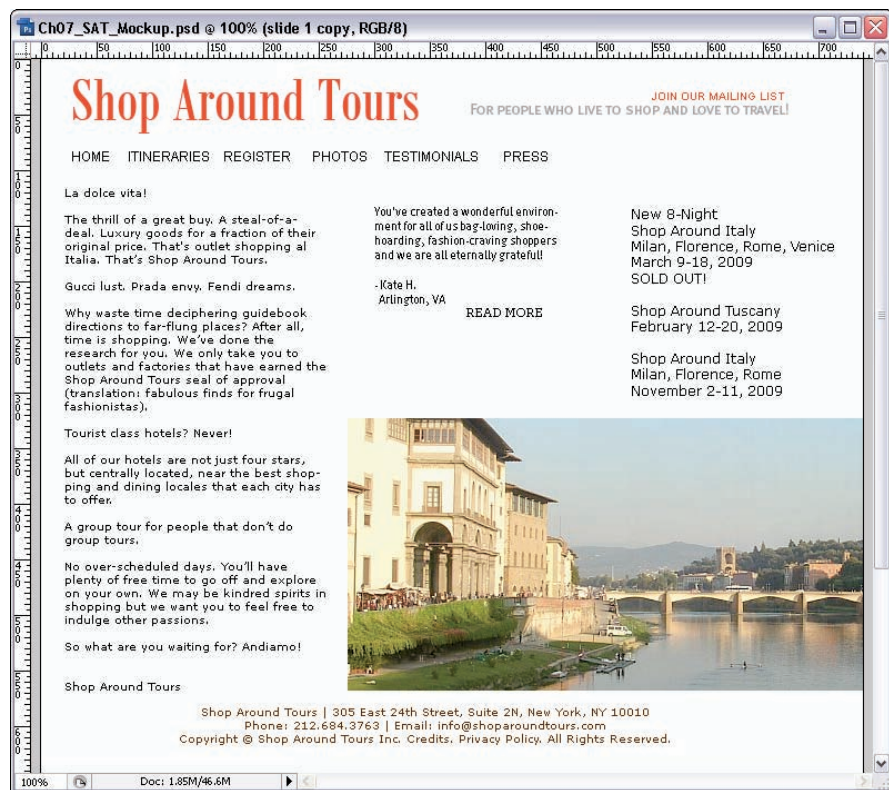


Figure 2-2: After adding the home page logo, text, and graphics, move each element to its approximate location.



To better envision how the final Web site will look, some designers set their monitor resolutions to 800 x 600 or 1024 x 768, take a screen shot of a maximized browser window, and then paste the screen shot into their mock-up file before returning their monitors to their preferred working resolution. Not only does this method show approximate available design space, but seeing the design as it might appear in a browser can assist in selling the layout to the client when it's time for review and approval.

Here are a few important points to keep in mind while blocking out parts of the page:

- ✓ **Navigation links:** As a general rule, navigation links are often placed either somewhere across the top of the layout directly below or to the right of the company logo, or along the left margin of the layout beneath the company identity. (Of course, you are welcome to break the rules and place your navigation elsewhere, but it's good for you to know the standards before you veer away from them!)

Each of the navigation text links should be clearly delineated by designing button graphics behind, around, or to the sides of the link text or by adding some kind of dividing symbol like a bullet (•) or vertical zero-width nonjoiner (!) between them. Footer links at the bottom of the layout, which often include both navigational links and other text like a copyright notice (as shown in Figure 2-3), are also often part of the mock-up.

Shop Around Tours | 305 East 24th Street, Suite 2N, New York, NY 10010
 Phone: 212.684.3763 | Email: info@shoparoundtours.com
 Copyright © Shop Around Tours Inc. Credits. Privacy Policy. All Rights Reserved.

Figure 2-3: Separate links with graphics or a dividing symbol.

- ✓ **Text:** For any text on your mock-up that will be rendered in HTML when you build the site, be sure to choose a cross-platform-compatible font and set the antialiasing to None so that the text in the mock-up looks like actual HTML text. Acceptable cross-platform fonts include Verdana, Arial, Arial Black, Helvetica, Courier, Courier New, Book Antiqua, Lucida, Palatino Linotype, Times, Times New Roman, Georgia, Geneva, Tahoma, Trebuchet, Comic Sans, Impact, Serif, and Sans-Serif. In addition, try to include examples in the main content text of any formatting styles that will be included on the site, such as headings, subheadings, bylines, paragraphs, and bulleted and/or numbered lists.

- ✓ **Color:** Apply color as needed to help define the different areas of the layout, such as background page color and/or design background color when the design is fixed in width. Color can be used as stripes, blocks, bars, lines, circles, squares, triangles, parallelograms, blobs, and other customized shapes. For instance, you might want to have a horizontal band of the primary color across the top of your layout on top of which the company logo will be placed; another band of the secondary color just below that for a row of navigation buttons; and an accent-colored box or series of boxes off to the left or right below that for subnavigation page links, ads, and other sidebar information.

Expect to put in at least 8–12 hours for the initial design if you’re an experienced designer — more if you’re new or relatively new to the Web design process.

Designing “above the fold”

A Web page has certain areas that visitors are more likely to drink in and linger upon, similar to the upper front page of a newspaper. The most valuable real estate on a Web page, therefore, is the area from the top edge of the browser window down to the bottom edge (in a maximized browser window), before a visitor needs to scroll to see any additional content.

The actual area of this part of a Web page, which is commonly referred to as the area *above the fold*, is determined by each visitor’s monitor resolution setting, which is going to vary from one monitor to the next.

To help you accurately determine the size of the area that is above the fold for your target audience, take a look at Table 2-1. In general, you need to take into account the resolution the target audience uses, and within that resolution, allow space for browser chrome — the stuff that takes up space inside the browser window, such as the scroll bars, navigation buttons, the status bar, and the address bar.

Table 2-1 Find the Design Area above the Fold	
<i>Target Audience’s Monitor Resolution</i>	<i>Approximate Design Space above the Fold</i>
800 x 600	760 x 420 pixels
1024 x 768	1000 x 600 pixels
1280 x 1024	1260 x 800 pixels



Because you truly have no control over what visitors' monitors are set to, the recommended standard is to put the most important information in your Web design in the top 420 pixels, with less critical information beneath that.

Your above-the-fold design area should include the following:

- ✓ Company branding (logo, company name, tagline, and so on)
- ✓ Navigation links or buttons
- ✓ Text and graphic information to tell the visitors about the products, services, and benefits the site has to offer

What should not be included above the fold? Typically, you should avoid including Flash intros and banners, advertisements, unnecessary text, and too many navigation links. Besides taking up valuable space, these unnecessary components can increase the size of the page, causing it to load more slowly in the browser window. If each page on a site should take fewer than 15 seconds to load, you really can't afford to waste any space putting unnecessary information above the fold that could be just as easily placed elsewhere on the page.

What's more, visitors should be able to scan the above-the-fold area within five to ten seconds to determine whether your site has what they are looking for. The home page needs to visually lure visitors into the rest of the site. At a quick glance, visitors should be able to do the following:

- ✓ Identify the site by its name or logo
- ✓ Get an overall flavor for the company by the site's layout and use of fonts, colors, and graphics
- ✓ Scan the navigation links for keywords that might help visitors find a particular page (such as products, services, or a contact)
- ✓ See quickly whether any specials or items of interest are on sale
- ✓ Scan or rapidly read introductory headlines or short comments about the company, key products, services, or news items
- ✓ Find quick-access links to customer service, shopping carts, help, coupons, site search, login, and other dynamic features



Including all of this important information above the fold encourages the visitor to stay on the site, keep looking around, and click through to the other pages on the site. That is the main goal of the home page!

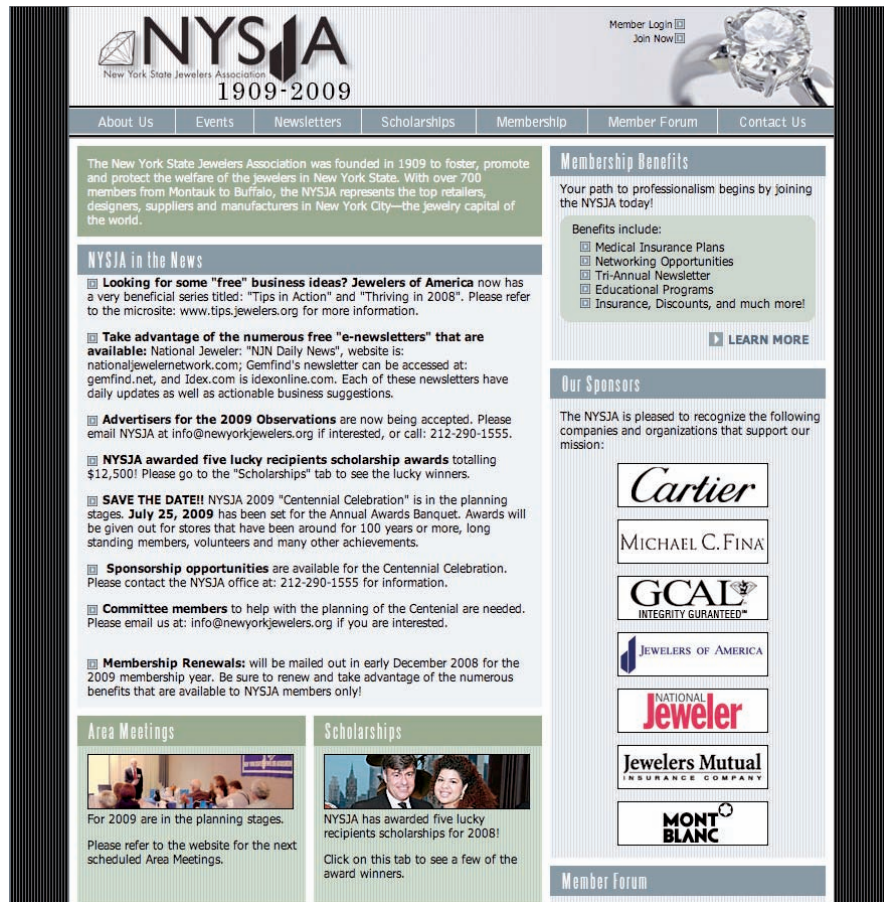
If visitors cannot find what they are looking for when they first enter the site, they are very likely to leave without further exploration.

If you don't believe this, think about your own browsing habits. How long does it take for you to decide whether a particular site has what you're looking for? Five seconds? Ten seconds? Thirty? The human brain processes so many calculations per second that, relatively speaking, ten seconds is a lot of time. Seasoned Internet users know exactly what they are looking for. They are an impatient bunch (when it comes to surfing the Internet) and will gladly keep searching elsewhere until they find what they're looking for.

To assist you in your design mock-up for the area above the fold, keep the following tips in mind:

- ✓ Add headlines, subheadings, bylines, and other visual design elements that clearly communicate the site's benefits to visitors. Think of a newspaper: The name of the newspaper is the first thing you see, then the main story headline and text. You also find information about news items in different sections of the paper, a few bylines, and introductory text to three or four stories.
- ✓ Other good elements for the area above the fold (besides your navigation and key content) are links that allow visitors to sign up for a newsletter, contact the site, search the site, access a shopping cart, log in to an account, and bookmark the page.
- ✓ As you're arranging all the elements on the page, refer often to the words selected to identify the ideal site visitor as well as the identity for the site. Also keep the overall purpose for the site in mind because that concept can sometimes provide cues to positioning things in the layout.

As long as you can accomplish these goals within the top 420 pixels or so, you can pretty much do whatever you like for the remaining areas of the layout. For example, after you set the tone of the mock-up, you can repeat design ideas like pinstriping, custom bullets, header treatments, and unique image cropping throughout the rest of the layout, as illustrated in Figure 2-4. As you can see, the sidebars each have a title and striped background, which echo other design elements within the layout.



Book II
Chapter 2

Mocking Up
the Design

Figure 2-4: Use repeating design elements throughout the mock-up to create a sense of visual unity.

Unifying the layout with design elements

The design process doesn't stop when all the elements are in the layout. Your next task is to start to add design elements to the mock-up that can really make the design unique. This includes adding things like horizontal rules, vertical divider lines, customized bullets, rounded or specially angled graphical corners, unusual textures, drop shadows, special effects, buttons,

borders, arrows, widgets, symbols, shapes, borders, and backgrounds to delineate the different areas of the layout, and other target-appropriate graphical embellishments.



The design process is a time of exploration and play. Take some design risks if they make sense in your layout. Repeating certain elements can provide instant unity to a design, but so can adding contrasting elements like using different-sized fonts and accent colors. You might even want to try making several versions of the mock-up to try out different text treatments and content-alignment options. If you get stuck for ideas during the design process, spend time looking at other sites for inspiration.

Above all, remember to be thorough. Each and every element in your Web layout needs to seem like it was meant to be in the position it is, in the color it is, in the size it is, and in the font it is, all relative to the other objects in the design. Make the layout interesting, original, and creative. Here are some ideas that can help you pull together the rest of the site's design:

- ✓ Organize the content so that the site will be easy to navigate.
- ✓ Have a single focal point for the visitor's eye to rest upon.
- ✓ Leave white space in the layout to balance out the areas with content.
- ✓ Use no more than three different font faces in the design.
- ✓ Use consistent spacing between like objects such as bullet points, navigation links, and table rows.
- ✓ Make the text easy to read by using a single column no wider than 600 pixels or by breaking the layout into two or four columns.
- ✓ Add headers, subheads, and pull quotes to break up large text areas.
- ✓ Use antialiased text for any text graphics and set antialiasing to None for text that will be displayed in HTML.



When HTML text is displayed on a Web page, the letter characters are rendered as hard-edged, low-resolution shapes with slightly jagged corners. By contrast, the letter shapes you see in projects created for print often have nice, smooth edges. That smoothness comes from a digital rendering technique called *antialiasing*, which softens any blockiness by adding semitransparent pixels to the letter edges so that the shapes have a smoother appearance. The smoothness is further enhanced by the resolution setting

of the graphic file. Whereas Web graphics should be set to 72 ppi (pixels per inch), print graphics should be at least 300 dpi (dots per inch).

The ppi is the consistent unit of measurement that refers to the number of pixels per inch that are viewable on a computer monitor. This should not be confused with dpi, or dots per inch, which refers to the measurements used to define the sharpness of a printed page.

In most graphic design programs, you should find a setting for text that allows you to control whether the text is rendered with or without antialiasing. Be sure to set the antialiasing to None for any text in your mock-up that you intend to be rendered in HTML in the completed Web page. For other copy in your layout, such as special instances of text that will be converted into Web graphics, apply one of the antialiasing options (Crisp, Strong, or Smooth) so that the text will retain a smooth edge.

To help you create a sense of visual unity within the design, consider doing any of the following to your mock-up:

- ✓ Use photographs and illustrations to add visual appeal.
- ✓ Add horizontal and vertical rules to create divisions between different areas on the layout.
- ✓ Use rounded edges, angles, and other shapes to break up the linear quality of the design.
- ✓ Show hypertext links with underlines in the same color they will appear on the finished site.
- ✓ When possible, use Web-safe colors for large, flat areas of color.



If you're ever feeling stuck, remember that a simple design is usually more compelling than one that is overcrowded with bells and whistles. Try the squint test. If the page seems organized and balanced when you squint at it, you've done a good job. If parts seem too jumbled or too heavy on one side and too light on another, keep playing with the pieces of the layout until you can find a nice balance. Remember, too, to keep spacing and alignment among the elements consistent or deliberate throughout the layout. Good design allows the eye to flow from one area of the page to another, with areas of white space surrounding the content so that the eye has somewhere to rest as it moves among the important items in the layout. Figure 2-5 shows an example of the completed Shop Around Tours mock-up that incorporates all these design principles.



Figure 2-5: Good design is often balanced and organized, like this layout for Shop Around Tours.

Finalizing the Mock-up

After you have completed the design to the best of your ability, put it away for a day or two and then return to it to be sure that it still looks fresh and finished. Not looking at it for a couple of days can provide you with a little

objectivity. If you find any inconsistencies in the design's spacing, alignment, coloration, or sizing, now is the time to make your corrections and adjustments. Use the following questions to help you review your design:

- ✓ Does the layout look unified?
- ✓ Is the navigation system easy to identify, and does it look clickable?
- ✓ Have you used too many fonts?
- ✓ Can any redundant elements be removed from the layout?
- ✓ Did you place the most important information above the fold?
- ✓ Did you remember to set the antialias setting of any HTML text in your mock-up to None?
- ✓ Does the site have more than one focal point?
- ✓ Are the visitor benefit messages clear and easy to identify?
- ✓ Does the eye have any white space to rest upon?
- ✓ Are any design elements repeated throughout the design to give it a look of cohesiveness and professionalism?

In addition to this checklist, the following sections help you finalize your mock-up and provide tips on how to best present your design to your client.

Showing the subnavigation

The navigation system in your mock-up will be consistently placed on all the pages throughout a site, allowing visitors to easily navigate to the different pages of a Web site. Navigation systems range from the simple to the complex and can be created using HTML, JavaScript, DHTML, Java, and/or other programming languages. The most popular systems are text based, list type, and JavaScript-enabled navigation. Some navigation systems, as I'm sure you're well aware, include subnavigation to help organize the destinations that visitors can select from. While you may know how you intend the navigation and subnavigation system to look and function, your client may not until the navigation has already been built, which would not be a smart thing to do.



One of the best secrets that can help you sell the finished mock-up to your clients is to add one more element to your mock-up. Namely, you should try to show at least one of the navigation buttons in the layout in its *rollover state* or *mouseover state* — that is, how the link will look when a visitor moves her mouse over one of the links on the main navigation, complete with a hand cursor pointing to the link and including the look and feel of any drop-down or tiered subnavigation menus, when applicable. The more realistic the

mock-up, the more easily you can help your client visualize the final product. Rollover graphics generally appear on the fly due to CSS or JavaScript in the page code that is activated when the user moves the cursor over the link. When the visitor moves the cursor away from the link, the link (and any sub-menus that may have appeared) typically returns to its normal state immediately or after a predetermined delay measured in milliseconds.

By adding this one simple graphical element to the layout, you can help your client get a feel for how the site's navigation will function when the site gets built in HTML. Figure 2-6 shows an example of a mock-up that includes how the subnavigation menu will look when a visitor mouses over a navigation link.

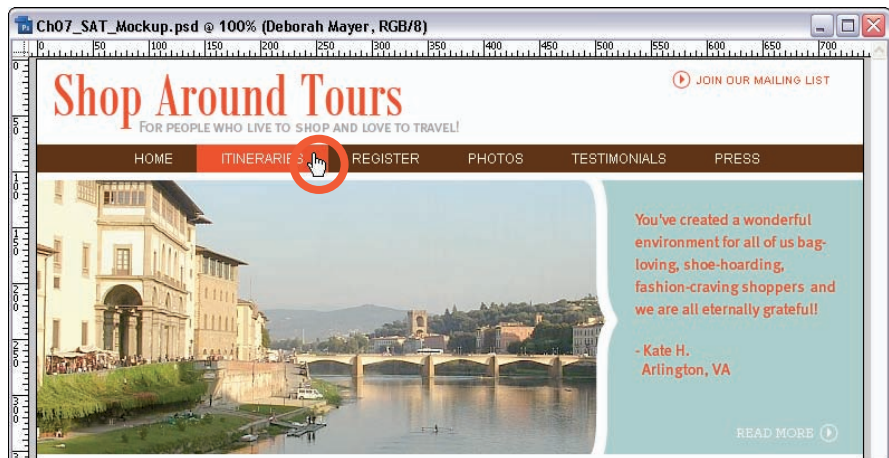


Figure 2-6: Include an example of rollover or subnavigation functionality on the Web site mock-up.



Your Web clients have hired *you* for their Web project because

- ✓ You're the expert, the person who knows how to visualize and create a mock-up before it becomes a working Web site.
- ✓ They have no idea how to mentally conjure up this kind of design themselves.

By showing your clients what they can expect to see in advance of the site-building phase, they will have very little room for confusion about what they will see when the site actually gets built.



Presenting the mock-up to the client

Though it's not a practice I use, some designers intentionally add some kind of noticeable error to the layout — a red herring — before presenting the mock-up to the client. The idea behind this technique is that by including something obviously wrong to the layout — like using the wrong font or color — the client will almost assuredly point out that error during the mock-up review (“Why is *that* blue?”), recommend a change of some kind (“Make it red.”), and come away from the process feeling valued for making a contribution to the design process. This sounds good in theory, but I've witnessed some clients say they actually liked the “bad” element, and removing something can be tough after the client has seen it. It's kind of like a Murphy's Law of Design: When given three options, the client will almost always choose the worst-looking design!

If you are truly satisfied and proud of your design after you've added a sample of the rollover state to the mock-up, you are finally ready to present the finished mock-up to the client.

Web mock-up presentations can take many forms, including the following:

- ✓ E-mailing a JPG, GIF, PNG, TIFF, or PDF file to the client for review
- ✓ Uploading a JPG, GIF, PNG, or PDF file of the mock-up to a test server and sending the client the URL to view it
- ✓ Uploading a JPG, GIF, PNG, or PDF file of the mock-up to a free drop site (such as <http://drop.io>) and sending the client a link and password to retrieve the file
- ✓ Meeting with the client in person to present the mock-up on a laptop, projector, or computer screen in its native graphics program format, or as a JPG, GIF, PNG, or PDF file
- ✓ Meeting with the client in person to show the mock-up as a printout (or series of printouts) that's been mounted to an archival presentation board

Unless the client wants to confer with other members of her team, or with friends and family before giving you her opinion, she should be able to review the mock-up in a reasonable amount of time, say one to five business days. After the review, she'll either provide you with feedback on what revisions she'd like you to make or grant you written approval to begin building her new site in HTML.

Getting written client approval on the design

If I could share only one tip with you about working with Web clients, it would be “Get everything in writing.” When communication is clear and both you and the client agree in writing about each of the steps of the design process, the project magically stays on track and within budget.

Start each project with a contract (and get a deposit), and tell the client in advance that you will be obtaining her signature of approval at key points during the development process. Get written approval on the site map, definitely get written approval on the completed mock-up before you begin building the site in HTML, and be sure to also get a final signature at the end of the project when the site is finished before sending the client your final invoice. These signatures are your insurance should the client change her mind about the site requirements while the project is under way. They also show the client that you’re committed to meeting your responsibilities as outlined and defined in the contract.

In my experience, most Web clients are great to work with and treat the designer with respect.

Nonetheless, clients occasionally push for more and don’t want to pay for any additional services. Do *not* be tempted to do any extra work for free, just to be nice, even when the client begs you to do something in a kind and gentle way. Your time is your livelihood, and you deserve to earn money for every minute you spend working on the project. Should additional changes be required beyond the scope of the project, the signatures allow you to feel confident about requesting more time, more money, and more materials to get the job done.

During your mock-up presentation, communicate to the client why the layout looks the way it does. Sell your ideas. Explain your reasons for selecting particular fonts, colors, and alignments. Point out all the key elements of the design, being sure to mention how it captures the identity of the site as well as caters directly to the target audience. Identify the focal point of the design and any other key elements you’re particularly proud of. The more you show that you have thoroughly thought about each of the design elements, the more your client will understand your visual point of view. Good luck!

Be prepared to make some changes. Most clients will have at least one thing to say about the mock-up that requires revision. If you disagree with anything a client wants to have changed, respectfully give reasons for why you believe the mock-up works the way you’ve designed it. If the client still disagrees and insists you make a particular change, say you’ll try her suggestion (even if you’re sure it won’t work well or look right). With bad ideas, many clients can actually see how your version looks better than their suggestions when the two versions are compared side by side.



Above all, however, remember that the client is paying you for the project, and she needs to be happy with the results. Be willing to compromise, be open to making changes, and keep working to the maximum number of allowable revisions in your contract until the design is approved. If additional changes are required beyond the scope of the project, remind the client of your fee for these extra services before you provide them to ensure that both you and the client are in agreement. Many clients often reign in their desire for multiple revisions in an effort to stay within budget and keep the project moving forward toward completion.

Creating Additional Web Graphics

Web sites often include additional page-specific graphic elements that must be designed to be consistent with your mock-up. After the client approves the Web mock-up and provides you with written acceptance of the design, you may begin creating these other graphics before you start to optimize the rest of the site's graphics and move into the site-building phase of the project.

These additional graphics, for lack of a better term, include things like the over states of rollover buttons, icons, bullet graphics for customized lists (with CSS, you can use your own graphics for bullets!), background images, navigation elements, curved corner graphics to make the hard corners on tables or layers look rounded, graphical horizontal and vertical rules or dividers, animated GIFs, antialiased page headers using fancy fonts, illustrations, and photographs.

For some of the additional graphics, like rollover graphics, you can create new layers in the existing mock-up file. For the rest of them, feel free to create as many new RGB, 8-bit, 72-ppi appropriately sized documents as you need. For example, you may create one file for rollover buttons, another file for header graphics, one for background images, and another for sale items.

In the following sections, you find out about creating other graphics for your site, including header graphics, rollover graphics, background images, and others, such as diagrams, photos, and illustrations.

Header graphics

Also called page headers, headings, or A-heads, header graphics refer to graphical renderings of page titles using specialized fonts that are placed at the top of the page content in lieu of creating headings using cross-platform-compatible fonts styled with CSS.

If your layout calls for specialty font headers instead of headers styled with CSS, create and save these graphics in a file separate from the mock-up. This way, if any new pages are added to the site during the page-building process or at sometime down the road after the site has been launched, you can easily go back into this header graphics file and quickly create the new headers that you need. When working in Photoshop, you can easily add new layers for each new bit of text, as illustrated in Figure 2-7, and then hide and show the layers as needed to generate the individual optimized graphics.

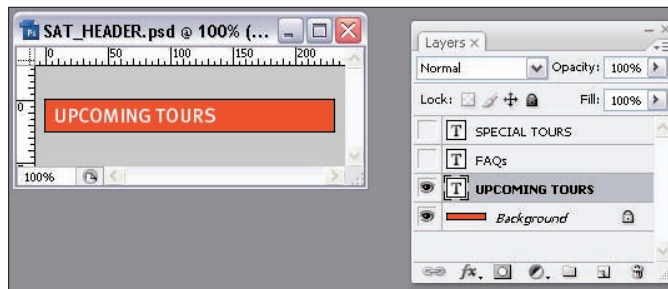


Figure 2-7: Use a single file repeatedly to create multiple versions of a graphic.

Rollover graphics

Rollover graphics change from one look to another when a visitor interacts with the graphic in some predetermined way, such as mousing over the normal graphic. The most common form of rollover graphic is the rollover button used in navigation or for some other function on a site, such as a submit button on a form. Rollovers can be any color or combination of colors; can be styled with special effects like bevels, drop-shadows, and stroked edges; and can include any content such as text, illustrations, and photos. In fact, as long as you make sure that both the normal state and rollover state graphics are exactly the same size, they can look like whatever you want them to.

To create rollover graphics, feel free to design them with color, text, and special effects using either a vector or raster graphics program, whichever suits your needs at the time. When your graphics are finished, you will optimize the two rollover state graphics as GIF, JPEG, or PNG files. Book II, Chapter 3 teaches you the ins and outs of Web graphic optimization. After you have optimized your graphics, the next step is to make the rollovers functional. To do that, you need to use a little bit of JavaScript with your HTML, which you find out how to do in Book III, Chapter 6.

Background images

Cascading Style Sheets (CSS) give you the amazing ability to use background images on your page, as well as inside any HTML container, such as a table, table cell, or layer using the `<div>` tag. Background images can be set with CSS to repeat along the X axis, the Y axis, or both the X and Y axes, or to appear once and not repeat. You can also position a background image precisely by using X/Y coordinates relative to the upper-left corner of a browser window or relative to an HTML container tag such as a table cell or layer. This means that background images no longer need be miles tall or wide to tile seamlessly horizontally or vertically within your layout. Wahoo!

Other graphics

Before you head off into the land of graphics optimization, look again at your site map and continue creating all the graphics you think you will need for the various pages of your site. Most of the graphics you'll need to make will be obvious when referring to the content the client intends to use on the individual pages of the site, such as page headers, diagrams, and board member portraits. Then, later on, while you're building the site, if you discover that you've forgotten to create a graphic or determine the need for a new one, you can come back to your graphics program and create one.

Chapter 3: Slicing and Optimizing Web Graphics

In This Chapter

- ✓ **Selecting the appropriate Web graphic format**
- ✓ **Choosing the right optimization program or tool**
- ✓ **Slicing and optimizing techniques**
- ✓ **Choosing the proper optimization settings**
- ✓ **Outputting optimized files and graphics**

Before you begin the optimization process of your Web graphics, you should have a clear understanding of how the graphics and photos you use on your Web site need to be different from the ones you might use in a print project, such as a brochure or annual report. For one, print graphics are high resolution, whereas Web graphics must be set to a low resolution. For another, print graphics depend on the CMYK color mode, whereas any graphics used for on-screen and Web presentation must use RGB color.

In this chapter, you find out about all the differences between Web and print, as well as everything you need to know to create graphics that are ready for Web optimization. Without this preparation — and especially without optimization — the graphics and photos on your Web sites would simply be too large to transmit over the Web and display in a browser on a visitor's computer monitor in an acceptable time frame.

In the following sections, you find an overview of Web graphics as compared to print graphics. You also discover tips about choosing an optimization program. Following that, you find out about the different Web file formats, including how to select the right format for different graphic types. The last three sections of the chapter include the finer points about selecting different optimization settings, slicing up images before optimization, and finally, choosing the right optimization output options to produce the desired output results.



Web Graphics 101

When people talk about Web graphics versus print graphics, what they're really referring to are the different ways in which Web graphics must be formatted compared to graphics intended for print.

Graphics, whether for print or the Web, can be created with a variety of software programs, the most popular among them being Illustrator, Photoshop, Fireworks, and Flash for the Web and Illustrator, Photoshop, QuarkXPress, and InDesign for print. The finished graphics may be then be saved in a variety of file formats, depending on their intended usage.

Of all the applications you can choose from to create your graphic images, one primary consideration is whether the artwork needs to be developed and saved as either vector or raster:

- ✓ **Vector:** A vector program uses mathematical equations to generate paths, lines, and shapes, which enables the image to be scaled up or down with no loss of resolution. Logos, for example, are best created in a vector program. When created as vector graphics, the logo artwork can be colored and scaled for any medium — online, newsprint, embroidered on a hat, printed on the side of a pen, and so on — and still look great at any size.
- ✓ **Raster:** Raster (or bitmap) programs represent images as a collection of tiny pixels or little squares of color, the size and quantity of which are determined by the file's resolution. The number of pixels in an image determines the image's quality; typically, the higher the number, the sharper the image, and the lower the number, the fuzzier the image. An image set to 300 dpi (dots per inch), for instance, has a high resolution and therefore is fine and clear enough to use in a printed piece, whereas an image set to 72 ppi (pixels per inch) has too few bits of information to print crisply, even though the image might look fine and clear on a computer monitor.

Table 3-1 shows a comparison of Web and print graphics based on a variety of key criteria. As you will see, some features of Web and print graphics are so different from one another that you must take extra care that you don't mistakenly set up your files incorrectly. For instance, your Web and print graphics use different color spaces because print requires CMYK inks and the Web relies on RGB technology to display color. In the section that follows, you discover details about each of the features listed for Web graphics in Table 3-1, along with a few others that can assist you in creating your graphics appropriately for the Web.

Table 3-1 Web and Print Graphic Comparison		
	Web Graphics	Print Graphics
Color mode	RGB	CMYK
Resolution	72 ppi	300 dpi
Unit of measure	Pixels	Inches, points, picas
File size	Smaller file sizes make images display faster on the Web.	Larger file sizes may produce sharper images in print.
Page size	Images can be placed on an adjustable-size Web page.	Images are placed on layouts with a fixed page size.
File format	GIF, JPG, or PNG	TIFF, EPS, PSD, PDF, BMP, AI, INDD, QXP, PPT, DOC

Color mode

Graphic artists use two color modes, or gamuts, to create and save their work for the Web and for print design: RGB (Red, Green, and Blue, which are additive colors) and CMYK (Cyan, Magenta, Yellow, and Black, which are subtractive). Typically you should select the appropriate color mode for your graphic file as you create the new document.

If you forget to choose the correct color mode for your graphic files and you create some or most of the layout or image adjustments before realizing the mistake, be forewarned that converting a file from one color mode to another can cause noticeable shifts in color in the image that might render the image color inaccurate and possibly unusable. This is especially true for images created in RGB but intended for CMYK print.

Figure 3-1 shows how the different color modes achieve their color and what happens when all colors in that mode are combined.



RGB colors are the additive colors of the visible spectrum, which means that when combined, the resulting color is white light. RGB is used primarily for

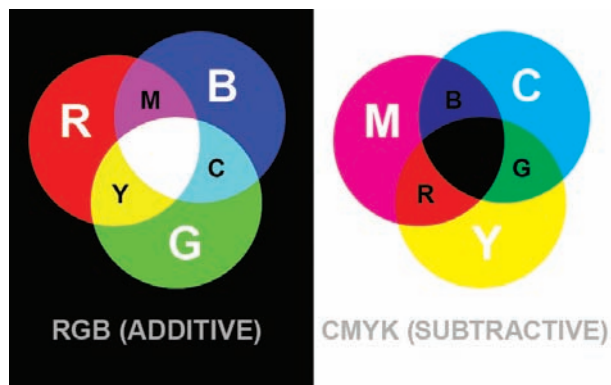


Figure 3-1: RGB should be used for the Web and any other on-screen presentations, whereas CMYK is used primarily for print.

on-screen (computer monitor) presentations, such as Web pages and PowerPoint slide shows, because computer monitors use RGB technology to display color.

CMYK colors are the subtractive print colors used mainly for four-color reproductions. When cyan, magenta, and yellow inks are combined and printed on paper, the resulting color is technically black, which represents the absence of light. I say “technically” here because in reality, combining these three inks creates a sort of muddy gray-black that isn’t quite as rich and saturated a “color” as you’re used to seeing when you think of black. This is where the K (for black) in CMYK comes in. To get a true rich black in any print job, the printer must add a separate black ink to the printing process.

Color gamut warnings

Compared to the visible spectrum and the 16.7 million colors you can see on a 24-bit computer monitor, the CMYK color mode is somewhat limited. The current U.S. standard CMYK technology, or SWOP (standard Web offset press), simply cannot reproduce with inks on a printed page the same full range of color you can see on a monitor. Any color that can’t be reproduced in print, therefore, is referred to as being *out of gamut*. Gamut refers to the range of reproducible colors on any given device, such as a printing press or computer monitor.

In addition to the bucket of colors that are out of gamut for print, another thing you might want to watch out for when creating graphics for the Web are the colors that don’t fall within the Web-safe palette. The Web-safe palette, as I describe in Book I, Chapter 4, refers to the 216 nondithering (solid) colors that can be accurately displayed in browsers on both Mac and PC computers with 8-bit monitors set to display a maximum of 256 colors. (The Web-safe palette has only 216 colors instead of 256 because 40 of the 256 colors appear differently on a Mac than they do on a PC.) Following this same gamut logic then, any color that can’t be represented on-screen on an 8-bit monitor is considered non-Web-safe and would be called out of gamut for the Web.



As I also describe, using the Web-safe palette is no longer as critical an issue in Web design as it once was in the early days of the Internet because most newer computers have monitors capable of rendering millions of colors. Though it might still be good to use a color from this palette when coloring large, flat areas of a Web page, such as a page or table cell background or when specifying the color of styled text with CSS, it’s no longer a general Web recommendation.

So how do you know whether a color you have selected for a Web or print project is out of gamut? Thankfully, both Photoshop and Illustrator (the two

programs used by most professional Web and print designers) have a feature within the Color Picker dialog box that alerts designers when a selected color is either out of gamut for print or non-Web-safe. To demonstrate how this works, follow these steps in Photoshop:

1. Launch Photoshop.

You can access the Color Picker tool without opening a new document, but feel free to open a new document if desired.

2. Click the Foreground Color box at the bottom of the Photoshop Tools panel.

This opens the Color Picker dialog box, which has several options for viewing color.

3. In the R, G, and B text fields, enter the color values of 0, 200, and 200, respectively.

After entering these numbers, the hollow circle icon within the large foreground color field moves to near the upper-right corner of the square, thereby selecting a turquoise color.

To the left of the OK and Cancel buttons, you can see a rectangle with the new color on top and the previously selected color on bottom.

Directly to the right of those color swatches, you see two sets of small warning icons, as shown in Figure 3-2. The top two icons indicate that the selected color is out of gamut for print, and the bottom two icons indicate that the selected color is not Web safe.

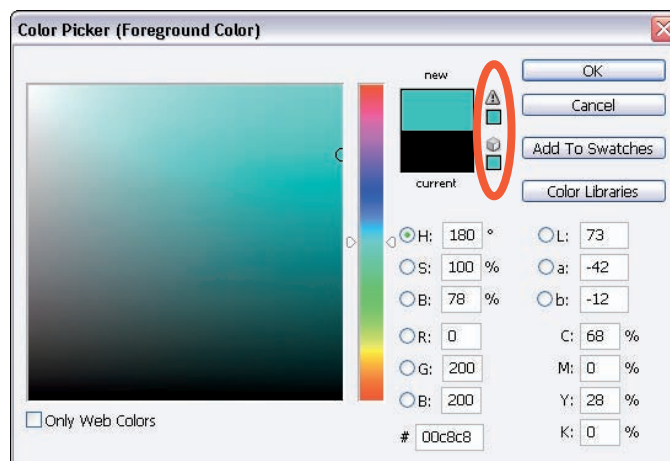


Figure 3-2: The Photoshop Color Picker dialog box shows warnings for any selected color that is either out of gamut for print or non-Web safe for a browser.

4. Click the top triangular out-of-gamut warning icon to have Photoshop locate the nearest in-gamut print color that you could use instead.

The new suggested color should have the RGB values of 50/191/194, which is now in gamut, but it isn't Web safe.

5. Click the bottom cubelike, non-Web safe color warning icon.

You now have a color that is Web safe (RGB values of 51/204/204) but is out of gamut for print!

Though it may be tough at times to find a particular color that is both Web-safe and printable, you have two ways to find such a color. First, you could click several times inside the large square color field until you find a color that is both in gamut and Web safe. The alternative is to select the Only Web Colors check box at the bottom of the dialog box. When this option is enabled, the nearest Web safe color is automatically selected. Then try clicking one of the other Web safe colors in the same color range and toggling on and off the Only Web Colors option until you find a value that is both Web safe and in gamut, such as RGB 102/204/204.



If you train yourself to get in the habit of paying attention to the gamut warning icons as you're selecting your colors for your print and Web projects, you can eliminate possible headaches that might occur if you show your client the mock-up with irreproducible colors. From now on, when you see an icon in the warning area, click it to ensure that the selected color meets your needs for print and the Web.

Resolution

Resolution refers to the number of *pixels per inch (ppi)* on-screen that are used to display an image. Though computer monitors can display different ppi settings, such as 640 x 480, 800 x 600, 1280 x 1024, and 1600 x 1200, Web browsers can display images only at a maximum of 72 ppi.

By contrast, your print graphics use *dots per inch (dpi)* to determine the quality of the printed output. To print a graphic at a high quality, you must set the resolution of your file to at least 300 dpi (though depending on the audience, in some cases, 150 dpi might be enough). Figure 3-3 illustrates the differences between dpi for print and ppi for the Web.



In print, the more dots there are, the larger and clearer the image. On the Web, however, increasing the number of pixels per inch is unnecessary because browsers and most regular computer and video monitors can't display images at resolutions higher than 72 ppi. Furthermore, a resolution higher than 72 ppi not only increases the file size but may also increase the dimensions of the graphic, making the image more time consuming to

download and potentially bigger than intended. That said, if you're going to be creating graphics for other output, such as projected PowerPoint presentations or images to be displayed on high-def monitors and TV screens, you could create graphics and 92 ppi and 96 ppi, respectively.

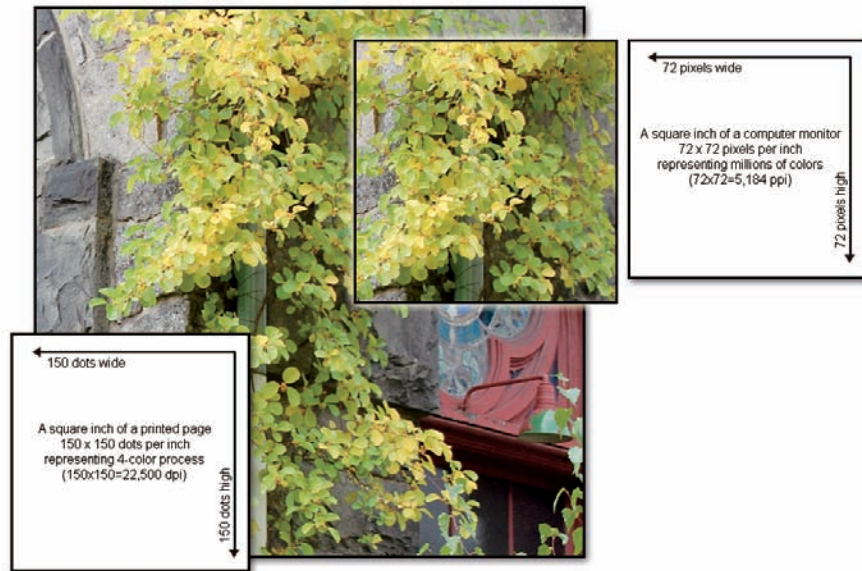


Figure 3-3: Print graphics are displayed in dots per inch, whereas Web graphics are displayed in pixels per inch.



This brings up a conundrum if you plan to create graphics that can do double duty for Web and print, as ideally you'll want your Web graphics to look great in a browser while also printing crisply for the visitor. Here are two suggestions you can use alone or in tandem for creating printworthy Web graphics:

- ✓ **Set the resolution to 96 ppi.** When creating new graphics for the Web, set the resolution of the files to 96 ppi instead of 72 ppi. This tiny bump in resolution might make the Web graphic look a tad crisper when printed without drastically increasing the size of the graphic. Remember that the higher the resolution, the larger the file size, and the larger the file size, the slower the image transmits and displays on the Web. The bump in resolution may also affect the file's dimensions, so be sure to keep an eye on that, too.

- ✓ **Add a print media CSS.** This technique presumes that you understand CSS and can configure it to correctly format and display the content on your Web site. After the master CSS file has been created, you could generate a second print media CSS that could replace the source code of the Web graphic files (`images/sample.jpg`) with print graphic files (`images/print/sample.jpg`) when the visitor prints a Web page.

Though this book doesn't get into that level of detail with CSS, I highly recommend you begin your journey to finding out more about advanced CSS techniques by reading Eric Meyer's article on CSS Design, "Going to Print," at www.alistapart.com/stories/goingtoprint. You might also enjoy reading *HTML, XHTML, and CSS All-in-One Desk Reference For Dummies*, by Andy Harris and Chris McCulloh (Wiley).

Unit of measure

Print graphics can be measured in any unit your graphics program supports, such as inches, points, pixels, picas, millimeters, ems, exs, and percentages, though the most popular units for graphic designers tend to be inches, points, or picas. The unit you choose to work in determines the size of the image when printed. By contrast, almost everything on the Web is measured in pixels, ems, or percentages. For instance, instead of saying, "Move that logo about a quarter inch to the right," you'd probably say something like, "Move the logo 37 pixels to the right."

To change the unit of measure in your graphics program, check the preferences to adjust the program's default unit of measure or adjust the measure unit through the Ruler's context menu.

File and page size

The size of a graphic file largely depends on the dpi or ppi settings and the dimensions of the image in inches or pixels. For example, a file set to 500 x 400 pixels at 72 ppi might only be around 6 x 5 inches in print size with a 585.9K file size, but an image set to 7 x 5 inches at 300 dpi might have a pixel dimension of 2100 x 1500 and a 12MB file size. On the Web, file sizes should be as small as possible while retaining the best-quality image, whereas in print, file size is somewhat irrelevant as long as the printed output is sharp and clear.

If you're used to working in print, the main thing you need to do differently for your Web graphics is to pay close attention to the document settings each time you create a new file. If you're new to both Web and print, as long as you select the appropriate color mode, resolution, file dimensions, and unit of measure for each new document you create before you begin designing your graphics, you'll do just fine.

Optimizing and Slicing Graphics

Up until a few years ago, the software program you used to create your Web graphics, such as Photoshop or Illustrator, was different from the program you used to optimize your Web graphics, such as Fireworks, which was specifically designed for the creation and optimization of Web graphics, or ImageReady, which came bundled with Photoshop and the Adobe CS Suite. Today that is no longer always the case, because Adobe has incorporated ImageReady's optimization engine into Photoshop and Illustrator. Of course, if you still have an old copy of ImageReady, feel free to use it. However, if you're new to the optimization process, stick with the optimization tools that I recommend in this and the following sections.

The following sections explain what optimization is, help you choose the right optimization program for your needs, and offer tips for optimizing and slicing graphics.

Understanding optimization

Optimization is a process whereby the software program applies an adjustable compression method to the digital information in an image to assist you in producing an output graphic in the smallest possible file size with the best possible quality. In other words, to be suitable for the Web, a graphic must be put through a compression process that takes the original image data and condenses it in such a way that the file size gets reduced to a point where the image quality is still acceptable. The smaller the file size, the faster the image can be transmitted over the Internet and displayed on a Web page. Even when the source file is already set to 72 ppi, the graphic must still be optimized and saved in an acceptable Web format.

In the simplest terms, optimization means reducing file size while trying to retain quality. During the optimization process, you can control how much compression to apply to an image. Remember, the ppi of the file is a requirement of Web graphics but a separate issue from the optimization process. Thus, when you create a new file at 72 ppi, that doesn't mean the file is already optimized. Rather, the file is at the correct size and resolution for optimization. If you need, therefore, to create a Web graphic from a high-resolution image, you must first reduce the resolution of the file to 72 ppi and then check the pixel dimensions to ensure that the size of the graphic is suitable before you optimize the graphic.

Choosing an optimization program

The following overview of the optimization tools and programs available can help you decide which one will work best for your needs:

- ✓ **Photoshop's & Illustrator's Save for Web & Devices:** Accessible from the File menu, the Save for Web & Devices optimization command, which has been incorporated into Photoshop and Illustrator since Adobe CS3, launches a dialog box, shown in Figure 3-4, that is the equivalent of the ImageReady image optimization dialog box. It has a 4-Up panel for comparing optimization settings before selecting one, a toolbox, a preview pop-up menu, and a Preview in Browser button, and it allows you to select individual slices to apply different optimization settings. It also offers several graphic output options, including saving optimized graphics along with a tables-based or CSS and layers-based HTML page.
- ✓ **ImageReady:** Available as a stand-alone software package or as a freebie bundled with the sale of Photoshop or the CS Suite up until CS2, ImageReady has an interface nearly identical to Photoshop, allows the creation of simple rollover graphics and GIF animations, has a special 4-Up panel for comparing optimization settings before selecting one, and offers several graphic output options, including saving optimized graphics along with a tables-based HTML page that includes prewritten JavaScript code for any rollover graphics you may have created.
- ✓ **Fireworks:** With Fireworks, you can design, edit, and optimize graphics all in one application by using a variety of tools similar to those found in Photoshop and ImageReady. Fireworks graphics, which are created and saved in the PNG format, can be sliced, optimized, and exported. Fireworks even has a preview tool that shows any cross-platform differences in color display. Fireworks users can create graphics for JavaScript rollover buttons, pop-up menus, and other interactive features that Fireworks can generate with the optimized files. Output includes HTML, images only, or both. Even better, Fireworks integrates seamlessly with Adobe Dreamweaver for easy round-trip editing.
- ✓ **Other optimization tools:** Several other less expensive and freeware Web optimization tools are available, so feel free to use whichever ones you think will produce the best results given your budget. For example, you might enjoy using the free GNU Image Manipulation Program, which also doubles as a full graphics program (www.gimp.org) or the JPEG Wizard (www.jpegwizard.com) and GIFWorks (www.gifworks.com) programs. Other programs and plug-ins include BoxTop's ProJPEG and PhotoGIF (www.boxtopsoft.com) and Equilibrium's Debabelizer (www.equilibrium.com).

Optimizing using Save for Web & Devices

Illustrator and Photoshop users can quickly and easily optimize graphics for the Web using the Save for Web & Devices dialog box, accessible through the File menu.

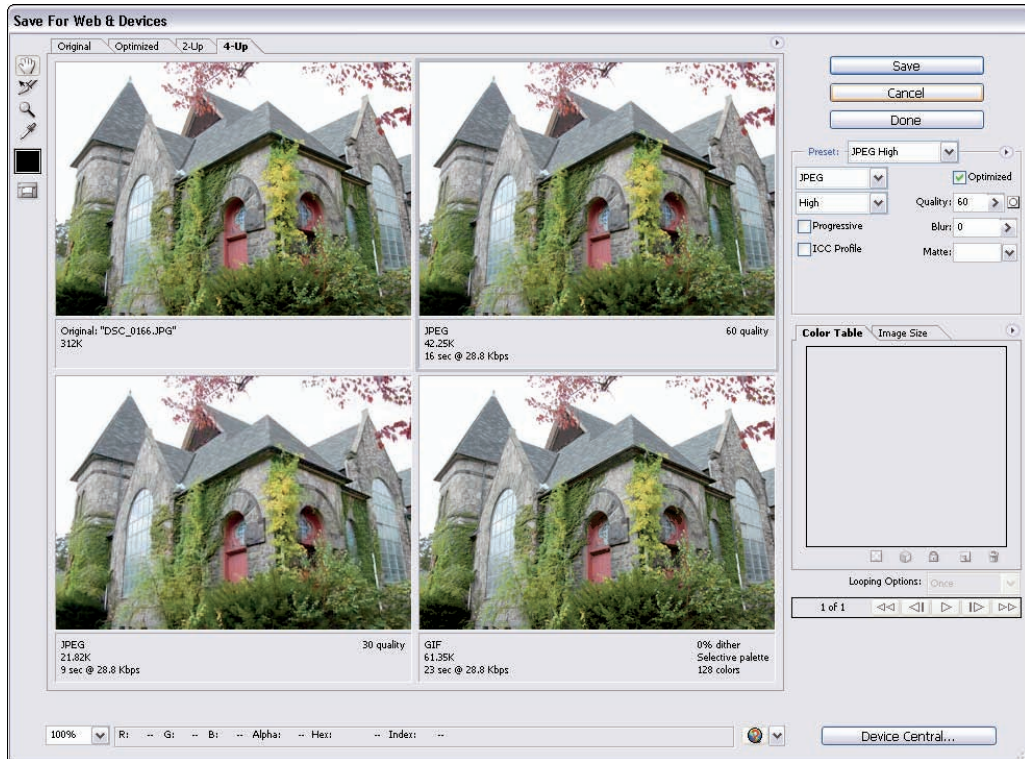


Figure 3-4: Optimize Web graphics directly within Photoshop and Illustrator in the Save for Web & Devices dialog box.

In the following set of instructions, you see how easy it is to use this tool, which works almost identically to the old ImageReady compression engine, to optimize and save your Web graphics. Follow these steps to optimize an image:

1. **Create your Web graphics in Illustrator or Photoshop, taking care to set the document resolution to 72 ppi, the color mode to RGB 8-bit, and (when applicable) the background to transparent or white.**
2. **To optimize and save a file for the Web, choose File→Save for Web & Devices.**

This opens the Save for Web & Devices dialog box (refer to Figure 3-4).

3. **Select either the 2-Up or 4-Up tab at the top of the dialog box to view your original image side by side with one or three versions of the image with different optimization settings.**

4. Set your optimization preferences for each comparison version of your graphic on the right side of the panel.

You find detailed descriptions of all the optimization settings, including output options, in the sections “Choosing Web Optimization Settings” and “Optimization Output Options,” later in this chapter.

5. Click the Save button to save the graphic to the specified location on your computer.

The dialog box automatically closes as the graphic is saved.

You can also use the Save for Web & Devices dialog box to generate graphics for rollover buttons and GIF animation if you are familiar with configuring those features in Photoshop. Alternately, you can also do these things quite easily in Fireworks.

If you want to improve your Photoshop skills, check out the “Designer’s Guide to Photoshop DVD” by Sue Jenkins. To find out more about working with Fireworks, see *Adobe Fireworks CS4 How-Tos: 100 Essential Techniques* by Jim Babbage (published by Adobe Press).

As you discover in the next few sections, most optimization engines have settings that allow users to select the output file format and control the quality of the compression. But first, a word about slicing up graphics.

Slicing up graphics

When I teach Photoshop and Dreamweaver classes, one of the most frequently asked questions I get from students is, “How do I take my Web site mock-up and turn it into optimized graphics?” And it’s a great question.

For most small graphics, such as buttons, background images, or a single photograph, the optimization process is fairly straightforward. However, if you want to optimize several graphics within a single file at once, you will need to slice your graphics before you optimize.

Slicing and *optimizing* describe the process of dividing a large image into individual pieces, or *graphics*, that are then compressed into GIF, JPEG, or PNG graphics and reassembled on the Web page with HTML, much like the pieces of a puzzle. Of course, the fewer graphics you have, the faster the page will load in a browser. The second best thing if you do have a lot of graphics is to carve them up into smaller pieces so that each piece can load more quickly than one or two larger ones. For example, if you had mocked up a navigation bar complete with rollover buttons, you could slice up the file in such a way that each of the individual buttons would become their own, separate graphics when the file is optimized.

Previewing your files in a browser

After you have optimized and saved a graphic for the Web, you don't need to wait until it's on a Web page to see what it will look like. To preview any of your optimized graphics in a browser window before you add them to a Web page or upload them to a remote server, simply drag and drop the Web graphic from the

location where you saved the optimized graphic (such as your desktop or inside a client folder) to any open browser window, regardless of whether you have a live Internet connection. The image will appear at its full size in the browser, aligned top and left on a white background.

When you digitally slice up your graphics, whether you're creating rollover buttons, banners, bullets, or other decorative elements, you must divide the larger rectangular whole into smaller rectangular pieces that can be easily reassembled on a Web page, usually inside some kind of container tag like a table cell (<td>) or a layer (<div>). Images are often sliced to help create rollover buttons and decorative graphics.

Photoshop, Illustrator, Fireworks, and ImageReady all have a similar Slice tool that you can use to slice up your larger mock-ups and graphic files into smaller pieces. The Slice tool's icon looks like sort of like an Exacto blade, making it easy to identify from the other tools on the Tools panel. In Photoshop CS4, the slice tool is located on the Crop tool's flyout menu.

To slice your graphics using the Slice tool in Photoshop, Illustrator, Fireworks, or ImageReady, use any of the following techniques:

- ✓ **Drag and release:** Select the Slice tool from the Tools panel and drag the cursor through the image to create a rectangular marquee-like selection. When you release your mouse button, the previously whole image is sliced into pieces where you made the incision marks with the Slice tool. For instance, if you were to drag the Slice tool to create a shape about 3 inches wide through the center of an image, when you released the tool, you'd end up with three slices, like the example shown in Figure 3-5.
- ✓ **Create slices from guides:** In Photoshop, Illustrator, or ImageReady, set your document to display rulers (choose View⇧Rulers) so that you can use guides as boundaries from which slices will be generated. Drag guides from the top and left rulers and release them into the file where you'd like the image to be sliced, as in the example shown in Figure 3-6. Place as many guides as needed into your file so that the application can use them to divide the larger image into smaller parts. Slice commands in various programs are as follows:

- *Illustrator*: Choose Select⇨Select All to select all the objects in your layout. Choose Object⇨Slice⇨Create from Guides.
- *Photoshop*: Select the Slice tool and click the Slices from Guides button on the Options panel.
- *ImageReady*: Choose Slices⇨Create Slices from Guides.

If, after creating slices from guides, you find that the application has created too many slices, or if the slices don't quite meet your needs, you can combine and further divide slices horizontally and vertically to your liking using the program's other Slice tool features.

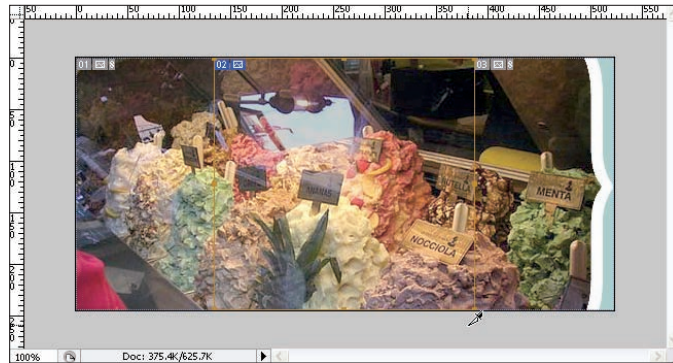


Figure 3-5: Slice your image into pieces before you optimize your graphics.



Figure 3-6: Drag guides into your layout to help you slice images with precision.

- ✓ **Create slices from guides:** In Fireworks, set down guides as you would in Photoshop, Illustrator, or ImageReady to divide the larger image into smaller pieces. Next, use the Slice tool to create the individual slices. As you drag with the Slice tool, each slice snaps to the Slice Guides.

To further illustrate how easy it is to slice up an image in either Photoshop or Illustrator, open your image and then follow these steps:

1. **With your file open in the document window, select the Slice tool from the Tools panel.**
2. **Drag a rectangular marquee shape around the first area in your file that you want to convert to a slice.**

As you release the mouse button, the program adds a slice border and slice number to the slice area. You might also notice that the remaining parts of your layout suddenly have grayed-out *auto slices*. The auto slices aren't actual slices, but rather are a way for the application to anticipate how the rest of the file will be sliced up. You can use these auto slices as a guide to create actual slices. Slices are numbered automatically from top to bottom and from left to right.

Photoshop users can use the Slices from Guides button to automatically create slices from the guides placed in the layout, or create slices from the contents of individual layers by choosing Layer↔New Layer-based Slice.

To toggle on and off the visibility of slices on the artboard, do one of the following:

- *In Illustrator:* Select View↔Hide Slices or View↔Show Slices.
- *In Photoshop:* Select View↔Show↔Slices.

3. **Repeat Step 2 to finish slicing up the rest of your graphic, like the example shown in Figure 3-7.**

You can also slice your image in any of the following ways:

- Select one or more objects in your layout and choose Object↔Slice↔Create from Selection.
- Select one or more objects in your layout and choose Object↔Slice↔Make.
- Let Illustrator create slices based on the document guides by choosing Object↔Slice↔Create from Guides.

4. **(Optional) Select an output option, such as Image, No Image, or HTML Text (Illustrator only).**

You can program image slices before optimization to support a variety of output options:





Figure 3-7: To optimize properly, slices should carve up the entire area of your layout.

In Illustrator: You can set slices to Image, No Image, or HTML Text by choosing Object→Slice→Slice Options.

In Photoshop: You can set slices to Image or No Image by accessing the Slice Options dialog box. To open this dialog box, click the Set Options for Current Slice button on the Options bar, or right-click on the slice and choose Edit Slice Options.

Here's the lowdown on the output options:

- **Image:** Slices with this setting are optimized as regular Web graphics. To assist with coding in CSS after the graphic has been optimized, enter a Name or ID for each slice, and if needed, set the URL of the HTML page that the graphic will link to when clicked. Linked slices should also include the Target for the URL, which can be set to `_self` so that the linked URL page opens in the same browser window. If desired, enter a short message that will appear in the

browser's status bar when a cursor is positioned over the graphic. Finally, add Alt text to identify the slice with a word or two; the words entered here are useful as text that can be read by a screen reader for visitors who are visually impaired. To set the background color of the slice, select one of the options from the Background type drop-down menu at the bottom of the dialog box.

- *No Image*: A slice with this setting does not optimize any graphics within the slice but instead sets the background color of the slice dimensions to match the background color specified, or when no background color is set, the slice remains blank. To add placeholder text to the slice area in the output HTML file, select the Text Is HTML check box if applicable and type your text into the Text Displayed in Cell field. Take care to only enter a few words that do not exceed the dimensions of the slice. If desired, apply any of the other settings available.
- *HTML Text (Illustrator only)*: This option only becomes active if text is turned into a slice using the Object⇨Slice⇨Make command. When activated, Illustrator converts the text and any of its basic formatting into HTML code during optimization.

5. (Optional) To adjust the borders of any of the slices, you must select all the slices that are touching to make the adjustments without breaking the order of the slices.

To shift the slice borders, select the Slice Selection tool (Illustrator) or the Slice Select tool (Photoshop), click once on the inside of the first slice that needs to be adjusted, and then Shift+click to add any additional slices as needed to the selection. Next, by placing your cursor above the slice boundaries, your cursor temporarily turns into a double-sided adjustment arrow that you can click and drag to move the slice boundaries to a new position. Repeat this select-and-drag process until the slice boundaries are in the desired position.

6. (Optional) If you want to combine, divide, duplicate, move, resize, align, distribute, release, delete, or lock selected slices, right-click (Windows) or Command+click (Mac) a slice to choose an option from the context menu and/or click any of the slice-specific buttons on the Control panel (Illustrator) or Options bar (Photoshop).

7. After you are finished slicing up your graphic, you can optimize and output your file in the appropriate file format.

Selecting the Right Web Format

You presently have three different file formats — GIF, JPEG, and PNG — that you can choose from when optimizing graphics for the Web. Each has different strengths and weaknesses, as illustrated in Table 3-2, making it fairly easy for you to identify which format will work best for your individual Web graphics.

Table 3-2 Graphics File Formats				
<i>Format</i>	<i>Is Best For</i>	<i>Maximum Colors</i>	<i>Transparency and Animation</i>	<i>Compression</i>
GIF	Images with large, flat areas of color	256 colors (8-bit) and grayscale (8-bit)	Supports both animation and background transparency	Lossless LZW compression
JPEG	Photographs and graphics with lots of color and gradient blends	Millions of colors (24-bit) and grayscale (8-bit)	No animation or transparency	Lossy compression
PNG	Replacing GIFs by the World Wide Web Consortium (W3C); images with large, flat areas of color	PNG-8: Maximum of 256 colors (8-bit) and grayscale (8-bit) PNG-24: Millions of colors (48-bit)	PNG-8: No animation or transparency PNG-24: No animation, supports background transparency	Lossless LZW compression

For years now, Web browsers have supported both the GIF (Graphics Interchange Format) and JPEG (Joint Photographic Experts Group) formats. However, it has only been more recently that the PNG (Portable Network Graphic) file format has increased in popularity and is finally also widely supported by most new browsers. This isn't to say that all currently available browsers should display PNGs. On the contrary, browser incapability issues still abound, so you may need to still be careful about how you use PNG files in your Web pages. For example, Internet Explorer 6 (IE6) displays the transparent areas in PNG files as a nice shade of baby blue instead of clear.

All three of these file formats use different compression algorithms to crunch data and produce smaller graphic files. The format you select ultimately determines how the final optimized graphic looks. Referring again to Table 3-2, it's quite easy to grasp which format you should use for each graphic you optimize. Choose JPEG for all your photographic images, and use the GIF or PNG-8 format for images that contain large, flat areas of no more than 256 colors. The PNG-24 format is really cool because it supports background transparency, like the GIF format, and displays millions of colors, like the JPEG format; however, it is not yet supported by browsers.



Web files that include additional information data like video, sound, or other multimedia must be compressed by other software applications for transmission over the Web. File formats for those types of Web objects include PDF, MP3, MPEG, Flash SWF, and Shockwave.

Saving your graphics in the correct Web file format is easy if you follow a few basic rules that will ultimately determine the final image's quality and file size. Take a look at Figure 3-8. The simplest way to decide which format to optimize it in should be based on the contents of the image:

- ✓ If the image is a photo or has a lot of gradient blends in it, choose JPEG.
- ✓ If the image has large, flat areas of color or text, choose GIF or PNG.



Figure 3-8: Optimize your Web graphics with photographs and gradients as JPEG and all others as GIF or PNG.

To better evaluate the benefits of each format and help you choose the right format for your graphics, here is a bit more information:

- ✓ **GIF (Graphic Interchange Format):** This format, created by the folks at CompuServe, is officially pronounced *jif*, like the peanut butter, but it's more commonly called *giff*, with a hard *g*, as in *get*. The GIF format supports a maximum palette of 256 colors and is great for images with text and large, flat areas of color.

Using a special LZW (Lempel-Ziv-Welch) lossless compression algorithm to shrink the file without removing detail, the GIF format can reduce file size up to about 60 percent of the original size during optimization.

Colors in a GIF image that aren't part of the 256-color palette can be *dithered* (two colors alternated in a grid pattern) to approximate the missing colors. For more on dithering, see the next section in this chapter.

GIFs support transparency, which means that parts of the image can be fully opaque (solid) while other parts are fully transparent, allowing you to see through to any objects or colors behind the image when placed on a Web page.

GIFs also support animation through a rather crude optimization technique that saves multiple images to separate frames inside a single GIF file. To create a simple GIF animation, place each of the different images in the animation onto layers inside any graphics program that supports the creation of GIF animation, such as Photoshop or Fireworks. You can then use the Make Frames from Layers command on the Animation panel options menu to create a frame-based animation sequence, as illustrated in Figure 3-9. Optimize the file as you would a regular GIF file. When viewed in a browser window, each frame of the animated GIF file plays one after the next (either with or without looping at the end), giving the illusion of a little movie.

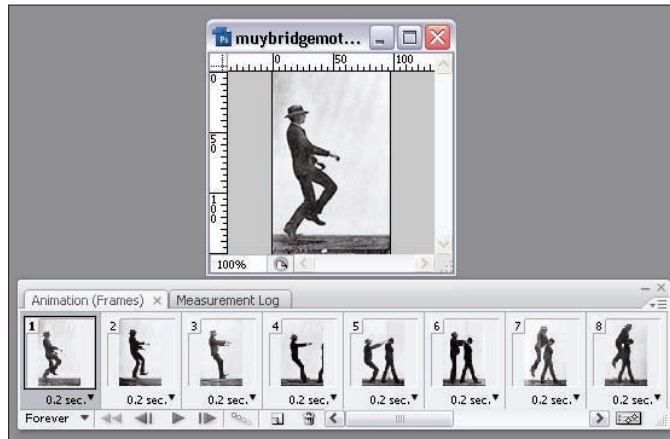


Figure 3-9: Create simple GIF animations from multiple layers in your file.

- ✓ **JPG (Joint Photographic Experts Group):** Choose the JPG/JPEG format, pronounced *jay-peg*, for any image that includes a photograph or a significant area of gradient color. The JPG format supports a palette with millions of colors and can help create photographic images for the Web with smaller file sizes than a GIF or PNG file would, because these other formats would resort to dithering to create any colors beyond the 256-color palette, thereby increasing the file size. JPEGs use a compression

method called *lossy*, whereby some digital information in the image is removed and becomes irretrievable (lost) after the compression.

As good as they are for photos and gradients, JPEGs do not support transparency or animation. Should you need either or both of these features, you must save the file as a GIF image and potentially suffer some loss of image quality in lieu of the gain of transparency and/or animation.

✓ **PNG (Portable Network Graphics):** This file format, pronounced *ping*, was created as a royalty-free raster alternative to GIFs, TIFFs, and occasionally JPEGs, combining the best features of compression algorithms, such as *lossless* compression and support for millions of colors and transparency. You find two flavors of PNG files:

- *PNG-8:* Because it also supports a maximum of 256 colors and grayscale images, the PNG-8 format has been recommended as a replacement for the GIF format by the World Wide Web Consortium, as long as the image does not contain any transparency and is not intended to animate. Most recent Web browsers (IE7 and IE8, Firefox, Opera, and Safari) support this format without issue; however, earlier browsers may not.
- *PNG-24:* The PNG-24 format, by contrast, is superior to the PNG-8 because it supports background or alpha transparency and millions of colors (but not animation), and this format should be supported by the newest browsers (IE8, Firefox, Opera, and Safari).

Nonetheless, before you go saving all your files as PNG-8 and PNG-24, be sure to test the images on a sample Web page in all your favorite browsers on both the Mac and PC to ensure that your files will be properly supported.

Probably the best way to really understand how Web graphics will look when compressed by one format or another is to see what each compression algorithm does to the image during the compression process. Figure 3-10 shows what happens to a graphic with flat color and text when saved as a GIF, JPG, and PNG. Figure 3-11 shows what happens to a photographic image when saved in the same three formats.



Figure 3-10: Save images with large, flat areas of color and text as a GIF or PNG for best results.



Figure 3-11: Photographic images and files that contain gradients are best saved as JPGs.

As you can see, the GIF format is the better choice for the graphic with flat areas of color because JPGs and PNGs tend to pixelate the flat areas, especially around the edges of letter shapes. Likewise, JPG is the better choice for the photograph because it handles gradients and shadow areas better than GIF and PNG formats, which tend to display those areas as bands of color rather than a single smooth transition of color. PNGs are good replacements for many GIF files, but if you decide to use them, pay attention to file size and be sure to test their displayability in all the browsers your target audience will be likely to use.

Choosing Web Optimization Settings

Most Web image optimization programs allow you to either choose from a preset list of default optimization settings or customize your own settings to create the optimized graphics to your specifications. The particular settings you choose determine several things about the output graphics, including the file format, size, number of colors, and quality.

Your main goal when optimizing graphics is to create the best-quality image with the smallest possible file size. If you reduce the file size too much, the quality might suffer, and if you make the quality high so that the image is sharp, the file size might be too large. With every image, a balancing point of good quality and decent file size exists, and you'll soon figure out which settings work best for all your different kinds of graphics.



As you're discovering which format and settings are best, take advantage of your optimization tool's 4-Up panel, which allows you to compare three different optimization configurations side by side so that you can easily select the one with the optimization settings that create the best-quality image at the smallest possible file size.

For best results with your image optimization, try using the following guidelines:

- ✓ Try to make your images as small as possible. Each image should be in the 10K or less range, though for larger graphics and photos, up to 30K should be fine.
- ✓ Add up the file sizes of all the images on your page. For quick display in a browser, the display time for an entire Web page should not exceed 8 seconds using a 56K modem. Ideally, each page should have no more than approximately 40K in images.

Now on to which format to choose and how to configure that format's settings so that you get the best possible optimized file every time.

GIF and PNG-8 optimization

The Save for Web & Devices dialog box in Photoshop and Illustrator has a special optimization panel that allows you to customize the settings for your Web graphics prior to optimization. If you plan to use the ImageReady program as a stand-alone application, configure the optimization settings through the program's Optimize panel. Fireworks users can apply optimization settings through the Optimize panel and export directly or with the help of the Export Wizard. When using any other Web graphic optimization tools, feel free to refer to the following list when choosing your optimization settings.

GIF and PNG-8 graphics share the following optimization settings:

Color Reduction Algorithm: The color algorithm establishes how the colors in the GIF or PNG file will be compressed. The algorithm is calculated by the color reduction type and the number of colors selected in the Colors field. You find four general algorithms:

- ✓ **Adaptive:** Produces a color palette in the image by sampling colors from the image itself. By reducing the number of colors in the palette, the reduction algorithm produces a smaller file.
- ✓ **Selective:** Produces a color palette like the Perceptual palette while preserving any Web-safe colors in the image and using only colors found in the graphic. This is the default option for GIF compression and tends to produce the most realistic color in the final output graphic.
- ✓ **Perceptual:** Produces a color palette that favors colors that the human eye is sensitive to.
- ✓ **Restrictive (Web):** Produces a color palette in your image that favors the 216 colors in the Web-safe palette and prevents colors in the image from being dithered. This means your graphic's colors will be automatically converted to Web-safe colors when choosing this algorithm. You can, however, adjust the percentage of colors that Web Snap in an adjacent slider. Unused colors are discarded from the palette.

Internet connection speeds

Even though well over half of all users on the Internet have cable, DSL, or T1 access, believe it or not, the rest of them are still visiting Web pages with 28.8K and 56K modems, which means pages for these visitors download at only about 4K per second. That's r-e-a-l-l-y slow. For visitors to see the graphics on a Web page, all the images must be transferred to the temporary cache area of the visitor's computer. For instance, a 30K file takes about 7 seconds to load on a computer with a 56K modem. For a lot

of visitors — especially the ones using older equipment with slower connection speeds — if the page takes longer than 8–10 seconds to load, the visitor might lose patience and leave the site before even seeing it! Unless you know for a fact that your target audience has a high-speed Internet connection, try your hardest to ensure that your pages (including HTML, images, CSS, JavaScript, and any other multimedia plug-ins or page enhancements) load within the 8- to 10-second time frame.

Lossy: (GIF only) This compression format used for GIF files removes image data, reducing the file size by as much as 40 percent. The larger the lossy number, the more data is removed. Typical lossy settings range between 0 and 10 with little noticeable image degradation. You cannot, however, use this feature with interlaced GIFs or when the Pattern or Noise Dither options are selected.

Colors: Use this setting to adjust the total number of colors that appear in the image, as indicated in the corresponding Color Table at the bottom of the panel. The maximum number of colors in a GIF or PNG-8 image is 256, and the minimum is 2 (black and white). Web-safe colors, when detected, appear in the Color Table with a tiny white diamond in the center of the color square; non-Web-safe colors appear as solid color squares.

Dither: Dithering is a color-simulation system whereby two colors are alternated in a checkered or random noisy pattern that tricks the eye into seeing a new solid color based on the combination of the two colors used. A dithered green color, for example, could be created by dithering blue and yellow pixels. This effect, while at times improving the image quality, does add to the overall file size.

Dithering can be good for images with flat areas of color and for images containing gradient colors that need to be saved in GIF format because, without it, the colors in the gradient tend to band out into stripes of color. This banding effect also occurs when viewing a Web page on a monitor with a resolution set to 256 colors.

The Dither setting has four different options:

- ✓ **No Dither:** No dither is applied to the image.
- ✓ **Diffusion:** A diffusion dither applies a more random dither pattern than the Pattern dither option. You can adjust the percentage of the dither in the Amount field on the Optimize panel to control the amount of dithering in the image. The larger the percentage, the more dithered colors are in the resulting image.
- ✓ **Pattern:** A Pattern dither creates dithered colors in a squarish pattern that might be more noticeable to the eye than a Diffusion or Noise dither. You can control the amount of dither by adjusting the dither percentage.
- ✓ **Noise:** A Noise dither uses a more random pattern for the dither than Diffusion or Pattern. Set the dither percentage.

Transparency: Both the GIF and PNG formats allow you to save images that contain transparent pixels. The transparent parts of the image appear as see-through when placed on a Web page, and any underlying colors on the page (such as the page background color, a table cell, or a table background color) show through those areas. The amount of transparency, on a scale of 0 to 100 percent, determines how many of the pixels will be transparent, semitransparent, or opaque. The transparency setting has the following four options:

- ✓ **No Transparency:** In images that contain transparency, any transparent and semitransparent pixels appear as opaque or semiopaque against the color selected in the Matte color field (see the following Matte description). For example, if the Matte color field is set to black, the transparent and semitransparent pixels in the image appear as if they are sitting on a black background.
- ✓ **Diffusion:** This option controls the dithering pattern of semitransparent pixels along the edge of the transparent area(s) in the graphic. When selected, a random dither pattern is applied to semitransparent pixels.
- ✓ **Pattern:** With this option, semitransparent pixels along the edge of the transparent area(s) in the graphic are dithered with the selected matte color in a squarelike pattern.
- ✓ **Noise:** This option creates a more random dithering pattern than Pattern and Diffusion by blending semitransparent pixels with the selected matte color in an irregular, almost haphazard pattern.

Matte: If you know what color the image with transparency will be “sitting on” in a Web page, select a matching matte color to ensure that semitransparent pixels in the image blend smoothly with the background color. For example, if you know the background color of a page is a particular red with the hexadecimal value of #cc3333, set the matte color for your image with transparency to the same hexadecimal value.

Interlaced: Selecting this option causes the image to be downloaded in the visitor's browser in multiple passes, giving the viewer something to see as the image gets drawn in the browser window. This option is good for larger images but not so necessary for individual images smaller than 10K.

Web Snap: The percentage number you select for Web Snap determines the number of colors in the color table that snap to the Web safe palette. The higher the number, the more the colors in the resulting optimized image will be forced to snap to colors in the Web-safe palette. Although the Web safe palette is no longer a major concern for designers, visitors with older monitors who view Web pages that include non-Web safe color will likely see a big color shift.

PNG-24 optimization

The PNG-24 setting only has three options to configure:

Transparency: When the transparency option is selected, the image is optimized with transparent parts. When the transparency option is deselected, any transparent areas in the image are filled with the specified matte color.

Matte: Choose the default white matte or specify another color to ensure that semitransparent pixels in the image are evenly filled in.

Interlaced: Select this option for larger files to make the file automatically download in the visitor's browser in multiple passes, giving the viewer something to see as the image gets drawn in the browser window.

JPG optimization

The JPG file format has many more different settings than the GIF and PNG formats. The following is an overview of the different JPG optimization settings:

Optimized: (Optional) Select this setting to enhance the compression of the resulting image. Though this option may slightly reduce the image file size, optimized graphics might not display well, or at all, in some older browsers.

Quality: The Quality slider controls the amount of compression applied to the resulting optimized image. Choose from five preset quality options (Low, Medium, High, Very High, and Maximum) from a drop-down menu, or use the Quality slider to manually select the quality on a scale from 0 to 100 percent. The higher the percentage, the better the image quality and the larger the file size. To find the best quality with the smallest file size, compare three different qualities using the optimization tool's 4-Up panel.

Blur: Images that use the lower-quality compression settings begin to produce jagged areas in the image. These *jaggies* are especially noticeable around the edges of contrasting colors, on flat areas of color, and along the edges of text. Jaggies can be somewhat reduced by applying a slight blur to the image, thereby reducing the file size. However, the more blur you add, the less crisp the resulting image. If you do choose to apply a blur to your file, keep the setting below 0.5 for best results.

Matte: When the original image has any areas that are blank (such as a transparent background layer), those pixels will be filled with the opaque color selected in the Matte field. If you can, try to match the matte color to the background color the image will be placed on top of on the Web page. For example, if the image will sit in a table cell with a red background with a hex value of #bf3b3b, set the matte color to #bf3b3b before optimization.

Progressive: Like the interlaced option for GIFs and PNGs, the progressive setting creates JPG images that display in multiple passes as the image downloads by displaying a low-resolution version of the file until the high-resolution version is finished downloading on the visitor's computer. Some older browsers might not support this feature, so be sure to test at least one progressive image in your target browsers on both Mac and PC before optimizing all your graphics with this setting.

ICC Profile: The ICC profile describes information about the graphic's RGB or CMYK color settings so that the color displays as intended. Select this option to preserve the image's ICC profile and have it embedded in the image to assist some browsers with color correction of the image.

Optimization Output Options

Whether you're optimizing a single graphic file or a fully sliced mock-up, you'll probably encounter a few of the same choices when it comes time to save your optimized files. For example, when using the Save for Web & Devices dialog box in Photoshop or Illustrator, or the Save for Web dialog box in ImageReady, you begin the optimization process by configuring the options for your graphic, whether that be a JPG, GIF, or PNG file. Then, to create the actual optimized graphic, you need to click the Save button and make a few output decisions regarding the file's extension, selected slices, output location, and output file types. For example, to automatically generate an HTML file and have all your sliced and optimized graphics placed into an images folder on your Desktop, choose the following Output settings:

- ✓ Save as: test.html
- ✓ Location: Desktop
- ✓ Format: HTML and Images

- ✓ Settings: Default Settings
- ✓ Slices: All Slices under Slices



The HTML file and images folder that the Save for Web & Devices dialog box outputs can be the starting point for your Web-building process when you are ready to build your first Web page. Therefore, rather than send the optimized files to your desktop, send them to the location on your computer where you'd like to save all the files for that specific Web project.

To keep your files organized, you might want to use some kind of naming convention, such as always calling your Web project folders `COMPANY_HTML` so that they are easy to identify from other files and folders. In fact, if you organize your client projects into individual folders, you can keep better track of all your projects and their attending documents. Consider, for example, keeping your files organized by using the method shown in Figure 3-12. Feel free to use this same naming and organization convention if you think it will help you keep track of your projects.

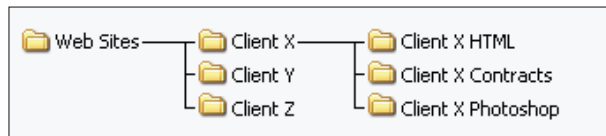


Figure 3-12: Using a uniform naming and filing convention.

Because each optimization program has similar, though possibly slightly different, options regarding how and where the files will be optimized and output, use the following steps (which are roughly the same for Photoshop, Illustrator, and ImageReady) as general guidelines:

- 1. Launch your program's Save for Web & Devices optimization dialog box and click the Save button to access the optimization output options in the Save Optimized As dialog box, shown in Figure 3-13.**
- 2. In the Save In drop-down list, choose a location on your local computer or network where the optimized file will be saved.**

This can be on your desktop or in a folder somewhere else on your computer, such as a folder for this particular Web project.

- 3. In the File Name field, enter the name of the graphic file.**

This filename will be used to save the graphic or graphic with HTML with the selected file type.

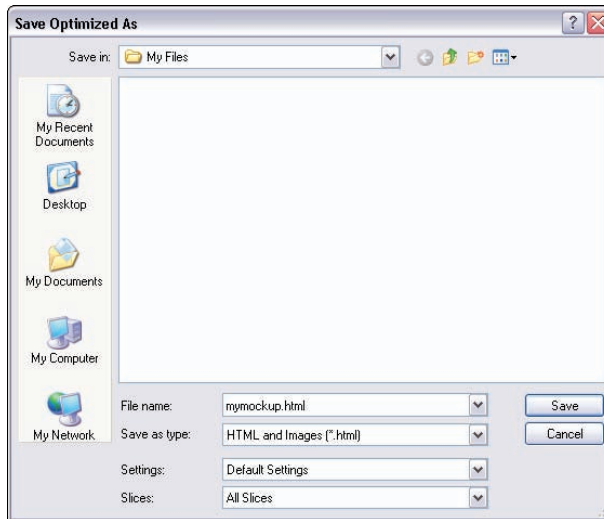


Figure 3-13: Choose a file type and format for your optimized graphic output.

Filenames can be anything you like as long as they use the proper file extension such as `.gif`, `.png`, `.jpg`, or `.html` (for example, `contact.png` or `about.html`).

4. In the Save as Type field, choose one of the following options:

- *HTML and Images:* Saves the image or selected slices as optimized graphics in your selected file format along with a tables-based HTML file that includes the optimized graphics you are about to optimize inside a folder called images.
- *Images Only:* Saves just the image or selected slices as optimized graphics in your selected file format into a folder called images.
- *HTML Only:* Saves just an HTML file formatted with a table to support the graphics in your document.

5. From the Settings drop-down list, choose Default Settings.

These settings determine the various output settings, such as placing all of the optimized graphics into a folder called images. If desired, you can customize these settings by selecting one of the other options from the drop-down menu (Custom, Background Image, XHTML, or Other).

6. For multisliced images, choose an option from the Slices drop-down menu.

Your options are All Slices, All User Slices, or Selected Slices. If you are unsure which option to select, choose All Slices. That way, you get a folder full of optimized images based on all the slices in your layout.

7. Click the Save button to complete the optimization output process.

Your optimized files are saved to your computer in the location specified in the Save In location.

After the Save Optimized As and Save for Web & Devices dialog boxes close, you can preview your graphics in a browser window. To view the HTML page that contains the optimized graphics, double-click the HTML file or drag and drop the file into any open browser window. To preview any individual optimized graphic, drag and drop the file into any open browser window.

Book III

Building Web Sites

The 5th Wave

By Rich Tennant



“Is this really the best use of Flash animation on our e-commerce Web site? A bad wheel on the shopping cart icon that squeaks, wobbles, and pulls to the left?”

When it's time to build your site, you'll be delving into the world of HTML, CSS, and JavaScript. Using your preferred HTML editor, you'll be creating all the individual pages of your site from the optimized graphics you created from your mock-up.

Here you find chapters about adding content to your pages, including text, images, hyperlinks, tables, lists, and media files. You also discover a host of information about working with Cascading Style Sheets to style and position your content in the most user-friendly, accessible, standards-compliant manner. This minibook is rounded out by additional chapters on creating layers-based layouts, working with navigation systems, designing Web forms, making your site interactive with JavaScript, and discovering how to work efficiently with templates and Server-Side Includes.

```

74 </head><body>
75 <h1>Sample CSS List Navigation Menu</h1>
76 <div id="tablismenu">
77   <ul>
78     <li><a href="#" title="About Us"><span>About Us</span></a>
79     <li><a href="#" title="Services"><span>Our Services</span></a>
80     <li><a href="#" title="Clients"><span>Our Clients</span></a>
81     <li><a href="#" title="Press Releases"><span>Press Releases</span></a>
82     <li><a href="#" title="Employment Opportunities"><span>Employment Opportunities</span></a>
83   </ul>
84 </div>
85 <p><strong>Here are the graphics you'll need copies of for
86   your mouse over each graphic, Right-click (Win) or Control-click (Mac)
87   to open the context menu, and choose Save As or Save Target As.
88 <div id="graphics">
89   <img alt="Graphic 1" data-bbox="150 230 250 300"/>
90   <img alt="Graphic 2" data-bbox="260 230 360 300"/>
91   <img alt="Graphic 3" data-bbox="370 230 470 300"/>
92   <img alt="Graphic 4" data-bbox="480 230 580 300"/>
93   <img alt="Graphic 5" data-bbox="590 230 690 300"/>
94 </div>
95 <div id="graphics">
96   <img alt="Graphic 1" data-bbox="150 310 250 380"/>
97   <img alt="Graphic 2" data-bbox="260 310 360 380"/>
98   <img alt="Graphic 3" data-bbox="370 310 470 380"/>
99   <img alt="Graphic 4" data-bbox="480 310 580 380"/>
100  <img alt="Graphic 5" data-bbox="590 310 690 380"/>
101 </div>

```



The screenshot shows a web form with the following fields and options:

- *Name:
- *Telephone: Ext:
- *Email:
- Username:
- Password:
- Reenter password:
- Password Hint: Select a question
- I am interested in:
 - Web Design
 - Illustration
 - Print Design
 - Other
- Join Mailing List:
 - Yes
 - No
-

Chapter 1: Adding Text, Images, and Links

In This Chapter

- ✓ Understanding HTML basics
- ✓ Working with semantic markup
- ✓ Inserting text and graphics
- ✓ Hyperlinking text and graphics to other pages
- ✓ Labeling objects in preparation for using CSS
- ✓ Making page content accessible with HTML

At this point in the design process, you have already discovered a little bit about HTML coding, syntax, and structure. To complement that knowledge, you've also made several important decisions about the look and feel of your design and have hopefully already mocked up the home page in your preferred graphics software program and presented your design to your Web client for review and approval. In this chapter, you find out how to put all those pieces together into a single HTML document.

To start, you find out about setting up a basic, bare-bones HTML page, which you can use for any Web project. After that, the specifics of your particular Web site come into play. The first couple of times you put a Web site together can certainly feel daunting, to say the least. That feeling of building a Web site from scratch can be similar to the feeling a painter has when looking at a fresh blank canvas. Where should you begin? What should you do first? While no one perfect solution exists, try not to let the options overwhelm you. Instead, focus your energies on building that first page. After the first page is built, constructing the rest of the site should come relatively easily.

In addition to the basics, this chapter also covers using meta tags; adding content (such as text and graphics) to the body of the page; creating hyperlinks to other pages from text and graphics; marking up content and labeling objects properly in preparation for using CSS and applying JavaScript; and improving page accessibility for all Web visitors, both human and machine.



Setting Up Basic HTML

To ensure that your Web pages display properly in a browser window, your HTML code must include several necessary components, each of which must be placed in the correct order. As you'll probably discover with a mistake, when the order is incorrect, if a tag is misspelled, or if any of the parts of the code are missing — like a period (.) or slash (/) — the page will probably not display correctly (or at all) when viewed in different browsers. Even more annoyingly, some browsers might display the page fine, while in other browsers, the page looks all crazy and crooked. In fact, the more you understand about building Web pages, the quicker you'll discover that a single page can look very different in different browsers, even when all the HTML, CSS, and JavaScript code is perfectly written!

A good, well-structured Web page has six core parts that should flow in the following order:

- ✓ DOCTYPE
- ✓ HTML tags
- ✓ Head tags
- ✓ Title tags
- ✓ Meta tags
- ✓ Body tags

As long as your Web page includes all these parts in the proper order, the page will have a solid foundation upon which the rest of your content can gently rest. Without them, your page may not display properly in a browser, nor communicate clearly to the person, browser, or device displaying the page, and perhaps worst of all, not provide the details necessary to help visitors find the site in a search engine. Because you clearly would want none of these problems, you need to follow a few simple rules to ensure that your pages are properly coded.

Adding the title, DOCTYPE, and metadata

As you find out in Book I, Chapter 4, a Web page uses a simple structure whereby HTML tags are hierarchically nested between opening and closing `<html>` tags, and between those, all the code in the page falls between either the `<head>` or the `<body>` tags, as follows:

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

In addition to these, which I call the “bones” of a Web page, you also need to include the page title, the DOCTYPE, and the meta tags for your page to be considered properly structured.

Adding a page title

The page title is the code that allows you to name and display a title for the page in the browser’s title bar. You can use up to about 70 characters, including spaces and punctuation, to describe the contents of the page. Designers often include the name of the site along with the name of the page and a few keywords that might help visitors find the page, such as “*Jet-Stream*” Showerhead(r) - Frequently Asked Questions about Low-Flow, High-Quality, Water-Saving Showerheads. You may (and should!) use unique, descriptive page titles for every page on your Web site because the title will help visitors find your site when using a search engine. When creating the page titles, try to use a short descriptive phrase that reads well, like the example shown in Figure 1-1, and keep the character count to about 70 characters or less; otherwise, the title may be truncated (cut off) in the title bar of the browser.

Adding a DOCTYPE

A DOCTYPE, also often referred to as a Document Type Definition (DTD) and sometimes called a Document Type Declaration (again, DTD), is a set of instructions to a browser that identifies the type of code that the page was written in, such as HTML, XHTML, or Frames. More importantly, a DOCTYPE informs the browser how the document should be interpreted as an application of the XML programming language.

The DOCTYPE is split into two parts:

- ✓ The first part defines (to the browser) the type of code that will be used on the page, such as HTML or XHTML.
- ✓ The second part displays the URL of a text file on the W3.org Web site that further describes how the DTD is to be used.

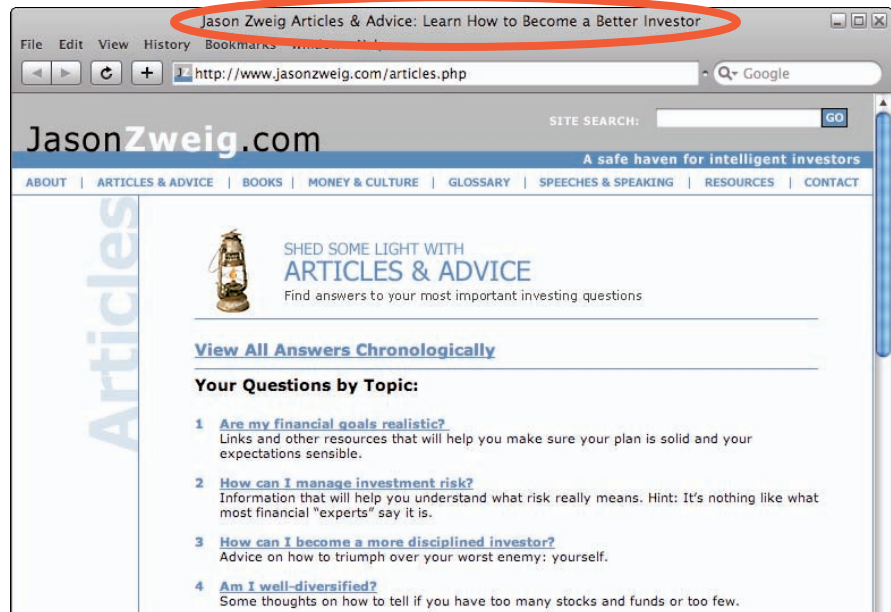


Figure 1-1: Each page on your site should have its own unique page title.

There are six different DOCTYPEs you can choose from, depending on whether you are creating an HTML or XHTML document. If you are unsure of which DTD to use, select 4.01 Transitional when writing HTML code and choose XHTML 1.0 Transitional when writing XHTML code because these are the two most forgiving and flexible DTDs of the bunch. For detailed explanations about all the HTML and XHTML DOCTYPEs, turn to Book IV, Chapter 1.

When using a visual editor such as Dreamweaver or an HTML code editor like BBEdit, those programs should automatically allow you to select which DOCTYPE DTD to include in the page, such as HTML 4.01 Transitional or XHTML 1.0 Transitional, and then insert the “bones” of a Web page into the code each time you create a new HTML document.

Here’s an example of the HTML bones you’d see if you created a new HTML document in Dreamweaver CS4 with the HTML 4.01 Transitional DTD:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
    Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
```



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

As part of the bones, Dreamweaver also drops the Content-Type meta tag into the page, which identifies the type of characters being used in the code. That tag can reside in the code either above or below the <title> tags, as long as it sits somewhere between the opening and closing <head> tags.

Adding metadata

Your meta tags are the special lines of HTML code that you add to your Web page between the opening and closing <head> tags to communicate important information about the site to Web browsers.

What's great about meta tags is that they are invisible to site visitors and do not appear in the browser window. The information you put in the various meta tags placed in your page can help with the indexing of a single page or an entire Web site in a search engine's database, as well as provide informative data, comments, and contact information to any visitors clever enough to view the page's source code.

To function properly, the meta tags themselves must be placed in the code somewhere between the opening and closing <head> tags of your page. More often than not, meta tags are inserted in the code directly after the closing </title> tag and before the closing </head> tag, like the example that follows. When using a visual or code editor, however, you may at times see the Content-Type meta tag automatically placed above the <title> tag, which is fine too:

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Snorkeling and Diving Adventures of Southern Florida</title>
<meta name="Keywords" content="Snorkeling, scuba diving, scuba divers,
      snorkelers, glass bottom boats">
<meta name="Description" content="Plan your aquatic vacation with SDASF. We
      offer snorkeling and diving adventures throughout southern Florida. Private
      charters and passenger vessels available.">
</head>
```

The two most important meta tags to include on every page to improve search engine indexing and ranking are Description and Keywords, as shown in the preceding example code:

- ✓ **Keywords:** While the `Keywords` meta tag is no longer read by most search engines, the tag is still worth including in your code because it provides at least one search engine — Yahoo! (which purchased Inktomi in 2003) — with keywords to help visitors find your site. In addition, you can use this meta tag to list obscure keywords or misspellings of commonly misspelled industry-specific terms. That said, you should only use the seven most important keywords, listed in order of relevance; any more than seven keywords will likely be ignored.
- ✓ **Description:** By contrast, the `Description` meta tag can use a maximum of 250 characters, including spaces and punctuation, to clearly describe what visitors can expect to find on the Web site. What's more, `Description` actually appears in the search engine results when the URL is listed after a visitor performs a keyword or key phrase search in his or her favorite search engine. That search will likely include a list of words, possibly separated by commas, that visitors might use to find the Web site's products or services.

Besides the `Description` and `Keywords` meta tags, you can add several other optional meta tags to your pages for communicating vital information to visitors and search engine robots and spiders:

- ✓ **robots:** This meta tag tells search engine spiders/robots whether the site should be indexed and whether links on pages should be followed when indexing is allowed. The default option is "index, follow", which both indexes the submitted URL and follows any hyperlinks to internal pages and external site links. You can customize the tag in any of the following ways, depending on your preferences:

```
<meta name="robots" content="All">
<meta name="robots" content="index, follow">
<meta name="robots" content="noindex, follow">
<meta name="robots" content="index, nofollow">
<meta name="robots" content="noindex, nofollow">
```

- ✓ **Content-Language:** This tag defines which written language is used for the content presented on the page, such as English, French, or German:

```
<meta http-equiv="Content-Language" content="en">
```

Get further information about language declarations from the W3C Web site:

www.w3.org/TR/i18n-html-tech-lang/#ri20050208.091505539.

- ✓ **Content-Type:** This meta tag instructs browsers about the type of character encoding, or set of letters, being used on the Web page, such as the A–Z alphabet or Chinese characters. Dreamweaver automatically includes this meta tag as part of the bones of a new Web document:

```
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
```

To find out more about character encoding, visit the W3C Web site at www.w3.org/TR/REC-html40/charset.html#doc-char-set.

- ✓ **refresh:** This meta tag instructs the browser to reload the page in the browser window at the specified number of seconds, such as 120 seconds for 2 minutes. Refreshing the page can be a useful tool for sites that provide up-to-the-minute data or sites that need to redirect visitors to a different URL. In the following example, the page would refresh after 30 seconds:

```
<meta http-equiv="refresh" content="30">
```



In recent years, this meta tag has sadly been misused by unethical Web bandits, and some search engines have begun penalizing sites that use it. Therefore, if you'd like to use the `refresh` meta tag to redirect visitors to a new permanent Web address, a 301 redirect would be a smarter solution.

For detailed instructions on creating a 301 redirect for your Web site, visit www.webconfs.com/how-to-redirect-a-webpage.php.

- ✓ **revised:** Displays the date the site was last revised:

```
<meta name="revised" content="Stephanie Dowling, Gym Kids, 1/23/09">
```

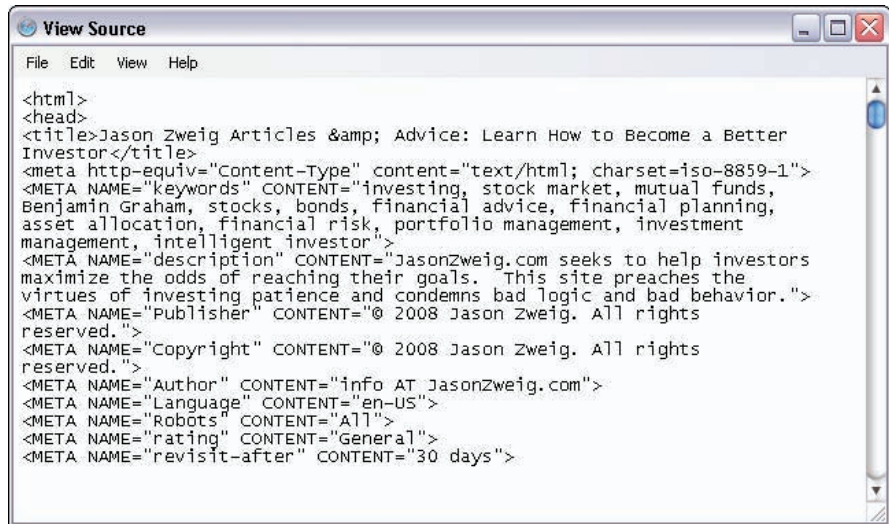


Unless you use some kind of programming or scripting language to automatically recognize and update the date for you, this meta tag must be updated manually.

- ✓ **Other tags:** Create and use other meta tags, such as `Publisher`, `Copyright`, `Author`, and `Reply-to`, to provide information about the publisher and author of a Web page:

```
<meta name="Publisher" content="More Love, More Joy!">
<meta name="Copyright" content="Copyright 2009, More Love, More Joy! All rights reserved.">
<meta name="Author" content="Jennifer Martin for www.morelovemorejoy.com">
<meta name="Reply-to" content="info-at-morelovemorejoy.com">
```

Figure 1-2 shows an example of how you might combine several of these meta tags onto your Web page.

A screenshot of a web browser's 'View Source' window. The window title is 'View Source' and it has a menu bar with 'File', 'Edit', 'View', and 'Help'. The main content area displays the following HTML code:

```
<html>
<head>
<title>Jason Zweig Articles & Advice: Learn How to Become a Better
Investor</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<META NAME="keywords" CONTENT="investing, stock market, mutual funds,
Benjamin Graham, stocks, bonds, financial advice, financial planning,
asset allocation, financial risk, portfolio management, investment
management, intelligent investor">
<META NAME="description" CONTENT="JasonZweig.com seeks to help investors
maximize the odds of reaching their goals. This site preaches the
virtues of investing patience and condemns bad logic and bad behavior.">
<META NAME="Publisher" CONTENT="© 2008 Jason Zweig. All rights
reserved.">
<META NAME="Copyright" CONTENT="© 2008 Jason Zweig. All rights
reserved.">
<META NAME="Author" CONTENT="info AT JasonZweig.com">
<META NAME="Language" CONTENT="en-US">
<META NAME="Robots" CONTENT="All">
<META NAME="rating" CONTENT="General">
<META NAME="revisit-after" CONTENT="30 days">
```

Figure 1-2: Meta tags help to identify site content to browsers for page indexing.

Coding pages by hand

If you are hand-coding your Web pages, the easiest place to begin building your first page is at the very top. Follow these steps:

- 1. In a blank text file in your preferred text or code editor, insert the desired HTML or XHTML DOCTYPE for your page.**

The DOCTYPE (DTD) needs to go at the very top of the code, above the opening `<html>` tag.

- 2. Add the opening and closing `<html>` tags to your page if your editor hasn't done so already.**



When creating XHTML files with Dreamweaver, the program should automatically add the `xmlns` attribute with the opening `<html>` tag when it drops in the DTD, as shown in the following example. If you're hand-coding an XHTML page, therefore, be sure to also type in that attribute.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
</html>
```

3. Enter the opening and closing `<head>` tags between the `<html>` tags.

The head area is where information that will not appear in the browser window goes. This includes the page titles, CSS styles, and JavaScript code and links to external CSS and JavaScript files, as well as any meta-data you might want to add to your page to provide information about the site to search engine robots.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
</head>
</html>
```

4. Directly above the closing `<head>` tag, enter the content type meta tag:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
</html>
```

5. Between the opening and closing `<head>` tags, enter opening and closing `<title>` tags, and between those, insert the text that will appear in the browser title bar.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Snorkeling and Diving Adventures of Southern Florida</title>
</head>
</html>
```

6. Beneath the `<title>` tags but before the closing `<head>` tag, add the Description and Keywords meta tags, as well as any other `<meta>` tags you would like the page to include.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Snorkeling and Diving Adventures of Southern Florida</title>
<meta name="Keywords" content="Snorkeling, scuba diving, scuba divers,
snorkelers, glass bottom boats">
<meta name="Description" content="Plan your aquatic vacation with SDASF.
We offer snorkeling and diving adventures throughout southern
Florida. Private charters and passenger vessels available.">
</head>
</html>
```

7. Place the content for the site between the opening and closing

`<body>` tags.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Snorkeling and Diving Adventures of Southern Florida</title>
<meta name="Keywords" content="Snorkeling, scuba diving, scuba divers,
snorkelers, glass bottom boats">
<meta name="Description" content="Plan your aquatic vacation with SDASF.
We offer snorkeling and diving adventures throughout southern
Florida. Private charters and passenger vessels available.">
</head>
<body>Page content goes here.
</body>
</html>
```

The details of this last step are what the bulk of this chapter is about — making sure that the text, images, tables, links, and more are all added to your HTML page. By the end of this chapter, you should have a Web page with code that is valid and ready to be formatted in CSS.

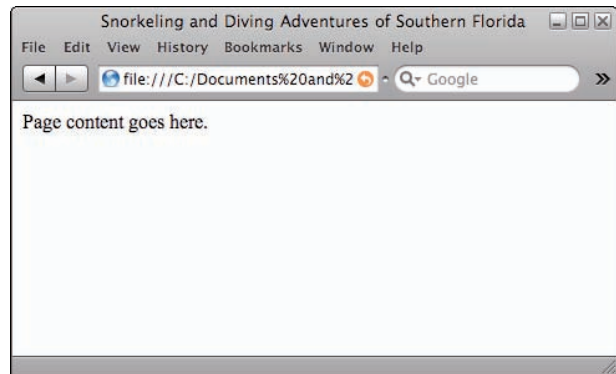


Figure 1-3: Metadata does not appear in the browser window.

All this metadata is invisible in the browser window. If you were to view this page in a browser, all you would see are the words Page content goes here., as shown in Figure 1-3.



Coding Your Pages

As you enter the code into your HTML files, here are a few important points to keep in mind:

- ✓ **If you plan on hand-coding your pages, be extra careful about entering all the code correctly.** You'd be wise to invest in an HTML book and/or spend as much time as you can learning HTML from an online tutorial to ensure that your HTML markup is compliant with the latest rules for HTML 4.01 (or XHTML 1.0). To code an image into a page by hand, for example, you need to insert the HTML tag for the image as well as include any attributes for it. *Attributes* are how you apply special characteristics to a tag using the syntax `attribute="x"`. Attributes include the width, height, and alternate text of an image; the alignment of a table cell; and the style of a bullet in an ordered list.
- ✓ **If you'll be using a code editor of some kind, be it a WYSIWYG code editor or an HTML-only editor, the application should do some of the coding work for you.** For example, when inserting an image with an editor, all you'll probably need to do is select the desired graphic file listed on your computer in a dialog box and, after it is inserted, add any desired attributes for the graphic through a special panel (like the Property inspector in Dreamweaver) that can then apply those attributes automatically to the code.

Regardless of the way you choose to enter your code, you'll soon naturally discover that any tiny mistake you make can have a big impact on how the page renders in different browsers. Believe it or not, a single extra space or a missing quotation mark can drastically change how the page is viewed!



Movies and other plug-ins must include the proper coding so that visitors can see them in a browser, and thus are a bit more complex to add to your pages, especially when hand-coding. In Book III, Chapter 8, you find out about more advanced content adding, such as embedding Flash movies and other multimedia plug-ins in a page.

When it comes to coding pages, the more you use semantic HTML, the easier it will be for you to identify the different parts of your content and use that content for a variety of purposes, such as applying CSS styles, pulling collected form data from visitors into a database, and adding JavaScript to objects to make the page more interactive.

Semantics refers to the proper usage of HTML 4.01 and XHTML 1.0 tags based on those tags' contents. For example, when adding a paragraph of text to your page, you'd mark up that paragraph with opening and closing `<p>` tags. Likewise, when creating a page heading, you'd use the `<h1>`

through `<h6>` tags, and when creating lists, use ``, ``, and `` tags. You get the general idea. What semantics does not refer to is how the contents of an HTML page will appear in a browser; the presentation of the page should, as much as possible, be defined with Cascading Style Sheets.

In addition to proper usage of tags, semantics also refers to avoiding the use of any deprecated tags, such as the old `<center>` tag for center alignment of objects on the page and `` tags for applying text styling. In fact, most books and online tutorials do not even cover these deprecated tags anymore. However, they're important to know about because you might, in the course of doing business, inherit an old site from a client for either site maintenance or a site redesign.



Deprecated refers to any HTML tag or tag attribute that has been phased out of usage in favor of better coding and styling methods. Most deprecated tags and attributes are not supported in HTML 4.01 and XHTML 1.0 markup, and thus might not work in newer browsers. Table 1-1 lists all the deprecated HTML tags along with suggestions for replacement methods.

<i>Deprecated Tag</i>	<i>Usage</i>	<i>Suggested Alternative</i>
<code><center></code>	Centers objects	Use the <code><div></code> tag with the alignment (<code>align</code>) attribute, as in <code><div align="center"></code> .
<code></code>	Applies styles to fonts	Style using CSS.
<code><basefont></code>	Sets default font style	Style using CSS.
<code><menu></code>	Creates menu lists	Use <code></code> or <code></code> .
<code><dir></code>	Creates directory lists	Use <code></code> or <code></code> .
<code><s></code> and <code><strike></code>	Applies strikethrough	Style using CSS.
<code><u></code>	Underlines	Style using CSS.
<code><applet></code>	Inserts applets	Use <code><object></code> .
<code><isindex></code>	Adds search fields to a page	Use <code><form></code> .

In addition to all these deprecated HTML tags, several HTML tag attributes, such as using `vlink` as an attribute of the `<body>` tag, have also been deprecated in favor of using CSS for styling text and other elements on a Web page. To avoid accidentally using any of these deprecated tag attributes, use CSS as much as possible to style the content and to position objects in your pages. That said, even if you do accidentally use a deprecated tag or tag attribute here and there, you will have a chance before you publish your

site to identify and fix it. As part of your prelaunch testing process, you'll verify the accuracy of your HTML/XHTML code with some online tools to ensure that the code meets the minimum standard requirements to be compliant with the DTD you selected for your pages.

Adding Page Content

Your page content refers to all the text, images, navigation, Flash movies, QuickTime movies, Flash animations, MP3 files, and other media files and plug-ins that can be viewed on a Web page. As long as your content is properly coded and placed between the opening and closing `<body>` tags of the page, it should appear in the body of the page in a browser window, though keep in mind that different browsers may display the content in slightly different ways.

In the following sections, you discover all the basics about how to add text and graphics to a Web page.

Inserting text

Text is, by far, the easiest thing to add to a page because you can quickly transform it into paragraph text and headings, organize it into a list, or place it into rows and columns within the structure of a table or layer.

As you add your copy to the page, try not to be too concerned about how the text looks (the font face, size, color, and so on), because the styling of the text will happen later with CSS.

Follow this simple process to add text to your Web page:

- 1. Type in or paste all your text onto the page before you do any formatting.**

It is much easier and faster for you to do all the formatting at once using CSS after the content is in place. The same goes for adding any dynamic functionality like JavaScript rollover buttons to the page; get the content on the page first and then add the dynamic functionality.

Take the following sample text, for example:

```
Creating Custom History Panel Commands
The History Panel is one of those tools in Dreamweaver that many users
don't take full advantage of. When the panel is open, it records all the
actions you make in an open document, up to a certain number of steps
(as specified in the General category of Dreamweaver's Preferences), and
lets you take multiple steps backward with the use of the panel's
slider.
```

2. Mark up the text with the appropriate HTML tags to define the different parts of the content.

Semantic tags have been added to the sample text below to identify both the main heading (`<h1>`) and the general paragraph (`<p>`) text:

```
<h1>Creating Custom History Panel Commands</h1>
<p>The History Panel is one of those tools in Dreamweaver that many
users don't take full advantage of. When the panel is open, it
records all the actions you make in an open document, up to a
certain number of steps (as specified in the General category of
Dreamweaver's Preferences), and lets you take multiple steps
backward with the use of the panel's slider.</p>
```

When viewed in a browser, this code is automatically formatted like the example shown in Figure 1-4.

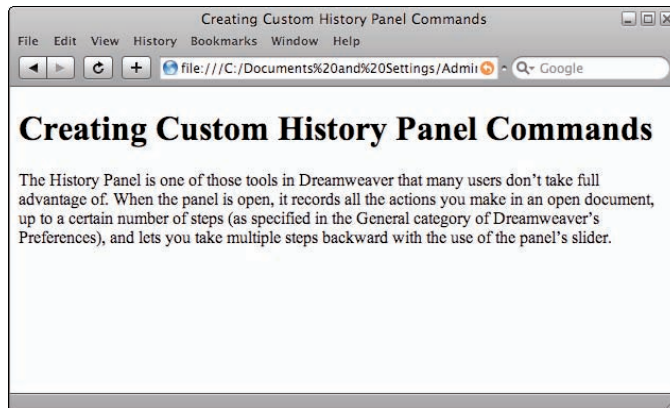


Figure 1-4: Semantic HTML means using the appropriate tags for the contents between them.

In the following sections, you find out about the different tags you can use to mark up your text.

Headings

Heading tags are special preformatted tags that identify the important parts of your text that are different from the regular paragraph text, such as headings, subheadings, and bylines, by making them bold and either slightly larger or slightly smaller than regular paragraph text, all without the use of CSS. Heading tags range from `<h1>` through `<h6>`, with `<h1>` being the largest preformatted text and `<h6>` being the smallest. Figure 1-5 shows how each of the following tags transforms the content in a browser:

```

<p>This is normal paragraph text</p>
<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<h3>This is heading 3</h3>
<h4>This is heading 4</h4>
<h5>This is heading 5</h5>
<h6>This is heading 6</h6>

```

Keep in mind that this kind of preformatting only determines the default look of content styled with these tags within a browser; with the magic of CSS, you can easily *redefine* the attributes of these preformatted tags.

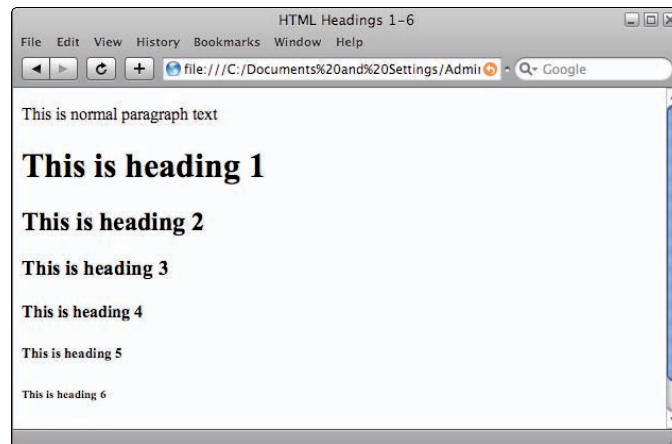


Figure 1-5: Use appropriate HTML tags to identify the different parts of the content, such as headings and paragraphs.

Bold and italic emphasis

When your text includes a word or phrase that needs to stand out from the rest of the content, use the bold or italic structural tags.



If you're familiar with the old HTML `` and `<i>` formatting tags, *do not use them!* Those tags are deprecated, and, while they may still be backward-compatible in HTML in some browsers, they won't be viewable in XHTML. Instead you should now exclusively use `` for bold and `` (which stands for emphasis) for italics. Use these newer structural tags in all your new Web pages, and convert any old bold and italic tags to the strong and emphasis tags when inheriting an old site for maintenance or redesign.

Strong and emphasis tags can be used independently or in tandem. When you need to make something both bold and italic, be sure to use proper tag nesting in the code, where the closing tags mirror the order of the opening tags, as follows:

```
<h1>Creating Custom History Panel Commands</h1>
<p>The <strong>History Panel</strong> is one of those tools in Dreamweaver that many users don't take full advantage of. When the panel is open, <em>it records all the actions you make</em> in an open document, up to a certain number of steps (as specified in the General category of Dreamweaver's Preferences), and lets you take <strong><em>multiple steps backward</em></strong> with the use of the panel's slider.</p>
```



One interesting fact about bold and italic emphasis that you may not know, especially because you can create CSS styles with these attributes rather than add these tags to your code, is that screen readers will actually inject different inflections into words that use the `` and `` tags! Even more interestingly, search engine algorithms will sometimes pay more attention to the contents between these tags than the other words on the page, which can both identify and improve the rankings of the page within a particular search engine.

Text alignment

Text can be aligned left, center, right, or justified, relative to the browser window or any container tag (such as paragraphs, headings, table cells, and layers created with `<div>` tags) inside which the content sits. To illustrate, the contents in Figure 1-6 use the left, center, right, and justify alignment tag attributes. However, keep in mind that not all browsers support these attributes (as with the justify alignment shown here) when added to the HTML.

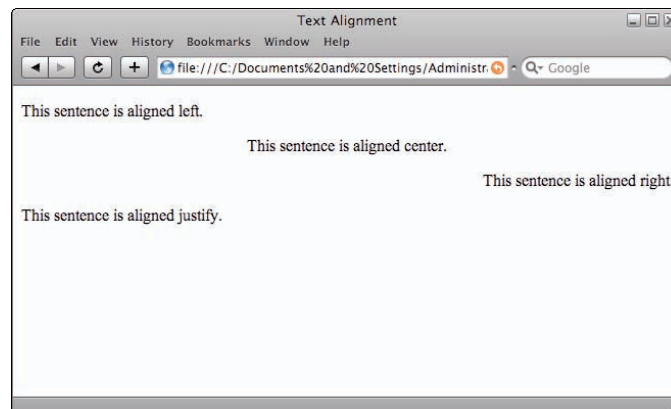


Figure 1-6: Text can be aligned to the left, center, right, or justified.

In the past, the `align` attribute would have been added to the opening tag, as in `<p align="left">`. Today, however, rather than using this attribute, you should leave the paragraph with no alignment markup and apply the alignment using CSS, which you explore in Book III, Chapter 3.

For an alternate method of text alignment, text can be indented from the left and right margins with the use of the `<blockquote>` tag, which is primarily meant for use with quotes:

```
<blockquote>Bookmark this site so that you can easily return here when you're
  ready to make a purchase.</blockquote>
```

However, if you need to add margin space to a block of text that was not a quote, use CSS to style that content instead.



You can decide on the font face, size, and color of text on a Web page when styling your content with Cascading Style Sheets, which you find out about in Book III, Chapter 3.

Adding graphics

In Book II, Chapter 3, you see how to optimize graphics for the Web into one of the three acceptable Web file formats: GIF, JPG, and PNG. With those images at hand, you are now ready to find out how to add them to your Web pages.

One of the best ways to keep all your Web graphics organized is to put all of them into a folder called `images` or `img`, and place that folder inside (at the root level) that project's HTML folder on your local computer. You can then repeat this same organizational structure for each project you work on, giving each project a root HTML folder (perhaps named something like `Emily K. HTML`) and placing a generic `images` folder inside it. When you create a folder for your images, it will be much easier for you to locate and insert them into your pages as you build your site. When your project requires that several categories of images need to remain separated, organize them into subfolders within the main `images` folder, such as a folder for images of skylines (`images/skylines/newyork.jpg`) or a folder for images of button graphics (`images/buttons/continueshopping.gif`).

When adding images to your page, you'll always use the same simple `` tag, regardless of the particular file format of your Web graphic. That is because the `` tag specifies the source (location) of the file, the filename, and the file format, as in the following:

```

```

What is most unusual about the `` tag is that it is one of the few tags in HTML that does not require a closing tag. That said, if you have selected the XHTML DTD, you need to close any unclosed tags like the image tag using the XHTML syntax of an extra space and slash before the right angle bracket of the tag itself:

```

```



For each of your images inserted onto a Web page, be sure that the filename and extension of each graphic in your code is spelled using all lowercase lettering. This helps you establish a uniform naming convention as well as ensure that your pages are XHTML compatible. Likewise, you must also be sure that the filename in the code exactly matches the filename as it is written in the `images` folder. Therefore, if your `images` folder has a file called `AmericanIdolJudge.jpg` and the code in your page refers to an image called `americanidoljudge.JPG`, you must rename both the file and the code so that both instances use all lowercase lettering, as in `americanidoljudge.jpg`. This tip is extremely important to remember because some servers simply will not display an image on a page if the filename is written one way in the code (`sjNav_03.GIF`) and another way in the `images` folder (`sjnav_03.gif`). Figure 1-7 shows an example of an image when the filename in the code is different from the filename of the actual file.

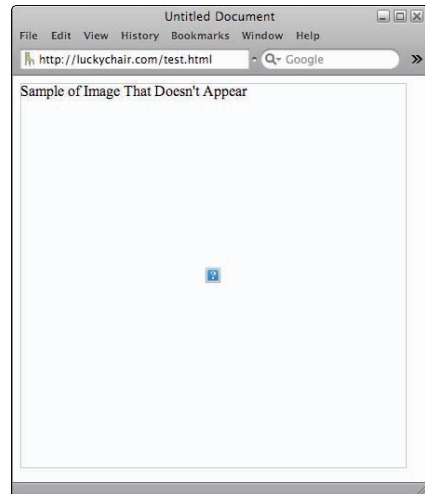


Figure 1-7: When the code and filename do not match, the image may not appear in the browser.

After your image has been inserted onto the page, you can add any of the following attributes to your code to further specify how the image should be displayed in a browser. While many of these attributes have been superseded by CSS styles, you should still know what you can do with regular HTML formatting before you veer away from it to the more robust capabilities of CSS.

Alternate text

Identify the contents of the graphic by adding a textual description to the `` tag with the alternate text (`alt`) attribute:


```

```

Size

After listing the source location and filename of the image in the image tag, include the `width` and `height` attributes of the graphic in pixels to ensure that the image displays with the correct dimensions:

```

```

Image sizes can also be noted in percentages. For instance, if you wanted to stretch out a 1-x-1-pixel image to span the entire width of the browser window or the width of a smaller area like a table cell, specify the width in percentages using the following syntax:

```

```

Border

With the `border` HTML attribute, you can apply a uniform black border to or remove any hint of a border from your images. Borders can be of any thickness by changing the number of pixels in the quotation marks:

```

```

By default, most images include a 1-pixel invisible border, like padding, around the outer edges of the graphic. This invisible border only becomes active (and visible in some browsers) when the graphic gets hyperlinked to another page, as demonstrated in Figure 1-8. If desired, you can remove this invisible border attribute by setting the attribute to "0". This ensures that the padding is removed from the outer edges of the graphic, allowing graphics that sit side by side on a page to touch each other cleanly with no hidden space between them. Setting the `border` attribute to "0" in the code looks like this:

```

```

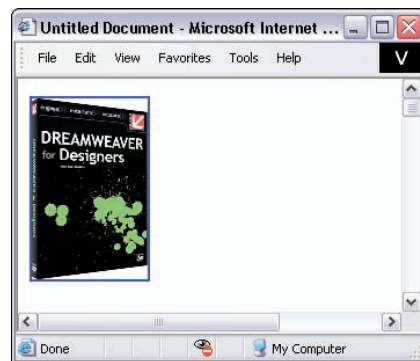


Figure 1-8: Hyperlinked graphics display a border unless you zero out the border attribute.

Using alternate text for images

While many HTML formatting attributes can be added to your images, there is one, the alternate text (`alt`) attribute, that should be added to each of your images every time, even when styling the image using only CSS. Alternate text does not appear on the Web page. Rather, it is a code tip that screen readers, other assistive devices, and search engine robots and spiders can use to identify the contents of the graphic for visitors with disabilities viewing your pages with assistive devices and to help index the content on your Web site for future visitors. As a matter of fact, some Internet browsers (like IE 6 on a PC) display the alternate text attribute as a screen tip when a visitor's mouse hovers over a graphic that has an alternate text attribute.

Here's an example of a graphic with the `alt`, `width` and `height`, and `border` attributes:

```

```

Alternate text attributes should be short, descriptive phrases, no longer than about 50 characters or so, that define the contents of the graphic, such as `alt="On Sale Now"` or `alt="Photograph of Half Dome, Yosemite, CA"`. If you find that you simply need more space to describe the contents of the graphic, you can use another tag attribute, called the long description, that links to a separate Web page that contains a longer description of the image:

```

```

In `dfddvd.html`:

Dreamweaver for Designers Training DVD:

Dreamweaver for Designers is the Telly Award–Winning ClassOnDemand self-paced, 10-hour instructional DVD that provides users familiar with design concepts but new to the Dreamweaver application with a step-by-step approach to building a professional Web site. Instructor Sue Jenkins is a Dreamweaver expert, Adobe Certified Expert/Instructor, and industry-renowned author of several Web-related instructional books. The DVD takes students through the fundamentals of Web site and application creation with Dreamweaver.

That said, there are bound to be several images on your pages that are used solely for decorative purposes and don't really need to have the alternate text attribute. For those types of images, you should still include the alternate text attribute to keep the tag HTML 4.01– and XHTML 1.0–compliant, but leave the attribute's contents blank, as in this example:

```

```

To add the blank `alt` attribute to an image in Dreamweaver, select the inserted graphic on your page in Design view and type **<enter>**, including the brackets, in the Alt text field in the Property

inspector. Or, when accessibility features are enabled for graphics, simply enter the alternate text (and a long description link, if desired) into the Image Tag Accessibility Attributes dialog box when it appears before the image gets inserted on the page.

Decorative images may include, but are not limited to, corner graphics for tables (to create a curved corner effect), horizontal and vertical dividing lines, spacer GIFs (transparent 1 x 1 pixels often stretched out to hold open empty table cells to specific sizes), bullets, and nontextual ornamental borders and graphics.

When you take care to include the alternate text attribute, whether with descriptive text or blank, you are helping to make your pages compliant with the recommended standards set forth by the W3C.

Now that you know what the `border` attribute does, other than zeroing out the border, you should never use this attribute to apply a border to your graphics (unless you absolutely have to) for three important reasons:

- ✓ The `border` attributes in HTML are limited to black.
- ✓ Using `border` attributes in HTML is old school; instead use CSS for all your styling and formatting.
- ✓ With CSS, not only can you create a border in another color besides black, but you can also apply different colors to each of the four sides of your graphic as well as specify the border's thickness and style (solid, double, grooved, dashed, and so on).

For more on CSS, refer to Book III, Chapter 3.

Image padding

Padding is the extra space around the outer edges of an image that can be used to help keep text and other objects from butting right up against it. For instance, padding can be useful when trying to separate an image from any text that wraps too closely around the image edges.



Though you'd probably do better to style any needed padding on your images by using CSS, you could still add uniform padding to your graphic using the old-school `hspace` (as in horizontal space) and `vspace` (as in vertical space) HTML tag attributes.

To add even spacing around the entire image, use the same number of pixels for both tag attributes:

```

```

The main drawback to adding padding with HTML is that the attributes uniformly apply padding to both the left and right sides of the graphic and both the top and bottom sides of the graphic, which may not cause surrounding content to wrap around the image as desired. Therefore, for best control over padding of your images, including being able to independently style each of the four sides of the graphic, use CSS.

Alignment

Before CSS came along, you could make text wrap to the left or right around an image, as shown in Figure 1-9, by adding the `align` attribute to the inside of the image tag, as follows:

```
<p>Learn Photoshop with Sue Jenkins on
      DVD. This training is perfect for the visual designer looking to make the
      leap into Photoshop's tools for print, video, and Web-based projects.</p>
```

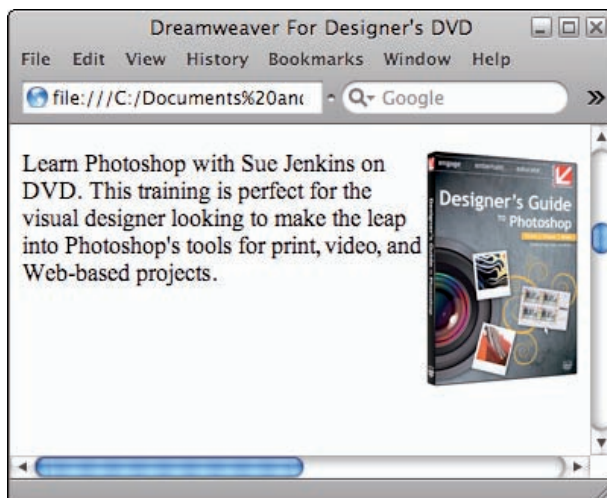


Figure 1-9: To make text wrap around an image, use the old HTML `align` attribute with the `` tag, or better yet, use CSS styling.

The location of the `align` attribute can dramatically change how content wraps around the image. For example, if you add the `align` attribute to the container tag instead of the image tag, as in the code example that follows and is illustrated in Figure 1-10, the image will be treated like another word within the paragraph:

```
<p align="right">Learn Photoshop with Sue Jenkins on DVD.
This training is perfect for the visual designer looking to make the leap
into Photoshop's tools for print, video, and Web-based projects.</p>
```



Figure 1-10: When the `align` attribute is added to the paragraph tag, images inserted between the paragraph opening and closing tags are treated the same as text.

These days, rather than apply alignment to the image or container, use CSS to position your images on the page relative to other content. To find out how to make text wrap around an image using CSS, see the section about box properties in Book III, Chapter 4.

Creating Hyperlinks

If you've ever clicked on an underlined word or button graphic and have been taken to another page on a Web site, you've experienced the magic of hyperlinks. Simply put, a *hyperlink* is any link in a file that leads you to some other file on the Web, whether that file is located on the same Web site as

the linking page or on another site on the Internet, such as clicking a link in a Google search results listing and visiting a Web site that has the information you were seeking.

Hyperlinks, or links, are typically created by pairing the `href` attribute with the `<a>` anchor tag. The `href` part is short for “hypertext reference” and defines the destination, while the anchor tag converts the contents between the opening and closing `<a>` tags into a link. Here is an example of a hyperlink that converts text into a link to another Web page:

```
<a href="about.html">Read More</a>
```

Hyperlink code is very versatile and can be used for a variety of reasons. In the following sections, you find out about local and global links, link targets, linking graphics, e-mail links, image map links, and named anchor links.

Understanding local and global links

Your hyperlinks can point to any location on the Internet; however, the destination often determines the syntax that must be used to code the link. You find two general types of links, local and global:

- ✓ **Local:** A local link is a hyperlink that points to a different page within the same Web site, such as going from the `about.html` page to the `contact.html` page.

When the linked file is local, only the filename (and relative location when other than at the root level) needs to be listed, as in the following examples:

```
<a href="services.html">Services</a>
<a href="about/mission.html">Our Mission</a>
```

Most Web sites contain primarily local links that refer to other pages within the site, as shown in Figure 1-11. The top navigation links use graphics, and the bottom set of links use regular HTML hypertext links.

- ✓ **Global:** A global link is a hyperlink that points to a page outside the current domain to some other Web site on the Internet and requires the specification of the full path of the file being linked to, such as going from the Google News Web page to an entertainment article listed on Eonline.com.

When the link is external to the site of the referring page, the full Web address, including the `http://` part and any other path information that leads visitors to the particular page, needs to be included in the hypertext reference between the quotation marks:

```
<a href="http://www.pandora.com/">Listen to Pandora Radio</a>
<a href=" http://www.classondemand.net/ClassOnDemand/adobe-
training/dw_designers.aspx">Dreamweaver For Designers</a>
```

Global links are generally reserved for search engines and pages within Web sites that list resources or other services outside their domain that may benefit visitors, such as the nonprofit resources list on the Craigslist Foundation Web site, shown in Figure 1-12.



Uses local hypertext links

Figure 1-11: Navigation links are typically all comprised of local links.

Linking targets

By default, most browsers automatically open the page or file specified in the hyperlink in the same browser window as the referring page. There is no special code you need to add to your link to make this “open in the same window” feature happen. When links open within the same browser window, we call it “targeting the same window,” because the destination browser window is typically referred to as the “target” window. To make the linked file open in a new browser window, specify the target location in the link code using the `target` attribute.

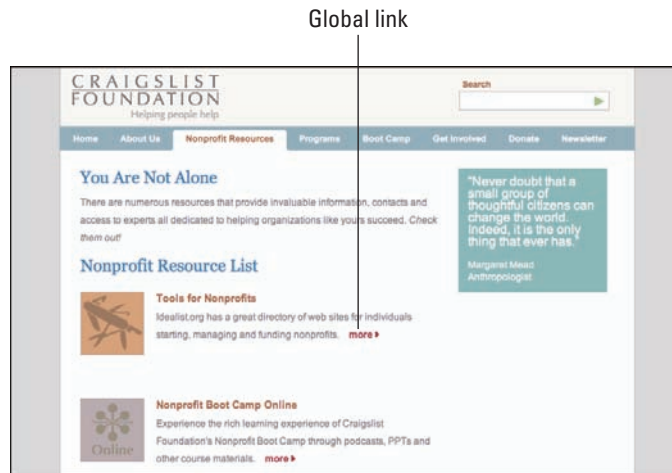


Figure 1-12: Navigation links to other Web sites are global links.

You can use four general `target` attributes to open links in different locations, depending on your needs:

- ✓ **Open link in a new window:** To make your link open a new browser window and display inside the new window, use the `"_blank"` target attribute:

```
<a href="http://www.weather.com" target="_blank">Weather</a>
```

- ✓ **Open link in the same window:** Use the target attribute `"_self"` to force the link to open in the same browser window as the referring page. For best standards compliance, you should always specify the target using the target attribute:

```
<a href="http://www.youtube.com" target="_self">YouTube</a>
```

- ✓ **Open link in a named window or frame:** When your visitors are likely to have more than one window open at the same time while viewing a particular site, you can target one of the open windows using the window's ID attribute. For example, if the site visitor has launched a pop-up window (given the ID attribute of `popup` in the code) to view a close-up image of a product and wants to see that product in another color, a link could be set to display the product in a different color in the same pop-up window, as illustrated in the Zappos.com browser windows in Figure 1-13. This feature also works with a particular frame within a frameset when the Web page has been built using frames:

```
<a href="red.html" target="popup">Red</a>
<a href="disclaimer.html" target="main">Read Disclaimer</a>
```

- ✓ (Frames only) **Open link in a window or parent:** When working with frames, you can use the `target` attribute to tell the browser where a particular page within the frame should be displayed:
 - **Top:** Use the `_top` attribute to target the entire browser window and break any preexisting frames.
 - **Parent:** Use the `_parent` attribute to target the master frame of a nested frameset.



Figure 1-13: Use the `target` attribute to tell a browser where a hyperlinked page should open.



Frames are a deprecated way of constructing Web pages and are not covered in this book. To find out more about this old-fashioned HTML technique, see www.w3schools.com/HTML/html_frames.asp.



As a courtesy, it's good to inform site visitors any time a new browser window will be opened from a link, especially when the linked file is a PDF, MS Word document, or some other non-HTML file, because some visitors might consider the launching of any new windows annoying. To help ease any possible discomfort, add a custom graphic next to the global link, such as the example shown in Figure 1-14, or insert some text in parentheses, as in (opens new window), so that visitors will know in advance that a new window will open.

Linking graphics

Linking graphics is as simple as linking text. To convert a regular image on a page into a hyperlinked image, just wrap the right bit of HTML code around the object. For example, the following code makes the `brochurerequest.gif` graphic into a clickable link that takes visitors to the `brochure.html` page:

```
<a href="brochure.html">
  </a>
```

To insert an image on your page and convert it into a hyperlink, follow these steps:

1. **Create a working folder on your desktop called Image Link Demo.**
2. **Inside this folder, create another folder called images. Inside this images folder, place a copy of one of your optimized Web graphics, which you will use for this exercise.**



If you don't have an optimized graphic ready to use for this exercise, feel free to borrow one from this book's Web page at www.dummies.com/go/webdesignaio. To save a copy of a graphic from the Web site, right-click (Windows) or Control+click (Mac) the image in the browser window. This opens the contextual menu, from which you can choose an option to save a copy of the file. After the image has been downloaded to your computer, move it into the `images` folder inside the new `Image Link Demo` folder.

3. **Using your preferred HTML editor, create a new document and save it to the Image Link Demo folder with the filename `imagelink.html`.**

Most modern HTML editors automatically insert the structural HTML tags (the bones) of the Web page for you so that you can instantly begin adding your own content to the body of the page. However, if you are hand-coding or using an older editor that does not insert the bones, add the following code by hand to your empty page:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
  Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
```

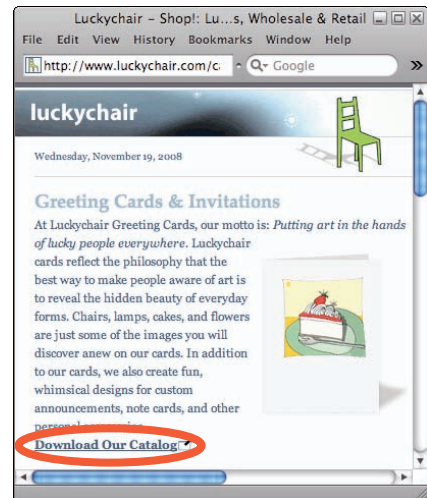


Figure 1-14: A simple illustrative graphic can alert visitors that a page or document will open in a new browser window.

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

4. Insert a copy of the image you just saved into the images folder of your Image Link Demo folder.

To do this, add the image code with the source of your image and alternate text (alt) attribute between the opening and closing <body> tags, as shown in the following example:

```

```

Be sure that the filename, file extension (.gif, .jpg, .png), and alternate text in your HTML code matches the specifics of your selected image. If you happen to know the width and height of the image, feel free to add those as attributes to the image code. For example, your code might look like this:

```

```

5. Turn the image into a hyperlink by wrapping the anchor tag and href attribute code around the image tag. For testing purposes, feel free to use Amazon for the “link to” address, as in the following code example:

```
<a href="http://www.amazon.com"></a>
```

6. With all the code now in place, save your file and launch a copy of it in a browser window.

To preview the saved file, drag and drop the HTML file by its icon into any open browser window, and the page should display, regardless of whether you have an active Internet connection. If you are using an editor like Dreamweaver, BBEdit, or HomeSite, you should be able to use a shortcut key (like F12) or press a button on the interface to launch the page in a primary browser for local testing purposes. Your page should look something like the example shown in Figure 1-15.

7. To test the accuracy of your code, click the hyperlinked graphic in your browser window. If your browser switches to Amazon, you've coded the link perfectly!

If the hyperlink didn't take you to Amazon (or the URL you inserted in the href attribute of your code), go back and check the spelling and syntax in your code. Something as small as a missing period, slash, angle bracket, quotation mark, or closing tag can make all the difference in the functionality of your page!

When the image becomes a hyperlink, the image automatically takes on a border with the default blue hyperlink color for unvisited links and the default purple hyperlink color for visited links. You can modify this attribute in one of three ways:

- ✓ Alter the thickness of the border by adding the border attribute with a value of 1 or more, as in the following:

```
<a href="http://www.amazon.com"></a>
```

- ✓ Remove the border by adding the border attribute with a value of "0", as in the following:

```
<a href="http://www.amazon.com"></a>
```

- ✓ Override the border color and style by creating a custom CSS style and applying it to the image using the class attribute, ignoring the border attribute in the code, as in this example:

```
<a href="http://www.amazon.com"></a>
```

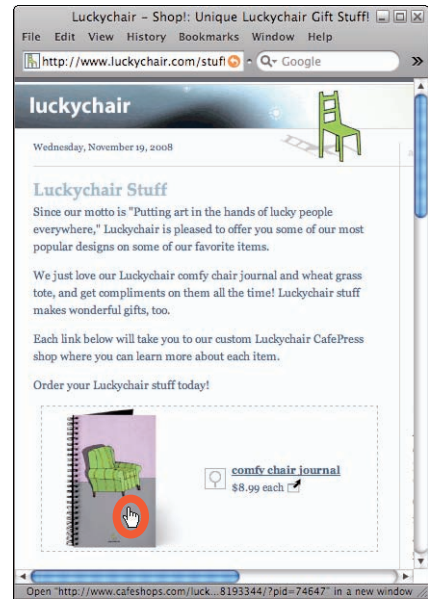


Figure 1-15: Hyperlinked images should display a hand cursor when the mouse is positioned on top of them in a browser window.

The custom border style you add to the head area of your code might be something like this:

```
<style type="text/css">
<!--
.borderstyle {
border: thin dotted #999;
padding: 10px;
}
-->
</style>
```

To preview the changes to the border of your image, save your file and relaunch your test page in a browser using one of the methods I describe in Step 6. If you added the `border` attribute to your image with a value of 1 or more, the image will appear with either a blue (default link color) or purple (default visited-link color) border, and if you added the `border` attribute with a value of "0", the default link color will be removed. If you added a CSS style to your code, your graphic might look like the example shown in Figure 1-16.



Figure 1-16: Apply a custom CSS style to override the default blue and purple hyperlink border colors.

Creating other link types

In addition to the regular old hyperlinks for text and graphics, you can create three other types of HTML links in your files: e-mail links, image map links, and named anchor links.

E-mail links

An e-mail link launches the site visitor's computer's default e-mail program and opens a blank e-mail message with the Mail To field set to the address in the e-mail link. E-mail links use the anchor tag and `href` attribute like a regular hyperlink, but also include a special `mailto:` syntax combined with the actual e-mail address that tells the browser that the link is an e-mail link, as in the following code:

```
<a href="mailto:contact@url.com">contact@url.com</a>
```

For times when you'd like to have the e-mail link automatically populate the Subject line of the outgoing e-mail message, you can do this simply by appending the e-mail address with a question mark, the word `subject`, and the actual line of text to be used as the subject line, as shown here:

```
<a href="mailto:contact@url.com?subject=Brochure Request">Request a brochure</a>
```



This is an urgent warning: HTML e-mail links are extremely vulnerable to spam-bots and other spam-harvesting tools, and therefore are a terrible way of adding an e-mail address to your Web site. What, then, should you do if you need to list an e-mail address on your Web site? Simple answer: Encrypt all your e-mail addresses by using any of the widely available encryption methods. To find out more about e-mail encryption, read the nearby sidebar, which recommends three Web sites that provide free e-mail encryption.

Protecting your e-mail addresses from spam

The Internet is an amazing place where information on nearly everything under the sun is readily available at our fingertips, including being able to communicate rapidly with our friends, families, and strangers through the revolutionary system of e-mails. Like the rest of the world, however, the Internet can also be a place for dishonesty, fraudulence, and unethical business practices. With regard to the privacy and sanctity of your e-mail addresses, this means trying to protect your e-mail address from being harvested for spam.

The main reason you get spam initially is that e-mail addresses listed on Web pages (either alone as text or especially when coded with the `<mailto:>` tag) are very vulnerable to being harvested by human-initiated spam-bots, which are e-mail-gathering applications that operate 24 hours a day in search of innocent e-mail addresses. And because the creators and users of these e-mail harvesters have no ethics, they're likely to sell your e-mail address, when harvested, to other companies that also profit from these illicit spam lists. Nice, huh?

To combat this rampant Internet crime, the best defense is to encrypt *all* e-mail addresses appearing on every site you create. Protect the e-mail addresses that you display on a Web page, first and foremost, by *not* listing them as text or hyperlinked e-mails. Instead, take a few extra minutes to encrypt or otherwise hide the e-mail addresses from those malicious spam-bots and their creators.

Here are two fine recommendations for encrypting and hiding e-mail addresses:

- ✓ **Encrypt with a JavaScript:** Several freeware and shareware versions of encryption tools are available online, including the following:

DynamicDrive: www.dynamicdrive.com/emailriddler

csarven.ca: <http://csarven.ca/hiding-email-addresses>

Address Munger: www.addressmunger.com/contact_form_generator

Dan Benjamin's Hivelogic Encoder Form: <http://hivelogic.com/enkoder/form>

✓ **Encode using entities:** When typing a regular e-mail address on a Web page, use code entities for the special characters instead of regular text, or for the entire e-mail address. For instance, because the entity for @ is %40 and the entity for . is %2, the code for `contact@myfavoriteshoesonline.com` would change to `contact%40myfavoriteshoesonline%2com`. To convert any e-mail address into URL unicode, visit the W3Schools Web site: www.w3schools.com/tags/ref_urlencode.asp.

With most JavaScript encryption applications, you usually type your e-mail address, press a button, and get some code back that you can copy and paste into your Web page. The code contains the encrypted e-mail address and, when viewed on a Web page, looks just like a regular hypertext e-mail address.

For example, if you used the encryption application on Dan Benjamin's Web site, you'd paste the JavaScript code returned from the Encoder Form into the place in your page's HTML code where you would like the e-mail address to appear. To illustrate, the encrypted e-mail address you might use to replace the following regular e-mail link

```
<a href="mailto:contact@myfavoriteshoesonline.com">
    contact@myfavoriteshoesonline.com</a>
```

might look like this:

```
<script type="text/javascript">
/*  */
function hivelogic_ekoder(){var kode=
"kode=\"nrgh%{@hgrn\\000,f+hgrFudkFprui1jq1uwV@.&gt;;54@.f,3?f+i1&gt;60,l+wDhgr"+
"Fudkf1hgrn@f~,..l&gt;kwjqho1hgrn?l&gt;3@l+uri&gt;**@{&gt;_/_/--.toup4/.kxk|kx4/--.zorv"+
"y4kjuqCkjuqA(qujkC(buqkjbb(bC-/.-otpu/4k.xy|kxk/4--z.royvk4ujCqjkqubbbaJb"+
"b(bius{tk4zx)zo.kbb(bbbbbbbbbbbbgBn&amp;kxClbbbbbbbbbbbbbbbbbb(bbbbbbbbbbbbg"+
"srouzi@tugzzisF1\\177|gxuzoykunyktuorkti4subbbb&amp;bbbbbbbbbbbbbbbbbb(b&amp;b"+
"bbbozrzCkbbbbbbbbbbbbbbbbbb(bbbbbbbbbbbbuiztigFz\\177sglu|oxkznykuuyrtto4"+
"kui&amp;sbbsbbbbbbbbbbbbbb(bbbbbbbbbbbbiDtugzzisF1\\177|gxuzoykunyktuorkti4s"+
"uB&amp;g5bbbbDbbbbb(b/bbbbbbbACbb(bjkquAbb(buqkjCju4kvyor.z--4/kxk|yx.k4/up"+
"to-./-(bA~C--Alux.oC6AoB.qujk4rktmzn37/Ao1C8/\\001~1Cqujk4ingxGz.o17/1qujk"+
"4ingxGz.o/333__qujkC-1.oBqujk4rktmznEjujk4ingxGz.qujk4rktmzn37/@-/-A(Ckjuq"+
"_%@hgrn%&gt;nrgh@nrghlvsolw+**,luhyhuvh+,lmlq+**,\\";x='';for(i=0;i&lt;kode.leng"+
"th;i++){c=kode.charCodeAt(i)-3;if(c&lt;0)c+=128;x+=String.fromCharCode(c)}kod"+
"e=x"
;var i,c,x;while(eval(kode));}hivelogic_ekoder();
/* ]]&gt; */
&lt;/script&gt;</pre>
</div>
<div data-bbox="154 762 846 830" data-label="Text">
<p>The only potential drawback to encrypted e-mail addresses is that visitors who have JavaScript enabled will not be able to use these links. To prevent that from happening, be sure to add <code>&lt;noscript&gt;</code> tags to your code, directly following the script tags, that include alternate contact information, as in the following example:</p>
</div>
<div data-bbox="165 837 486 882" data-label="Code-Block">
<pre>&lt;noscript&gt;
Send email to info ~at~ company ~dot~ com
&lt;/noscript&gt;</pre>
</div>
<div data-bbox="895 513 960 543" data-label="Page-Header">
<p>Book III<br/>Chapter 1</p>
</div>
<div data-bbox="915 561 960 668" data-label="Page-Header">
<p>Adding Text,<br/>Images, and Links</p>
</div>
```

Image map links

An image map is a graphic that has one or more hotspots defined on it with coordinates in the HTML code that make those hotspots clickable to another page on the Internet like regular hyperlinks. You can also attach behaviors to hotspots to call JavaScript functions, such as displaying another image when the hotspot is moused over or clicked, or creating a link that launches another browser window.

Images can have as many image map hotspots as you need, as long as each hotspot coordinate is specified in the code. For example, a software training school Web site might have a group photograph of all the in-class instructors who work there. Each instructor in the photograph could then have an associated hotspot that, when clicked, transfers visitors to an individual instructor bio page. (For a visual example of this, see www.nobledesktop.com.)

Image map coordinates are fairly time consuming to create when hand-coding but are super-easy to make using a WYSIWYG coding program like Dreamweaver. In Dreamweaver, you can use the Rectangular, Circle, or Polygon Hotspot tool to draw one or more precise hotspot shapes on an image. After the hotspot coordinates are defined in the code, the hyperlink to the desired Web page can then be specified along with a target destination for the link to control where the linked page will open. The following HTML code shows an example of how a rectangular hotspot can be mapped onto a graphic:

```

<map name="amazon" id="amazon">
<area shape="rect" coords="0,0,100,156" href="http://www.amazon.com/"
      target="_blank" alt="Order today from Amazon.com">
</map>
```

In Dreamweaver, the hotspot area is displayed in Design view by a see-through light blue shape, as demonstrated in Figure 1-17.

Named anchor links

These special hyperlinks, which are sometimes referred to simply as *anchor links*, help visitors jump from one part of a Web page to another part of that same page when clicked. Anchor links are particularly useful as quick links to sections referred to in a table of contents or the sequence of answers in a list of FAQs. You can also use named anchor links to jump visitors back to the top of the page after long passages of text.

Named anchor links require two parts in the HTML code to function correctly: the linked text or graphic that points to the destination elsewhere on the page, and the anchor, which is located at the destination.

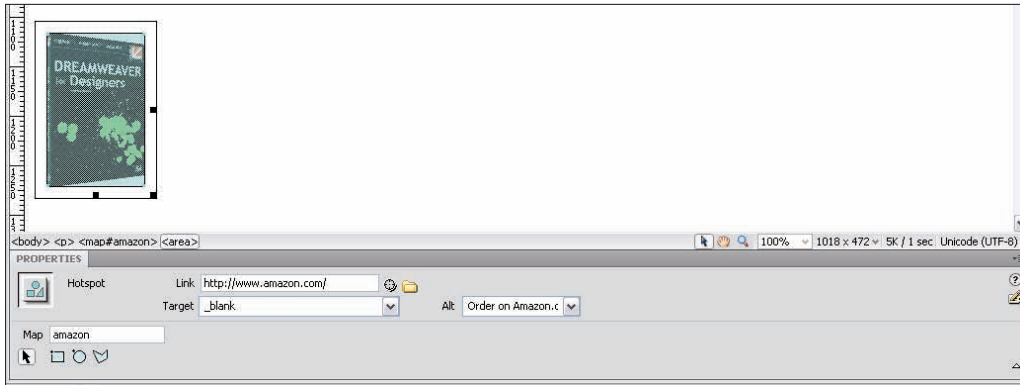


Figure 1-17: Image map hotspots are a snap to create in Dreamweaver.

Because all anchors need to be given their own specific name for the anchor link to function correctly, try to name each of your anchors after its functionality or purpose. For example, if your table of contents is listed numerically, you could name the anchor by its section, such as 23179 or faq8. Anchor names may include any combination of numbers and letters but cannot contain any spaces or funny characters (such as / or *). For instance, to make a link at the bottom of a long page take visitors, when clicked, back to the top of the page, you'd need to insert the link that refers to the anchor at the bottom of the page and the anchor itself near the top:

- ✓ **Link:** `Top`
- ✓ **Anchor:** ``

To better help you digest the concept, the following steps demonstrate how to set up your page with a few named anchor links.

- 1. Open your favorite HTML editor, create a new document, and save it with the filename `anchors.html` into the Image Link Demo folder you created in the last exercise.**

If you have already deleted that folder, just create a new one.

- 2. Inside the `anchors.html` file, type the following content between the opening and closing `<body>` tags:**

```
Table of Contents:

Document Setup
Using Fonts
Text Flow
```

Document Setup: Define your page size as the final trim size of your document. To create a bleed, extend your page elements .125 (one-eighth) inch off the document's edge.

Using Fonts: Computers use screen fonts to display your type on the monitor and printer fonts to send to the laser printer or for printing.

Text Flow: Because of kerning, tracking, hyphenation, and "target printer" preferences specific to your software, your text may reflow if you ask us to make type changes.

3. To insert the first named anchor, place your cursor in the code right before the words Document setup: Define . . . and type

```
<a name="docsetup"></a>
```

If you're using a code editor, the program may have a way to quickly insert the named anchor for you rather than hand-coding it. Dreamweaver, for instance, has a Named Anchor button in the Common area of the Insert panel that is shaped like a golden anchor. When clicked, this button quickly inserts the appropriate code for the named anchor, after providing the program with the anchor name, in the spot where your cursor is located on the page.

The Document Setup section of your code should now look like this:

```
<p><a name="docsetup"></a>Document Setup: Define your page size as the  
final trim size of your document. To create a bleed, extend your  
page elements .125 (one-eighth) inch off the document's edge.</p>
```



As you can see, the named anchor tags do not surround any content, but rather mark the territory so that an anchor link can jump to this destination. In other words, you do not want to create a regular hyperlink here from surrounding content; you merely want to create a destination hotspot on the page.

4. Create the link to the new named anchor by inserting a regular hyperlink around the text that will link to the anchor. To do this, rather than adding a filename to the href attribute, you add a number symbol (#) along with the name of the anchor.

Table of Contents:

```
<a href="#docsetup">Document Setup</a>  
Using Fonts  
Text Flow
```

5. To test your new named anchor, save the file and launch the page in a browser window. The words `Document Setup` in the Table of Contents area should appear in the browser as an underlined hyperlink that, when clicked, should jump the page to the Document Setup section farther down on the page.



To test the functionality of the named anchor link in a way that makes more visual sense so that you can see the jump occur, reduce the dimensions of your browser window so that the Document Setup text area is hidden from your view before you click the link.

6. Repeat Steps 3 and 4 for the “Using Fonts” and “Text Flow” entries.

As long as each named anchor has its own unique name, you can have as many of them on the page as you like.

In addition to using named anchor links to jump from one spot to another on the same page, you can also use named anchors to link to a specific spot on another page. For instance, you may want site visitors to be able to click a hyperlink in one page and be taken to the named anchor destination in the middle of another page. To make that happen, just append the regular filename in the `href` with the number symbol and the name of the desired anchor, as shown in the following example:

```
<a href="schedule.html#danceworkshop">Sign up for the next Dance Workshop</a>
```

Now that you see how easy it is to create named anchor links, you might never create a long page without them!

Labeling Content for CSS Markup

Before you get into the nitty-gritty of CSS coding in Book III, Chapter 3, it would be a good idea for you to get into the habit now of *labeling*, or assigning a name to, all the objects in your pages using the `id` attribute. This is especially important to do for any objects you plan to style with CSS and/or make dynamic with the use of JavaScript code.



The W3C (World Wide Web Consortium, w3.org) refers to the objects on your page as *elements*, which really just means anything, including text surrounded by paragraph or header tags, that can be added to a Web page.

To label an element, add the `id` attribute with a unique name to the element’s opening tag. If you take this extra step as you insert each object onto the page, you will save yourself some time later on when you’re ready to style and position your content with CSS. To add the label, the proper syntax is `id="name"`.

When adding the `id` attribute to your elements, keep the following general usage rules in mind:

- ✓ **No two objects can use the same `id` value in a single HTML file.**
- ✓ **The `id` is case sensitive and must begin with a letter.** It may, however, contain any combination of letters and numbers as well as periods, underscores, hyphens, or colons.
- ✓ **For ease of use and semantic integrity, name your `ids` after their purpose as much as possible.** Although technically, you can call `ids` whatever you want, a descriptive name, such as `sidebar1` or `sale_items`, will be more helpful than calling something `bigBlueHeader` or `LeftPad20px`, because certain attributes (like the `Blue` or `20px` part) may change as the site is being built.

In addition to using the `id` attribute as CSS selectors and script elements, `ids` can also be used as anchors for hypertext links, names of objects, and identifiers for other applications that might be parsing data from your pages, such as when a script extracts data from a form field into a database.

The `id` attribute can also be added to text blocks, tables, images, `<div>` tags, plug-ins, media files, form fields, and any other objects or elements you plan to style with CSS and/or make dynamic with JavaScript or other programming languages. The following examples show how and where to add the `id` labels to a few different tags:

```
<table width="500" cellspacing="0" id="productdetails">
<div id="sidebar2">
<p id="highlight">Get 'em while they're hot!</p>
```

In some cases, depending on the browser, the `id` attribute is used to replace the old `name` attribute, while in other instances, the `name` attribute is still required to display that object correctly. The situation can get even more confusing when you begin dealing with form fields because the `id` and `name` attributes function differently, depending on which browser the visitor is using to view the page with! Therefore, to avoid mass confusion and reduce your troubleshooting time, you may want, in certain circumstances, to include both attributes within particular tags, as with images and form fields in the following code examples:

```

<input type="text" name="textfield" id="textfield">
<frame src="nav.html" name="navigationFrame" id="navigationFrame" title="
navigationFrame">
```

Of course you're not expected to know which ones need both and when. Instead, you can simply code with both attributes, or better yet, when using a code editor like Dreamweaver, the program will automatically know which tags to add both attributes to!

Making Content Accessible

In earlier chapters, you find out about some of the must-have site accessibility features, such as HTML footer links, a site map page with simple hyper-text links to all the pages on the site, alternate text attributes for all your images, unique page titles for every page, and the use of description and other optional meta tags in the head of the page to help visitors find your site through search engines. You can further improve page accessibility by including additional accessibility tags and tag attributes in the code.

Though some designers might prefer to first get all their content on the page and then go back and improve it for accessibility, you may find that it is more efficient to add the accessibility features to your code during the page-building process. When you add accessibility as you code, you're assured of including all the different tag attributes and other accessibility enhancements to each element, rather than having to second-guess and do double the work by going back and checking everything at a later time to see whether you remembered or forgot to add them.

Table 1-2 lists the most common accessibility coding enhancements. Use them as often as possible on all your Web pages.

Table 1-2 Accessibility Improvement Tags		
Name	Description	Sample Usage
Alternate text attribute in an image tag	Provides descriptive text for images to visitors using assistive devices to access the Web, as well as non-human visitors like search engine robots and spiders.	<code></code>
Title tag	A unique descriptive page title can be applied to each page on a site. Title tags are scanned by search engine robots and can help improve search engine listings and rankings when peppered with site-relevant descriptive keywords.	<code><title>James Graham : Oil portraits of animals and pets by New York painter.</title></code>

continued

Table 1-2 (continued)

<i>Name</i>	<i>Description</i>	<i>Sample Usage</i>
Object labels	Use the <code>id</code> attribute to label objects on the page that will be styled with CSS or made dynamic with JavaScript. When labeling images for JavaScript, be sure to also include an identical <code>name</code> attribute.	<pre></pre> <p>or</p> <pre></pre>
Long description	The alternative text (<code>alt</code>) attribute can only handle about 70 characters max of text. To provide a longer description for any image, add the <code>longdesc</code> attribute to the image that links to a separate Web page that contains the longer textual description.	<pre></pre>
Title attribute for hyperlinks	Add the <code>title</code> attribute to all hyperlink tags. Title attributes help search engines index pages, provide pop-up screen tips in some browsers, and aid visitors using assistive devices.	<pre>Download the 1040 Form (PDF)</pre>
Table title attributes	Similar in function to <code>alt</code> text for an image, the <code>title</code> attribute goes in the opening table tag and provides a title and description for the contents of a table.	<pre><table width="400" border="1" cellspacing="0" id="earnings" title="Earnings" summary="Projected 3rd Quarter Earnings"></pre>
Link tags in the <code><head></code> of the page that point back to the home page and site map page	These links help improve accessibility of these all-important pages to visitors using assisted viewing devices. In addition, they assist search engine robots/spiders in more readily indexing pages in a database. The <code>rel</code> part of the link defines the relationship between the current page and the page being linked to.	<pre><link rel="Index" href="index.html"></pre> <pre><link rel="Site Map" href="sitemap.html"></pre>

<i>Name</i>	<i>Description</i>	<i>Sample Usage</i>
Footer links	Add navigational text hyperlinks at the foot of each page to mirror any graphic-only links to the main pages on a site. Footer links may also include links to other important pages such as privacy policy or terms of service pages.	<p>Home - About Us - Contact</p> <p>Each word in the footer should be hyperlinked to its respective page using the standard hyperlink anchor tag with href attribute, as in</p> <pre>About Us</pre>
A site map page	This page includes hyperlinks to all the accessible pages on a site, typically in list format with any sub-pages listed under the relevant main category page listing.	<p>Home About Us Company History Board of Directors Our Mission Contact</p> <p>Each word or phrase is hyperlinked to its respective page, as in:</p> <pre>Board of Directors</pre>
Contrasting foreground and background colors	One of the most common disabilities for visitors on the Internet is color blindness. To assist these visitors in having a positive experience on your Web site, use colors with a strong enough color contrast that the difference can be easily detected.	<p>Check out the accessibility color wheel by Giacomo Mazzocato: http://gmazzocato.altervista.org/colorwheel/wheel.php.</p> <p>Use this tool to assist you with choosing color combinations that can improve page readability for those with partial or full color blindness.</p>
Hyperlink targets	Add the target attribute to your hyperlinks so that the browser knows where you want the linked page to load, whether that's in the same or a new browser window.	<pre>Home</pre>
Access keys	Add keyboard shortcuts with the accesskey attribute that visitors can use (typically in combination with the Alt or Option key, as in Alt+C) to quickly move their cursor to links, form fields, and other accessible objects on a page.	<pre>Home</pre>

continued

Table 1-2 (continued)

<i>Name</i>	<i>Description</i>	<i>Sample Usage</i>
Tab index	Often used in conjunction with the <code>accesskey</code> attribute, the <code>tabindex</code> attribute allows visitors to use the Tab key to advance from one link, form field, or other accessible object on a page to the next. If desired, the order of the tab index need not follow a direct top-to-bottom, left-to-right progression.	<pre>Home</pre>
Form input tag labels	When the label tag is used in conjunction with a label and form field, visitors can click anywhere in the vicinity of the radio button, checkbox, or label text to select that option.	<pre><label for="checkbox"> Tea</label> <input type="checkbox" id="tea" name="tea" value="tea" accesskey= "t" tabindex="10"></pre>

Chapter 2: Organizing Content with Tables and Lists

In This Chapter

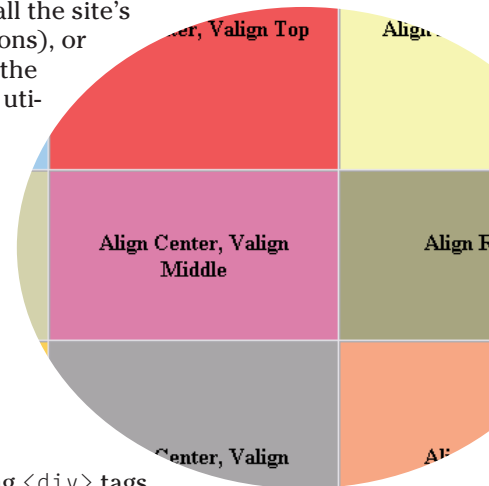
- ✓ Using tables to organize content
- ✓ Adding content to tables
- ✓ Formatting tables and table cells
- ✓ Nesting tables
- ✓ Organizing content with lists
- ✓ Setting the list type
- ✓ Nesting lists

When it comes to keeping your content organized so that visitors can quickly find what they are looking for, consider marking up your content with HTML tables and lists. Tables are useful for displaying tabular data and information with multiple categories, such as financial figures, store locations, and menu offerings. Lists can be a great resource for times when you need to show things in sequence, such as how a set of items are structured relative to one another (as with links to all the site's pages on a site map page or a set of navigation buttons), or for times when you need your content to display in the order in which a set of items should be accessed or utilized (as with items in a numbered list).

In this chapter, you find out how easy it is to mark up basic content into table and list format. In addition, you read about adding and formatting content in tables, nesting tables for complex page elements, organizing and structuring list content, and list nesting to create multitiered lists.

Inserting Tables on a Page

While most modern page layouts rely on layers using `<div>` tags to organize content within the browser instead of tables, which was once the norm, tables themselves are still very useful within the body of the page for organizing data and other content that requires gridlike arrangement. In



the following sections, you find out more about the basics of organizing your content in tables.

Discovering what you can do with tables

Tables are the perfect way to organize multirow and multicolumn data and other content within regions of a Web page, particularly because anything that can go on a page can be placed inside a table cell. Furthermore, in a table, you can control the alignment, width, and height of the table cells, as well as style the contents of those table cells with CSS, providing you with a whole new level of control over the contents of the table.

Tables can have any number of rows and columns, be any width and height, have any colored border, and have any background color or tiling background image. What's more, the cells of the tables can also have unique widths, heights, and background colors and/or background images that sit on top of whatever styling attributes happen to be applied to the table.

One of the only real drawbacks to working with tables is that they require a lot of code to do the job they do, which can increase the overall size of the HTML file. For example, the table in Figure 2-1 looks fairly straightforward, but on the code level, it looks like this:

Winter 2009 Schedule	MON	TUES	WED	THURS
<i>First Class</i>	1/5/09	1/6/09	1/7/09	1/8/09
<i>No Class</i>	2/16/09	2/17/09	2/18/09	2/19/09
<i>Last Class</i>	3/30/09	3/31/09	4/1/09	4/2/09

Figure 2-1: Tables require a lot of HTML coding to display content in an orderly manner.

```
<table id="schedule">
  <tr>
    <td class="toprow">Winter 2009 Schedule</td>
    <td class="toprow">MON</td>
    <td class="toprow">TUES</td>
    <td class="toprow">WED</td>
    <td class="toprow">THURS</td>
  </tr>
  <tr>
    <td class="cellstyle">First Class</td>
    <td class="cellstyle">1/5/09</td>
    <td class="cellstyle">1/6/09</td>
    <td class="cellstyle">1/7/09</td>
    <td class="cellstyle">1/8/09</td>
  </tr>
```

```

<tr>
  <td class="cellstyle">No Class</td>
  <td class="cellstyle">2/16/09</td>
  <td class="cellstyle">2/17/09</td>
  <td class="cellstyle">2/18/09</td>
  <td class="cellstyle">2/19/09</td>
</tr>
<tr>
  <td class="cellstyle">Last Class</td>
  <td class="cellstyle">3/30/09</td>
  <td class="cellstyle">3/31/09</td>
  <td class="cellstyle">4/1/09</td>
  <td class="cellstyle">4/2/09</td>
</tr>
</table>
    
```

Understanding the structure of a table

To display correctly in a browser, each table requires several components.

The first part is the opening and closing `<table>` tags, which define the table's beginning and end within the HTML. The opening `<table>` tag is where you can add attributes to the code (for things like setting the width of the table) and apply any special formatting to the table using CSS styles, as in the following example:

```

<table width="300" id="menu">
</table>
    
```

In between the table tags come the opening and closing table row tags, `<tr>` and `</tr>`, which define a row within the table. To have more than one row, insert another pair of table row tags, as in the following example:

```

<table width="300">
<tr></tr>
<tr></tr>
</table>
    
```

To define individual table cells, which make up the columns across each row in the table, insert a pair of table data tags, `<td>` and `</td>`, between the table row tags. To have more than one column in a row, insert another pair of table data tags between the table row tags, as in the following example:

```

<table width="300">
<tr>
  <td></td>
  <td></td>
</tr>
<tr>
  <td></td>
  <td></td>
</tr>
</table>
    
```



To hold a table cell open so that it doesn't collapse when no content is inside of it, insert the “nonbreaking space” code entity ` ` into the code between the opening and closing table cell tags:

```
<td>&nbsp;</td>
```

The minimum number of rows and columns you can have in a table is one each, which creates a one-celled table using a single pair of `<tr>` and `<td>` tags:

```
<table width="300">
<tr>
  <td>&nbsp;</td>
</tr>
</table>
```

After you set up the basic table structure of your table using `<table>`, `<tr>`, and `<td>` tags, you can specify the desired number of rows and columns, insert content in the cells as needed, and style the table, table cells, and cell content using CSS.

That said, you can also still use a number of HTML table and table cell attributes to alter the structure of the table in a manner that meets your needs and ensures that the table will appear as you envisioned in the browser. For instance, you can merge and split table cells to create unusual table layouts and have content span across a series of table cells both vertically and horizontally.

Fortunately, with the advent of HTML editors, coding tables by hand is no longer a necessary evil. Most code editors should allow you to quickly and painlessly split and merge cells, set table and cell widths, and add color to borders and backgrounds with just a few clicks. Nonetheless, even though you'll likely use a code editor to build your tables for you, you should still understand the underlying structure of tables so that you can easily manipulate them at the code level if needed.

The following sections walk you through the steps of adding content to your tables, formatting the table and table cells using HTML attributes, and nesting tables for more complex content organization. As for styling the table and table content with CSS styles, Book III, Chapters 3 and 4 tell you more about working with CSS.

Adding content to table cells

As I mentioned previously, any content that can go elsewhere on the Web page can be placed inside a table cell. This means you can add text, graphics, Flash movies, animations, Shockwave movies, applets, plug-ins, audio files, video files, e-mail addresses, JavaScript, hyperlinks, named anchor links, layers, form fields, Spry widgets, tabular data, dynamic data, blog

data, PayPal buttons, shopping cart elements, and even hidden elements that reside in the code but do not appear on the Web page. Figure 2-2 shows an example of two tables stacked on top of each other, one with a background image and HTML text, and the next with one row and three columns, each cell displaying a linkable graphic and hypertext link to detail pages about each product.

1-celled table with HTML text and background image



Table with 1 row and 3 columns

Figure 2-2: Table cells can hold any content that can appear elsewhere on a Web page, such as text, graphics, links, animations, and videos.

To add content to a table, place your cursor inside any table cell (between any pair of opening and closing `<td>` tags, if you're hand-coding) and simply add the desired content. If you're using an HTML editor, you will likely have special tools, buttons, or menu options within the program that you can use to add the content more quickly and efficiently. For instance, in Dreamweaver, you can use the Insert menu commands or any of the buttons in the Insert panel to quickly add links, graphics, images, and media files. After the data is inserted, you can style and format the table and table content with CSS and continue adding data in the other table cells.

Formatting Tables

When styling your tables and the content within them, you should do as much as you can with Cascading Style Sheets. Of course, within the code of your tables, you can still set table and cell widths, table and cell alignment, cell padding, and cell spacing, but try to use CSS exclusively to specify any other table and cell attributes that have to do with how the table looks. This will provide you with greater control of the presentation of the table content as well as make the content more easily editable, because all the site's styling will then be contained in the CSS rather than on the individual pages of the site.

The id attribute

Anytime you intend to style a part of your table with CSS — whether it's the table background, table border, cell background, cell alignment, or any of the other attributes discussed in the following sections — be sure to add an `id` attribute to each of your tables in the opening `<table>` tag. This helps make your tables easier to identify from one another as well as allows you to quickly apply both CSS and JavaScript to them as needed:

```
<table width="400" border="1" cellspacing="10"
      cellpadding="0" id="storehours">
```

The `id` attribute can also be added to individual `<td>` tags as well:

```
<table id="offices">
<tr>
  <td id="locations">Atlanta</td>
</tr>
</table>
```

Table widths and heights

Each of your tables can be set to have specific widths and heights within the page. Those attributes can be specified in the code in the opening `<table>` tag. However, while it was once a common practice to specify the height in the table, the `height` attribute has been deprecated, so if you need to set a height, you must now do that in the CSS.

However, the width can be set in either the CSS (which is best) or within the HTML as an attribute of the opening table tag. Table widths are most often notated in the code in pixels or in a percentage that is relative to the size of the viewing browser window (or other container tag, such as a layer or table). For example, as illustrated in Figure 2-3, a table's width can be fixed at 450 pixels or be set to a width that is equal to 70 percent of the browser window:

```
<table width="70%">
<table width="450">
```

Anytime a percentage is specified in the code (or in the CSS), that number will maintain the same aspect ratio to the browser as the user increases or decreases the size of the browser window on his computer's desktop.

On the other hand, should the width of the table not be specified in the opening table tag or in the CSS, the size of all the table cells within the table will collapse to fit the contents inside them, whatever those contents might be. This means that the contents with the largest width and height in any one cell will determine the width of the entire column and/or the height of an entire row.

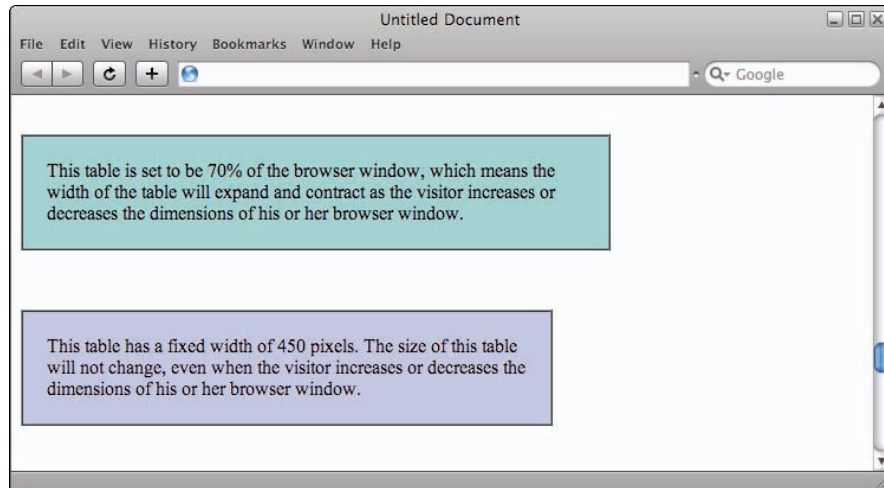


Figure 2-3: Table widths can be specified in fixed pixel dimensions or as a percentage relative to the size of the browser window or surrounding container tags.

However, when you set the size of the table cell, the contents of a cell will have no effect on the width and height of its respective column or row. The only exception to this is when the contents of the cell exceed the preset cell size. Typically, to set the width of an entire column, you only need to add the `width` attribute to the first cell in the first row; all other cells in the column beneath that cell automatically take on the same dimensions. Here's what the code for a 500-pixel-wide, 2-row, 2-column table with a 1-pixel border and set cell widths looks like:

```
<table width="500" border="1">
  <tr>
    <td width="300">Select Item</td>
    <td width="100">Price</td>
  </tr>
  <tr>
    <td>#TA57694</td>
    <td>$150.00</td>
  </tr>
</table>
```

Of course, a better method for setting the width of the table and cells is to use CSS, in which case your code might look more like the following:

```
<table id="products">
  <tr>
    <td>Select Item</td>
    <td>Price</td>
  </tr>
```

```

<tr>
  <td>#TA57694</td>
  <td>$150.00</td>
</tr>
</table>

```

Table and cell alignment

A table can be aligned within the browser window or within another container, while table cells can align the content within them both vertically and horizontally. With table alignment, you can either use CSS for the greatest control or resort to using the old `align` attribute — with a value of `left`, `right`, or `center` — in the opening table tag, as in the following:

```

<table width="300" align="center" cellpadding="2"
  cellspacing="0">

```



Unfortunately, because this attribute has been deprecated, you might find that adding the `align` attribute to the `<table>` tag produces uneven results in certain browsers.

To resolve that issue, simply create a custom CSS style such as the following:

```

.tablealign {
  margin-right: auto;
  margin-left: auto;
}

```

When applied to your table using the `class` attribute in the opening table tag, this style will horizontally align the table within the browser window:

```

<table class="tablealign">
  <tr>
    <td>Select Item</td>
    <td>Price</td>
  </tr>
  <tr>
    <td>#TA57694</td>
    <td>$150.00</td>
  </tr>
</table>

```

This latter method is more widely supported by different browsers on both Mac and PC platforms, and thus is more likely to accurately display your content with the desired alignment.

To set the alignment of the contents inside any table cell, you could apply the `align` (horizontal alignment) and `valign` (vertical alignment) attributes in the opening `<td>` tag of the cell that requires alignment:

```
<td width="200" rowspan="2" align="left"
    valign="top">Kittens</td>
```

Horizontal alignment options are left, center, and right, and vertical alignment options are top, middle, bottom, and baseline (which aligns the image bottom to the baseline of text within the table cell and at times looks no different than the bottom attribute).



When using CSS to set the alignment of content in a table cell, there's good news and not-so-good news. The good news is that you can replace the regular HTML `align` attribute with the CSS `text-align` attribute to create left-, right-, center-, and justify-aligned content within a table cell. The not-so-good news is that no simple CSS equivalent exists for vertical alignment within a table cell. As such, until something better is created for the next version of CSS, you may continue using the `align` and `valign` attributes for your table cell alignment needs.

Figure 2-4 shows an example of the variety of cell alignment options you can create using the `align` and `valign` attributes for your table cell content.

Align Left, Valign Top	Align Center, Valign Top	Align Right, Valign Top
Align Left, Valign Middle	Align Center, Valign Middle	Align Right, Valign Middle
Align Left, Valign Bottom	Align Center, Valign Bottom	Align Right, Valign Bottom

Figure 2-4: Use the `align` and `valign` attributes to align content in your table cells.

Table borders

When adding a border to a table with the old HTML `border` attribute, the size you specify in the code refers only to the thickness of the table's outer edge. Any border width larger than 1 pixel creates a beveled table edge, as shown in Figure 2-5.



If you hate the bevel (which is very old school and can make your table look awkwardly retro), you can create a non-beveled outer table border of any thickness and color, like the one shown in Figure 2-6, by using CSS. See Book III, Chapter 4 for details on border formatting with CSS.

Likewise, because all tables by default have 1 pixel of cellpadding and cellspacing (even when these attributes are not specified in the code), when you apply a 1-pixel border to a table with the HTML border attribute, the border looks like a double line instead of a solid line, as shown in Figure 2-7.

To remove this default spacing and create a solid 1-pixel border with the HTML border attribute, include both cellpadding and cellspacing attributes in the opening <table> tag. Set the cellpadding to the desired amount and set the cellspacing to 0:

```
<table width="600" border="1"
  cellpadding="10"
  cellspacing="0">
  <tr>
    <td width="300">Google</td>
    <td width="300">Yahoo</td>
  </tr>
</table>
```

Better yet, zero-out both the cellpadding and cellspacing in the HTML and apply a custom border and padding to the table with CSS.

Cellpadding and cellspacing attributes

Because content can sometimes look too cramped within a table cell, you can apply two special HTML table attributes to your table to give the content a little more breathing room. These two attributes, cellpadding and cellspacing, uniformly apply extra space to all the cells and cell walls, respectively, within a table:

Lunch Menu:		
Lite Fare	Garden Salad	\$8.00
	Soup	\$4.00

Figure 2-5: Adding a border larger than 1 pixel on a table using HTML code produces a border with a beveled edge.

Lunch Menu:		
Lite Fare	Garden Salad	\$8.00
	Soup	\$4.00

Figure 2-6: For more control over your table borders, use CSS.

Today's Menu		
Lite Fare:	Garden Salad	\$8.00
	Soup	\$4.00

Figure 2-7: Tables with a 1-pixel border and no attributes specified for cellpadding and cellspacing display a double-lined border effect.

- ✓ **cellpadding:** This is the pixel space that can be adjusted between the contents of the table cells and the cell walls. The larger the number of pixels, the more padding is added between the cell walls and the cell contents. Think of this as if you are moving your furniture away from the walls of the room so that you can more freely walk the perimeter of the room unencumbered.
- ✓ **cellspacing:** This is the pixel space, or thickness, within the cell walls between the cells. You can think of this as having thick or thin walls between your bedroom and your pretend-brother's; the thicker the walls, the less his loud music will annoy you.

All tables by default have 1 pixel of cellpadding and cellspacing, even when these attributes are not specified in the code. Therefore, to remove this default spacing, you must “zero out” these attributes in the code, as follows:

```
<table width="200" border="1"
  cellpadding="0"
  cellspacing="0">
  <tr>
    <td width="300">Peanuts
    </td>
    <td width="100">Popcorn
    </td>
  </tr>
</table>
```

You can use the cellpadding and cellspacing attributes alone or together to achieve the desired results. For example, the four tables shown in Figure 2-8 all have 1-pixel borders, but the first has cellpadding and cellspacing set to 0, the second has cellpadding set to 10 and cellspacing set to 0, the third has cellpadding set to 0 and cellspacing set to 10, and the fourth has both cellpadding and cellspacing set to 10.

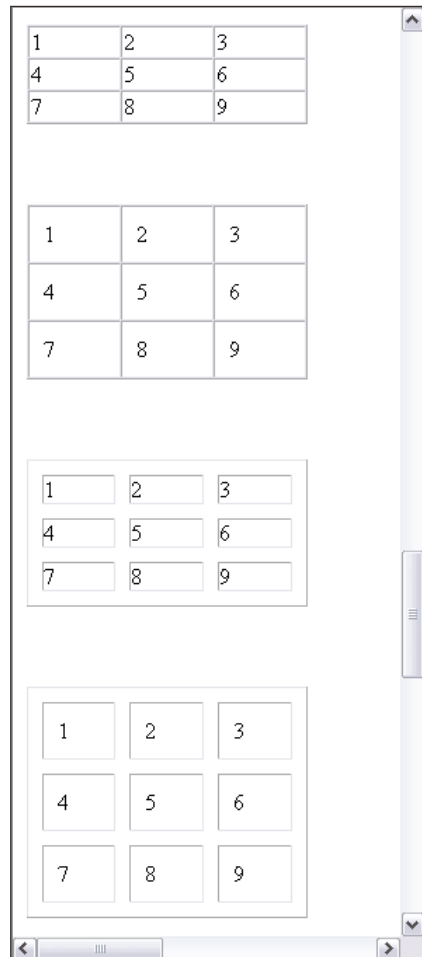


Figure 2-8: Use the cellpadding and cellspacing attributes to structure how content sits within a table.



For an alternate method of applying spacing between the content in your table cells and the table cell walls, create a CSS style that adds padding to contents of one or more cells within the table.

Table headers

In the past, when you wanted to format the topmost row and/or leftmost column in the table to be structurally different (typically bolder and one font size bigger) than the rest of the table's content, you'd use the old `<th>`, or table header, tags in place of `<td>` tags.



Today, however, table headers aren't used anymore because the `<th>` tag has several attributes (including `width`, `height`, `bgcolor`, and `nowrap`) that are being deprecated in the XHTML 1.0 Strict DTD. Therefore, instead of coding with the `<th>` tag, go right to CSS and create and apply custom styles to any table cell, row, or column that requires special formatting.

For example, to create a table header row or column with CSS, create a custom style and apply it selectively to the desired table cell, table row, or table column:

```
<tr>
  <td class="tableheadstyle">Specialty Drinks:</td>
  <td >Iced Cafe Mocha</td>
  <td >$3.75</td>
</tr>
```

The nowrap attribute

Another deprecated tag in XHTML 1.0 Strict code is the `nowrap` attribute, which used to override any specified column width applied to a table cell so that the contents of that cell would display in a complete line without any line breaks or text wrapping:

```
<td nowrap="nowrap">Address, Telephone, and Store Hours</td>
```

To perform the same function today, the better solution for keeping cells wide enough to display the cell's contents without wrapping is to ensure that the cell widths are set to the proper size within the CSS to accommodate the content:

```
<td class="nowrap">Address, Telephone, and Store Hours</td>
```

Splitting and merging table cells

Tables need not be totally uniform with an equal number of rows and columns. If your table requires it, you can split any single cell into two or more columns or rows. Likewise, you can merge any two or more contiguous (touching) table cells into a larger rectangular shape. What's more, if

you're using a good code editor, the program should handle all of the merging and spanning code for you so that you can focus on the table's layout.

To uniformly merge table cells, whether horizontally or vertically, one of two special attributes, `colspan` or `rowspan`, must be added to the `<td>` tag that initiates the merge. In effect, both of these attributes define a region that spans *n* number of rows or columns.

As shown in the following code example and illustrated in Figure 2-9, when the `colspan` attribute is used, only one `<td>` tag is needed for the part of the row being spanned. Likewise, when the `rowspan` attribute is used, any row or rows after the first cell in the rows being spanned do not need a `<td>` tag:

```
<table width="450" border="1">
  <tr>
    <td height="23" colspan="3">Overview:</td>
  </tr>
  <tr>
    <td width="126" rowspan="2">Key Features:</td>
    <td width="183">Snowball Maker</td>
    <td width="323">Make round compact snowballs.</td>
  </tr>
  <tr>
    <td>Snow Castle Molds</td>
    <td>Build your own forts!</td>
  </tr>
</table>
```

Overview:		
<i>Key Features:</i>	Snowball Maker	Make round compact snowballs.
	Snow Castle Molds	Build your own forts!

Figure 2-9: Split and merge table cells using the `colspan` and `rowspan` attributes.

Background and border colors

Like many of the other attributes for tables, you can create custom styles in CSS that do the same thing, only better. Nonetheless, you should still understand everything that can be done with HTML, in the event you inherit an old site that uses these attributes.

The background color (`bgcolor`) attribute is one of those attributes that used to be applied to both the table in the opening `<table>` tag and to individual table cells in the opening `<td>` table cell tags. Similarly, you could apply a border color (`bordercolor`) attribute to the entire table as well as

to table cells, though the `bordercolor` attribute for individual table cells is now inconsistently supported in browsers and isn't a recommended practice.

To illustrate how you might use these attributes (though, as I mention previously, it would be much better if you applied these styles to your table with CSS instead), you could have an overall background color on the entire table, a border color for the table, and a different background color set for one or more of the table cells, as illustrated in Figure 2-10.

Blue Plate Specials	
Meat Loaf & Mashed Potatoes	\$8.95
Old Fashioned Macaroni & Cheese	\$7.95

Figure 2-10: Tables can be styled with background colors, border colors, and cell background colors.

When styling with custom styles in CSS, your code might look like the following:

```
<table class="blueplates">
  <tr>
    <td colspan="3" class="specials">Blue Plate Specials</td>
  </tr>
  <tr>
    <td class="menuitems">Meat Loaf & Mashed Potatoes</td>
    <td class="menuitems">$8.95</td>
  </tr>
  <tr>
    <td class="menuitems">Old Fashioned Macaroni & Cheese</td>
    <td class="menuitems">$7.95</td>
  </tr>
</table>
```

By contrast, HTML code using the old `bgcolor`, `bordercolor`, and other deprecated table attributes would look something like this:

```
<table width="350" border="1" cellpadding="10" cellspacing="0"
  bordercolor="#000066" bgcolor="#99ccff">
  <tr>
    <td height="23" colspan="3" bgcolor="#99ccff"><strong>Blue Plate
  Specials</strong></td>
  </tr>
  <tr>
    <td width="286" bgcolor="#ffcc99">Meat Loaf & Mashed Potatoes</td>
    <td width="68" bgcolor="#ffcc33">$8.95</td>
  </tr>
```

```

<tr>
  <td bgcolor="#ffcc99">Old Fashioned Macaroni & Cheese</td>
  <td bgcolor="#ff9900">$7.95</td>
</tr>
</table>
    
```



To keep your code as light and clean as possible, and to separate form from content, I strongly recommend — no, *urge* — you to use CSS for background and border color styles instead of using the `bgcolor` and `bordercolor` attributes described here.

Tiling background images

In addition to, or instead of, adding a background color to a table or table cell, you can also apply CSS styles that will apply a *tiling* background image to a table or table cell. A tiling image is an image that repeats endlessly both horizontally and vertically, like the one shown in Figure 2-11. You tile a background image by creating a special background style and then applying it to a table or table cell:

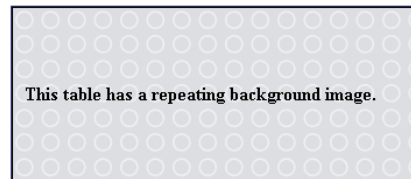


Figure 2-11: Use the `background` attribute to apply a repeating image to the background of a table or table cell.

```

<td colspan="2" class="patternbg">Get Connected!</td>
    
```

By contrast, the old HTML way of adding a tiling background image involved applying the `background` attribute, which specifies the location and filename of a graphic to tile in the background, to the opening `<table>` tag and/or to any opening table cell (`<td>`) tag within the table:

```

<td colspan="2" background="images/funkypattern.png">Get
  Connected!</td>
    
```

The two main drawbacks of applying a background image to a table or table cell using the `background` attribute are as follows:

- ✓ The image tiles, giving you no control over how it repeats.
- ✓ CSS does the job much better, including allowing you to set the starting point of the background image relative to the container as well as how and where the background image repeats.

Therefore, if you need to tile an image in a table — or in any other container for that matter — use CSS.

Nesting tables

When your table data has more complex layout requirements, tables can be nested inside other table cells. For example, you might need to create a section on your page to display product information in such a way that you can organize a product image, product name, description, details, stock number, and price. With a single table, you may not be able to size all the cells exactly as you'd like, but with a table nested inside another table cell, you could, as shown in the Dreamweaver nested table set of Figure 2-12.



Figure 2-12: Use nested tables to create customized content layouts for your data.

You may nest tables as often as you need to create the layout desired. Each nested table can be formatted and structured as needed, and the tables and all their content can be styled uniquely using CSS.

Inserting Lists on a Page

Lists are one of the best ways to organize content on your Web pages because they're easy to implement and can be tiered to create multilevel lists. Better yet, when combined with CSS, you can use list formatting to create dynamic standards-compliant HTML navigation systems that include both text and graphics.

To convert regular text into list format, you must add two tag components to the HTML code. First, the entire list must be surrounded by list tags. Second, each item in the list must be surrounded with list item tags. But, before you start coding, you have to first decide which kind of list you want to add to your page.

Examining the two list types

You can make two different kinds of lists in HTML: ordered lists, which use `` tags, and unordered lists, which use `` tags. Ordered lists display items in a sequence with an Arabic or Roman numeral or letter next to

them, while unordered lists simply list items with one of three types of bullets next to them. In both instances, the type of bullet or number is an attribute of the ordered or unordered list tag.

After you choose whether you want to create an ordered or unordered list, you can add individual items to the list using the opening and closing `` list item tags, as in the following example:

```
<ol>
  <li>Ham & Swiss Cheese</li>
  <li>BLT</li>
  <li>Grilled Cheese</li>
  <li>Egg Salad</li>
  <li>Tuna Salad</li>
  <li>Chicken Salad</li>
</ol>
```

In addition, you may also specify a list type as an attribute of the ordered or unordered list so that the list displays your items with the desired appearance:

- ✓ **Ordered lists** may be set to have one of five different appearance types: numbered (1, 2, 3), lowercase lettered (a, b, c), uppercase lettered (A, B, C), lowercase Roman (i, ii, iii), and uppercase Roman (I, II, III). To specify the list type for ordered lists, enter 1, a, A, i, or I in the `type` attribute of the opening ordered list tag:

```
<ol type="1">
  <li>Crosswords</li>
  <li>Sudoku</li>
  <li>Word Search</li>
</ol>
```

- ✓ **Unordered lists** are essentially bulleted lists that can be set to display all the items in the list with one of three types of bullets next to it: disc, circle, or square, as shown in Figure 2-13. The type of bullet can be set by adding the `type` attribute to the opening unordered list tag:

```
<ul type="circle">
  <li>Mercury</li>
  <li>Venus</li>
  <li>Earth</li>
</ul>
```

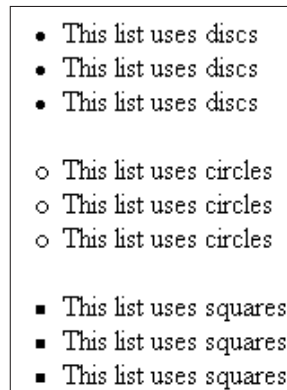


Figure 2-13: Use HTML markup to specify one of three different bullet types in an unordered list.

With both ordered and unordered lists, it is okay to leave off the `type` attribute. By default, all ordered lists use the numbered type, and all unordered lists use the disc type.

List item tags: To close or not to close

In some HTML training materials, both in book form and online, you may find that it is okay to omit the closing `` tag from your list HTML markup. In fact, the W3C openly states that this is fine to do in HTML; however, that doesn't necessarily mean the same is true when coding in XHTML. This is especially true when using the XHTML 1.0 Strict DTD, which, among other things, requires that all tags be closed. Furthermore, if you use any of the XHTML DTDs

and fail to close your list items with closing `` tags, the code in your page may not validate based on accessibility and standards compliance, which you read about in Book IV, Chapters 1 and 2. Therefore, to avoid possible validation issues, make your pages as accessible as possible, and to ensure a clean transition when migrating from HTML to XHTML, always close your tags.



If you'd rather use a custom bullet graphic instead of the three HTML list bullet types mentioned here, take heart. You can easily create and use your own custom bullet graphics when you combine your list with special CSS styles. Turn to Book III, Chapter 4 to find out more about styling lists.

Nesting lists

To create a nested list, which is a list that has one or more subsections or sublists in it, just add a full set of list markup tags to the code in line with any of the primary list items. In the following example, an unordered list is nested inside an ordered list:

```
<ol type="A">
  <li>Coffee
    <ul type="disc">
      <li>Regular</li>
      <li>Decaf</li>
    </ul>
  </li>
  <li>Tea</li>
  <li>Juice</li>
</ol>
```

Each of your lists may contain as many nested lists as needed to format your list in the desired configuration. To see how fast and easy it is to create a list, follow these steps:

1. Enter the following content into a blank Web page in your favorite HTML or code editor, making sure to add the following text between the opening and closing `body` tags:


```

Digital Cameras - SLR
  Nikon
    10.1 megapixel
    15.1 megapixel
    21.1 megapixel
  Canon
  Panasonic
Digital Cameras - Point & Shoot
  Casio
  Canon
  Fujifilm
  Kodak
    
```

Your code editor should have already added the bones of the code to your Web page, including the DTD, head, title, and body tags.

2. Select a list type for the main list, such as unordered, and add the markup for the list type and list items. Repeat for the other two list levels.

For instance, you might make Digital Cameras - SLR and Digital Cameras - Point & Shoot into a numbered list and then make the subitems in each list bullets, as in this example:

```

<ol>
  <li>Digital Cameras - SLR
    <ul>
      <li>Nikon
        <ul>
          <li>10.1 megapixel</li>
          <li>15.1 megapixel</li>
          <li>21.1 megapixel</li>
        </ul>
      </li>
      <li>Canon</li>
      <li>Panasonic</li>
    </ul>
  </li>
  <li>Digital Cameras - Point & Shoot
    <ul>
      <li>Casio</li>
      <li>Canon</li>
      <li>Fujifilm</li>
      <li>Kodak</li>
    </ul>
  </li>
</ol>
    
```

3. Save your file with the .html extension and select the Preview in Browser option in your editor to view your markup in a browser.

Most editors provide the option of viewing the code in a browser of your choice. For example, in Dreamweaver, you'd choose File→Preview in Browser and then select one of the browsers from the editable list of installed browsers on your computer. Figure 2-14 shows an example of how this list appears in a browser window.

Adding content and formatting a list

What kinds of content can appear in your list? The sky is the limit. List items can include text, hyperlinks, graphics, tables, layers, Flash movies, and anything else that can appear elsewhere on a Web page. In addition, you can use bold and italic tags to add emphasis, turn a list item graphic into a link, add named anchor links to content farther down on the same page, and best of all, select a font face, color, size, custom bullet, and more by creating and applying a custom CSS style to your list. Figure 2-15 illustrates how multiple CSS styles can be applied to different parts of a list. For details about customizing your lists with CSS, see Book III, Chapter 4.

Some browsers may still support a few deprecated list tag attributes that can alter when a list begins relative to the starting point. For instance, to make an ordered list begin later than the first item, such as starting with the letter *F* in a list with the alphabetical type attribute or the number 20 in a numbered list, add the `start` attribute to the opening `` tag:

```
<ol type="A" start="6">
  <li>Coffee
    <ul type="disc">
      <li>Regular</li>
      <li>Decaf</li>
    </ul>
  </li>
  <li>Tea</li>
  <li>Juice</li>
</ol>
```

-
- ```
1. Digital Cameras - SLR
 o Nikon
 ■ 10.1 megapixel
 ■ 15.1 megapixel
 ■ 21.1 megapixel
 o Canon
 o Panasonic
2. Digital Cameras - Point & Shoot
 o Casio
 o Canon
 o Fujifilm
 o Kodak
```

**Figure 2-14:** List markup can be used to create complex multitiered and multityped lists.

- 
- ```
1. Digital Cameras - SLR
   • Nikon
     ■ 10.1 megapixel
     ■ 15.1 megapixel
     ■ 21.1 megapixel
   • Canon
   • Panasonic
2. Digital Cameras - Point & Shoot
   ■ Casio
   ■ Canon
   ■ Fujifilm
   ■ Kodak
```

Figure 2-15: For best results, use CSS to format the content in your lists.

Though in ordered lists you cannot break up a list and continue the numbering from one part to another, you can reset the number in a list item by adding the `value` attribute to the `` tag. This sets the new starting value for all subsequent list items:

```
<ol type="1">
  <li value="10">Coffee
    <ul type="disc">
      <li>Regular</li>
      <li>Decaf</li>
    </ul>
  </li>
  <li>Tea</li>
  <li>Juice</li>
</ol>
```

Keep in mind that these attributes have been deprecated and that you may get inconsistent results in different browsers. Thus, for best control over the style of your list content, use CSS, as in the following example:

```
<ol id="menu">
  <li class="beverages">Coffee
  <li class="beverages">Tea</li>
  <li class="beverages">Juice</li>
</ol>
```


Chapter 3: Styling with Cascading Style Sheets

In This Chapter

- ✓ Getting familiar with the CSS syntax
- ✓ Creating inline, internal, and external CSS
- ✓ Understanding CSS selectors

In earlier chapters of this book, you discover that Cascading Style Sheets are really the way to go when it comes to styling the content on your pages. Yet until now, I haven't gotten into the nuts and bolts of how to create and apply CSS to your HTML. That's what this chapter is all about.

To make the task of finding out about styling pages with CSS flow as smoothly as possible, this chapter breaks CSS down into several easily digestible parts. First, you read about the anatomy of the CSS style syntax. Then you discover the difference between inline, internal, and external style sheets and find out how to link an external CSS to an HTML file. After that, you find out the basics of creating custom styles, redefining default tag styles, setting up ID styles, and using compound CSS styles to style several tags at once. Plus you find out how to create custom link styles for various sections within your Web page.

Understanding CSS Basics

Back in the late 1990s, a tool called Cascading Style Sheets was developed as an enhancement to traditional HTML markup that enabled designers to place the styling information for an entire Web site into a single, centralized external document, thereby decreasing the file size of all the HTML pages while at the same time reducing the amount of code required for styling in every page of a Web site.

In addition to keeping the HTML code less cluttered than it used to be when coding with the old `` and other HTML tags for text formatting, the use of CSS provides several other benefits, including the following:



- ✓ **Faster page download times:** Both the HTML files and the CSS file will load quicker when all the page-styling information is contained in the CSS.
- ✓ **Improved site access for visitors with disabilities:** Screen readers and other assistive devices can disable or otherwise ignore CSS, providing easier access to the content on a Web site.
- ✓ **Improved management of visual presentation:** A single external CSS file means that you can quickly make style modifications to an entire site by modifying the CSS, which is much faster than it would be if you were using the old `` and other formatting tags.

To illustrate, consider that you were using old HTML tags for styling, which means you'd have to specify the font face, size, and color each time a new paragraph required a different font appearance:

```
<p><font color="#FF6666" size="2" face="Georgia,Times,serif"><a href="casestudies.html"><b>Read our Case Studies</b></a></font></p>
```

Now compare that to working with CSS. When all the font attributes are transferred to the default styling for paragraphs (the `<p>` tag) in an external Cascading Style Sheet, the HTML code becomes much cleaner:

```
<p><a href="casestudies.html"><strong>Read our Case Studies</strong></a></p>
```

- ✓ **Easier site maintenance after the site is published:** With the old `` and other HTML style tags and attributes, any time the site's look needed changing, each file would have to be individually modified, and then *all* the changed files would need to be uploaded to the server before those changes would take effect for site visitors. With CSS, only the updated CSS file needs to be uploaded to the remote server before everyone can see the style changes. Simple!

Using CSS as a Web standard

You find countless benefits of working with CSS, the most striking of which is that when you place all your styling information in a single external CSS file, rather than embedding it within the HTML markup of your individual Web pages, you get the maximum amount of control over the look of your entire site. This is especially desirable during the design phase as you are constructing the pages for your site, and again later during the maintenance phase when it becomes time to make site-wide style changes.

The World Wide Web Consortium (W3C) strongly recommends CSS for styling Web pages because it gives designers and programmers the highest degree of control over how Web content is presented in a browser window. Much like a word processor or page-layout program's style sheets, CSS for the Web enables you to set default formatting options so that all the text and other content, such as graphics, forms, animations, and videos, follow

the design specifications of the mock-up. CSS also allows you to create customized styles that control the look and position of the objects and elements on your pages. You can then use those custom styles by selectively applying them to the various text, images, lists, tables, and other objects on a page as needed.

As you'll quickly find out, CSS can be used to control nearly everything about your site content's presentation in a browser. Use CSS to apply detailed settings to pretty much any page element, including the following:

- ✓ Font face, size, style, and color
- ✓ Margins, padding, and indenting
- ✓ Line and letter spacing
- ✓ Background colors and background images
- ✓ Border colors, size, and styles
- ✓ Table and list formatting
- ✓ Layer size, style, and positioning
- ✓ Hyperlink formatting

Thankfully, CSS is pretty easy to understand and use. By the end of this chapter, if you diligently read the text and follow along with each of the exercises found here, you should be well on your way to styling your own pages with Cascading Style Sheets.

In the sections that follow, you find out about the structure of a CSS style, the different ways you can add CSS to your pages, setting media types for displaying CSS on different devices, and more.

Taking a look at the anatomy of a style

To understand Cascading Style Sheets, you must first understand the anatomy of a style, which is made up of a selector and one or more declarations, both of which make up the rules for the style and instruct a browser how to display the content. Figure 3-1 diagrams the anatomy of a CSS style to help you understand how styles are written.

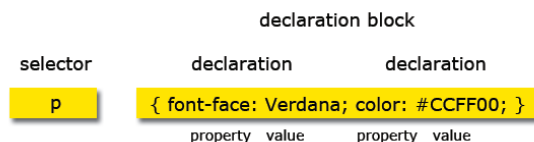


Figure 3-1: Each style in a Cascading Style Sheet has a selector and declaration.

The syntax of a style runs generally as follows, including all the funny punctuation and spacing:

```
selector { property: value; }
```

To explain more precisely, the selector and declaration work together as follows:

- ✓ **Selector:** The selector is the name of the style, which can be a tag name like `p` or `h1`, a custom name like `.superscript` or `.tableborder`, or the ID of an object preceded by the number symbol (`#`), such as `#sidebar` or `#menutable`. The selector tells the browser which tag(s) in the HTML to select and apply the style to, such as all content on the entire site that is surrounded by `<h1>` tags.

As you find out later in this chapter, you can use four different types of contextual selectors. Generally, each selector identifies a single element on the site to be styled in a particular way. However, for maximum efficiency of your code, you can also group selectors together, separated with commas without spaces, when you would like certain selectors to share particular style attributes, as in the following example, which sets the default font to Georgia for all paragraphs, heading 1s, and heading 2s:

```
p,h1,h2 {font-family: Georgia, Times New Roman,  
        Times, serif; }
```

- ✓ **Declaration:** The declaration, which is the part of the style that is listed between opening and closing curly braces (`{}`), can contain a single style definition or several style definitions in a block, each separated by semicolons. The declaration provides the details of the style, including what should be changed (the properties) and to what degree (the values of those properties), like the color of an element being changed to a particular blue. In addition, while only four types of selectors exist, declarations can include an unlimited number of property-value pairs (style attributes) from any of the eight different style categories, which are outlined in the following section.

In the actual usage, a CSS style might look something like the following code example, where the style selector is `h1` and the declaration defines the desired color of any content placed between `h1` tags:

```
h1 { color: #CC3333; }
```

Exploring inline, internal, and external CSS

The actual CSS code can be placed in three locations relative to the HTML code it will be styling and formatting: inline, internal, or external.

Inline CSS

These styles sit right next to the HTML code, typically preceding the content they'll be styling. Though this type of CSS code isn't used much for full Web site styling (because designers prefer to use either internal or external styles instead), some designers use inline styles for HTML e-mail formatting. The following example shows inline CSS applied to the `<p>` tag of HTML code:

```
<p style="color: #3399CC; font-size: 18px;">The Solar System</p>
```

Internal CSS

These style definitions (sometimes also called *embedded* styles) must be placed between the `<head>` tags of an individual Web page's HTML code for the styles to be applied to the content on that page. To take effect, internal CSS styles must be surrounded by `<style>` tags so that the browser can identify the content between them as CSS styles. Furthermore, within the `<style>` tags, the styles must also be surrounded by comment tags (`<!--` and `-->`) to prevent older browsers from displaying the style definitions as text in the body of the Web page.

For example, in the following code, the internal CSS for the `<p>` tag appears between the `<head>` tags so that all instances of text within the `<p>` tags in the body of the HTML are formatted as 12px with the hexadecimal color of #003366:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
  Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
  charset=iso-8859-1">
<title>Web Design All-in-One Desk Reference For
  Dummies</title>
<style type="text/css">
<!--
p {font-size: 12px; color: #003366;}
-->
</style>
</head>
<body>
<p>Crafting the Elements of Design</p>
</body>
</html>
```



While it is true that internal CSS is more efficient than styling your content using the old HTML `` tags, keep in mind that the definitions will only be applied to the page that includes them, which isn't very useful or economical when dealing with multipage Web sites.

External CSS

External style sheets, which some refer to as “linked styles” or “linked style sheets,” are the gold standard for CSS because they truly separate your styles from your content. External CSS files are useful for any size Web site, from sites as small as just a couple of pages to sites that contain hundreds of pages. External CSS is so fantastic because it keeps all the styling in one centralized file for easy access and easy maintenance.

External style sheets must be saved with the `.css` file extension. The file can then be placed anywhere within the site, but most designers place it either at the root level of the site or inside a folder at the root level, such as `css/mycssfile.css`. Your style definitions will be placed inside this external CSS file using the same style syntax of selector and declaration; however, you do not need to wrap the style information in style or comment tags. In other words, the only things in the external CSS file are the individual style definitions, such as

```
p {font-size: 12px; color: #003366;}
```

After you create your external style sheet, to get it to begin styling your pages, just insert a special link back to the CSS file. This link to the external CSS file must be added to all the pages on your site that you want to use it. In the HTML, you add a link to the external CSS file between the opening and closing `<head>` tags, as shown in the following example:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
    Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
    charset=iso-8859-1">
<title>Web Design All-in-One Desk Reference For
    Dummies</title>
<link href="mycssfile.css" rel="stylesheet" type="text/css">
</head>
<body>
<p>Crafting the Elements of Design</p>
</body>
</html>
```

You find additional details about linking external CSS files to HTML pages in the next section.

Combining different types of CSS on your site

While it is certainly more likely that you will use a single master external CSS file for styling an entire site, nothing is stopping you from incorporating inline, internal, and external CSS in any combination to achieve the desired results. For example, you might have one external CSS file, a few internal

CSS styles on a couple of pages, and a few instances of inline styles on a particular page of a site.



Should you choose to combine different types of CSS on your site, keep in mind that all style definitions are hierarchical in nature. That's the cascading part of Cascading Style Sheets. This means that any style definitions that sit closest to the content on your pages — your text, graphics, and so on — will override any styles that sit farther away. In other words, external CSS is overridden by internal CSS, which is overridden by inline CSS or any other inline styling tags and attributes, including any old `` tags. To make matters even more interesting, a visitor's browser is truly the closest thing to the content, which means that if his browser has any customized preferences set with regard to the display of CSS, it may override your CSS styles, regardless of whether your styles are inline, internal, or external.

This hierarchical rule also applies to redundant or conflicting CSS within the same document. For instance, if a file contained something like two `<body>` tags with conflicting font-family declarations, the style definition that sits lower in the list of styles, closer to the content, will be the style applied to the page. In the following example, the body selector that specifies Geneva would be used to style the page, while the other style rule would be ignored:

```
body {
  font-family: Georgia, "Times New Roman", Times, serif;
}
body {
  font-family: Geneva, Arial, Helvetica, sans-serif;
}
```

Additionally, as you'll quickly come to realize, after you set certain style attributes in the CSS for particular tags like the body or paragraph tags, only attributes that differ from those need to be specified for subsequent styles. You could, for example, make Verdana the default font for all paragraph text on all the pages on the entire site by setting that as a style attribute for the selector called `p`. When this style is applied to your pages, all the paragraph text throughout your site would display in Verdana unless specified differently in subsequent style definitions.

Linking external CSS to a page

To link an HTML page to an external CSS, just insert a single line of code into the head of your page that references the name and location of the external CSS file relative to the root level of the server on which the site resides. The line of code for the link must be placed somewhere between the opening and closing `<head>` tags of every HTML page on your site that you want to be styled with it. When the filename of the CSS is accurately entered in the HTML file using the correct link syntax, the link code provides instructions to the browser about how the CSS style information should be interpreted

and applied to the page, which in turn determines how the page appears in the browser.

Here's an example of a link to an external CSS file with the filename `main.css`, where the file that is being sourced uses the `href` attribute of the link tag, which is an unclosed tag in HTML:

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

When the link code is added to an XHTML page, the tag must be closed by adding an extra space and forward slash before the end:

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

The other attributes within the `<link>` tag besides the `href` are required to help the browser interpret the data on the linked CSS file:

- ✓ The `rel` attribute identifies the linked file as a style sheet
- ✓ The `type` attribute specifies that the linked file is written in `text/css` format.



For consistency, try to place the CSS link in your Web pages in the same location within the code from page to page. For instance, you might want to add the link tag directly following the last meta tag, or place it right above the closing `</head>` tag. Being consistent can help you quickly find the tag should you ever need to modify it.

Besides the placement of the link within the HTML pages that will use the external CSS file, you should also pay some attention to where the CSS file(s) is located relative to the other files within the site.

Most often, each site will have just one CSS file, and that file will sit at the root level, which is just a fancy way of saying that the CSS file will be in the same location as the `index.html` file, which is the home page for your site. The root level refers to the ground floor of your site, whether it's a local copy of your site sitting on your computer in front of you or a copy of the site located on the remote host server. With most sites, the home page sits at the root level along with an `images` folder and all the other main pages of the site. For larger sites, some designers create subfolders at the root level to house other things like external JavaScript files, external CSS files, CGI scripts, or groups of pages that fall into a similar category, such as all the pages relating to a company's products or services.

Should your site require two or more Cascading Style Sheets (perhaps one for all the pages and a second for a handful of pages that will be printed; see the next section), it may be beneficial for you to create a separate folder called `css` at the root level of the site and then save those CSS files together inside it. You may then access each CSS from that location, providing that the `href` of your CSS link indicates the new location of the folder along with the filename:

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

Setting CSS media types

In Web-speak, the *media type* is the specification within a Cascading Style Sheet that identifies the device that will be used to access the HTML file being styled. Examples of a media type include a computer screen, printer, handheld gadget, Braille translator, speech synthesizer, or other type of assistive device.

Exploring different media types

By default, it is presumed that all Web pages should be accessible by any and all devices, or media types, that can access a Web page, and as such, you don't need to specify a media type in the tag that links to the external CSS. However, when your pages do require one or more CSS files to style the content for different devices, it's a good idea to add the media type attribute to the link tag.

Table 3-1 identifies all the media types currently in use.

Table 3-1 CSS Media Types	
Media Type	Definition
all	Good for all devices, recommended as the default catchall type when multiple cascading style sheets are specified.
aural	Used with text-to-speech devices.
Braille	Used for Braille tactile feedback devices.
embossed	Used for paged Braille devices.
handheld	Used for small-screened devices with limited bandwidth capabilities and often monochromatic or limited color displays, such as the BlackBerry or a Web-enabled cell phone.

continued

Table 3-1 (continued)

<i>Media Type</i>	<i>Definition</i>
print	Best for files intended for print, whether actually printed or viewed only in Print Preview mode.
projection	Used for overhead projectors or documents turned into transparencies for projection.
screen	Best for color monitors.
tty	Good for teletype machines, special text terminals, and other “fixed-pitch character grid” devices. Note: When creating CSS for this type, avoid specifying the sizes of any objects on the page in pixels.
tv	Used for TV-type devices that might have less robust features than a regular color computer monitor.

Creating separate CSS files for different media types

While many CSS style declarations work across all media types, some are to be used only with specific media, like the speech-rate property that can be used only with aural devices. Likewise, when certain style declarations are shared by two devices, those declarations might need adjusting in the secondary CSS so that those styles look good on both devices. In cases like this, creating two separate CSS files, one for each device, might be necessary to improve the experience of the HTML document on both devices.

Here’s an example. Consider that you have a Web page that looks good on-screen, but when the content is printed, some of the graphics on the page take up so much room that they force the printer to cut off some of the text along the right edge of the page, as well as push any overflow text onto another page. You can easily solve this problem by creating two separate style sheets: one for all media types and another for printers.

The following steps illustrate how you can set a document to use two CSS files with different media types:



- 1. With a live Internet connection, launch your favorite Web browser and open the sample CSS media types demo file `cssmediatypes.html`, which can be found at www.dummies.com/go/webdesignaio.**

This page, shown in Figure 3-2, contains a `<link>` tag with an `all` media type attribute that specifies a particular external CSS.

- 2. Under the View menu, look for and select the option that will allow you to view the source code of the page.**

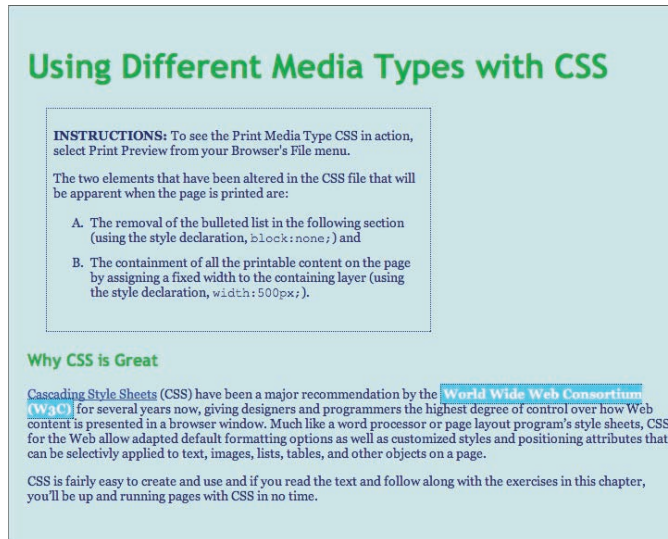


Figure 3-2: A page styled with a linked external CSS file with a media type set to `all` will display nearly identically in all Web-enabled devices.

The exact command varies by browser. For instance, in Firefox the command is View⇨Page Source, in Internet Explorer it's View⇨Source, and in Safari it's View⇨Source.

Right before the closing `</head>` tag, you can see a second link to an external CSS file that instead uses the `print` media type:

```
<link href="cssdemo.css" rel="stylesheet" type="text/css" media="all" />
<link href="cssmediatypes.css" rel="stylesheet" type="text/css"
media="print" />
```

- 3. To see how the content on the page looks differently when you try to print it, choose File⇨Print Preview to open the Print Preview dialog box.**

Figure 3-3 shows how the `print` media type CSS looks in the Print Preview dialog box. The `cssmediatypes.css` style sheet contains a few attributes that are different than the `cssdemo.css` style sheet. Namely, in the `cssmediatypes.css` file, the entire bulleted list at the bottom of the page has been styled as a hidden block by using the `block: none;` declaration, and the contents on the entire page have been placed inside a layer that has been styled to have a fixed width of 500 pixels when the page gets printed.

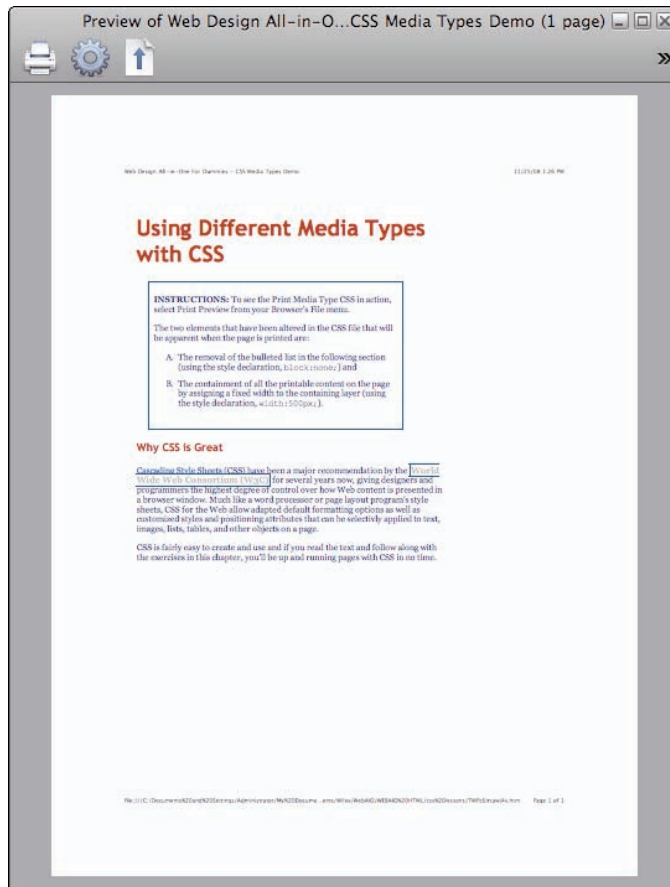


Figure 3-3: A page styled with a linked external CSS file with a media type set to print will not become apparent until the page gets printed.

Adding media-dependent style sheets to your HTML files

You can add external media-dependent style sheets to the head area of your HTML files in two ways: linking and importing. While both methods can essentially do the same thing for entire external CSS files, importing accommodates style-specific applications for different media types.

Here's an overview of the two CSS attachment methods:

- **Linking:** Use the `<link>` tag to specify both the location and filename of the external CSS and the desired media attribute, such as screen, all, or print, to define the media type:

```
<link rel="stylesheet" type="text/css" media="print" href="forprint.css">
```

- ✓ **Importing:** Use the `@media` or `@import` at-rules surrounded by `<style>` and comment tags (to hide the style specification from displaying in the body of the page when the page is viewed in older browsers). The syntax is slightly different for `@media` than it is for `@import`, but both methods do essentially the same thing when the specified CSS is applied to the page:

```

<style type="text/css">
<!--
@media print {
  /* printversion.css */
}
-->
</style>
or
<style type="text/css" media="print,handheld">
<!--
@import url("css/printheadheld.css");
-->
</style>

```

When you want to use internal CSS as opposed to an external CSS file, the `@media` method can be used to indicate one or more media types as well as include particular style definitions that will remain internal to the HTML page, as in this example:

```

<style type="text/css">
<!--
@media screen, print {
  body { font-family: Georgia, "Times New Roman", Times,
        serif; }
}
-->
</style>

```

Alternatively, when certain attributes will be different on-screen than they will be for another media type, you can stack the style definitions on top of one another, as shown here, allowing you to apply media-specific rules to specific styles within your page:

```

<style type="text/css" media="all">
<!--
@media screen {
  p { font-size: 10px; }
}
@media print {
  p { font-size: 12px; }
}
-->
</style>

```

If, on the other hand, your CSS files will be external to the pages on the site, sitting at the root level or inside a `css` folder, the `@import` or `<link>` method will be more useful. For instance, you could use the `<link>` method to list the `all` media type for the first linked CSS file, and beneath that, add another link tag to list another media type and CSS file:

```
<link rel="stylesheet" type="text/css"
      media="all" href="mystylesheet.css">
<link rel="stylesheet" type="text/css"
      media="print" href="mystylesheetprint.css">
```

Linking CSS with Dreamweaver

Dreamweaver users can easily insert a link tag to an external CSS without having to commit to memory all the required code or the proper syntax. Moreover, when creating links to an external CSS file in Dreamweaver, you can also use that opportunity to select a CSS media type, if desired.

In the following steps, you find out how to link an external CSS file with the `all` media type to an open HTML file in Dreamweaver.



To complete all the steps, you need Dreamweaver along with sample HTML and CSS files, which you can quickly create on your own. Or if you'd like to follow along with the example, feel free to download copies of the HTML (`cssdemo.html`) and CSS (`cssdemo.css`) sample files at www.dummies.com/go/webdesignaio.

Place both copies of the saved files into a folder on your computer and then proceed with the following steps:

1. Launch Dreamweaver and open the HTML file that you will be adding the CSS link to.

The file you use should have some type inside of it and be marked up with paragraph, `h1`, and list text.

2. To add the link, click the Attach Style Sheet icon (which looks like a little piece of chain) at the bottom of the CSS Styles panel.

Clicking the icon opens the Attach External Style Sheet dialog box, shown in Figure 3-4. If you don't see the CSS Styles panel in the Dreamweaver workspace, choose `Window` ⇨ `CSS Styles` to open it.

3. In the File/URL text field, type the name of the CSS file you'd like to link to, or click the Browse button to find and select the desired CSS file.

If you're using the sample files you just downloaded from the Web, click the Browse button to navigate to the location where you saved the sample files and select the file `cssdemo.css`.

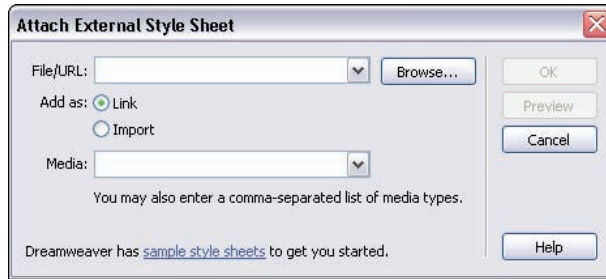


Figure 3-4: Use Dreamweaver's Attach External Style Sheet dialog box to select the desired CSS file and media type.

- 4. In the Add As area of the dialog box, click the Link or Import radio button to select the desired method for adding the CSS file to your HTML page.**

The Link option adds the CSS as an external file using the `<link>` tag:

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

The Import option specifies the external CSS within a style link inside the head of the page by using the `@import` at-rule:

```
<head>
<style type="text/css">
<!--
@import url("cssdemo.css");
-->
</style>
</head>
```

- 5. In the Media drop-down menu area, type in the word all, or click the menu's down arrow and select the all media type.**

To specify multiple media types rather than just one, enter the names of each of the desired media types, separated by commas and no spaces, as in `screen,print,tty`.

6. If desired, click the dialog box's Preview button to see how the newly linked CSS file styles your sample HTML file.
7. Click the OK button to complete the attachment of the external CSS file.

Dreamweaver's CSS Styles panel now displays the newly attached CSS file and lists all the styles inside it, and the sample HTML file is styled with the style rules on the linked CSS.

Working with CSS Style Selectors

Now that you understand the differences between inline, internal, and external CSS styles, you are ready to discover the four different contextual selectors, or CSS selector types:

- ✓ Custom class styles
- ✓ Tag redefine styles
- ✓ ID styles
- ✓ Compound styles, which include customized CSS styles like descendant selectors, hyperlinks, and advanced combinators

Though they all use roughly the same syntax for the style declarations, each type determines which precise parts of the HTML will be modified.

Applying custom class styles

Custom class styles, which some refer to as *custom styles* or *custom classes*, are for those times when you want to create a special style and then selectively apply it to particular bits of text or objects on a Web page. For example, in the sentence "Our Daily Deals newsletter brings you the hottest sales, promotions, and special offers at the most popular stores in one easy-to-read daily e-mail," you could create a custom class style to modify the words *Daily Deals* and then apply that style to those words in the HTML.

When writing the custom class styles in the CSS file, be sure to include a period (.) directly before the selector name, as shown here:

```
.dailydeals {
  font-family: Georgia, "Times New Roman", Times, serif;
  font-size: 23px;
  font-weight: bold;
  color: #369;
}
```

The presence of the period performs two functions:

- ✓ It helps you to quickly identify, at a glance, the custom styles from other types of styles when reviewing your CSS code.
- ✓ Perhaps more importantly, it informs browsers that the style is a custom class that will be selectively applied to content on the page.

When you create a custom class style, the selector can be named anything you like, as long as it is not the name of a currently used HTML tag. For example, it would be a really bad idea to create a custom style called `.body` or `.p`. So, in keeping with the concept of semantic HTML, try to name your custom styles after the function they'll be performing, such as `.highlight` or `.imageborder`.

After you are finished writing out the style rules in the CSS for your custom class style, you can apply the style to any object in the HTML document by adding the `class` attribute to the opening container tag of the object or content being styled:

```
<p class="dailydeals">Our Daily Deals newsletter brings you the hottest sales,
  promotions, and special offers at the most popular stores in one easy-to-
  read daily e-mail.</p>
```



When specifying the custom style in the HTML code with the `class` attribute, the period that is required in the CSS when creating the style definition does not need to be placed in front of the style name in between the quotation marks, as shown in the previous code.

If you prefer that the custom style be applied only to one or two words or a short phrase instead of all the elements within a container tag, you can selectively apply the custom style to your object(s) using the `` tag with the `class` attribute:

```
<p>Our <span class="dailydeals">Daily Deals</span> newsletter brings you the
  hottest sales, promotions, and special offers at the most popular stores in
  one easy-to-read daily e-mail.</p>
```

The `` tag is an empty HTML container tag that does nothing until you tell it to do something by applying an attribute to it, such as adding the `class`, `id`, or `style` attribute to style content. Figure 3-5 illustrates the difference between adding a `class` attribute to a `<p>` tag versus adding it to a `` tag that surrounds specific content.

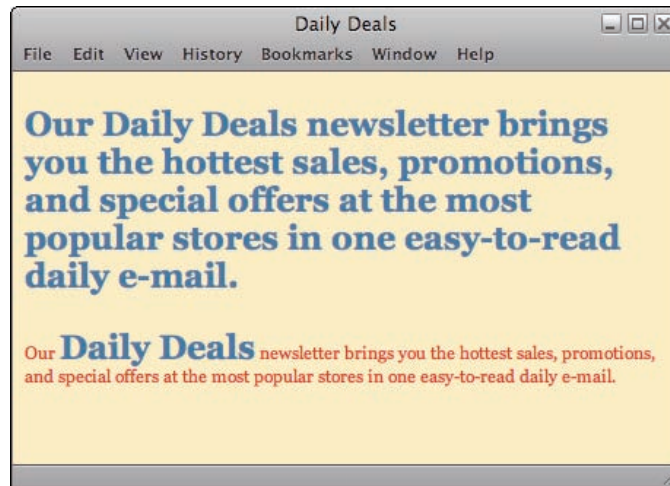


Figure 3-5: Custom class styles can be added to an existing container, such as a `<p>` tag (top), or wrapped around specific content using the `` tag (bottom).

Making CSS tag redefine styles

By default, all HTML tags are structurally preformatted to look a particular way and perform specific functions. Take the `<h1>` tag, for example. This tag is preformatted to be big, black, and bold, and it is intended to identify the main heading within the text, as opposed to the regular text, which is marked up with `<p>` tags in the content.

When you create a *tag redefine style*, you use the tag name as the selector name to change the preformatted look of any existing HTML tag, such as `<p>` and `<h1>`. The preformatted style can be changed with a tag redefine style into anything you like, such as tailoring the default look of all content between `<h1>` tags to match the design and color scheme of your site, whether that be Impact, 28 pixels, bold, italic, and #000000 or Palatino Linotype, 32 pixels, bold, and #FF99FF:

```
h1 {
  font-family: Palatino Linotype, Book Antiqua, Palatino,
    serif;
  font-size: 32px;
  font-weight: bold;
  color: #FF99FF;
}
```

Redefining existing tags is one of the best ways to globally style content on a site without having to selectively apply the styles here and there, as you must with custom class styles. In fact, to streamline the CSS process, most designers at a minimum begin each CSS file by creating tag redefine styles for the `<body>`, `<p>`, `<h1>`, and `<td>` tags. A tag redefine style for the `<body>` tag, for instance, can take on many of the attributes that were formerly applied to the opening `<body>` tag in HTML code, such as the default page margin spacing and page background color.

All Web pages, unless otherwise specified, have a default 9-pixel margin of space between the edge of the browser window and its contents, as illustrated in Figure 3-6.

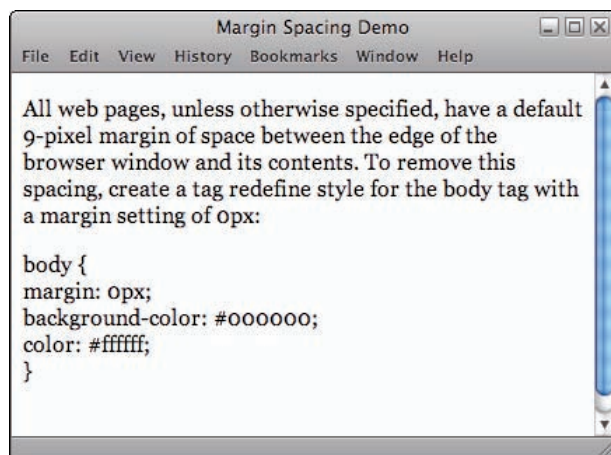


Figure 3-6: All pages have a default 9 pixels of margin spacing around the entire browser.

To remove or otherwise change this setting, the `margin` attribute may be redefined in the `body` tag. Figure 3-7 shows an example of how a style for the `<body>` tag might be redefined in the CSS, where the margin spacing is set to 0 on all four sides of the browser window, the background color of the page is set to black, and the font color for text within the body is set to white, as in the following style code:

```
body {
margin: 0px;
background-color: #000000;
color: #ffffff;
}
```

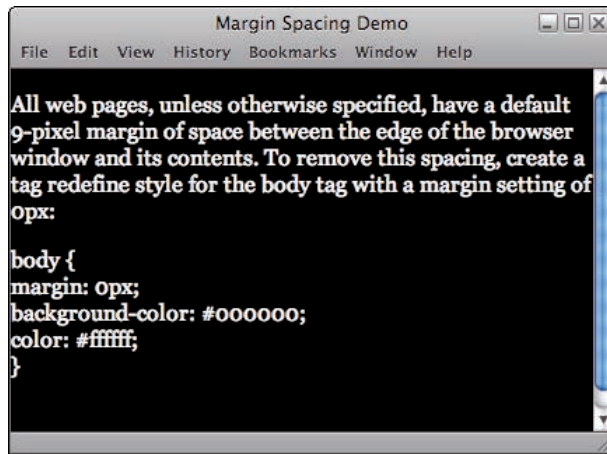


Figure 3-7: Create a tag redefine style for the body to remove any unwanted default margin spacing.

Creating ID styles

An *ID style* is a kind of hybrid CSS style rule that combines certain elements of both custom class and tag redefine styles. With an ID style, the declarations defined in that style are automatically applied to any object on the page that has an `id` attribute that matches the `id` name in the ID style.

To style an object with an `id` attribute, you must first create a selector that includes the number symbol (`#`) followed by the `id` name, such as `#border`, and then add as many declarations to the style as desired, as in the following example:

```
#border {
    border: 1px dashed #cad0d6;
    margin: 0px 1px 0px 0px;
    padding: 10px;
}
```

After you've created your style, add the `id` attribute and style name to the opening tag of the object or other HTML container tag that will use the style, as in `<div id="border">`.

To further illustrate this idea, presume that you have a layer on your page (using the `<div>` tag) that contains a header that says "Popcorn Makers" and a listing of five different brands of popcorn makers for sale, such as the one shown in Figure 3-8.



Figure 3-8: To style an object automatically with an ID style, you must first add the `id` attribute to the object's opening tag.

When you give the layer an ID attribute of `popcornmakers`, using the syntax `id="popcornmakers"`, and create a style using the syntax `#popcornmakers`, that ID style is automatically applied to the layer, as illustrated in Figure 3-9.



Figure 3-9: ID styles in the CSS are automatically applied to objects on the page with the same ID.

Here's an example how an ID style is written in the CSS:

```
#popcornmakers {
  font-family: Georgia, "Times New Roman", Times, serif;
  background-color: #FFC;
  border: 1px solid #039;
  position: absolute;
  width: 330px;
  height: 180px;
  z-index: 1;
  left: 50px;
  top: 50px;
  padding: 10px;
}
```

And here is what the HTML markup looks like:

```
<div id="popcornmakers">
  <h3>Popcorn Makers</h3>
  <ul>
    <li>Presto Hot Air Popcorn Maker</li>
    <li>Classic Popcorn Popper</li>
    <li>Retro Popcorn Maker</li>
    <li>West Bend Housewares Popcorn Maker</li>
    <li>Cuisinart EasyPop Popcorn Maker</li>
  </ul>
</div>
```

Just like the tag redefine selector, which automatically changes how contents surrounded by a particular tag appear, anytime that a style name uses the `#id` syntax, the style definition is automatically applied to the object with the matching `id` attribute.

Building compound styles

After you've mastered working with custom class, tag redefines, and ID styles, you can step into the big leagues of compound styles. This advanced selector type is where a lot of the fancy footwork in CSS happens because the selector can be written in a variety of ways including the following most popular usages:

- ✓ **Custom hyperlinks:** Used to modify the look of hyperlinks in different areas of a Web page, a compound style can be written as a two-part selector where the anchor tag is separated by a colon followed by the name of the link state, as in `a:link`, `a:visited`, `a:hover`, and `a:active`.
- ✓ **Multiple selectors:** Used to apply the same styles to several tags, the advanced selector is divided by commas and no spaces, as in `body,th,td`.

- ✓ **Advanced combinators:** For times when styles need to be applied only to specific content areas on a Web page, the advanced combinator selector, or dependent selector, can be written to include any combination of tags, tag redefines, ID styles, and custom class names, as in the following selector name that would only style the list items in an unordered list inside an object with an `id` attribute called `popcornmakers`: `#popcornmakers ul li`.

One of the most common usages of the compound styles is to create custom hyperlinks, which can be used to modify the default color and attributes of hyperlinks. The appearance of a hyperlink is determined by a visitor's interactivity with it in a browser:

- ✓ **Normal links** are unvisited links.
- ✓ **Visited links** are links that the visitor has already clicked.
- ✓ **Hover links** change their appearance when a visitor hovers the cursor over them.
- ✓ **Active links** change their appearance when a visitor clicks them.

As you may well know, all hyperlinks by default display in either blue underlined text (unvisited link) or purple underlined text (visited links). Because these colors are unlikely to match the particular colors in your site's design, you can override the default link styles by creating custom hyperlink styles. Even better, in addition to the unvisited and visited hyperlink states, with CSS, you can add styles for two additional hyperlink states, namely, the hover state, which occurs when a visitor mouses over a link, and the active state, which appears when a visitor clicks a link.

To change just the color of a hyperlink for all four link states, add the following style definitions to your CSS, replacing the hexadecimal values in this example with your desired color values for each of the link styles:

```
a:link {
    color: #CC0000;
}
a:visited {
    color: #339933;
}
a:hover {
    color: #000000;
}
a:active {
    color: #99CC33;
}
```



To further make things interesting, you can add additional style declarations for any of or all the four link states. For instance, you might want to remove the underline, add a background color, or apply a dotted border around the hyperlink, as demonstrated here and shown in Figure 3-10:

```
a:link {
    color: #CC0000;
}
a:visited {
    color: #339933;
    text-decoration: none;
}
a:hover {
    color: #000000;
    background-color: #0CC;
}
a:active {
    color: #ffffff;
    text-decoration: none;
    border: 2px dotted #CCC;
    background-color: #000;
}
```



Figure 3-10: Create four distinct styles for each of the four hyperlink states.



When creating these link styles, you must take care that each style gets added to the CSS in the same order it will be experienced on a Web site by a site visitor; link, visited, hover, active. If the styles are added to the CSS out of order, it may not work properly when viewed in a browser. Therefore, get in the habit of creating the normal link state first, then the visited state, then the hover state, and finally the active state, as shown in the preceding example.

Creating a master CSS file

Other HTML tags that Web designers often redefine in the CSS include `<html>`, `<h1>` through `<h6>`, `<td>`, ``, and ``. To tell the truth, many designers create their own version of a master CSS file, which they then adapt to the needs of each individual site, to help speed the process of building a Web site from scratch.

As you build more and more Web sites, some styles will become a regular part of your standard design practice. For instance, you might always want to set your page margins to 0, specify a page background color, choose a default font for all text content, create redefine styles for paragraphs and headings, specify style attributes for at least two (link and visited) if not all four hyperlink states, and make a custom bullet style for styling lists.



To create your own master CSS file, follow the steps below:

1. **Before you begin creating the master CSS, download a copy of the sample HTML file called `sample.html` from www.dummies.com/go/webdesignaio.**

You will use this file to test the styles for your CSS as you create the master CSS file. The sample HTML file includes paragraph text, a heading 1, a heading 2, an unordered list, and a couple of functioning hyperlinks.

2. **Create a new blank document, without any HTML coding, and save it with the filename `master.css`.**

Save this CSS file in the same location on your computer as your `sample.html` file.

3. **Inside the `<head>` area of your `sample.html` file, add a link to the new external CSS file that includes the media type set to `all`:**

```
<link href="master.css" rel="stylesheet" type="text/css" media="all">
```

This link tells the `sample.html` file to use the style definitions in the linked external CSS.

4. **Inside the `master.css` file, create a redefine style for the `<body>` tag that sets the top, left, bottom, and right page margins to `0px`; the padding on all four sides of the page to `10px`; and the background to a light peachy orange color with the hexadecimal value of `#fc3bb6`.**

Your style code should look like this:

```
body {
    margin: 0px;
    padding: 10px;
    background-color: #fc3bb6;
}
```



When all four sides of an object use the same value, as with the margin spacing and padding shown here, the value only needs to be specified in the CSS once. However, when the value is different on one or more sides, you must specify values for each of the sides:

```
body {
    margin: 10px 10px 0px 0px;
    padding: 20px 0px 0px 20px;
    background-color: #fc3bb6;
}
```

5. **Create a redefine style in your CSS file for the `<p>`, `<h1>`, and `<h2>` tags by specifying the font, font size, and font color for each.**

Use any font, size, weight, and color you like because you can customize the values later to match any specific project. Here's an example of the code you might use:

```
p {
    font-family: Georgia, "Times New Roman", Times,
    serif;
    font-size: 12px;
    color: #000000;
}
h1 {
    font-family: Arial, Helvetica, sans-serif;
    font-size: 36px;
    font-weight: bold;
    color: #000066;
}
h2 {
    font-family: Arial, Helvetica, sans-serif;
    font-size: 24px;
    font-weight: bold;
    color: #000066;
}
```

6. To change the default hyperlink style, you can create styles for each of the four hyperlink states.

You may specify any attributes you like for each of the four states, from changing the font or font weight, to modifying the text color or background color, to altering the default text decoration.

Here's an example of the code you might use for the four link states:

```
a:link {
    font-weight: bold;
    text-decoration: underline;
    color: #0099cc;
}
a:visited {
    font-weight: bold;
    text-decoration: underline;
    color: #990000;
}
a:hover {
    font-weight: normal;
    text-decoration: none;
    color: #ffffff;
    background: #ff9933;
}
```

```
a:active {
    font-weight: normal;
    text-decoration: none;
    color: #ffffff;
    background: #cc0000;
}
```

- 7. To style the unordered list, you can either redefine the `` tag or create a custom style that can be selectively applied to any `` tag with the `class` attribute. If desired, specify an image to replace the default bullets.**

The style definition will look the same whether you redefine the `` tag or create your own custom class style; only the selector will be written differently, as either `li` or perhaps as `.bullet`.

Your code for the bullet redefine style might look something like this:

```
li {
    list-style-position: outside;
    list-style-image: url(images/bullet.gif);
    font-family: Arial, Helvetica, sans-serif;
    font-size: 12px;
}
```

- 8. Save your HTML and CSS files and launch your HTML file in a browser window.**

To view the page in a browser, you can either double-click the HTML file or drag and drop the file icon into any open browser window.

The file should display with all the style attributes you just created in your master CSS file, as shown in Figure 3-11. If it doesn't look quite right or if certain elements aren't displaying properly, reopen the files and check the accuracy of all your code, fix any errors you find, and retest. Be sure you've remembered to add the period (.) before all your custom class names and a number symbol (#) before all your hexadecimal color values.

- 9. Test your new hyperlink styles in the browser window by**

- a. Mousing over a link to see the hover style
- b. Clicking and holding the mouse over a link to see the active style
- c. Clicking a link and returning to your sample page by clicking the browser's Back button to see how the link changes from the normal to the visited link state

Cascading Style Sheets

Thus far this book has mentioned and touted the benefits of working with [Cascading Style Sheets](#) but has not gotten into the fundamental aspects of how to create and apply CSS to your HTML. Here you'll learn some of the reasons why CSS makes designing Web pages easier.

Why CSS Is Great

Cascading Style Sheets (CSS) have been a major recommendation by the [World Wide Web Consortium](#) (W3C) for several years now, giving designers and programmers the highest degree of control over how Web content is presented in a browser window. Much like a word processor or page layout program's style sheets, CSS for the Web allow adapted default formatting options as well as customized styles and positioning attributes that can be selectively applied to text, images, lists, tables, and other objects on a page.

CSS allows you to take control over many aspects of presentation, including:

- ▶ Font face, size, style, and color
- ▶ Margins, padding, and indenting
- ▶ Line and letter spacing
- ▶ Background colors and images
- ▶ Border colors
- ▶ Table and list formatting
- ▶ Layer size, style, and positioning
- ▶ Hyperlink formatting

CSS is fairly easy to create and use and if you read the text and follow along with the exercises in this chapter, you'll be up and running pages with CSS in no time.

Figure 3-11: Creating a master CSS file, like the one styling this page, can make building each site go much faster.

Now that you have your first master CSS file, rather than re-invent the wheel each time you start a new Web project, you can use this file as the starting point. Of course, for some projects, building the CSS from scratch might be easier or more practical, but if having a master CSS file will save you time, by all means use it as a design technique.



One last thing about your CSS files. Try to keep your CSS styles organized as much as possible. If needed, consider grouping similar styles and labeling the different groups using CSS comment tags (which are different than HTML formatting tags, `<!--` and `-->`), as in the following sample master CSS file:

```
/***** Global Settings *****/

body {
    background-color: #B3D2E1;
    margin: 0px;
    padding: 0px;
}
body,th,td {
    font-family: Georgia, "Georgia Ref", Tahoma, "Palatino Linotype", Palatino,
    serif;
    font-size: 12px;
    color: #26506c;
}

/***** Headings *****/

h1 {
    margin: 0px;
    padding: 30px 0 20px 0;
    font-size: 32px;
    font-weight: bold;
}
h2 {
    margin: 0px;
    padding: 20px 0;
    font-size: 18px;
    font-weight: bold;
}
h3 {
    margin: 0px;
    font-size: 14px;
    font-weight: bold;
}

/***** Common Formatting *****/

p {
    font-family: Georgia, "Georgia Ref", Tahoma, "Palatino Linotype", Palatino,
    serif;
    font-size: 11px;
    color: #26506c;
    margin: 0px;
    padding: 0px;
}
img {
    border: 1;
    padding-top: 5px;
    padding-right: 5px;
    padding-bottom: 5px;
    padding-left: 5px;
}

/***** Links *****/

a:link {
    font-weight: bold;
    text-decoration: underline;
    color: #26506c;
}
```

300 Working with CSS Style Selectors

```
a:visited {
    font-weight: bold;
    text-decoration: underline;
    color: #26506c;
}
a:hover {
    font-weight: bold;
    text-decoration: none;
    color: #3C474F;
    background: #CAD0D6;
}
a:active {
    font-weight: bold;
    text-decoration: none;
    color: #FFFFFF;
    background: #99CA3C;
}
}
```



To download a free master CSS file like this code, go to www.dummies.com/go/webdesignaio.

Chapter 4: Understanding CSS Style Properties

In This Chapter

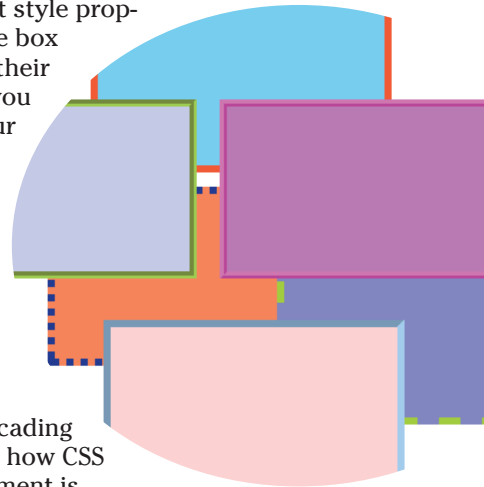
- ✓ Understanding the CSS box model
- ✓ Using the eight style categories
- ✓ Working with style attributes
- ✓ Creating custom link styles with CSS
- ✓ Extending your knowledge of CSS with online resources

In earlier chapters of the book, you find out that CSS is the best way to add formatting to the elements on your Web pages, from font sizes to image placement to link colors. And, in Book III, Chapter 3, you discover the fundamentals of Cascading Style Sheets and how to apply CSS styles to your pages.

As a complement to the previous chapter, this chapter is designed to help you with choosing the right attributes for all of your styles so that you can best style and position your content with CSS. Here you are introduced to the CSS box model concept and the eight different style property categories of CSS. A strong understanding of the box model along with these style categories (each have their own special set of CSS style declarations) can help you choose the attributes you need when you create your own style sheets. In addition, you find instructions on formatting the different elements on a page with CSS, and a helpful list of the best online CSS resources available, should you decide you want to start using the more advanced capabilities of CSS.

Working with the CSS Box Model

When styling and positioning your content with Cascading Style Sheets, it helps to understand the logic behind how CSS handles elements on a Web page. Each object or element is treated like a rectangular box that has margin space surrounding it, padding space inside of it, a border around it, and content inside of it, as illustrated in Figure 4-1.



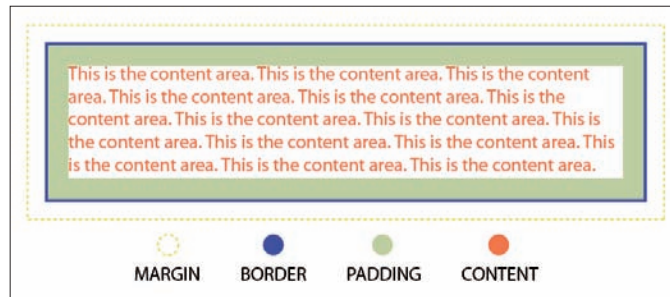


Figure 4-1: With the CSS box model, all Web page elements have margin, padding, border, and content areas.

When creating styles, the margins, padding, or border attributes are totally optional. When needed, they can be specified in the code, and when not explicitly specified, these areas have a default width of 0. These three attributes can be set to have any thickness and can be applied both uniformly and nonuniformly to the individual sides (top, right, bottom, left) of the rectangle. Though padding and borders must be set with 0 or positive values, margins can be assigned either positive or negative values. Margins and padding are always transparent, and borders can be set to any of several different styles.

When calculating the dimensions of the rectangular box surrounding an element, you must always measure by the outer margin box. In other words, if you have an element that has a specified width and height of 100 pixels by 200 pixels, and you then add a 10 pixel margin, 20 pixels of padding, and 5 pixels of border uniformly to all four sides of that element, the total width and height of that element would be 170 pixels by 270 pixels.



To help you create your Web page layouts, keep in mind that each element's rectangular box can contain any number of additional boxes and nested boxes. This simple fact allows for some pretty complex organization and styling, which is especially useful when laying out pages using layers.

Another important fact to remember is that there are two general kinds of boxes in CSS, *block* and *inline*:

- ✓ **Block:** These boxes are generated by certain Web page elements like paragraphs, headings, layers, lists, and tables. Blocks take up the full width available within the browser or containing element and add a new line of space both before and after the element. Blocks can also be containers for other block and inline elements, such as placing a paragraph of text inside a `<div>` tag.

- ✓ **Inline:** These boxes are generated by other Web page elements such as general text, images, anchors, and tags like ``, ``, and ``, and only take up as much space as is needed to display the element with its CSS styling. Inline boxes do not force elements onto new lines, but rather allow them to sit next to one another, side by side.

With CSS, however, you can override these default boxes and apply styling that forces certain elements to appear in the browser in precise ways. For example, lists are typically block elements that display in a vertically stacked list, like the one shown in Figure 4-2. Here's how the code for that illustration might look:

```
<ul>
  <li><a href="#">Home</a></li>
  <li><a href="#">About</a></li>
  <li><a href="#">Services</a></li>
  <li><a href="#">News</a></li>
  <li><a href="#">Contact</a></li>
</ul>
```

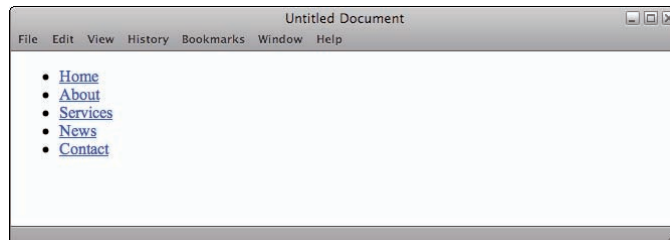


Figure 4-2: Lists are block elements that take up the full width of the browser.

By applying the `display:inline;` attribute to the `` tag redefine in the CSS and adding the class `navbar` to the `` tag as shown below, you can force the list items to appear in a horizontal row like a set of navigation links, like the example in Figure 4-3:

```
<style type="text/css">
  .navbar li {
    display:inline;
    color:#FFF;
    background-color:#F93;
    border: 1px solid;
    border-color:#f33 #900 #900 #f33;
    margin: 0;
    padding: 4px;
  }
</style>
```

```
<ul class="navbar">
  <li><a href="#">Home</a></li>
  <li><a href="#">About</a></li>
  <li><a href="#">Services</a></li>
  <li><a href="#">News</a></li>
  <li><a href="#">Contact</a></li>
</ul>
```

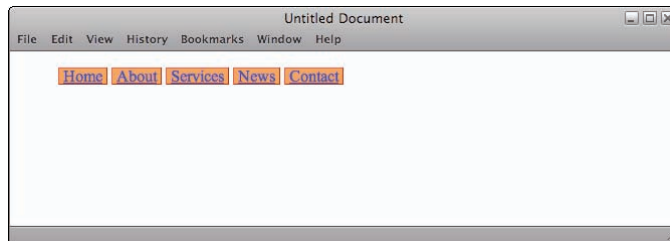


Figure 4-3: Lists can be turned into inline elements with CSS.

In addition to sitting inside one another, blocks can be positioned relative to other blocks and elements in any one of three ways:

- ✓ **Normal:** This is the default position of a block level element. Block boxes always flow vertically top to bottom, and inline boxes flow horizontally from left to right.
- ✓ **Float:** A block element can be set to float to the left or right of another element using the float property. For example, floats are often used to make text wrap around a left- or right-aligned image.
- ✓ **Absolute:** Block elements using absolute or fixed positioning are removed from the normal flow of a Web page and will appear in the exact position on a Web page as specified in the CSS.

Now that you have a general understanding about working with the CSS box model, you are ready to find out more about the different style properties available in CSS, as described in the next section.

Exploring the Eight Style Property Categories

To help you with choosing the right declarations for your styles, whether creating custom class, tag redefine, ID, or compound style, you should become familiar with the eight different CSS style categories. Then, when you know the category you need, choosing the style values from within it will be much more intuitive and easy.

The eight style categories in CSS are as follows:

- ✓ Type
- ✓ Background
- ✓ Block
- ✓ Box
- ✓ Border
- ✓ List
- ✓ Positioning
- ✓ Extensions

For each of these eight categories, your styles will take on the same general format, with a selector providing the name of the style and a declaration outlining the property-value pairs that make up the style.

The following sections describe the specific style rules in each of the eight style categories.

The type properties

The type properties include attributes that can modify the way text appears on a Web page. Attributes include font face, font size, font style, font color, font decoration, font weight, font variant, font case, and line height.

Font-family: Specify a font or font set that you are confident your visitors will have installed on their computers, regardless of whether they are visiting on a Mac or a PC. The set of “Web-safe fonts” that you can confidently choose from include, in no particular order, Arial, Verdana, Helvetica, Geneva, Georgia, Georgia Ref, Tahoma, Courier, Courier New, Times, Times New Roman, Palatino, Palatino Linotype, Trebuchet, Impact, serif, and sans serif.

For maximum control over the display of fonts on your Web site, select a *font set* or, in CSS terms, a *font-family*, rather than a single font. Font sets identify the preferred font face followed by alternate font choices should the first font be unavailable on the visitor’s computer. A typical font set might include Verdana, Arial, Helvetica, and sans serif, and would be written in the CSS code as follows:

```
p { font-family: Georgia, "Times New Roman", Times, serif; }
```

Font-size: Fonts can be specified to display in any numerical size in a variety of units, such as 1.5em or 12px. Standard font sizes include 10, 12, 14, 15, 18, 20, 24, and 36 pixels. However, you may set them to any size and unit desired, including px (pixels), pc (picas), pt (points), in (inches), mm (millimeters), cm (centimeters), em (ems), ex (exs), or % (percentage). Because precise sizes can override a browser's capability of increasing or decreasing font sizes, consider using ems or percentages instead of pixels or points.

```
p { font-size: 1.5em; }
td { font-size: 12px; }
```

Font-style: The style refers to an attribute of the font's face. The default style for most fonts is normal, which doesn't need to be specified in the CSS. Other options available in most fonts for the `font-style` include `italic` or `oblique`.

```
p { font-style: italic; }
```

Color: The color of a font can be any one of the 16.7 million colors that you can see on a 24-bit computer monitor, including the 216 Web-safe colors you find out about in Book I, Chapter 4. As long as the desired color has a hexadecimal value for on-screen rendering, the font can be that color. When specifying color for fonts or any other attributes, be sure to include the number symbol (#) before the hex number, such as #ffffff, so that the color will display correctly. Without the number symbol, the color attribute may be ignored in some browsers.

```
p { color: #990000; }
```

Line-height: Line height is what print designers often refer to as *leading*. Leading refers to the space between lines of type from the baseline of letters on one line to the baseline of letters on another line. The word comes from old printing-press days when strips of lead were used to create the spacing between blocks of metal type. The default line height (normal) automatically calculates a standard line height based on a ratio of font size to line height. To adjust the setting precisely, include this attribute with a numeric value in px (pixels), pc (picas), pt (points), in (inches), mm (millimeters), cm (centimeters), em (ems), ex (exs), or % (percentage).

```
p { line-height: 18px; }
```

Font-weight: By default, all fonts use a normal font weight. Other weight options include `bold`, `bolder`, `lighter`, and `bold` settings in increments between 100 and 900, with 400 being roughly equivalent to normal and 700 equal to bold.

```
p { font-weight: bold; }
```

Font-variant: The variant refers to whether the font will display in normal (uppercase and lowercase) font characters or in small caps.

```
p { font-variant: small-caps; }
```

Text-transform: Regardless of how text has been entered into a Web page, the case in which it displays in a browser can be modified with CSS using the `text-transform` attribute. Options include `capitalize`, `uppercase`, `lowercase`, and `none`. Choosing `capitalize`, for example, changes the text to display all words with initial capitals letters.

```
p { text-transform: capitalize; }
```

Text-decoration: This attribute specifies how text can be decorated. Most of the attributes, however, either aren't useful for the Web (`line-through` and `overline`) or aren't supported by all the different browsers (`blink`). The two decorations that are often useful, especially when creating styles for custom link states, are `underline` and `none`.

```
p { text-decoration: underline; }
```

Here's a brief overview of the five text-decoration styles:

- ✓ `underline`: This option, which is the default decoration for hyperlinks, displays an underline beneath the text.
- ✓ `overline`: This option adds an overline above the text.
- ✓ `line-through`: Choose this option to make text look like it's been struck through with a line.
- ✓ `blink`: This attribute makes styled text blink in the browser window. This attribute isn't supported in all browsers.
- ✓ `none`: Use this option to remove any default type decoration, such as removing the underline on a hyperlink.

The background properties

You can apply background properties to a number of different objects on a Web page, including the whole page, a particular layer, a table, a table cell, and even text.

Background-color: A background color can be applied to most objects on a page, including text, tables, table cells, layers, and the body of a page using a hexadecimal value. When specifying color for any style, remember to add the number symbol (`#`) before the hex value, as in `#cc9900`, for best browser display results.

```
p { background-color: #33ff00; }
```

Background-image: Images, like a background color, can be applied to the background of many different objects on a Web page, including the body of a page, tables, table cells, and layers. You can control how the image tiles (repeats) by using the `repeat` attribute.

```
.mylayer { background-image: url(images/car.gif); }
```

Background-repeat: The `repeat` attribute tells a browser how the background image should be repeated in the area it's filling. By default, and unless otherwise specified, all backgrounds will tile vertically and horizontally to fill the entire background space of the styled tag or object.

```
body {  
background-image: url(images/zigzag.gif);  
background-repeat: repeat-x;  
}
```

The `repeat` attribute has four variables, each of which is illustrated in Figure 4-4:

- ✔ `repeat`: This option is the same as the default setting for background images and tiles the background image both horizontally and vertically.
- ✔ `repeat-x`: Use this option when you want the background image to tile only along the horizontal axis. If desired, use it in conjunction with the horizontal and/or vertical `background-position` attribute.
- ✔ `repeat-y`: Use this option when you want the background image to tile only along the vertical axis. If desired, use it in conjunction with the horizontal and/or vertical `background-position` attribute.
- ✔ `no-repeat`: This setting displays the background image as a single static image with no repeating in either direction.

Background-attachment: This attribute refers to how the background image interacts with the content above it. The background image can behave in three different ways — `scroll`, `fixed`, and `inherit` — but not all three are consistently supported by all browsers, so be sure to test whichever option you select in a variety of browsers and browser versions on both Mac and PC platforms.

```
body {  
background-image: url(images/biodiesel.gif);  
background-attachment: fixed;  
background-repeat: repeat-y;  
}
```

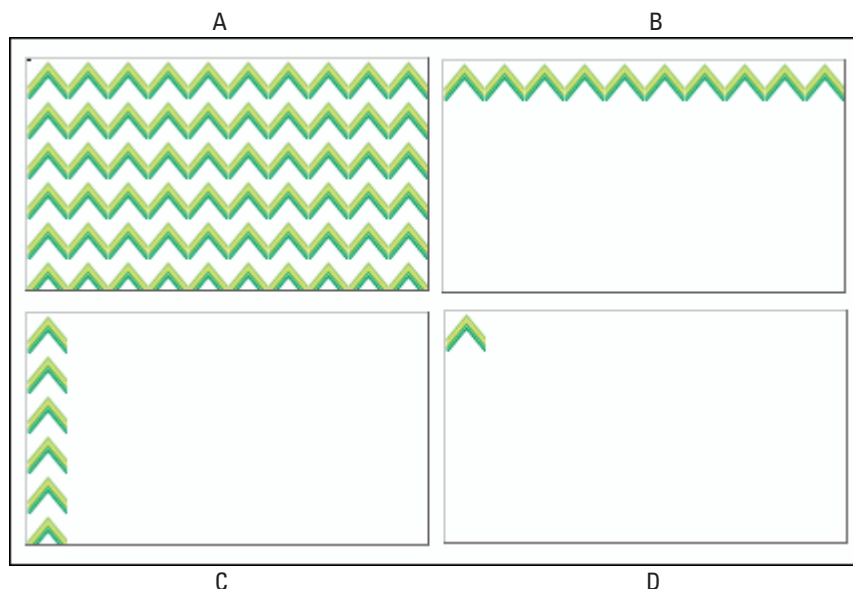



Figure 4-4: Use CSS to control how images repeat within a container. A. repeat, B. repeat-x, C. repeat-y, D. no-repeat.

Here is an explanation of the different background attachment styles:

- ✓ **scroll:** This is the default option for how the background image is attached to the page, which works the same whether the attribute is specified or unspecified in the CSS. With this option, the background image scrolls along with any text and other objects on the page.
- ✓ **fixed:** The fixed attribute keeps the background image fixed to the browser window while text and other objects on the page scroll past it.
- ✓ **inherit:** When you specify this option, the background image inherits the attachment rule, whether scroll or fixed, from its parent container, as with a table cell inside a table.

Background-position (X): Set the horizontal background-position attribute to control where in the browser window the background image will display and repeat. Choose left, center, or right or type any value in px (pixels), pc (picas), pt (points), in (inches), mm (millimeters), cm (centimeters), em (ems), ex (exs), or % (percentage).

```
p {
background-image: url(images/recycle.gif);
background-repeat: repeat-x;
```

```
background-position: left;
}
```

Background-position (Y): Set the vertical `background-position` attribute to control where in the browser window the background image will display and repeat. Choose `top`, `center`, or `bottom` or type any value in `px` (pixels), `pc` (picas), `pt` (points), `in` (inches), `mm` (millimeters), `cm` (centimeters), `em` (ems), `ex` (exs), or `%` (percentage).

```
p {
background-image: url(images/gogreen.png);
background-repeat: repeat-x;
background-position: center;
}
```

When both the horizontal and vertical background positions need to be specified in the CSS, list them together separated by a space:

```
p {
background-image: url(images/earthsafe.jpg);
background-repeat: repeat-x;
background-position: left center;
}
```

The block properties

Block properties control the alignment and spacing of objects on a page through their tags and attributes. Blocks, which you are introduced to in the first section of this chapter, include text, content inside `<div>` tags (both with and without positions specified), tags using the `display:block` style, and images or paragraphs set with absolute or relative positions.

Word-spacing: To adjust the spacing between individual words, use any positive or negative number in `px` (pixels), `pc` (picas), `pt` (points), `in` (inches), `mm` (millimeters), `cm` (centimeters), `em` (ems), `ex` (exs), or `%` (percentage), such as `word-spacing: 1pt;`

```
p { word-spacing: 2px; }
```

Letter-spacing: With this attribute, you can uniformly increase or decrease the space between characters by specifying a positive or negative value in `px` (pixels), `pc` (picas), `pt` (points), `in` (inches), `mm` (millimeters), `cm` (centimeters), `em` (ems), `ex` (exs), or `%` (percentage), such as `letter-spacing: 1em;`. Note that changing the `letter-spacing` attribute overrides any pre-existing text justification.

```
p { letter-spacing: 1.5em; }
```

Vertical-align: You can vertically align text along the text baseline, `sub` (script), `super` (script), `top`, `text-top`, `middle`, `bottom`, and `text-bottom`,

or by any value, positive or negative, in px (pixels), pc (picas), pt (points), in (inches), mm (millimeters), cm (centimeters), em (ems), ex (exs), or % (percentage), such as `vertical-align: super;`

```
p { vertical-align: top; }
```

Text-align: This option can be applied only to text. Alignment options include `left`, `right`, `center`, or `justify`.

```
p { text-align: center; }
```

Text-indent: Also to be used only with text, this style rule creates a first-line indent that can be set to any positive or negative value in px (pixels), pc (picas), pt (points), in (inches), mm (millimeters), cm (centimeters), em (ems), ex (exs), or % (percentage), such as `text-indent: 12px;`



To indent nontext objects on a page, use the `<blockquote>` tag in the HTML or, better still, wrap the content with a pair of `<div>` tags and style the opening `<div>` tag with `margin` and/or `padding` style attributes.

```
p { text-indent: 10px; }
```

White-space: White space inside or around an object can be displayed in three different ways: `normal`, `pre`, and `nowrap`. Choose `normal` to ignore any white space, `pre` to leave the white space in with the text as it was coded, or `nowrap` to force any text to wrap only if the code has line break (`
`) tags.

```
p { white-space: pre; }
```

Display: This attribute controls how the styled object displays in the browser. Value options include `block`, `compact`, `inline`, `list-item`, `marker`, `none`, `run-in`, and `table`.

```
p { display: none; }
```

Choose from any of the following settings:

- ✓ `none`: Use this option to hide a styled element from displaying in the browser. This option is extremely useful when creating multiple style sheets so that some elements can be hidden from view on one device but not another, as with a secondary CSS for the print media type.
- ✓ `inline`: Use this option to display the object styled inline with other elements, often in the same block.
- ✓ `block`: This turns any styled element into a block, after which further block-styling attributes may be applied. Block-level elements take up the

full width of available space, including line space above and below the element, similar to the way paragraphs have space above and below them.

- ✓ `list-item`: This option converts styled text into a bulleted list, similar to `` and `` tags.
- ✓ `run-in`: This feature is either unsupported, incompletely supported, or fully supported, depending on the browser. Currently the only browsers that provide full support include IE 8 and Opera 5+ on a Mac. Add the `run-in` attribute to force a block box following a run-in box to become an inline box of the block box.
- ✓ `inline-block`: Use this option to make a block behave as an inline block with a specified width. This feature is supported by Firefox 3.0+, Safari 3.0+, Opera 9.5+, Konqueror, and IE 8.
- ✓ `compact`: This option is a still quite buggy and is currently only haphazardly supported in Opera 5+, Konqueror 3.5.7, Safari 3.1 for Windows, and iPhone. When specified, this attribute forces other blocks in the code after it to display along its side.
- ✓ `marker`: This converts content in a display block into a marker box, using the `:before` or `:after` pseudo-element, inside which you can further style the content.
- ✓ `table`: Though extremely buggy unless you're using Firefox, Opera, or Konqueror, you should be able to use this attribute to display elements inside a table without having to use HTML tables. In theory, any nested elements would display as if they were `table-row` and `table-cell` elements. Additional display table values for this property include `inline-table`, `table-row-group`, `table-header-group`, `table-footer-group`, `table-row`, `table-column-group`, `table-column`, `table-cell`, and `table-caption`.
- ✓ `inherit`: When you specify this option, the styled object inherits the display value from its parent element.



Blocks are one of the property categories that have a lot of capabilities beyond the basic ones described here. Not all properties will be consistently supported by all browsers, but depending on the target audience, some of them might be perfectly suited for a particular client. To find out more about display properties, review the block information pages at the W3C Web site. For further discussion of block display attributes, visit the W3C Web site:

- ✓ www.w3.org/TR/REC-CSS2/visuren.html#display-prop
- ✓ www.w3.org/TR/REC-CSS2/tables.html#value-def-table-column
- ✓ www.w3.org/TR/REC-CSS2/generate.html#markers

The box properties

With the box properties (refer to Figure 4-1), you can position styled objects anywhere in a browser window, position objects relative to the other objects on the page, and apply the padding and margin box style rules selectively to any of or all the four sides of the styled object, such as left and bottom or top, left, and right. When styling less than all four sides, be sure to add 0 values to the sides that should not contain values, rather than leaving them blank.

Width/Height: Use the `width` and `height` attributes to style content that sits inside a container such as a table, table cell, or layer. Set the attributes to `auto` to force the size of the object to match the contents of the object, or enter any value, positive or negative, in `px` (pixels), `pc` (picas), `pt` (points), `in` (inches), `mm` (millimeters), `cm` (centimeters), `em` (ems), `ex` (exs), or `%` (percentage).

```
#rings { height: auto; width: 475px; }
```

Float: Use this style to control the side (`left`, `right`, or `none`) on which other objects will float around the styled object.

```
.saleitems { float: right; }
```

Clear: Often used in conjunction with the `float` property, this style attribute controls whether other objects can appear next to the styled object. Variables for this attribute include `left`, `right`, `both`, and `none`. For example, when a layer appears on the side of an object with the clear side specified, that object will be bumped to the area below the layer.

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

Padding: This property is like the margin, only with padding you apply extra space between the styled object and any border surrounding it, as with a sentence or a word inside a table cell. Set the padding size on the left, right, top, and/or bottom sides using any value, positive or negative, in `px` (pixels), `pc` (picas), `pt` (points), `in` (inches), `mm` (millimeters), `cm` (centimeters), `em` (ems), `ex` (exs), or `%` (percentage), such as `padding: 10px 0px 0px 10px;`. When uniform sizes are applied to all four sides of the styled object, only one value, as in `padding: 10px;`, needs to be listed in the declaration:

```
.sidebarimage { padding: 12px; }
```

Margin: Use the `margin` property to add or subtract additional space between the page edge (or parent container) and the object being styled, such as the area surrounding a word or layer. Set the margin size on the left,

right, top, and/or bottom sides using any value, positive or negative, in px (pixels), pc (picas), pt (points), in (inches), mm (millimeters), cm (centimeters), em (ems), ex (exs), or % (percentage). You may also use the `auto` value on both sides of a styled object to center the object within its parent container.

```
#contact { margin: 0px auto 0px auto; }
```

The border properties

Border properties define the color, style, and width of borders around any styled object. Because borders can go on all four sides of an object, each side can have totally different border attributes! For best results, as with margins and padding, be sure to add a 0 or None to any side property not being styled:

```
.tablecell {  
  border-top: 0px none;  
  border-right: 2px dotted #069;  
  border-bottom: 1px solid #09C;  
  border-left: 2px dotted #069;  
}
```

Style: You can specify borders in any of the following styles: `solid`, `dotted`, `dashed`, `double`, `groove`, `ridge`, `inset`, `outset`, or `none`. **Style must be specified in conjunction with the color and width.**

```
.tablecell { border: 2px dashed #330066; }
```

Width: You can set options for the thickness of the border to `thin`, `medium`, or `thick`. For more precise measurements, use as any value in `px` (pixels), `pc` (picas), `pt` (points), `in` (inches), `mm` (millimeters), `cm` (centimeters), `em` (ems), `ex` (exs), or `%` (percentage), such as for the top, right, bottom, and left sides of the styled object. Select the width along with the color and style.

```
.tablecell { border: 1px dotted #660033; }
```

Color: To colorize the border attribute, specify the hexadecimal value of the desired color and be sure to include the number symbol (`#`) before the hex value. Also include a style type and width size for the border.

```
.tablecell { border: 5px solid #003366; }
```

Figure 4-5 shows examples of each of the different border styles.

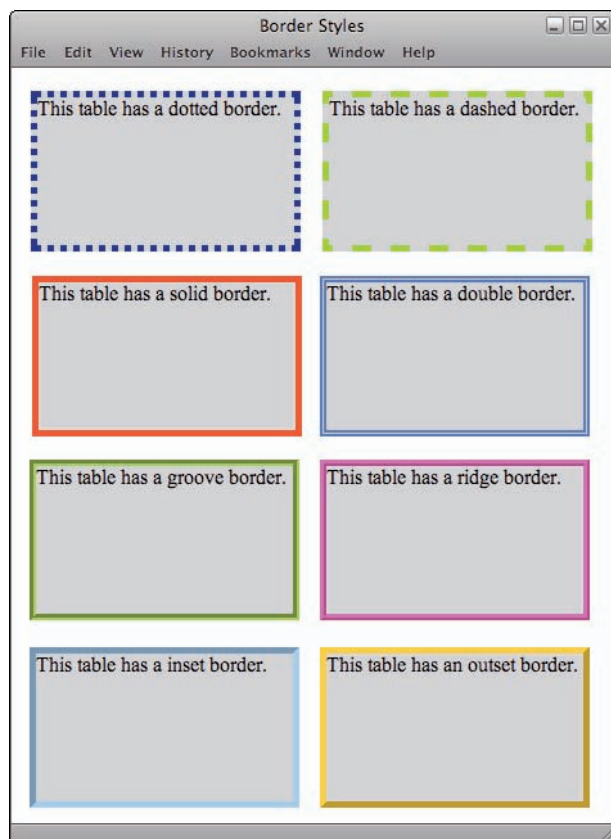


Figure 4-5: Borders come in eight flavors and can be applied to any of or all the sides of a container.

The list properties

Lists styled with CSS are much more robust than lists styled with standard list HTML formatting. With CSS, you can easily select the list type for both numbered and bulleted lists, set the position of the bullets relative to the contents within the list, and even choose to use your own graphic for the bullet image.

List-style-type: For ordered lists, set the list type to `decimal`, `lower-roman`, `upper-roman`, `lower-alpha`, `upper-alpha`, or `none`. When creating unordered lists, choose the `disc`, `circle`, or `square` list type. Figure 4-6 shows examples of each of these list types.

```
li { list-style-type: circle; }
```

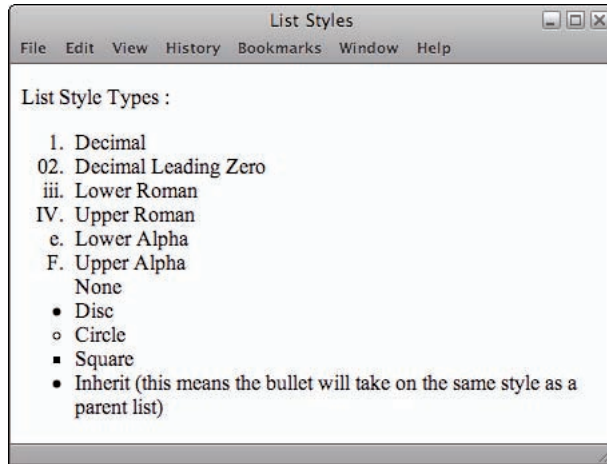



Figure 4-6: Use CSS to set the list type for your ordered and unordered lists.

List-style-image: To use your own custom image as a bullet, enter the location and filename of the desired image as an attribute for the unordered list tag. You can also remove the bullet part of list styling by using the property value none.

```
ul { list-style-image: url(images/mybullet.gif); }
```



TIP

To create a list with different graphics for each list item, create ID styles for each list item and then append each `` tag with the matching `id` attribute, as illustrated in Figure 4-7 and shown in the following code example, which also makes each item in the list a hyperlink:

```
#redarrow {
list-style-image: url(redarrow.gif);
}
#bluearrow {
list-style-image: url(bluearrow.gif);
}

<ul>
<li id="redarrow"><a href="http://www.thiswebsite.com" />Go
to This Site</a></li>
<li id="bluearrow"><a href="http://www.thatwebsite.com" />Go
to That Site</a></li>
</ul>
```



Figure 4-7: For customized unordered lists, use your own bullet graphics.

List-style-position: With the position property, you can position the bullet relative to content inside each list item. The position can be located either inside or outside the text. As illustrated in Figure 4-8, when set to `inside`, the text wraps beneath the bullet along the left margin, and when set to `outside`, the bullet stays outside any wrapped text, like a hanging indent.

```
li { list-style-position: outside; }
```

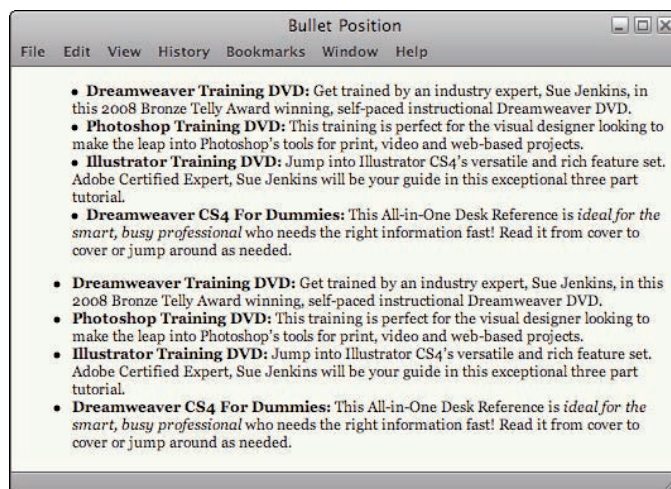


Figure 4-8: Bullets may be positioned inside (top) or outside (bottom) list item content.

The positioning properties

The positioning attributes are used primarily to style layers using the `<div>` tag, though you can also use them to style the position of an image, container, or block-level element within the browser. For layers, both the contents as well as the container can be styled with attributes in this category.

Position: Determines how a styled element should be positioned in a browser window. Specify whether the position is `absolute`, `fixed`, `relative`, or `static`.

```
#footer { position: relative; }
```

When setting the footer position, use one of the following style attributes:

- ✓ **absolute:** Sets the element's position absolutely based on the numeric values entered for the element's placement relative to the closest absolutely or relatively positioned parent element, or to the upper-left edge of the browser window.

- ✓ **fixed:** Sets the element's position absolutely based on the numeric values entered for the element's placement relative to the upper-left edge of the browser window.
- ✓ **relative:** Sets the element's position by the numeric values entered for the object's placement relative to the styled element's position in the file's text flow.
- ✓ **static:** Sets the element in an exact location within the text flow.

Width: Use this attribute to set the width of an element, such as a layer or other container, by using `px`, `pt`, `in`, `cm`, `mm`, `pc`, `em`, `ex`, `%`, or `auto`, which tells the element to match the size of the contents.

```
#layer1 {  
width: 760px;  
}
```

Height: Use this attribute to set the height of an element, like a layer, by using `px`, `pt`, `in`, `cm`, `mm`, `pc`, `em`, `ex`, `%`, or `auto`, which tells the element to match the size of the contents.

```
#layer1 {  
height: 100px;  
}
```

Visibility: This attribute determines the initial visibility value of an element, which can be set to `hidden`, `inherit`, or `visible`, when the page first opens in a browser window.

```
#layer1 {  
visibility: hidden;  
}
```



Visibility should not be confused with the `display` attribute, which determines whether an element should be treated as a block or inline element, or be completely ignored by the browser with the `display: none;` attribute. With visibility, you are dealing with the initial visibility state of an element, such as a layer, when the page first loads in the browser. This attribute can also be toggled on and off by using JavaScript to hide and show elements on the page and add a bit of interactivity for the visitor.

To modify the visibility of your element, add the `visibility` property to your CSS selector with one of the following values:

- ✓ **hidden:** This option hides a layer from displaying when a page initially opens in a browser.

- ✓ **inherit**: This option causes any layer to inherit the visibility of a parent layer; if a parent doesn't exist, the layer will be visible. When the visibility is unspecified, `inherit` is the default attribute.
- ✓ **visible**: Choose this option to force the layer to be visible, regardless of any parent layer's visibility setting, when the page first opens in a browser window.

Z-index: This attribute specifies a layer's stacking order relative to any other layers on the page as they are viewed in a browser. Set the z-index to `auto` when the number is noncritical, set it to `inherit` to make the layer inherit a parent layer's z-index value, or enter a specific positive or negative number, such as 100, when the number is important relative to the other layers on the page. The higher the number, the closer the layer will appear to the front or top of the page closest to the visitor; the lower the number, the closer the layer will appear to the browser's background.

```
#lastchance {  
  z-index: 4;  
}
```

Figure 4-9 shows an example of several layers on a page with different z-index values.

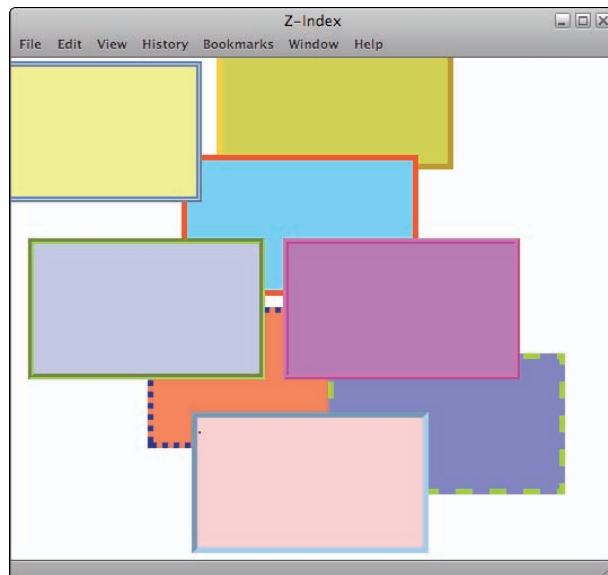


Figure 4-9: Each layer on a page may have its own z-index to represent the stacking order of the layers on the page.

Overflow: The `overflow` setting can be used to specify what happens to any contents within a layer that exceed the size of the layer as defined within the CSS. Set the `overflow` attribute to `auto`, `hidden`, `scroll`, or `visible`.

```
#aboutus {  
overflow: scroll;  
}
```

Here is a description of each of the overflow values you can use:

- ✓ `auto`: Choose this option to have the browser automatically add scroll bars to the layer if and only if the contents exceed the layer's defined width and height.
- ✓ `hidden`: When this option is selected, the size of a layer is maintained and any content exceeding that size is cut off or clipped from view in a browser window.
- ✓ `scroll`: Choose `scroll` to add scroll bars to the layer, regardless of whether the content fits or exceeds the layer's width and height. This attribute isn't supported by all browsers, so be sure to test it before publishing.
- ✓ `visible`: Choose this option to have the layer expand vertically and then horizontally, if needed, to fit any contents that exceed the specified layer width and height so that all the contents are visible.

Placement (Left, Top, Right, Bottom): Use the placement attribute to specify the exact size and location (based on the specified type) of a styled element in a browser window. By default, the pixel size and placement of an element are specified for the top, left, bottom, and right edges. However, you can use other units (including `pt`, `in`, `cm`, `mm`, `pc`, `em`, `ex`, or `%`) if desired, or set the value for any of the sides to `auto`. For layers, if the contents within that layer exceed the specified size, the layer will expand to fit the contents.

```
#specialitems {  
left: 500px;  
top: 300px;  
right: 0;  
bottom: 0;  
}
```

Clip: Use this attribute to specify a smaller visible rectangular area within a layer relative to that layer's upper-left edge. When clipped, the hidden area can be manipulated with JavaScript or other programming to create special effects that can hide and show the hidden content. Set values for the clipped area to the top, left, right, and bottom edges of the layer by using `px`

(pixels), pc (picas), pt (points), in (inches), mm (millimeters), cm (centimeters), em (ems), ex (exs), or % (percentage), or a value of auto.

```
#bunnygame {
clip: rect(10px,100px,0px,60px);
}
```

The extension properties

With extension properties, you can change the way the cursor displays in different circumstances, create page breaks, and add special-effect filters to certain elements on the page. Unfortunately, since their inception, very few of them are supported by the most popular browsers. If you'd like to use any of these attributes, test them in as many browsers as you can on both Mac and PC platforms to make sure that they work and/or fail in an acceptable way.

Page-break-before/-after: This attribute forces a page break when printing a page, either before or after the object styled with this attribute. Choose left, right, always, or auto for before and/or after the desired styled object, as in the following:

```
#sidebar3 {
page-break-before: always;
page-break-after: left;
}
```

Cursor: You can specify a different cursor to appear when a visitor mouses over an object that is styled with this attribute. Choose from crosshair, text, wait, default, help, e-resize, ne-resize, n-resize, nw-resize, w-resize, sw-resize, s-resize, se-resize, and auto. As long as your target audience will be using IE 4.0+ (which should be pretty much everyone but less than 1% of all Internet users), these effects should be supported by all newer browsers.

```
#helpmenu {
cursor: help;
}
```

Filter: You can choose from several special-effect filters, including Alpha, BlendTrans, Blur, Chroma, DropShadow, FlipH (flip horizontal), FlipV (flip vertical), Glow, Gray, Invert, Light, Mask, RevealTrans, Shadow, Wave, and Xray. Most filters require numeric input such as the mask filter, which must contain the hexadecimal value of the color for the mask, as in filter: Mask(Color=#ffcc33);.

```
#details {
filter: invert;
}
```

While it is true that these filters can do some cool and unusual things, to date, none of them have worked in any browsers other than IE. To see an example of a few of them, open the following link within an IE browser: www.xentrik.net/css/filters.php.

Styling the Content on Your Pages

Now that you know all about CSS styles and style sheets, you should be able to quickly and efficiently style the content on your pages. In the following sections, you find out about styling the different areas of your pages as well how to perform a few advanced CSS techniques.

Styling paragraphs, headers, and footers

When you are styling paragraphs, headers, and footers, most of the work can be automatically accomplished by creating tag redefine styles for the `<p>` tag and however many heading tags you intend to use, such as `<h1>`, `<h2>`, and `<h3>`:

```
p {
  font-family: Georgia, "Times New Roman", Times, serif;
  font-size: 12px;
  color: #039;
  background-color: #FCEBB6;
}
```

In some circumstances, you may want to create a custom style and apply that selectively to individual words or phrases throughout your text using the `class="stylename"` attribute as part of the opening tag that surrounds the content, whether that be a paragraph, heading, or span tag:

```
.stylename {
  font-family: Georgia, "Times New Roman", Times, serif;
  font-size: 10px;
  font-style: italic;
  font-weight: bold;
  font-variant: small-caps;
  color: #006;
}
```

```
<p class="stylename">This entire paragraph will be styled
  using the stylename class, which overrides the redefined
  paragraph style.</p>
```

When styling footer content, you need to create styles to format all the content that goes there. To start, many designers isolate the footer content into its own layer and then style that layer using an `id` style:

```
<div id="footer">
<a href="index.html" target="_self">Home</a> | <a
  href="about.html" target="_self">About</a> | <a
  href="services.html" target="_self">Services</a> | <a
  href="contact.html" target="_self">Contact</a><br />
&copy; 2009 CompanyName.com. All Rights Reserved.
</div>
```

Within that `id` style, you could set the font family, size, weight, and color for the layer's contents; apply a background color and border attributes to the layer; and set the layer's width, height, z-index, and visibility.

```
#footer {
  font-family: Verdana, Geneva, sans-serif;
  font-size: 10px;
  font-weight: bold;
  color: #FFF;
  background-color: #666;
  padding: 10px;
  height: auto;
  width: 100%;
  border: 1px solid #000;
  visibility: visible;
  z-index: 10;
}
```

After that is done, you can begin to create individual custom styles, ID styles, custom hyperlinks, and advanced combinators to style the various parts of the footer content. For instance, you could create a set of hyperlink styles that would only be applied to the links within the footer:

```
#footer a:link {
  color: #CC0;
}
#footer a:visited {
  color: #0C6;
}
#footer a:hover {
  color: #FFF;
  background-color: #CC0;
  text-decoration: none;
}
#footer a:active {
  color: #FFF;
  background-color: #0C6;
  text-decoration: none;
}
```


Styling all four sides

For any styles you create that include declarations from the box, border, or positioning property categories, pay special attention to the rules that will be applied to container tags, like a layer, table, or table cell, that include top, bottom, left, and right sides. Rather than only adding a rule to just the sides you want to style, you should enter values for all four sides to improve the way the page is rendered in a variety of browsers.

Typically your options will be to style the unused sides with a zero value or a value of None. For example, the `padding` attribute in the box properties category can take on values for `top`, `bottom`, `left`, and `right`. To add padding only on the top and left sides of an object, be sure to enter 0 as the unit for the right and bottom sides of the style definition:

```
.mystyle {  
  padding-top: 5px;  
  padding-right: 0px;  
  padding-bottom: 0px;  
  padding-left: 5px;  
}
```

Likewise, the `border` style attribute in the border properties category has four sides. However, to specify that you don't want a particular side styled, use the `none` property value:

```
.test {  
  border-top: 0px none;  
  border-right: 1px solid #039;  
  border-bottom: 0px none;  
  border-left: 1px solid #039;  
}
```

Figure 4-10 demonstrates how a layer styled with the `#footer` and `link` CSS styles shown here would look in a browser.

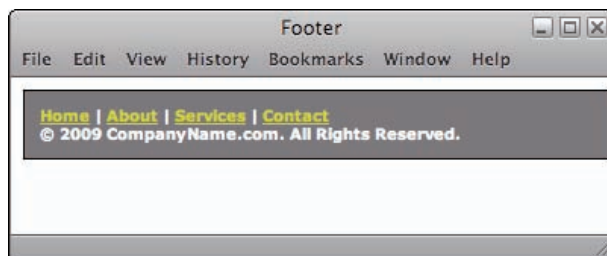


Figure 4-10: Use CSS to create custom styles for the links and other text in the page footer.

Styling lists and tables

Styling lists and tables is a bit different than styling content out in the body of the page, because both lists and tables have specific tags that can be redefined to control how content sits within those structures.

Lists

With lists, you can select type attributes, such as font, size, and alignment, and apply a background color and border to each list item. In addition, you can select the desired list style type, select the bullet's position relative to the list item content, and choose to use your own custom bullet graphic.

The secret to keeping all the styles in order is to provide `id` attributes to the `` and `` tags, as well as to wrap the entire list between layer tags so that you can easily create advanced combinators and dependent selectors. This is especially useful when creating navigation systems with list formatting:

```
<div id="navigation">
  <ul>
    <li id="nav1"><a href="#"><span>Home</span></a></li>
    <li id="nav2"><a href="#"><span>About</span></a></li>
    <li id="nav3"><a href="#"><span>Services</span></a></li>
    <li id="nav4"><a href="#"><span>Contact</span></a></li>
  </ul>
</div>
```

Here are the styles you might create for your list in the CSS. This code includes graphics for both the background of the navigation bar and the hover state graphics, all of which should have text to label each of the navigation buttons:

```
span {
  display: none;
}
ul {
  position: relative;
  width: 412px;
  background: url(images/bg_nav.gif) no-repeat;
  height: 92px;
  list-style-type: none;
  margin: 0;
  padding: 0;
}

li#nav1 a, li#nav2 a, li#nav3 a, li#nav4 a {
  background: transparent;
  position: absolute;
  width: 100px;
  height: 25px;
  bottom: 0;
  text-decoration: none;
}

/***** adjusted values for IE6 *****/
* html*li#nav1 a, * html*li#nav2 a, * html*li#nav3 a, * html*li#nav4 a { bottom:
-1px; }
```

```
li#nav1 a { left: 0px; background: url(images/home_hover.gif) 0 0 no-repeat;}
li#nav2 a { left: 105px; background: url(images/about_hover.gif) 0 0 no-repeat;
}
li#nav3 a { left: 205px; background: url(images/services_hover.gif) 0 0 no-
repeat; }
li#nav4 a { left: 305px; background: url(images/contact_hover.gif) 0 0 no-
repeat; }
li#nav1 a:hover, li#nav2 a:hover, li#nav3 a:hover, li#nav4 a:hover { background-
position: 0 -40px; }
```

Figure 4-11 illustrates one way that you might create graphics for this type of CSS navigation list. To find out more about navigation systems, turn to Book III, Chapter 6.

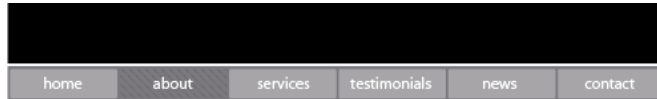


Figure 4-11: Create custom list styles, including simple navigation bars, with CSS.

Tables

With tables and CSS, you can style the entire table, individual table cells, whole rows or columns, and the contents of each cell. Begin by inserting a table and adding the desired content to each of the table cells. Apply any cell padding and cell spacing to the table as desired. Then to make the process of styling your table easier, be sure to add an `id` attribute to the opening table tag and the `class` attribute to the first opening table row tag:

```
<table id="longjohns" width="300" border="0" cellpadding="10"
cellspacing="0">
  <tr class="longjohnsth">
    <td width="220">Children's Silk Long Johns</td>
    <td width="80">Price</td>
  </tr>
  <tr>
    <td>Crewneck Long Underwear Top</td>
    <td>$17.95</td>
  </tr>
  <tr>
    <td> Long Underwear Pant</td>
    <td>$17.95</td>
  </tr>
</table>
```

Next create a style for that `id`, which can be used to format the main parts of the table, such as the table's size, background color, and border:

```
#longjohns {
  background-color: #996;
  height: 200px;
  width: 300px;
  color: #FFF;
}
```

After the table's main formatting is complete, you can create advanced combinators (which are also sometimes called *dependent selectors*) and custom styles for the different table cells, as illustrated in Figure 4-12.

```
#longjohns tr td {
  border-top-width: 0px;
  border-right-width: 0px;
  border-bottom-width: 1px;
  border-left-width: 0px;
  border-top-style: none;
  border-right-style: none;
  border-bottom-style: dotted;
  border-left-style: none;
  border-bottom-color: #FFF;
}
.longjohnsth {
  font-weight: bold;
  background-color: #C9C9AD;
  color: #000;
}
```

Children's Silk Long Johns	\$Price
Crewneck Long Underwear Top	\$17.95
Long Underwear Pant	\$17.95

Figure 4-12: Use CSS to create custom styles for the individual cells within your table.

Styling images and AP elements (layers)

Images need not just sit on your page exactly where you stick them. Instead, use CSS with your images to position them relative to other content, and add padding and border attributes to one or more sides.

To style an image, you have three options:

- ✓ Create a tag redefine style for the `` tag to create a style that will apply to all images on your page(s).
- ✓ Create a custom class style that can be selectively applied to individual images as needed.

```
img { border: 2px dotted #069; }
```

```
.myimg { border: 2px dotted #069; }
```

```

```

- ✓ Create an ID style that will automatically style the image with a matching id attribute.

```
#mystyle { padding: 5px; border: 5px dashed #F90; }


```

In addition to these general image-formatting options, you may also add CSS style rules to your images that control how other objects sit on the page relative to the styled image. Using the `float` and `clear` style attributes, you can control which side (left, right, or none) other objects will float around the styled object, or be pushed into the area below the styled object. For example, to make a series of images float and clear to the right of a block of text, as shown in Figure 4-13, create a custom style with the `float` and `clear` attributes:

```
.imgfloatright {
  margin: 0 0 10px 10px;
  float: right;
  clear: right;
}
```

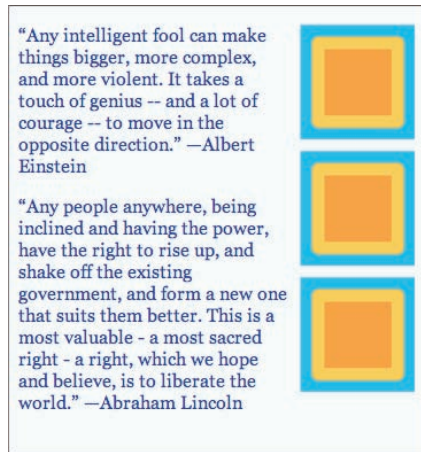


Figure 4-13: The `float` and `clear` styles can make objects float beside other content.

Then apply that style selectively to each image within the text block. The images should sit at the top of the block of text, as shown here:

```



<p>Any intelligent fool can make things bigger, more complex, and more
violent. It takes a touch of genius -- and a lot of courage -- to move in
the opposite direction.&#8221; &#8212;Albert Einstein </p>
<p>Any people anywhere, being inclined and having the power, have the
right to rise up, and shake off the existing government, and form a new one
that suits them better. This is a most valuable - a most sacred right - a
right, which we hope and believe, is to liberate the world.&#8221;
&#8212;Abraham Lincoln </p>
```

Formatting layers with CSS is a complex affair, because you can pretty much control everything related to the style and positioning of a layer. In most instances, designers begin by giving each of their layers a unique ID, such as `id="sidebar"`. After the layer has been named in the HTML, an ID style can be created to control the positioning, width, height, z-index, font family, font size, font color, background color, padding, and border of the layer.

The following declaration shows example CSS for an ID style:

```
#sidebar {
  position: relative;
  width: 350px;
  height: 200px;
  z-index: 1;
  font-family: Arial, Helvetica, sans-serif;
  font-size: 16px;
  background-color: #F5F5F5;
  color: #666;
  border: 5px double #999;
  padding: 20px;
}
```

Beyond that, the layer contents can be further styled with any combination of tag redefines, ID styles, custom class styles, and advanced combinators.

Finding CSS Resources Online

When you're first getting to know CSS, you have so much to take in and understand that the application of styles to a Web page often gets limited to styling text. Hopefully, this chapter has helped you to understand how robust CSS can be so that you can really take charge of your page layouts with it. With CSS, you can do the following:

- ✓ Style page attributes like margins, background color, and background image
- ✓ Style text like paragraphs and headings
- ✓ Style tables and table cells including color, border, and background images
- ✓ Style the four link states for all hyperlinks
- ✓ Style lists and bullets
- ✓ Style and position your layers and their contents

And that's only the tip of the iceberg!

One of the best ways to discover more about CSS is to look at the code developed by others. By reverse-engineering the CSS and HTML code, you can really get an in-depth understanding of how positioning and styling of content with CSS work. Then you can implement the same or similar methods in your own Web pages.



Several amazing Web sites showcase innovative and unique CSS styling, some of which also allow you to view the source code behind the work. My favorite resource is Dave Shea's www.csszengarden.com, shown in Figure 4-14, but you can easily find many others by searching for "CSS," "CSS tutorials," "CSS tips," and so on. Table 4-1 lists some great sites you should definitely visit and bookmark.



Figure 4-14: Check out other designers' CSS for free at www.csszengarden.com.

Table 4-1 CSS Online Resources	
<i>CSS Resource Name</i>	<i>CSS Resource Web Address</i>
W3Schools Tutorial	www.w3schools.com/css/default.asp
W3C Tutorial	www.w3.org/MarkUp/Guide/Style
W3C's CSS	www.w3.org/Style/CSS

<i>CSS Resource Name</i>	<i>CSS Resource Web Address</i>
CSS Zen Garden	www.csszengarden.com
CSS Vault	www.cssvault.com
CSS Beauty	www.cssbeauty.com
Max Design	http://maxdesign.com.au
Site Point	http://reference.sitepoint.com/css
CSS Play	www.cssplay.co.uk

Having a reference guide at your fingertips is also extremely useful. You can find a number of more-detailed CSS references by searching for titles at www.wiley.com.

Finally, for a nice listing of CSS resources and access to a CSS blog where you can share your ideas and ask questions of other CSS designers, visit www.mezzoblue.com/zengarden/resources.

Chapter 5: Creating Web Layouts

In This Chapter

- ✓ Discovering the benefits of standards-compliant, accessible layouts
- ✓ Understanding the difference between layouts using tables and layers
- ✓ Finding out how to create a simple layers-only layout
- ✓ Creating tables-based layouts for HTML e-mails
- ✓ Finding and using free online layers-based layouts

In this chapter, you start by finding out about the benefits of creating standards-compliant, accessible layouts and get a primer on how to successfully work with layers. You discover how to create a simple CSS-styled, layers-based layout on your own, and how to use the old-school HTML tables-based layout in case any of your clients ask you to create an HTML e-mail or HTML newsletter for them. At the end of the chapter, you find a helpful list of resources for finding excellent, free CSS layers-based layouts on the Web.

Creating Standards-Compliant, Accessible Layouts

The World Wide Web Consortium (W3.org) is an international vendor-neutral organization that defines standards on the Web to improve Web accessibility and hardware/software interoperability. By following the recommendations of the W3C, designers, programmers, and Web developers can design and build interoperable Web sites that look great, are coded semantically, and function efficiently. Thus, to build a standards-compliant Web site means to use the recommended standards and guidelines put forth by the W3C.



What, then, is *semantic HTML*? Simply put, it is the use of HTML tags and coding that match either what the content is, like using `<p>` tags for paragraph text, or what the content is for, like using the `<label>` tag for form controls such as text fields and radio buttons, which don't have implicit labels. In other words, by following a few simple rules of the HTML road, you can quickly be on your way to developing sites that make everyone happy.

By the way, that “everyone” includes making your pages accessible to the widest-possible human and nonhuman audience. Think of *accessibility* as a fancy way of saying that you can make code enhancements to your Web site that improve how visitors with disabilities and search engine robots/spiders access the information on your site's pages. Common coding enhancements include adding footer links, a site map page, alternate text for images, page titles, meta tags, object labels, titles and targets for links, link tags to the home and site map pages in the header, access and tab index keys, and form input labels.

In addition to semantic coding, standards compliance also means exclusively (or nearly exclusively!) using layers-based layouts, which is another great way of making your pages accessible to the widest possible audience. Remember, you have no control over what computer and browsing tools your target audience will be using to access your Web site. As illustrated in Figure 5-1, those tools will likely include all the different browsers and browser versions on both Mac and PC, handheld devices like cell phones and BlackBerrys, speech synthesizers, video game consoles, and Web spiders or robots, to name a few.

Sadly, though standards compliance has been a hot issue in the Web design world for the last five-plus years, most sites today barely comply with Web standards, especially the older sites that were built using tables-based layouts and sites that are built by the less informed and nondesigners.



As a designer today, you can help make the future of the WWW a better place by following the HTML standards recommended by the W3C when building any new site or redesigning an existing site. Working with layers fits right into the overall mission of standards by allowing designers to precisely position and style content using CSS instead of HTML, which means that code is lighter, pages load faster, pages are easier to maintain, and more visitors (whether they're humans using a browser or other device or nonhumans such as search engine robots) can access that content.

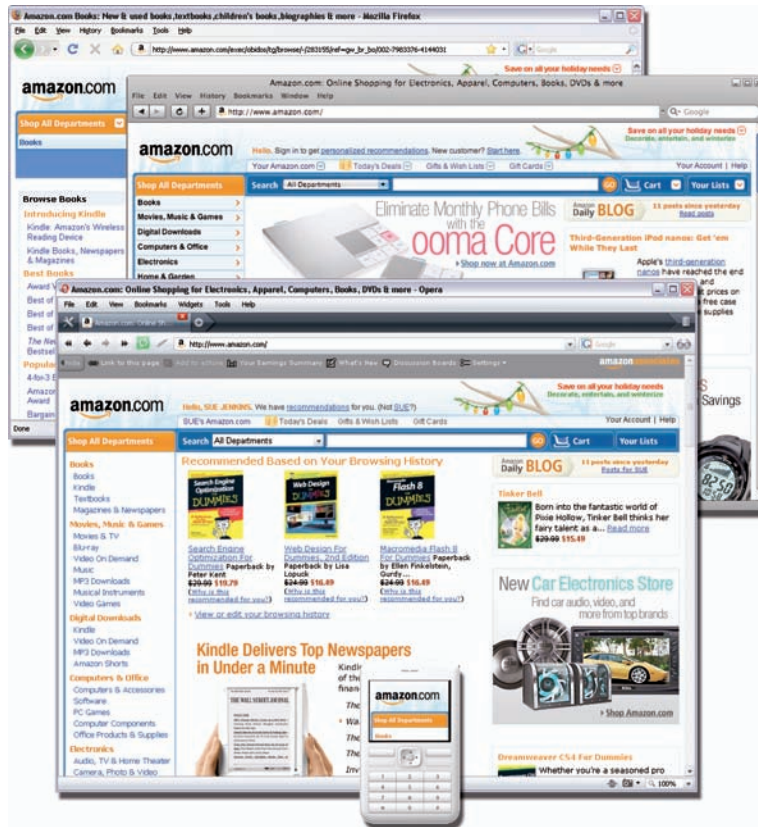


Figure 5-1: Web visitors will likely use a variety of tools to access your Web site.

Working with Layers

If you are brand new to the world of Web design, finding out how to build layers-based layouts from the start will be easy enough to do. However, if you have been doing a bit of Web design on your own and have been working with tables-based layouts but are now ready to discover how to work with layers, you'll probably need to unlearn a few habits you might have developed from working with tables.

Discovering the benefits of layers-based layouts

To be sure, some Web design purists will tell you unequivocally to create only layers-based layouts, while the more “old-school” designers might say that working with a hybrid table/layers layout is also fine. Because you want to do the right thing, your goal should be to try to use layers-based layouts whenever possible. Nonetheless, rules are made to sometimes be broken, so if parts of your design require tables, feel free to use them and don’t feel guilty about it. A hybrid tables/layers layout is far more user friendly than a tables-only layout.

Before launching into how wonderful layers are, rest assured that using tables is not a bad thing and that if you choose to continue using tables for laying out your pages, no Layers Police will come in the dead of night and take you to Table User’s Prison. Layers are simply the more politically correct technique to use because they make pages that are more accessible. In addition, layers are more flexible than tables because when combined with CSS, they can be positioned relatively or absolutely anywhere on a Web page. Tables, by contrast, can be aligned only to the left, center, or right of a page.

In addition to making pages more accessible, you find several wonderful benefits of creating layouts with layers:

- ✓ The code is cleaner because it uses less HTML markup.
- ✓ Layers free you from having to design within the traditional rows-and-columns framework of tables.
- ✓ You can stack layers directly on top of other layers using the `z-index` style attribute. The larger the `z-index` number, the closer the object appears to the viewer in the browser window.
- ✓ You can control layer visibility in a browser window with JavaScript.
- ✓ You can position layers anywhere on a page, including overlapping with other layers.
- ✓ Layers can be nested inside other layers.
- ✓ Layers can be styled and positioned on the page with CSS using `#layerID` as the selector name.
- ✓ Layers can contain any content that can be placed elsewhere on a page, just like tables, and that content can be styled with CSS.
- ✓ By modifying the CSS alone, you can completely change the look and position of the layer’s contents.

Understanding what layers are

Think of layers as a cousin to the single-celled table or as a boxlike container that can be placed anywhere on a Web page, including alongside, above, below, and nested inside other layers.

The code for a layer that can be easily styled with a CSS `id` attribute is very simple:

```
<div id="LayerName">Here is a sentence inside a layer.</div>
```

When a layer tag (the `<div>` tag) includes an `id` attribute, you can create an advanced selector CSS style for it to style and position the layer and its contents. Here's an example of the CSS for the `id` of a `<div>` tag:

```
<style type="text/css">
<!--
#LayerName {
    position: absolute;
    left: 200px;
    top: 100px;
    width: 500px;
    height: 400px;
    z-index: 1;
    font-family: Verdana, Arial, Helvetica, sans-serif;
    font-size: 16px;
    color: #FFF;
    background-color: #99cc66;
    margin: 0px;
    padding: 10px;
    border: 10px solid #000000;
    font-weight: bold;
}
-->
</style>
```

Figure 5-2 illustrates the difference between an unstyled layer and a layer in Dreamweaver using the `#LayerName` CSS style.

Each layer on your page can have its own CSS style, and that style can include as many different style and positioning attributes as you like. You can also create custom CSS styles to control the style and positioning of any part of the contents of a layer, including text, graphics, media, and other file types. While at first working with layers and CSS may not feel as intuitive as you'd like it to — especially if you've already worked with tables — with a little practice you'll soon come to appreciate and respect the vast customization capabilities of working with layers.

Furthermore, as you soon see in the next section, with the knowledge of a few simple techniques, you can soon be on your way to creating your own simple layers-based layouts.



Figure 5-2: Add a CSS style to your layer to control the layer's look and position.

Creating a Layers-Only Layout

All HTML tags are like containers that have specialized functions. Paragraph tags, for instance, surround paragraph text, and image tags display images. Regarding layers, think of them as free-spirited, single-celled tables. Layers can hold any kind of content, including text, graphics, Flash movies, and other media, and both the layer and its contents can be styled and positioned within a Web page. In fact, using CSS for positioning, layers can be placed next to other layers as well as nested inside, placed above, positioned below, and set on top of other layers. Furthermore, with CSS, you can even style and position a series of layers to look similar to tables without all the extraneous code that tables require.

In the sections that follow, you find out how to add layers to your pages, and how to build and style a CSS layers-based layout.

Adding a layer to a page

Adding a layer to a page is quite simple, because it requires only the `<div>` tag with an `id` attribute for attaching a CSS style. You can then place any content inside it, such as a graphic:

```
<div id="header"></div>
```

When the `<div>` tag is assigned an `id` attribute, you can create a CSS style for the layer, such as in the following example, and place that style either in an internal style sheet in the `<head>` area of the page or in a linked external CSS file:

```
<style type="text/css">
<!--
#header {
  position: absolute;
  left: 0px;
  top: 0px;
  width: 760px;
  height: 200px;
  z-index: 1;
  font-family: Georgia, Times New Roman, Times, serif;
  font-size: 12px;
  color: #43BFC7;
  background-color: #2B3856;
  margin: 0px;
  padding: 10px;
  border: 10px solid #000000;
}
-->
</style>
```

One of the most wonderful things about layers is that when combined with CSS styling, you can create unlimited variations with essentially the same HTML markup. This means that you can change the entire look of a Web site by modifying the CSS and changing out some of the graphics. This concept is expertly demonstrated by the various CSS layout designs submitted to Cszengarden.com.

Building a CSS layers-based layout

To illustrate how easily a page layout can be completely transformed with a few minor changes to the CSS, the following steps show you how to create a simple two-column page layout with layers. Then, in the section that follows, you find out how to apply CSS to control the look of the page.

The page you're about to build will have a three-part layout that consists of a header, a two-column main content area, and a footer. Within the two-column main content area, both the left and right columns will be fixed in width, and the entire layout will be centered within the browser window.

Furthermore, the left column will be divided into two sections, the top of which will be used for navigation and the bottom half used for additional sidebar content.

For this layout to function properly, you must add a series of layers to the page in the proper order. The `id` attributes of each of the layers are named semantically to match the function they'll perform, such as `navigation` and `footer`.

To create this CSS layers-based layout, follow these steps:

1. Set up a folder on your computer to save the files associated with this project.

For example, you could create a folder on your desktop called `Web Layout` and then make sure that you save the HTML, images, and other files to this folder.

2. Using your favorite HTML editor, open a new blank HTML document, select either the HTML 4.01 Transitional or XHTML 1.0 Transitional DTD (this is a line of code that tells the browser how to interpret the HTML), and save the file with the name `mypage.html` to the Web Layout folder you just created.

3. In your code, between the opening and closing `<body>` tags, insert six layers using the `<div>` tag and give each one its own `id` attribute:

```
<div id="container">
  <div id="header">
  </div>
  <div id="content">
  </div>
  <div id="navigation">
  </div>
  <div id="sidebar">
  </div>
  <div id="footer">
  </div>
</div>
```



To help you easily identify each layer and further understand how the layers are organized on the page, add some placeholder content to identify each part, as in the following code:

```
<div id="container">
  <div id="header"><p>Insert Header</p>
  </div>
  <div id="content"><p>Insert Main Content</p>
  </div>
  <div id="navigation"><p>Insert Navigation</p>
  </div>
```

```

        <div id="sidebar"><p>Insert Sidebar Content</p>
        </div>
        <div id="footer"><p>Insert Footer</p>
        </div>
    </div>

```

You can go back into each section later and replace the placeholder text with real content. When inserting text, be sure to use semantic HTML by wrapping the correct tags around the content, such as `<p>` tags for paragraph text and `<h1>` to `<h6>` tags for headings.

4. Inside the layer with the `id="header"`, insert a logo graphic or type in the name of the site.

If you'll be using an image for the logo, as illustrated in Figure 5-3, your HTML markup should look something like this:

```

    <div id="header">
    </div>

```



Figure 5-3: Logo graphics are often used in place of HTML to brand a Web site.

Otherwise, if you'll be using only text for the header, your markup should look similar to the following:

```

    <div id="header"><h1>Springfield</h1>
    </div>

```

5. Inside the layer with the `id="content"`, type in a heading and a few paragraphs of text. Wrap the heading in `<h1>` tags and the paragraph text in `<p>` tags:

```

    <div id="content"><h1>Heading For Text Below</h1>
    <p>Sample paragraph text. Sample paragraph text sample
        paragraph text.</p>
    <p>Sample paragraph text. Sample paragraph text sample
        paragraph text.</p>
    </div>

```

6. Inside the layer with the `id="navigation"`, drop in another `<div>` tag with the `id="navlinks"`, and inside it, create an unordered list with five null hyperlinked list items:

```
<div id="navigation">
  <div id="navlinks">
    <ul>
      <li><a href="#">Link 1</a></li>
      <li><a href="#">Link 2</a></li>
      <li><a href="#">Link 3</a></li>
      <li><a href="#">Link 4</a></li>
      <li><a href="#">Link 5</a></li>
    </ul>
  </div>
</div>
```

By surrounding the navigation list with its own named layer, you can make each item in the list look like navigation buttons by creating a style called #navlinks. For more information about creating navigation systems and styling list items with CSS, refer to Book III, Chapter 6.

7. Create another layer that will sit directly beneath the navigation layer and display additional sidebar content:

```
<div id="sidebar">
<p>Sidebar: Use this area for adding page specific
  sidebar content such as Site Search box, Newsletter
  Signup form, Advertisements, Daily Specials, and
  Latest News Items.</p>
</div>
```

8. In the footer layer, repeat the main navigation links and add any additional footer copy, such as a link to a site map page and a copyright notice, as demonstrated here:

```
<div id="footer"><a href="#">Home</a> | <a
href="#">Link 1</a> | <a href="#">Link 2</a> | <a
href="#">Link 3</a> | <a href="#">Link 4</a> | <a
href="#">Link 5</a> | <a href="#">Site Map</a> |
Copyright &copy; 2009 Springfield</div>
```



The HTML code `©` is the special HTML entity used for the copyright symbol. Use this entity instead of the regular copyright symbol (©) to ensure that the copyright symbol will display more accurately in the widest variety of browsers and can be read properly by screen readers and other assistive devices.

When previewed in a browser, your code should create a page that looks like the example shown in Figure 5-4.

Creating descendant selectors

One of the best ways to find out more about CSS is to take a look at what others have done before you. Beyond the basics of CSS, you can do quite a few unusual and wonderful things with CSS. One of those more advanced techniques is to create what's called a *descendant selector style*, which uses a unique form of CSS syntax to apply a style to the set of tags specified in the style's selector, such as a `border` attribute for an image inside a table cell inside a particular named layer that sits inside another named layer:

```
#container #sidebar td img {
    border: 1px solid #000000;
}
```

Descendant selector styles can be used for all kinds of styling tasks, including creating custom link states for different parts of a Web page. Here's an example of a selector for the hover state of a hypertext link that applies only to links in a layer called `sidebar`:

```
#sidebar a:hover {
    padding: 10px;
    border: 1px solid #6699CC;
    background-color: # 336699;
    color: #fff;
}
```

Here's another example of a descendant selector style that modifies the hover state of any linked item in an unordered list that sits inside a layer called `linklist`:

```
#linklist ul li a:hover {
    color: #990033;
}
```

Descendant selector styles can even help you to control how an image or text sits inside a layer. For instance, rather than apply a style that would add padding to an entire layer, you could selectively apply padding to the contents of that layer using a special descendant selector style that only applies to specified tags nested inside the layer, such as the `<h1>` tag, as in the following code:

```
<div id="sidebar"><h1>Today's Specials</h1>
</div>
```

To apply a style to the contents of that `<h1>` tag, your descendant selector style would look something like this:

```
#sidebar h1 {
    margin: 0;
    padding: .75em;
}
```

When both the HTML and the CSS are combined, the new style is automatically applied to the `<h1>` tag inside that layer, but not to any other `<h1>` tags on the page.

Create descendant selectors for any set of tags and preexisting styles, including links in a footer, images in a navigation bar, and text in a sidebar layer or any other location on your page.



Figure 5-4: With no CSS styling, the HTML on the page has limited formatting.

Styling a CSS layers-based layout

In the steps that follow, you'll create the CSS that controls how the page content will be styled, positioned, and presented in a Web browser.

When styling content with CSS, it is often best to style the page from the outside in by starting with redefining the `<body>` tag, then adding styles for various layers, and finally creating any *descendant selectors* (see the nearby sidebar) to style the content within the layers, including any `<p>` and `<h1>` through `<h6>` tags.



For demonstration purposes, the CSS in the following steps will be placed inside the page rather than on an external style sheet. However, if this layout were to be used for a real Web site, the CSS should be moved into an external CSS file before you create any additional pages for your site, because an external CSS file makes site management much easier.

Follow these steps to apply CSS the layers-based layout you created in the preceding section:

- 1. Begin by adding an internal style area between the `<head>` tags on the page using the style tags with comments:**

```
<style type="text/css">
<!--
-->
</style>
```

2. Between the comment tags (`<!--` and `-->`), add a redefine style for the `<body>` tag to set the page margins and padding to 0, which will make the site layout begin at the upper-left edge of the browser. If desired, include a style declaration for the page background color.

```
body {
    margin: 0px;
    padding: 0px;
    background-color: #ffffff;
}
```

The `<body>` tag defines the attributes for the body of the entire Web page. However, not all container tags nested inside the `<body>` tag will *inherit* the font attributes specified when the `<body>` attributes are redefined in the CSS.



In CSS, to *inherit* is to automatically take on the properties of a parent class. For instance, if the font is specified in a redefined style for the body (parent), all paragraph tags (child) should automatically inherit the same font unless otherwise specified in the CSS. Unfortunately, over the years, not all browsers handle inheritance the same way (see www.ericmeyeroncss.com/bonus/render-mode.html), especially when it comes to tables and lists. Therefore, to ensure that all containers on the page use the same font and other global attributes within the redefined body style declaration, create a secondary advanced selector style that specifies the `<body>` and other subcontainer tags:

```
body {
    margin: 0px;
    padding: 0px;
    background-color: #ffffff;
}
body,th,td {
    font: 12px Verdana, Arial, Helvetica, sans-serif;
    text-align: center;
    color: #000000;
}
```

Alternatively, you could simply create a tag redefine style for your paragraphs and headings, and custom styles for your tables and table cells.

Next, you create styles for each of the `<div>` tags so that the individual layers are styled and positioned accurately on the page.

3. Create an advanced selector style for the header layer by using the `#layerid` syntax and set the position to relative:

```
div#header {
    position: relative;
    height: 100px;
    border-top-width: 0px;
    border-right-width: 0px;
```

```
border-bottom-width: 1px;
border-left-width: 0px;
border-top-style: none;
border-right-style: none;
border-bottom-style: solid;
border-left-style: none;
border-bottom-color: #62D1F0;
}
```

- 4. Create a descendant selector style for any content that sits inside an <h1> style within the header and set the following properties:**

```
div#header h1 {
  height: 70px;
  line-height: 70px;
  margin: 10px;
  padding-left: 10px;
  background-color: #EEE;
  color: #CC0000;
}
```



If you're using a hexadecimal value that has the same number or letter for each pair of the three-pair number, you can use a shorthand syntax where only the first number or letter of the pair needs to be specified, such as #f00 for #ff0000 or #f36 for #ff3366.

- 5. Create a descendant selector style for the container layer to set the text alignment to left, the width to 700 pixels, and the margins to 0:**

```
div#container {
  text-align: left;
  width: 700px;
  margin: 0 auto;
}
```

- 6. To style the content, you need two descendant selector styles: one for the content layer and another for the heading and paragraph text inside the layer:**

```
div#content {
  float: right;
  width: 500px;
}
div#content h1, p {
  line-height: 1.5;
  margin: 10px;
}
```

Creating descendant selectors for paragraph text within a layer helps ensure proper alignment, because some browsers are buggy about the way contents inside a layer get displayed.

- 7. To style the navigation layer, create a style that sets the background color and width of the layer, as well as the position within the container, which in this case is `float: left;`:**

```
div#navigation {
    background-color: #FC6;
    float: left;
    width: 200px;
}
```

The `float` attribute tells the browser to keep that layer always positioned on the left side of the page. The `float` attribute can be set to `left`, `right`, `none`, or `inherit`.

- 8. Create another similar style for the sidebar layer, specifying a background color, width, and position in the container:**

```
div#sidebar {
    background-color: #FC0;
    float: left;
    clear: left;
    width: 200px;
}
```

Adding the `clear: left;` attribute to this style tells the browser that the sidebar layer must both float to the left of the other layers within the same container and appear below them.

- 9. To style the list of navigation links, develop a style for the `navlinks` layer as well as styles for the `` and `` tags, as in this example:**

```
div#navlinks {
    display: block;
    width: 200px;
    clear: right;
    float: right;
    text-align: left;
    margin: 0px;
    padding: 0px;
    background-color: #FC6;
}
div#navlinks ul {
    list-style-type: none;
    width: 200px;
    margin: 0px;
    padding: 0px;
}
div#navlinks li {
    background-color: #9FC749;
    padding: 0px 10px;
    border-bottom-width: 1px;
    border-bottom-style: solid;
}
```



```
border-bottom-color: #62D1F0;
font-family: Geneva, Arial, Helvetica, sans-serif;
color: #62D1F0;
display: block;
margin: 0px;
}
```

If desired, you can also create special styles for each of the four link states.

- 10. Define three styles for the footer area: one for the footer layer, one for any paragraph text within the footer, and a third for any hyperlinks within the footer, which can be different from hyperlink styles found elsewhere on the page:**

```
div#footer {
background-color: #EEE;
color: #C00;
clear: both;
width: 100%;
padding: 10px;
}
div#footer p {
margin: 0;
padding: 1em;
}
div#footer a {
display: inline;
padding: 0;
color: #9FC749;
}
```

An em is one of the units of measure acceptable on the Web that also makes pages more accessible to a wider audience. What makes the em different from pixels and percentages, however, is that each em unit varies, because its size is equal to the point size of the specified font face. For instance, if the font for a page is set to 10px, one em is exactly 10px.

Notice, too, that the `div#footer` style includes an attribute called `clear: both;`. This is what makes the footer layer fall to the bottom of the page, rather than float next to any nearby layers.

- 11. Save the code changes you have made to your Web page and preview your layers-based layout in a browser.**

You can preview the file by double-clicking the HTML file icon or by dragging and dropping the HTML file into an open browser window.

Your layout should look similar to the example shown in Figure 5-5.



Figure 5-5: Test your CSS for display accuracy by previewing your page in a variety of different browsers.

12. To see how the CSS can be easily modified, go back into the code and make a change or two, such as changing the background color of the sidebar and changing the font attributes for the text appearing in the main content area.

The more you play around with CSS, the better you begin to understand how to use this powerful tool.

Building an Old-School HTML Tables-Based Layout for HTML E-Mail and Newsletters

Now that you understand the ins and outs of designing layouts using layers, you can dip your toes into the pool of designing pages using old-school tables. Though you really shouldn't create Web sites using tables-based layouts anymore, you can still use them to good effect for organizing specific content area in your pages as well as for designing a good, old-fashioned HTML e-mail or newsletter.

Understanding the benefits of tables-based layouts

Here are some of the benefits of working with tables in HTML e-mails and newsletters:

- ✓ Tables are easy to use.
- ✓ You can stack tables above and below one another.
- ✓ You can easily nest tables inside other tables.
- ✓ Tables can contain any content that can be placed elsewhere on a page.
- ✓ You can style tables, table cells, and cell contents with CSS.
- ✓ Tables stay on the page exactly where you put them.
- ✓ You can size tables with precise pixel dimensions or with percentages relative to the browser size.
- ✓ Although tables use a lot of code, there is a logic to how you can construct and style them.

As explained in Book III, Chapter 2, a *table* is a gridlike HTML container that can be placed on your page and set to have any number of rows and columns. Each section in the table grid is called a table cell, which is defined by a pair of `<td>` tags. Here is an example of the HTML code you'd find for a typical two-row, two-column table:

```
<table width="300" border="1" cellspacing="3"
  cellpadding="3">
  <tr>
    <td>...</td>
    <td>...</td>
  </tr>
  <tr>
    <td>...</td>
    <td>...</td>
  </tr>
</table>
```

HTML tables are quite flexible in that contiguous cells can be merged or split to create more unusual table structures. In addition, tables and individual table cells can be precisely sized or sized relative to the browser window or another containing HTML tag. Whole tables can even be nested inside other table cells to create meticulous layouts that accommodate even the most complex designs and presentation requirements. To change the appearance of the table and any of the table's contents, you can easily style the table and its parts with CSS by either creating tag redefine styles for the `<table>`, `<tr>`, `<th>`, and/or `<td>` tags, or by creating custom class styles that are then selectively applied to any of those tags and to the contents of any table cell.

Despite all their fancy abilities, tables on their own simply can't do everything a Web designer might want them to do. As a work-around, you might even need to use special code hacks to get the contents of the tables to line

This creates a single-celled table that spans the entire width of an e-mail program's preview window and sets the background color of the e-mail and the area inside which you can nest a two-column table.



Figure 5-6: Stack and nest tables to achieve a layout that matches your design.

- To nest a second table inside the first one, place your cursor between the first table's <td> tags and insert a one row, one-column table with the following attributes:**

```
<table width="600" cellpadding="0" cellspacing="0"
      bgcolor="#FFFFFF" height="75" align="center"
      border="0">
  <tr>
    <td>&nbsp;</td>
  </tr>
</table>
```

- Inside the second table's <td> tags, insert the logo graphic for your client.**

In this case, the logo and any other graphic information, such as a tagline and photograph, can be any size up to 600 pixels wide and 75 pixels high.



The contents inside table cells, such as a logo graphic and/or text, sometimes don't automatically align properly when viewed in some e-mail applications. Therefore, when working with tables for HTML e-mail and newsletters, be sure to set the horizontal alignment (*align*) and vertical alignment (*valign*) attributes for every table cell in the code, such as

```
<td align="left" valign="top">Contents of cell</td>
```

Table cell horizontal alignment options include *left*, *center*, and *right*, and vertical alignment options include *top*, *middle*, *bottom*, and *baseline*.

- Position your cursor and modify the code so that you can insert another table in the right spot. Place your cursor in the code of your page directly following the nested table's closing </table> tag and before the main table's closing </td> tag.**

Having your cursor in this position allows the next table you insert to appear stacked directly underneath the previous one.

- Insert a two-column, single-row table on the page with the following specifications:**

```
<table width="600" bgcolor="#FFFFFF" cellpadding="20"
      cellspacing="0" align="center" border="0">
  <tr>
    <td>Main copy</td>
    <td>Right column</td>
  </tr>
</table>
```

The left side of this two-column table will display the main copy of the e-mail, while the right column will be used to highlight other items of interest, showcase interesting facts and new items, and provide links to the company Web site or other online resources.

7. Directly following that table's closing <td> tag, insert another single-row, single-column table with the following attributes:

```
<table width="600" cellpadding="0" cellspacing="0"
  bgcolor="#FFFFFF" height="75" align="center"
  border="0">
  <tr>
  <td>Footer</td>
  </tr>
</table>
```

Most e-mails have a footer of some kind that includes text, contact information, and a method for readers to unsubscribe to the mailing list.

8. Add styling to the content of the various areas of the layout.

With regular HTML Web pages, you'd use internal or external CSS exclusively to style and position the content. However, with an HTML e-mail or newsletter, you need to resort to using simple inline CSS to style the markup within the code. For example, to add font face, font size, font color, font weight, and line height formatting to a heading, you could add inline CSS markup as follows:

```
<p><span style="font-size: 20px; font-weight: bold;
  color: #82AB1E; font-family: arial; line-height: 110%;">Heading
1</span></p>
```



As an alternative to styling your HTML e-mail with inline CSS, you might be able to get away with using internal CSS because many of the newer e-mail clients can read internal CSS. With internal CSS, the HTML code requires less markup. To use an internal CSS, begin by adding the internal CSS style tag to the head area of your code, including comment tags, and apply a body redefine style (to zero out your margins and padding) as your first style declaration, as in the following code:

```
<head>
<style type="text/css">
<!--
  body { margin: 0px; padding: 0px; }
-->
</style>
</head>
```

After the internal CSS is established, you can create as many tag redefine and custom CSS styles as needed and then apply those styles to your content where applicable. For example, to add a special style for

the title and subtitle of your main article as well as a style that would automatically format any text that appears inside any table cell, you might create styles for them as follows:

```
.title { font-size:20px; font-weight:bold; color:#82AB1E; font-
family:arial; line-height:110%; }
.subTitle { font-size:10px; font-weight:normal; color:#333333; font-
style:italic; font-family:arial; }
td { font-size:12px; color:#000000; line-height:150%; font-
family:verdana, arial, sans-serif; }
```

Figure 5-7 shows an example of the HTML e-mail both before and after being styled with inline CSS.

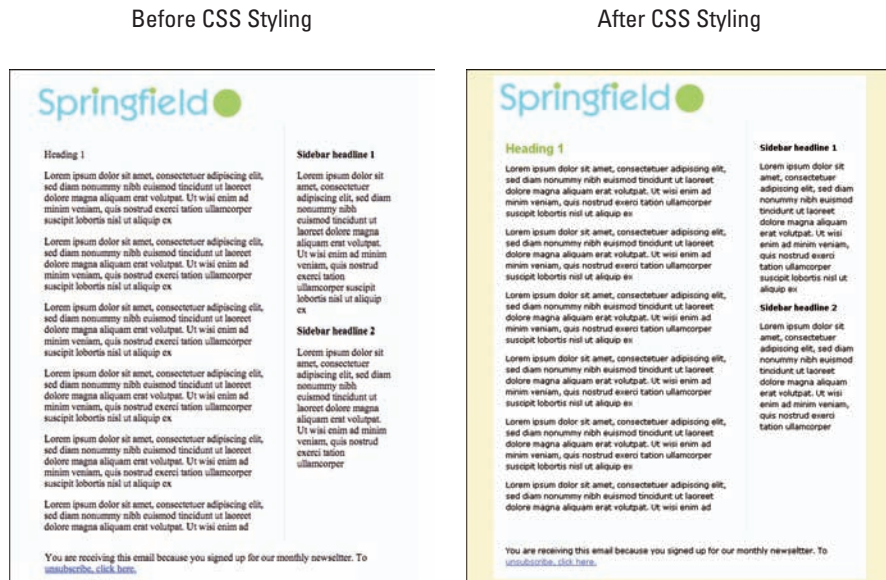


Figure 5-7: Style your HTML e-mails using internal or inline CSS.

When you have finished adding content and styling text with inline or internal CSS, save the file and send a test e-mail to as many e-mail clients as you can and make modifications to the code as needed. Because some e-mail clients can only interpret text and not HTML, and many others interpret the HTML in different ways, stay open-minded. If, after multiple attempts to make the e-mail look uniform, you can't find a solution, be willing to accept a variety of results because some display issues may be irresolvable. Figure 5-8 shows an example of how my test e-mail looks in Gmail (left) and Outlook (right). While the overall structure is the same, some differences exist in the way the layout renders in each e-mail program.



Figure 5-8: By previewing and comparing your HTML e-mail in a variety of different e-mail applications, you can attempt to fix any display issues with the code before you send out the e-mail.



To find out more about working with tables, see Book III, Chapter 2. To discover more about creating HTML e-mails, visit the MailChimp Resource Center at www.mailchimp.com/resources and download a free copy of MailChimp’s Email Marketing Beginners Guide.

Finding Online Resources for Layers-Based Layouts

One of the best ways to find out about CSS and really understand the power of layers-based layouts is to begin the design process using a predefined CSS layers-based design template. Several resources are available online where you can obtain free, clean, standards-compliant templates, and then either use those layouts as is or as a starting point to work from. You can then modify the templates to match your site design by tweaking and adding your own CSS styles.

Dreamweaver users can use any of the 32 beautifully commented CSS layers-based page layouts that come built into Dreamweaver. These page-design guides are accessible from the New Document dialog box in the Blank Page category under HTML in the Page Type column, as shown in Figure 5-9, and in the Blank Template category under HTML Template in the Template Type column.

The 2 column hybrid, right sidebar, header and footer layout, shown in Figure 5-10 for example, comes with its own CSS file (which can be placed internal or external to the file) and has a header for the site's branding, a main area with a liquid two-column design for the site's main and sidebar content, and a footer, all of which can be customized through Dreamweaver's CSS Styles panel.

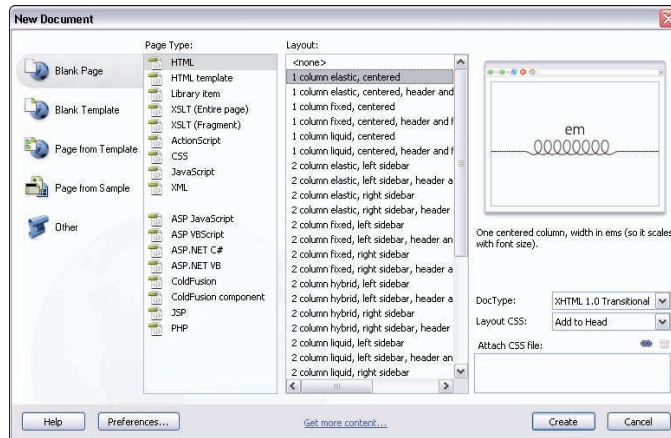


Figure 5-9: Dreamweaver includes 32 prewritten CSS layers-based layouts that you can use as a starting point to build your own Web site.

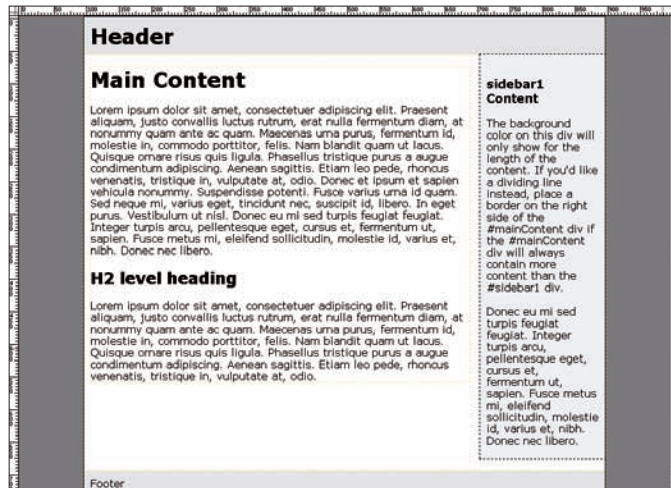


Figure 5-10: Dreamweaver's 2 column hybrid, right sidebar, header and footer layout includes areas for a general page layout.

In addition to working with Dreamweaver's templates, another great source for free CSS layout pages is the MaxDesign site (www.maxdesign.com.au/presentation/page_layouts), which features 20-odd layouts.

The more time you take to explore all the wonderful things you can do with CSS, the more all of it will start to make sense to you. Visit as many of the sites listed (in no particular order) in Table 5-1 as you can, and be sure to also do a search for "CSS layouts" in your favorite search engine to find additional CSS page-layout resources on the Web.

Site Address	Freebies, Tutorials, Tips, and More
http://websitesitips.com/css/templates/#csstemplates	A great list of links to sites that offer free CSS layouts
http://webhost.bridgew.edu/etribou/layouts	Free CSS layouts with special features like style switchers and a very nice free CSS menu
www.oswd.org	Free blog templates
www.freecsstemplates.org	Free CSS/XHTML layouts and templates for Web sites and blogs
www.thenoodleincident.com/tutorials/box_lesson/boxes.html	16 free CSS layouts
http://csscreator.com/tools/layout	An online CSS page-layout generator
www.maxdesign.com.au	Over 20 free CSS page layouts with additional tutorials
www.bluerobot.com/web/layouts	Free two- and three-column CSS page-layout templates
www.ssi-developer.net/main/templates/index.shtml	Five free CSS page-layout templates
www.htmldog.com/guides/cssadvanced/layout	15 free CSS page layouts
www.glish.com/css	Free layouts and doodads for Web sites, blogs, and more
www.adobe.com/devnet/dreamweaver/articles/css_page_layout_basics.html	Dreamweaver Layout Tutorial

Chapter 6: Constructing Navigation Systems

In This Chapter

- ✓ Considering the site's organization and target audience
- ✓ Finding out about navigation systems
- ✓ Creating text navigation menus
- ✓ Developing rollover button graphic navigation menus
- ✓ Building multitier Spry menus in Dreamweaver
- ✓ Creating CSS list navigation menus

Architecturally, the single most important element of a Web site is its navigation system because it is the one unifying feature of a site that enables visitors to quickly survey a site's structure and access the desired information. Think about your own habits when you go to a new site. Most visitors, unconsciously or not, expect to be able to find what they are seeking within one to three clicks. If they can't find what they are looking for — or worse yet, if the links they do click mislead them in some way — they will promptly leave that site and go search somewhere else, unless of course they know that it's the only online resource that has what they're looking for. For these reasons alone, a site's navigation system must be easy to find and easy to use. Better yet, make it visually appealing too, and you have yourself a great navigation system.

In this chapter, you find out how to choose and then create the right navigation system for a Web site. You start by discovering some basics about matching a site to the right navigation system based on its target audience. Then you follow along with steps that show you how to build each of the four most popular navigation menu systems: text, rollover button graphic, multitier Spry, and CSS list.



Assessing the Navigational Needs of Your Site

To select the right navigation system for a site, you really need to go back to the site architecture — the site map — which you find out how to create in Book I, Chapter 3. With a strong understanding of the site's structure and organization, you can then choose the right type of navigation for the site. For example, if the site is rather small and will only have five main navigation links, you can be a bit more creative with your selection of a navigation system. By contrast, if you're building a site that has ten main navigation links, each of which includes subnavigation that may also include another level of subnavigation links — in other words, a big site that requires a multitiered navigation system — you need to understand quite a bit about HTML, CSS, and JavaScript *before* you convert the navigation in your mock-up into a working navigation system on your Web site.

As you can see, having a good idea of the organization and structure of all the pages on any particular site provides you with the best understanding of the type of navigation that is needed. By reviewing the architecture, you can easily identify the labels for each page and the order of all the main navigation links and subpages of the site. Each of the site's pages should fit into one of three categories for the navigation:

- ✓ **Main navigation links:** These pages represent the main pages on the site, such as About and Contact, that should be easily accessible from any other page on the site through the navigation system.
- ✓ **Subnavigation links:** Subpages are pages that fall logically into a category beneath one of the main pages, such as a Directions page in a submenu under a Contact Us page that provides visitors with information on how to get to a company's facility. These pages will be accessible through some kind of submenu off the main navigation. A subnavigation menu can be a pop-up menu from the main navigation link, a second row of links that appears below the first row when activated by the main navigation link for that group, or even a sidebar area that appears somewhere on the page when activated by the main navigation link of that group.
- ✓ **Non-navigational links:** Many Web sites have pages that do not need to be included within the main navigation or subnavigation areas, but must still be accessible to visitors through hyperlinks located in various spots throughout the site, such as in the footer of all the pages on a site or in the body text area of a particular page. Links like these might include Privacy Policy, Site Credits, or Site Map pages.

In addition to the site organization, three other factors may further influence which type of navigation works best for a Web site:

- ✓ **General usability:** Consider the general usability of the navigation system. The menu must be intuitively easy to read and use, include concisely written and meaningful labels, and be in the same location on every page of the site so that visitors don't need to go searching on different areas of the page to find it.
- ✓ **Target audience:** The target audience can often help determine the relative complexity or simplicity of the navigation menu type. For instance, a target audience comprised mostly of seniors and visitors with disabilities might require larger fonts for menu buttons than an audience made up primarily of high school and college students. Likewise, the navigation for an indie electro-band Web site could be far more obscure and unusual than the navigation system for a site that sells MP3 downloads.
- ✓ **Expandability:** Find out in advance whether the site's navigation is anticipated to grow with additional pages and/or subcategories. If the site will be growing at some time in the future, make sure that you choose a navigation system that will be easy to update and expand, because some navigation systems take less time to reconfigure than others. For instance, text and CSS-styled list menus are far more readily expandable than menus that rely on precisely sized rollover graphics powered with JavaScript.

Discovering the Basic Principles of Navigation Systems

All Web sites, whether they contain two pages or 200, must supply a method that allows visitors to move freely among all the pages on the site. The simplest form of page navigation is the hypertext link, which only requires that the destination filename be specified in the link code. Although extremely functional and accessible to the widest possible audience on nearly any Web-enabled device, a text-only navigation menu can tend to look kind of, well, boring. To spiff things up a bit, consider giving your site a more complex-looking menu that uses some combination of text, graphics, CSS, and possibly even JavaScript or some other programming language to handle its dynamic functionality, such as giving the menu cool-looking rollover buttons or drop-down menus.

The navigation on a site needs to provide a simple route to all the most important pages on the site. How that route looks and functions is the key role of the navigation system. Discuss with your client about where on the page the navigation menu area will be placed relative to the other content. If the navigation menu also requires subnavigation, be sure you discuss with your client how and where the subnavigation items should be displayed.

Wide versus deep menus

To further guide you in the selection and implementation of your navigation system, take to heart the following important principle in Web design navigation. *The visitor should be able to find what he or she is looking for in the fewest number of clicks.* To help realize this standard, you may choose to design a navigation menu that is either wide or deep, depending on the number of main navigation links in the menu:

- ✓ **Wide:** A *wide* menu refers to a navigation system that lists links to all the main pages on a site in a single horizontal row, as shown in Figure 6-1. If the site is rather small, having four to seven main navigation links across the page might be a suitable solution. On the other hand, for a site that has 14 main pages, there is probably no visually appealing way to list all 14 links in a single horizontal row. You could try breaking the links into two rows of seven, but that might confuse site visitors unless careful attention was paid to the design and layout. Alternatively, you could present the wide menu to visitors as a single vertical list of navigation buttons, as is done along the left side of the AOL.com home page. If, however, the large number of main pages is simply the result of the client's failure to organize content, help your client rethink the architecture in terms of a deep navigation menu system instead.



Figure 6-1: Wide navigation menus are best for sites with a small number of main pages and no subpages.

- ✓ **Deep:** In a *deep* menu, all the Web pages are grouped into categories of similar interest to reduce the total number of main navigation links, which can be displayed either horizontally or vertically on the pages of the site. Each category has a main page (such as About Us) and one or more subpages (such as Company History, Board of Directors, and Our Sponsors) that can be accessed through some kind of customized sub-navigation menu system, like the one shown in Figure 6-2.

Depending on the orientation of the main navigation menu, the subnavigation menus can either pop up, drop down, or fly out from the main menu. For instance, the subnavigation might be displayed as a second row of links that appears below the first row of links when activated by the main navigation link for that group, or perhaps it might be displayed as a sidebar area that only appears somewhere on the page when activated by the main navigation link of that category.



Figure 6-2: Deep navigation menus are suitable for sites with multiple pages and subpages.

Single-tier menus

Single-tier, or single-level, menus are always wide menus, regardless of whether the navigation links are displayed horizontally or vertically on the page. Because single-tier menus don't have many links, they provide you with the most freedom when it comes to choosing a method to construct them with. For instance, you might want to create graphics for the entire menu so that you can use specific fonts and graphic effects. You may even want to use JavaScript or some other programming language to dynamically create rollover buttons or insert some other form of unique interactive menu features on the page. Whether you're using HTML text, graphics, CSS, HTML, JavaScript, programming, or any combination of these tools to create your single-tier menu, the sky really is the limit.

Multitier menus

Multitier menus use links with submenus to create tiers of navigation. Each tier may include additional submenus, and every link, no matter where it lies within the tiered navigation scheme, provides visitors access to any page on the menu with a single click.

With multitier menus, you must have a firm understanding of how to construct a menu before you design it and get approval on it from the client. New designers often create a menu and sell their client on its functionality before they know whether they have the skills to build it! To avoid putting yourself in a situation like that, familiarize yourself with all the different kinds of menus that can be easily replicated.

Choosing the Right Menu for Your Site

To build a deep or wide single-tier or multitier navigation system, you can choose from several kinds of different coding solutions, depending on the site's specific needs:

- ✓ **Text-only:** These horizontal or vertical navigation menus are made up of plain hyperlinks that are often separated by some kind of character (like a dash) or small graphic (like a vertical divider).
- ✓ **CSS list:** The most user-friendly and accessible form of navigation system is built with HTML text links and CSS styling. Using CSS list formatting, CSS list menus can be either single-tier or multitiered and can be made to sit horizontally or vertically across the page. With a more enhanced understanding of CSS, you can even understand how to build CSS list menus using your own custom graphics and CSS styles to create rollover type menus.
- ✓ **Rollover buttons:** Rollover menus can be created by combining HTML, CSS, and graphics. If you prefer to use graphics instead of HTML text links, create your own custom button graphics and combine them with JavaScript to create interactive buttons. When a visitor moves her cursor over the button graphic, the JavaScript can temporarily display an “over state” graphic there instead. When the visitor moves her mouse off the graphic, the “over state” graphic will disappear and the original “normal state” graphic will be restored. Rollover buttons can also be created by building a CSS list menu with HTML text and styling the individual buttons with CSS.
- ✓ **Forms (jump menu):** Form-style jump menus let visitors select a destination page from a predetermined set of link options. Upon their selection of an option from a jump menu, the form can either force the browser to automatically redirect the visitor to the selected page, or the visitor may be required to click an attending Go or Submit button before the redirect occurs. Though jump menus can be styled somewhat with CSS, not every browser displays the CSS styling in the same way.
- ✓ **JavaScript multitier:** Like the text-only navigation menu, a JavaScript multitier menu uses JavaScript to create and display the different levels of subnavigation. Though typically built using only HTML text hyperlinks, these menus can also use rollover menu buttons in both the main and submenu areas. More often than not, however, the main menu is created with rollover buttons, while the subnavigation uses HTML text hyperlinks.
- ✓ **Flash:** Flash menus can be created using Adobe Flash, which uses its own form of interactive scripting language called *actionsript*. Flash menus may contain text, graphics, rollover effects, sound, and other types of animation and special effects. After the menu is built in Flash, it can be exported and saved as an `.swf` file. The `.swf` file can then be inserted into a Web page as a multimedia file that automatically plays

when the page first loads inside a browser window. For examples of what you can do with a Flash menu, like the sample shown in Figure 6-3, visit www.flashmenus.net.

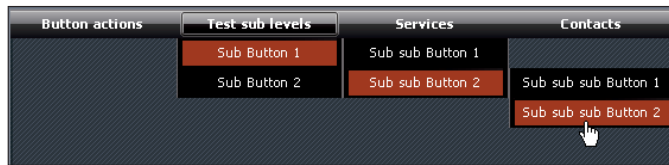


Figure 6-3: Flash menus can include text, graphics, rollovers, sound, and more.

- ✓ **Tree-style:** Similar in layout to the Explore feature on PCs running the Windows OS, tree-style menu list directories (main menus) and subdirectories (submenus) using tiny icons for folders and files along with little plus (+) or minus (–) symbols to indicate whether any of the items in the menu are expanded or collapsed. Because these menus appear more application oriented than they do as navigation systems for a Web site, tree-style menus are used rather infrequently on typically technically oriented sites.
- ✓ **DHTML (Dynamic HyperText Markup Language):** These menus use hypertext links or graphics embedded inside layers using a set of `<div>` tags, which are then hidden or revealed in the browser by the visitor's interactivity with the menu through the use of JavaScript.
- ✓ **Java applets:** Java applets are special menu applications that have been created, probably by someone else, in the Java programming language (not to be confused with JavaScript, a markup language). The benefit of a Java applet is that it is OS independent. In other words, it does not matter whether the visitor is using a Mac, PC, or Linux operating system, as long as that computer has the free Sun Microsystems Java Applet reader software installed. Though not the most user-friendly type of menu system from an accessibility standpoint, new features within the programming language are currently being developed to improve that.
- ✓ **Web application:** Like JavaScript menus, these menus make use of a server-side programming language (such as PHP, Perl, ASP.NET, C#, VB.NET, Ruby, and so on) to create interactive menu systems that use a combination of HTML text, CSS styling, and optimized Web graphics.

With so many options to choose from, it may seem like a difficult task to select the right navigation system (or blend of them) for a Web site. But truly, the answer should be quite clear based on several factors, including the number of navigation and subnavigation links on the site map, the client's preferences, and the target audience profile. For instance, your

client may come to you with a particular preference, such as “I want the navigation menu to sit across the top of the page, below the logo, and have the subnavigation display in a row directly below it,” that simply must be catered to.

After creating a few of the different types yourself, you’re likely to develop a personal preference for one or two of the menu styles over the others, and you can then propose these menu styles to your Web clients before you even begin your mock-ups. Alternatively, you may be the kind of person who loves to figure things out as you go, and you are willing to discover new skills and do whatever it takes to make the best navigation system for the site. No matter how it functions or what it looks like, as long as the menu is easy to find, easy to read, easy to use, and always in the same spot on every page, the one you decide upon can be the right choice for your site.

In the remaining sections of this chapter, you find out how to create four of the most popular types of navigation systems.

Creating Text Navigation Menus

The most accessible, user-friendly, search engine-friendly, easy-to-build navigation system is the text navigation menu. Text-only menus generally consist of a series of HTML hypertext links to the main pages of the site.

In the following sections, you find out more about the two options for presenting your text-only menus as well as how to create a text-based navigation bar.

Exploring your layout options

Text navigation menus can be presented on a Web page in one of two ways:

- ✓ **Horizontal/vertical list:** Similar to an HTML footer menu, the text navigation menu can include all the HTML hypertext links in a single row or column with each link separated by a character of some kind, such as a bullet (•), dash (-), or vertical line (|) when listed horizontally, or by a line break (
) when listed vertically. See Figure 6-4 for an example.

Horizontal:

[Home](#) | [About](#) | [Services](#) | [Clients](#) | [Contact](#)

Vertical:

> [Home](#)
> [About](#)
> [Services](#)
> [Clients](#)
> [Contact](#)

Figure 6-4: Text navigation menus can be displayed vertically or horizontally either with or without special character separators.

- ✓ **Table cells:** The second way to display a text menu is to isolate and separate each of the links by placing them inside individual table cells. This technique allows a little more styling pizzazz with CSS and tends to render more uniformly in browsers, even though it does make use of HTML table tags, which some purist CSS designers frown upon. Figure 6-5 shows an example of how navigation links can be organized inside a table.

Home	About	Services	Clients	Contact
------	-------	----------	---------	---------

Figure 6-5: Place links inside table cells to help visitors more easily identify each link and to create more of a button appearance.

By applying CSS styling to your simple hypertext links (which some folks also call *hyperlinks*), you can give your links more of a button appearance by changing the normal, visited, hover, and active states of the links. In addition, if you add a little JavaScript to the mix, you can do more interesting mouseover states for simple text links inside a table cell.

Creating a rollover text-based navigation bar

In the following steps, you find out how to create a text-based navigation bar in a table format, and then add CSS and JavaScript to style the buttons and create a rollover effect. Because this is a demo and you'll only be creating two styles for the rollover effect, the CSS can be internal to the page. Otherwise, if you were creating this navigation for a real Web site, you'd be better off putting all the CSS in an external CSS file.

Follow these steps to create a text navigation menu:

- 1. In your favorite HTML editor, open a new blank HTML file and save it on your computer with the filename `textnav.html`.**

Where you save the file is entirely up to you. For instance, you might create a folder on your desktop called `Navigation Systems` and then save the `textnav.html` file inside it.

- 2. Insert a 500-pixel-wide table into your page. The table should have 1 row, 3 pixels of cell padding, and 0 pixels of cell spacing. The number of columns in the table should match the number of buttons you want to create.**

For the example, create a table with five columns for the five buttons you want to create.

If you happen to insert your table before you set the number of rows, columns, cell padding, and cell spacing, you may adjust these attributes by hand in the code or add them to your table using your HTML editor's

interface. Dreamweaver, for instance, allows you to set these attributes for the selected table in the Property inspector.

3. In each of the table cells, type the words that you want to appear on the buttons, from left to right.

For this example, type **Home, About, Services, Clients, Contact**.

Because your table is 500 pixels wide, each cell in your table should be approximately 100 pixels wide.

4. Convert all your navigation text into hyperlinks.

To do this, you can either enter filenames for each hyperlink href attribute, such as ``, or add a null link to the href attribute, as in ``.

Your table code should look like the following:

```
<table width="500" border="0" cellpadding="3"
  cellspacing="0">
  <tr>
    <td><a href="#">Home</a></td>
    <td><a href="#">About</a></td>
    <td><a href="#">Services</a></td>
    <td><a href="#">Clients</a></td>
    <td><a href="#">Contact</a></td>
  </tr>
</table>
```

5. To see how the table looks before you style it with CSS, launch the page in your favorite browser.

By default, all hyperlinks are royal blue and have an underline, as shown in the example in Figure 6-6. To make the links look more button-like, you'll next create a style that removes that underline.

Be sure to continue viewing your page in a browser after each of the remaining steps.



Figure 6-6: Your table looks pretty bland before adding style to your links with CSS.

6. Before the closing `<head>` tag in the code, type the following code for an internal CSS, but don't write any style declarations yet:

```
<style type="text/CSS">
<!--
-->
</style>
```



Internal CSS requires that these style and comment tags surround your style declarations, while external CSS files do not.

- 7. Create a custom class style in the internal CSS called `.cells` and enter hexadecimal values for the background color and 1-pixel border. (See the Cheat Sheet at the front of this book for a list of common hexadecimal values.)**

For the example, choose a blue background color with the hexadecimal value of `#66CCCC` (which can be shortened to `#6CC`) and a 1-pixel solid gray border with the hexadecimal value of `#333333` (which can be shortened to `#333`).

Your internal CSS code should now look like this:

```
<style type="text/CSS">
<!--
.cells {
  border: 1px solid #333;
  background-color: #6CC;
}
-->
</style>
```

- 8. Apply the `.cells` custom style to all of your table's opening `<td>` tags, as shown in the following line of code:**

```
<td class="cells"><a href="#">Home</a></td>
```

Custom styles, which are discussed in Book III, Chapter 4, must be hand-applied to tags that need to be styled before those styles can be rendered in a browser window. Figure 6-7 shows what your table should look like at this point.

Launch the page in a browser now to see how adding the `.cells` style improved the look of the navigation table.

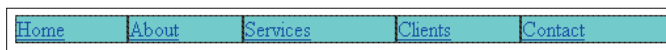


Figure 6-7: Add style for the table cells to help customize your navigation menu.

- 9. Create another custom class style and call it `.navlinks`.**

This style will have an attribute that removes any text decoration, such as the default link underline. The CSS code should now look like this:

```
<style type="text/CSS">
<!--
.cells {
  border: 1px solid #333;
  background-color: #6CC;
}
```

```
    }  
    .navlinks {  
        text-decoration: none;  
    }  
-->  
</style>
```

You could also create CSS styles for all the link states, but for this example, that's unnecessary because you'll be changing the background color of the table cells with JavaScript.

- 10. Now that you've created the style to remove the underlines, apply the `.navlinks` custom style as attributes to each of the table cell's link tags, as shown in the following code:**

```
<td class="cells"><a href="#"  
    class="navlinks">Home</a></td>
```

Relaunch the page in a browser. The blue underlines are gone!

The default table cell alignment is left and middle, which doesn't make the navigation links look very organized, so . . .

- 11. Override cell alignment settings by adding the center text align property to the existing `.cells` style:**

```
<style type="text/CSS">  
<!--  
.cells {  
    border: 1px solid #333;  
    background-color: #6CC;  
    text-align: center;  
}  
.navlinks {  
    text-decoration: none;  
}  
-->  
</style>
```

To see the center alignment style take effect, relaunch the page in a browser.

- 12. Now comes some fancy footwork! To change the background color of the table cells, insert some simple JavaScript in each opening `<td>` tag.**

The JavaScript contains inline CSS style instructions that tell the browser how to style the links for the mouseover and mouseout states:

```
<td class="cells" onmouseover="style.backgroundColor='#FC3';  
    onmouseout="style.backgroundColor='#6CC'"><a href="#"  
    class="navlinks">Home</a></td>
```

Apply the JavaScript — including all the little single and double quotation marks and other characters — to each opening `<td>` tag in the table. If desired, feel free to substitute the hexadecimal colors in this example with different colors using the hex values of your choice.

13. Launch the page in a browser to test the JavaScript.

Now, not only are the underlines gone from your links, but the background color of each table cell also changes when you position your cursor over any part of the desired table cell in a browser window. Figure 6-8 shows an example of this navigation menu with one of the buttons in its over state.

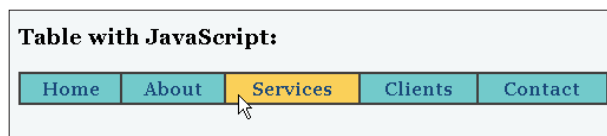


Figure 6-8: By adding CSS and JavaScript, you can customize the look and interactivity of your text navigation menus.



To view a working version of horizontal, vertical, and table/JavaScript text-based navigation systems, visit www.dummies.com/go/webdesignaio.

Creating Rollover Button Graphic Navigation Menus

Now that you know how to create rollover text-based navigation, you can apply the same principles to create custom rollover button graphic navigation menus. Rollover button graphic navigation menus consist of a series of hyperlinks that use JavaScript to control the visibility of two same-sized button graphics: One graphic shows the button’s normal state, and the other graphic shows the button’s “over” state, as illustrated in Figure 6-9.

In the following sections, you find out how to build rollover graphic menus, output rollover graphics in Fireworks, and create rollover buttons in Dreamweaver.



Figure 6-9: Rollover buttons need two graphics: one for the normal state (top) and another for the over state (bottom).

Understanding how to build rollovers

To build a rollover, you need two button graphics for each link on your menu. Both button graphics in each set must have the same width and height so that the JavaScript rollover effect displays both graphics smoothly. Otherwise, if one of the graphics is different in size than the other, the over state graphic will be stretched or squashed to match the same dimensions as the normal state graphic, giving the over state graphic a skewed effect. Not good.

When building your rollover button graphic navigation menu, place each of your links inside a table cell, or simply list them side by side in the code inside a `<div>` tag so that you can apply CSS to the layer if needed.

JavaScript for rollovers usually requires up to three different script parts (depending on how the script was written), and those parts must be placed in specific locations within the HTML code for the JavaScript to function properly in a browser:

- ✓ **Preload script:** This part of the script must be placed in the `<head>` area of the code. When the page is loaded into a browser window, this bit of script is what preloads the rollover state graphics into the visitor's browser's cache so that by the time the visitor mouses over a button, the over state button graphic is ready to appear. Without the preload script, a delay would occur in the rollover functionality.
- ✓ **Event handler script:** This part of the script gets added to the opening body tag and tells the browser when to process a preload script. With a preload script that uses the `onload` event handler, the "when" means right as the page loads in the browser. Common event handlers include `onmouseover`, `onclick`, and `onload`.
- ✓ **Rollover script:** This part of the JavaScript is added to your HTML in line with the button graphics and contains the instructions to the browser on how to handle the rollover functionality for the normal and over mouse states.



These aren't the only locations for JavaScript within a Web page. For instance, when flanked by `<script>` tags and placed between the opening and closing `<body>` tags of the page, JavaScript code executes immediately as the page loads in a browser. To have the script load at other times, it must be moved elsewhere in the code and then called upon to execute when a visitor's mouse movement triggers a particular event.

In the case of rollover button graphics for a navigation menu, the JavaScript must be added to all three of the following locations:

- ✓ **Between the opening and closing <head> tags:** These function scripts must be placed between two <script> tags somewhere inside the head of the code. For rollovers, this script contains parameters for a preload function.

```
<head>
<script type="text/JavaScript">
<!--
...
-->
</script>
</head>
```

- ✓ **Inside the opening <body> tag:** The event handler part of the script instructs the browser how and when to execute any scripts coded in the <head> of the page, often using the onload attribute, as in

```
<body onload="preloadImages();">
```

or

```
<body onload="MM_preloadImages('images/button1-
over.gif')">
```

- ✓ **In the code, between the <body> tags:** JavaScript placed in line with the rest of the HTML code will be executed in the browser based on the parameters and instructions contained within the JavaScript in the <head> and opening <body> tag. For example, a JavaScript for a rollover button contains instructions for swapping two images based on two mouse events, onmouseover and onmouseout:

```
<a href="contact.html" onmouseout="MM_swapImgRestore()"
onmouseover="MM_swapImage('Image1','','images/button-contact-
over.gif',1)"></a>
```

The actual JavaScript you use for the rollover functionality that identifies the graphics that will be swapped for each button can be written in several different ways, depending on who wrote the script, even though, in all likelihood, the scripts would all essentially function the same.



To find the right code for your menu, you have two options: You can use the JavaScript supplied by your HTML/code editor or image optimization program (such as Dreamweaver or Fireworks), or you can search the Internet for free JavaScript that can handle the rollover effect. Two sites that provide quite a few free rollover button and other JavaScript scripts include <http://javascript.internet.com> and www.javascript.com.

Outputting rollovers in Fireworks

Up until a few years ago, you used to be able to design a set of rollover buttons in Photoshop and then quickly reopen the file in ImageReady. There you could easily apply rollover behaviors and output both parts of the graphic (normal and over states) along with the HTML and JavaScript to make the rollovers work. Sadly, ImageReady was discontinued after the Adobe CS2 Suite was released, and Adobe has not chosen, for whatever reason, to move ImageReady's rollover button functionality to Photoshop. My guess is that they might want more people to start using Fireworks. Still, a lot of designers have decided to either stick with ImageReady or just optimize the graphics separately and use an HTML editor like Dreamweaver to write the rollover button JavaScript for them.

If you're interested in working with Fireworks, use the following steps to create a simple button with normal and rollover states and then output the optimized graphics along with an HTML file loaded with JavaScript:

- 1. Open a new document in Fireworks by choosing File⇨New.**
- 2. When the New Document dialog box appears, set the canvas size to 100 pixels wide and 30 pixels high, leave the resolution at 72 pixels/inch, and set the canvas color to white. Then click the OK button.**

A new untitled Fireworks .png document window appears in the workspace.

- 3. The button needs to have two parts for the normal and over button states. To achieve this, first draw a rectangle on the artboard using the Rectangle tool found in the Vector shapes area of the toolbar.**

To draw the rectangle, drag a rectangular shape on top of the artboard and release your mouse button. The shape remains selected, allowing you to modify that shape's properties in the Properties panel.

- 4. In the Properties panel, set the width and height of the rectangle to 100x30 and the X/Y coordinates to 0/0.**

This positions your rectangle directly on top of the artboard space. The rectangle will be filled with the last color used. If desired, select a different color through the fill color box in the Properties panel or by clicking a swatch in the Swatches panel. For example, you may want to select a bright green with the hex value of #99CC00.

- 5. To add text to the button, select the Text tool from the toolbar and specify the desired font attributes in the Properties panel before adding the text.**

To type, simply click once to set the insertion point with your cursor and begin typing a word, such as **CONTACT**.



For example, you could select Arial, Bold, 16 pixels, and black as the font. While the text can be any color, the stroke color should be set to None.

To reposition the word in the center of your rectangle, select the Pointer tool (the black arrow) from the toolbar and then click and drag the word into the desired location.

6. Create the second state or frame for the rollover state of the button graphic.

In the upper-right corner of the States panel (or in the Frames and History panel if using Fireworks CS3 or older), click the options menu and choose Duplicate State (or Duplicate Frame).

This creates a duplicate of the current state (frame) with the same panel, with the same colored rectangle and text.

7. With the second state (frame) still selected in the States (Frames and History) panel, select the second state rectangle and change its color in the Properties panel.

If desired, you may also change the color of the button text in State 2, as illustrated in Figure 6-10.

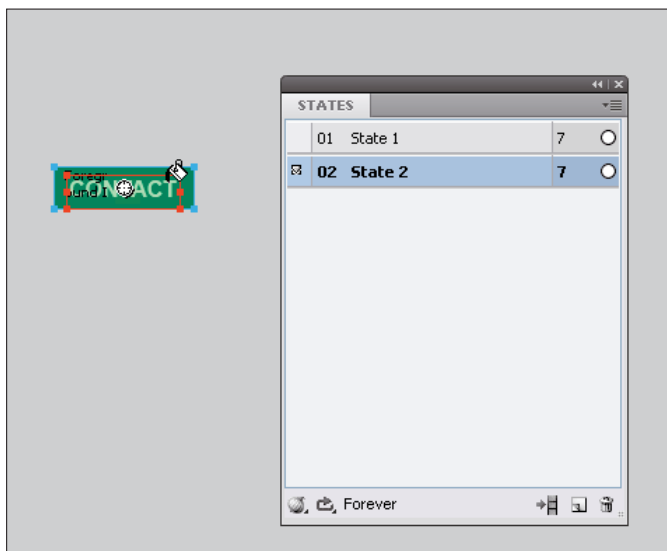


Figure 6-10: Select the rectangle and text in State 2 to modify the colors and other attributes for the button’s rollover state.

8. To apply the JavaScript behavior and create the rollover button effect, you must first apply a slice to your rectangle. Select the outer rectangle shape on State 1 (Frame 1 in earlier versions) and choose Edit⇨Insert⇨Rectangular Slice.

9. Open the Behaviors panel by choosing Window⇨Behaviors, click the plus sign in that panel, and choose Simple Rollover.

Upon selecting this option, the `onMouseOver` simple rollover action is added to the Behaviors panel. This is the JavaScript.

10. To preview your rollover button in Fireworks, click the Preview button at the top of the document window and move your cursor over the button graphic.

When you move your cursor over the button graphic, the normal state button is replaced with the over state button, and when you move your cursor off the button, the normal state reappears.

11. To add a filename or URL to the button graphic, enter the filename in the Link field in the Properties panel.

You may also add alternate text for the button in the Alt field and set a link target on the Target menu in the Properties panel. For example, if you wanted your button to take visitors to the Contact page, you'd enter **contact.html** in the Link field, **Contact** in the Alt field, and **_self** in the Target field.

12. When you are ready to save your rollover button, you must first select an appropriate output option. Open the Optimize panel and select the GIF option (not animated GIF) from the Export File Format drop-down menu.

If your graphic contains gradients or photos, the JPEG format might work better. Alternatively, like the GIF setting, the PNG 8 option also works well for button graphics with large, flat areas of color.

13. To export your rollover button, choose File⇨Export.

14. When the Export dialog box opens, enter the desired filename (such as `rollover.html`) and set the Export option to HTML and Images, the HTML option to Export HTML File, and the Slices option to Export Slices. When ready, click the Save button.

Upon clicking the Save button, Fireworks exports button graphics for each of the two button states and an HTML file that includes the JavaScript for the rollover functionality.

15. To test the button within the HTML file, drag and drop the HTML file into an open browser window.

Figure 6-11 shows an example of how your rollover button might look in a browser.

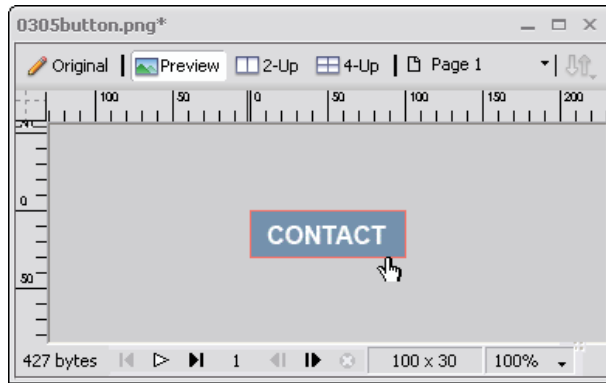


Figure 6-11: Test your graphic rollover buttons in a browser window.

- 16. Save your Fireworks button graphic file for future use as a Fireworks PNG file by choosing File⇨Save.**

Creating rollovers in Dreamweaver

Dreamweaver users can create rollover buttons by two methods, both of which use different JavaScript code. The first method involves attaching the Swap Image and Go to URL behaviors to already inserted normal state graphics on the page. The second method, described in the following steps, allows you to choose both the normal and rollover graphics, select the page the rollover button will hyperlink to, and enter alternate text for the button link, all using the Insert Rollover Button option.

A *behavior*, in this context, is any interactive JavaScript that Dreamweaver can insert into the code of your page. Behaviors pair an event with an action that is triggered by the event, such as changing the normal state button graphic to the over state button graphic (the action) when a visitor hovers the mouse above it (the event). Dreamweaver comes preinstalled with about 22 behaviors to help designers quickly configure the interactive features on their Web pages.

To complete the following steps, you need two optimized button graphics of the same width and height: one for the normal state and the other for the mouseover state of the rollover button. Your button graphics can be any color and size and include any special effects and styles as desired. For example, you might create a graphic that is 80 x 20 pixels in size, includes the text SHOP in the center, has a smooth, beveled edge, and has a blue background for the normal state and a green background for the mouseover state. If you've already completed the Fireworks steps in the preceding section, feel free to reuse your two button graphics for the Dreamweaver steps here.



Follow these steps to make Dreamweaver insert a rollover button on your page and automatically write all the necessary JavaScript in your code:

1. **Launch Dreamweaver and choose File→New to open a new document in the Dreamweaver workspace.**
2. **Save the file with the filename `rollover.html` to the folder of your choice. Within that folder, create another folder called `images` and place a copy of your two optimized button graphics inside it.**

For best results, define a managed site in Dreamweaver with the root-level directory set to the folder you just saved the `rollover.html` file in. For a tutorial on defining a Dreamweaver site, see www.adobe.com/go/lrvid4050_dw.

3. **In Design view, place the insertion point inside the `rollover.html` file where you'd like the rollover button to be added to the page.**

You must specify the location before adding the graphics so that Dreamweaver knows where to insert the JavaScript and images.

4. **To add your rollover button to the page, choose Insert→Image Objects→Rollover Image.**

This opens the Insert Rollover Image dialog box, as shown in Figure 6-12.

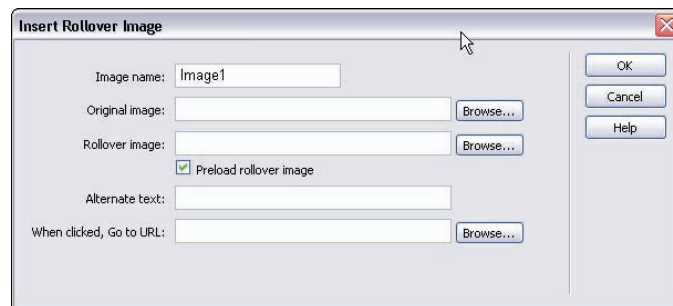


Figure 6-12: Use Dreamweaver's Insert Rollover Image dialog box to quickly add rollover buttons to your pages.

5. **In the dialog box, type a name for the rollover button, browse to and select the graphics to use for both the original and rollover image states, and add alternate text to mirror any text on the button graphic. Then add a filename, URL, or null link for the hyperlink.**

The image name that you provide acts as an `id` attribute for the image that the JavaScript attaches to. The Preload Rollover Image option is enabled by default to add the preload script to the HTML. Leave that option enabled.

6. Click the OK button to close the dialog box.

Upon closing the dialog box, Dreamweaver inserts the rollover button normal state graphic and JavaScript onto the page at the insertion point.

Take a peek at the code so that you can see that JavaScript has been placed between the <head> tags and in the opening <body> tag, as well as at the insertion point in the body of the page inside the tag that contains the normal state button graphic.

7. Save the page and preview it in a browser by selecting any of the browsers listed on Dreamweaver’s File→Preview in Browser menu, or click the Live View button to preview the button within the Dreamweaver workspace.

To test the rollover button functionality, move your cursor over the button graphic.



To make an entire navigation menu using this Dreamweaver JavaScript rollover button technique, create a table with the appropriate number of cells across a single row or column, or insert a layer using the <div> tag and repeat the preceding steps to insert each of the rollover buttons in the menu.

Creating Multitier Spry Menus in Dreamweaver

When it comes to multitiered JavaScript menus, many of them can look and function well even without the use of button graphics. In fact, you can create an entire menu with any number of fly-out submenus, all with the use of JavaScript and a little CSS styling. Like any JavaScript function, you can write the code in a variety of ways — or simply use scripts that are free or for sale. Some free scripts come preinstalled with code editors like Dreamweaver, and others are shared openly online. Specialty scripts that are for sale often incorporate interesting effects like fades, swipes, and other unusual DHTML (dynamic) transitions.

Up until a couple of years ago, adding JavaScript menus through the Behaviors panel was one of the fastest and easiest ways to create a multitiered navigation menu in Dreamweaver. Then, starting with Dreamweaver CS3, a new JavaScript menu system was introduced using the Spry framework (a robust combo of HTML, CSS, and JavaScript) to replace the old Dreamweaver Show Pop-Up menu. This new menu system, which uses the Spry Menu Bar widget, can be easily inserted, configured, and customized through the Property inspector and CSS Styles panels. Spry menus support an unlimited number of submenus for each main menu item and are comprised of list item HTML tags (, , and <a>) with CSS styling.

To use the Spry Menu Bar widget, just insert the widget onto your page and use the Property inspector to specify the name and order of each of the main links and subnavigation menus. When the structure is in place, you can further customize the presentation features of the menu by changing such things as the font face, font color, and position through the CSS Styles panel. Best of all, Dreamweaver handles all the code writing for you! Figure 6-13 shows an example of a completed Spry Menu Bar as it appears in Design view in the Dreamweaver workspace.

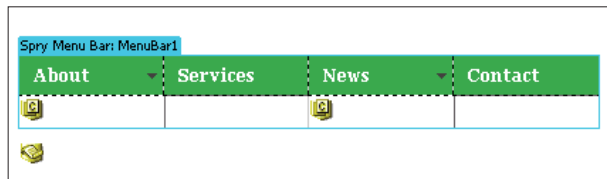


Figure 6-13: Dreamweaver’s Spry Menu Bar uses HTML, CSS, and JavaScript.



In contrast to the JavaScript menu examples discussed earlier in this chapter — which may have only added scripts to the `<head>` area, to the `<body>` tag, and inside the body of the page — each time you insert a Spry widget into your page, a corresponding CSS file (such as `SpryAssets/SpryTabbedPanels.css`) and JavaScript file (such as `SpryAssets/SpryTabbedPanels.js`) are added to the root level of your managed site inside a folder called `SpryAssets`. Sometimes, the widget may include graphics too, and those are also automatically added to the `SpryAssets` folder. You must upload this folder and its entire contents, along with any pages that contain the Spry widget, to the host server for the widget to function properly.

Follow these steps to insert a Dreamweaver Spry Menu Bar on your page:

- 1. Choose File⇒New to open a new document in the Dreamweaver workspace. Name the file `sprymenu.html` and save it to the folder of your choice.**

For best results, define a managed site to the folder in which you saved the `sprymenu.html` file.

- 2. Place your cursor inside the area on your page where you want to insert the Spry widget and click the Spry Menu Bar widget button in the Spry category of the Insert panel.**

When inserting the Spry Menu Bar, you are prompted to select either a horizontal or vertical layout before Dreamweaver adds the code to the page. For this example, choose horizontal.

By default, the widget is generically named something like `MenuBar1`.

- 3. To change the name of the widget, modify the widget's `id` attribute in the Menu Bar name field on far left side of the Property inspector.**

After the name change has been made, Dreamweaver automatically updates all instances of the widget's ID within the corresponding JavaScript and CSS files in the `SpryAssets` folder.

- 4. To customize the items on the menu, select the Item 1 menu link in the Property inspector and adjust that button's settings in the Property inspector Text, Link, and Title fields.**

To help you understand how to customize the Spry menu, Dreamweaver automatically configures each inserted Spry Menu Bar with a default set of four menu buttons that include submenus for menu buttons 1 and 4.



When modifying the Spry menu's items through the Property inspector, you must select the entire menu by its blue Spry Menu Bar tab in Design view to see the Menu Bar properties. If you accidentally try to select a button in Design view instead, which changes what's displaying in the Property inspector, simply reselect the Spry menu in Design view by its blue tab.

- 5. Using the tools on the Property inspector, customize your menu by adding, removing, labeling, and reordering the menu and submenu items, and by applying links, link titles, and link targets.**

Use the following fields on the Property inspector to assist you in customizing your menu:

- *Text:* Type a text label for each menu item, such as About Us.
- *Add Item:* Click the Add Menu Item (+) button above the first menu box to add another menu item. To add a submenu item, select the desired main menu item and then click the Add Menu Item (+) button above the second or subsequent menu boxes. This ensures that the main menu includes the correct corresponding submenu item(s). Repeat as needed to create further sub- and sub-subnavigation menu items.
- *Remove Item:* To delete an item from the main menu or from any of the submenus, select the menu item within the desired menu box and click the Minus (-) button.
- *Link:* Type the filename with extension or the entire URL of the target link destination, as in `about.html` or `http://www.adobe.com`. If desired, use the folder icon to browse for and select a file on your computer.
- *Target:* To improve code accessibility, select a target or frame for the link that will determine where the link opens relative to the linking browser window. Unnamed frames will not appear in the list. To find out more about link targets, turn to Book III, Chapter 1.

- **Move Item Up/Down:** Adjust the order of any menu or submenu items by selecting the item that needs to be moved and clicking the up or down arrow until the item is in the desired location.
- 6. To modify the appearance of your menu buttons, you must modify the CSS through the linked `SpryMenuBarHorizontal.css` file listed in the CSS Styles panel.**

Use the following guide to assist you with editing specific styles:

- *Submenu Border style:* To change the border attribute for submenus, modify the CSS for the style called `ul.MenuBarHorizontal ul`.
- *Normal button style:* To change the text color and background color for the buttons on the menu, modify the CSS for the style called `ul.MenuBarHorizontal a`.
- *Menu Over button style:* To edit the text color and background color for the over state of the menu buttons, modify the CSS for the style called `ul.MenuBarHorizontal a:hover`, `ul.MenuBarHorizontal a:focus`.
- *Submenu Over button style:* To edit the text color and background color for the over state of the submenu items, modify the CSS for the style called `ul.MenuBarHorizontal a.MenuBarItemHover`, `ul.MenuBarHorizontal a.MenuBarItemSubmenuHover`, `ul.MenuBarHorizontal a.MenuBarSubmenuVisible`.



To find out more about customizing your Spry menus through the CSS Styles panel, select the Spry menu by its tab in Design view and click the **Customize This Widget** link on the Property inspector or visit www.adobe.com/go/learn_dw_sprymenubar_custom.

- 7. Save the HTML page with your Spry menu by choosing File⇨Save.**

Dreamweaver alerts you with a dialog box reminder, like the one shown in Figure 6-14, about the `SpryAssets` folder that must be uploaded to the server for your Spry menu to function properly in a browser.

- 8. To preview your Spry menu in a browser window, select one of the browsers listed on the File⇨Preview in Browser menu.**

To experience the Spry menu dynamic functionality, move your cursor over any of the menu buttons. Figure 6-15 shows an example of what the menu might look like when you move your mouse over one of the Spry menu buttons.

If you want to make changes to the contents or style of your Spry menu, you may make further adjustments to it by reselecting the menu by its blue tab in Design view and by altering the styles in the CSS Styles panel.



Figure 6-14: Be sure to upload the special SpryAssets folder to the server along with your Spry menu pages to ensure that your menu functions properly.



Figure 6-15: Spry menus dynamically display dynamic rollovers and submenus.

Creating CSS List Navigation Menus

One of the cleanest ways to create a navigation menu is to combine HTML list formatting with CSS. This method allows you to use text and graphics while keeping the code uncluttered. It also produces the fastest-loading and most accessible type of navigation system when compared to any of the JavaScript navigation methods.

Menus created with HTML list code can be styled with CSS in several ways, and depending on your needs, one approach might be more suitable for a particular site than another. The simplest method is to nest an unordered list inside a `<div>` container tag with an `id` attribute and then create CSS styles for that `id` as well as for the `` and `` tags and all the hyperlinks on the menu. You can then style each hyperlink with two graphics: one that forms the left side of the menu buttons and one that forms the buttons' right sides. However, rather than creating two sets of graphics for each of the normal and over states as you would with simple JavaScript rollover buttons, with the CSS method you simply stack the normal and over button shapes inside a single graphic file. In other words, in the left-side button graphic, you stack the normal state left button edge directly on top of the over state left button edge. Then you do the same thing for the right button, as shown in Figure 6-16.

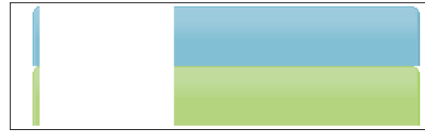


Figure 6-16: Stack the normal and over state button designs inside a single optimized graphic for both the left and right side of the button.



In the following steps, you discover how to build a list navigation menu, like the one shown in Figure 6-17, using graphics and CSS. To do this, you can either create and use your own graphics or visit www.dummies.com/go/webdesignaio to download a copy of the images shown here.



Figure 6-17: CSS list navigation menus can use any kind of graphic and any length of button text.

As you will see after downloading them, these images are a lot larger than the menu list buttons they'll be styling. This is because the images are styled in the list as background images, aligned to the upper-left and upper-right corners for both the regular link and hover link states, and are meant to be big enough to accommodate both short and long buttons.

If you will be creating your own graphics with different dimensions than the ones shown here, you will need to adjust the code — specifically the offset in the hover — so that the mouseover effect functions properly. What is essentially happening here in this example is that you are creating CSS that

toggles between the top half and the bottom half of the graphic so that the rollover button displays both normal and over states.

Follow these steps to build a CSS-styled list navigation menu:

- 1. In your favorite HTML editor, open a new blank HTML page and save it with the name `csslistmenu.html`.**

To help keep things organized, set up a folder on your computer called `List Menu` and save your `csslistmenu.html` file to that folder. Also inside that folder, create another folder called `images`, and inside that `images` folder, place a copy of the two button graphics.

- 2. Within the body of the page, create an unordered list with the following six list items: About Us, Our Services, Our Clients, Press Releases, Employment Opportunities, and Contact Us.**

You don't have to include a list type attribute on your `` tag to specify which bullet type to use for this particular unordered list because you'll be styling the list with graphics in the CSS.

- 3. Wrap the entire list with a pair of `<div>` tags and give the opening `<div>` the ID of `id="tablistmenu"`.**

Your code should look like this:

```
<div id="tablistmenu">
  <ul>
    <li>About Us</li>
    <li>Our Services</li>
    <li>Our Clients</li>
    <li>Press Releases</li>
    <li>Employment Opportunities</li>
    <li>Contact Us</li>
  </ul>
</div>
```

- 4. Convert each item in the list into a hyperlink, either by adding a filename or using the null link number symbol (`#`), and include a title attribute for each of the `<a>` tags to make the links accessible, as in the following example:**

```
<li><a href="#" title="Contact Us">Contact Us</a></li>
```

Figure 6-18 shows an example of what your list may look like at this point when launched in a browser window.

Null links are great to use as stand-ins for real links when building components of a page like navigation menus. Later, when the whole component or page is complete, you can replace the null links with the real links.



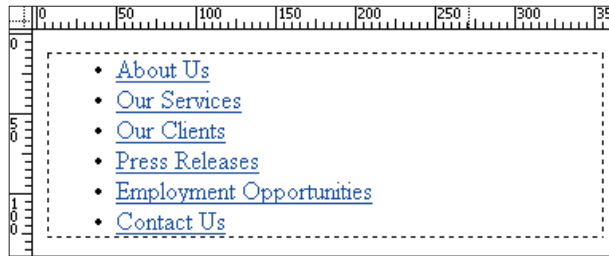


Figure 6-18: This type of CSS navigation system starts as a simple unstyled list.

5. Add a pair of `` tags around each link item, close to the content between the link `<a>` tags, as in the following sample code:

```
<li><a href="#" title="Contact Us"><span>Contact
Us</span></a></li>
```

Each button on the menu needs a background image with curved corners; however, because the width of each menu item is determined by the length of the text within each link item, you must use two graphics to build the button background: one graphic for the left side and another for the right side. The left side of the button will be applied by creating a style for the `` tags, and the right side of the button will be applied using these `` tags. When combined, they give the illusion of a single background image.

6. To style the list, create an internal CSS and make a tag redefine style for the `<body>` with a `0px` margin and `0px` padding, and the font set to Verdana, bold, 11px, with a 1.5em line height:

```
<style type="text/CSS">
<!--
body {
margin: 0px;
padding: 0px;
font-family: Verdana, Geneva, sans-serif;
font-size: 11px;
line-height: 1.5em;
}
-->
</style>
```

Later, when the navigation is complete, you can move the CSS (on your own) to an external CSS file.

7. To separate the body style from the menu styles you're about to create, add a Menu Styles comment to the internal CSS, right after the `<body>` redefine style definition:

```

<style type="text/CSS">
<!--
body {
    margin: 0px;
    padding: 0px;
    font-family: Verdana, Geneva, sans-serif;
    font-size: 11px;
    line-height: 1.5em;
}
/* :::MENU STYLES::: */
-->
</style>
    
```

See the nearby sidebar for more about the benefits of adding comments to CSS files.

- 8. Create a new compound selector style in the internal CSS for the `<div>` tag with the ID of `tablistmenu` with the following style attributes:**

```

#tablistmenu {
    float: left;
    width: 100%;
    background: #cae4ef;
    border-bottom: 1px solid #0e5d7e;
    font-size: 95%;
    line-height: normal;
}
    
```

This style defines a space behind the navigation buttons that stretches across the full width of the browser window, has a blue background color, includes a 1-pixel solid navy blue border along the bottom edge of the `<div>` container, and slightly reduces the font size of the text in the menu list.

- 9. To style the items list, create a tag redefine style for the ``.**

However, rather than simply list `` as the selector, use compound CSS syntax to write a contextual descendant selector that identifies the `<div>` with the ID of `tablistmenu` as the sole location for the `` styles, like this:

```

#tablistmenu ul {
    margin: 0px;
    padding: 10px 10px 0px 50px;
    list-style: none;
}
    
```

This `` style sets the margin for the entire list to 0px; adds padding on the top, right, and left sides of the list; and sets the list style (where a customized bullet graphic might go) to none.

- 10. Create a style for the `` tags so that each item will display in line (in a row) with no margin or padding:**

```
#tablistmenu li {
    margin: 0px;
    padding: 0px;
    display: inline;
}
```

If your code editor doesn't have a preview or design pane, launch your page in a browser to see the list displaying in line. At this stage, your list should look something like the example in Figure 6-19.



Figure 6-19: Styling the `` tags gives your CSS navigation system a horizontal layout.

- 11. To add the left side of the background image to each linked item in the list, create a compound ID style that will be automatically applied to each link inside the list. This style specifies the background image and a few other important CSS attributes:**

```
#tablistmenu a {
    float: left;
    background: url(images/cssmenuleft.gif) no-repeat
        left top;
    margin: 0px;
    padding: 0px 0px 0px 4px;
    text-decoration: none;
}
```

Note that the background image is set to `no-repeat` with an orientation relative to the upper-left corner of the list item link. This image handles creating the look for the left side of each button on the navigation menu. To style the other half of the buttons, you must create a second style that wraps around the link text using the `` tag.

- 12. To add the right side of the background image to each linked item in the list, create a new compound style that will be applied automatically to the `` tags that surround each link in the list:**

```
#tablistmenu a span {
    float: left;
    display: block;
    background: url("images/cssmenuright.gif") no-
        repeat right top;
    padding: 5px 10px 4px 6px;
    color: #036;
}
```

Adding comments to CSS files

To help you keep track of where certain styles begin and end in the CSS without worrying about having those comments being displayed somewhere on the Web page, add special comments between CSS comment tags whenever different style sets, like the menu list styles or link states, are introduced. In addition to being helpful to anyone else viewing your CSS, comments also give your CSS a more organized, polished look.

Comment tags within a CSS use the `/* text */` syntax (which is different from HTML comment tags, `<!-- like this -->`), where the text part can contain any text you like, spanning as many lines in the code as needed.

To illustrate, you might add the following comment and comment tags to your CSS code to mark the beginning of the menu button styles:

```
/* :::MENU STYLES::: */
```

The semicolons (`:`) in this example help the eye identify the comment tag more readily. Certainly, however, you can add other special characters

or use none, leaving only descriptive text between the comment tags, in normal uppercase and lowercase lettering (Menu Styles) or in all caps, as shown here.

When any section of styles is fairly long, you might also want to include ending comments and comment tags for the section, such as

```
/* :::END MENU STYLES::: */
```

In external CSS files, these CSS comment tags (and the contents inside of them) can stand alone between the style rules. However, when including CSS comment tags on an internal CSS, you must make sure that the special CSS comment tags fall between the regular HTML comment tags (`<!--` and `-->`) in the code:

```
<!-- /* .....MENU STYLES..... */ -->
```

If you forget to include the regular HTML tags, both the style declarations and the comments will appear in the body of the page! Therefore, for simplicity's sake, try to put all your CSS into an external CSS file.

This background image is also set to `no-repeat` but has an orientation relative to the upper-right corner of the list item link. In addition, this style specifies the color of the text on each link — in this case, navy blue using the hex value of `#003366`, or simply `#036`.

Internet Explorer 5 for the Mac has a bug that prevents this list from displaying correctly. You must add the following `#tablistmenu a span` style information to your CSS to fix that bug. Using the comments can also help visually identify this “hack” from the rest of the styles in your CSS:

```
/* ::: This hack fixes a bug in MAC IE5 ::: */
#tablistmenu a span {float: none;}
/* ::: End of hack for MAC IE5 ::: */
```

13. Create three styles in the CSS to control the position of the background graphics on the left and right side of the navigation menu buttons when a visitor hovers his or her mouse over the buttons.



This part requires three new CSS styles, which work as follows:

- The first style sets the color that the link text will change to for the hover state of each link.
- The second style handles the position of the left background image when a visitor hovers his cursor over the buttons. In the case of this menu, the background position of the over state part of the graphic begins at exactly -42 pixels down from the upper-left edge of each graphic.
- The third style does the same thing for the background image on the right side of the button. The horizontal and vertical background positioning attributes must be adjusted for both the list link and link span hover states to function accurately:

```
#tablistmenu a:hover span {
    color: #0e5d7e;
}
#tablistmenu a:hover {
    background-position: 0% -42px;
}
#tablistmenu a:hover span {
    background-position: 100% -42px;
}
```

14. Save the page and preview it in a browser by selecting any of the browsers listed on the File⇨Preview in Browser menu.

With the page open in your browser, move your cursor over the list items to see how each of the button graphics changes on mouseover!

The beauty of this particular CSS list menu is its simplicity. Each button uses HTML list item text styled with CSS, and the same two graphics are used repeatedly for the background of each button. As with all Web graphics, after the image is loaded into the visitor's browser cache, the reuse of that image no longer requires downloading and recaching. This menu also saves tons of file space, which translates into faster page download times in the browser. Better still, when visitors using assistive devices come to the page (which ignore CSS), as well as any visitors who may have turned off their browsers' CSS functionality, the links in this particular menu will still be organized and accessible.

Chapter 7: Designing Web Forms

In This Chapter

- ✓ Determining which data to request from visitors
- ✓ Encrypting collected form data
- ✓ Building validating Web forms
- ✓ Using Dreamweaver's Spry Form fields
- ✓ Testing and publishing Web forms

If you have ever filled out an online survey, signed up for a Web site's newsletter, or purchased something on the Internet, you've probably used a form. While the forms themselves come in many shapes and sizes, all forms contain specific HTML tags, often combined with JavaScript or some programming language, that allow sites to collect information from visitors for a variety of different reasons, including to sign up for services, request information, join a mailing list, purchase products, register for events, pay bills, handle online banking, and much more.

Though not every Web site includes a form, as a designer you should understand what forms are and how they work so that you are poised to build one when the need arises. Furthermore, despite its somewhat complicated sounding functionality, building a form in HTML is pretty easy because you only need to use a handful of tags to create the individual form fields. After you determine which information you'd like to collect from visitors, you can begin to organize the form contents into a neat table format, complete with form labels on one side and form fields for user input on the other.

After building the form in HTML, the next thing you have to contend with is how to process that data. Unfortunately, by default, all forms are unsecure files. This means that any data collected could be easily pilfered — unless you take certain security measures. Though you might think that security shouldn't matter much unless you're collecting personal information like someone's name and address or credit card number, it does. Everything you collect is personal and requires protecting, from an e-mail address, account number, and username to whether someone reads magazines about fly-fishing or is interested in receiving further information about debt consolidation.

*Name:

*Telephone: Ext:

*Email:

I am a:

I am interested in: Web Design Illustration
(check all that apply) Print Design Other

Join Mailing List: Yes No

* Required fields

In this chapter, you find out how to build a Web form in HTML, add JavaScript validation to the form so that visitors will be assisted in completing the form accurately, insert and use self-validating Spry Form fields in Dreamweaver, and submit the data collected from a valid completed form to a remote location for secure processing. In addition, I briefly discuss form encryption and other security measures you can take to help keep that collected information safe.

Deciding What Visitor Information to Collect

To organize the layout of your Web form, take your cues from the information you'll be collecting from visitors. You may request any kind of data you like, including the following:

- ✓ Personal information, such as a visitor's name, address, phone number, fax number, and e-mail address
- ✓ Purchasing information, such as items ordered, a credit card, a billing address, and a shipping address
- ✓ Private account information, such as a username, account number, password, and password hint question
- ✓ Miscellaneous information, such as visitor feedback and opinions, visitor interests about a particular topic, or even something silly like the visitor's food preferences or favorite contestant on *American Idol*

Form information can be gathered from visitors using a variety of form fields, including single-line and multiline text boxes, check boxes, radio buttons, drop-down menus, and buttons. Figure 7-1 shows an example of a typical form that includes several of the more common form fields.

Online transactions, as with transactions made in person, often require the collection and sending of secure and personal data from one party to another. By federal law, this data — whether collected online or not — provides consumers and individuals with certain legal and ethical rights. For instance, a site may not legally collect information from minors. Furthermore, every Web site has a moral and legal obligation to inform visitors how collected data may be used. Many sites include a link to a privacy statement or similar policy that outlines what specific data the site owner is collecting and how that owner may use that data or disclose it to other parties, affiliates, and subsidiaries. To find out more about federal privacy laws, visit www.ftc.gov/privacy/ and www.usdoj.gov/oip/privstat.htm.

Figure 7-1: Use forms to collect personal, purchasing, and account data from visitors.



Try not to be too intrusive in your request for visitor information. Collect only what you (or your client) really need and nothing more. For instance, if the site plans to send regularly scheduled e-newsletters to registered visitors, is it really necessary to collect the visitor's complete name, mailing address, phone number, fax number, and cell phone number along with the e-mail address? Maybe, but maybe not. Collect only the information that has relevant usage for the site, because requesting too much information might deter people from submitting any personal information. Some Web sites get around the issue of wanting and needing information by including all the desired fields in the form but making only certain form fields required, rather than all the fields, for the form to be successfully submitted.

To simplify the process of collecting information from visitors, make your Web forms as user-friendly as possible. Forms should be easy to navigate with a mouse or with the Tab key, be easy to read, and not be too long. Forms should also be clear in their request of specific information, including providing hints about how the collected data should be formatted in the form input fields, such as omitting dashes and spaces from a phone number (2125551212) or making sure to include them (212-555-1212). Providing hints that instruct visitors how to fill in fields, as well as indicators (like an asterisk [*]) that identify which fields are required and must be completed before the form can be successfully submitted, are great additions to enhance form usability.

The easiest way to build a form is to set up all the form labels first. Many designers use tables and other HTML formatting code, such as paragraphs and line breaks, to organize their form labels and form input fields. Most forms can fit nicely inside a multirow, two-column table. Along the left column of a two-column table, you'd add the form labels, like Name, Address, and Telephone Number, to individual table cells. Along the right column of the table, you'd add the form fields inside the cells next to each label. When those steps are complete, you can add any field input hints (enter a 6- to 8-digit passcode) and required field indicators (* required).

After the form is built, you can then apply CSS styling to make the form match the look of the rest of the site. For unusual layouts, feel free to use nested tables to organize the fields and labels. As long as all the form fields reside inside a single set of `<form>` tags, whether or not those fields sit inside tables or in nested table cells will not affect form functionality.

Your main goal is to keep the information neat and organized so that the visitor knows intuitively how to navigate through the form.

Encrypting and Processing Collected Form Data

On their own, all Web pages are unsecure documents. Therefore, when transmitting files and collected data over the Internet, you must take special precautions to protect the information you're collecting, especially when that information is private and confidential, such as a credit card number, account number, username, password, or other personal information that a visitor might feel uneasy about sharing online. This online security is the full responsibility of the Web owner, or of the service provider performing the form data collection. When visitors come to your site, they need to be assured in some way that a dishonest outsider can't hack the information they're transmitting online, which could result in some form of identity theft.

In the following sections, you find out more about what data encryption is and how you can secure your site.

Deciding whether to purchase an SSL digital security certificate

The ultimate type of Web site security is the SSL (Secure Sockets Layer) digital security certificate. The SSL helps provide secure connections between the visitors and the host computer by encrypting all the collected and transmitted data. When an SSL certificate is detected, the browser alerts visitors of the site's security in the following ways:

- ✓ Most browsers display the URL with an `https://` instead of the normal `http://`.

- ✓ Some kind of icon informs the visitor of that site's security. Internet Explorer and Firefox, for instance, indicate to the visitor that a site uses SSL by displaying a small lock icon in the lower-right corner of the browser window. Likewise, some browsers display a lock icon in the address bar or at the upper-right edge of the browser window, or display the entire address bar field in another color.
- ✓ Other browsers provide additional proof to indicate that SSL certificates are present on a site, such as generating a special pop-up window that notifies the visitor when he or she has entered and exited a secure area on the site.

All these visual indicators provide the visitor with a sense of security about submitting personal information to a site. Figure 7-2 shows a few of the SSL secure lock icons you might see in your browser when visiting a secure Web site.

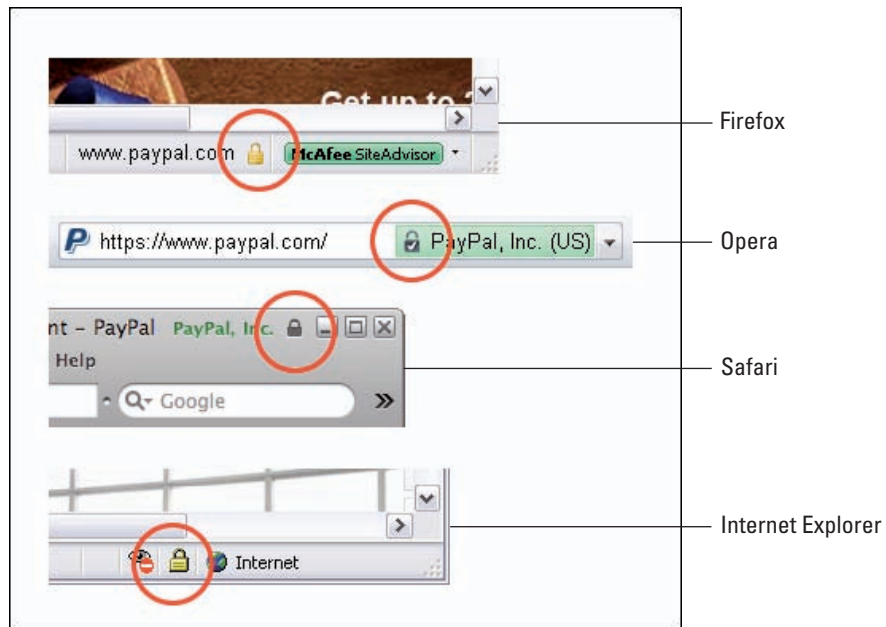


Figure 7-2: Secure sites that have SSL certificates display some kind of lock icon within the browser to alert visitors of the site's security.

You can purchase SSL certificates from several agencies, but before you do, be sure to check with your site host provider first to see whether its servers support the SSL company you'd like to use. The most popular (and most expensive) SSL providers are VeriSign (www.verisign.com), Thawte

(www.thawte.com), and GeoTrust (www.geotrust.com), but a few others exist, too. SSL certificates range in price from affordable to the more expensive. Expect to pay an annual rate between \$250 and \$1,500 for the certificate. Your host provider can often handle the procurement of the certificate for you, either with or without an added installation fee.

In some cases, purchasing an SSL certificate simply won't be cost effective. In those cases, you have three possible options for making the form data you collect from visitors as secure as possible:

- ✓ **Shared SSL:** Check with your host provider to see whether it offers some kind of less expensive, shared SSL certificate, where you buy into a group license that is installed on the server hosting your domain.
- ✓ **Data encryption:** Look into using alternate encryption solutions for the collected data, such as using a combination of JavaScript and browser cookies, creating a secure login script, or creating XForms with XML (all of which fall beyond the scope of this book). (For a good tutorial on XForms, visit www.w3schools.com/xforms.) At a minimum, you should use simple data encryption at the host end.
- ✓ **Third-party services:** Consider using a third-party, online credit card processing service, such as PayPal or Authorize.net, which for a minimal transaction fee, can take on the risks of liability and information security that comes with credit card processing.

Understanding how data encryption works

If you don't quite understand what data encryption means, here's a simple analogy: When you submit data on a Web form, the data passes from your browser to the destination server, much like a letter might pass through several post offices en route from your mailbox to the recipient's mailbox. Also much like a letter, it is possible that while it's traveling, someone could open it up and look at it. On the Web, if your data (or letter) is unencrypted, anyone who intercepts it can see exactly what your data is. When your data (or letter) is encrypted, however, if someone opens it up, the text is like a secret code that only the recipient can understand.

Data encryption, then, is a process whereby information submitted to a server from a completed Web form is scrambled during the data transfer by an application or some kind of server-side script written in PHP, CGI, ASP, JSP, ColdFusion, or Perl.

In addition to transmitting the data securely to the site owner, most scripts also include instructions for returning information to the visitor after completing the form, such as forwarding the visitor to a Thank You page (see the nearby sidebar, “Form-forwarding thank-you scripts”), replacing the form with a Thank You message within the same page, and/or sending notification of the completed submission to a specified e-mail recipient, such as the Webmaster.

When using the Thank You page method, you can include as much or as little information on the page as desired. For example, the Thank You page might include a thank-you message, links to popular product categories, and a link back to the site’s home page. Alternatively, the Thank You page may just contain the company logo, navigation, and simple thank-you message, like the example shown in Figure 7-3.



Figure 7-3: Create a Thank You page that visitors can see after submitting a form.

Check with your host provider to see what it offers in the way of form-encryption services, as many different technologies are available. Most host providers have some kind of uncomplicated form-processing services to handle simple form data transmission and encryption with a Perl, ASP, PHP, or CGI script. CGI (which stands for Common Gateway Interface), for instance, is a Web interface that lets Web pages communicate securely with server-side Web applications for both data collection and data feedback, and Perl (Practical Extraction and Reporting Language) is a programming language used for building CGI programs that perform server-side information processing such as encrypting data submitted to a server from a form. Perl scripts can also be configured to output data securely to another source, such as an e-mail address or database.

Typically, the script comes preinstalled with the hosting plan, or the host provider can install it for the domain upon request. Two of the most popular scripts are the `formmail.pl` script created by Matt Wright at www.scriptarchive.com/formmail.html and the enhanced `formmail.pl` script by Brian Evans at www.geocel.com/webMail. To make these form-processing scripts function properly, most must be installed in the `cgi-bin` folder on the server hosting the site. In addition, you must also do a tiny bit of code customization so that the script will forward the visitor to a particular page (`thankyou.html`) in the browser and securely transmit the collected data to a specified e-mail form collection address (`sales@yoursite.com`).

Scripts like these are extremely straightforward to use if you're willing to put in a little time to read any accompanying README file that comes with the script, as well as to perform the necessary testing before publishing the form to site visitors to ensure that the form is processing collected data as desired.

Form-forwarding thank-you scripts

When visitors submit a form online, most of them like to see some kind of acknowledgment of their submission, such as a sentence that appears automatically on the page that says something like "We received your submission" or having the browser automatically transfer them to a special "Thank You page" that tells them more about how their submission will be processed.

In most cases, creating and configuring a special forwarding Thank You page is as simple as creating a Web page, adding a hidden field or two to the form, and inserting the URL of the forwarding page in a specific location inside the script that encrypts and processes the collected form data.

A good Thank You page looks structurally and graphically like the rest of the pages on the site, including navigation buttons and branding. It also contains text that acknowledges the visitor's submission. How that particular text will be worded is completely up to the site owner. The rest of the space on the Thank You page can include other information that might be helpful to the visitor,

such as shipping policy information for a retail purchase, store hours, sale information, additional products the consumer might be interested in purchasing, or other special news that the visitor might not otherwise be aware of.

Because every script or program is different, your particular site and hosting account may require a different configuration. That said, most forwarding options are fairly similar, and the following instructions can tell you generally what to do and where to do it on your host account:

1. Create the Thank You page based on the existing look and feel of the site. Save the thank-you file with a name like `thankyou.html` so that it will be easy to remember.
2. Add a redirecting hidden form field at the top of the form in the HTML code, directly following the opening `<form>` tag.

The particular script you're using might instruct you to add recipient and subject hidden fields to the form, too:

```
<form action="cgi-bin/email.pl" method="post" name="MyForm" id="MyForm">
<input type="hidden" name="redirect" value="http://www.mysite.com/thankyou.html">
<input type="hidden" name="recipient" value="info@mysite.com">
<input type="hidden" name="subject" value="Company Info Request">
```

3. Specify the SMTP server, domain name, recipient e-mail address, and forwarding URL inside the data processing script file.

Good scripts include comment tags that clearly indicate what needs to be customized in the script and exactly where to make those changes.

The SMTP server is your domain or the domain of the host provider, such as `$smtp_server = "smtp.hostdomainname.com";`.

The recipient address should be the address at the Web site that will receive notification of a form submission, such as `@recipient_addresses = ('info@mydomain.com');`.

The recipient domain name is the address of the Web site that will be submitting the form data, such as `@recipient_domains = ('mydomain.com');`.

4. Upload the customized script to the domain host's cgi-bin folder (which the host provider should have already installed on the domain), and upload the form and Thank You pages to the root level of the live server for testing.

You must test the form and script in a live server environment to ensure that everything is functioning properly. To assist with the testing process, you can modify the form filename, SMTP server (Simple Mail Transfer Protocol is the protocol used on the Internet to send e-mail messages), and recipient e-mail address and then change that information to the correct names when the form is ready for publishing. For example, you could use your own e-mail address while configuring and testing the form and then change the e-mail address to the client's address right before the site gets published.

Correct any errors that you find. Many times you will not know about an error or processing issue until the form is published. When the form is error free, you may begin collecting data from visitors.

Understanding the Structure of Web Forms

Building forms on your Web pages requires only a small number of HTML tags. Because each tag is fully customizable, however, you can use them to create an infinite number of form layouts. The main HTML form tags are `<form>`, `<input>`, `<select>`, `<textarea>`, `<label>`, `<fieldset>`, and `<legend>`. With the addition of specific tag attributes, such as `type`, `id`, and `value`, you can customize each of these tags to create all the different types of form fields, such as hidden fields, text fields, text areas, check boxes, radio buttons, lists, menus, and buttons.

Structurally, you can add the HTML form fields to the page in any configuration you need, as long as the entire form is encased in `<form>` tags. The opening `<form>` tag is where you provide processing instructions and other information to the server. Without these tags, the browser and server cannot collect, encrypt, and process data from site visitors.



Establish a great new habit right now by always remembering to build your forms starting with the insertion of a pair of `<form>` tags, along with the `name`, `id`, `method`, and `action` attributes:

```
<form id="form1" name="form1" method="post" action=""></form>
```

In between these `<form>` tags, you may insert a table with any number of rows and columns to house the labels and form fields for the information being collected. For instance, you may need to collect visitors' first names, last names, and e-mail addresses. This would only require a two-column table with four rows: three rows for the labels and form input fields and the fourth row to insert a Submit button to process the collected data, as illustrated in Figure 7-4.

*First Name:	<input type="text"/>
*Last Name:	<input type="text"/>
*Email:	<input type="text"/>
* Required fields	<input type="submit" value="Submit"/>

Figure 7-4: A simple Web form.



While the `<input>`, `<select>`, and `<textarea>` tags insert form elements on the page for straightforward data collection, the last three form elements in the preceding list, `<label>`, `<fieldset>`, and `<legend>`, along with the tag attributes `accesskey` (which assigns keyboard shortcuts to particular form fields) and `tabindex` (which advances a visitor through form fields using the Tab key), are accessibility-enhancement features that you can use in conjunction with other form tags to facilitate assistive devices like screen readers with accessing the information on the form, as in the following code examples:

```
<input type="checkbox" value="checkbox" name="checkbox" id="checkbox"
      accesskey="e" tabindex="5">
<label for="checkbox">Chocolate</label>
<fieldset><legend>The legend assigns a caption to a fieldset, which is used to
      group related items in a form.</legend></fieldset>
```

You use the `<label>` element to help screen readers find control fields like radio buttons and check boxes, the `<fieldset>` element to group sets of related controls, and the `<legend>` element to include a caption with any `<fieldset>`.

Creating a Web Form

To show you how easy it is to build a Web form on your own, the following steps show you how to create a simple Web form in Dreamweaver that is configured to process collected data using a dummy Perl script installed on a mock server. If you're not using Dreamweaver, you can still follow along to see the code elements and general structure of the form-processing component.

In the following sections, you first create the structure of a form and then add text fields, check boxes, menus, radio buttons, and a submit button.

Creating the structure of the form

Follow these steps to create the general structure of the form in Dreamweaver. If you are using another code editor, you can easily adapt the steps by using your editor's form tools and commands to perform the same tasks.

- 1. In Dreamweaver, choose File ⇨ New to open a new blank HTML document in your workspace, and save it with the filename `myform.html`.**

To help keep track of things while you are working, save the HTML file to a new folder on your desktop called `Forms`. Then manage a site to that `Forms` folder before proceeding to Step 2.

To create a managed site, choose Site↪New Site. When the Site Definition dialog box opens, click the Advanced tab. In the Local Info category, enter a name for the site in the Site Name field, and in the Local Root Folder field, browse to and select the Forms folder you just created on your desktop.

2. With your insertion point positioned in Design view at the top of the open HTML page, choose Insert↪Form↪Form to insert a set of <form> tags on the page.

You can also click the Form button (which looks like a little square with a dotted red outline) in the Forms category of the Insert panel.

If the Input Tag Accessibility Attributes dialog box appears, input an ID, a label, an access key (to assign a keyboard shortcut to any element on the page), and a tab index number (to set the tab order of that tag relative to other form fields or objects on the page).

In Design view, the <form> tags create a bounding area with a 1-pixel, dashed, red, rectangular border, like the one shown in Figure 7-5. Within this border, you can add *form objects*, such as text fields, check boxes, radio buttons, lists, menus, and buttons. The line is meant to help you visually define the bounds of the form data within the Dreamweaver workspace, and it is not visible in a browser window. If the line doesn't appear for you in Design view, you can enable this feature by choosing View↪Visual Aids↪Invisible Elements.

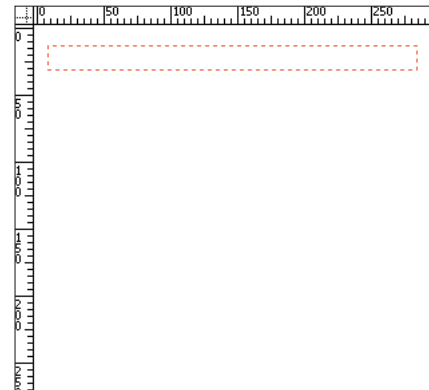


Figure 7-5: Dreamweaver marks the boundaries of a form with a dotted red line.

In Code view, the inserted <form> tags automatically contain several tag attributes, including the `id` and `name`, which should be identical to one another and will be used later to assign JavaScript and apply CSS styles to the form:

```
<form id="form1" name="form1" method="post" action=""></form>
```

The code also includes the `method` attribute, which tells the server whether this form will collect or transmit data, and the `action` attribute, which instructs the browser on where to send the collected information.

By default, all newly inserted forms use the *post* method of data transmission, but you may change that to default or *get*, if desired. For this example, leave the method set to *post*, but read the following descriptions of all three methods so that you'll know for future use what each one does:

- *Post*: This method hides collected data during the HTTP request between the visitor's computer and the remote server, but doesn't encrypt it. Therefore, whenever possible, try to use a secure server connection (using SSL) when transmitting private information, or at least use a script or program to encrypt the data from transmission to receipt.
- *Get*: This method appends the URL of the form page with the actual value of the collected information when the data is sent. When the form is gathering simple data, this is usually okay, but this method has some serious drawbacks. The main drawback is that the get method makes the URL bookmarkable, making the page data vulnerable to spybots and hackers. Also, the URL can contain only a maximum of 8,192 characters, which limits the length of form data you can process with this method.
- *Default*: This setting relies on the visitor's browser's default settings to choose the transmission method. Because the default form data transmission method can differ from one browser to another, however, it's much better to choose either post or get instead.

3. Customize your form by adding form-processing details to your `<form>` tag, starting by replacing the default identical form name and id (form1) attributes with a unique form name and id of your choice.

For instance, you could use a custom name/id of `reqinfo`, as in the following code example:

```
<form id="reqinfo" name="reqinfo" method="post" action=""></form>
```

Naming your forms both identifies the particular form on your page so that the processing script can be easily attached to it and helps you more readily identify a particular form when a site includes more than one form.

4. In the Action field in Dreamweaver's Property inspector (or in the code, if you prefer to hand-code your HTML edits), type the filename and location (relative to the root level of the host server) of the script that will process the collected form data.

For instance, if you're using a CGI script stored in a `cgi-bin` folder on your remote host, type in the path to the script, such as `cgi-bin/email.pl`. Your HTML code should now look something like this:

```
<form id="reqinfo" name="reqinfo" method="post" action="cgi-bin/email.pl"></form>
```

Both CGI and Perl scripts must be placed inside the host server's `cgi-bin` folder, which the host provider should have already installed on the server when the hosting package was purchased. If you don't see this folder, contact your host provider.



If you plan to use some other type of form-processing system (such as sending the collected data to an application on the same or on another server), check with your system administrator and/or host provider about how to properly configure the `action` attribute of the `<form>` tag.

5. (Optional) In the Enctype drop-down list in the Property inspector, choose `application/x-www-form-urlencoded` encoding type for the data being sent to the server.

By default, this field is blank. If you'd like to change it to the default type for the post method, select `application/x-www-form-urlencoded`. Or when adding a file-upload field to a form, select the `multipart/form-data` type.

If you're unsure of what to select here, leave the field blank and check with your host provider or system administrator.

6. (Optional) Set the target browser location for any returned data or documents in the Target drop-down list:

- `_blank`: Displays the returned document or data in a new browser window
- `_self`: Displays the returned data in the same window
- `_top`: Uses the current open window even if other windows are open
- `_parent`: Uses the parent window of the current file

Your opening `<form>` tag should now look like the following code:

```
<form action="cgi-bin/email.pl" method="post" enctype="application/x-www-form-urlencoded" name="reqinfo" target="_self" id="reqinfo">
```

Figure 7-6 shows an example of this same data when listed in your Property inspector.

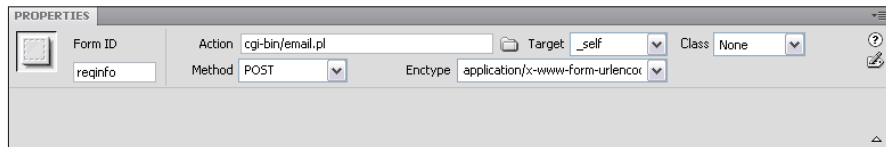


Figure 7-6: View the same data in the Dreamweaver Property inspector.

7. With your insertion point inside the dashed red line of the form in Design view (or in Code view, with your cursor between the opening and closing `<form>` tags), choose **Insert ⇨ **Form** ⇨ **Hidden Field** to add a hidden field to the form.**

See the nearby sidebar for more on hidden form fields.

- 8. Using the Property inspector, change the name/ID of this field from "hiddenField" to "subject" and apply a value attribute with the label of "Info Request".**

```
<form><input type="hidden" name="subject" id="subject" value="Info Request"></form>
```

- 9. Again, with the insertion point between the opening and closing <form> tags directly following the hidden tag you just inserted, choose Insert→Table to insert a seven-row, two-column table with cellpadding=0, cellspacing=4, and border=0.**

The table provides the structure to organize the form fields and labels.

- 10. Along the left side of the table, enter the following text in each of the cells starting from the top: *Name, *Telephone, *Email, I am a, I am interested in (check all that apply), Join Mailing List, and *Required fields.**



Labeling the form before inputting the different form fields can assist you in selecting the right form fields for the data being requested.

Hidden form fields

Hidden fields are special form fields that do not appear anywhere in the browser window but allow you to store and send information on the form along with the information the visitor inputs and submits. Hidden fields are often needed to assist scripts with processing the form data. The name/ID of the hidden form field can be used for things such as Redirect, Recipient, Subject, and Title to apply a sentence, e-mail address, URL, or other information to the form data being collected.

Here are some examples of how you can use hidden fields:

As a redirect to another page after form submission:

```
<input type="hidden" name="redirect" value="http://www.mysite.com/thankyou.html">
```

As an identifier of the e-mail address to receive notice of a form submission:

```
<input type="hidden" name="recipient" value="contact@mysite.com">
```

As an e-mail subject line prescript for the recipient hidden form field:

```
<input type="hidden" name="subject" value="Info Request">
```

As a title for the collected data to be sent along with the data:

```
<input type="hidden" name="title" value="Website Info Request">
```

The only downside to hidden fields is that anyone (including spambots) can view hidden fields by looking at the HTML source code in the browser, so be careful about using hidden fields for private information like e-mail addresses and secret passwords.

Adding individual form fields

Next, you add individual form fields along the right column of the table in the cells next to the labels:

- 1. Add a text field by placing your insertion point in the upper-right table cell and choosing Insert⇨Form⇨Text Field.**

Text fields are used to collect text or numerical data, such as a name, address, telephone number, e-mail address, or password.

If the Input Tag Accessibility Attributes dialog box appears for this or any of the following form fields, input an ID, a label, an access key (to assign a keyboard shortcut to any element on the page), and a tab index number (to set the tab order of that tag relative to other form fields or objects on the page). If you do not want to add any accessibility attributes, click the Cancel button to close the dialog box and insert the form field without accessibility attributes.

- 2. Select the newly inserted text field and use the Property inspector to change the `name` attribute of the `TextField` from "textfield" to "Name".**

When you give each text field a unique name, or ID, you are identifying the input that will be collected. Names can contain numbers and letters as well as the underscore character but can't include any spaces or special characters.

When you modify the default text field name to something else, Dreamweaver automatically inserts both the `name` and `id` attributes into the tag with the new name provided. Both fields are inserted so that CSS and JavaScript can be easily applied to the field. In addition, the text field name helps to identify the collected data.

Text fields can be set to be single line, multiline, or password fields:

- *Single line* uses the `<input>` tag with the `type=text` attribute.
- *Multiline* creates multiline text input fields and uses the same HTML code as a text area form field. Multiline fields use the `<textarea>` tag with the `cols` attribute for character width and the `rows` attribute for number of lines. Enter the number of lines desired in the Property inspector.
- *Password*, which uses the `<input>` tag with the `type=password` attribute, makes asterisks or bullets appear when typing inside the form field in a browser. The data, however, is not encrypted; you must use some kind of data encryption with the form to secure the data.

Figure 7-7 shows an example of single line, multiline, and password text fields.

Figure 7-7: Text fields can be configured as single line, multiline, or password fields.



In addition, you can set the text field to display a word or phrase when the page loads in the browser by entering text into the Init Val box. This text can then be replaced with information from the visitor.

- 3. In the next two table cells down, insert two additional text fields and use the Property inspector to name the new text fields after the label in the table next to them. Name the text field next to Telephone "Telephone" and the text field next to Email "Email".**

To fix the width of the form field to a precise size, you could select the form field and add a number in pixels, such as 23, to the Char width field in the Property inspector. However, controlling the width of any input field with CSS is a better option because different browsers interpret this HTML attribute in different ways. To apply CSS style to a form field through the Property inspector, select the desired custom CSS style from the Class drop-down menu.



If desired, enter a number in the Max Chars field to set the maximum number of characters allowable for the specified form field. This is especially useful for limiting phone numbers to ten digits, zip codes to five digits, or other data that requires a limited number of character input. If a visitor enters more characters than defined by this field, the browser tells the visitor's computer to make an alert sound.

- 4. Next, you want to add a menu. With your insertion point in the empty cell to the right of I am a, choose Insert⇨Form⇨List/Menu.**

This inserts a blank List/Menu field, which can display on the form as either a list or a menu, depending on which type you select in the Property inspector. Use this form field to provide a list or menu to visitors that allows users to make a selection from a set of items, such as a state or country.

For this example, leave the type set to Menu, and next you'll enter a series of options that the visitors can choose from and select.

5. **Select the new list/menu form field in Design view and click the List Values button in the Property inspector to open the List Values dialog box. Enter the labels for each item in the menu along with values for those items. Click OK when you're done.**

The labels represent the options that appear in the menus, whereas the values identify those options with another number, word, abbreviation, or code, such as 01, 02, 03 or AR, AL, or AK.

To add more items and values to the menu, click the plus (+) button. To delete an item, select it and click the minus (-) button. You may also reposition items relative to one another by selecting an item and clicking the up or down arrows.

As shown in Figure 7-8, the first item in the menu can be instructional. For instance, by entering the words **Select one**, you are informing visitors on how to use the menu field on the form. With instruction fields, you may enter a value or leave the value field blank (`value=""`). The rest of the list items in the menu correspond to options the visitor can select.

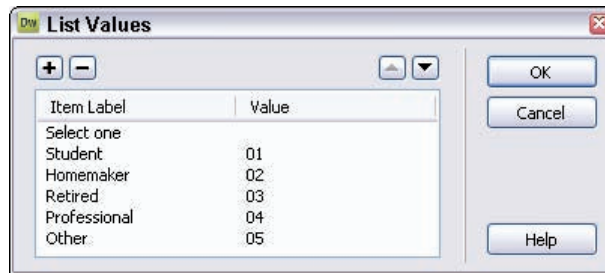


Figure 7-8: Use the List Values dialog box to input items that will appear in the List/Menu form field.

Next, you add check boxes to the form that allow visitors to select as many options as desired within a particular category. To help keep the information neat and organized, insert a nested table.

6. **To the right of the cell containing I am interested in, insert a table (choose Insert→Table) with two rows and two columns.**

You'll be entering a set of check boxes into each of the nested table cells.

7. With your cursor inside the upper-left cell of the new, nested table, choose **Insert**→**Form**→**Checkbox**.

This should open the Input Tag Accessibility Attributes dialog box, as shown in Figure 7-9. Do not close this dialog box; you will enter specific data inside of it in the next step.

Check box fields allow users to specify multiple responses when presented with a single question, such as the responses to the statement I enjoy learning about:. You can add as many check boxes to the form as you want to support the question being asked.

8. Enter the word **Interests** in the **ID** field and the words **Web Design** in the **Label** field, select **Wrap with Label Tag** for the style, choose **After Form Item** as the position, and if desired, enter an access key and tab index number. When finished, click the **OK** button.

This inserts a check box with a text label of `Web Design` directly following it, along with the `id` and `name` attributes set to `"Interests"`:

```
<label>
<input type="checkbox" name="Interests"
      id="Interests">
Web Design</label>
```



Figure 7-9: Insert check boxes with labels in the Input Tag Accessibility Attributes dialog box.

9. Select the check box and use the Property inspector to set the Checked value to "Web":

```
<label>
<input type="checkbox" name="Interests" id="Interests"
value="Web">
Web Design</label>
```

The checked value helps to identify the collected data with the visitor's specific selection. In this case, the value of "Web" indicates the visitor is specifically interested in Web Design.

10. Repeat Step 5 to insert check boxes inside each of the remaining three cells in the nested table. Leave the name set to "Interests" for all of them, but set the individual Label and Values to "Print", "Illustration", and "Other", respectively.

Giving all the check boxes the same name of "Interests" instructs the browser that all the check boxes belong to the same group.

For visitors who select the Other option, insert a text field next to the word "Other" with the name/id attribute of "Interests".



If desired, set the initial state of one or more of the check boxes to be selected. For instance, if you'd like to prompt visitors to join a mailing list, one of the check boxes can have that option preselected. Visitors can then select and deselect the check boxes before submitting the form.

11. Add a radio button by placing your cursor in the cell to the right of Join Mailing List and choosing Insert→Form→Radio Group.

Radio button fields allow users to specify "either/or" choices when presented with a question. You can have as many radio buttons as you want for any question, but the user can only select one answer.

12. When the Radio Group dialog box appears, change the name of the group from RadioGroup1 to MailingList. Then, in the Label/Value area of the dialog box, enter "Yes", "yes" and "No", "no" as the label/value pairs. Leave the Lay Out Using option set to Line Breaks and click the OK button.

Your code for the radio button group should look something like this:

```
<label>
<input type="radio" name="MailingList" value="yes"
id="MailingList_0">
Yes</label>
<label>
<input type="radio" name="MailingList" value="no"
id="MailingList_1">
No</label>
```

In a radio group, both buttons use the same name so that a visitor can only make an either/or selection as defined by the different value attributes, yes or no.

If desired, you may also set the initial state of the Yes button to checked using the Property inspector, which adds the following initial state code to the input field:

```
<input type="radio" name="MailingList" value="yes"
      id="MailingList_0" checked="checked">
```

In form fields that have a default selection enabled, that selection will remain in effect unless the visitor notices and changes it.

- In the last cell in the lower-right corner of the table, insert a Submit button by choosing Insert→Form→Button.**

Submit buttons are the default button type for forms.

- If you want to change the label that displays on the button, modify the Value field from "Submit" to anything else, such as "Send".**

Et voilà, your form is now complete! Figure 7-10 shows the finished example form in Dreamweaver's Design view.

Figure 7-10: Preview your form in a browser to ensure that you like the way it looks, and be sure to upload the form to a server for testing before publishing it on the Web.

When you are finished entering fields for your form, save the file and preview it in a browser to see how each of the form fields functions. If you like what you see, you can then style the form with CSS, add a validation script, test it, and publish it.

Using a graphic instead of a button

Forms need not look boring! You can spruce them up with CSS and even use your own graphics instead of the default form buttons by using the Image Field option. For instance, you might want to replace the regular old Submit button with a graphic button that says "SIGN ME UP!" When you replace the Submit button with an image, the image buttons become clickable by default, unless you apply a different JavaScript to the image.

To add an image field to your form, follow these steps:

1. Place your cursor inside the form area on your page where you want to insert an image field.
2. Choose Insert→Form→Image Field.

You can also click the Image Field button in the Forms category of the Insert panel.

3. When the Select Image Source dialog box appears, browse to and select the image you want to add to the form.

If you enabled accessibility features, the Input Tag Accessibility Attributes dialog box opens. Complete the dialog box and click the OK button to insert the image or click the Cancel button to insert the field without accessibility attributes.

Your code for a graphic form button should look similar to the following:

```
<input type="image" name="signup"
      id="signup" src="images/signup.gif"
      />
```

If you'd like to make your button perform another function, apply the desired JavaScript behavior to it. For example, your button could run a script that launches a pop-up browser window with technical specifications about a particular product.

Validating Web Forms

A Web form by itself is pretty good, but it doesn't do everything it could to help the visitor in filling out the form fields and submitting the correct information accurately the first time. By contrast, a self-validating Web form can alert visitors when they've missed information or have entered their data in the wrong format.

In the following sections, you find out what a validating form is and how to add a validating form behavior to an existing form.

Understanding what a validating form is

A *validating form* refers to a form that includes JavaScript that can verify whether the visitor has correctly completed all required fields on the form before the data gets transmitted over the Web. That way, should a visitor submit incorrect or incomplete information, an error message appears

either somewhere on the page or in a special pop-up alert message window above the open browser, identifying the problem or omission. These error messages help the visitor locate and fix the problem before resubmitting the form.

On *dynamic* Web sites (sites that use programming to dynamically display content from a database), validation is typically done with a programming language such as PHP, ASP, JSP, or ColdFusion. However, for small, non-dynamic sites, more often than not form validation is accomplished with JavaScript.

If you're using Dreamweaver, you can automate the task of adding a JavaScript validation script to any form by using the special Validate Form behavior in the Behaviors panel.

After inserting the script, you can customize it to validate the form fields to help make sure that the user fills in all the fields correctly. The validation script is made up of a series of validation events that you can add to some of or all the fields on the Web form. By selectively configuring each form field, you can choose whether an error/alert message appears to the visitor either as the visitor completes individual fields (so that he can correct mistakes as he goes from form field to form field) or after the visitor clicks the Submit button (so that he can correct all his mistakes at once).

Regardless of which option you select, the validation is performed on the client side, before the form is submitted to the server to ensure that the server won't collect any data until the form passes validation.

Of course, non-Dreamweaver users can hand-code validation scripts into any form, but that can be a lengthy process that requires some knowledge of JavaScript, especially if you intend to configure a validation rule for each field in the form. The easier and faster method of validation is the aforementioned Dreamweaver validation behavior, which thankfully doesn't presuppose any knowledge of JavaScript.

Adding a Validate Form behavior to a form

Follow these steps to apply the Validate Form behavior to an existing form on a Web page:

- 1. Open your Web page that contains the form inside the Dreamweaver workspace.**

Your form should include several fields, including a place for the visitor's name, phone, and e-mail address. If desired, feel free to use the `myform.html` page you created earlier in the chapter.

2. To validate the entire form, select the opening `<form>` tag. Otherwise, to validate individual form fields, select the first form field that needs validation.

To select the opening `<form>` tag, click the upper-left corner of the form's border in Design view. You can tell that the tag is selected if the Property inspector displays the form properties.

3. In the Behaviors panel (choose **Window**→**Behaviors**), click the **Add Behavior plus (+)** button and select the **Validate Form** option from the resulting drop-down menu.

The Validate Form option is located near the bottom of the menu. Upon selecting this behavior, a Validate Form dialog box like the one shown in Figure 7-11 appears, listing all the named form fields in the form.

Any form fields that you forgot to provide customized names to through the Property inspector will be listed with a default form field name, such as `input "textfield"`. If you see any fields that are unnamed, click the dialog box's Cancel button, update the form fields with custom names, and reselect the Validate Form option from the Behaviors panel.

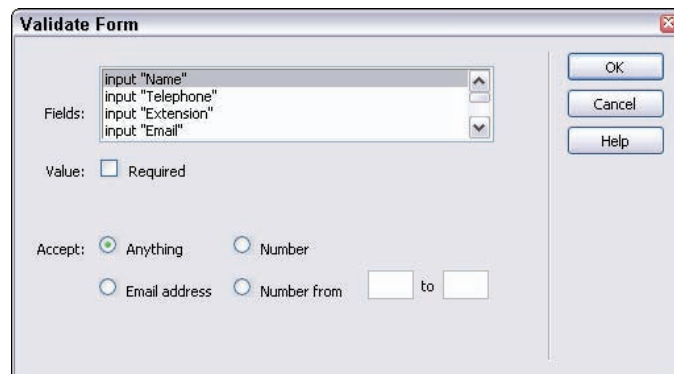


Figure 7-11: Dreamweaver's Validate Form behavior lets you apply an easy-to-use JavaScript validation script to your form.

4. According to how you'd like to validate form fields, select one of the following options:

When validating a single form field:

From the Fields list, select the form field by its label, such as `input "Name"`, and configure the rest of the dialog box as desired. For example, make that field required or specify that only a numeric value or e-mail address may be entered in the field.

When validating the entire form:

One by one, select each form field from the Fields listing and specify the Value and Accept fields:

- *Fields:* All the fields on the form are listed here. Select each form by name and assign validation preferences using the rest of the dialog box options.
- *Value:* Select the Required check box when the field selected in the Fields list must be completed by the visitor.
- *Accept Anything:* When this option is selected, the field in the Fields list will accept any type of input.
- *Accept Number:* With this option selected, the field in the Fields list will accept only a numeric value.
- *Accept Email Address:* Choose this option to have the validation script check for proper e-mail address syntax. This part of the script cannot verify that the entered address is valid, but it can check that it meets the `name@sitename.extension` e-mail format.
- *Accept Number From/To:* Select this option to allow visitors to input a range of numbers predetermined by you, such as any number between 1 and 5.

5. Repeat Step 4 for each form field in the Fields listing. When finished, click the OK button.

Whether you chose to validate individual form fields or all the fields in the entire form, Dreamweaver automatically adds the appropriate JavaScript validation code to the `<head>` of the page, along with additional script, either in-line with the form field or within the opening `<form>` tag at the top of the form.

When validating single fields, Dreamweaver uses the `onblur` or `onchange` validation event in-line with the code so that visitors can see any error message as they input data.

```
<input name="Name" type="text" id="Name"
  onblur="MM_validateForm('Name','','R');return
  document.MM_returnValue">
```

By contrast, when validating the entire form, the `onsubmit` validation event handler is used, and any error messages appear to the visitor after he or she clicks the Submit button.

```
<form action="cgi-bin/email.pl" method="post" name="myform" id="myform"
onsubmit="MM_validateForm('Name','','R','Telephone','','NisNum',
'Extension','','NisNum','Email','','RisEmail');return document.MM_
returnValue">
```



To test the validation, launch the page in a browser and try entering incorrect data in each of the form fields before submitting the form. Because the validation script processes data on the client side (not the server side), the form does not need to be uploaded to a server to be tested. It will, however, need to be on a working server to test the form-processing script.

Building Spry Web Forms in Dreamweaver

Spry validation is the newer, cooler way to validate your forms in Dreamweaver because it uses the more advanced tools in the Spry framework. Using HTML, CSS, and JavaScript, the Spry framework helps designers build XML-rich Web pages that provide more interactive experiences for site visitors. If you want interactive experiences for your visitors, use Spry forms instead of regular form fields.

Taking a look at the Spry validation widgets

All the Spry validation form fields, or *widgets*, mirror the regular form fields. The Spry widgets for forms include the Text Field, Textarea, Checkbox, Select (Menu), Password, Confirm, and Radio Group widgets. To use any of these widgets, just insert the Spry fields into your form instead of inserting the regular HTML form field tags.



After the Spry validation widgets are inserted in your form, you may customize how each field validates data, as well as modify the look and feel of how the field appears in the browser using CSS. In fact, when you insert even one Spry widget onto your page, Dreamweaver automatically adds a special SpryAssets folder to your managed site that includes an external CSS file (`SpryAssets/SpryValidationTextField.css`) to help you further customize the look of your Spry validation widgets. In addition to this CSS file, Dreamweaver also adds a JavaScript file (`SpryAssets/SpryValidationTextField.js`) to the SpryAssets folder, which helps to manage the interactive dynamic features of the Spry validation widgets.



You must upload the entire `SpryAssets` folder to the server, along with the file(s) that contain the Spry forms, for the Spry validation widgets to function properly.

All the Spry validation widgets can be added to your page using the `Insert`→`Spry` menu or by clicking one of the Spry Validation buttons in the Spry category of the Insert panel, as shown in Figure 7-12.

The main difference between the Spry widgets is how they are styled with CSS and the kind of interactive opportunities they offer to site visitors. For example, with the Spry Validation Text field, visitors can be prompted to enter the correct telephone number format. They see an error message (“Invalid format” or “A value is required”) when the format entered does not match the formatting hint (800-555-1212) you provided or when no input is made to the field.

Adding Spry validation fields to a form

To add Spry validation fields to your form, follow these steps:

1. **With your cursor inside the area in your form where you would like to insert a Spry Validation field, click the desired Spry Validation field button in the Spry category of the Insert panel.**

You may also insert Spry validation widgets by choosing `Insert`→`Spry` and choosing the desired option from the submenu that appears.

If, after selecting your Spry field, the Input Tag Accessibility Attributes dialog box opens, complete the fields and click the OK button to add the Spry field to your page.

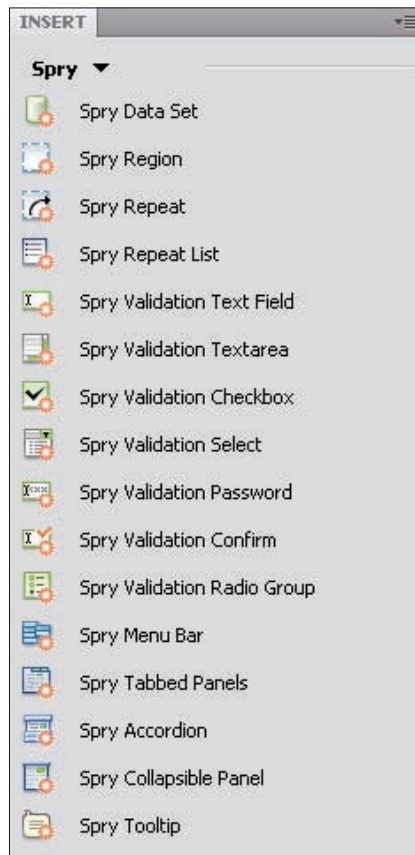


Figure 7-12: Add Spry validation fields to your form through the Spry category of the Insert panel.

Spry Validation form fields look just like regular HTML form fields, only when selected in Design view, they are surrounded by a blue tab and outline that identify the form field as a Spry Validation form field, like the one shown in Figure 7-13.



Figure 7-13: In Dreamweaver, selected Spry Validation fields display with a blue outline and tab in Design view.

2. Select the new Spry Validation field in Design view by clicking the field, and then customize the field in the Property inspector.

Set any desired parameters for the Spry field, as you would for a regular form field.

3. Select the Validation field's tab in Design view to set the validation type and format in the Property inspector.

When you click the Validation field's tab, a different set of information appears in the Property inspector. You need to select the right type and format to match the data you intend to collect from site visitors, such as a phone number, currency, date, or credit card.



For a full listing of all the Spry types and formats, visit the following Web site:

http://help.adobe.com/en_US/Dreamweaver/10.0_Using/WS455A2B6C-8E96-4879-8195-4B47394B9BA3a.html

4. In the Validate On field, select an option to set when the validation event will occur.

Choose from Blur, Change, and/or Submit. You can choose as many as you like, or disable them all.

- *Blur*: The field validates when the user clicks outside the field.
- *Change*: The field validates while the user enters text inside the field.
- *Submit*: The field validates when the user submits the form.

5. (Optional) Continue using the Property inspector for the selected Spry widget to set other options as required by the Validation field.

For example, when you select the Spry Validation Text Field tab, other settings for this widget include setting the minimum and maximum number of characters, displaying a character counter and/or preview states, changing the required status of a field, creating a customized form field text hint, and blocking extra characters.

After setting the Spry widget's properties in the Property inspector, the field's style and error message (if any) can be further customized in the CSS. Finding the right field, however, can sometimes be tough, so be sure to consult the online help files to ensure that you're editing the correct CSS style.



For more tips about working with Spry and styling your Spry menu with CSS, visit www.adobe.com/go/learn_dw_spryselect_custom.

Testing Validated Web Forms

Whether you're using regular or Spry validation form fields in your form, the next step before making the form available to the public is to test the form for accuracy in validation and processing. In most instances, the validation script will not affect how the form gets processed, but rather it will help to determine whether visitor input is properly entered and formatted before the collected data is submitted to the server for actual processing.



For your testing, you will get the best results when you upload the file that contains the form to the root level of the intended host or test server. This is especially critical when the encryption of the collected form data will be performed by a Perl or CGI script specified in a particular directory (`cgi-bin\filename`) on the host server. If you do not perform testing in the intended live environment, you might encounter issues later after the form gets published. To avoid that scenario, test in a live environment so that you can be 100 percent certain that your form passes validation and encryption.



To help keep the form away from potential visitors during the testing phase, change the filename on the page to something else until you're ready to officially publish it. For example, if the form page will be ultimately be called `contact.html` when you publish it, consider changing the filename to `testcontact.html` during the testing phase. You can rename the file to `contact.html` when the page is ready for visitors to use.

To test the accuracy of your form's validation script and data processing capabilities, follow these steps:

- 1. Upload the form (via FTP or another access method) to a testing server or to the live server using a testing filename.**

The testing server can either be the server hosting the site or another server that mimics the hosting environment of the hosting server. For simplicity's sake, if you are unsure which route to take, test the form directly on the hosting server to ensure that the form will work without error when the site is published. While this does mean that the form

page will need to be uploaded to the host server for testing, no one should be able to access it during the testing unless they happen to know the direct URL to that test page.

To find out more about File Transfer Protocol (FTP) and how to transfer files to a remote server, turn to Book V, Chapter 2.

2. To ensure that the form can handle correct input, enter incorrectly formatted test data in each field on your form and click the Submit button.

After the collected data has been transmitted over the Web, verify that the data is being sent to the proper recipient e-mail address (which can be your own, for testing purposes) when using an e-mail recipient, or to a database at the Web site collecting the data. Also check to see that any return information to the visitor is functioning (such as a thank-you message within the page or a forwarding script that sends visitors to a Thank You page) and that all other parts of the script are working as intended. For example, your script may automatically send visitors a confirmation e-mail upon submitting the form. Check everything you can possibly check.

Should you encounter any issues processing the collected data after uploading the form to a test server, the trouble will most likely be with the script or programming code being used to process the data. Reread any configuration files that came with the script and continue testing until you get the form to work. If you still continue to have trouble with the form, contact the host provider or system administrator for assistance. Often the script doesn't function properly because a part of the script was not configured properly or some kind of software still needs to be installed or enabled on the host end. If, after enlisting the help of your host provider or system administrator, you still cannot fix the problem, you may need to hire a programmer to assist you.

3. To see how your form handles bad input, enter incorrect data in each form field and click the Submit button.

Be sure to test for every scenario you can think of (letters in number fields, too many characters, not enough characters, wrong input, missing input, incorrectly formatted e-mail addresses, and so on) and make any changes to the form fields, validation widgets, and validation script as needed to correct potential coding errors before publishing the page.

For example, with the Spry validation widgets, visitors can be prompted with screen hints to fill in form fields accurately, but if you've misspelled a word in the hint or the hint doesn't make sense, now is the time to fix it. Figure 7-14 illustrates how visitors might be prompted with screen hints in a Spry form when entering incorrect or incomplete data.

To make adjustments to an existing regular form-validation script in Dreamweaver, select the form by its opening `<form>` tag and reopen the Validation Form behavior dialog box by double-clicking the yellow spoke icon for the Validate Form behavior in the Behaviors panel.

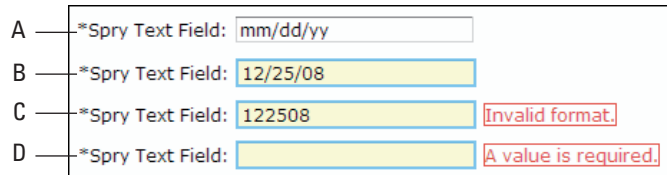


Figure 7-14: Visitors are prompted to enter data correctly on a Spry form. A. Hint activated, B. Correctly entered data, C. Incorrect data entered, D. Missing data in required field.



Whatever you do, do not publish a nonworking form. It is far better to allow visitors to send a simple e-mail to a site than to display a broken form. Take the time to work out the kinks; then publish the form.



To get the most out of the Spry Validation form fields, you should have a firm understanding of HTML, CSS, and JavaScript. For further assistance with Spry, read Adobe's help files about building Spry pages visually at the following Web site:

http://help.adobe.com/en_US/Dreamweaver/10.0_Using/WS2442184F-3DF4-4240-96AF-CC6D792E2A62a.html.

Chapter 8: Making Your Pages Interactive

In This Chapter

- ✓ Adding interactivity with JavaScript
- ✓ Creating customized rollover effects
- ✓ Building image maps
- ✓ Creating a slide show
- ✓ Inserting multimedia such as Flash, QuickTime, and MP3s
- ✓ Adding content that changes daily

Hands down, one of the most interesting aspects of using the Internet is the fact that visitors can interact in a variety of ways with Web sites. Many computer users claim, in fact, that they like to spend several hours each day surfing the Internet to listen to music, watch movies, chat with friends, play games, read the news, look up the local weather, participate in forums, browse for products on eBay, research school topics, and more.

There are many different ways to make Web sites interactive, from the very simple (such as rollover buttons) to the very complex (such as Flash components, QuickTime videos, and interactive games). For simple functions, you can easily find free JavaScript code online to add to your pages without even knowing anything about JavaScript. For more complex interactive site features, you need both the multimedia components and their attending browser extensions or plug-ins (many of which are free), and you must find out how to properly insert them onto your pages.

In this chapter, you are introduced to ways of using JavaScript to make your Web sites more interactive — such as creating multi-image rollover effects, building complex image maps, and opening new browser windows. You also discover how easy it is to insert multimedia files on a page. Finally, I show you how to add a daily tip on your site on the topic of your choice and a free JavaScript puzzle or game to entice visitors to return to the site daily.

Free Seminars Job Board Workbooks

Register Now! Get a Bro

Dw Fl Fw ID Ps Ai
Dreamweaver Flash Fireworks InDesign Photoshop Illustrate

Exceptional Computer Graphics Training in New York City

Your Instructors...

Noble Desktop, LLC
594 Broadway, Suite 1202, New York, NY 10012 212.226.4149
Send all inquiries to educator-in-chief@nobledesktop.com.

...ing courses in desktop public...

Getting to Know JavaScript

JavaScript is one of the quickest and easiest ways to add pizzazz to your Web pages by turning static, regular Web pages into dynamic, interactive, interesting — and sometimes extraordinary — Web pages.

Without even knowing how JavaScript works, you can use free scripts on your site to do things like making rollover buttons, launching new browser windows, turning the browser's status bar into a message ticker, displaying the current date and time on the site, creating simple interactive games, and turning static images into slide shows. Other scripts can be used to generate browser cookies and build dynamic navigation bars, create special effects with images and sounds, add computer utilities and perform math functions, apply password protection to a page, validate forms and e-mails, and even play interesting background and cursor DHTML animations within the browser window. Figure 8-1 shows an example of a script from www.javascriptkit.com that gives your cursor a trailing cursor effect when a visitor moves his or her mouse around in the browser.

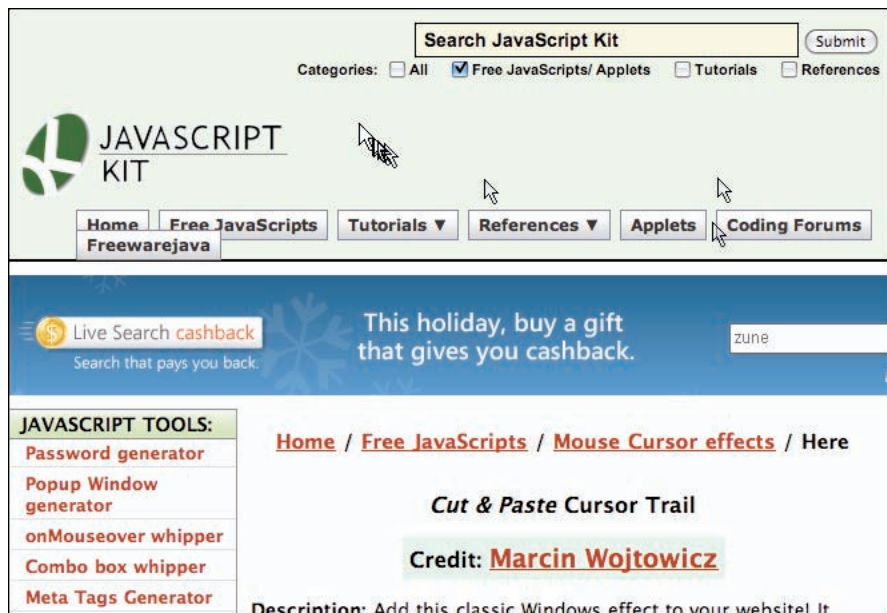


Figure 8-1: JavaScript can enhance a visitor's experience on a Web site.

Table 8-1**Free JavaScript Resources**

<i>JavaScript Resource</i>	<i>JavaScript Resource Web Address</i>
W3Schools	www.w3schools.com/js/default.asp
DynamicDrive	www.dynamicdrive.com
EarthWeb JavaScripts	http://webdeveloper.earthweb.com/webjs
Internet.com's JavaScript	www.javascript.com
IRT.org	www.irt.org/articles/script.htm
JavaScript Kit	www.javascriptkit.com/cutpastejava.shtml
JavaScript Learning Center	www.javascriptmall.com/learn/contents.htm
HTML Source	www.yourhtmlsource.com/javascript
Quirksmode	www.quirksmode.org/js/intro.html
JavaScript Source	http://javascript.internet.com

Keep in mind, however, that not everyone can experience the coolness of JavaScript. Roughly 5 percent of all Internet traffic comes from visitors who use assistive devices that don't understand JavaScript and from visitors who have purposefully disabled the JavaScript in their browsers. These visitors simply can't experience JavaScript's neat effects or, as is sometimes the case, even access the information that the JavaScript reveals. For those visitors, you must always try to provide alternate access to the same content through different techniques, such as placing relevant content inside a pair of `<noscript>` tags or making plain-text content available on another page. See Book IV, Chapter 2 for more on accessibility improvements.

For the other 95 percent of visitors who do have JavaScript-enabled browsers, take advantage of this fact and use JavaScript to make your site more robust, appealing, exciting, and easy to use.

If you're using Dreamweaver, you automatically have access to about 22 of the more common JavaScript effects through the Behaviors panel. For instance, you'll find Swap Image and Validate Form behaviors, discussed in previous chapters, plus other cool scripts that open new browser windows, show pop-up messages, insert jump menus, and more!

In addition to Dreamweaver's scripts, you can find tons of free scripts online by doing a search for "free JavaScript" in your favorite search engine. Table 8-1 lists the most popular JavaScript resources, though the longer you search, the more you'll find, especially if you refine your search with keywords when looking for specific scripts, such as "free JavaScript menus" or "free JavaScript image gallery." In exchange for the free scripts, often the only thing you need to do is leave the script author's contact information and script comment tags inside the script on your page. For example, the following comments appear in Kurt Grigg's Comet Trail script at www.dynamicdrive.com/dynamicindex13/comet.htm:

```
//Comment trail script- By Kurt (kurt.grigg@virgin.net)
//Script featured on Dynamic Drive
//Visit http://www.dynamicdrive.com for this script and more
```



Like many Internet technologies, JavaScript can be used for good reasons as well as for really annoying (creating unsolicited pop-up browser windows) and sometimes computer-damaging (installing adware on a visitor's computer) purposes. Some of the bad reasons have actually prompted some browser developers to create pop-up blockers for your computer, which unfortunately can prevent your honest JavaScript windows from being displayed too. To make sure that you're getting your scripts from reputable sources and not from sites with corrupt hidden agendas, stick with the sites listed in Table 8-1.

Because JavaScript is a scripting language and not a programming language, it is relatively easy to learn after you understand its basic syntax and structure. W3Schools has a great, easy-to-follow tutorial at www.w3schools.com/js/default.asp. In addition, you can find many fantastic free online JavaScript tutorials with a quick search for "JavaScript tutorial," some of which can be found at the sites listed in Table 8-1.

So that you can see for yourself just how easy it is to use JavaScript, the next part of this chapter introduces you to creating multiple rollover effects, opening new browser windows, and setting up complex image maps.

Creating Multipart Rollover Effects

In Book III, Chapter 6, you find out about creating navigation systems in which you insert simple rollover buttons through Dreamweaver's Insert menu. The rollover button uses JavaScript and two same-size graphics, one for the button's normal state and the other for the button's over state. When viewed in a browser, the normal graphic appears until a visitor moves the cursor over it. At that moment, the rollover state graphic appears in its place, and when the visitor moves the cursor off the button, the original graphic returns to view. This kind of rollover button is super-easy to set up, providing an effective way of adding a dynamic vibe to a site.

While the rollover button is admittedly great, it's not the only way to use JavaScript and graphics to achieve dynamic effects with images on your page. Believe it or not, you can actually make multiple parts of a Web page change in response to a single visitor action, all with the magic of JavaScript! For instance, you might change the color of the layout, play a GIF animation, or display a special slogan or advertisement on the page or in the browser's status bar.

With a little bit more code than the simple rollover button — and a few more graphics for the new rollover states of all the graphics that will be swapped — you can make a single mouseover movement control several image or page property changes at once! What's more, if any of those images are animated GIFs, the animation can be played for either or both the normal and rollover graphic states.

To create a multipart rollover effect in Dreamweaver, you need at least four to eight graphics, preferably one pair of graphics for a rollover button, and a couple more that can be used to replace existing graphics on your page during the mouseover part of the multipart rollover. For each multipart rollover effect you conceive of, you will need to create and optimize all the necessary graphics yourself. Also, keep in mind that each rollover requires two graphics, one for the normal state and one for the over state.



If you'd like, you may create and use your own graphics with the following steps. Otherwise, download and use the set of graphics and HTML files in the Rollovers demo file at www.dummies.com/go/webdesignaio. Non-Dreamweaver users can examine the JavaScript and HTML code in the completed sample file that comes with the free download.

To create a multiple graphic rollover effect using Dreamweaver, follow these steps:

1. **Download the Rollovers file from** www.dummies.com/go/webdesignaio **and save it to a folder on your desktop called Rollovers.**

The Rollovers file contains a folder that includes two HTML files — one you can use for this step list and another that shows the completed file (in case you want to peek at the results before starting) — along with an `images` folder full of graphics.

2. **Create a managed site in the new Rollovers folder.**

Choose Site→New Site to create a managed site. Click the Advanced tab at the top of the Site Definition dialog box, and in the Local Info category, enter a name for the site (such as Rollovers) in the Site Name field. In the Local Root Folder field, browse to and select the new `Rollovers` folder. The name of the site helps you identify the project and does not appear anywhere on the published site. The local root folder tells Dreamweaver where to find the files for this site to perform special functions like site-wide modifications.

3. **Open the `rollovers.html` file in the Dreamweaver workspace and select the graphic, in Design view, that says "MAC".**

The rollovers.html file consists of a table with five graphics, each of which is sitting inside its own table cell, as illustrated in Figure 8-2. In the images folder, you find additional graphics that can be used to create the multi-image rollover effect.

4. **Open the Dreamweaver Behaviors panel (choose Window⇨Behaviors), click the click the Add Behavior (+) button, and choose Swap Image.**

The Swap Image dialog box opens, listing (by name or id attribute) all images in the open file.

Unfortunately, none of the images (except for the one called Services) have been given identical name and id attributes, either by hand-coding or applying that attribute with the Property inspector. Without names to identify each image, it would be difficult to know which images are the right ones to “call” with the JavaScript and create the desired rollover effect.



Figure 8-2: Rollover graphics can be located anywhere on a page, including inside table cells.

5. **Click the Cancel button to close the Swap Image dialog box.**

For the multipart rollover JavaScript technique to work, you must go back into the HTML code to name all images before applying any behaviors to them.

6. **Select the image at the top of the table that says Computer Superstore, and in the ID field in the upper-left corner of the Property inspector, enter superstore, as shown in Figure 8-3.**

While you’re at it, use the Property inspector to give the graphic some descriptive alt text, such as Computer Superstore, to make the graphic more accessible to visitors.

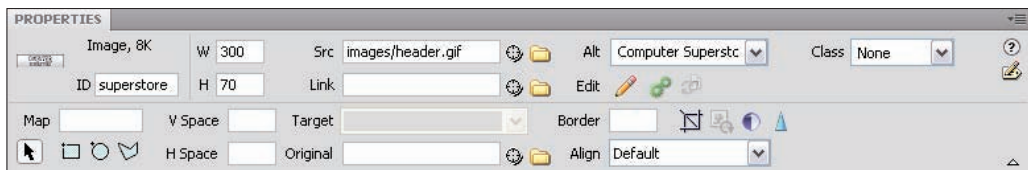


Figure 8-3: Label all the images with the id attribute before applying JavaScript.

7. Repeat Steps 3 through 6 for the rest of the graphics in the HTML file, providing each with an ID and alt text. Name the images dell, sony, and main, and for the alt text, add Dell, Sony, and <empty>.

The <empty> alt text entry in Dreamweaver inserts an empty alt attribute (<alt="">) to the image tag, making the image compliant with accessibility standards.

Now you're ready to add the JavaScript.

8. In Design view, select the MAC graphic, click the Add Behavior (+) button in the Behaviors panel, and choose Swap Image.

The Swap Image dialog box neatly displays each of the images with the names you just gave them with the Property inspector, as shown in Figure 8-4.

Next you'll use this dialog box to set the rollover state for all the graphics that will be changing for this multipart rollover — all of which will be triggered by the visitor with a single mouseover event.

9. With the image "mac" graphic selected in the Images area of the Swap Image dialog box, click the Browse button next to the Set Source To field and choose the graphic called b_mac_over.gif from the Rollover folder's images folder.

After selecting the overstate graphic, Dreamweaver places an asterisk (*) next to the words image "mac" in the Images field. This is your visual indication that the b_mac_over.gif graphic will appear when a visitor moves the cursor over the normal b_mac.gif graphic in a Web browser.

Leave the Preload Images and Restore Images onMouseOut options enabled, because both elements are a necessary part of the JavaScript that Dreamweaver adds to the page.

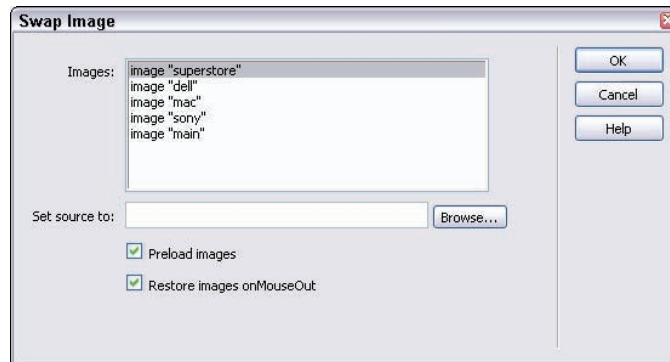


Figure 8-4: Labeled images are neatly listed in the Swap Image dialog box.

10. Click the OK button to close the dialog box and save the Swap Image setting. Then save the HTML page (choose File⇨Save) and preview the page in a browser window (choose File⇨Preview in Browser) to test the rollover effect.

Moving your cursor over the MAC graphic in the browser should make the button's rollover state appear, showing the word MAC with a blue, rounded rectangular outline. Moving your cursor away from the MAC graphic makes the graphic return to its normal state.

11. Back in the open `rollovers.html` file in Dreamweaver, reselect the MAC graphic, and then in the Behaviors panel, double-click the yellow gear icon next to the action called Swap Image.

Clicking the gear icon reopens the Swap Image dialog box for the MAC swap image behavior.

Make sure that you have selected the Swap Image and not the Swap Image Restore icon in the Behaviors panel. If you accidentally try to open the Swap Image Restore behavior, you'll see the Swap Image Restore dialog box instead of the Swap Image dialog box.

Now comes the cool multipart stuff!

12. In the Images area of the open dialog box, select the image called `superstore` and click the Browse button to set the source for that image's rollover state to `new.gif`. Next, select the Choose a Computer image and set its source for the rollover state to `mac.jpg`. Click the OK button.

The JavaScript that Dreamweaver has inserted into the code now has instructions to change three different graphics when a visitor mouses over the single MAC graphic.

However, you're not finished yet. Even though the mouseover effect will work, clicking the MAC graphic will not yet take a visitor to an adjoining Mac page. To do that, you have to also assign a hyperlink behavior to that graphic.

13. To add a hyperlink, select the MAC graphic again in Design view, click the Add Behavior (+) button in the Behaviors panel, and select Go to URL.

In the Go to URL dialog box that opens, you find two areas: the Open In box (that lists the Main window) and the URL field.

Because this page is not part of a larger frameset, only the Main window option is listed in the box. Were the page instead part of a frameset, the individual frame names would be listed here, from which the target destination within the frameset could be selected. Frames is an older HTML technique, not used much anymore, that uses `<frameset>` and `<frame>` tags instead of `<body>` tags to make two or more pages display in a single browser window.



14. In the URL field, type (or browse to) the destination page URL. For demonstration purposes, type `http://www.apple.com` and click the OK button.

When the URL is local (such as `mac.html`), only the filename and extension are needed. However, when the file resides in a subfolder off the root level or is external to the site, type the path and filename (for example, `computers/mac.html`) or the complete URL (for example, `http://www.apple.com/macbookpro`).

15. That's it! Save the file, preview it in a browser, and move your cursor over the MAC button to watch the multipart magic.

You should observe these results, as illustrated in Figure 8-5:

- Upon mouseover, the MAC, Computer Superstore, and Choose a Computer graphics all change — and you see a flashing “NEW” animation within the Computer Superstore graphic.
- Upon mouseoff, all the graphics return to their normal states.
- When you click the MAC link, the page transfers to Apple.com.



Figure 8-5: With multipart rollovers, you can make several graphics change with a single mouseover event.

If, for some reason, your page doesn't function as expected, you might have missed a step or accidentally typed the wrong character somewhere along the way. Go back into the code of your page to see whether you can identify and correct any errors; then retest the page in the

browser. After that, if you're still having trouble, take a look at the `rollovers_complete.html` file. To isolate the problem in your code, compare your page side by side with the code in the `rollovers_complete.html` file. Look carefully, because the problem may be as tiny as the omission of a period (.) or a quotation mark (").

Feel free to use this multipart graphic rollover technique on any series of graphics within the same page. This is a fantastic way to showcase your creativity while enhancing the visitor's experience on the site.

Launching a New Browser Window

On most Web sites, the visitor is taken from page to page within the same browser window. By contrast, some sites open additional browser windows for a variety of different reasons.

The following sections give you some general guidelines for opening new browser windows as well as explain how to add pop-up windows to a page using an HTML code editor as well as one of Dreamweaver's built-in behaviors.

Deciding when to launch a new browser window

Some designers open browser windows for many of the following reasons:

- ✓ To display a close-up image or detailed views of a product or item
- ✓ To provide a special logon area for registered visitors
- ✓ To open a secure window inside which registered members can access their account information
- ✓ To view an Adobe Acrobat PDF file within the browser
- ✓ To display special notices, sale information, or shipping details
- ✓ To open a resource page on another Web site
- ✓ To provide technical data or other information in a printable format

These diverse uses are all good reasons to open new browser windows while leaving the original window intact; however, not all Web designers feel the same way about launching new browser windows.

Some designers think that new browser windows should be launched anytime a link takes visitors away from the main domain they are visiting, and when any non-HTML file, like a PDF, is opened. Other designers take a more firm and limiting approach, with the view that new browsers should never

be opened without the express consent of the visitor. These designers believe additional windows are a nuisance, like those spamlike pop-up advertising windows that appear automatically when you visit certain Web sites (or even worse, the pop-up windows spawned by other pop-ups that plaster the screen, against the visitor's will, and sometimes can only be stopped by shutting down the computer).

Whatever your personal view, here are general guidelines for you to follow regarding when (and when not) to open new browser windows on your site:

- ✓ Do open new browser windows for links to external Web sites.
- ✓ Do open new browser windows for links to non-HTML documents like PDFs.
- ✓ Do notify site visitors, either visually or with text, when a new window will be opened. This can be done using an icon, like the example shown in Figure 8-6, or screen tip of some kind, such as the words "View close-up" or "Page opens in new window."
- ✓ Do understand that users may have customized their browsers to treat pop-ups differently than you intend. For instance, you may have visitors who allow pop-ups but use the TabMixPlus add-on to force pop-ups to open in another tab rather than in their main browser window or as a regular pop-up window.
- ✓ Never launch an advertising window. Visitors don't want to be forced to see anything, let alone have to close an annoying advertisement window. Unless it is an emergency or you have a really good reason to launch a pop-up, keep general information within the same browser window.



Figure 8-6: Alert visitors with a graphic or screen tip when a link will open a new window.



As with all JavaScript, remember that pop-up windows may be inaccessible to visitors who use screen readers and other assistive devices. To ensure that all content in a pop-up window is accessible to the widest possible audience, make sure that you include special `<noscript>` tags in the code directly following the script to launch the pop-up. Between the `<noscript>` tags, you can include the content you would like these visitors to access, or include a regular hypertext link to the pop-up page so that visitors can access that page directly. Additional enhancements to accessibility include placing the notification of a new window opening in the `alt` text attribute for linking images, adding the `title` attribute to text hyperlinks, and ensuring that all pop-up windows include a Close button or Close hypertext link to make closing the window as easy as possible.

Hand-coding the script to launch a pop-up window

Opening another browser window, as you discover in Book III, Chapter 1, can be easily accomplished by including the `target` attribute (`target="_blank"`) within a hyperlink, as in the following code:

```
<a href="http://www.wiley.com" title="Search For Books"
  target="_blank">Search For Books</a>
```

This method opens the target URL inside a new browser window, and that new window has the same attributes and dimensions as the parent window that spawned it. In other words, if the parent window (the page with the link) has a navigation toolbar, bookmark toolbar, and status bar, the child pop-up window (the linked page or document) has those features too.

Sometimes, however, you may want the child browser window to be precisely sized and stripped of some of its “browser chrome.” That’s where the JavaScript comes in. To control the size, browser chrome (window attributes), and screen position of your pop-up browser window, follow these steps:

- 1. Open a new file in your favorite HTML code editor, type `Popup` and select it, and then convert the selection into a null link.**

Though you could use the number symbol (`#`) to create the null link, in this example you’ll be applying a JavaScript behavior to the link. To do that, apply a JavaScript null link to the code instead, as in this example:

```
<a href="javascript:void(0);">Popup</a>
```

See the nearby sidebar for more on null links.

2. In the code of the page, between the opening and closing `<head>` tags, insert the following script:

```
<script type="text/JavaScript">
<!--
function MM_openBrWindow(theURL,winName,features) { //v2.0
    window.open(theURL,winName,features);
}
//-->
</script>
```

This script informs the browser how to handle a JavaScript request, which you add to the code in the next step.

3. Add the additional JavaScript to the hyperlink HTML, which instructs the browser to open the linked page in a resizable pop-up window named `MyWindow` that is 200 x 200 pixels in size and includes a toolbar, location bar, status bar, menu bar, and scroll bars:

```
<a href="javascript:void(0);"
onClick="MM_openBrWindow('http://www.google.com','MyWindow','toolbar=yes,location=yes,status=yes,menubar=yes,scrollbars=yes,resizable=yes,width=200,height=200')">Popup</a>
```



The browser window chrome, such as the toolbar and menu bar, can be easily removed from the pop-up window by omitting those attributes from the JavaScript. For example, to launch a 500-pixel square pop-up window with no browser attributes, the code would look like this:

```
<a href="javascript:void(0);"
onClick="MM_openBrWindow('http://www.google.com','MyWindow','width=500,height=500')">Popup</a>
```

4. To see how this script functions, save the page and open it in a browser window.

When you click the `Popup` link, your custom-sized browser window pop-up opens above the parent window and displays the Google Web page. Figure 8-7 shows an example of the link page and resulting pop-up window.

5. If you'd like to also control where on the visitor's screen the new pop-up window appears (relative to the upper-left corner of the parent browser window), hand-code the `top` and `left` attributes into the JavaScript:

```
<a href="javascript:void(0);"
onClick="MM_openBrWindow('http://www.google.com','MyWindow','width=200,height=200,top=80,left=200')">Popup</a>
```

Some browsers may not launch a pop-up and will instead open the new page in another tab. Therefore, if you don't see the pop-up result in your preferred testing browser, try another browser.

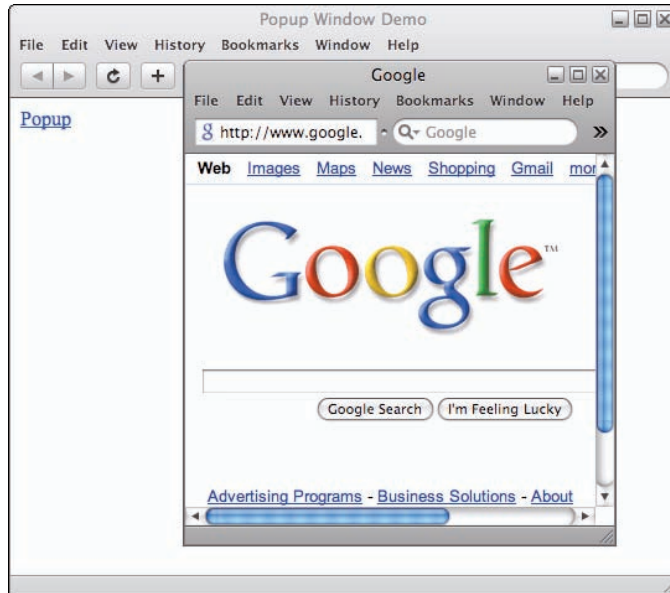


Figure 8-7: Launch a new pop-up window with a tiny bit of simple JavaScript.



Pay close attention to the JavaScript syntax used here. You see no quotation marks around the top and left pixel attributes (as you'd normally use in HTML code), and the entire sizing area is enclosed between two apostrophes. If you happen to use the wrong characters, the JavaScript will not function in a browser.

Adding a pop-up window to your page with Dreamweaver

If you're a Dreamweaver user, you can add the script for launching a pop-up window through the Behaviors panel and not have to fuss with any hand-coding.

To add a pop-up window to your page in Dreamweaver, follow these steps:

- 1. Open a new file in the Dreamweaver workspace and type the words you want to use for the link. Then select the words and convert the selection into a null link by typing `javascript:void(0);` into the Link field of the Property inspector.**

For example, if you type **Get Directions** and add the null link, your code would look like this:

```
<a href="javascript:void(0);">Get Directions</a>
```

Working with null links

A *null link* is an undesignated hyperlink that does nothing and goes nowhere when the visitor clicks the link. Null links can be used as placeholders for real links, when the URL isn't yet known, but you still want to indicate that the text or object is a link, and for attaching behaviors to an object or text link in Dreamweaver through the Behaviors panel.

The simplest way to create a null link is by using a number symbol (#) as a placeholder for the filename or URL of the link, as in this example:

```
<a href="#">Google</a>
```

Unfortunately, because named anchor links are also preceded by the number symbol (#), the browser becomes confused when encountering null links like this. In other words, when the browser can't find the name after the number symbol, as it would with a regular named anchor link, the browser "jumps" the visitor back to the top of the current page, abandoning the visitor's former location within the file.

To avoid having the browser jump to the top of page when it encounters a null link, use either one of the following JavaScript alternatives:

The first method is to create a null link with the `javascript:void(0);` syntax, like this:

```
<a href="javascript:void(0);">Google</a>
```

The second method is to create a null link with the `JavaScript:;` syntax, which Adobe Dreamweaver recommends when attaching behaviors to text or object links, like this:

```
<a href="JavaScript:;">Google</a>
```

While both of these methods are quite good, due to inconsistent browser support they may not always be the best methods for every situation.

As an alternative, you could create a null JavaScript link using the following function:

```
<a href="#" onclick="aFunction();return false;">Google</a>
```

The `return false` part of this bit of JavaScript (which must be the final function call) tells the browser that the action of returning a reference to a URL (including internal links) should return as false. Therefore, the link does nothing and goes nowhere.

2. **With your cursor between any of the letters in the null link (for example, between the *D* and the *i* in *Directions*), click the Add Behavior (+) button in the Behaviors panel (choose Window⇨Behaviors) and select Open Browser Window.**

The Open Browser Window dialog box opens, as shown in Figure 8-8.

3. **Enter a URL to display in the pop-up, input the pixels for the window's width and height, select any of the desired window attributes, and type a unique window name for the pop-up.**

Select only the browser window attributes you want to include in the pop-up. If you do not want a particular attribute, leave that check box deselected, and those attributes will be automatically omitted from the pop-up window. For example, to create a 400-x-400-pixel chromeless

pop-up window to the Google Maps Web site, enter **http://maps.google.com** in the URL to Display field, **400** and **400** in the Window Width and Window Height fields, and **GoogleMaps** in the Window Name field.

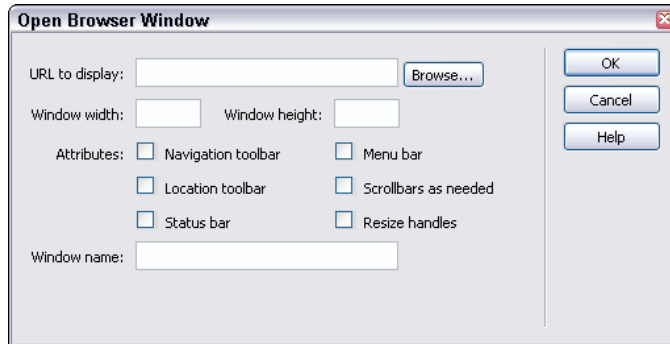


Figure 8-8: Use the Open Browser Window dialog box in Dreamweaver to set the URL and browser window attributes for a new browser window.

4. Click the OK button.

Upon closing the dialog box, Dreamweaver automatically inserts the JavaScript code to make the pop-up window work. If you look at the code, you can see parts of the JavaScript both in the <head> area and as part of the hyperlink tag within the body of the page. Here's the script you see for the example:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Open Browser Window Demo</title>
<script type="text/JavaScript">
<!--
function MM_openBrWindow(theURL,winName,features) { //v2.0
    window.open(theURL,winName,features);
}
//-->
</script>
</head>

<body><a href="javascript:void(0);"
    onclick="MM_openBrWindow('http://maps.google.com
    ','GoogleMaps','width=400,height=400')">Get Directions</a>
</body>
</html>
```

5. To control the position of the pop-up window relative to the visitor's parent browser window, go into the code and hand-apply the top and left attributes to the JavaScript:

```
<a href="javascript:void(0);"
  onclick="MM_openBrWindow('http://maps.google.com
    ', 'GoogleMaps', 'width=400,height=400, top=450,left=150')">Get
  Directions</a>
```

You may need to test the pop-up script a few times to get the exact positioning the way you want it. Without the addition of the `top` and `left` attributes, the pop-up window will open directly on top of the parent, starting at the parent window's upper-left corner.

Building Image Maps

In Book III, Chapter 1, you briefly find out about working with image maps. An *image map* is a graphic with one or more clickable hotspots that take visitors to another page, much like regular hyperlinks.

You can turn any image into an image map by adding coordinates in the HTML code that map out a distinct shape somewhere on top of the image and then assigning a hyperlink to that shape. Hotspots can use the area-shape attributes of `circle` (for circle shapes), `rect` (for rectangles), or `poly` (for irregular polygon shapes).

For example, to make a circular hotspot on a rectangular graphic like the one shown in Figure 8-9, you insert a `<map>` tag and then apply the correct coordinates, relative to the rectangular graphic's upper-left edge, for that circular shape. After the coordinates are right, you then add a link for the hotspot map along with the link target and alt text. The last thing to do to pull all the pieces together is to add the `usemap` attribute to the image tag, as in the following example:

```

<map name="guildmap" id="guildmap">
  <area shape="circle" coords="73,76,62"
    href="http://www.gag.org" target="_self" alt="Graphic Artists Guild" />
</map>
```



Figure 8-9: Use an image map to create linkable hotspots on an image.



Adding image maps with Dreamweaver is much, much easier than hand-coding them. However, if you don't have access to Dreamweaver, check out the free online image map editor on Adam Mascheck's site (www.mascheck.hu/imagemap/imagmap) shown in Figure 8-10. If you enjoy using the online editor, you might consider downloading the desktop editor from www.mascheck.hu/imagemap/download.

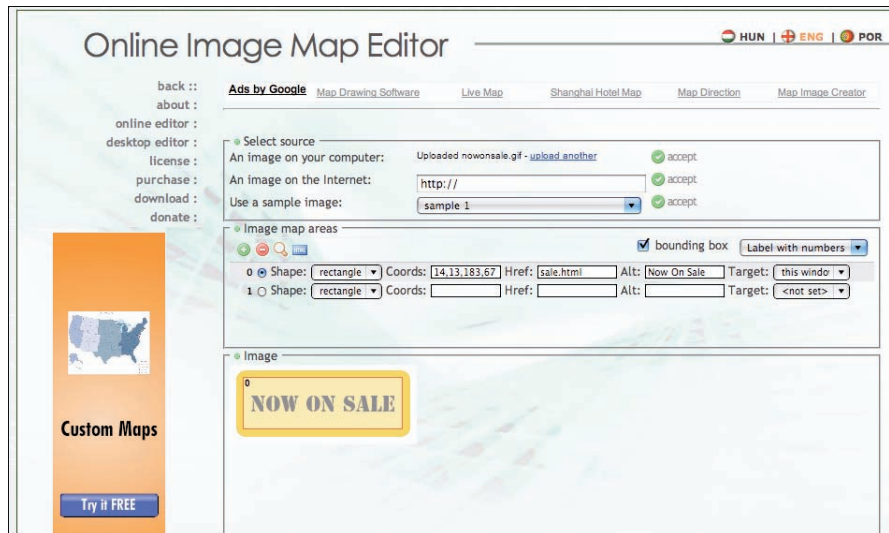


Figure 8-10: Create image maps easily with a free online image map editor.

Adding an image map to a graphic

To add an image map to a graphic using Dreamweaver, follow these steps:

- 1. Open a new blank Web page and insert an image of your choice using the Insert→Image command.**
- 2. Select the image in Design view and give the image a name in the Map field on the left side of the Property inspector.**

The name can be anything you like and will be used to identify this particular image map within the code. This is especially useful when a page contains more than one image map. If you do not give your map a name, Dreamweaver will automatically assign one for you using the default syntax: Map, Map2, Map3, and so on.

- 3. With the image still selected in Design view, select either the Rectangle, Circle, or Polygon Hotspot tool in the Property inspector and create a hotspot shape on top of the image.**

Both the Circle and Rectangle Hotspot tools allow you to create shapes by clicking, dragging, and releasing the mouse. To use the Polygon Hotspot tool, click as many times as needed on top of the image to set down hotspot dots (the coordinates) around the outline of the desired hotspot shape.

After releasing your mouse button, the hotspots appear in Dreamweaver's Design view as light blue, semitransparent cloaks on top of the graphic — with anchor-point squares at each angle along the hotspot path, as shown in Figure 8-11.

At the same time, you may see an alert dialog box that says you need to describe the image map in the alt field. This is just a reminder to make your image maps accessible to visitors with disabilities.



Figure 8-11: Dreamweaver displays image map hotspots as semitransparent blue shapes that sit on top of the image.

- 4. Select the Pointer Hotspot tool (the black arrow in the Property inspector) to select the hotspot you just created, and use the Property inspector to add a hotspot link with a target and alt attribute.**

For example, to add a link to the Adobe Dreamweaver CS4 Resources page, you'd set the target to `_blank` and the alt to Dreamweaver Online Help, and insert the following URL in the Link field:
http://help.adobe.com/en_US/Dreamweaver/10.0_Using.

Your code should look something like this:

```
<map name="MyMap" id="MyMap">
  <area shape="circle" coords="36,36,23" href=
    "http://help.adobe.com/en_US/Dreamweaver/10.0_Using/"
    target="_blank" alt="Dreamweaver Help" />
</map>
```

- 5. To finish the coding, deselect the image and hotspot by clicking away from the image, anywhere inside the page in Design view.**

If desired, create additional hotspots on the same image by repeating Steps 2–5. Otherwise, save the file and launch the page in your favorite browser to test the functionality of your new hotspot.

Building complex image maps

As you can see in the preceding section, a simple image map is super-easy to create with the help of Dreamweaver. What a lot of people don't know, however, is that you can use this same technique to create some really cool

and complex linking systems, especially when you combine the image map with other JavaScript behaviors. For example, check out the instructor group photo on the home page of the Noble Desktop Web site at www.nobledesktop.com, also shown in Figure 8-12.



Figure 8-12: Each person shown here has a clickable hotspot that links to a bio page.

Each instructor in this image has a unique hotspot that links to an individual instructor page. What's more, when you mouse over any instructor's picture, the animated black bar graphic below the instructors' pictures (Your Instructors . . . Straight from the Trenches of Web and Graphic Design) gets swapped with a black bar that identifies the selected instructor by name. To create this effect, several graphics were needed including the main image of the instructors and several versions of the black bar with each of the instructors' names.

To make multiple hotspots on a single image, start by following the steps outlined in the previous section. After you have created the first hotspot, reselect the image and create the next hotspot. Repeat this process until all of the hotspots have been made.

After the image map hotspots are set up on the main image, you can add a little JavaScript to make each hotspot trigger one or more images within the page to change on mouseover. Here's how to do it:

1. Select each of the images on the page, one at a time, and provide names for them in the Name field in the Property inspector.

All the images on the page must have unique names to identify the graphics before you add the Swap Image behavior. This includes the image with the hotspots and all the other images on the page, especially the ones that will be changing when a visitor interacts with the hotspot.

If you're hand-coding, decide what to call each image and then use that in both the `name` and `id` tag attributes, as in the following code example:

```

```

2. Using the Hotspot Selection tool in the Property inspector, select the first hotspot on the graphic, click the Add Behavior (+) button in the Behaviors panel, and choose Swap Image.

When the Swap Image dialog box appears, you should see a listing of all the named graphics on the page.

3. In the Images area of the dialog box, choose the named graphic that you'd like to swap with another image. Then click the Browse button next to the Set Source To field and choose the desired rollover state graphic.

An asterisk appears next to the selected image in the Images field to identify that the image has a graphic assigned to its rollover state.

Leave the Preload Images and Restore Images on MouseOut options selected; both are necessary for Dreamweaver to add the correct JavaScript to the page.

To make more than one graphic change with the same Swap Image behavior, select another named image from the Images area of the dialog box and set the source graphic in the Set Source To field.

4. Click the OK button to close the dialog box.

Dreamweaver adds the appropriate JavaScript to the code to make the rollover effect appear in a browser when a visitor interacts with the hotspot.

5. Repeat Steps 2–4 for each additional hotspot on the image; then save and test the file in a browser.

Moving your cursor over the hotspot areas on the graphic in the browser should make the rollover state graphics appear elsewhere on the page. For enhanced effect, use animated GIFs for parts of the image map rollover configuration.



Adding Multimedia Files

A *multimedia file* is any kind of file that can be viewed, listened to, and interacted with on the Internet — including music, videos, puzzles, QuickTime movies, slide shows, quizzes, animations, and games.

Sadly, because all the different Internet browsers have been developed by different companies over time, not every browser — or even browser version — is equipped to handle media files in quite the same way. For example, some browsers will automatically play certain media files, while other browsers require the installation of special plug-ins, applets, or ActiveX controls (specified in the code along with the media file) before those media files will work. To make matters even more complicated, every media file has its own format, as indicated by its file extension — and not all formats are playable in all browsers or all operating systems.

On the other hand, there is good news. As Internet users have gotten savvier, so have all the major browser developers. Over the past few years, certain media formats have become more standard than others. For instance, MP3, WAV, and RAM are the most popular formats for sound, and FLV, SWF, QuickTime, Shockwave, and MPEG are the most popular formats for video. What's more, all the newest versions of the most popular browsers (Internet Explorer, Firefox, and Safari) automatically have these popular media-file readers preinstalled in them so that visitors can experience the multimedia files in their browsers without having to download any special files from third-party vendors, which even when they are free, are somewhat of a hassle to install.

Adding a multimedia file to your page

The quickest way to add multimedia files to a Web page is to use an HTML editor that can automatically insert the appropriate code for each file type. For example, with the click of a button, Dreamweaver can add Flash animations, Flash video, QuickTime movies, Shockwave movies, Java applets, ActiveX controls, and other plug-ins to your Web pages.

To add a multimedia file to your page in Dreamweaver, follow these steps:

- 1. Create a managed site to a new folder on your computer called** `Multimedia`.

Choose Site⇨New Site to create a managed site. Click the Advanced tab at the top of the Site Definition dialog box, and in the Local Info category, enter a name for the site (`Multimedia`) in the Site Name field. In the Local Root Folder field, browse to and select the new `Multimedia` folder.

2. **Inside the `Multimedia` folder, put a copy of all the media file(s) you'd like to insert onto your page.**

If you will be working with multiple files, you may want to place all the media files inside a media folder at the root level of your site.

3. **Create a new blank page, or open the page you want to add the media file to, and place your cursor at the spot on the page where you'd like to insert the media file.**

Setting the insertion point tells Dreamweaver where to drop in the appropriate code.

4. **From the `Insert`⇨`Media` menu, select the desired media type.**

Alternatively, you can click the desired Media button in the Common category of the Insert panel (shown in Figure 8-13) to access the same list with helpful icons next to each media type. Options in this menu include SWF, FlashPaper, FLV, Shockwave, Applet, param, ActiveX, and Plugin.

To find out how to add sound to your page, see the section later in this chapter called “Adding Sound with Dreamweaver.”

After you select the desired media type, Dreamweaver opens the Select File dialog box.

5. **Browse to and select the media file in your managed site that you want to insert on the page. Then click the OK button.**



Figure 8-13: Add multimedia files to your page with Dreamweaver’s `Insert`⇨`Media` command.

Dreamweaver uses the filename and location of the selected file to ensure that the appropriate path to the file gets added to the inserted code. For example, when inserting a Flash SWF animation file, Dreamweaver adds a ton of code that enables the visitor to view the animation in most browsers, replete with comment tags and conditional rules depending on the visitor's browser scenario:

```
<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000" width="210"
  height="180" id="FlashID" title="MySWF">
  <param name="movie" value="images/sample.swf" />
  <param name="quality" value="high" />
  <param name="wmode" value="opaque" />
  <param name="swfversion" value="6.0.65.0" />
  <!-- This param tag prompts users with Flash Player 6.0 r65 and
  higher to download the latest version of Flash Player. Delete it if
  you don't want users to see the prompt. -->
  <param name="expressinstall" value="Scripts/expressInstall.swf" />
  <!-- Next object tag is for non-IE browsers. So hide it from IE
  using IECC. -->
  <!--[if !IE]>-->
  <object type="application/x-shockwave-flash"
  data="images/sample.swf" width="210" height="180">
  <!--<![endif]>-->
  <param name="quality" value="high" />
  <param name="wmode" value="opaque" />
  <param name="swfversion" value="6.0.65.0" />
  <param name="expressinstall" value="Scripts/expressInstall.swf" />
  <!-- The browser displays the following alternative content for
  users with Flash Player 6.0 and older. -->
  <div>
    <h4>Content on this page requires a newer version of Adobe Flash
    Player.</h4>
    <p><a href="http://www.adobe.com/go/getflashplayer"><img src=
    "http://www.adobe.com/images/shared/download_buttons/get_
    flash_player.gif" alt="Get Adobe Flash player" width="112"
    height="33" /></a></p>
  </div>
  <!--[if !IE]>-->
</object>
<!--<![endif]>-->
</object>
```



In addition to adding the code, Dreamweaver also installs some special dependent files in a `Scripts` folder in your local root folder. This folder, which will include an SWF file and JavaScript file, must be uploaded to the server for the object or behavior to function properly in a browser.

Depending on which media type you select, you may encounter the Object Tag Accessibility Attributes dialog box. If you see that dialog box, enter the requested accessibility information (if desired) and click the OK button to proceed.

6. With the media file still selected in Design view, enter any desired additional parameters, attributes, and dimensions for the media file.

For example, with an inserted Flash video, you can set the width and height of the FLV player, select a skin, apply a CSS style, and choose whether the video will autoplay and/or autorewind.



To find out more about adding media files to your Web pages, take the free online tutorial at www.w3schools.com/media/default.asp.

Creating slide shows

You have several different ways to create a slide show on your site. Some slide shows pair images with JavaScript in a variety of layouts with different functionality, while others combine Flash with text, animation, and graphics.

Some of the galleries, like the jQuery slideViewer shown in Figure 8-14, use the jQuery plug-in (a JavaScript library; see www.jquery.com), which allows you to quickly create an image gallery with just a few lines of HTML code, as in this example:

```
<div id="mySlideViewer" class="svw">
  <ul>
    <li></li>
    <li></li>
    <li></li>
  </ul>
</div>
```

To add the jQuery slideViewer to your page, follow these steps:

- 1. Download the free jQuery slideViewer 1.1 JavaScript file named `jquery.slideviewer.1.1.js` from the following Web site, and save the file to the root level of a managed site in Dreamweaver:**

www.gcmingati.net/wordpress/wp-content/lab/jquery/imagestrip/imageslide-plugin.html

Choose Site→New Site to create a managed site. Click the Advanced tab at the top of the Site Definition dialog box, and in the Local Info category, enter a name for the site in Site Name field. In the Local Root Folder field, browse to and select the folder you will use for this exercise.

- 2. Download the latest version of the free jQuery JavaScript file from www.jquery.com.**

The file will likely be called something like `jquery-1.2.6.min.js`. Save the file to the root level of the managed site.



Figure 8-14: The jQuery slideViewer presents your images in a neat gallery format.

3. **Download the latest version of the free jQuery Easing plug-in from** <http://gsgd.co.uk/sandbox/jquery/easing>.

The filename should be something like `jquery.easing.1.3.js`. Also, save this file to the root level of the managed site.

4. **Add the following code to the <head> area of your page, which will call each of these JavaScript files when the page with the jQuery slideViewer gallery is viewed in a browser:**

```
<script src="jquery-1.2.6.min.js" type="text/JavaScript"></script>
<script src="jquery.easing.1.3.js" type="text/JavaScript"></script>
<script src="jquery.slideviewer.1.1.js" type="text/JavaScript"></script>
```



Take extra care to ensure that the version numbers of the script source match the version numbers of the files you just downloaded. The gallery will only function when the numbers match.

5. **Place all your gallery images into a folder called `images` on the root level of your managed site. If you don't have an `images` folder yet, create one.**



All the images in your gallery must have the same width and height. If they are not the same, some images may be stretched or scrunched to match the size of the first image.

- 6. Select and copy all the slideViewer's basic CSS from the following Web site, paste it into a blank new file, and save the CSS in an external file called gallery.css:**

```
www.gcmingati.net/wordpress/wp-content/lab/jquery/imagestrip/imageslide-
plugin.html
```

- 7. Add a link to the new external CSS file in the head of the page.**

The link code should look like this:

```
<link href="gallery.css" rel="stylesheet" type="text/css" media="all" />
```

- 8. In the body of your page, insert a <div> with a unique ID (such as mySlideViewer) and the class="svw" CSS attribute. Inside the <div>, type in an unordered list of images, as in the following code:**

```
<div id="mySlideViewer" class="svw">
  <ul>
    <li></li>
    <li></li>
    <li></li>
  </ul>
</div>
```

The alt text, as with any image on your page, will be used to make the images more accessible to visitors with disabilities and to search engine spiders and crawlers.

- 9. Directly below the link to the external CSS file in the head of the page, insert the following JavaScript code, which controls how one image slides into the next in the gallery:**

```
<script type="text/JavaScript">
  $(window).bind("load", function() {
    $("#div#mySlideViewer").slideDown( {
      easeFunc: "easeInOutBack",
      easeTime: 1200
    });
  });
</script>
```

The timing of the slide animation from one slide to the next is counted in milliseconds. To change the speed, edit the easeTime number. The smaller the number, the faster the transition.

- 10. Save any open files and preview your new jQuery slideViewer in a browser.**

To change the border and number indicators from red to another color, modify the styles in the CSS.



For additional enhancements to the jQuery slideViewer, read the documentation at www.gcmingati.net/wordpress/wp-content/lab/jquery/imagestrip/imageslide-plugin.html.

This slide show uses HTML, JavaScript, and CSS to display your same-sized images in a unified layout with side-swiping transitions and easy-to-use navigation buttons — all without your having to understand a lick of JavaScript. While it is not necessary that you understand all of the code that makes this script function, what does matter is that you like the end results and can easily re-create it with your own color scheme and graphics.

Adding sound with Dreamweaver

Adding sound to your pages with Dreamweaver is as simple as choosing the right media file format and then configuring that object's properties so that the visitor can experience and interact with the file. With the exception of inserting a Flash media type for SWF sound files, most other media files can be inserted with the ActiveX media type.

After inserting the sound file on your page (using the steps described in the earlier section “Adding a multimedia file to your page”), follow these steps to configure the sound object's properties using Dreamweaver's Property inspector:

- 1. After inserting the sound file on your page through the Insert⇨Media menu, set any desired properties for the inserted object in the Property inspector, such as Width, Height, and Embed. Be sure to also enter the appropriate class ID to identify the file type to the browser.**

Table 8-2 contains a listing of common media type class IDs.

<i>Media Type</i>	<i>Class ID</i>
QuickTime Player	clsid:02BF25D5-8C17-4B23-BC80-D3488ABDDC6B
RealPlayer	clsid:CFCDA03-8BE4-11cf-B84B-0020AFBCCFA
Windows Media Player	clsid:02BF25D5-8C17-4B23-BC80-D3488ABDDC6B

- 2. (Optional) To set the parameters of an embedded ActiveX sound file, select the sound object in Design view and click the Parameters button in the Property inspector.**

You must select the sound file before the button appears on the Property inspector. Once clicked, the Parameters dialog box opens.

Click the plus (+) button to add the appropriate parameters and values, as described in Tables 8-3 through 8-5. When finished, click the OK button.

3. In the ID text field or Base text field in the Property inspector, fill in the following field for your chosen media type:

- **QuickTime:** Enter `http://www.apple.com/qtactivex/qtplugin.cab` in the Base text field.
- **RealPlayer:** Enter `rvocx` in the ID text field.
- **Media Player:** Enter `mediaplayer2` in the ID text field.

To preview your media file, save any changes and then launch the page in a browser window. If you've entered `true` as the `autostart` value, your file should begin to play immediately when the page loads in the browser. In some cases, your file may not play until you upload the files to your server, so be sure to test both locally and on a remote server before making the page available to visitors.

Table 8-3 QuickTime Player Parameters and Values

<i>Parameter</i>	<i>Value</i>
<code>autoplay</code>	<code>false</code>
<code>controller</code>	<code>true</code>
<code>pluginspage</code>	<code>http://www.apple.com/quicktime/download/index.html</code>
<code>target</code>	<code>myself</code>
<code>type</code>	<code>video/audio</code>
<code>src</code>	(enter the path and filename to your file)

Table 8-4 RealPlayer Parameters and Values

<i>Parameter</i>	<i>Value</i>
<code>src</code>	<code>myfilename.rm</code>
<code>autostart</code>	<code>false</code>
<code>controls</code>	<code>ControlPanel</code>
<code>console</code>	<code>audio</code>
<code>type</code>	<code>audio/x-pn-realaudio-plugin</code>

<i>Parameter</i>	<i>Value</i>
src	(enter the path and filename to your file)
autostart	false
showcontrols	true
showstatusbar	false
showdisplay	false
autorewind	true
type	application/x-mplayer-w
pluginspage	http://www.microsoft.com/Windows/Downloads/Contents/MediaPlayer

Providing Daily Interactive Content on Your Pages

One of the best ways to keep visitors coming back to a site regularly is to entice them with fresh, daily content. The content can be anything you like, as long as it is appealing and relevant to the target audience. For example, you might add daily entries in a blog, insert daily Flash movies and games, offer visitors polls and poll results on popular topics of interest, request feedback from visitors using forms, include free online tools for visitors such as calculators and converters, publish regular site-related articles and tips, offer free printable coupons for products and services, host special contests and sweepstakes, and send out daily, weekly, or monthly newsletters with links back to the site.

While each of these ideas may not be suitable to every Web site, some of them function quite well on nearly any site. For example, you can offer daily tips, wisdom, coupons, information about sale items or new items, or other desirable content, as well as post free games that visitors can return to each day to play or replay.

Daily tip or news item

Giving visitors a reason to return to your site each day can increase the chance of them wanting to learn more about the site's products and services. That, in turn, can result in increased Web traffic and improved sales!

Aside from daily blog entries, one of the quickest ways to entice visitors to your site is to provide them with some kind of relevant information each day. This could be in the form of a daily tip or suggestion, a horoscope or

fortune, a poem, a joke, a famous quote, a photograph, a coupon, or details about a new product or service. The daily item can truly be anything you like, as long as you customize the JavaScript to accurately display the desired information.

Searching with Google

Make visitors feel in control over their experience on your site by providing them with a site search tool. You can add the free Google search bar to your site by inserting a tiny bit of code on your page.

To install the Google free search tool with Site Search, follow these steps:

1. Type the following code at the spot on your page where you'd like the search bar to appear:

```
<!-- SiteSearch Google -->
<FORM method=GET action="http://www.google.com/search">
<input type=hidden name=ie value=UTF-8>
<input type=hidden name=oe value=UTF-8>
<TABLE bgcolor="#ffffff"><tr><td>
<A HREF="http://www.google.com/">
<IMG SRC="http://www.google.com/logos/Logo_40wht.gif"
border="0" ALT="Google"></A>
</td>
<td>
<INPUT TYPE=text name=q size=31 maxlength=255 value="">
<INPUT type=submit name=btnG VALUE="Google Search">
<font size=-1>
<input type=hidden name=domains value="YOUR DOMAIN
NAME"><br><input type=radio name=sitesearch value=""> WWW
<input type=radio name=sitesearch value="YOUR DOMAIN NAME"
checked>YOUR DOMAIN NAME<br>
</font>
</td></tr></TABLE>
</FORM>
<!-- SiteSearch Google -->
```

2. Replace the words `YOUR DOMAIN NAME` in the code (in all three locations) with the full URL of your Web site's home page, such as `http://www.mydomain.com`.

To have the search feature appear on all the pages of your site, include this code on each page in the same location for consistency.

Businesses can also harness the power of Google with a fully customized Web site search. Google searches can help increase Web traffic, which can result in increased sales. For more information about Google Site Search, visit www.google.com/coop/cse.

The JavaScript used in the following steps was originally developed by Mike W. at dvol.com and can be found in its entirety at The JavaScript Source (<http://javascript.internet.com>). The script contains two parts: One part must be placed in the `<head>` of the code; the other goes in-line, between the `<body>` tags of the code, at the location on the page where the daily tip content should appear.

Adding a daily tip or news item to your page

To add a daily tip or news item to a page on your Web site, follow these steps:

- 1. Open an HTML file in your favorite code editor and add the following script between the opening and closing `<head>` tags (or download it from <http://javascript.internet.com/text-effects/daily-tip.html>):**

```
<SCRIPT LANGUAGE="JavaScript">
<!-- Original: Mike W. (mikew@dvol.com) -->
<!-- Web Site: http://www.dvol.com/~users/mikew -->
<!-- This script and many more are available free online at -->
<!-- The JavaScript Source!! http://javascript.internet.com -->
<!-- Begin
var msg = new Array();
Stamp = new Date();
today = Stamp.getDate();
msg[1] = "Tip 1";
msg[2] = "Tip 2";
msg[3] = "Tip 3";
msg[4] = "Tip 4";
msg[5] = "Tip 5";
msg[6] = "Tip 6";
msg[7] = "Tip 7";
msg[8] = "Tip 8";
msg[9] = "Tip 9";
msg[10] = "Tip 10";
msg[11] = "Tip 11";
msg[12] = "Tip 12";
msg[13] = "Tip 13";
msg[14] = "Tip 14";
msg[15] = "Tip 15";
msg[16] = "Tip 16";
msg[17] = "Tip 17";
msg[18] = "Tip 18";
msg[19] = "Tip 19";
msg[20] = "Tip 20";
msg[21] = "Tip 21";
msg[22] = "Tip 22";
msg[23] = "Tip 23";
msg[24] = "Tip 24";
msg[25] = "Tip 25";
msg[26] = "Tip 26";
msg[27] = "Tip 27";
msg[28] = "Tip 28";
msg[29] = "Tip 29";
msg[30] = "Tip 30";
msg[31] = "Tip 31";
```

```
function writeTip() {
document.write(msg[today]);
}
// End -->
</script>
```



To use this free script, you must leave the four comment tags (`<!--` like this `-->`) at the beginning of the code that identify the developer's name and the location for downloading the free JavaScript; if you remove them, you will be in breach of the author's copyright.

- In Code view, place your cursor at the spot on your page where you'd like the daily tip to appear and add the following JavaScript:**

```
Daily Tip: <script>writeTip();</script>
```

This script "calls" the JavaScript in the `<head>` of the page and returns the results of the script on the page in the browser. For example, if today is the seventh day of the month, the daily tip named `msg[7]` (which is currently uncustomized and will say "Tip 7") will automatically be displayed on the page when viewed in a browser.

If you are working in Dreamweaver, be sure that you add this JavaScript to the code in Code view, not in Design view.

- Now that you know that the script is functioning properly, it is time to customize each tip. Go back into the JavaScript in the `<head>` of the page and customize each of the tips between the opening and closing quotes.**

For example, to customize the first tip, you'd change

```
msg[1] = "Tip 1";
```

to

```
msg[1] = "Change will not come if we wait for some
other person or some other time. We are the ones
we've been waiting for. We are the change that we
seek. &#8212; Barack Obama";
```

To display an image as the daily tip instead of or in conjunction with text, add the `` tag between the quotation marks of the JavaScript tip area and replace the quotation marks for attributes inside the inserted image with apostrophes. The result should look something like this:

```
msg[1] = "Text and a graphic <img src='images/
coupon1.gif'>";
```

Converting a daily tip JavaScript into an external .js file

JavaScript, like CSS, can be placed in an external JavaScript file. By placing some of or all the JavaScript into an external file, you help to minimize the code on the actual Web page while at the same time centralizing the

instruction for the JavaScript into one location that an unlimited number of HTML files can access and use via a link in the HTML code to the external file, like this one:

```
<script src="dailytip.js" type="text/JavaScript"></script>
```

When multiple JavaScript files are used on a site, all the .js files can be housed in a single `scripts` folder at the root level of the site, which further helps keep your site tidy and organized:

```
<script src="scripts/dailytip.js" type="text/JavaScript">
</script>
```

To convert the daily tip JavaScript into an external .js file, follow these steps:

- 1. Cut the JavaScript code from the `<head>` of the page, paste it into a blank file, and save it with the .js extension at the root level (or in a folder at the root level) of the Web site it will be used in.**

To keep things easy to remember and in the spirit of semantic HTML, name the external file after the function in the script. In this case, you might name the external file `dailytip.js`.

- 2. Inside the new external `dailytip.js` file, delete the opening and closing `<script>` tags. Then, save the file, close it, and return to the open HTML file that will display the tip.**

The comment tags should stay in and will not affect the functionality of the script.

- 3. Between the opening and closing `<head>` tags, insert a link to the external `dailytip.js` file, like this:**

```
<head>
<script language="JavaScript" type="text/JavaScript"
  src="dailytip.js"></script>
<noscript>
This page includes a daily tip generated with JavaScript. To see a
  complete listing of this month's tips, visit our <a href="http://
  www.mysite.com/dailytips.html">Monthly Tips</a> page.
</noscript>
</head>
```

To make the script more accessible, the example includes a set of `<noscript>` tags with descriptive information about what the JavaScript does and (when applicable) how visitors can access that information.

After you've updated all 31 tips in the JavaScript, and presuming that you installed the script near the first of the month, you can virtually forget about it for about 27 days or so — until it's time to change all the tips again for the following month. Keeping your site fresh keeps your visitors interested.

Daily word game

One great place to find free content for your Web site is The Free Dictionary Web site:

www.thefreedictionary.com/lookup.htm#sitecontent

At this site, you can get the free code for any or all the following content options: Word of the Day, Article of the Day, This Day in History, Today's Birthday, In the News, Quote of the Day, Spelling Bee, Word Match Up, and Hangman.

For example, to add a free Hangman game to your site, like the one shown in Figure 8-15, just enter the code in Listing 8-1 somewhere on your page.



Figure 8-15: Get free daily site content, like this Hangman game, at www.thefreedictionary.com.

Listing 8-1: Hangman Game

```
<!--Hangman by TheFreeDictionary.com-->
<div style="width:350px;position:relative;background-color:;padding:4px">
<div style="font:bold 12pt '';color:#000000">Hangman</div>
<style>
#Hangman {border:1px #000000 solid;background-color:;height:120px}
</style>
<iframe id="Hangman"
src="http://www.thefreedictionary.com/_/WoD/hangman.aspx?#,x000000,x0000FF,1
0pt,'" width="100%" scrolling="no" frameborder="0"></iframe>
<div style="font:normal 8pt '';color:#000000">
<a style="color:#000000"
href="http://www.thefreedictionary.com/lookup.htm#Hangman">Hangman</a>
provided by <a style="color:#000000"
href="http://www.thefreedictionary.com/">The Free Dictionary</a>
</div></div>
<!--end of Hangman-->
```

Daily blog entries

If your site has a blog, making daily, weekly, or monthly entries is a must. In fact, the more content you make available to visitors, the more visitors you will get. Entries can be as simple as a new photo with a caption, tip of the day, or a recipe to as complex as a tutorial with screen shots or an opinion piece about a current news item or proposed change to public policy.

You have several ways to add a blog to your Web site. For a refresher on working with blogs, see Book I, Chapter 1.

Chapter 9: Building Web Sites

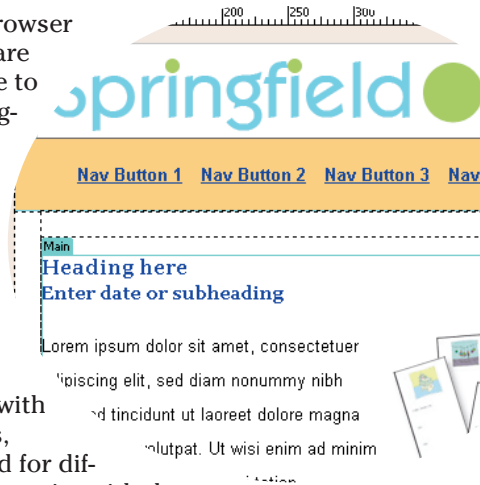
In This Chapter

- ✓ Understanding the benefits of using templates
- ✓ Creating and using templates
- ✓ Working with Server-Side Includes
- ✓ Understanding site-root and document relative paths
- ✓ Comparing templates and SSIs

At this stage, you've probably already designed your mock-up for the site, optimized all the graphics, and chosen how to lay out your pages using layers and tables, and you have a basic understanding about how to style the content with CSS. The next step you take in building your Web site is to create a master page from which you generate all the other pages on the site. Though each page will have different content, the general layout for every page will likely be the same, with the logo, navigation, and other elements in the same location throughout the site. While it really doesn't matter which page you decide to use to build the site's master page, the home page is usually a good choice, unless for some reason it's drastically different in look and layout from the rest of the site.

Saving time, eliminating busywork, and leveraging browser (and ISP) caching to improve the speed of your site are three of the main reasons why you use a master page to build your site. A fourth reason has to do with managing future site updates after your site is finished. In particular, you have two site-construction solutions for the nondynamic Web site that work beautifully: templates and Server-Side Includes, or simply SSIs. These tools, whether used singly or in conjunction with one another, enable you to change every page on a site simply by making the changes once to your master page or master SSI.

In this chapter, you find an overview about working with templates and Server-Side Includes. Both techniques, which work in markedly different ways and are suited for different purposes, can help you make global updates to a site with the least amount of effort. In addition, this chapter shows you how to work with both methods, plus gives you a comparison chart to assist you in determining which solution will work best for any site you happen to create.



Building the Master Page

To build this master page, you should have certain tools at your disposal so that starting a new page with the master requires the least amount of redundant work. You also want a master file that enables you to quickly and efficiently make updates to the pages at a later time. You might not think you need to consider site updates until after you build the site, but in fact the projected frequency of the updates can be a major determinant for how you create the site's pages from the start.

Whichever page on your site you choose to develop, your master page should include all the features that are common to all pages on the site. For instance, if a navigation bar appears across the top of the layout and a set of footer links appears across the bottom of the pages, the code for these elements will not need to be rebuilt for each of the remaining pages on the site. For that matter, even copying and pasting the code from one page to another is too much work.

The two most effective ways for creating a master site page and building your site are templates and SSIs, which are described in the remaining sections of this chapter. Both methods enable you to make site-wide changes quickly and efficiently.



When you build a master page from the site's home page, an added benefit is that it can provide your clients with a peek into the site-building process and get them even more excited about seeing the finished product. It also gives them an opportunity to review a sample HTML page for layout accuracy and functionality. That way, should the client raise any issues regarding the layout or navigation, you can correct those concerns before generating any of the other pages on the site.

Building Web Sites with Templates

If you're using a code editor like Adobe Dreamweaver or Microsoft Expression Web, your application probably contains some kind of system for creating and using templates. A template not only establishes visual consistency between the pages but can also significantly reduce the time it takes to build the rest of the pages on the site and make changes to those pages anytime the site needs global modifications, both during the site-building process and for any postlaunch site maintenance.

Templates are a great solution for most small- to medium-sized Web sites (under about 30 pages) that use little to no dynamic capabilities because

template-based pages can be updated by the application rather quickly with minimal effort on the part of the designer.

After the template file is created, you can specify which areas on that master page will be editable in any template-based pages. For example, you might want to create an editable area for the main content on the page and another area for the sidebar content. With the editable areas established, anytime a new template-based page is created, only the content in those editable areas can be altered, whereas the rest of the content is locked down and remains uneditable.

In the following sections, you find out how to create Dreamweaver templates, set up editable regions, and create and manage template-based files. If you are using another code editor, you should be able to easily adapt the examples found here to build your site using your program's template solution.

Using Dreamweaver templates

The true beauty of a template-based Web design is that, should any part of the locked, uneditable part of the template need altering (like a navigation button that is no longer needed on the site or sidebar content that needs to be added in), rather than having to individually update all the pages, only the template would need adjustment. The code editor then automatically updates the changed, locked code on all the template-based pages.

While most sites have only one master template, you are free to create as many templates for a site as you need to; there is no limit to the number or complexity of the templates you create for each Web project. Templates can have more than editable regions, too. Designers working with Dreamweaver can create nested templates, optional editable regions, and repeating regions within the template:

- ✓ **Nested templates:** These are special templates that are created by embedding a template inside another template, such as when one section of the site uses a special layout that requires its own set of editable regions and that layout falls within the editable area of the parent template, as with a product details page on an e-commerce site.
- ✓ **Optional editable regions:** This is an editable region that you specify on a template that can either be shown or hidden in a template-based page, such as a link back to the top of a page on pages with lots of content.
- ✓ **Repeating regions:** This editable region of a template can be used as a repeating element that can be repeated as often as needed inside a template-based page, such as an entire table row containing a set of editable regions.



Another really useful aspect of working with Dreamweaver's templates is the fact that you can integrate them with Adobe Contribute software. Sites built with Dreamweaver templates can be managed by the designer through Dreamweaver, while Contribute users (like the client) can be authorized to modify the editable regions of that site without needing to know anything about Web design. Contribute is extremely easy to learn and use and puts the client in control of some of his or her site's editability.

This chapter only skims the surface of the amazing things you can do with Dreamweaver templates. As your skills grow, you can experiment more with advanced techniques, such as nested templates and so on. To discover everything there is to know about working with Dreamweaver templates as well as how to create a Dreamweaver site and manage it with Contribute, get *Dreamweaver CS4 All-in-One For Dummies*, by Sue Jenkins (published by Wiley).

Preparing a page to become a template

You can build a Dreamweaver template in two ways. You can either build templates from scratch with a blank HTML template page, or convert any existing HTML/XHTML page into a Dreamweaver template by choosing File⇨Save as Template.

Because you are new to Web design and presumably new to Dreamweaver too, building a sample page in HTML or XHTML first and then converting it into a template file will probably make more sense than building a template file. Furthermore, by creating the HTML file first, you will have a chance to work out any kinks in the regular (X)HTML document before it becomes the master template you use to create all the other pages on your site.

To prepare your page for becoming a Dreamweaver template, follow these steps in roughly the same order:

1. **Create a new folder on your desktop (Template Demo, for example), create a new managed site pointing to this new folder, and place a new, empty folder named images at the root level of the managed site.**



Managing a site provides access to Dreamweaver's advanced site-management features, and you must perform this step when working with templates so that Dreamweaver can appropriately write code to manage template-based file updates.

To create a managed site, choose Site⇨New Site. When the Site Definition dialog box opens, click the Advanced tab. In the Local Info category, enter a name for the site in the Site Name field (such as **Template Demo**), and in the Local Root Folder field, browse to and select the folder (Template Demo) you just created on your desktop.

2. Create a new blank HTML or XHTML page with the desired DTD (Document Type Definition) and save the file with any name you like, such as `master.html`, at the root level of your managed site.
3. Build the sample page. Set up the layout using `<div>` tags. Get all the parts of the layout in their proper places by adding the content and inserting the graphics with `alt` text attributes. Add footer links at the bottom of the page.
4. Insert `<meta>` tags in the `<head>` of the code and any of the other accessibility attributes to the code you can think of to make the page as accessible as possible.
5. Add dynamic functionality to your page where desired, such as rollover buttons or initial layer visibility, and insert hyperlinks throughout the page where needed.

See Book III, Chapter 8 for more on making your site interactive.

6. Style and position all the content with CSS.

See Book III, Chapter 3 for the lowdown on CSS.

7. Run a spell check and read through the code looking for errors.
8. Test the page in as many browsers and browser versions as possible on Mac, PC, and Linux platforms and on any other devices you can test. Fix anything that needs fixing.

Also be sure to test for code accuracy using Dreamweaver's Validation and Browser Compatibility reports in the Results panel. If desired, validate the page using an online validator.

9. When the page looks good on every browser and you are happy with the results, show the page to your client to get feedback and input.
10. Make any adjustments to the site's layout if needed and get the client's approval again, in writing, before you build more pages.

Step 10 warrants further explanation. The reason it says "get the client's approval again, in writing" is that clients may sometimes have "brilliant ideas" at this stage. They might think of another page to squeeze into the navigation, want to move something over (even though they've already seen the design and approved it), or add a whole new section to the layout. If your contract states that no modifications can be made at this stage without an addendum to the contract (and it should!), remind your client of this clause and tell him you'd be delighted to make any changes he'd like for an additional fee. If the client is serious about the change, he'll agree to the addendum. But if his brilliant idea was just whimsy or, more often than not, just a natural inclination to want to feel a part of this process, he'll change his mind about wanting to make any modifications to the design and give you his signed approval to continue.



Creating a Dreamweaver template

After you and your client are fully confident in the layout and styling of your sample Web page, you'll be ready to convert that page into a Dreamweaver template. Use your own file if you have one prepared, or use the sample file called `createtemplate.html` at www.dummies.com/go/webdesignaio.

Follow these steps to convert an HTML or XHTML file into a Dreamweaver template:

1. **Using the `master.html` sample Web page you created in the previous steps, open the file in Dreamweaver's workspace, choose **File** ⇨ **Save as Template**.**

The Save as Template dialog box opens, as shown in Figure 9-1. The name of your managed site (Template Demo, for example) should appear in the Site drop-down menu.

If you've defined a site but that site name isn't showing on the drop-down menu, click the menu's down arrow to select the name of the site for this project.

Unless you or someone else has already created a template for this managed site, the Existing Templates field should be empty with the words (no templates) listed inside it.

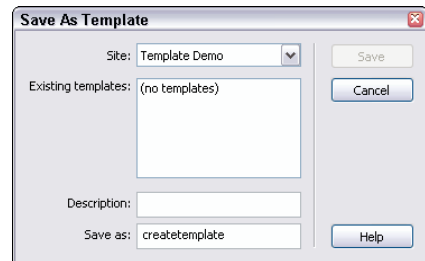


Figure 9-1: Use the Save as Template dialog box to provide details about the template.

2. **(Optional) In the Description text field, type a short description for the template you're about to create.**

This description can be a sentence that describes the Web project or a short phrase about what this template will be used for.

For example, if the template is for a photographer's portfolio Web site, the description might read something like **Diallo Photography**. If you're using the example page, you might enter **Springfield** or **Template Demo**.

3. **In the Save As field, enter a filename for the template.**

A good naming convention for templates is to use the client's name or something similar. For instance, if the client's company is called Home Design Consultation, you might name the template **homedesign** or **HDC**.

4. **Click the Save button to keep your template settings, and when the Update Links dialog box appears, as shown in Figure 9-2, click the Yes button.**

Dreamweaver converts the HTML file into a template with the .dwt file extension. At the same time the template is being created, Dreamweaver also creates a new `Templates` folder in the root directory of the managed site and saves your template file inside it.



Clicking the Yes button in the Update Links dialog box allows Dreamweaver to automatically update links to graphics and other files within your HTML file with the appropriate document relative path syntax (which you find out about in the later section “Editing Paths to Work with SSIs”) to match the new location of the template file inside the `Templates` folder. Saying Yes ensures that any template-based files created from the template will use accurately syntaxed paths.

Figure 9-2: Click the Yes button when the Update Links dialog box appears.

For example, a link to `about.html` in the original HTML file will be updated to `../about.html` inside the new template file. Likewise, a link to an image such as `` would be changed to ``. The dot-dot-slash (`../`) before the link location tells a browser to go up a level in the server’s directory to find the specified file.



If you accidentally click the No button in the Update Links dialog box, the links in the template won’t function properly. Should that happen, close the new template file, delete it from the new `Templates` folder, and begin again at Step 1. Then make sure you click the Yes button when the Update Links dialog box appears again.

Creating templates with editable regions

By default, the entire Dreamweaver template is locked, which means that any pages generated from it at this point can’t be edited, unless you want to make global changes to all the pages by editing the template itself or unless you specify one or more editable regions.

Follow the next steps to create an editable region in the template:

- 1. With the template file open in the Dreamweaver workspace, place your insertion point inside the document, preferably in Design view, and select all the text in the main content area of the page.**

If you’re using the `createtemplate.html` file (which you may have downloaded while completing the “Creating a Dreamweaver template” section earlier in this chapter), select the contents within the file from “Heading Here” to the last line of greeking text that ends with “. . . amet, consectetur adipiscing elit.” If you are using your own template file, select a region within the file that you would like to be editable in your template-based pages.

The selected content will define the first area on the template that will be converted into an editable region.

2. **Choose Insert→Template Objects→Editable Region. When the New Editable Region dialog box appears, enter a name to identify the region, such as Main, and click the OK button.**

The editable region name, as shown in Figure 9-3, identifies the region inside the template as well as inside any template-based pages you create from it.

3. **Select the next area in your document that you'd like to convert to an editable region by selecting content, choosing Insert→Template Objects→Editable Region, and giving each new region a unique name.**

Templates may contain an unlimited number of editable regions, but editable regions may not be nested inside other editable regions.

4. **Save and close the template file.**

You find out how to create a template-based page in the next section of this chapter.

5. **To modify the template at any time, and thereby update any locked areas in any template-based files, reopen the template file, make the changes to the code, and save the page.**

Upon saving any changes to a template, Dreamweaver asks whether you want to automatically update all the template-based pages. As long as you're in a managed Dreamweaver site and you say yes, Dreamweaver will handle rewriting all the changed code to any template-based file within that managed site.



Figure 9-3: Editable regions display with a blue, named tab and an outline.



All Dreamweaver template files contain special template comment tags that identify the beginning and end of each editable region within the file. *Do not* modify these comment tags; without them, Dreamweaver cannot understand how and where to make global modifications to template-based page code.

Here's an example of what these special comment tags might look like in your template file:

```
<!-- TemplateBeginEditable name="Main" -->
<h2><a href="#" title="Heading Title">Heading here</a></h2>
<h3>Enter date or subheading</h3>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit,
    sed diam nonummy nibh euismod tincidunt ut laoreet dolore
    magna aliquam erat volutpat...</p>
<!-- TemplateEndEditable -->
```

In addition to the beginnings and endings of editable regions, Dreamweaver templates also include an editable area in the `<head>` of the HTML code, which can be used for inserting JavaScript, CSS, and other `<head>` area elements that need to be placed inside the head of a template-based file. These tags have the name attribute of `head`:

```
<!-- TemplateBeginEditable name="head" -->
<!-- TemplateEndEditable -->
```



As with the editable region comment tags, *do not* modify these `<head>` comment tags either, because they are used to help you manage site content.

Creating and editing template-based files

Template-based files are easy to create and use because the pages behave just like any other HTML page except that they have locked-down, uneditable regions that are controlled by the template.

To create a template-based page in Dreamweaver, follow these steps:

1. Choose **File** ⇨ **New** to open the New Document dialog box, click the **Page from Template** category, select the name of your managed site from the Site listing, and select a template from the list of available templates for that site.
A preview of the selected template appears in the Preview panel, as shown in Figure 9-4.
2. Click the **Create** button to open a new, untitled, template-based page in the Dreamweaver workspace window.
3. Choose **File** ⇨ **Save** to save the new template-based file with a filename and extension of your choice (like `services.html` for a Services page) to the same managed site.

For best results, the untitled file should be saved before you begin adding content to the editable regions.

The editable regions in the template-based file are outlined in blue and display a blue tab with the region name at the upper-left corner of each editable area. Uneditable regions will be inaccessible (neither clickable nor selectable) both in Design view and in Code view.

4. Edit the content as needed in the editable regions of the template-based file.

Add text, create hyperlinks, and insert images, tables, and other content. Style your content with CSS and add any behaviors or other dynamic features to enhance the page.

5. Choose File→Save to save your page after making changes.

6. Repeat Steps 1–3 to create additional template-based files.

You may also create additional template-based files by taking an existing template-based file and using the File→Save As command to create a duplicate copy of that file with a new filename.

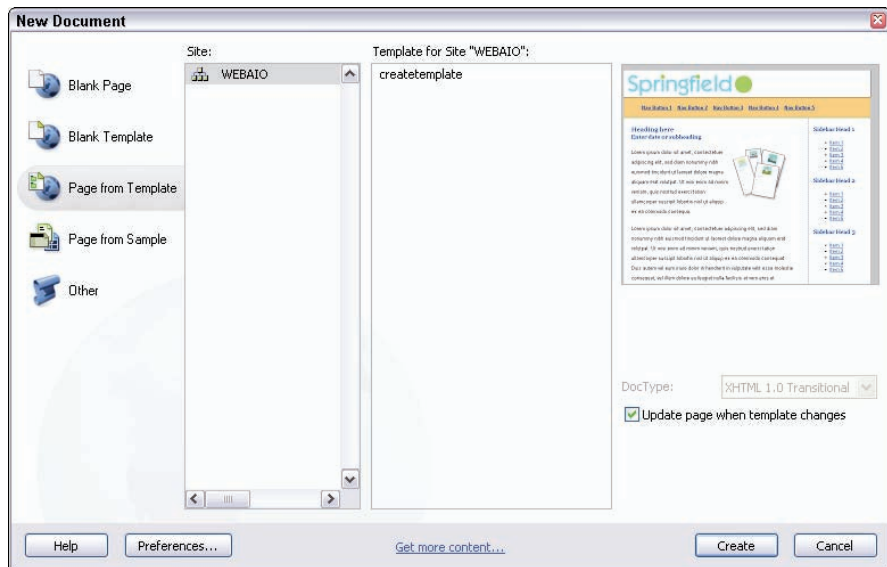


Figure 9-4: Create a new document from your existing template with the New Document dialog box.

Working with Server-Side Includes (SSIs)

Another way to manage sites — which may be a better solution than using Dreamweaver templates for larger sites, sites that use programming to control dynamic content, and sites that will need frequent updating — involves using a process called Server-Side Includes (SSIs).

Understanding what SSIs are

Unlike templates, which hard-code all the pages on a site and require the user to set editable regions for template-based pages, SSIs break Web pages into components, kind of like pieces of a puzzle. To assemble the whole page, you pair the main page content, which is unique to the individual page, with the different SSI pieces, which will be the same on every page throughout the site:

- ✓ The main page contains the general HTML structure and perhaps a few layout elements on the page that never change.
- ✓ The pieces, composed of HTML, are actually external HTML files that are plugged into the main page with a single line of code where the content should appear.

The browser that displays the page seamlessly integrates any SSI content by grabbing the information inside the SSI files from the server and displaying that content as if it were hard-coded into the main page.

SSI files are best for repeating page elements like navigation bars and footer links, as well as other components or content on pages that require frequent updating, such as sponsor listings and promotions. SSI files can be composed of text, graphics, JavaScript, Flash movies, and anything else that might go on a regular Web page. In fact, the only HTML code the SSI files shouldn't contain are the “bones” of the Web page, namely the `<html>`, `<head>`, `<meta>`, `<title>`, and `<body>` tags. SSI files do not need those structural code elements because those tags are already provided by the main HTML page where the SSI content is included.

Including an SSI file inside a page

Say, for example, that you have decided to put a site's footer content into an SSI file called `footer.html` partially because it is a repeating element on every page and partially because you know the site will be growing and the footer's navigation links will need updating before too long. That `footer.html` page might include links to all the current main pages on the site, some descriptive text, a link to an RSS feed, and copyright information with the year, as shown in Figure 9-5.

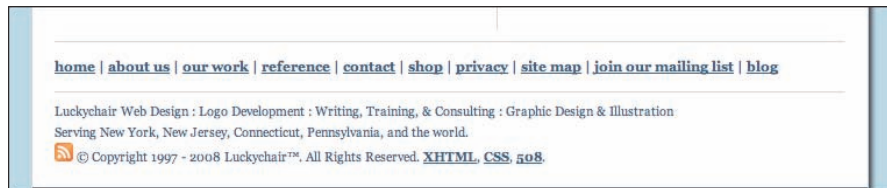


Figure 9-5: SSI content, like this footer area, can include anything found elsewhere on a site, such as text, hyperlinks, and images.

To include the `footer.html` SSI file's content inside another page, you must add a special link to your page in the location the SSI's content should be displayed. The link uses comment tags to identify the link as an SSI and includes the filename and path to the external SSI file, as in the following code example:

```
<!--#include file="footer.html"-->
```

By using that same code link, presumably in the same location on every page, you can include the `footer.html` SSI content on as many pages of a site as desired. And because the content is pulled from the SSI file and displayed within the page by the browser on the fly as the visitor views the page, the visitor can't tell the difference between actual hard-coded content and content parsed in an SSI file. In fact, if visitors were to view the source code of a page that had an SSI link inside it, they wouldn't see the `include` link tag in the code; instead, they would see the actual content from the SSI file, in-line with the rest of the code, as if you had written the page without SSIs. The server and the browser do all the work to ensure that visitors see the parsed SSI content presented seamlessly as part of the page in the browser.

Parsing, by the way, refers to the browser's ability to extract information from one file and apply it to another file. When referring to SSIs, this process involves the browser requesting information from the server and then embedding the returned data into the page that displays in the browser at the specified location in the code.

After you've inserted the SSI `include` link on a page, nothing else needs to be done with the page to ensure that content displays in the specified location. Later, when the SSI content needs updating (such as updating the year in the copyright or adding a new button to a navigation bar), the only file you'll need to edit and reupload to the remote server hosting the site is the external HTML SSI file (in this example, `footer.html`). After the updated SSI file has been uploaded to the server, all the pages on the site that use that `include` will automatically display the updated version of the external SSI file's content.

Editing an SSI file

Editing an SSI file is just like making changes to any other HTML file. Simply open the SSI file in your favorite HTML code editor, as illustrated in Figure 9-6, make the changes, and save the file. As long as you remember to upload the updated version of the SSI file to the remote server, the changes will appear on any page that includes a link to that particular SSI file.



Figure 9-6: SSI files can be updated in an HTML editor just like regular HTML files.

Ensuring success with SSIs

While SSIs might seem like a perfect solution for creating easily updatable Web sites, a few things must be done beforehand to ensure that you can successfully use them:

- ✓ **The remote host server must be capable of parsing the SSI data.** Before you do any work, contact your host provider or system administrator to find out whether the host server is capable of this function. Some servers can do the work but must have special software installed to do it, while others simply can't be configured to perform the task.
- ✓ **The extension on files including any SSIs may need to be changed from .html to .shtml.** This means that visitors trying to reach a particular page, such as `www.yoursite.com/contact.shtml`, will not be able to access that page if they directly type in the address as `www.your site.com/contact.html`.
- ✓ **The SSI link code must be accurate.** The link to the SSI file can contain a couple different variables. If these variables are incorrect, the server hosting the pages will not be able to understand how to parse the data contained in the SSI file. The syntax of an SSI link is also slightly peculiar and must be accurate to ensure that the SSI file is parsed.

- ✓ **You must alter the paths to any links and graphics inside the SSI file.** SSI files require site-root relative paths to function properly. You find out how to do that in the next section.
- ✓ **Files that require parsing place greater demands on the server, which can translate into slightly longer download times.** The delay might be only a fraction of a second longer, but it's there, and in some cases might be very noticeable. Any perceived delay could affect visitor loyalty.

If your server can handle SSIs and you determine that this is the solution for you, read on to find out how to create, insert, and test your SSI files.

Creating, Including, and Testing SSIs

After you've ensured that your host server can process SSIs, find out whether you'll need to change the file extension on pages *that contain* SSIs (not the SSI include files). The host provider should be able to tell you the answer, yes or no. If yes, the change would require you to add an *s* before the normal file extension, such as changing `index.html` to `index.shtml` (or from `index.htm` to `index.shtm`). The *s* in front of the `.html` or `.htm` extension tells the server that SSIs are included in the file being displayed and that the server needs to do a little extra work to parse the SSI content. If no, that's great news, meaning you may use the regular `.html` or `.htm` extensions for all your files, including those containing SSIs.

After that's sorted out, your next step is to ensure that the SSI `include` link code within the body of your page is accurate. Depending on the host server's type, you'll either use the word *file* or the word *virtual* in the SSI link, as in the following examples:

```
<!--#include file="footer.html"-->  
<!--#include virtual="footer.html"-->
```



Notice how the `include` link begins and ends with comment tags. Pay particular attention to the lack of space between the opening comment tag, the number symbol, and the word *include*. This entire SSI include link must be typed accurately for the server to provide the data to the browser when the page is viewed.

The name of your SSI file can be anything you like, but in keeping with the goal of semantic HTML, try to name your SSI files after the function they serve or the content they contain, such as `header.html` or `copyright.html`. When saving your SSIs, you can save them to any location on the host server as long as that path is referenced in the link to it. For instance, the SSI can reside at the root level of your site, or if you plan on having multiple

include files, they can all be stored in a folder called `ssi`, as in the following example:

```
<!--#include file="ssi/footer.html" -->
```

If your SSIs will reside in a folder on a completely different URL on a different server, just be sure to specify the full path to the SSI file:

```
<!--#include file="http://www.mysite.com/ssi/footer.html" -->
```

The last step to using SSIs successfully is to update any paths to graphics or other files in the HTML of the SSI `include` files from document relative to site-root relative. In other words, any hypertext links or references to objects or images that appear on the page must use the site-root relative paths. You find out how to do this in the “Editing Paths to Work with SSIs” section, later in this chapter.

In the following steps, you convert part of a Web page — the footer — into an external SSI and then include that external SSI back into the page it was removed from:



1. Open a completed HTML file in your favorite HTML editor.

If you need a sample file to work with, use the `createssi.html` file, which you can download from www.dummies.com/go/webdesignaio.

This page contains a header with a graphic, a navigation bar styled with CSS, a page heading and subheading, body text, a sidebar, and a footer.

2. With your cursor in the code, select and cut all the footer content, from the opening `<p>` tag below the `<div>` tag with the id of footer to the closing `</p>` tag right before the closing footer `<div>` tag, as shown in Figure 9-7.

```

62         nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut
63         wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit
64         lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure
65         dolor in hendrerit in vulputate velit esse molestie consequat, vel illum
66         dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio
67         dignissim qui blandit praesent luptatum zzril delenit augue dui dolore
68         te feugait nulla facilisi.Lorem ipsum dolor sit amet, consectetur adipiscing
69         elit,t.</p>
70     </div>
71     <div id="footer">
72         <p><a href="link1.html" title="Link 1" target="_self">Link 1</a> | <a href=
73         "link2.html" title="Link 2" target="_self">Link 2</a> | <a href="link3.html" title="Link 3" target=
74         "_self">Link 3</a> | <a href="link4.html" title="Link 4" target="_self">Link 4</a> | <a href=
75         "link5.html" title="Link 5" target="_self">Link 5</a><br>
76         Copyright &copy; Sitename YEAR &#8226; All Rights Reserved &#8226; <a href=
77         "http://www.sitename.com" title="www.sitename.com" target="_self">www.sitename.com</a></p></div>
78     </div>
79 </body></html>

```

Figure 9-7: An SSI can be created from existing content within an HTML page.

If you are using the `createssi.html` file, the footer code you select and cut from the page should look like this:

```
<p><a href="link1.html" title="Link 1" target="_self">Link 1</a> | <a
href="link2.html" title="Link 2" target="_self">Link 2</a> | <a
href="link3.html" title="Link 3" target="_self">Link 3</a> | <a
href="link4.html" title="Link 4" target="_self">Link 4</a> | <a
href="link5.html" title="Link 5" target="_self">Link 5</a><br>
Copyright &copy; Sitename YEAR &#8226; All Rights Reserved &#8226; <a
href="http://www.sitename.com" title="www.sitename.com"
target="_self">www.sitename.com</a></p>
```

After you select and cut the code from the page, the only tags remaining in that part of the code should be the opening and closing `<div>` tags for the footer.

The cursor should now be flashing in the code in the space between the footer `<div>` tags. Leave your cursor there.

- 3. Open a new, blank HTML file and delete all the HTML code that your code editor might have automatically placed in the file, such as the `<dtd>`, `<html>`, `<head>`, `<meta>`, `<title>`, and `<body>` tags.**

The document should be completely blank, with no HTML tags, markup, or other content inside it.

- 4. Place your cursor at the top of the blank file's Code view and paste the footer content you just cut from your original file (`createssi.html`, for example).**

The pasted code should be exactly the same as what you cut from the other file, as shown in the code example. If the code has any extra line spaces, you may remove them. Figure 9-8 shows how the code should look in your new file, with none of the structural HTML tags.

- 5. Save this new file with the pasted code as `footer.html` to a new folder called `ssi` inside the same folder where you saved the original file. Then close the file.**

The root level of the folder you're working in should now contain three items: the original file (`createssi.html`, for example), a folder called `ssi`, and inside the new `ssi` folder, the new `footer.html` file.

- 6. Back in the original file, your insertion point should still be blinking between the footer `<div>` tags where you previously cut the code. At this exact spot, type the link to the new external SSI file inside the new `ssi` folder:**

```
<div id="footer">
<!--#include file="ssi/footer.html" -->
</div>
```



```

1 <p><a href="link1.html" title="Link 1" target="_self">Link 1</a> | <a href="link2.html" title="
  "Link 2" target="_self">Link 2</a> | <a href="link3.html" title="Link 3" target="_self">Link 3</a>
  | <a href="link4.html" title="Link 4" target="_self">Link 4</a> | <a href="link5.html" title="
  "Link 5" target="_self">Link 5</a><br>
2   Copyright &copy; Sitename YEAR &#8226; All Rights Reserved &#8226; <a href=
  "http://www.sitename.com" title="www.sitename.com" target="_self">www.sitename.com</a></p>
3

```

Figure 9-8: The Web content saved in an SSI file does not need to be surrounded by any structural HTML tags.

Dreamweaver users can use the Insert⇨Server-Side Include command to select the SSI file and have the link automatically inserted in the code. Upon adding the code, you may even suddenly see the SSI content magically appear in Design view, in-line with the rest of the page content, while still only seeing the `include` link in Code view, like in the example shown in Figure 9-9.



Note that the word *file* is used here as the type of SSI. In some servers, the SSI will not appear unless you change the type to *virtual*. Either way, you have a 50/50 chance of being right the first time, so if *file* doesn't make the SSI appear on the page when viewed in a browser and the server is SSI capable, *virtual* should do the trick.

7. Save the page and preview it in a browser to see the results.

The SSI content should appear at the bottom of the page, just as it would in a regular HTML file.

Dreamweaver users who do not see the SSI content in their browsers may need to enable the Preview Using Temporary File option in the Preview in Browser category of the Preferences panel. Non-Dreamweaver users need to upload the sample files to a live server that can parse SSI files to see the SSI in action.

If you still can't see the SSI file, be sure to also test whether the file extension needs to be altered. To get the test file working, you might need to change the file extension on the page that has the SSI link in it from `.html` to `.shtml`, and even possibly modify the include type attribute from `file` to `virtual`.

If you do see the SSI file on your page, you're about 90 percent finished with the SSI configuration. To make your SSIs fully functional, you must ensure that any links inside the SSI file use site-root relative paths. In the later section "Adjusting paths in an SSI file from document relative to site-root relative," you modify the paths to work properly and test them in a continuation of the preceding steps.

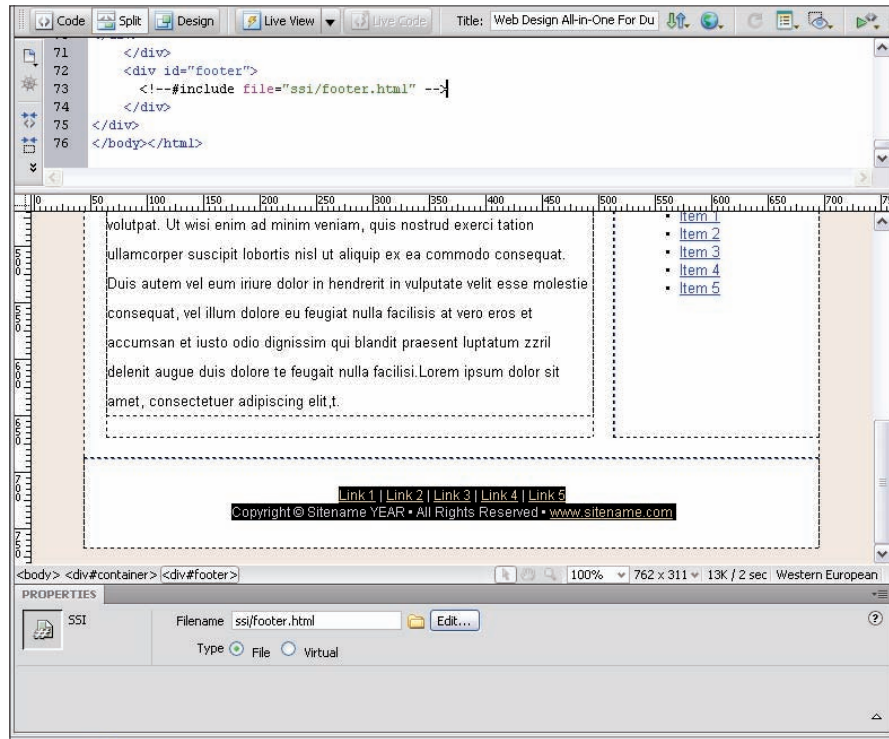


Figure 9-9: Dreamweaver users can see the `include` link in the code while viewing the content in Design view.

Editing Paths to Work with SSIs

In Web design, two types of paths can be used in HTML when referring to files, graphics, and other objects: document relative paths and site-root relative paths. On most Web sites, you use document relative paths, which have been used thus far in this book. For SSIs to work, however, the paths inside any external SSI files must be changed to site-root relative paths.

The following sections explain more about the two path types as well as show you how to adjust the paths in an SSI file from document relative to site-root relative.

Understanding the different path types

The following two sections outline the major differences between document relative and site-root relative paths.

Document relative paths

Document relative paths are the default type of path system in Web pages. With document relative paths, the path code is always relative from starting point A to destination point B, and the full URL is generally omitted from the path reference. For example, a document relative path from the home page to a subpage of a Contact Us section called Directions might look like the following code example, where the `href` specifies a file called `directions.html` that sits inside a folder named `contact` that resides at the root level of the Web site's server:

```
<a href="contact/directions.html">Directions</a>
```

Because the Directions page sits in a directory (folder) called `contact`, both the directory and filename must be declared in the path. If you were to turn the path into a sentence, you might read it as, "Starting from here, go to the `contact` folder and open the page called `directions.html`." When that page is displaying in the browser window, a link on that page back to the home page would need a path that tells the server how to get to the home page from its present location by using the `../` syntax before the filename, which tells the browser to look for the named file "one level before" the current level on the server:

```
<a href="../index.html">Home</a>
```

Site-root relative paths

Site-root relative paths require a little bit more code to tell the server to always look for the named file or object relative to the site's root. To do that you add a slash before any link or sourced object. If you were to turn a site-root relative path into a sentence, you might read it as, "Starting always from the root level of the site, go to the `contact` folder and open the page called `directions.html`." Likewise, to display an arrow graphic, you might say that the path tells the browser to start at the root level, look for a folder called `images`, and display a graphic inside that folder called `arrow.gif`, as in the following:

```
<a href="/contact/directions.html">Directions</a>
```

Furthermore, when you open the Directions page, the path back to the home page from the `directions.html` page would look like this:

```
<a href="/index.html">Home</a>
```

As you can see, the path to the image stays the same because in site-root relative lingo, it always tells the server to start at the root level, look for the `images` folder, and then find and display the named graphic.

In the case of the `footer.html` SSI example in the earlier section “Creating, Including, and Testing SSIs,” all the links in the SSI file to the main navigation pages would need to use site-root relative paths for the server to parse the included data properly on all the pages of the site.



A good analogy to help you remember the difference between document relative and site-root relative paths is to think about giving directions. Document relative directions would tell you how to get to New York City from your current location, wherever in the world you might be. Site-root relative linking, by contrast, gives you directions to New York City, or any other destination for that matter, always starting from the same place in Los Angeles.

Adjusting paths in an SSI file from document relative to site-root relative

To adjust the paths in an external SSI file from document relative to site-root relative, follow these steps:

- 1. Open the SSI file that you created in the earlier section “Creating, Including, and Testing SSIs” (`ssi/footer.html`) in your favorite HTML editor.**

The footer file contains text and a handful of hyperlinks to other pages. As you can see in the code, each hyperlink lists just the filename and file extension, such as `link1.html`.

- 2. In Code view, insert a slash before each linked filename, as in ``. When finished, save and close the file.**

Adding the slash makes the path to each of the hyperlinks site-root relative so that the server and browser can properly parse the SSI file data.

This file doesn’t contain any graphics, and because you’d also need to modify the paths to graphics where they are part of the SSI code, in the next steps you add a graphic to the footer and then modify the path to be site-root relative.

- 3. In the code of the `footer.html` file (directly before the word `Copyright`, for example), place your cursor where you want to insert a graphic. Then insert the graphic of your choice.**

For example, if you were to insert the graphic called `bluearrow.gif` from an `images` folder at the root level of your site, your code would look something like this:

```
<p><a href="link1.html" title="Link 1" target="_self">Link 1</a> | <a
href="link2.html" title="Link 2" target="_self">Link 2</a> | <a
href="link3.html" title="Link 3" target="_self">Link 3</a> | <a
href="link4.html" title="Link 4" target="_self">Link 4</a> | <a
href="link5.html" title="Link 5" target="_self">Link 5</a><br />
Copyright &copy; Sitename YEAR All Rights Reserved </p>
```

Normally, an image like this, which is located in an `images` folder, would use the default document relative path with the `../` before the folder name for the source of the image. However, because this image is now inside an external SSI file, the path must be converted to one that is site-root relative.

4. Edit the path to the image from document to site-root relative by removing the `../` syntax and inserting a single slash before the `images` folder name instead:

```

```



A lot of people forget about this step when creating SSI files; however, it is easy to remedy if you do happen to forget. In fact, as long as you are testing the page prior to publishing, you should be able to catch any forgotten paths to graphics during your testing phase. The same goes for regular hyperlinks: Although the links themselves might look normal in the browser, when clicked, their paths will not bring a visitor to the correct page until you convert the paths into site-root relative paths.

To help you remember all the different things that must be done to ensure that your pages using SSI will parse correctly when displayed in a browser, here is a list of questions you can ask yourself when working with SSIs:

- ✓ Can my host server handle SSIs?
- ✓ Does the SSI link specify the proper type, file or virtual, for my host server?
- ✓ Are all the SSI include code links properly syntaxed?
- ✓ Do I need to change file extensions on pages containing SSI links from `.html` to `.shtml`?
- ✓ Are all the paths to documents and images in the external SSI file site-root relative?

As long as you can remember to check all these things, you should be able to use SSIs to construct your Web sites.

Comparing Templates and SSIs

Now that you understand Dreamweaver templates and Server-Side Includes, you'll probably agree that both are great solutions to building sites that need regular updating. Furthermore, you may have also come to the conclusion that they each have different benefits and drawbacks. Templates are good for smaller sites managed by only one or two people, whereas SSIs are good for large sites that may be making regular updates to certain pages.

Choosing which method to use to build your site can be accomplished by answering a few questions about the site's size, functionality, server type, and future management plans. If you're unsure whether to use templates or SSIs on your site, use the suggestions in Table 9-1 to help you decide.

Table 9-1 Building Web Sites: Templates versus SSIs

<i>Use Templates If . . .</i>	<i>Use SSIs If . . .</i>
The site is small (under 30 pages).	The site is large (over 30 pages).
The site will be managed by only one or two people.	The site will be managed by two or more people.
Site updates will be made on a regular or semiregular basis.	The site will require regular or frequent global and partial updates.
Anytime changes are made to the template, all the updated template-based pages must be uploaded to the site's server before visitors can see those changes.	Any time changes are made to any SSI files, only those updated SSI files need to be uploaded to the site's server before visitors can see the changes.
The code editor has a template system.	The host server supports SSIs.
	The site is database driven (ASP.NET, JSP, PHP, CFML).

Book IV

Web Standards and Testing

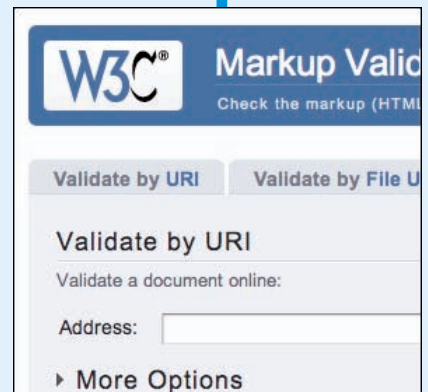
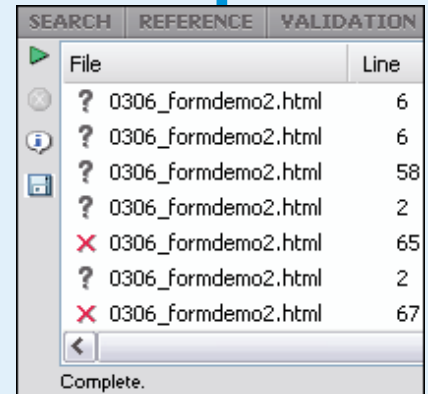
The 5th Wave By Rich Tennant



“Why don’t you try blurring the brimstone
and then putting a nice glow effect
around the hellfire.”

Following the World Wide Web Consortium (the W3C, www.w3.org) recommendations for writing standards-compliant, accessible code is a must in today's Internet world. The more you can find out about this important topic, the more visitors will be able to access your sites.

Chapters in this minibook cover details about following Web and accessibility standards and topics that relate to performing prelaunch testing, cleaning up and correcting common problems in your code, and validating your HTML and CSS markup.



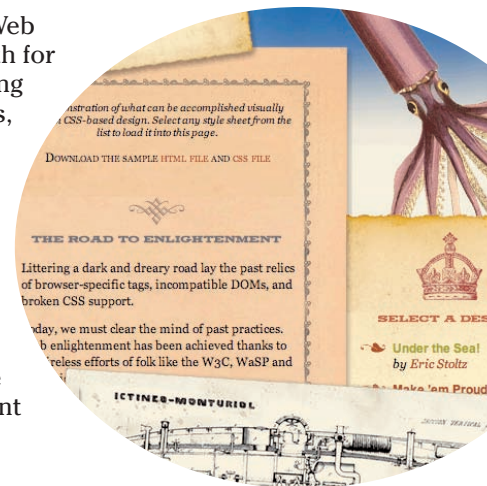
Chapter 1: Following Web Standards

In This Chapter

- ✓ Following Web standards
- ✓ Finding out about W3C standards online
- ✓ Using the right DOCTYPE
- ✓ Understanding the differences between HTML and XHTML
- ✓ Discovering why CSS is better than HTML for styling
- ✓ Writing Section 508–accessible code

Web standards are an important part of the Web that every designer, coder, and programmer needs to understand and use. The standards generally focus on how a Web page works under the hood, but they can also have some significant implications for a site’s design. Most importantly, these standards help ensure that anyone and any device (such as a screen reader or search engine robot) using the Web — regardless of their browser, device, or operating system — can view the content on a Web page.

In this chapter, you find out about the World Wide Web Consortium (W3C) and some of the goals it sets forth for Web design. You also find an introduction to following some of these standards, including using DOCTYPEs, styling page content with CSS instead of HTML tags, and writing valid semantic HTML and XHTML code. In addition to the standards that keep pages accessible and running smoothly across the Web, the federal government outlines another set of standards for making Web pages *accessible* to people with disabilities. At the end of the chapter, you find a discussion about accessibility issues and how the federal government’s Section 508 amendment to the Rehabilitation Act prescribes additional ways content should be coded for the Web.



Working with Web Standards

In the early days of the Web (which was developed in 1989), it was a lot like the Wild West. Anyone who was willing to take the time to explore its uncharted territories was welcome to do so, making up his or her own coding and presentation rules along the way to survive in the then largely unknown Internet world. Because designers had no rules to follow, Web site navigation took on any and every form, making many Internet users feel frustrated and confused as to how they should go about finding the information they sought.

The Internet, at its start, was primarily a place for sharing information and not really a place for commerce or artistic expression. Yet as more and more businesses began using the Web space as a way to market and advertise their products and services, the urgent need for Web standards became apparent. Regarding standards compliance today, thankfully, most of the big software companies are on board. This means that as you find out more about creating Web sites, your HTML and WYSIWYG code editors should be guiding you along the way by writing standards-compliant code. Of course, you will still likely encounter issues with incompatible scripts, IE-only features, and other browser-specific coding concerns, but for the most part, standards have helped smooth a path to make the job of the Web designer a little easier. As with acquiring many new skills, if you follow the right way to do things from the start, you shouldn't have any bad habits to break later on to keep improving your skills.

In the following sections, you find out more about the importance of designing sites that follow Web standards, the W3C recommendations for Web standards, and layering Web content.

Understanding the importance of writing standards-compliant code

First and foremost, designing Web sites that follow Web standards helps ensure that anyone using the Web — regardless of their browser, device, or operating system — can view the content on a Web page. Additionally, following Web standards also makes sites easier to maintain and thus makes them an even more cost-effective method for communicating with site visitors than traditional methods of marketing and communication. The more all Internet software and hardware manufacturers comply with these W3C Recommendations, which are described in the following section, the better all Web visitors' experiences can be. That's where you come in.

As an added bonus, besides being accessible to the widest possible audience, standards-compliant Web sites are more likely to load faster in a browser and tend to have better search engine rankings than their nonstandards-compliant

counterparts. Ultimately, though, the number one reason to use Web standards is that by following them, you can honestly and proudly present yourself as a professional Web designer or developer. This not only makes you look good, but it also makes your Web clients look good too, and that's good for everybody's business.

To help be a part of this Internet utopia, you must do your part to follow the recommendations when writing the HTML, XHTML, CSS, JavaScript, and other programming code for your Web sites. Most WYSIWYG code editors, such as Dreamweaver and Expression Web, do a respectable job of writing standards-compliant code (particularly when certain Accessibility preferences are set within the applications). However, it's ultimately up to the designer — especially if that person intends to hand-code, hand-edit, or use HTML code editors like TextWrangler and BBEdit — to ensure that the code is written in correct, valid, *semantic HTML* (code that uses tags to accurately define contents, such as `` tags for list items) that follows the recommendations of the *World Wide Web Consortium (W3C)*, the organization that helps develop these standards.

Taking a look at W3C recommendations

In 1994, Tim Berners-Lee founded the World Wide Web Consortium (W3C) as an international vendor-neutral group dedicated to bringing standards to the Web with the ultimate goal of making all software and hardware Web accessible. The W3C's mission is “to lead the World Wide Web to its full potential by developing protocols and guidelines that ensure long-term growth for the Web.” Since its founding, the W3C has published nearly 100 “W3C Recommendations” for Web standards, including the following:

- ✓ **The conformity to uniform methods of coding HTML and XHTML:** HTML has new rules to follow that would improve the presentation of pages across a wider array of devices. *XHTML*, an enhanced version of HTML, follows stricter coding rules to improve the accessibility of pages across browsers, operating systems, and other devices that access the Internet.
- ✓ **The inclusion of DOCTYPEs in Web code:** By adding a DOCTYPE tag to the code of all HTML and XHTML pages, the browser can interpret a Web page as an application in the XML programming language. As a standard, this is important because XML allows programmers to create their own proprietary markup languages through which even more information can be exchanged on the Web. You find details about adding DOCTYPEs later in this chapter.
- ✓ **The use of Cascading Style Sheets to style Web content:** Separating content (HTML) from presentation (CSS) and interaction (JavaScript and other programming languages) frees designers and programmers to streamline their code and centralize their presentation markup.

Figure 1-1 shows an example of an early, unorganized Web site from 1998, which was built with frames (which aren't used much anymore) and offered three different navigation options for visitors, including a jump menu, graphic buttons, and hypertext links. Now that we have standards, the Web is a much more organized, efficient place to shop, research, mingle, and interact.

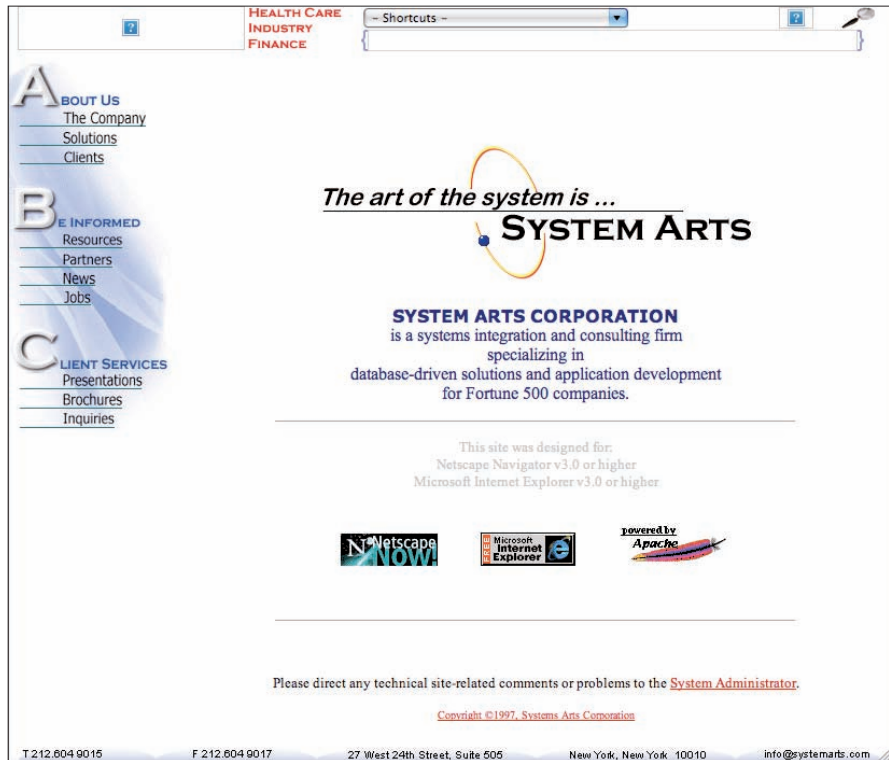


Figure 1-1: Early Web sites were often a jumble of links, graphics, and text.

Exploring the W3C Web site

Indisputably, the W3C Web site is the primary source for finding the latest information about anything Web related, from CSS and document formats to browser compatibility and Web graphic formats. You can even find documentation about more complex issues and initiatives that most designers have never even heard of.

If for some reason you haven't had the chance to visit the W3C Web site, shown in Figure 1-2, take a few moments now to explore some of the following areas of the site (at www.w3.org):

The screenshot shows the W3C website with the following layout:

- Header:** W3C WORLD WIDE WEB CONSORTIUM logo and the tagline "Leading the Web to Its Full Potential...".
- Navigation:** Activities | Technical Reports | Site Index | New Visitors | About W3C | Join W3C | Contact W3C
- Main Content:**
 - Validator Donation Program:** Includes a "Make a donation" button and text about becoming a sponsor.
 - W3C Supporters:** Text about making a donation through the W3C Supporters Program.
 - Employment:** Section for "W3C is seeking a Managing Director" and "Current W3C Fellows Program openings".
 - W3C A to Z:** A list of links including Accessibility, Amaya, CC/PP, Compound Document Formats (CDF), CSS, CSS Validator, Databinding, DOM, Efficient XML Interchange, and Geolocation.
 - News:**
 - Tim Berners-Lee Speaks at TED2009:** Article dated 2009-02-04 about Tim Berners-Lee's talk on Linked Data.
 - Security for Access to Device APIs from the Web: Workshop Report Published:** Article dated 2009-02-05 about a workshop report on API challenges.
 - Search:** A Google search bar with "Search W3C" and "Go" buttons, and a link to "Search W3C Mailing Lists".
 - Testimonials:** A testimonial for Fotosearch Stock Photography and Footage.
 - Members:** Links for "Member Home Page" and "Member Submissions".

Figure 1-2: The w3.org site is your primary source for Web standards.

- **XHTML documentation:** On the left side of the screen, under W3C A to Z, scroll down to and click the link for XHTML. Your browser jumps to the new XHTML2 Working Group Home Page, which outlines all the HTML and XHTML resources on this site. To find out more about XHTML specifications, click the Specifications link near the top of the page.
- **W3C site layout:** To find out more about how the W3C site is structured, click the New Visitors link at the top of the page. This site contains a *ton* of information, which for some can be overwhelming. However, with an understanding of where to find the data you are seeking, the site should become easier to navigate and be ultimately more useful to you.

Click the Recommendations link in item 2 under the heading “How is the W3C Web site organized?” The Recommendations page lists all the W3C’s recommended Web standards. New standards are added regularly as each new standard is approved.

- **Search feature:** To find information about a particular topic, return to the W3C home page and use the Google Search W3C feature in the upper-right corner of the page.

Try to spend some time understanding these standards. The more you can gain knowledge about them, the better you will be able to build Web sites that are standards compliant and accessible to the widest audience.



In addition to the W3C, several other organizations exist to provide recommendations for following standards on the Web, to fight for consistency and accessibility, and to offer suggestions and resources for compliance. As soon as you can, be sure to visit and bookmark the URLs shown in Table 1-1 so that you can review their offerings and quickly return to them anytime you have a question about Web standards and accessibility. All these organizations have a bold commitment to creating standards that set precedents for structural markup languages (like HTML and XHTML), presentation languages (like CSS), scripting languages (like JavaScript), object models, and other additional markup languages, such as MathML and SVG. Be sure to also review and bookmark the Max Design Web Standards Checklist at www.maxdesign.com.au/presentation/checklist.htm.

Table 1-1	
Web Standards Resources	
<i>Resource</i>	<i>Web Address</i>
W3C	www.w3.org
The Web Standards Project	www.webstandards.org
The Web Standards Group	www.webstandardsgroup.org
The Equality and Human Rights Commission	www.equalityhumanrights.com
Section 508	www.section508.gov
Web Accessibility in Mind	www.webaim.org

As a matter of fact, you can do something to let visitors know that you've taken care to write standards-compliant code and have your code successfully pass HTML/XHTML validation tests. After reviewing your code and passing compliance validation tests, you can mention somewhere on your Web site (usually in the footer of every page) that the site complies with Web standards. For example, Jeffrey Zeldman, the design standards guru who had one of the first personal Web sites online back in 1995, notifies visitors to his www.zeldman.com site that his code both conforms to XHTML, CSS, RSS, and Section 508 standards and provides on-the-fly validation of his pages by adding simple text links in the footer of his pages, as shown in Figure 1-3.

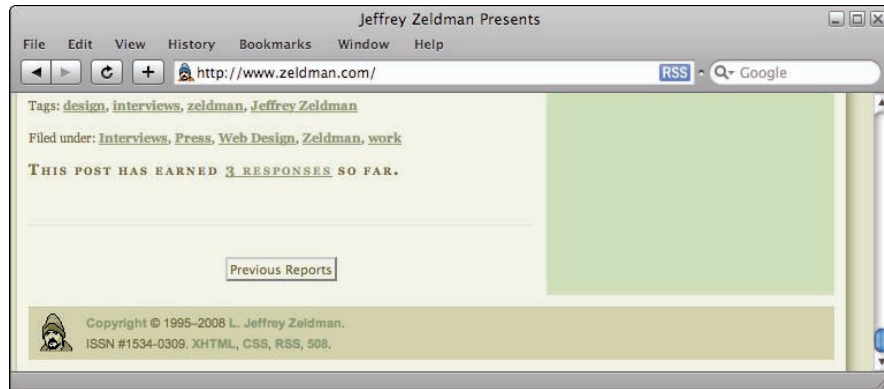


Figure 1-3: Adding simple text links in the footer of your pages.

Using DOCTYPEs (DTDs)

Though DOCTYPEs have been around since about 1999, only in the past couple of years have they started getting the kind of respect and attention they were intended to have. A *DOCTYPE* — also often referred to as a Document Type Definition, or DTD, or sometimes even a Document Type Declaration (again, DTD) — is a set of instructions in the code of an HTML page that tells a browser how to identify the type of code that the page was written in as either HTML, XHTML, or Frames. More importantly, the DOCTYPE informs the browser how the document should be interpreted as an application of the XML programming language. *XML*, which stands for eXtensible Markup Language, is an easily customizable programming language, like SGML (Standard Generalized Markup Language), for the communication of information and application services between people and computers using structured and meaningful semantic code.



By taking care to use the proper DTD on all your Web pages, you can improve the accessibility of your Web site to both human and nonhuman visitors alike while also ensuring that your page code is valid.

Selecting a DOCTYPE

The DOCTYPE, as I mention briefly in Book III, Chapter 1, is a line of code that gets added to the top of each Web page, directly above the opening HTML tag (or opening XHTML tag in an XHTML file). In addition to informing

the browser which markup language the page uses, the DOCTYPE associates an XML or SGML file with a DTD. As shown in Figure 1-4, which uses the HTML 4.01 Transitional DTD, the DOCTYPE must be placed at the top of the HTML code, before the opening `<html>` tag, shown here:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

```
1 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
2   "http://www.w3.org/TR/html4/loose.dtd">
3 <html>
4 <head>
5 <title>My Page Title</title>
6 </head>
7 <body>
8   Page content goes here
9 </body>
10 </html>
11
```

Figure 1-4: Place the DTD above the opening HTML tag.

The DTD itself, whether it's for an HTML-, XHTML-, or Frames-based page, is composed of two parts:

- ✓ **Definition:** The first half is the markup language identifier, which matches the DTD type to the type of code used in the Web document. For example, the following DTD would be used when writing HTML 4.01 transitional code:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
```

- ✓ **Declaration:** The other half of the DTD specifies the URL of a Web-accessible text file that contains more information about that DTD's usage:

```
"http://www.w3.org/TR/html4/loose.dtd">
```

The W3C recommends that all HTML, XHTML, and Frameset Web pages include a DOCTYPE specifying a DTD. The DTD identifies the type of code being used in the document so that a Web browser knows how to interpret or process the information in the code and display the content on the page a little faster.

By now you should already know what HTML is, and probably have a good idea that XHTML is some kind of advanced form of HTML. What you may not know, however, is what Frames means. *Frames* refers to a Web page presentation technique that uses `<frameset>` tags instead of `<body>` tags to make two or more pages display within a single browser window, as illustrated in Figure 1-5. For a tutorial on working with framesets, visit www.w3schools.com/HTML/tryit.asp?filename=tryhtml_frame_mix.

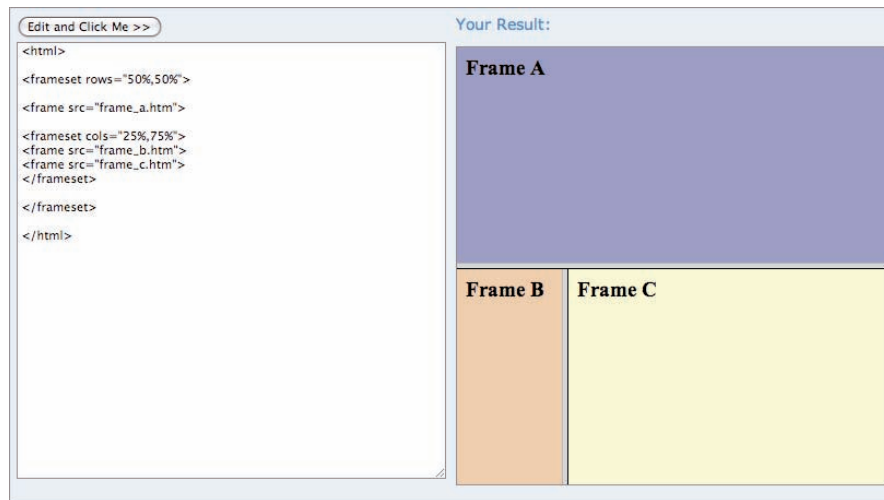


Figure 1-5: The old Frames technique lets you present multiple Web pages within the same browser window.

HTML DOCTYPEs

You can use three types of DTDs with HTML 4.01 on your pages. The first can be used for most, if not all, of your pages because it tells browsers to use the strictest, most accurate, standards-compliant page rendering. Keep in mind, however, that it does require that the HTML contain no coding errors or deprecated tags:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```


The second HTML DTD should be used for pages that might contain legacy code, deprecated tags, and possibly some minor coding mistakes, such as improper tag nesting, all of which do not or cannot comply with strict DTD guidelines. The transitional loose setting tells browsers to be a bit forgiving when interpreting any out-of-date tags and common code blunders:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
```

The third HTML DTD should be used for HTML documents that use frameset tags to display two or more pages within a single browser window:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
    "http://www.w3.org/TR/html4/frameset.dtd">
```

XHTML DOCTYPEs

When working with XHTML code (which you may want to use exclusively instead of HTML when working with programs that use XML), you must choose the correct XHTML DTD. You can choose from three kinds of DTDs when writing XHTML 1.0 code. The first can be used for most of or all your XHTML files that use CSS for page content presentation and adhere to the strictest possible interpretation of standards-compliant code:

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

The second DTD is for XHTML files that might still contain styling and presentation code within the file as well as certain tags and attributes that the strict DTD disallows:

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

The third XHTML DTD should be used for XHTML documents that use frameset pages with XHTML syntax rules:

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

In addition to these DTDs, you find two other XHTML DTDs. Strict XHTML 1.1 is a newer version of Strict XHTML 1.0 based upon the modularization of XHTML:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd"><html
    xmlns="http://www.w3.org/1999/xhtml">
```

This DTD should be used only if you are certain that you can comply with the stricter coding requirements of this form of XHTML.

Likewise, the Mobile 1.0 XHTML is a DTD used to describe code that's been developed for wireless display. Choose this form only if you're developing Web page content for wireless mobile devices:

```
<!DOCTYPE html PUBLIC "-//WAPFORUM//DTD XHTML Mobile 1.0//EN"
    "http://www.wapforum.org/DTD/xhtml1-mobile10.dtd"><html
    xmlns="http://www.w3.org/1999/xhtml">
```

There has been a lot of debate over the past few years about whether to switch from HTML to XHTML. Some designers believe that no reason exists to make the change, while others think that following the recommendations of the W3C is the right thing to do, even if all the benefits of XHTML haven't yet been fully realized. For an intelligent discussion on the topic, see www.sitepoint.com/forums/showthread.php?t=393445#q7.



All in all, when a DOCTYPE is specified in the head of an (X)HTML file, a browser can recognize the code and parse it more quickly. Additionally, the DTD helps ensure that your Web pages can be tested for accuracy by using an online markup validator like the one at <http://validator.w3.org>. (You find out more about testing, accessibility, compliance, and validation in Book IV, Chapter 2.)

Adding a DOCTYPE in Dreamweaver

If you're a Dreamweaver user, you almost don't have to think about the DTD because the program automatically inserts the selected DOCTYPE and DTD into the code each time you create a new document through the New Document dialog box. Of course, you have other ways to create a new file in Dreamweaver, and in those cases, the default DTD, as specified in the program's preferences, will be automatically inserted.

No matter which DTD you select, after it has been specified in the code, Dreamweaver automatically writes DTD-specific code. For example, if you choose to build pages using the XHTML 1.0 transitional DTD, Dreamweaver's code editor will automatically write XHTML-compliant code.

To select the appropriate DTD for your new documents in Dreamweaver, follow these steps:

1. **Launch Dreamweaver and choose File⇨New to open the New Document dialog box and select the Blank Page option, as shown in Figure 1-6.**

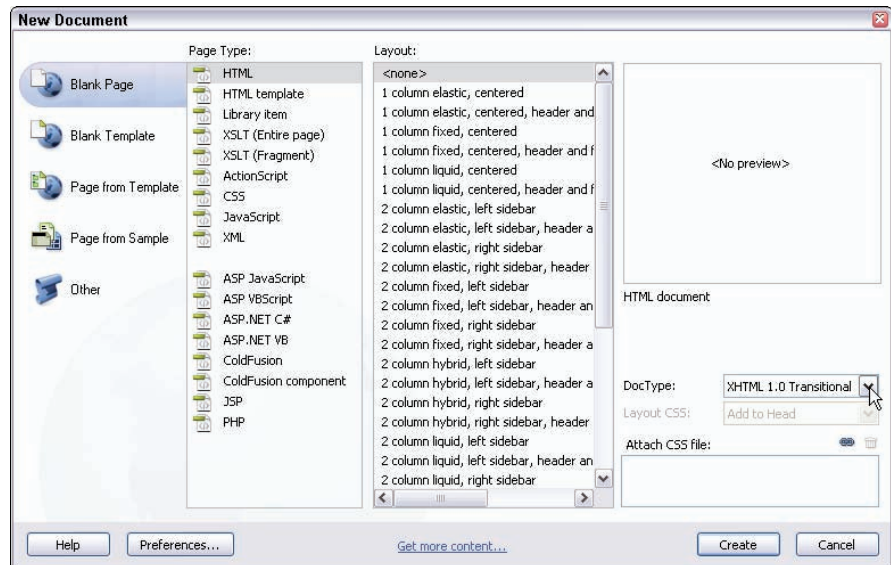


Figure 1-6: Use the New Document dialog box to select a DOCTYPE for your code.

2. Select the HTML option from the Page Type Category, and choose <none> from the Layout column.

This tells Dreamweaver to open a new blank HTML file.

3. In the DocType area in the lower-right corner of the dialog box, use the menu to select the desired document type (DTD).

The default DocType in Dreamweaver CS4 is XHTML 1.0 Transitional. Other options include None, HTML 4.01 Transitional, HTML 4.01 Strict, XHTML 1.0 Transitional, XHTML 1.0 Strict, XHTML 1.1, and XHTML Mobile. *Note:* The Frames HTML or Frames XHTML DTD will be automatically inserted into the page code when Frames are used.

4. Click the Create button to open the new, blank HTML or XHTML page in the Dreamweaver workspace.

If you check the code in Code view, you can see the selected DTD at the top of the page, above the opening HTML tag.



To modify the default DTD in your copy of Dreamweaver, open the Preferences dialog box by choosing Edit⇨Preferences (Windows) or Dreamweaver⇨Preferences (Mac), click the New Document category, and choose the desired Default Document Type from the DTD menu, as illustrated in Figure 1-7.



Older versions of Dreamweaver, including MX and earlier, either didn't provide an option to select the DTD or coded in only part of it automatically. If you're still using an old version of Dreamweaver or another code editor that doesn't include the DTD, be sure to hand-code the appropriate DTD into your pages or upgrade your software so that your program does it for you automatically.

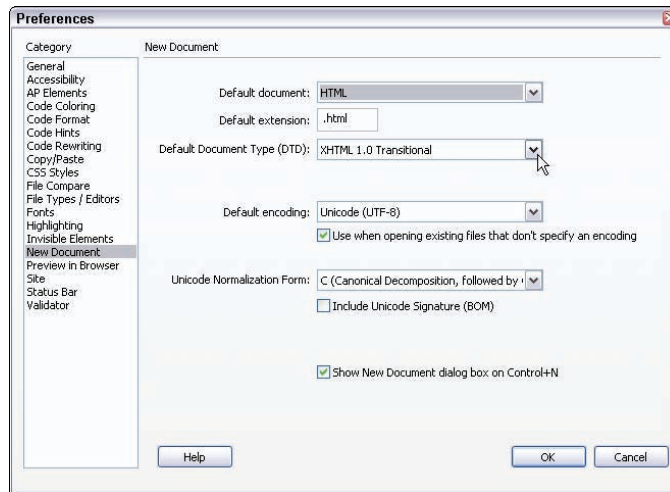


Figure 1-7: Set the default DTD in the Preferences dialog box.

Writing Semantic HTML and XHTML Code

Most of the best code editors these days — whether you're using a code-only editor or with a WYSIWYG view — automatically write code for you that conforms to HTML 4.01 or XHTML 1.0 standards based on the DTD you select when creating your pages. That's a very good thing, especially for HTML newbies. Where your code can start to unravel, however, is when you

“go rogue” by hand-coding, hand-editing, and using any free code and scripts that you find online. Another potentially complicating factor that may jeopardize the validity of your code can happen when you start working with legacy HTML pages (perhaps created by someone else a few years ago) that still use some of the older HTML tags that have been deprecated (phased out) over the past few years.

Whether you’re hand-coding or not, either with or without the help of an editor, one of the best insurance measures you can take to avoid any potential browser rendering and code validation issues is to know the rules that govern HTML 4.01 (and the rules of XHTML 1.0) so that you can write valid, semantic (X)HTML.



If you aren’t yet familiar with the rules of the HTML road, quite a few very good free online tutorials are available, such as those found at W3Schools and WebMonkey. Getting yourself a good HTML handbook is another smart thing you can do so that you always have a guide at your fingertips when questions arise. You can find several wonderful titles at www.wiley.com.

Regarding the differences between HTML and XHTML, and deciding which markup language to use for your Web pages, you might want to start coding your projects with HTML and then advance later to XHTML when you feel confident about the structure and usage of HTML. After you master using HTML, you can easily transition to XHTML code, at which point the stricter rules of XHTML might make more sense to you. Alternatively, if you’re feeling confident about the future extensibility of your site, you may decide to just jump right into the world of XHTML from the start. Whichever version you decide to use, make sure to use it consistently throughout your site. To help you do this, Dreamweaver users can customize the default DTD through the Preferences dialog box.

One benefit of starting with HTML is that some tags and attributes are backward compatible with many older browsers, whereas XHTML isn’t supported in part or in full by many of the older browsers (I’m talking old, old — like Netscape 4 and IE 4 and 5 old). In addition, programs like Dreamweaver and Expression Web will code properly syntaxed semantic HTML and XHTML, but the programs aren’t human, so you need to intervene occasionally to ensure that the code is properly formatted and remains that way anytime you make alterations to your code by hand.

Table 1-2 shows a side-by-side comparison of writing HTML versus XHTML code. Review the rules for both and then use the markup that best meets your needs.

Table 1-2 HTML and XHTML Markup Comparison	
HTML	XHTML
Code structure must be ordered correctly, but forgotten tags may be forgiven and cause a page to fail acceptably, such as forgetting to close the <code><title></code> or <code><head></code> tag.	All code elements must be closed and placed in the proper location hierarchically within the opening and closing <code><html></code> tags, as in <pre><html> <head>...</head> <body>...</body> </html></pre>
HTML files should have, but aren't required to have, an HTML DOCTYPE declaration above the opening <code><html></code> tag.	All XHTML files must include an XHTML DOCTYPE declaration above the opening <code><html></code> tag, as in <pre><!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/ xhtml1-transitional.dtd"> <head> <title>Add your title here</title> </head> <body> ... </body> </html></pre>
Tags can be written in either uppercase or lowercase, but lowercase is preferred, such as <code><title></code> instead of <code><TITLE></code> .	Tags must always be written in all lowercase letters, as in <code><head></code> , <code><body></code> , and <code><p></code> .
HTML cannot be an application of XML.	XHTML takes advantage of XML.
Tags and objects can be improperly nested with little consequence, so <code><i>Citizen Kane</i></code> would still be displayed in bold and italics.	Tags and objects must be properly nested to display accurately in a browser, so <code>Marshmallows</code> would be incorrect and <code>Marshmallows</code> would be correct.
Tags needn't always have closing tag elements, as with the <code><p></code> , <code><hr></code> , and <code>
</code> tags.	All tags and objects must be properly closed. Tags that didn't typically need to be closed in HTML should now be closed by placing a space and slash inside the tag, as in <code>
</code> , <code><hr /></code> , and <code></code> . All other tags, such as <code><p></code> , <code><td></code> , and <code></code> , must also be properly closed.

continued

Table 1-2 (continued)

<i>HTML</i>	<i>XHTML</i>
Values inside attributes of tags can be written either with or without quotation marks, as in <code></code> or <code></code> .	Values inside attributes of tags must be written inside quotation marks, as in <code><div align="center"></code> and <code><td width="145"></code> .
Attributes can use shorthand to minimize the code that needs to be written when the value matches the desired option, as with <code><input disabled></code> .	Attributes can no longer use shorthand and must always use the full syntax of the HTML code, as with <code><input disabled="disabled" /></code> .
Objects use the name attribute, as in <code></code> .	Objects use either the <code>id</code> attribute instead of name, as in <code></code> or use both the <code>id</code> and the name attributes together, as in <code></code> Using both name and id attributes together helps older browsers interpret and display HTML data.

After reviewing Table 1-2, it might appear that XHTML follows a much stricter set of rules than HTML. This is true, but XHTML is also the current recommended standard for the Web and has many benefits that HTML lacks. For instance, because of all its standards, some say XHTML is easier to learn than HTML. Not only that, but it's also easier to keep up and make changes to, plus it's both XML- and XSL-ready. (XSL [eXtensible Stylesheet Language] is similar to CSS but is used exclusively to define the presentation of content in XML files in a browser.)

Whichever markup language you select, you can test the validity of the code for compliance to the W3C Recommended standards using the techniques you find in Book IV, Chapter 2. Until then, as long as you tell your HTML editor to code in the desired markup language and pay attention to syntactical rules while making any code adjustments by hand, you should be able to code your pages properly.

Formatting with CSS Instead of HTML

Before CSS came along, all the presentation markup was added to the page using old HTML formatting tags. Today, although you can format your text in HTML, the standards-compliant way is to format your pages using CSS. When you use CSS, you can define site-wide styles for the tags used to mark up your content and create custom tags to selectively style page elements.

This is in stark contrast to the old way, where you have to apply a formatting tag to every block of text, graphic, and other objects that require particular formatting.

Comparing CSS and HTML formatting

In the context of a chapter dedicated to following Web standards, knowing the proper way to style your content is worthy of further explanation. To illustrate, I'll review the differences between old HTML formatting and formatting using CSS.

First the old way. Using the old formatting and font styling tags, each line, paragraph, and block of text needed to be surrounded by its own set of `` tags, which included styling information about the font's face, size, and color. Often, tags and tag attributes that have since been deprecated were also used to handle formatting for things like alignment, bold, and italics:

```
<p align="center"><font color="#FF3399" size="2" face="arial, "lucida console",
  sans-serif"><a href="meetourstaff.html"><i>Meet Our Staff</i></a></font></p>
```

Notice how much space the `font` tag and `center` alignment attribute in the `<p>` tag take up, relative to the amount of content. Now think about this: Each character and space in the formatting markup take up a fraction of a byte in file size, which means that when you add all the individual bits of code together, the styling markup can dramatically increase the size of an HTML file, thereby causing the page to load slower in a browser window.

Now take a look at the same content and code when styled using CSS instead, where the `<p>` tag style is redefined in the CSS:

```
<p><a href="meetourstaff.html">Meet Our Staff</a></p>
```

The styling and positioning for the `<p>` tag might look like this:

```
p {
  font-family: arial, "lucida console", sans-serif;
  font-size: 12px;
  font-style: italic;
  font-weight: bold;
  color: #ff3399;
  text-align: center;
}
```

Much simpler code! The alignment is now controlled by the `text-align: center` style for more consistent rendering in different browsers, the old `<i>` tag gets replaced by the `font-style: italic` attribute, and the rest of the font styling is handled by the remaining font rules within the style declaration. Perhaps best of all, one of the greatest advantages of CSS is the

reusability of the style; no matter where else on the page you have additional paragraphs, you automatically get this formatting rather than have to reapply all the font attributes to each `<p>` tag.

Taking a look at the benefits of CSS

The benefits of styling your HTML and XHTML content with CSS are vast:

- ✓ CSS is easy to use and easy to learn.
- ✓ CSS makes HTML pages smaller in file size, thereby speeding page download times.
- ✓ CSS is one of the W3C's core recommendations, so your CSS-styled site will comply with the current standards. Moreover, most HTML formatting tags are being deprecated by older browsers and aren't even supported in XHTML code, so you have no good reason not to use CSS for most, if not all, of your content styling and positioning needs.
- ✓ CSS is infinitely editable, giving you the flexibility of changing the look of your pages as often as you like without ever altering the content.
- ✓ CSS helps separate *presentation* (how the page looks) from *content* (what's on the page) by moving all the page-styling instructions into a centralized location. That location can either be in-line with the code or internal in the head area of the Web page, or in an external CSS document to which all the pages on a site are linked, the latter being the most useful method for working with CSS. The benefit of having an entire site's style information contained in a single external CSS file is that doing so allows for instant site-wide style updates.
- ✓ CSS styles your content *semantically*, which means that it requires fewer styles than the old HTML formatting tags. For example, CSS allows designers to redefine the presentation of content contained inside particular tags, such as automatically adding a particular color and font face to any content marked up with H3 tags or applying the same background color and border attributes to any tables on the site.
- ✓ CSS can be used to style the look of text, images, and objects as well as to position objects on a Web page. This feature alone drastically reduces the amount of code required to display objects on a page. For instance, objects contained in `<div>` tags can be absolutely positioned on a page with CSS. Before, to place something in an exact spot on a page required code hacks involving the use of tables with empty table cells and spacer gifs. All that extra code goes away with CSS. Other code hacks involve the unorthodox and creative use of HTML, CSS, JavaScript, and other code to manipulate objects on a Web page and/or work around existing limitations of the Web to achieve a desired visual effect.

- ✓ CSS is a more affordable solution for styling content because it takes less time to implement and update than the older styling techniques did. With the old way, even simple changes might require the hand-editing of all the individual pages on a site. With CSS, one change there can update a style across an entire Web site.

After you begin styling your content with external CSS, you'll probably never want to go back to using the old font tags for styling. In fact, if you inherit someone else's site and are hired to do a redesign, you can create new CSS styles for the content and strip the old formatting tags from all the existing pages on the site as part of the redesign project. You might even use this *font tags-to-CSS conversion* as a selling point of your services by explaining all the aforementioned benefits of CSS to your client. Not only should the client be impressed by the scope of your knowledge on the matter, but she might even feel somewhat inclined to hire you for her next redesign project sometime down the road if she ultimately likes your design and enjoys working with you this time.

Exploring pages styled with CSS

To further illustrate how wonderful it is to use CSS for semantically styling content, these steps show you some really amazing examples of how the same content can look completely different when styled with different CSS.

1. **Point your browser to** www.csszengarden.com.

What you see here is the home page of the site as styled with the CSS developed by CSS mastermind, Dave Shea.

2. **On the right side of the page, under the red Japanese Torii gate where it says Select a Design, click any of the links.**

Each link takes visitors to separate pages using the same page content but styled with CSS created by different designers. This listing is regularly updated, so depending on when you read this, the list you see may be different from the one you find a month or two from now. Regardless, each set of links shows how the same content can be styled with CSS to create a totally different look!

Figure 1-8 shows a comparison of the CSSZenGarden.com home page design (left) with one of the designs (right) available in the list at the time of publication. The HTML tags in the code are exactly the same on both pages. It's the CSS style definitions — which include fonts, colors, graphics, and positioning, among other things — that have been changed to achieve the new layout and look.

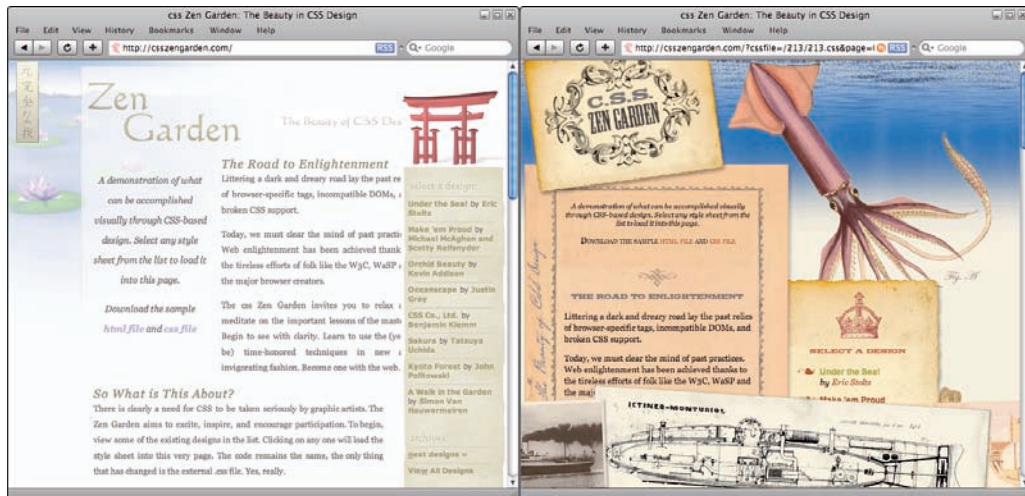


Figure 1-8: CSS magically transforms the appearance of the same content.

3. Scroll down the page until you see the Select a Design section and click another link.

Same content, different design! At this point, you should be totally convinced of the amazing power of CSS!

4. Click another link in the Select a Design column.

What makes this particular site extremely brilliant is that any designer who submits his or her design to CSSZenGarden.com agrees to make the CSS open source. This means anyone, including *you*, can go examine the CSS and find out how all the visual effects were created.

5. To view the CSS for any particular design, look for a Resources section and click the View This Design's CSS link.

When you click this link, the CSS for that page automatically opens in your browser window. Figure 1-9 shows the CSS for the design on the right side of Figure 1-8. In the CSS, you can see how the General Properties styles format the appearance of the page background and text links, and the Text Properties styles define the look of paragraphs and headers. Likewise, in the Div Properties section of the CSS, content contained in `<div>` tags with these named IDs (for example, `#preamble`) will be automatically styled by the style rules contained within them, such as the section on the page that contains preamble text, where it says “The Road to Enlightenment.”

```

/* css Zen Garden submission 213 - 'Under the Sea', by Eric Stoltz,
http://www.ericstoltz.com/ */
/* css released under Creative Commons License -
http://creativecommons.org/licenses/by-nc-sa/1.0/ */
/* All associated graphics copyright 2007, Eric Stoltz */

/* IMPORTANT */
/* This design is not a template. You may not reproduce it elsewhere without the
designer's written permission. However, feel free to study the CSS and use
techniques you learn from it elsewhere. */

/* "Under the Sea" by Eric Stoltz
December 20, 2007
www.ericstoltz.com */
/* css released under Creative Commons License -
http://creativecommons.org/licenses/by-nc-sa/1.0/ */

html {
margin: 0px;
padding: 0px;
height: 100%;
width: 100%;
}
body {
margin: 0px;
padding: 0px;
font-family: Georgia, "Times New Roman", Times, serif;
font-size: 0.9em;
height: 100%;
width: 100%;
background-image: url(water.png);
background-repeat: repeat-x;
background-position: top;
background-color: #F0E6D6;
}

```

Figure 1-9: View the CSS code to find out how designers created different effects.

6. To save a copy of the opened CSS to your local computer, choose File→Save or File→Save Page As.

The filename you give the CSS is up to you, so feel free to change it to whatever you want. What does matter is the extension; all CSS files must be saved with the .css file extension to work properly.

7. To see even more designs using different CSS, return to the main CSSZenGarden.com site and click the View All Designs link in the Archives section, which takes you to the CSS Zen Garden archive on Dave Shea's MezzoBlue site.

The CSSZenGarden.com Web site is a truly inspired idea with the ultimate goal of teaching you the benefits of styling with CSS.

Finding Out about Accessibility Standards

According to the W3C, people with disabilities who surf the Web make up nearly 10 percent of the Internet's population. Believe it or not, that's a larger percentage than the roughly 3.6 percent of all Internet consumers using the Mac OS (as opposed to Windows or Linux). Without a doubt, then,

people with disabilities make up a large enough group of potential Web site visitors that you should definitely pay considerable attention to them and their particular needs when designing Web sites.

To make your Web sites accessible, modify the code so that visitors with disabilities and search engine robots/spiders can access the information on the site's pages. Common coding enhancements include adding footer links, a site map page, alt text for images, page titles, meta tags, object labels, titles for links, link tags to the home and site map pages in the head, access and tab index keys, and form input labels.

One of the primary Web standards organizations with an online presence devoted to the subject of Web accessibility for people with disabilities is the Section 508 government site (www.section508.gov), shown in Figure 1-10. This organization is dedicated to compliance of Section 508 (29 U.S.C. 794d) of the Rehabilitation Act, especially with regard to the accessibility of Web sites to all people, whether employees of the federal government or not, including those with disabilities. Although their prescriptions for accessibility are technically only legally applicable to federal agencies using, developing, maintaining, and procuring information technology, many Web designers and developers are now informally broadening the scope of Section 508 to include access to any and all information that is readily available on the Web to anyone, with or without disabilities.

The screenshot shows the homepage of the Section 508 website. At the top left is the Section 508 logo with the URL www.section508.gov. To the right is a 'Site Layout Controls' panel with options to change the font (Verdana) and font size (12px), and an 'Enter' button. Below this is a navigation menu with links for '508 Law', '508 & You', '508 Training', '508 Coordinators', 'Accessibility Forum', 'FAQs', 'Events', and 'Help'. The main content area is divided into three columns. The left column has a search bar for 'Section508.gov' and a list of 'Additional Links' including 'Communications/Media', '508 Tools & Resources', 'AT Showcase', 'Emergency Planning', and 'Contact Us'. The middle column features a '508 News Release' section with a sub-heading 'January 2009 Assistive Technology Day' and a detailed paragraph about the event at the GSA. Below this is a section for 'The Interagency Disability Educational Awareness Showcase (IDEAS)'. The right column contains three red text boxes with links for 'Secure Login', 'Register For 508 Universe', and 'User Guidance'. At the bottom right is an 'Icon Key' with buttons for '508 Templates' and 'Save Item'.

Figure 1-10: The Section 508 Web site recommends guidelines for making Web pages accessible to visitors with disabilities.

The Section 508 amendment, which was passed in 1998, is often broken down into two parts: The first relates to HTML usage, and the second part deals specifically with JavaScripts, plug-ins, and other multimedia enhancements found on Web pages. Because you're more likely to focus your efforts on HTML compliance than on multimedia compliance, the following excerpt from www.section508.gov lists the standards from the HTML part (Web-based intranet and Internet information and applications) and is interspersed with notes on how you might meet each standard:

- (a) A text equivalent for every non-text element shall be provided (for example, via “alt”, “longdesc”, or in element content).**

When building pages for the Web, each page must necessarily pass or fail each of the standards as set forth in Section 508. For example, a graphic on a Web page will fail Standard (a) when the image is missing its alt text attribute description, as in ``, but will pass Standard (a) when the image contains a properly syntaxed alt description within the image tag, as in ``.

- (b) Equivalent alternatives for any multimedia presentation shall be synchronized with the presentation.**

An example equivalent might include the specification of a long description in the HTML code that describes the content in a multimedia presentation, as with this code example that plays an animated GIF:

```

```

- (c) Web pages shall be designed so that all information conveyed with color is also available without color, for example from context or markup.**

All Web pages must visually make sense, both with and without color style markup. Test your pages to see whether the removal of color changes the experience of visiting the site.

- (d) Documents shall be organized so they are readable without requiring an associated style sheet.**

When using CSS, try toggling the CSS on and off to see whether the ordering of the content makes sense without it, because most assistive devices ignore CSS and strictly read content from top to bottom.

- (e) Redundant text links shall be provided for each active region of a server-side image map.**

Server-side image maps are rarely used anymore compared to client-side image maps; however, if you must include these on your page, make sure to include text links for each region on the image map.

Server-side refers to any application or program that must run on a server rather than on a client machine to function properly, such as with Server-Side Includes (SSIs), whereas *client-side* refers to any code, application, or code processing that is performed on the client's computer rather than on a server, such as image maps that specify areas on a graphic that link to other pages on the site.

- (f) Client-side image maps shall be provided instead of server-side image maps except where the regions cannot be defined with an available geometric shape.**

Client-side image maps can usually accommodate any special active region shapes, so this shouldn't be much of an issue.

- (g) Row and column headers shall be identified for data tables.**

When data is displayed in tables and header information is also included, the appropriate `<th>` tags must be used instead of `<td>` tags to define the header rows/columns.

- (h) Markup shall be used to associate data cells and header cells for data tables that have two or more logical levels of row or column headers.**

Use `<th>` tags instead of `<td>` tags to define table cells used as header cells.

- (i) Frames shall be titled with text that facilitates frame identification and navigation.**

Though using frames is highly discouraged from an accessibility standpoint, when they are used to present multiple pages in a single browser window, each page that displays in a frame must contain its own title tag, and each frame must have an appropriate frame name.

- (j) Pages shall be designed to avoid causing the screen to flicker with a frequency greater than 2 Hz and lower than 55 Hz.**

In layman's terms, don't add any animations to your pages with a super-fast flicker rate because certain frequencies can trigger seizures in visitors with a particular kind of epilepsy.

- (k) A text-only page, with equivalent information or functionality, shall be provided to make a Web site comply with the provisions of this part, when compliance cannot be accomplished in any other way. The content of the text-only page shall be updated whenever the primary page changes.**

Anytime content cannot comply with accessibility guidelines, the URL to an alternate text-only page that contains instructions or information about the noncompliant content must be specified in the code.

- (l) When pages utilize scripting languages to display content, or to create interface elements, the information provided by the script shall be identified with functional (understandable) text that can be read by assistive technology.**

For instance, if your site uses JavaScript to create a rollover effect for all the main navigation links, the code should contain attending `<noscript>` tags that provide the visitor with information about the script's function as well as links to any pages that the script provides access to.

Assistive technology refers to any device (such as a screen reader) or nonhuman application (such as a search engine robot) that accesses content on the Internet through means other than a Web browser.

- (m) When a Web page requires that an applet, plug-in or other application be present on the client system to interpret page content, the page must provide a link to a plug-in or applet that complies with 1194.21(a) through (l).**

You should do this for all applications and devices that interpret content; always provide a link for your visitors to download any necessary plug-ins that are needed to view page content.

- (n) When electronic forms are designed to be completed online, the form shall allow people using assistive technology to access the information, field elements, and functionality required for completion and submission of the form, including all directions and cues.**

This means adding labels, coding access keys and tab order attributes, and including other form accessibility tags and attributes to all the form fields.

- (o) A method shall be provided that permits users to skip repetitive navigation links.**

Use anchor links combined with a tab index to allow visitors to skip repeating navigation links. For example, you may want to make the first link in a nav bar called Skip Navigation combined with an anchor link that always takes the visitor to the first line of text on the page.

```
<a href="#start">Skip Navigation</a>
<p><a name="start" tabindex="1">Welcome</a></p>
```

- (p) When a timed response is required from the visitor, the user shall be alerted and given sufficient time to indicate more time is required.**

Do not use meta tags, JavaScript, or any other kind of programming to make the page refresh or forward to another page without also providing visitors with one or more alternate ways to adjust the timing and/or access the other information.

508 *Finding Out about Accessibility Standards*

To view all the Section 508 standards, including technical standards (1194.21 Software applications and operating systems), visit

www.section508.gov/index.cfm?FuseAction=Content&ID=12/#Web

For an informative and comprehensive look at what it means to pass or fail each standard, visit WebAim's 508 checklist:

www.webaim.org/standards/508/checklist.php

The W3C also offers accessibility checkpoints at

www.w3.org/TR/WAI-WEBCONTENT/full-checklist.html

Additional extremely helpful information on the topic can be obtained at the following Web sites:

www.diveintoaccessibility.org
www.joeclark.org/book/sashay/serialization
www.alistapart.com/articles/wiwa



The benefits of designing for accessibility don't stop with making sites more accessible to those with disabilities. By following accessibility guidelines, the content on your pages can be more easily accessed using a larger group of lesser known (but no less wonderful) Web browsers, such as Opera, Mozilla, Lynx, and Amaya. The more devices that can access a Section 508-compliant site, the greater the likelihood of increased visitor traffic and potentially increased sales of products and services.

Chapter 2: Testing, Accessibility, Compliance, and Validation

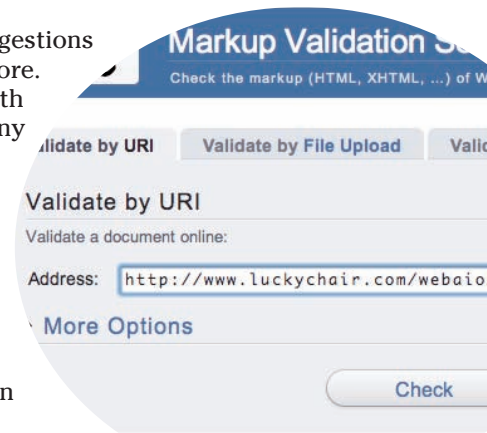
In This Chapter

- ✓ Checking your code for errors
- ✓ Testing on different platforms and browsers
- ✓ Fixing common code errors
- ✓ Checking (X)HTML syntax
- ✓ Making pages CSS, (X)HTML, and 508 accessibility compliant
- ✓ Fixing noncompliant code issues
- ✓ Displaying proof of code validation

Congratulations on making it this far! You're almost to the finish line, and you only have a few more things to do before you can publish your site for all to see. At this stage, it's time to put all the pages on your site through a rigorous review to catch potential problems like spelling errors, code issues, broken links, and missing code attributes like `alt` text attributes for images and `title` attributes for hyperlinks.

In this chapter, you'll find some helpful tips and suggestions on validation, testing, standards compliancy, and more. Most HTML code editors have tools to assist you with testing your pages so that you can identify and fix any problems before visitors have a chance to see the site. For instance, in this chapter, you find out how to use several testing tools, including ones that clean up code and check spelling, and I discuss a tool to find and replace text and source code throughout a site. In addition, you discover how to clean up redundant and unnecessary code, apply uniform source formatting to pages, and fix some common coding problems such as identifying broken links and orphans.

A few examples in this chapter use Dreamweaver. However, if you are using another program you should be able to find many similar tools. For details about any specific tools, be sure to consult your application's Help guide.



the markup validity of W

Understanding the Process of Validating Your Code

When you take a patient and careful look at everything on your site, you can usually find and fix any display and functionality issues before your pages go online. That is not to say that the cleanup and validation process is a breeze. On the contrary, site testing can be a demanding task at times. In fact, it is quite common to find new coding issues at each phase of the validation and compliance testing process.

More than likely, however, what you will find is that performing the validation testing is relatively easy, and though it may take you some time to research, retest, and fix any problems you find during the validation process, the whole experience will enrich you and refine your skills, ultimately making you a better designer for any future projects.

Validation offers many important benefits to both you and your site's visitors:

- ✓ Validation makes a site more accessible to the widest audience, which can translate into increased visitor traffic and potentially more sales of products and services.
- ✓ Validated Web pages display faster in a browser window.
- ✓ Validated, clean code improves search engine accessibility and search engine results rankings.
- ✓ Validated pages are much easier to maintain and update.
- ✓ A well-validated site can be your calling card for obtaining future business.

To help you keep track of your progress, the validation process itself can be performed methodically over the course of several days. This chapter walks you through each of the steps. You begin by converting all the syntax on every page to match the specified DTD in the code. Then, you perform (X)HTML, CSS, and accessibility testing on every page. If you find any errors, you spend time fixing them and then retest to ensure that you've either fixed everything or that problematic code fails acceptably. After that, you're done. As a final optional step, you can display some kind of proof of validation on your site to show the world that you cared enough to spend the time doing the testing in the first place.

Performing Prelaunch Testing

Building a Web site is one of those activities that requires you to remember a lot of little details. Besides designing the mock-up, optimizing all the graphics, creating a template, building the individual pages, pasting in all

the relevant content in every page, dropping in JavaScript and media files, and formatting everything with CSS, you have to remember to include meta tags in the code, add customized titles to every page, give all your images alternate text attributes, and add `target` and `title` attributes to your links, especially any that will open in separate browser windows. The list of tasks could go on and on. That's where good organization comes in handy.

In the following sections, you discover how to create your own Web-testing checklist so that you can remember all the items on the site that you need to test. Additionally, you find out how to choose which platforms and browsers to test.

Creating a Web-testing checklist

To help you remember all the tasks you need to do, consider creating your own Web-testing checklist, like the starter list example shown here, and use it when reviewing every site you build prior to publishing:

- ✓ Have you performed a spelling and syntax check, included the correct DTD, applied source formatting, organized and commented your CSS, and cleaned up any HTML and Word HTML coding errors?
- ✓ Have you tested all the pages on your site in multiple browsers and browser versions on Mac, PC, and Linux platforms and found solutions for any glaring errors?
- ✓ Do all the pages on the site include appropriate meta tags?
- ✓ Does each page have a unique, descriptive title?
- ✓ Do all the images on the site include descriptive `alt` text attributes, including empty `alt` text attributes for decorative images that don't need `alt` text descriptions?
- ✓ Do all the hyperlinks include `title` and `target` attributes?
- ✓ Do links that will open in a separate browser window provide some kind of indicator to visitors that the link will open a new browser window?
- ✓ Have you hand-checked all the internal and external hyperlinks for accuracy? Do the links go where you want them to go? Did you find any broken links that need fixing?
- ✓ Are the site's forms and tables fully accessible and functional?
- ✓ Does the site have any unused files, images, or folders that can be safely moved to another location or deleted?



For a comprehensive checklist that covers everything from code quality and accessibility to basic usability and site management, visit www.maxdesign.com.au/presentation/checklist.htm.

Testing on multiple platforms, browsers, and devices

Possibly the most effective method for ensuring that your pages look and function the way you want them to is to test them in as many different scenarios as possible. For instance, because viewing pages in a browser is the best way to evaluate how your site's pages will look to visitors, the wider the set of browsers, browser versions, and operating systems (Windows XP, Vista, OS X, and so on) you can test in, the greater the likelihood you will find any hidden mistakes in the code and fix them before you launch your site.

Browser testing should be done regularly throughout the master page and site-building process and again at the end of the project prior to publication. Presuming that you've been previewing your pages in one or two browsers (most new designers often mistakenly only test in just Internet Explorer and Firefox on a PC) during the building phase of the site, you've probably already dealt with some of the more glaring display issues. However, at this stage, the focus is on how well the site displays in different browsers and in different operating systems.



According to the statisticians at W3Schools.com (shown in Figure 2-1), as of January 2009, 89.5 percent of all Internet users use PCs running some version of Windows; 5.8 percent of all Internet traffic comes from Mac OS users; and another 3.9 percent comes directly from Linux OS users. The remaining percentage of Web users experience the Internet with alternative tools, such as handheld devices, WebTV, and text-only assistive devices. These statistics alone should tell you that to be fully thorough in your testing, you must simulate how all visitors experience the site and correct any glaring mistakes prior to publishing.

Deciding which platforms and browsers to test

Realistically, testing in every possible browser scenario isn't really an affordable solution for the average Web designer. At a minimum, try to test on both Mac and PC in the most popular browsers and browser versions, which, at the time of this writing, include the following:

- ✓ **PC:** IE 6–8, Firefox 3.0, Chrome 1.0, Safari 3.2, and Opera 9.6
- ✓ **Mac:** Safari 3.2, Firefox 3.0, and Opera 9.6

Sure, this list still seems like a big one, but the time it takes for you to download, install, and test on these browsers will be well worth the effort. The main reason to do all this testing is that some versions of the same browser can display your pages drastically differently from other versions,

depending on the platform. A case in point is the IE 6 browser. On a PC, IE 6 works pretty well, but no Mac equivalent exists because the IE developers chose to abandon the Mac platform altogether. IE 6 and IE 7 on a PC also present different display issues that must be dealt with until the majority of IE users shift from IE 6 to IE 7 and IE 8. Likewise, Safari browsers often render pages — especially text — differently than IE and Firefox do. The sooner you understand the different display requirements of each browser, the sooner you can understand how to code your pages for nearly any browsing environment.



Because most Internet traffic comes from PC users, one useful strategy would be to test your pages in a bunch of browsers on a PC first before testing everything again on several Mac browsers. Many times — though not always — any display issues that you fix for your PC browsers will be resolved by the time you begin testing on a Mac.

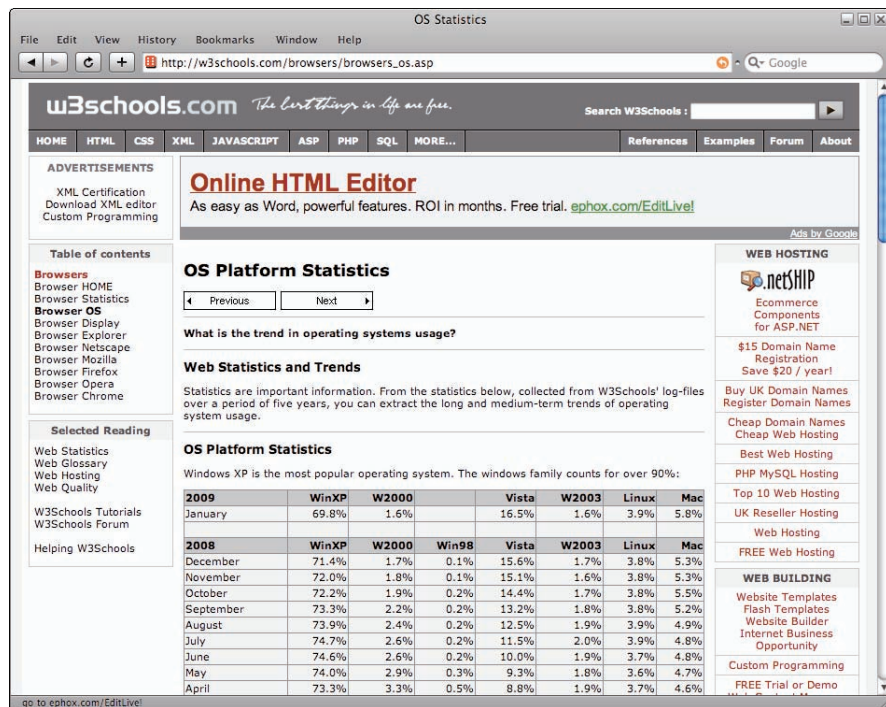


Figure 2-1: Most Internet users access the Web on a PC running one of the Windows operating systems.



Don't forget about other operating systems besides the PC and Mac. Linux is the third most popular OS, and with 3.8 percent of the Internet population using it, you should test your pages in a Linux environment too. To do that, get the *Lynx Viewer*, which is a text-only browser simulator that you can download for free at Delorie.com. This browser simulator identifies how visitors using text-only browsers will see your site so that you can correct any issues that those visitors might experience. To use the free simulator, designers must register on the Delorie site (www.delorie.com/web/lynxview.html) and install a test file on the hosting server before the simulator can function. Delorie also offers a WebTV simulator (www.delorie.com/web/wpbcv.html) for visitors who use legacy WebTV units or new WebTV users who use the updated MSN TV boxes.



For the best, most thorough results of your testing, consider uploading your site to a test directory on the same server that will be hosting the site. Having the pages on the destination server can help you uncover unforeseen server issues before the site gets published.

Testing a page

For a testing strategy, try this: For each Web page you review, ask yourself whether the page looks and functions the way you intended it to. If you need to resolve any display or functionality issues, go back into the code and make the appropriate changes:

- ✓ Review the home page first in every browser on a PC and a Mac.
- ✓ Review any pages on the site that include HTML forms and verify that the forms correctly process the data.
- ✓ Review any pages that contain dynamic data and test them to ensure that the information is displaying correctly.
- ✓ Review the rest of the pages on the site and check the functionality of any interactive elements that use JavaScript or other scripts, programming languages, or applications, including any rollover buttons, links, and multimedia files.

Finding problems is one thing, but fixing those problems can be quite something else. When you identify problems with how the pages appear, exactly how to correct those issues might be a time-consuming matter of trial and error. In some instances, the problem will be easy to correct, as with a glaring coding mistake. However, in other instances, finding a way to make the page look right might involve several different solutions. Be patient and persistent, and remember that the more mistakes you make and correct, the better designer you'll be.

Using third-party testing tools

In addition to testing the pages yourself, other tools may be helpful. For instance, several third-party software tools (like the fee-based services at BrowserCam.com) can assist you with your cross-platform/cross-browser testing.

One fairly useful testing tool is the free Web Site Viewer at AnyBrowser.com, which can help you identify coding issues based on slightly modified HTML 3.2 coding specifications, regardless of whether you're testing pages on a Mac or a PC.

To test your pages with the Web Site Viewer at AnyBrowser.com, follow these steps:

1. Point your Web browser to `http://anybrowser.com/siteviewer.html` and enter the URL of the page online that you would like to test.

The test page must be uploaded to a live server before it can be tested. To protect the privacy of your pages before publishing

them for visitors to see, you may want to upload the entire site to a test directory and then enter the complete URL to that location, such as `http://www.mywebsite.com/test/index.html`.

2. Click the View Page button.

The browser window refreshes and displays the URL of the page you entered. If you see any display errors that simply can't be ignored, go back into the code, fix the problems, and reupload your test pages to the test directory.

3. Repeat Steps 1 and 2 for the rest of the pages on the site and for any pages that require code correction.

If desired, you can also configure the Viewer tool to provide testing results based on different HTML levels, such as HTML 4.0 Transitional and Strict syntax rules.

Cleaning Up Your Code

Whether you're hand-coding or working with a code editor, you must always make a practice of checking the hierarchical order, syntax, and spelling of your HTML, CSS, JavaScript, and other programming code. Even when you use the best programs available, at some point, your code could include errors because you are human. No matter how carefully you build a site, the HTML in your pages will probably inadvertently become cluttered with redundant tags, unnecessary markup, and outright detrimental code that can negatively impact the presentation and functionality of your Web pages.

A lot of these errors often happen when you paste content onto your pages from other sources like a Word or Excel file, a Web site, or another code editor or application. Other errors might happen from an honest typo or from coding mistakes that happened behind the scenes when moving elements around the page by clicking and dragging or cutting and pasting.

One very common code problem is finding empty hyperlinks on your page, which some code editors occasionally leave in the code for no apparent reason when you move a link from one spot on the page to another. These empty links are essentially just bits of code that don't surround any content, such as ``, which means they won't appear in the browser. As is often the case, such extra code might not be noticeable during the building phase of your site, but it will often rear its head during the testing phase prior to publishing.



To minimize coding errors on your Web pages, be sure to perform code cleanup at least twice during the building phase:

1. After building your master pages or templates, clean up the code before building out any of the remaining pages for the site.
2. Do another round of cleanup on all the pages of the site at the end of the building phase, prior to site launch.

To help you with the cleanup process, take advantage of any special tools your coding application may recommend for this task. Dreamweaver, for example, offers several tools that can automate the cleanup process within any managed site, such as a Find and Replace tool, a Spell Checker, a Word Clean Up command, a Code Clean Up command, a Source Formatting tool, and a Convert Syntax tool. Although many of the examples you find here include step-by-step instructions specifically for Dreamweaver, most other coding applications may have similar tools.

In addition, you can find quite a few online tools to check your pages for certain errors and help you identify those problems in the code. The only drawback to online tools, however, is that many of them allow you to input only one URL at a time, which can make the cleanup process rather tedious.

Finding and replacing errors

By far, one of the most robust tools you'll find in most HTML coding editors is the Find and Replace tool. With this one tool, you can search for and optionally replace any text, source code, or specific tag throughout an entire page, selected pages, a specified folder, or an entire managed site. Suppose, for example, you realize that the content provided by the client contains a certain word that is misspelled throughout the entire site. This tool can find that misspelling and replace all instances with the correct spelling on all the site pages. Pretty powerful stuff!



The only thing about this kind of tool that might be a deterrent to some designers is the fact that these kinds of global operations typically cannot be undone, which can be especially critical when modifying pages site-wide. Therefore, to ensure that you can revert innocent mistakes should they happen, make a backup copy of your entire managed site *before* using this

tool. That way, in the event that something does goes awry, you can always revert your site to the state it was in before you began making changes. After a few weeks, when you know you will no longer need to revert to that particular state of the site, you can safely delete the backup files.

To find a misspelled word or phrase and replace that text with new text throughout all the pages in a managed Dreamweaver site, open the document that contains the spelling error and select the misspelled word or phrase. Then choose Edit⇨Find and Replace to open the Find and Replace dialog box, as shown in Figure 2-2.

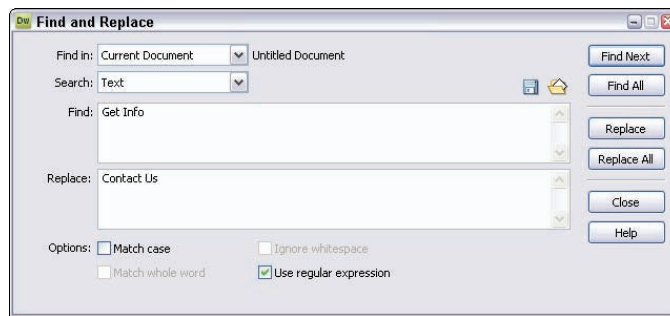


Figure 2-2: The Find and Replace tool helps you fix coding errors.



Changing words and phrases is only one example of how to use a tool this powerful. You can also use the Find and Replace tool to edit whole chunks of code, replace filenames throughout the site, and remove tags and other unwanted coding like comments and other unnecessary markup.

Checking spelling

Notwithstanding you and your client's best efforts, spelling errors will inevitably happen. Sometimes they take the form of regular typos and grammatical errors, like *htis* instead of *this* and *they're* instead of *their* or *there*. Other times, typos come in the form of commonly misused words, such as *accept* and *except*. Occasionally your spelling errors might even come in the form of accurately spelled words that are contextually wrong, like typing *moon* when you meant *noon*.

Possibly the best defense against most spelling errors is a two-pronged approach. First, check the spelling for errors using your own eyes and a spell checker. Second, enlist a group of volunteers, possibly even including the client, to read the site content in search of any spelling errors and grammatical mistakes prior to publishing. The more people who help with this

task, the greater the likelihood that you'll find and fix all the errors. With luck, you may even discover some other site issues that need repair, such as broken links and missing images.



Checking the spelling of all the content on a site is quite a big task. If you'd rather not be responsible for the accuracy of the site's spelling — especially because clients often provide content that is rife with spelling errors! — you can include a stipulation in your design contract that states that the client is ultimately responsible for the accuracy and substance of the content. In any case, consider running a spell check to correct any obvious mistakes that you may have made while working on the pages, such as *the* misspelled as *teh*.

To help you with the spell checking on your site, be sure to take advantage of any automated spell checking commands that come as a standard feature of your HTML editor. Dreamweaver has a decent spell checker that works very similarly to the one found in Microsoft Word. To open the spell checker, choose **Commands** ⇨ **Check Spelling**, as shown in Figure 2-3.



Figure 2-3: Dreamweaver's Check Spelling command is similar to the spell checker in Microsoft Word.

Removing unwanted formatting

Most code editors have a command that automatically cleans up the common errors in your code. For instance, the **Clean Up Word HTML/XHTML** command in Dreamweaver is a must for any Web page that includes content that was copied from Word or any other Microsoft documents. The reason this is important is because Microsoft files often embed extra markup when pasted into a Web page to make the file's content retain its formatting. Unfortunately on the Web, all this extra Microsoft HTML code is unnecessary. Therefore, you must use a tool like this to strip that extra markup out of your pages to ensure that your code — not Microsoft's — dictates the look of the site within the browser.

To access the **Clean Up Word** feature in any open document in Dreamweaver, follow these steps:

1. Choose **Commands ⇨ **Clean Up Word HTML**.**

The **Clean Up Word HTML** dialog box opens, as shown in Figure 2-4.

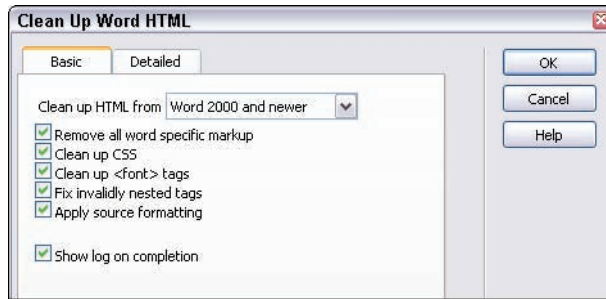


Figure 2-4: Cleaning up unnecessary markup copied from Microsoft documents.

2. Enable or disable the cleanup settings as desired on both the Basic and Detailed tabs of the dialog box.

By default, all the options on both tabs are automatically enabled to provide the most robust form of cleanup. Here's an overview of the options on the Basic tab:

- **Remove All Word Specific Markup:** This setting deletes any special markup that is required to format the page in Word and Word HTML files but is unnecessary in a normal HTML file.
- **Clean Up CSS:** Enable this option to delete any Microsoft Word-specific CSS markup, such as `<p class="MSO Normal" style="Margin-Top:3em">`.
- **Clean Up Tags:** This setting deletes any instances of old `` tags for styling.
- **Fix Invalidly Nested Tags:** Word sometimes adds markup to a page outside normal heading and paragraph tags, which don't conform to valid tag-nesting standards. This option removes those tags.
- **Apply Source Formatting:** Source formatting is determined by the options specified in Dreamweaver's `SourceFormat.txt` file as well as the Code Format settings in the Preferences panel. Leave this one enabled.
- **Show Log on Completion:** Select this check box to see a summary of the cleanup results.

3. Click the OK button to run the Clean Up operation.

Upon completion of the cleanup process, Dreamweaver displays an alert box, like the one shown in Figure 2-5, with details about the cleanup.

Applying consistent (X)HTML syntax

Another useful tool that most HTML editors should have is one that will clean up the HTML and fix any inconsistent syntax rules. Dreamweaver, for instance, has the Clean Up HTML/XHTML command. This tool looks for problematic code within an open document and automatically cleans up any errors that are found. This tool is especially helpful in ensuring that your code uses consistent markup syntax.

Inconsistent syntax can sometimes happen when Dreamweaver codes in one markup language (XHTML) and you code in another (HTML).

Most Clean Up HTML/XHTML commands take their cue from the stated Document Type Definition (DTD) in the open document and convert any tags and syntax that are inconsistent to the proper format.

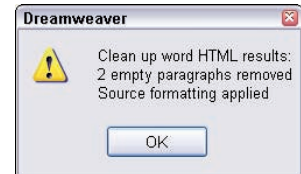


Figure 2-5: The cleanup alert message.

To illustrate how this works, follow these steps to apply this command to an open document within the Dreamweaver workspace:

1. Choose **Commands** ⇨ **Clean Up HTML** or **Commands** ⇨ **Clean Up XHTML**.

Dreamweaver automatically recognizes the DTD in the page code and displays the HTML or XHTML Clean Up command in the **Commands** menu to match that DTD.

The Clean Up HTML/XHTML dialog box opens, as shown in Figure 2-6.

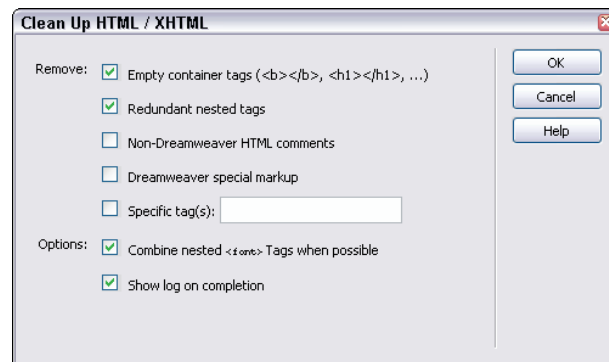


Figure 2-6: Clean your code of common syntax errors.

2. In the Clean Up HTML/XHTML dialog box that opens, adjust the cleanup settings.

By default, a few of the options are automatically enabled, but you may disable them and configure the rest of the settings to suit particular cleanup needs. For instance, you could use the Specific Tag(s) field to clean up empty `` tags.

To view cleanup results after running the command, be sure to enable the Show Log on Completion option.

3. Click the OK button.

When the Show Log on Completion option has been enabled, a small alert box appears, listing a helpful Clean Up summary.

Applying source formatting

In addition to allowing users to choose specific source formatting options for all newly created documents, another helpful tool that your code editor may have is one that will apply *source formatting*. This is a way of telling your code editor to automatically format pages in a particular way and to clean up any code that may not conform to the DTD you are using. For example, in Dreamweaver, you can adjust some of the settings in the program's Code Hints Preferences, which instructs the software to automatically create closing tags around selected content after typing `</` at the end.

Another fantastic Dreamweaver feature (which your code editor may also have) is the Apply Source Formatting option. This command applies the formatting options to the code of your page as specified in Dreamweaver's HTML format preferences and in the application's `SourceFormat.txt` file (itself an editable document that helps Dreamweaver determine how to code new files), thereby overwriting any coding inconsistencies that might have come about since the file was originally created.

To apply source formatting to any open document in Dreamweaver, choose **Commands** ⇨ **Apply Source Formatting**. Dreamweaver then applies the settings in the Code Format preferences to HTML, automatically updating any code that didn't match those preferences.

If desired, you can also apply the source formatting to a selection of contiguous code instead of an entire document.

Converting syntax by DTD

When most designers select a DTD for their Web docs, they have all the intention in the world of writing code that's compliant with that DTD's syntactical rules. Sometimes, however, with all the cutting and pasting and inserting and hand-coding, the syntax gets out of whack here and there, and when the time comes to test pages, you might find errors that cause the pages to not display exactly as you intended.

This is especially true when designers make the shift to using an XHTML DTD but are still in the habit of hand-coding tags in HTML syntax. One of the easiest mistakes to make in this regard is forgetting to add the extra space and slash for certain tags in XHTML when their HTML counterparts don't require it, like forgetting to write `
` instead of `
`.

If you happen to be a Dreamweaver user, you're in luck because Dreamweaver has a sweet little Convert Syntax tool that automatically converts all the code in a single document to conform with the syntactical rules of any selected DTD. If you are using a different code editor, check that editor's help files to see if you can find a similar command.

Using Dreamweaver's Convert Syntax command, you can change all the HTML syntax into XHTML. In other words, the code on any page — regardless of the original DTD and syntax used to code the page — can be automatically adjusted to match the syntax for any of the following DTDs:

- ✓ HTML 4.01 Transitional
- ✓ HTML 4.01 Strict
- ✓ XHTML 1.0 Transitional
- ✓ XHTML 1.0 Strict
- ✓ XHTML 1.1
- ✓ XHTML Mobile 1.0
- ✓ XSLT 1.0

To illustrate how this tool works, take a look at Listings 2-1 and 2-2, which show an example of how the Convert Syntax tool converts a page using HTML into the proper syntax for the XHTML 1.0 DTD.

Listing 2-1: Before Syntax Conversion (HTML)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Untitled Document</title>
</head>
<body><p>It's easy to change the syntax with<br>
the Syntax Conversion Tool</p>
<hr>
</body>
</html>
```

Listing 2-2: After Syntax Conversion (XHTML)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Untitled Document</title>
</head>
<body><p>It's easy to change the syntax with<br />
    the Syntax Conversion Tool</p>
<hr />
</body>
</html>
```

To convert the syntax of any open document in Dreamweaver — which includes having Dreamweaver automatically insert the selected DTD when one isn't detected or overwrite any existing DTD — follow these simple steps:

- 1. With your document open in the Dreamweaver workspace, choose File⇨Convert and select the desired DTD from the submenu, as shown in Figure 2-7.**

Dreamweaver immediately updates the DTD in the open file to match the one you just selected and converts any tags within the code of the page to match that DTD's syntax rules.

- 2. Save the file and repeat this process on any additional open files to ensure that all the files within the same managed site use the same DTD and coding format.**

Currently this tool does not perform any site-wide conversion; therefore, be sure to apply this Convert command to each document in your managed site.

If you happen to be working on a site that uses templates, feel free to apply this tool to normal documents, templates, and template-based files. Any changes made to the template should be automatically applied to any template-based pages upon save. However, be sure to also apply the Convert Syntax command individually to all the template-based files on the site so that the content inside any editable regions of the template-based files will also be converted to match the selected DTD.

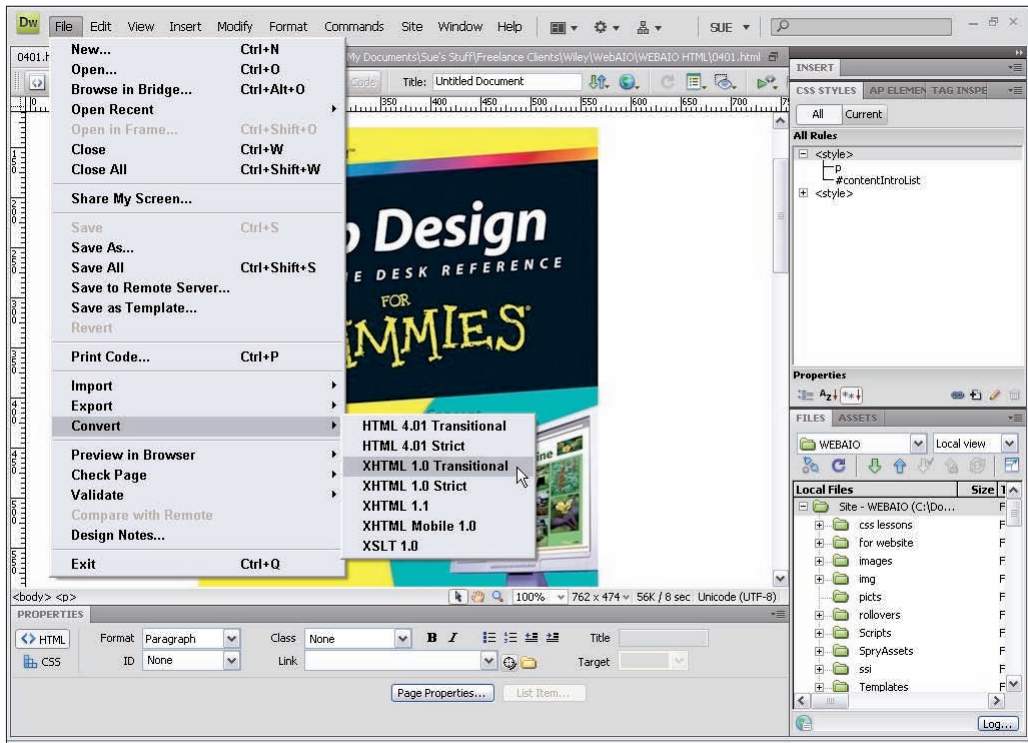


Figure 2-7: Use the Convert command to update a page's code syntax by DTD.

Fixing Common Code Errors

After you're done with the basic code cleanup, you can perform a handful of minitests to check your code's accuracy and functionality. Most of these tests help you check for things that need remembering but are easily forgotten, such as verifying that all the images include `alt` text attributes and all the links work properly, so checking for them at the end of the site-building phase makes good sense.

Although a lot of these tests can be automated for you when working in an HTML code editor, some must still be performed by hand. For those instances, using a methodical system of some kind can help you tackle the task in a more manageable way:

- ✓ **Checking page titles:** To check whether all the pages on a site include unique titles, open each page one at a time and verify that the title has

been entered. Untitled pages will display *Untitled Document* between the title tags. Alternatively, you could run a Dreamweaver report instead to check specifically for untitled documents, but the hand-check is a more thorough method of verification because you might find typos that the report simply wouldn't identify.

- ✓ **Checking meta tags:** To verify that the desired meta-tag information has been added to the head area of each file's code, take a look at the template file(s). When you modify a template, all the template-based pages will be automatically updated with new information. For non-template-based pages, check each page one at a time. To streamline the process of adding or updating meta tag content, copy the updated content to the computer's Clipboard and paste the updated meta information into the `<head>` area of each file's code. Save everything and close the files.
- ✓ **Using built-in testing tools and reports:** Check to see whether your code editor offers any kind of built-in testing tools and reports. Oftentimes these tools will be grouped together in some kind of panel. For instance, you can open Dreamweaver's Results panel by choosing Window⇨Results or by pressing F7. The Results panel contains eight different tabs, each of which provides access to special types of site information and testing tools. Four of them in particular are extremely useful for prelaunch testing: Validation, Browser Compatibility, Link Checker, and Site Reports.

In the case of Dreamweaver, each of the tools in the Results panel work in a similar way:

- ✓ To activate a tool, click the green Play button at the panel's upper-left edge.
- ✓ The selected task can be performed on the currently open file, all open files, a specified folder, selected files in the Files panel, or on all the files in the currently managed local site.
- ✓ Each tool can be customized to meet specific testing needs.

After running each tool's function, the tool's results are displayed in the Results panel window. Results are often listed by filename first, then by the line of code containing the issue in question, followed by a description and any other pertinent details. Each issue identified also displays with one of three icons next to each line — Error, Warning, or Message — as shown in Figure 2-8. Errors should always be addressed and corrected, warnings need to be looked into but might not cause serious display problems, and messages typically identify code issues that although incorrect, might not affect how the page displays.

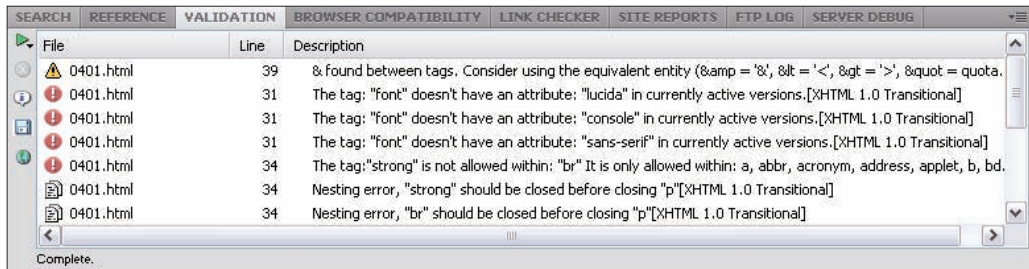


Figure 2-8: Validation results — errors, warnings, or messages.

For each problem identified in the code, you can right-click (Windows) or Control+click (Mac) the results line to access a contextual menu from which you can select further options, such as requesting more information about a particular issue. What's more, when you click any error in the Results panel, Dreamweaver takes you directly to the page that contains the error and highlights the exact line(s) of code that contains the error, making it easy for you to identify and correct.

Read on to find out more about the powerful Validator, Target Browser Check, Link Checker, and Site Reports features in Dreamweaver.

Validating your markup

No matter what code editor you happen to use, you absolutely must use a built-in or standalone validation tool to check the accuracy of your code. Most code editors have some kind of feature that allows you to validate code in the currently open file, a series of selected files, or the specified managed site, and you should be able to use this tool to validate a number of different markup languages, including HTML, XHTML, JSP, CFML, XML, and WML. If your code editor does not include a code validator, use an online validator. You find out about online code validation in the later section, "Validating HTML and CSS Markup."

To illustrate how to use a built-in validator, the following steps show you how to run Dreamweaver's Validator on a single open HTML file:

- 1. From the Validation tab in the Results panel, click the green arrow button in the upper-left corner and select Validate Current Document.**

To validate the whole site or selected files, choose the appropriate option from the pop-up menu.

Dreamweaver automatically runs the report and displays any problematic results in the bottom part of the Results panel.

- 2. To view or correct any of the errors, warnings, or messages listed in the results area, double-click the filename of the error in question.**

Dreamweaver automatically opens the selected document and highlights the line(s) of code that contains the error.

3. Correct any errors within the HTML code as needed and rerun the report.

Occasionally, fixing one error can result in another, so it's always a good idea to rerun the report at least once to ensure that you have identified and corrected every error, warning, and message.

When no problems are found in the code, Dreamweaver displays the message No warnings or errors found [DTD].

Checking browser compatibility

One of the biggest issues with building Web pages is ensuring that those pages are cross-browser compatible. This means that they should look good and function properly in as many browsers as possible.

Many code editors include some kind of browser compatibility tool that takes a look at all the tags in your files and determines whether those tags and any attending attributes are compliant with the latest W3C recommendations for your selected DTD in the most popular current browsers. These tools should be able to quickly identify coding issues, such as whether the code contains any deprecated tags like `<center>` and ``, which should be identified and changed.

Though the browser compatibility test typically doesn't show you how any found errors will look in browsers that don't support a particular tag or attribute, the test results often list the browser type and version that may have difficulty displaying the identified content so that you can do your own testing to correct the error. For instance, using Dreamweaver's Browser Compatibility tool, you might see an error like the one shown in Figure 2-9, which identifies the Three Pixel Text Jog issue, which cannot be properly displayed in Internet Explorer 6.0.

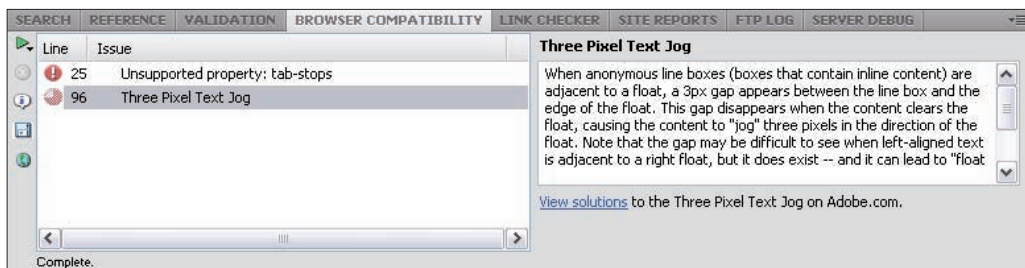


Figure 2-9: Identifying coding issues that can cause browser display problems.

To run the browser compatibility test in a single open HTML file in Dreamweaver, follow these steps:

1. From the **Browser Compatibility** tab in the **Results** panel, click the **green arrow button** in the upper-left corner and select **Check Browser Compatibility** from the drop-down menu.
2. Depending on the number of found issues, the report might take a few minutes to generate. Be patient.

Like the other tabs in the Results panel, the results for this tool display in a list at the bottom of the Results panel with an Error, Warning, or Message icon next to each issue found. In addition, this panel displays detailed results about each found issue along the right side of the panel.

3. To correct any issues found, double-click the line in the Results panel that contains the issue, and Dreamweaver automatically opens the page that contains the error.



The code in question will be highlighted and/or displayed with a red or green wavy underline in Code view to assist in making a correction or adjustment. To reveal a pop-up tip window within the code that identifies the error and lists the browsers that don't support it, place your cursor on top of the wavy underline, as shown in Figure 2-10.

```

94  "http://www.wecansolveit.org/content/solution/enhanced_energy_efficiency/">more »</a></p>
95  <!-- end div.entry-->
96  <div id="navlist-5"> <a href=
    "http://www.wecansolveit.org/content/solution/enhanced_energy_efficiency/">more »</a></p>
97  <h2><a href="http://www.wecansolveit.org/content/solution/enhanced_energy_efficiency/">more »</a></h2>
98  Leadership</a></h2>
99  <p>The technology exists. What's missing is innovative leadership."</p>
100  exist. What's missing is innovative leadership."</p>
101  innovative leadership."</p>
    When anonymous line boxes (boxes that contain inline content) are adjacent to a float, a 3px gap appears between the line box and the edge of the float. This gap disappears when the content clears the float, causing the content to "jog" three pixels in the direction of the float. Note that the gap may be difficult to see when left-aligned text is adjacent to a right float, but it does exist -- and it can lead to "float drop" in tight layouts.
    Affects: Internet Explorer 6.0
    Likelihood: Likely
102  "http://www.wecansolveit.org/content/solution/enhanced_energy_efficiency/">more »</a></p>
103  </div>
104  </body>
105  </html>
106
  
```

Figure 2-10: Displaying questionable code in Code view.



As great as the Browser Compatibility tool is, keep in mind that it checks only for the validity of code within a subset of browsers and browser versions. What it does not do is verify the accuracy of the code or of any functionality of any JavaScript or other code used in the file(s).

Verifying internal and external links

A link checking tool can help you to check your pages for broken internal links (for example, you misspelled `index.html` as `indez.html`), to see a list of all the page's external links, and to identify any orphaned files (unused or unlinked-to files) that you can remove from the site to help save room on the server.

To illustrate how to use Dreamweaver's Link Checker tool on a single open HTML file, follow these steps:

- 1. From the Link Checker tab in the Results panel, click the green arrow button in the upper-left corner and select Check Links in Current Document from the drop-down menu.**

To check links in an entire site or selected files in the Files panel, choose one of the other options from the drop-down menu.

- 2. By default, broken internal links (if any exist) are displayed at the bottom of the Results panel. To correct a link, click the URL under the Broken Links column and edit the text to correct the link.**

An internal link is any hyperlink that doesn't work because the link contains a typo, incorrect path, or improper syntax, or the linked file is not on the server.

Corrected links automatically disappear from the listing.

If no broken links are identified, congratulations!

- 3. Click the Broken Links drop-down menu at the top of the Link Checker area, just below the Results panel tabs, and choose External Links.**

Any links going to pages outside the site to a different URL are listed here. Though you can't test these links from within Dreamweaver, the list can be a useful tool for identifying the links that need to be verified.

- 4. Click the External Links drop-down menu at the top of the Link Checker area, below the tabs, and choose Orphaned Files.**

An *orphan* is any file, image, or other asset in a Web site that isn't linked to any other files in the site and may therefore (in most cases) be deleted or otherwise relocated to a backup location.

This report feature can only be used to check for orphans on an entire managed site, so when the alert box appears, click the OK button.

- 5. Click the green arrow button in the panel again, but this time, choose Check Links for Entire Current Local Site from the drop-down menu. When the results appear, click the Broken Links drop-down menu and choose Orphaned Files.**

Any file, image, or other asset saved to the local managed site that isn't being referenced by another file within the site (linked to) will be displayed in the Results panel.

- 6. If you know that a particular file in this listing is unnecessary to the functionality of the site, either archive it in another location or delete it.**
- 7. As a practice, rerun the report after making any changes to ensure that you've caught and corrected as many errors as possible.**

Generating site reports

Many code editors also include some kind of HTML reports feature you can run to help you find common code mistakes that can affect page download time and create performance and display issues. Run these reports for every site to make sure that you catch those little problems that might otherwise slip through the cracks.

Although each code editor may have slightly different reporting tools, most should have certain features in common. For example, Dreamweaver's reports can identify the following issues:

- ✓ **Combinable nested font tags such as `Hello`, which could be either rewritten as `Hello`, or better yet, the font tags could be stripped so that you can style the page with CSS.**
- ✓ **Accessibility issues, which identify ways the code can be improved.**
- ✓ **Missing alt text attributes for any `` tags.**
- ✓ **Redundant nested tags that can be safely removed, such as deleting the extra `` tags around the words *vernal equinox* in the following sentence: `Spring begins on the vernal equinox`, which is usually March 20 in the Northern Hemisphere.**
- ✓ **Removable empty tags that are unnecessary and can be deleted from the code, such as a tag pair that surrounds no content (``), or an opening tag that is missing its closing tag, as in `<p>Hello world.`**
- ✓ **Untitled documents and files with empty `<title>` tags.**
- ✓ **Workflow for designers working within a group setting. These special reports help identify files that have been checked out by particular teammates, locate files with associated Design Notes, and display files that have been recently modified.**

To run the HTML Site Reports feature on an entire managed site in Dreamweaver, follow these steps:

- 1. From the Site Reports tab in the Results panel, click the green arrow button in the upper-left corner.**

The Reports dialog box opens, as shown in Figure 2-11.

- 2. Select each of the options listed in the HTML Reports area of the dialog box.**

Leave all the Workflow reports options deselected.

- 3. Click the Run button to run the report.**

Results are listed at the bottom of the Results panel and are each identified by filename, code line number, and a description.

- 4. Make corrections directly inside the Results panel or double-click any of the entries to directly edit the document in question.**

For instance, one warning might be that a particular image is missing its `alt` attribute. To fix it, double-click the error and update the HTML code.

- 5. When you're finished making corrections, save any changes in the file and then rerun the report with the same settings.**

Rerunning the report can help you feel confident that all the errors have been addressed.

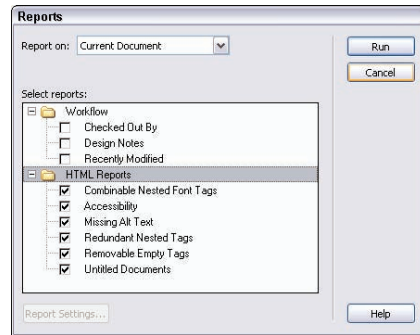


Figure 2-11: Run reports to find common coding mistakes.

Validating HTML and CSS Markup

Validating your markup, both the HTML and the CSS, is the last step in the process of testing your pages before you publish your site. The validation process itself is automated, which means that you just run your pages through a system and then correct any coding errors that are found along the way.

Dreamweaver, as explained in the previous sections, has a good set of built-in code validation tools in the Results panel that help designers improve their code prior to publishing. Other code editors and online validators use similar tools to achieve the same goals. Although using these tools isn't mandatory, they can definitely help designers identify and fix problem code locally before testing pages using the online validation tools, which are described in the following section.

Using free online validation tools

Fortunately, a couple of really fantastic free online validation tools are at your disposal. The W3C and WDG both provide tools that conform to and validate against the W3C's latest recommended standards for CSS, HTML, XHTML, and 508 accessibility. Table 2-1 lists the names and URLs of all the recommended free online validation tools.

Table 2-1 Online HTML, XHTML, and Accessibility Validators	
<i>Validator Name</i>	<i>URL</i>
W3C Markup Validator	http://validator.w3.org
W3C Link Checker	http://validator.w3.org/checklink
W3C CSS Validator	http://jigsaw.w3.org/css-validator
W3C Log Validator	www.w3.org/QA/Tools/LogValidator
WDG HTML Validator	www.htmlhelp.com/tools/validator
WDG CSS Checker	www.htmlhelp.com/tools/csscheck
WebAim WAVE 4.0 Accessibility Tool	http://wave.webaim.org
HiSoftware Cynthia Says Portal	www.contentquality.com/Default.asp
AnyBrowser Web Site Viewer	www.anybrowser.com/siteviewer.html
A listing of free validators	www.thefreecountry.com/webmaster/htmlvalidators.shtml

Discovering what these tools test

Most of the free validators allow you to test in up to three different ways, depending on the location of the files being tested and the particular validator being used:

- ✓ **Validate by URL:** To test by URL, you must upload the page you're testing to a live, working server. This means you could upload the files to a testing server or a hidden directory on the actual server that will be hosting the site, such as <http://www.mywebsite.com/test>, and validate by URL from there.
- ✓ **Validate by upload:** This method allows you to browse for (on a local computer or at some remote destination), select, and upload a single HTML file for validation. For more advanced options when using the W3C validator, go to <http://validator.w3.org/file-upload.html> (which may not work in Internet Explorer on PCs running Windows XP SP2).

- ✓ **Validate by direct input:** To test the code on a single Web page before that file has been uploaded to a live server, copy the entire document — from DTD to closing `</html>` tag — and paste it into the Direct Input or other appropriate testing text area on the desired online validation page.



Although these tools are free, performing the validation on all the pages might take a while because you can validate only a single page at a time. The exception to that rule is the WDG batch validation option at www.htmlhelp.com/tools/validator/batch.html, which allows you to input multiple pages by separating each URL with a new line in the input area, as illustrated in Figure 2-12.

Figure 2-12: Validate several pages at once with the WDG batch validator.

Using the W3C Markup validator

All the free online validators in Table 2-1 are quite easy to use. To illustrate, follow these steps to validate a Web page using the W3C Markup validator:

1. **Point your browser to** <http://validator.w3.org>.
This is the W3C's main Markup Validation page.
2. **In the Validate by URL Address field, shown in Figure 2-13, type the complete path of the following test page:** www.luckychair.com/webai/validation.html.



Figure 2-13: Enter your test URL and click the Check button to validate your code.

The remaining steps use this sample file to illustrate specific validation issues. When you have completed this exercise, feel free to repeat the process using one of your own Web pages.

3. Click the Check button to validate the page.

The W3C server processes the validation and returns results in the same browser window.

The results should display a big red bar across the top that reads Errors found while checking this document as XHTML 1.0 Transitional! Below the red bar, you should see details of two found errors on lines 44 and 46 regarding the omission of alt text attributes on two of the page's graphics:

```
Line 44, Column 84: required attribute "alt" not specified.  
..." width="100" height="30" id="dell" /></td>  
Line 46, Column 84: required attribute "alt" not specified.  
..." width="100" height="30" id="sony" /></td>
```



TIP

The W3C validator returns line numbers in the error descriptions. To see which actual line numbers the validator is testing against relative to your code, you need to resubmit the test page. At the top of the validation failed page, you should see an area labeled Options. Inside this area, enable the Show Source option and click the Revalidate button. When the results are returned, you will see a new Source Listing area below the results that contains the line numbers in the code.

4. In a new browser window, point your browser to www.luckychair.com/webai/validation.html and save a copy of this page to your local computer using the browser's File menu.

The Save option in your browser is typically something like File→Save Page As or File→Save Page.

You're now going to correct the errors and retest the page.

5. Open the downloaded `validation.html` page in your favorite HTML editor and add the `alt` text attribute to each of the images as indicated in the W3C failed validation results.

Insert descriptive `alt` text attributes to the code as follows:

```
<tr>
  <td></td>
  <td></td>
  <td></td>
</tr>
```

6. Return to the main Validator page at <http://validator.w3.org> and click the Validate by Direct Input tab at the top of the page.

In this area, you can validate the code by direct input, without needing to upload a test file to a working host server.

7. Save the changes to your updated `validation.html` file. Then select and copy all the code on the page (from DTD to closing `<html>` tag), paste it into the Validate by Direct Input field, and click the Check button.

Pasting in the HTML code from a local copy of a file can sometimes be faster than the Validate by URL and Validate by File Upload methods.

When the results appear, you see a green bar across the top of the results page that reads *This document was successfully checked as XHTML 1.0 Transitional!* — like the one shown in Figure 2-14.



This tool, like most the others, can only verify one page at a time. Fortunately, single-page validators can often identify site-wide coding issues, which can be quickly fixed on all the pages of a site. For instance, if the Validator finds a missing tag, you could use an automated process like Dreamweaver's Find and Replace tool to find the problematic code and replace it with the corrected code in all the pages of a managed sited.

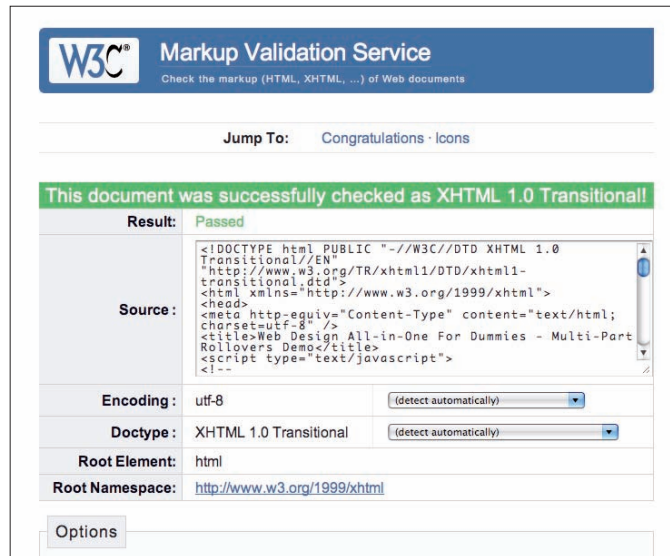


Figure 2-14: Pages that pass validation display a “passed validation” notice.

Fixing noncompliant code

For each coding issue identified by a validator, you need to determine what course of action to take. Although some culprits that repeatedly crop up are easy to fix, like missing `alt` text and `<noscript>` tags, you’re bound to find coding issues that completely baffle and stump you. For instance, if you get an error message that reads XML Parsing Error: Opening and ending tag mismatch: `br` line 52 and `body`, it might be difficult to figure out what that means, let alone why it was caused and how you should fix it. As a strategy then, try to fix the issues within the code from the top down, as they are listed in the validation results, because sometimes fixing one issue resolves another. With the XML parsing error, that issue might disappear when you correct for an omitted closing element on a `
` tag listed earlier in the error results.

Most often, you can fix noncompliant code by hand or with the help of a good HTML editor. While you’re busy making all the corrections, remember that making your pages pass validation makes the site more accessible to more visitors, which can translate into more visitors overall and potentially more sales. Cleaner code also means faster page downloading times, improved search engine accessibility, and quicker page maintenance and site updates in the future.

The best way to find out how to code better and make fewer mistakes before validation testing is to make lots of honest mistakes and figure out

how to correct them on your own. To help you identify some of the more common coding mistakes, Table 2-2 lists several code issues along with suggestions about how to fix them.

Table 2-2 Common Noncompliant Code Fixes	
<i>Problem</i>	<i>Solution</i>
alt text attribute missing from tag	Add the alternate text attribute, either with or without a description, as in <code></code> <code></code>
<noscript> tags missing from code	Add <noscript> tags below each instance when JavaScript is present in in-line JavaScript or at the end of the content before the closing body tag. Between the <noscript> tags, insert HTML content (text, graphics, media files, and so on) that describes the function of the JavaScript and, when appropriate, how visitors can access the information revealed by it, as shown here: <code><script language="JavaScript" src="bookmark.js" type="text/javascript"></script><noscript>The JavaScript used on this page provides a quick link that allows visitors to automatically bookmark this page. As an alternative, please use your browser's Bookmark This Page feature.</noscript></code>
Flashing or flickering element(s) detected, such as animated GIFs, Java applets, and other multimedia plug-ins	Adjust the speed of any animations to avoid causing the screen to flicker with a frequency between 2 Hz and 55 Hz. Animations that exceed these two measures may cause seizures in visitors with photosensitive epilepsy. For further details, see www.access-board.gov/sec508/guide/1194.22.htm#(j) .
No DOCTYPE specified	Add a valid DOCTYPE above the opening <head> tag.
No HTTP charset parameter specified	This special meta tag specifies the character set used in the HTML code. Some HTML editors include it automatically when generating new blank Web pages. If validation finds that this tag is missing from your code, insert the following code by hand: <code><meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"></code> For further information visit www.w3.org/International/0-charset.en.php .
No <title> tag specified	Add a unique title between <title> tags in the HEAD area on each page.
No <meta> tags specified	Add meta keywords and meta description tags to the HEAD of each page. These can be identical on every page on the site. If desired, you may also add additional meta tags as needed.

continued

Table 2-2 (continued)

<i>Problem</i>	<i>Solution</i>
No Robots tags specified	Add the Robots <code><meta></code> tag in the HEAD of the page to instruct Web spiders and robots whether to index the page and follow any hyperlinks, such as <code><meta name="Robots" content="All"></code> .
Deprecated <code></code> tags detected	Move all the presentation markup of the HTML (page, fonts, tables, links, and so on) to an external CSS file and remove all <code></code> tags and formatting attributes.
Deprecated table height attribute detected	Control table cell heights, when necessary, with CSS styles.
Style attributes detected in the opening <code><body></code> tag	Move <code>body</code> attributes, like margin attributes and background page color, to a <code>BODY</code> tag redefine style in an external CSS file.
Type attribute not specified for JavaScript or CSS	Add the <code>type="text/css"</code> attribute for <code><style></code> tags and the <code>type="text/javascript"</code> attribute for <code><script></code> tags: <pre><style type="text/css" > <script type="text/javascript"></pre>
Entity name used instead of entity number	Change the entity name to an entity number, such as using <code>#\$169;</code> instead of <code>&copy;</code> to create the copyright symbol (c).
No background color attribute was specified for a CSS style that specifies text color	Provide each style that contains a <code>text color</code> attribute with an attending <code>background color</code> attribute. The background color should match, or closely match, the background color upon which the text will display on.

When you're finished identifying and adjusting all the noncompliant code identified by the validation tools, and have fixed everything that needed fixing, move on to the retesting and acceptable failure phase of the testing process.

Retesting and failing acceptably

Your ultimate goal when testing for code accuracy is to get a clean bill of health from the various HTML, CSS, and accessibility validators for each page on your site. To achieve that goal, you need to spend some of your time retesting each page after making adjustments to the code. Regrettably, you may run across some issues that are simply unfixable given your time frame and budget. These issues often crop up when you use special HTML and CSS hacks to make the page display a particular way in certain browsers.

For those times when a coding error causes a page to display an object not exactly as you intended, but the lack of styling doesn't alter the overall layout of the site, those pages are referred to as *failing acceptably*. Likewise, when an error in the code makes the page look completely jumbled in a browser, that page is referred to as having *failed*.

To illustrate how a page might fail or fail acceptably, think of a page that displays perfectly in nearly every browser on both a Mac and a PC, but appears skewed when viewing it in Internet Explorer 6 on a PC and Safari 2.0 on a Mac. To resolve the display issue, you could create separate CSS files for each browser and use browser-detection JavaScript to automatically pair the right CSS with the viewer's browser. Alternatively, if the skewing is minor, say just a shift of about 10 pixels in one tiny area of the page, you could choose to live with the disparity.

When you cannot find a solution to a particular coding error or display issue, you must determine whether you can live with the consequences of having a site that is either aesthetically inconsistent across multiple browsers or not as accessible as it should be. Certainly if the page looks good in IE but looks funky in all the other browsers, your validation and testing work is far from finished. Yet if the page looks good in IE, Firefox, Opera, Safari, and Chrome but just doesn't seem to display exactly the same way in Safari on a Mac (as might be the case with the way text is rendered and how it wraps on the page, leaving orphans and widows within the text, as illustrated in Figure 2-15), perhaps you can live with that particular audience (approximately 2 percent of all Internet users) not being able to see your page in its ideal way. Ultimately, it's up to you. Before you give up, however, keep in mind that in most cases, you should be able to find a solution to your coding issue if you research the issue online.

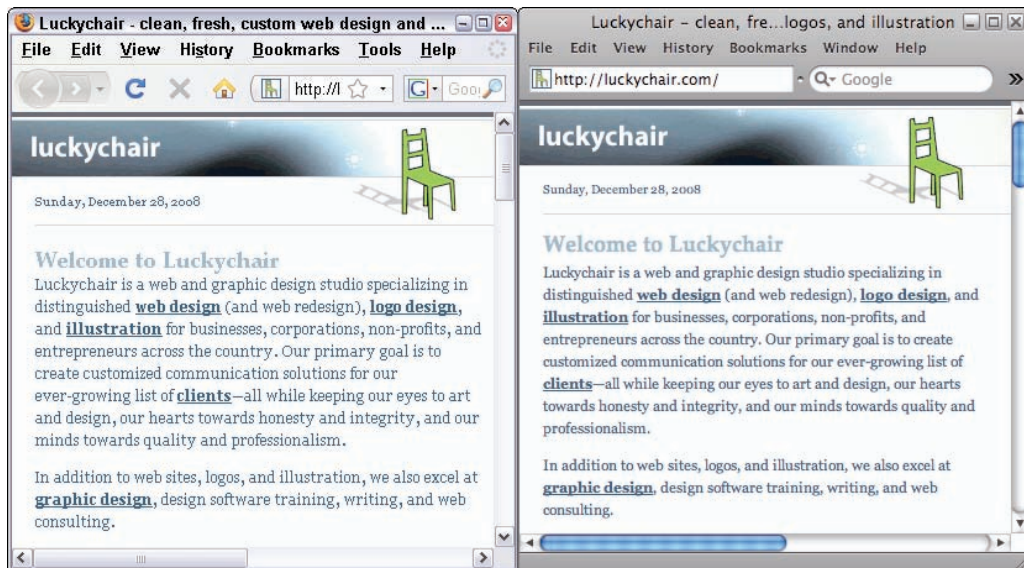


Figure 2-15: Text renders differently in Firefox (left) than it does in Safari (right).

Obtaining proof of validation

After running through the validation process, fixing any issues that needed adjustment, and ultimately getting a green light on the validity of the code on your pages, you have just one more question to ask yourself before you publish your site: Do you want to add any icons to the site as proof of validation?

Along with displaying successful validation notices, most validation sites include directions on how to add a validation icon to your page as proof of your page having passed the validation process. For example, the W3C offers several free validation icons, like the ones shown in Figure 2-16, that you can proudly display on your site for HTML 4.01, XHTML 1.0, CSS, and WCAG 1.0 (Grades A, AA, and AAA) code compliance. WCAG (Web Content Accessibility Guidelines) is a set of recommended accessibility guidelines that can help designers make Web pages more accessible to people with disabilities. To find out more about WCAG, visit www.w3.org/tr/wai-webcontent.



Figure 2-16: The honor system at work.



The W3C states that validated pages are those that have stated DTDs against which validation using either an SGML or XML parser took place. Valid pages, thus, are considered interoperable (meaning they conform to standards and can be accessed with a variety of devices such as Web browsers, search engine robots, text-to-speech applications, and other assistive devices) and are granted permission to display the appropriate validation icon.



Claims of compliance and validity of your pages are not in any way verified or endorsed by the W3C, and you are entirely responsible for the proper usage of these validation icons.

To illustrate how to add a validation icon to your pages, follow the steps shown here to place the HTML 4.01 validation icon on a sample Web page:

1. **Open a blank HTML file in your favorite HTML editor and add the following HTML code to the location within the page where you'd like the HTML 4.01 validation icon to appear:**

```
<p>
  <a href="http://validator.w3.org/check?uri=referer">
  </a>
</p>
```

This code displays a Valid HTML 4.01 Transitional icon.

2. Save the file and launch it in a browser window.

The W3C HTML 4.01 Transitional validation image should appear on the page. As long as the entire paragraph of code remains intact, the logo can be placed anywhere inside a valid page. What you may not do is alter the look of the image in any way or add any special formatting to it.

The code provided by the W3C contains a URL to the W3C for the source of the graphic. If you'd prefer to have the graphic display from your own site rather than having to access the W3C site to display, you may create a copy of the desired graphic in either GIF or PNG format, place your copy of the graphic into the `images` folder of your own site, and change the source within the code to reflect the graphic's new location, like this code that shows proof of valid CSS:

```
<p>
  <a href="http://jigsaw.w3.org/css-validator/">
  </a>
</p>
```

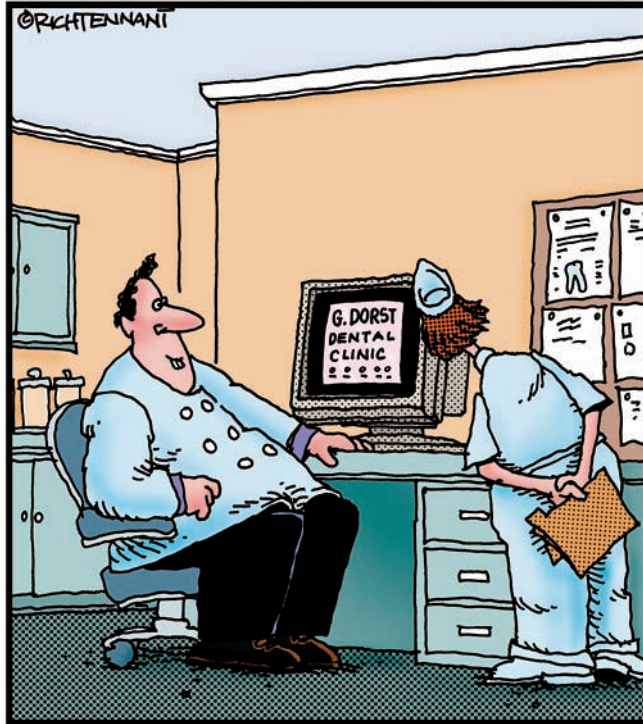
Now that you know how to insert a validation icon onto your pages, visit the validation logo information and usage page on the W3C Web site to get the code for the other icons: www.w3.org/Consortium/Legal/logo-usage-20000308.html.

Book V

Publishing and Site Maintenance

The 5th Wave

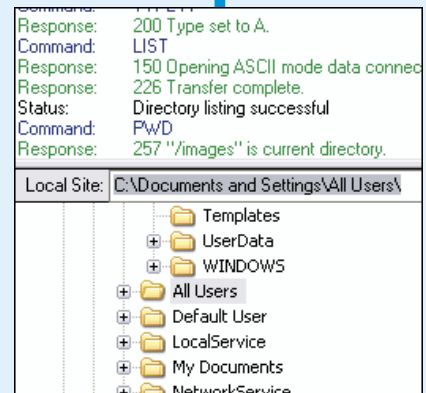
By Rich Tennant



"Well, it's not quite done. I've animated the gurgling spit sink and the rotating Novocaine syringe, but I still have to add the high-speed whining drill audio track."

Getting your site online often starts with registering a domain, securing a hosting plan, and publishing your finished site on the Web using File Transfer Protocol (FTP).

Publishing a site is often just the beginning of your Web site's existence. You also want to look for ways to improve your site's search-engine friendliness using Search Engine Optimization (SEO) techniques as well as making sure that your site stays current with new information so that visitors will come back to your site regularly. This minibook covers domain registration, hosting, publishing, SEO, and site maintenance.



Chapter 1: Domain Registration and Hosting

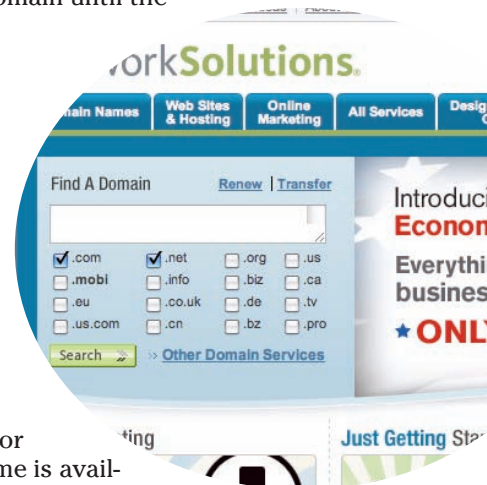
In This Chapter

- ✓ Selecting a domain name for your site
- ✓ Getting help from domain name generators
- ✓ Registering a domain name
- ✓ Finding the best hosting plan
- ✓ Designing a placeholder page for the new domain
- ✓ Uploading a placeholder page to the new domain

This chapter focuses on what you need to do to prepare your site for publishing. You find out about domain name selection, name generators, domain verification, and domain registration. You also discover how to find a good hosting plan, including what to look for in a plan, where to find a host, and general pricing structures. The last part of this chapter shows you how to create a customized *placeholder page*, which is a single, simple Web page with company branding, an e-mail link, and a smattering of other contact information that can hold the place on a new domain until the new Web site is fully built and ready to publish.

Understanding How to Get Your Site Online

Before you can make a new Web site available on the Internet, it must first be assigned its own special Web address, which is commonly referred to as the *domain name*. The process of acquiring a domain name can be a fun adventure and only requires a little bit of work. To start with, someone needs to think of a good name for the site and then check to see whether that name is available for use or whether it has already been taken. If the desired name is available, the name needs to be registered, and that can be done either through a domain registrar or a host provider.



After the domain name is registered, the site needs a hosting plan so that it can reside on a server and thus become available to the public. All of this needs to be done before you can publish the site. Most people like to register the domain and get a hosting plan at the start of any new Web project to ensure that the domain name is secure during the site-building process. And, because building a site takes a good bit of time, most folks also like to have a customized placeholder page designed and uploaded to the domain until the new Web site is ready for publishing.

If you are working as a freelance Web designer, some of your Web clients will have already registered their domain and secured a hosting plan before contacting you about your design services. Other new clients, however, will not have done any of these things and will need a fair amount of hand-holding from you as you take them through each of the steps. Being able to provide information about these topics to your clients can both enrich your skills as a Web professional and enhance their experience with you as a designer. This fact alone can be very good for business because happy clients are more likely to return to you for Web site maintenance services after their site gets published, as well as refer other friends and business acquaintances to you if they feel confident in all your Web-related skills.

If this is your first time dealing with domain names and hosting plans, you may want to try setting up a domain name and hosting plan for your own Web site before handling these tasks for any Web clients.

Selecting a Domain Name

Choosing a domain name for a Web site is something that you, in your role as a designer, may or may not be involved in when working with Web clients, depending on their individual needs and how Web-savvy they are. Some Web clients will have already selected a domain, registered it, and secured hosting, whereas others will say they don't really understand anything about all that stuff and are relying on your experience to help them figure it all out, or in some cases do it all for them because they don't care to know. Some clients, of course, will fall somewhere in between these extremes, needing a little help with some but not all of these domain-related responsibilities.

In the following sections, you find out more about domain names, how to help select a domain name for your client, and then how to check to make sure the name is available.

Understanding what a domain name is

Simply put, a *domain name* is a name that is used to identify an address on the Internet for a particular Web site and any e-mail addresses configured for that site, such as `http://www.cleanfordreams.com` and `info@www.cleanfordreams.com`. The Web address itself is composed of four distinct parts, as diagrammed in Figure 1-1:

- ✓ **Protocol:** The first part of a Web address, `http://`, is the HyperText Transfer Protocol (HTTP), which identifies the protocol that allows a computer to browse the Web by getting information from a remote server. Secure access to the Internet (that is, anytime a domain has an SSL [Secure Sockets Layer] certificate installed on the host server for encrypting private data) requires the use of the `https://` (note the *s* for secure) protocol.
- ✓ **www:** The second part refers to the World Wide Web and identifies the type of page that will be delivered in a browser window. You might notice that some sites still display in your browser without the `www` part of the address, such as typing just `google.com` into your browser's address bar instead of `www.google.com`, but that function is typically server dependent and isn't a universal feature of domain names. Another type of Web address includes domains where the `www` is omitted, such as in `http://maps.google.com`. This type of address refers to a subsite or subdomain that resides on the main domain's servers but is separate from it.
- ✓ **Domain name:** The third part identifies the unique name of the Web site as registered by the owner of the site. Domain names may contain any combination of uppercase and lowercase letters and numbers. In addition, though less often used, domain names may also include hyphens but no other special characters, as in `www.Jet-StreamShowerhead.net`.
- ✓ **Extension:** The fourth part identifies the type of site visitors should expect to see at the address, such as `.com` for commercial business sites, `.org` for nonprofit organizations, and `.edu` for educational sites.

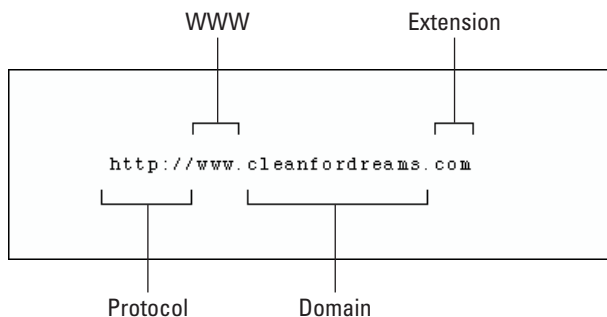


Figure 1-1: The four distinct elements in a Web address.

Although several unrestricted domain extensions are in use by all kinds of businesses around the world, the most familiar extensions should really be used as they were intended. For example, the `.org` extension should be used for nonprofit organizations, and the `.gov` extension should be used exclusively for government agencies. Table 1-1 lists the more common extensions from around the world.

Table 1-1 Common Web Domain Extensions	
<i>Extension</i>	<i>Usage</i>
<code>.com</code>	Commercial, but is commonly used for just about any kind of business
<code>.net</code>	Internet administrative site, but is also commonly used for other types of sites
<code>.org</code>	Organization, most often used by nonprofit groups and trade associations
<code>.info</code>	Information, the fourth most popular extension
<code>.biz</code>	Business
<code>.us</code>	United States
<code>.name</code>	Personal Web sites
<code>.at</code>	Austria
<code>.be</code>	Belgium
<code>.bz</code>	Belize
<code>.cc</code>	Cocos (Keeling) Islands
<code>.cn</code>	China
<code>.de</code>	Germany
<code>.eu</code>	European Union
<code>.gs</code>	South Georgia and the South Sandwich Islands
<code>.ms</code>	Montserrat
<code>.mx</code>	Mexico
<code>.nz</code>	New Zealand
<code>.tc</code>	Turks and Caicos Islands
<code>.tv</code>	Tuvalu, but often used for television
<code>.tw</code>	Taiwan
<code>.uk</code>	United Kingdom
<code>.vg</code>	British Virgin Islands
<code>.ws</code>	Western Samoa, but is often used for Web sites



For a complete listing and description of all the domain name extensions available to you and your Web clients, visit the following Web site:

www.networksolutions.com/domain-name-registration/popup-extensions.jsp

Finding a domain name for your client

The first thing to do when researching a domain name for a business is to see whether the name of the business is available. For example, if your Web client's company is Station Organization, the most fitting domain name for the company would be `www.stationorganization.com`. With an unusual or unique business name, selecting domains can be fairly quick and easy. It is when a client has a common business name, which is likely to have been already taken by someone else, that things can become tricky.

If the desired domain name is already taken, you or your client needs to come up with a new name, and tinkering with the desired name is a good place to start. Easy solutions work best, such as adding a city name or state reference, inserting a hyphen between certain letters, or using common abbreviations within the name. For example, if the company name is Rochester Apartments and it's located in New York, the client could consider using `rochesterapartmentsny.com`, `rochester-apartments-ny.com`, `rochesteraptsny.com`, or `rochester-aps-ny.com`. Conversely, the client might also consider using a different domain extension, either with or without the other name adjustments, such as `rochesterapartments.net` or `rochester-apartments.info`. If your client gets really stumped trying to find the perfect domain name, you can help him find the right one using a domain name generator, as described in the next section.



Part of your job in helping a Web client choose a domain name also includes helping to select an appropriate domain extension. In some cases, the domain names with the chosen extension will already be taken by another company with the same name. In those instances, the Web client needs to use a different extension with the desired domain name, alter the spelling of the desired domain name, or come up with a similar but different domain name with the desired extension.

Using domain name generators

To get help finding a suitable domain name, whether the one you want is already taken or you are just interested in seeing what kinds of domains are available based on a few keywords, turn to one of the popular online domain name generators. In addition to helping you come up with new and unusual name ideas, these services can also suggest suitable alternatives based on real-time domain name availability. Most generators take whatever word or words you'd like to include in the domain name, then shake them out in a

variety of combinations either with or without other words, and present a resulting list of potential names for you to choose from.

The most popular and useful domain name generators can be found at NetworkSolutions.com, DomainsBot.com, NameTumbler.com, and NameBoy.com. At Nameboy.com, shown in Figure 1-2, you can enter a primary and optional secondary word to begin the search and choose whether returned results include hyphens between characters and rhyming, which can sometimes make the domain name easier to remember. It also allows you to verify domains you're interested in with its handy WHOIS search form. You can even search for and register domains that have expired or are about to expire.



The screenshot shows the Nameboy domain search website. The header features the Nameboy logo (a cartoon boy with spiky hair) and the text "nameboy® domain search". Below the logo, it says "The world's most popular domain appraiser is now offering Free Domain Name Appraisals". The navigation menu includes "New Search", "FAQ", "About Us", "Affiliates", "Appraisals", "Renew Domains", and "Clubhouse". The main content area is titled "Available domain names and domain name ideas from Nameboy domain name generator." and includes a search form with "Primary Word" and "Secondary Word (optional)" fields. Below the form are checkboxes for "Allow hyphens" and "Rhyme". A promotional banner on the right says "NOW \$15 .COM .NET .ORG" and "Find the keywords you".

Figure 1-2: A domain name generator can help you find suitable alternatives for your ideal domain name.

No matter which service you use, be sure that you verify the domain name availability (as described in the next section) before plunking down any money to register the domain. You may also want to find out about registration services and hosting plans before you ultimately commit.

Checking domain name availability

Even if you do not require the services of an online domain name generator, you still need to verify that the domain name you have chosen for your Web site is really available for registration. This means ensuring that no one else

is currently using the domain you want, or has already registered it but hasn't begun using it yet.

Thankfully, verification is free and quick to do on a number of Web sites, including many of the domain registrar and domain name generator sites. For example, NetworkSolutions.com provides fast results along with providing automatic alternate name suggestions should the name you enter already be taken. With a good verification tool, finding the right name for your site should be only a matter of a having a bit of patience and open-mindedness while you perform the search and verification process.

To show how easy it is to use these domain name verification services, follow these steps to check for the same name on both NetworkSolutions.com and DomainsBot.com:

1. **Point your browser to `www.networksolutions.com`, as shown in Figure 1-3.**



Figure 1-3: Network Solutions is a reputable place to register your domain name.

2. **In the Find a Domain text box in the upper-left corner of the page, type the domain name you want to register (leaving the .com and .net options selected) and click the Search button.**

For example, type `rochesterapartments`.

In the search results that appear, you see that this domain is already taken for the .com, .net, .org, and .info extensions. Other extensions that are available include .mobi, .us, .us.com, .tv, and .biz.

3. **Do another search. This time, enter a variation of the desired domain name in the Find a Domain text box and click the Search button.**

For example, type **rochester-aps**.

Aha! This domain name is available with both the .com and .net extensions and could be registered today if you wanted it (presuming that no one has registered this domain since the time of this writing).

4. Now go to the DomainsBot.com Web site and type the words your first choice of domain name into the search field.

For example, enter **rochester apartments**.

Before you click the Search button, notice how a little pop-up message window, like the one shown in Figure 1-4, opens to reveal the availability of any domains that use the two words in the search field. The rochesterapartments.com domain name is included in this list and is noted as Not Available.

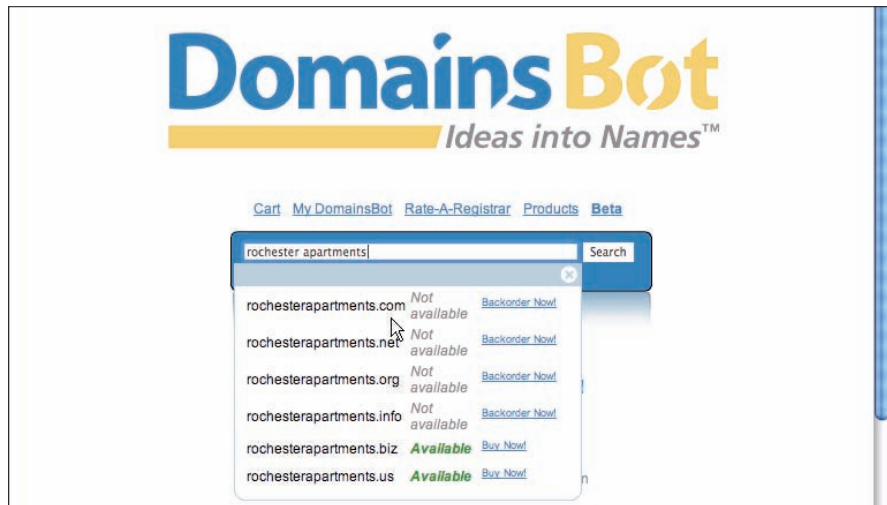


Figure 1-4: DomainsBot can help you find suitable alternatives for domain names that are already taken.

5. Click the Search button to have DomainsBot run a search.

As you can see, these search results are more detailed than those from NetworkSolutions.com. In addition, the page includes extra tools to help you refine the search based on selected criteria.

6. To modify the domain search results, adjust the settings in the LiveBot panel on the right side of the page.

For example, adjust the following settings:

- Under Apartments, select lofts and suites.

- Under View Only, deselect all the extensions except .com, deselect everything under Include except for Hyphens, and deselect the Expiring and For Sale options under Status.

7. Click the Update Results button.

Almost immediately, the results listing updates to show the availability of domains that match the new search criteria. If the apartments are located somewhere in East Rochester, for example, EastRochesterLofts.com and EastRochesterApartments.com might be good choices, for example. However, if the domain name really needs to exactly match the company name, the client may be out of luck and will need to come up with some other ideas to find a suitable domain name to register.

If you can't find exactly what you're looking for the first few times around, be open to new ideas and try searching on other domain name generator Web sites. The more you search, the more likely you are to come upon a domain name that really works.

Registering a Domain Name

As soon as you find the right domain name for your Web site, try to get it registered as quickly as you can. Think of registering a domain name like reserving a rental car: To reserve the car, you must provide the car agency with your name and contact info, credit card, car preference, and rental dates. To actually use the car, you must go to the rental car agency, sign a contract to rent the car for a predetermined period of time, and provide payment information before the agency gives you the car keys. Likewise, registering the domain is a means of placing the name on reserve for later use.

To register your domain name, you can go about it in one of two ways; Register your site with a domain registrar or sign up for a hosting plan and have the host provider register the domain for you as part of your hosting package. Both options have unique benefits and drawbacks.

Using a domain registrar

When registering a domain name through a domain registration service, you get the benefit of quick and affordable registration without having to worry about hosting until you are ready to publish your site. In most cases, you should be able to choose a time frame for the rights to use your chosen domain name. Typical terms for domain registration are for one, two, three, five, and ten years. Rates for the registration range from as low as \$1.67 to over \$35 per year. Many services include e-mail accounts and other services for this fee; however, some do not, so be careful to understand exactly what you are paying for. Most domain registration rates fall somewhere in the \$10–20 per year range, and the rate often drops significantly when you

increase the term of the registration to two years or more. The .com domains are usually the most sought-after domains and tend to cost more, whereas the less trendy .info and .mobi extension domain names can now be grabbed up for as little as \$.99 per year.

Choosing the right domain registrar — if you want to use one instead of doing it through your host provider — depends on your time frame, budget, and needs. The most popular domain registrars tend to be the ones that charge the least amount of money. Keep in mind, though, that while more affordable, the less expensive services may not necessarily provide the best customer care, which depending on your level of knowledge, might be an important factor to you and your client. Therefore, shop wisely and do your research before you procure a domain for yourself or a Web client. Alternatively, you could pass on this part of the process and recommend that your client registers on his own through a particular registrar or host provider.

In addition to domain registration, most registrars these days provide additional Web-related services, including domain verification, domain name generators, domain transfers, hosting, and Internet access.

If you just need to register the domain name for a time before the site is ready to publish, but aren't interested in e-mail or hosting until right before the site gets launched, it would probably be fine for you to use one of the cheaper domain registrar services. However, if you know you will be needing other services such as Web hosting and e-mail, obtaining an SSL certificate, and setting up an e-commerce shopping cart, go with one of the companies that also provides those services, such as NetworkSolutions.com, Enom.com, Tucows.com, DomainDirect.com, LunarPages.com, DirectNic.com, GoDaddy.com, and Register.com, or any of the myriad hosting services that might have been personally recommended to you by friends and business associates.

Before you do that, however, consider your long-term needs:

- ✓ **Your site is ready or will have a short turnaround time (say, 30 days or less):** You may want to speed the domain registration process by registering the domain through the host provider that will be hosting the site.
- ✓ **Your site will not be ready for publishing for quite some time:** You can save some money by registering the domain and not worrying about hosting until you're ready to publish the site. If this is the route you decide to take, just make sure that you understand that if you register the domain with one company and use another company for hosting, you will need to do the DNS transfer when the hosting plan has been secured. To avoid having to do the DNS transfer, simply register the domain name while you sign up for a hosting plan with your preferred host provider.

Using a host provider

When registering a domain name through a host provider, all you need to do is tell the provider your chosen URL when you sign up for your hosting account. It registers the domain for you as part of the hosting plan, often without any additional fees. To find out more about hosting, see the later section “Finding the Best Hosting Plan.”

Activating your domain

To use the registered domain name and allow visitors to access an actual Web site through that name, you must secure a hosting plan and activate the domain name. When you register a domain with one company and host with another (which a lot of people do to try to save a few bucks), the site can be activated for hosting only through a *DNS transfer*. By contrast, when you register your domain and host with the same company, the site becomes active almost immediately after payment.



The DNS (Domain Name Server or Domain Name System) helps create a permanent address for every domain name. Every computer and server that connects to the Internet has its own IP (Internet Protocol) address, typically written in four sets of numbers separated by dots, as in 123.45.67.890. By parking a domain name on a server and creating an alias for the IP to match the domain name, you let visitors begin to use the domain name to find a site.



To get the IP address of your computer, go to <http://whatismyip.address.com>.

A DNS transfer, therefore, means that the host provider’s servers are pointing to the domain name as an alias for the server’s IP address. For the host provider to create that alias, the domain must be transferred from the registrar to the host provider. This ensures that the domain name points to the server that hosts the site so that the site can be properly accessed by everyone on the Web. You find out how to select a hosting plan later in this chapter.

Finding the Best Hosting Plan

A hosting plan is like a parking space for a Web site that you rent out by the month (or year) on a host provider’s server. While there, and as long as the domain name is pointing to the host’s servers, the site will be accessible to anyone surfing the Internet with knowledge of the Web address.

In the following sections, you find out how to find the right host provider and evaluate hosting plans.

Researching host providers

Like domain registrars, host providers are everywhere online, which means that finding the right one for your needs may require a little research. You could consult one of the hosting plan review sites, like [Hosting-Review.com](#), [Top10WebHosting.com](#), or [TheHostingChart.com](#), to find the names of the most popular host providers. However, bear in mind that the host providers on those kinds of lists might be rated more for their pricing than for quality hosting and customer service.

Ideally, you want to pay a reasonable rate, get great technical support and customer service, have enough Web space on the server to host all your files, get the right number of e-mail accounts for your needs, have access to some good site-reporting tools, and be eligible for any special services and discounts that the host provider may have to offer.

On a personal note, having worked with many different host providers over the years, and having experienced firsthand the difference between good and bad customer support, I can highly recommend [LunarPages.com](#) for both domain registration and hosting plans.

[LunarPages.com](#), as shown in Figure 1-5, is a full-service Web development and hosting company that offers competitively priced domain registration and hosting plans with top-rated 24-hour telephone and e-mail technical support. Right now, [LunarPages.com](#) is offering two special discounts to readers of this book who sign up for new 12/24-month hosting plans.



The screenshot shows the LunarPages website interface. At the top, there is a navigation bar with links for Home, About Us, Login, Legal, Contact, Education Program, and Testimonials. The main header features the LunarPages logo and a contact number: 1-877-586-2772. Below the header is a navigation menu with options like Web Hosting Solutions, Support Center, Services, Community, Affiliates, and Webmaster Tools. The main content area is dominated by a large 'WINTER SPECIAL' banner for 'UNLIMITED SPACE! UNLIMITED BANDWIDTH!' at '\$4.95 PER MONTH'. To the right of the banner is a grid of hosting plans, each with a 'BUY NOW' button and an 'INFO' link. The plans listed are: Personal Website (\$4.95), Small Business (\$21.95), Build a Site (\$12.95), Microsoft Hosting (\$9.95), Dedicated Servers (\$99.00), and VPS Hosting (\$39.95). At the bottom of the banner area, there is a domain search field with a dropdown menu set to '.com' and a 'Go' button.

Figure 1-5: LunarPages has fantastic customer support and offers both domain registration and hosting services.



Both discounts include unlimited storage, unlimited data transfer, and \$775 worth of free bonus programs included with your hosting account:

- ✓ **Save \$10:** Mention coupon code WebDesign10 and save \$10 when signing up for any 12 or 24 month Basic Hosting Plan.
- ✓ **Save \$30:** Mention coupon code WebDesign30 and save \$30 when signing up for any 12 or 24 month Business, Windows, or LPQuicksite Plan, or any 3, 6, 12, or 24 month VPS or Dedicated Account.

To find the right host provider for your (or your client's) project, keep the following tips in mind:

- ✓ **Referrals:** A friendly recommendation can often be the best method for finding a reputable hosting plan. Most people either love or hate their host providers. If you keep hearing praise for the same host provider from different people, that can be a good sign.
- ✓ **Customer service:** The single, most important feature of any hosting plan is customer service. Having 24-hour telephone and e-mail support is absolutely essential, so make sure that the host provider you are interested in offers this. If it doesn't, keep looking. Customer service is so important because you will, at some point, need help and shouldn't have to wait a long time to get it. Ideally someone should be there, 24/7, to help answer your technical questions and resolve any server-related issues. Furthermore, if you keep irregular work hours, it would be nice to be able to have your questions answered anytime, day or night.

Make a list of all the host providers you are interested in and then call them to ask questions about their hosting plans. You can get a good sense about a company's customer service by talking to one of its customer service representatives about hosting plans and customer support.

- ✓ **FTP and control panel:** Be sure to inquire about ways that each hosting plan allows site access. At a minimum, you should be able to upload files using FTP (File Transfer Protocol). Some host providers offer only a custom-built "site console" or "control panel" with limited capabilities for uploading files. With FTP access, you have better control over uploading files when it's time to publish the site.
- ✓ **Cancellation policy:** Be sure to also ask each company about its cancellation policy and whether that includes a refund. If for any reason you decide you want to switch plans to another host provider sometime in the future, it would be nice if you could get a refund. The host providers that do offer prorated or partial refunds are often the ones with the best customer service and hosting plans.

Evaluating hosting plan packages

While you are researching the different hosting plans from the various hosting companies, make sure to look for the type of hosting package that's appropriate to your needs. Shared hosting plans tend to come in four distinct flavors: the bare-bones starter plan, the small-business plan, the big-business plan, and the e-commerce plan. Most plans have certain features in common, with tiered levels of benefits, as outlined in Table 1-2.

<i>Feature</i>	<i>Starter</i>	<i>Small Biz</i>	<i>Big Biz</i>	<i>e-Commerce</i>
Monthly fee	\$8	\$14	\$19	\$40
Account	24-hour uptime, customer service, and technical support			
Technical features	5GB hard drive space, 200GB data transfer, 20 million page views, dedicated IP address, 3 user accounts	10GB hard drive space, 400GB data transfer, 40 million page views, dedicated IP address, 6 user accounts	20GB hard drive space, 600GB data transfer, 60 million page views, dedicated IP address, 12 user accounts	40GB hard drive space, 1000GB data transfer, 100 million page views, dedicated IP address, 24 user accounts
Domains	Dedicated domain name, domain transfers, registration of new domains, domain pointers, and so on			
Site management	Online control panel, Web site builder, FrontPage extensions (scripts that provide dynamic functions on sites built with Microsoft FrontPage), 24-hour FTP access, free Web-based statistics, access to raw log files, and so on			
E-mail	250 accounts + 1GB hard drive space, spam guard, virus protection, online e-mail access, autoresponders, forwarders, and so on	500 accounts + 1.5GB hard drive space, spam guard, virus protection, online e-mail access, autoresponders, forwarders, and so on	750 accounts + 2GB hard drive space, spam guard, virus protection, online e-mail access, autoresponders, forwarders, and so on	1,000 accounts + 3GB hard drive space, spam guard, virus protection, online e-mail access, autoresponders, forwarders, and so on
Scripting	ASP, ASP.NET, CGI-BIN folder, Perl, PHP support, and so on			
Database and indexing services	mySQL max, 25MB hard drive space	+ mySQL server, 2 odbc data source names, ms access, 100MB hard drive space	+ mySQL server, 4 odbc data source names, ms access, 250MB hard drive space	+ mySQL server, 6 odbc data source names, ms access, unlimited hard drive space
E-commerce services	SSL secure servers, shared SSL certificates, merchant tools with and without credit card processing, Google Checkout, and so on			
Data center	Firewall and antivirus protection, daily backups, redundant servers with UPS			

features power backups, and so on

Although monthly rates can run as low as \$1.50 and as high as \$99, most plans range from about \$4.95 to \$39.95 per month and differ by the terms of service they provide:

- ✓ **Uptime:** The total time within any 24-hour period where the site is accessible to visitors on the Internet. Anytime a host provider's server goes down, for whatever reason, domains on that server go offline, which is commonly referred to as *downtime*. An uptime of 100 percent is the ultimate goal of all host providers, but most will only guarantee a 99 percent uptime rate.
- ✓ **Hard drive space:** This is the total number of megabytes (MB) or gigabytes (GB) of space allotted for the domain on the host provider's server. To determine your Web site's hard drive space needs, multiply the number of pages by 30K and then factor in enough space to account for all the additional files required for the site, including all the graphics, documents, PDFs, and multimedia files. You may also be able to estimate the total number of megabytes for your site through your HTML code editor. For example, in Dreamweaver, you can select all the files through the expanded Files pane and read the total byte count in the status bar. Typical small sites can make due with as little as 500MB–1GB of space, whereas e-commerce sites can require upward of 30GB, depending on the number of products being sold.
- ✓ **User account:** Depending on the Web site's needs, the hosting plan can accommodate from one to several user accounts. Each account provides password-protected host server access to site management tools such as passwords, e-mail setup functionality, and billing information.
- ✓ **Data transfer and page views:** The data transfer and page view figures refer to the maximum allowable number of times that visitors can access the pages (that is, the text, graphics, and other content) on the hosted site within a given time frame, such as a 30-day period. If the site is very popular, there could easily be over 30 million page views in a month!
- ✓ **Web-based statistics:** Web-based stats can help site owners track the number of visitors to their site, including such details as the entry and exit URLs, the number of hits and page views, keyword analysis, and the number of returning versus new visitors.
- ✓ **Dedicated IP address:** Domains with a dedicated IP address will be hosted on their own servers as opposed to sharing a server with other domains. Dedicated IP plans are more expensive than Web sites that use a shared IP address, but they are also ultimately more reliable because a dedicated server can more accurately monitor its own Web traffic and provide faster server response times. A dedicated IP address can also sometimes be a requirement for sites that need an SSL certificate

depending on the host provider's setup, so be sure to ask about this if you intend to get an SSL certificate.

- ✓ **Domain pointer:** This feature, which may cost a few extra bucks per month, allows one domain to automatically reroute visitors to another domain. Domain pointers can often be useful when a business wants to provide for misspellings of a domain name so that anytime visitors try to view the misspelled domain, they're automatically directed to the correctly named site, such as `www.yahooo.com` pointing to `www.yahoo.com`.

Shop around, do your research, speak to friends and business associates, and make your decision. After you choose a host provider, just sign up for the desired hosting plan. If you're also registering a domain for the first time when signing up for the plan, the site should be ready for use right away. If the domain was registered elsewhere, you can do the DNS transfer from the registrar to the host's servers as soon as you are ready to publish the site. You (or your client) can also set up e-mail boxes and adjust them at any time after the plan is paid for. Later, if you (or your client) are not happy with a particular host provider, you can always switch to another provider at any time.

The only thing left to do now, until the new site is ready for publishing, is to design, build, and upload a placeholder page, as described in the next section.

Creating a Custom Placeholder Page

A *placeholder page* is exactly what it sounds like: It's the default home page that visitors will see at a particular Web address — your domain name — when you (or your client) have both registered the domain and set up a hosting plan, but have not yet published the new Web site there.

The default placeholder page provided by the host provider can vary. Sometimes that page may identify the domain name and IP address of the domain, and sometimes it doesn't. More often than not, what you will see are some instructions to the site owner on how to access the host's servers to manage the new hosting account. The rest of the page is usually filled with information and links to the services of the domain registrar or hosting company where the site is parked, as in the example from LunarPages.com shown in Figure 1-6.

Fortunately, the domain registrar or host provider placeholder page will only stay online until it is removed or overwritten by either you or the site owner (or the Webmaster or whomever else the client might hire to manage the site after you design it). The smartest thing to do, then, is to take advantage of this paid-for open advertising space and design a customized placeholder page for the domain that can sit there until the new site is ready to publish.

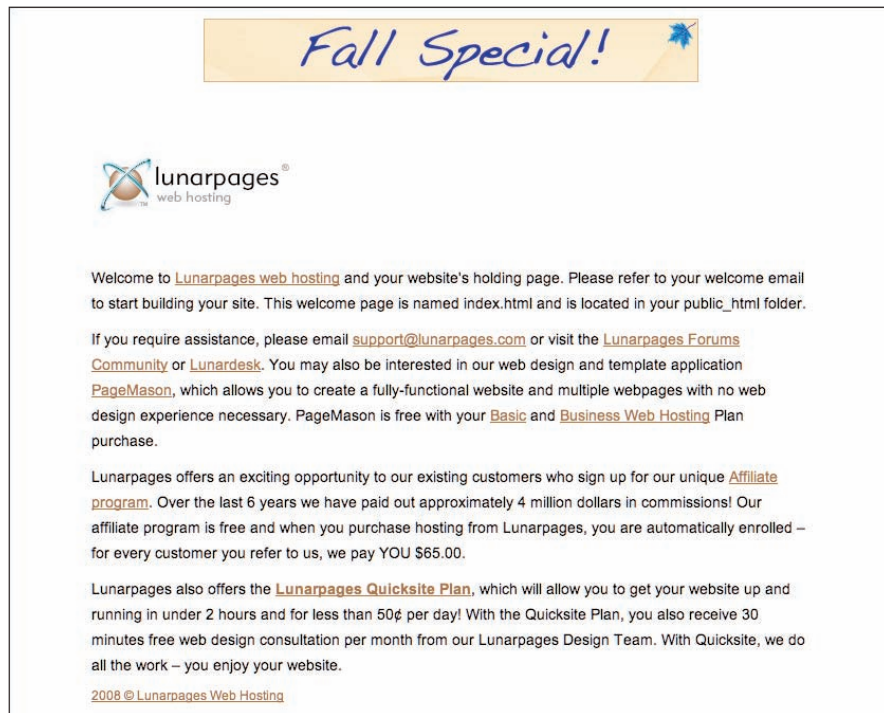


Figure 1-6: A host provider placeholder page often provides site setup tips to the site owner but no information to potential visitors about products or services.

The best custom placeholders are simple HTML Web pages that identify the site (logo, name, tag line, and other branding) and offer a means of contacting the site owner by both snail mail and e-mail. Anything else on the page, like a blurb about the company or some recent news items, is gravy. Figure 1-7 shows an example of a typical placeholder page that includes company name, an encrypted contact e-mail address, and general information.

In the following sections, you find out how to design your own placeholder and then publish it.



Figure 1-7: A customized placeholder page can include the company logo, an (encrypted) e-mail address, and a statement about the company's services.

Designing a placeholder page

To create your own custom placeholder page, you need access to the following things:

- ✓ A text or HTML code editor to build the placeholder page
- ✓ The site owner's company name/logo in GIF, PNG, or JPG format
- ✓ The descriptive statement about the site that will be included on the placeholder page
- ✓ The site owner's e-mail address, which should use the new domain name rather than a personal e-mail address from AOL, Gmail, AT&T, or Yahoo!, for example

The following two sections show you how to create a customized placeholder page styled with internal CSS.

Creating the page

Follow these steps to create the placeholder page:

1. **Create a new folder on your computer desktop called Placeholder, and inside that folder, create another folder called images.**

The placeholder page and image you're about to create will be saved to this folder structure.

2. **Save the company logo to your new images folder inside the Placeholder folder on your desktop.**

If you want to follow along with the example, point your browser to www.dummies.com/go/webdesignaio and download a copy of the logo graphic named Luau-a-go-go (`luauagogo.gif`).

To save a copy of the graphic, as shown in Figure 1-8, right-click (Windows) or Control+click (Mac) the image and choose Save (This) Image As. This opens the Save As dialog box, where you can choose the save-to location.



3. **Using your preferred code editor, or your computer's text editing program, open a new blank document.**

To use your computer's default text editing program, choose Start⇨All Programs⇨Accessories⇨Notepad if you're using a PC, or on a Mac, launch your Applications folder and double-click the TextEdit icon.



Figure 1-8: Use this logo to create a sample placeholder page.

A new untitled document should open automatically. If that doesn't happen, choose File⇨New to open a new file.

4. **If you're using a text editor, type the following basic HTML page structure, including the HTML 4.01 Transitional DTD and Content-Type meta tag:**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <title></title>
  </head>
  <body>
  </body>
</html>
```

Otherwise, if you're using an HTML code editor, which automatically drops in the structural code for you, skip ahead to Step 5.

5. Between the opening and closing <title> tags, type a title for the page (for example, Luau-a-go-go).

Your code should now look like this:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <title>Luau-a-go-go</title>
  </head>
  <body>
  </body>
</html>
```

This code sets the title for the page, which appears in the browser's title bar.

6. Between the opening and closing <body> tags, type the company name, a descriptive statement or tag line, and an e-mail address (which you convert into an encrypted e-mail address in Step 10).

For example, type the following bold text, making sure to add the paragraph <p> and break
 tags where indicated:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <title>Luau-a-go-go</title>
  </head>
  <body>
    <p>Luau-a-go-go</p>
    <br>
    <h1>Hawaiian Themed Catering</h1><br>
    <p>Luauus * Special Events * Birthdays * Anniversaries *
    Celebrations<br>
    Santa Monica, CA<br>
    For further information contact<br> info@luauagogo.com</p>
  </body>
</html>
```

The <body> tags hold the text and other content that appear in the browser window.

7. Choose File → Save to open the Save As dialog box.

You need to save the document to your new Placeholder folder.

8. In the File Name field, type index.html; in the Save In field, select the Placeholder folder; and in the Save as Type field, select the All Files option. Then click the Save button.

This document does not include any logo graphic yet, so you'll need to modify the code.

9. Delete the line of code that says `<p>Luau-a-go-go</p>` and replace it with the following line of code, which inserts the logo graphic onto the page:

```

```

Replace the text in italics if you're using your client's logo.

10. Convert the e-mail address into a working hyperlink using an e-mail encryption service (instead of the regular `mailto: e-mail link`, which is vulnerable to spambots), like the one found at `www.dynamicdrive.com/emailriddler`.

Your page code should now look something like this:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Luau-a-go-go</title>
</head>
<body>

<br>
<h1>Hawaiian Themed Catering</h1><br>
<p>Luauas * Special Events * Birthdays * Anniversaries *
Celebrations<br>
Santa Monica, CA<br>
For further information contact<br>
<script type="text/javascript">
/**]
/*****
* Encrypt Email script- Please keep notice intact
* Tool URL: http://www.dynamicdrive.com/emailriddler/
* *****/
&lt;!-- Encrypted version of: info [at] *****,*** //--&gt;
var
    emailriddlerarray=[105,110,102,111,64,108,117,97,117,97,103,111,103,
    111,46,99,111,109]
var encryptedemail_id62='' //variable to contain encrypted email
for (var i=0; i&lt;emailriddlerarray.length; i++)
encryptedemail_id62+=String.fromCharCode(emailriddlerarray[i])
document.write('&lt;a
    href="mailto:'+encryptedemail_id62+'"&gt;'+encryptedemail_id62+'&lt;/a&gt;')
/*]]&gt;*/
&lt;/script&gt;&lt;/p&gt;
&lt;/body&gt;
&lt;/html&gt;</pre>
</div>
<div data-bbox="122 797 209 868" data-label="Image">
<img alt="REMEMBER icon: a hand with the index finger pointing up, surrounded by a yellow circle with the word 'REMEMBER' above it."/>
</div>
<div data-bbox="262 814 843 849" data-label="Text">
<p>For more on e-mail encryption, see the sidebar “Protecting your e-mail addresses from spam” in Book III, Chapter 1.</p>
</div>
<div data-bbox="232 855 809 889" data-label="List-Group">
<p>11. Save the changes to your file and preview the page in a browser window.</p>
</div>
```

To preview the page in a browser, drag and drop the icon of your new `index.html` page into any open browser window. No Internet connection is required to preview the page locally.

The page looks okay, like the one shown in Figure 1-9, but it could definitely benefit from a little styling, as described in the next section.

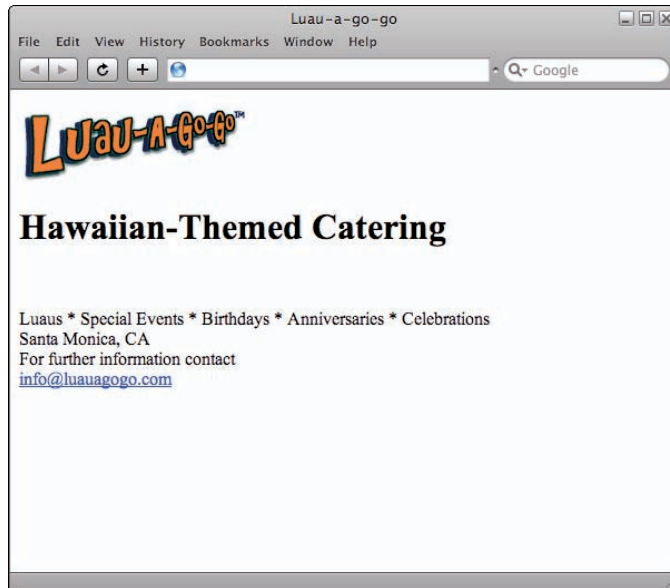


Figure 1-9: Before you add CSS styling, a placeholder page can often look bland with just graphics and text.

Styling the page

Follow these steps to add CSS styling to the placeholder page:

- 1. Above the closing `</head>` tag in your code, type the internal CSS markup you want to use on your page. (See Book III, Chapter 3 for more on CSS.)**

If you're following along with the example, type the following markup:

```
<style type="text/css">
<!--
#luauagogo {
  height: 250px;
  width: 500px;
  margin-right: auto;
  margin-left: auto;
}
```

```

margin-top: 15%;
margin-bottom: auto;
text-align: center;
font-family: Georgia, "Times New Roman", Times, serif;
font-size: 12px;
color: #F60;
padding-top: 20px;
padding-right: 0px;
padding-bottom: 0px;
padding-left: 0px;
background-color: #FFF;
border: 1px dashed #0CC;
}
a:link {
    color: #0CC;
}
-->
</style>

```

This CSS markup contains instructions that tell the browser to do the following: Center the content in the browser window, style the text in the Georgia font in 12px and an orange color with the hexadecimal color of #ff6600 to match the logo, add a blue dashed border around everything, and use blue as the link color to match the border.

Before this CSS markup can work, you must apply this style to the page's content.

- 2. Wrap a pair of DIV tags around the page content. Put the opening <div> tag directly after the opening <body> tag and the closing </div> tag directly above the closing </body> tag.**

DIV tags are container tags that can be styled and positioned with CSS when you include the `id` attribute in the opening tag that matches the name of the style in the CSS markup.

- 3. Add the attribute `id="uniqueid"` (for example, `id="luauagogo"`) to the opening <div> tag.**

Replace `uniqueid` with the ID of your choice.

The body part of your code should now look like this:

```

<body>
<div id="luauagogo">
  
  <br>
  <h1>Hawaiian Themed Catering</h1><br>
  <p>Luau * Special Events * Birthdays * Anniversaries *
  Celebrations<br>
  Santa Monica, CA<br>
  For further information contact<br>
  <!-- encrypted email address code here --></p>
</div>
</body>

```

4. Save the file again to save the changes you just made and preview the page in a browser window.

Figure 1-10 shows what the page example should look like.

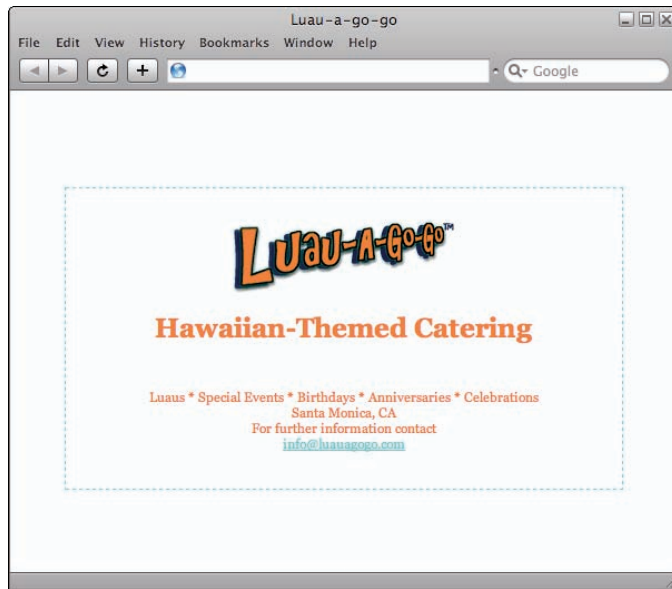


Figure 1-10: Use CSS to give your placeholder page more style and personality.



When creating your own placeholder page, feel free to generate as many graphics and other CSS styles as needed to make the page look exactly as you'd like it to. If you've already created a mock-up for the Web site, you may even want to use some of the same design features within the placeholder page. When the page is finished, you need to upload the page to the host server so that the page can brand and identify the domain until the new site is ready to publish.

Uploading a placeholder page

When a custom placeholder page is created for a Web client, be sure you get your client's approval on the page before publishing it. This gives the client the opportunity to review your work and suggest any changes that need to be made. After all, this page represents the client's company and she will want to put her best face forward. If the client does recommend some changes, make any adjustments to the code as needed and then resubmit

the page for approval. When the client grants approval, you can then upload the placeholder HTML file, the `images` folder that contains any graphics, and any other files needed to support the page, to the host server.

To transfer the placeholder page and supporting files to the host server, you can use the host provider's control panel, a stand-alone File Transfer Protocol (FTP) program, an FTP program that is built into an HTML editor (Dreamweaver has one), or a browser that supports FTP. You discover more about FTP in Book V, Chapter 2, so for now, I discuss how to upload your files through a browser.

Follow these steps to transfer your local placeholder files to the remote host server using Internet Explorer:

1. Get the FTP address, username, and password for the domain from the host provider or from your Web client.

This information is typically sent by e-mail to the person who signed up for the hosting account. The FTP address should be something like `ftp.domainname.com`.

2. With a live Internet connection, open Internet Explorer, type the FTP address into the browser window's address bar without the `http://` protocol (as in `ftp://ftp.domainname.com`), and press Enter (or Return on the Mac).

A Log On As dialog box, like the one shown in Figure 1-11, should appear, inside which you can enter your FTP access information.

3. Enter the site's username and password in the appropriate fields and click the Log On button.

To save the logon information, select the Save Password check box before clicking the Log On button.

The browser window then automatically refreshes and displays all the files on the host server for the domain. Those files will probably include a default placeholder home page named `index.html`, a CGI-BIN folder for processing scripts, and possibly a few other preinstalled files and folders that the host server requires to make the site accessible to visitors.



Figure 1-11: Enter the FTP username and password in the Internet Explorer Log On As dialog box.

4. **Drag and drop a copy of the Placeholder folder's index.html file along with a copy of the images folder into the open IE browser window.**

Your browser begins copying the files to the remote host.

5. **If prompted to overwrite the index.html page, click the Yes button so that your new custom placeholder page will appear as the new default home page.**

When the files have fully been copied to the server, you are finished *ftp-ing*.

6. **To test the success of the file transfer, type the URL of the domain into the browser's address bar, such as `http://www.mydomain.name.com`.**



Your new placeholder page should appear. If you do not see it, try refreshing the browser window (usually by pressing F5) and/or clearing your browser's cache. For simple directions on how to clear your browser's cache, visit www.bnl.gov/itd/webapps/browsercache.asp. If you still do not see your placeholder page, your server might use the default.html homepage naming convention. To test that theory, rename index.html to default.html and upload the file to the server. The new placeholder page should appear if your theory is correct.

If you still have any difficulty with the FTP process or the viewing of your placeholder page, contact the host provider for the domain for assistance.

Chapter 2: Publishing Your Site

In This Chapter

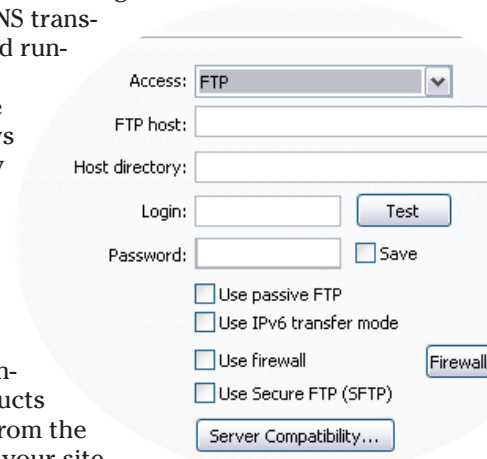
- ✓ Finding out about FTP programs
- ✓ Establishing a remote connection
- ✓ Testing files in a test directory
- ✓ Transferring files with FTP
- ✓ Publishing your site

It's finally time to publish your site! At this stage, you have done quite a bit of work. You've planned, organized, and gathered information for your site; designed a mock-up; optimized all the graphics; built out all the pages; tested and validated an entire Web site; and registered a domain and secured a hosting plan. Now you are truly ready, at long last, to share your site with the world. To officially publish your site and get it online for all to see, you need to transfer all the files that make up the site — that is, all the HTML files, images, CSS, external JavaScript files, SSIs, media files, and any other documents for files that are accessible through the site — to the remote server that is hosting the site.

If you have registered a domain but have not secured a hosting plan, now is the time to do that and put in for the DNS transfer because you'll need the hosting plan to be up and running before you can transfer files to the server.

Otherwise, if the hosting plan is ready, go dig up the information the host provider sent to you that shows the plan's username and password and includes any special instructions about FTP (File Transfer Protocol) and transferring files to the host's remote server. FTP is the most common way to transfer files to a remote server, so that's what I discuss in this chapter.

In addition to finding out how to set up a remote connection to a host server with FTP, this chapter instructs you on how to transfer your local files both to and from the server, create a test directory on the server, upload your site to the test directory for a final round of testing, and finally, upload the site to the root level of the remote server to officially publish the site on the Web.



The image shows a screenshot of an FTP client configuration window. The window has a light blue background and contains the following elements:

- Access:** A dropdown menu with "FTP" selected.
- FTP host:** An empty text input field.
- Host directory:** An empty text input field.
- Login:** An empty text input field with a "Test" button to its right.
- Password:** An empty text input field with a "Save" checkbox to its right.
- Use passive FTP:** A checkbox.
- Use IPv6 transfer mode:** A checkbox.
- Use firewall:** A checkbox.
- Use Secure FTP (SFTP):** A checkbox.
- Server Compatibility...:** A button.
- Firewall:** A button on the far right edge.

Uploading Files with File Transfer Protocol

File Transfer Protocol, which you can call FTP for short, refers to a standard TCP/IP Internet protocol that allows the exchange of files between remote computers over the Internet. To initiate an FTP session, a client (you) must use special software or some kind of Internet interface to log on and gain access to the remote server. Logging on typically requires the input of a special username or ID and a password that the host provider furnished when you (or your client) purchased the hosting plan. For example, if your name is Mary Miller and your site is called MillerCheeseSticks.com, your host provider might automatically generate a username/ID and password for you, such as mmillmiller and zc79ole7. Not all host providers generate the username and password combo for you. Some provide you with temporary account information and the opportunity of resetting your username/ID and password to something else after you log on to your site.

After access to the remote server has been established for the FTP session, you may begin getting (downloading) and putting (uploading) files between your local computer and the remote server. Remember, the remote server is the live host, which means that as soon as files are copied onto the remote server, they're publicly accessible on the Internet! When you are finished transferring your files, to end the FTP session, log off or otherwise disconnect from the remote server. The whole process is surprisingly simple.

Choosing the right FTP program

You can use many different FTP applications to transfer your files. Although their interfaces may be somewhat different, most FTP applications allow you to do the same things with your files, such as viewing a listing of files by name, date, and size and allowing you to transfer, copy, rename, and delete files and directories on the remote server.

FTP programs come in four different flavors; a stand-alone software application, an integrated feature of another software program, a tool on a host provider's Web site control panel, or a component of a browser interface:

- ✓ **Software programs:** Stand-alone software programs, such as WS-FTP or Fetch, must be installed on your local computer and launched like any other program each time you need to access the remote server. You may use the same program to access as many sites as you like, as long as you have the correct username and password combination for each domain. Each site can have its own "server profile." Saved profiles archive the FTP URL, username, and password information to make future logons run faster.

Though some FTP programs have a drag-and-drop interface where you can drag files from your local desktop into the remote view of the host server, most programs consist of a single window with two panes that represent views of the local site files and the remote site files, as shown in Figure 2-1. Data may then be transferred both to and from the remote server using common interface controls such as Get, Put, Change Directory, Make Directory, Rename, Delete, and Refresh.

- ✓ **Integrated application:** Some “FTP clients” are built-in components within other software programs that allow you to transfer files to and from a specified remote server through a special FTP panel. For example, Dreamweaver’s Files panel can be used as an FTP tool. When you expand the Files panel, you can even see both the remote and local views of the files being transferred and transfer files in either direction.
- ✓ **Internet control panel:** Your host provider may include some kind of special Internet control panel through which you can transfer files to and from the host server. These panels are often customized Web interfaces developed by host providers and are composed of a handful of specialized Web forms. These forms allow the host’s customers (you or your client) to upload files to the remote server and occasionally to also select and download files from it. Most control panels restrict uploads to single files at a time, rather than enabling users to specify and upload several files or folders at once. Control panels also rarely let clients have full control over the files on the remote server and may even restrict access to certain tasks such as renaming and deleting files.

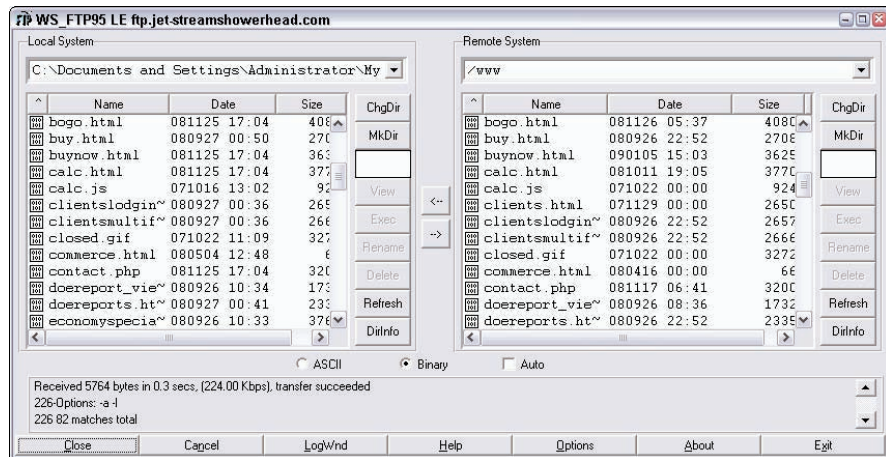


Figure 2-1: The stand-alone WS-FTP program has a single, two-paned window for displaying local and remote files.

- ✓ **Browser interface:** Many browsers include an FTP client interface that allows you to access the remote server with a simple Internet connection from your local computer. To establish a connection to the remote host, you simply enter a special FTP address into the address bar, after which time the browser prompts you to enter a username/ID and password. That information is then passed to the remote server and authenticated when the logon information is correct. After the connection is established, the same browser window is used to display the files and directories on the remote server into which you may drag and drop files from your local computer.

If you are using Dreamweaver or some other HTML coding application that has its own built-in FTP tool, feel free to use that tool to transfer your files to and from the remote server. On the other hand, if you're looking for a stand-alone FTP application, Table 2-1 lists the names and URLs of some of the better software programs for both Mac and Windows platforms (my personal favorites are WS-FTP and FileZilla). Take a few minutes to visit each of these sites and download the software application that appeals the most to you.

<i>Program</i>	<i>URL</i>	<i>Est. Cost</i>	<i>OS</i>
FileZilla	http://filezilla.sourceforge.net	Free	Win/Mac/Linux
WS_FTP	www.ipswitch.com/products/filetransfer.asp	\$69	Win
CoffeeCup Free FTP	www.coffeecup.com/free-ftp	Free	Win
FlashFXP	www.flashfxp.com	\$25	Win
SmartFTP	www.smartftp.com	\$36	Win
CuteFTP	www.globalscape.com/products/ftp_clients.asp	\$39	Win/Mac
Cyberduck	http://cyberduck.ch	Free	Mac
Fetch	www.fetchsoftworks.com/downloads.html	\$25	Mac
FTP Client	www.ftpcient.com	\$35	Mac
Fugu SFTP	http://rsug.itd.umich.edu/software/fugu	Free	Mac
RBrowser	www.rbrowser.com	Free/\$35	Mac
Yummy FTP	www.yummysoftware.com	\$25	Mac

Setting up a remote connection

After you have chosen your method of FTP (which may include purchasing, downloading, and installing a stand-alone application), your next step is to set up the remote connection to the host server. This process involves configuring the FTP client with a session profile for the connection to your domain. Profiles normally include the following bits of information:

- ✓ The URL of the domain being accessed or a special FTP address provided by the host, as in `www.mywebsite.com` or `ftp.mywebsite.com`
- ✓ The host-provided username and password to access the site via FTP
- ✓ When applicable, additional host-related information that may be required by the FTP application or server to assist with making the connection to the remote server, such as the host type, account name, initial remote directory, and firewall information

To illustrate how this works, take a look at the interface for configuring a new session profile in WS_FTP in Figure 2-2. Before a connection to the remote server can be established, each FTP session profile requires a profile name, host name/address, host type (which if unknown can be set to Auto Detect), user ID, and password.

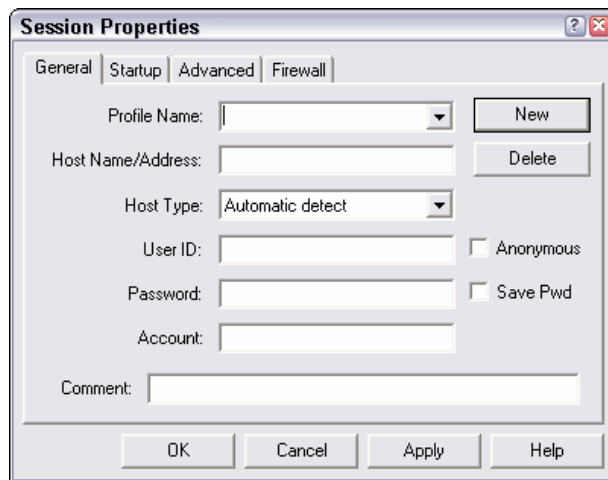


Figure 2-2: FTP sessions require a host address, user ID, and password before allowing access to a remote server.

By contrast, Dreamweaver users must manage a site and set up the remote access information before being able to use the program's built-in FTP tools. Follow these steps to set up a remote FTP connection in Dreamweaver. If you are using another coding editor for FTP, you should be able to easily adapt most of these steps.

1. Launch Dreamweaver and choose Site→Manage Sites from the program's main menu.

The Manage Sites dialog box opens. If you have already managed a site for the files you intend to transfer, that site's name should appear in the dialog box. However, if you haven't managed a site for this Web project, you must manage a site before continuing.

To create a managed site, click the New button in the Manage Sites dialog box and choose Site from the drop-down menu. Select the Advanced tab, and in the Local Info category, enter a name for the site in the Site Name text box and the location of the site on your local computer in the Local Root Folder text box, as illustrated in Figure 2-3. When finished, skip ahead to Step 3.

The name you give your site is solely for your own use and does not appear anywhere on the published site, but it does help you identify the site by name within the Manage Sites dialog box. The local root folder tells Dreamweaver where to find the files for this site so that it can perform special functions like site-wide updates.

2. Select the desired managed site from within the Manage Sites dialog box and click the Edit button.

This opens the Site Definition dialog box for the selected managed site, showing the Advanced tab, Local Info category.

3. In the Category section on the left of the dialog box, select the Remote Info option. On the right, choose FTP from the Access drop-down menu.

The dialog box now displays FTP configuration fields, as shown in Figure 2-4. The Access setting defines the protocol by which files will be transferred between your local computer and the remote server.

4. In the FTP Host field, enter the host address where the files will be uploaded to.

This can be either the domain name preceded by `www` or `ftp`, such as `www.mydomain.com` or `ftp.mydomain.com`, or the domain's IP address (which should have also been given to you by the host provider).

If you're not sure which one to enter, refer to the information about FTP access furnished by the host provider or system administrator.

5. (Optional) If your host provider requires the additional input of a host directory, enter that information in the Host Directory field.

A *host directory* is the location on the remote server where files for your domain will be kept. Typical host directories are often `www`, `public_html`, or some specialized name such as `/b52/domainname`.

If this information is required for FTP access on your host server, your host provider would have furnished this information to you along with the user ID and password. Therefore, if you don't have (or think your site doesn't have) a host directory, leave this field blank.

6. Enter the Login and Password information for the remote server.

Unless otherwise specified by your host provider, the Login should be the same as the host-provided username or user ID.

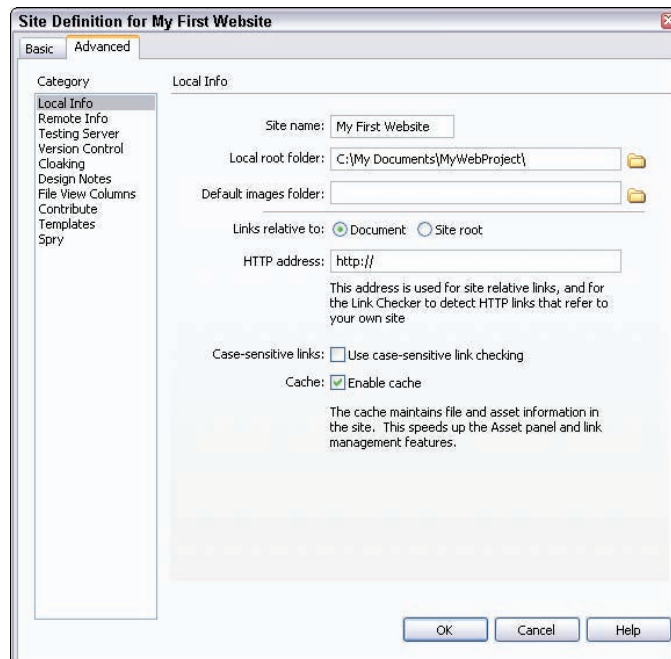


Figure 2-3: Dreamweaver users must set up a managed site before configuring the connection to the remote host.

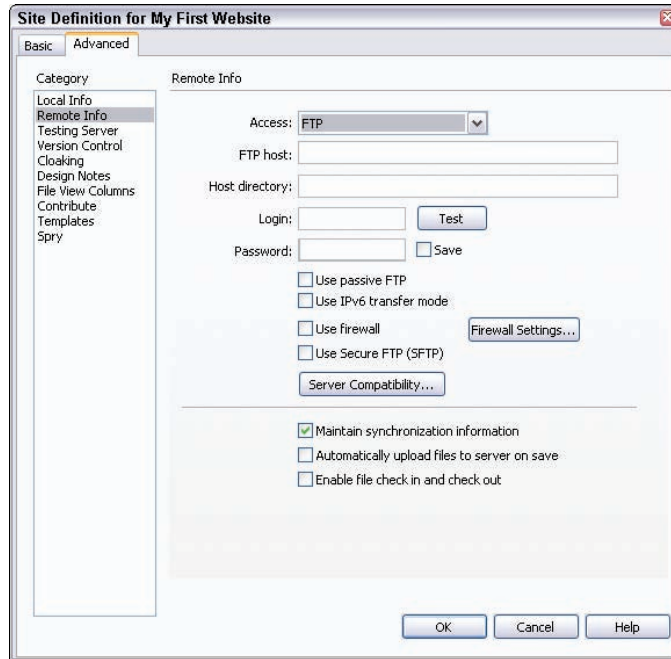


Figure 2-4: You can select FTP as the remote access method in the Remote Info area of this dialog box.

7. Click the **Test** button to verify the connection to the remote server.

If the remote information you have entered is accurate and your computer can make an FTP connection to the remote server, Dreamweaver will display a **Connection Established** success message. Congratulations! You may proceed to Step 8.

If, on the other hand, your computer fails to make a connection to the remote server, you will see an error message. Go back and check the spelling of all the information you have entered in the dialog box to ensure that the information is accurate. Typos and incorrect letter case can prevent the connection from being established. Test again until a connection is established.

If you are sure that you have entered the Login and Password information correctly, try successively using passive FTP and/or some of the other options available in the dialog box, such as Use Firewall and Use Secure FTP, and click the Test button after each configuration modification to see whether you can establish a connection to the server.



- *Use Passive FTP* tells the client to use the local computer rather than the remote server to establish a connection.
- *Use Firewall* settings allow you to custom-configure the FTP connection, hosting preferences, and transfer options.
- *Use Secure FTP (SFTP)* uses encryption for a totally secure connection.

After a connection is established, Dreamweaver displays a `Connection Established` success message, and you can proceed to Step 8. If, despite your attempts, you still can't establish a connection to the remote server, contact your host provider or system administrator for assistance. Sometimes resetting the password allows you to establish a connection.

8. To save this configuration so that you may use this remote connection in the future, select the Save check box next to the Password text box.

This is a very useful feature because it allows you to forget about the username and password and establish a connection to the remote server more quickly when transferring files in the future.

9. The remote site configuration is complete. Click the OK button to close the Site Definition dialog box and click the Done button to close the Manage Sites dialog box.

If you've decided to use a different FTP client than Dreamweaver, take a moment right now to configure your chosen FTP client with the appropriate session profile details for your Web site. Most FTP tools allow you to either test the connection or make a live connection to verify that your session profile information is accurate. Should you need any help setting up or troubleshooting the connection, consult the FTP client help files and contact your host provider or system administrator.

After you have established a successful remote FTP connection with your domain's host computer, you are ready to find out how to transfer files between your local computer and the remote host. In the next section, you find instructions on setting up a test directory, transferring files, performing last-minute testing, and publishing your site.

Setting Up a Test Directory

Though you are undoubtedly very excited about publishing your Web site, you have one more task to perform beforehand. You are going to set up a test directory where you can upload your site to the hosting server for one final round of quick testing.

A *test directory* is a folder you create that sits at the root level of the remote server, as illustrated in Figure 2-5. By uploading a copy of all the site files from your local computer to this remote test folder, you can review the remote files on the Internet, as though they are “live,” for the final round of testing. This keeps the customized placeholder page (`index.html`) intact and visible to any potential visitors to the domain until you’re ready to publish the site. (See Book V, Chapter 1 for the lowdown on creating a placeholder page.)

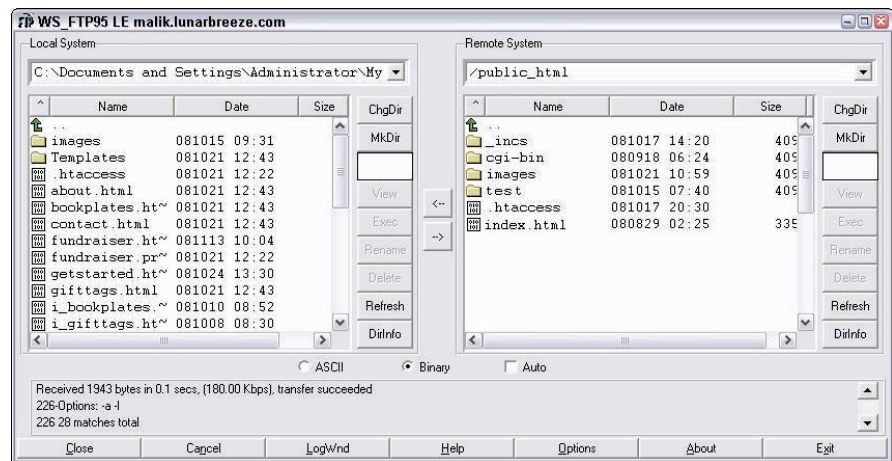


Figure 2-5: Test a site in a live environment by placing a copy of it in a test folder on the remote server.

To create a test directory on your remote server, follow these steps:

- 1. Launch your FTP client and establish a connection with the remote server.**

Though each FTP client is slightly different, they all should allow you to establish a connection by selecting the desired session profile and clicking an OK or Connect button.

- 2. Create a new folder with the name of your choice at the root level on the remote site.**

You should see an option somewhere in the FTP interface to make a new directory (often called `MkDir`) or create a new folder using the `File→New Folder` command.

Your test directory folder can be named anything you like, such as `test`, `temp`, `dev` (for development), `secret`, `lab`, `check`, `wip` (for work in progress), or `trial`. Alternatively, you could name the test directory after the abbreviation of the client's site, such as `RWMD` for Ryan West Marketing + Design.

While you are creating your test directory, you may notice that the host server already contains some files and folders. Most host providers display default home pages (usually called `index.html` or `default.htm`) that are used as placeholders for your domain until you're ready to publish your site. When those pages include graphics, you'll also likely see an `images` folder at the root level of the host server for your domain. If you have already designed and uploaded your own customized placeholder page to the server (as described in Book V, Chapter 1), you can see that file on the remote server too.

In addition to a default home page, many host accounts include a `cgi-bin` folder, inside which you can place script files to handle form processing and other programming. Anything else that you see there is probably required by the host provider for your site to be fully functional, so be sure to leave those files where they are. Creating a test directory should not affect those files, as long as you leave your customized `index.html` file and `images` folder in place at the root level until the fully tested site is ready for publishing.

Now that you have your test folder in position on the remote server, you are ready to discover the file transfer process, final-round testing, and publishing.

Getting and Putting Files

FTP file transfers are normally bidirectional, which means that you may send and receive files both to and from the remote server location. When you are transmitting your files to the remote server, you call that *uploading* or “putting” files, and when you are receiving files from the remote server, that is called *downloading* or “getting” files.

In most cases, the transfers will be putting copies of your local files on the remote server for testing and publishing purposes. From time to time, however, you may want to get a copy of some of or all the files from the remote server for your local computer to, say, restore a broken version of a file or to get a copy of the site onto a new computer. In either case, file transfers via FTP are a way of making copies from one location to another. Like a document through a fax machine, the original files always stay in one location while the copies are transmitted to the remote location.

Putting files on the remote server

To put a copy of your local site files into the new testing folder on the remote server using FTP, follow these steps:

1. Launch your FTP client and establish a connection with the remote server.

Ideally, your FTP interface should display and provide access to both your local files and your remote files, so you may need to configure your FTP tool to display both locations. If your FTP client has a drag-and-drop-style interface, have the folder to your local site on your computer open in a window right next to it so that you can easily drag and drop files between the two locations.

2. Open the test folder on the remote server by double-clicking the folder's icon.

This folder needs to be open so that the files can be transferred to it, rather than to the root level of your domain on the remote server.

3. Put (transfer) all the local files that make up your site, including all HTML files, CSS, JavaScript, images, and other assets, in the test folder on the remote server.

Depending on your FTP client interface, this step can involve a drag and drop of all the selected files on your local computer into the remote test folder, or the selection of all the desired files and the clicking of a Copy, Put, or Transfer button.



While transferring files in some FTP clients, you may be prompted to choose whether to also transfer *dependent* files. Dependents are any additional files associated with the HTML document(s) being transferred, such as images, media files, PDFs, SSIs, CSS, and JavaScript files. When the transfer involves sending information for the first time, including dependents is a good idea. However, when putting updated files on the server during final testing, it might be faster to not send the dependents, unless they've been modified too.

Transferring files with Dreamweaver

In Dreamweaver, you have a number of ways to transfer files through the Files panel:

- ✓ Use the file transfer buttons that display along the top of the collapsed Files panel.
- ✓ Use the FTP commands in the Files panel's options menu.

- ✓ Expand the Files panel and use any of the FTP buttons that display across the top of the Files panel window or any of the commands on the Files panel options menu, or drag and drop any selected files between the remote and local views of the managed site.

The smartest way to transfer files is to use the expanded Files panel so that you can view both the local and remote files at the same time, much like a stand-alone FTP client.

To transfer files to the remote server with Dreamweaver's expanded Files panel, follow these steps:

- 1. Launch Dreamweaver and select the desired managed site from the drop-down menu at the top of the Files panel.**

If you do not see your site listed here, you need to manage a site. Choose Site→New Site to open the Site Definition dialog box, inside which you can enter a name for the site in the Site Name text box and the location of the site on your local computer in the Local Root Folder text box. When finished, click the OK button.

- 2. Click the Expand/Collapse button in the upper-right corner of the Files panel to expand the panel, as shown in Figure 2-6.**

After the panel is expanded, the Files panel displays two separate panes, one for the local files and one to display remote files. By default, the local files display in the right pane and the remote files display on the left.

To swap the location of the local and remote files, collapse the Files panel by clicking the Expand/Collapse button and open Dreamweaver's Preferences by choosing Edit→Preferences (Windows) or File→Preferences (Mac). In the Site category of the Preferences dialog box, modify the Always Show and

On The drop-down menus to suit your particular needs, such as Always Show Local Files On The Left. Click the OK button to close the dialog box with your new settings and re-expand the Files panel before proceeding to Step 3.

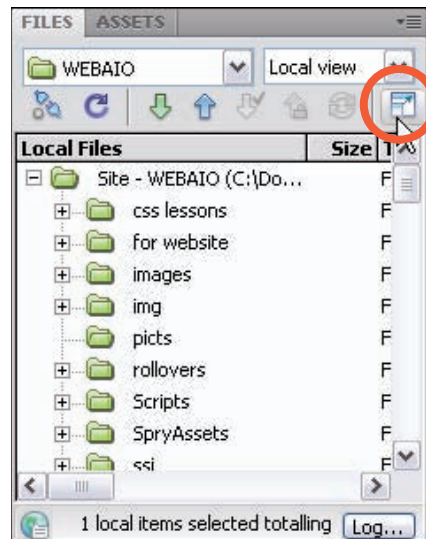


Figure 2-6: Expand the Files panel.



TIP

You will not see any files in the remote listing until you establish a connection via FTP.

- 3. Click the Connects to Remote Host button on the toolbar at the top of the expanded Files panel to establish a connection.**

The button looks like a blue plug and socket. When clicked, the plug connects with the socket, and a green light appears next to it, indicating that the connection was a success. Upon connection, you also see all the files on the remote server appear in the Remote Site pane of the Files panel, as shown in Figure 2-7.

- 4. To transfer files from the local site to the remote site, select the file(s) that you'd like to transfer from the Local Files pane and click the blue up-arrow Put button on the panel's toolbar.**

To transfer your files to a specific folder on the remote server, you must open that folder (by double-clicking it) before transferring files from the local pane.

To select more than one file at a time for a multiframe transfer, press and hold the Ctrl (Windows) or ⌘ (Mac) key while selecting each additional file.

- 5. Upon clicking the Put button, Dreamweaver displays a Dependent Files dialog box. Click the Yes button to upload dependent files or the No button to upload only the selected file(s).**

Depending on your Internet connection speed and the size of the files being transferred, the upload may take anywhere from one second to several minutes.

- 6. To end the FTP session, click the Connects to Remote Host button to disconnect from the remote server, and then collapse the Files panel by clicking the Expand/Collapse button.**

To reestablish a connection with the remote server for future transfer sessions, repeat Steps 2–6.



Dreamweaver's expanded Files panel can do more than just transfer files. You can also use it to sort and view both the local and remote files by size, type, and modification date. If desired, you can even set additional file attributes for the display by configuring the File View Columns category in Dreamweaver's Site Definition dialog box. Another useful feature of the expanded Files panel is the Synchronize command, which assesses both local and remote files and then transfers files to and from the remote site so that both locations contain the most recent version of each file in the managed local site.

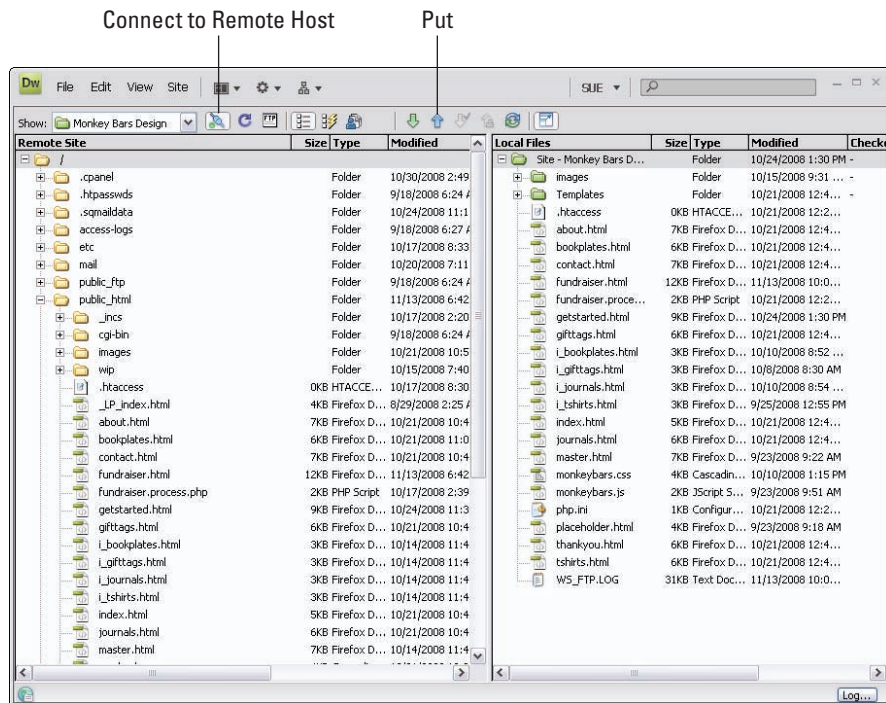


Figure 2-7: Create a connection to the remote server using the expanded Files panel.

Performing Final Site Testing

When you upload the site to the test directory on the remote server, you have one final opportunity to review all the pages before you make them accessible to the public. This gives you the time you need to do any testing and make last-minute corrections to your site.

Broken links, spelling errors, and missing images are often found during this final testing stage, so spend the few extra minutes, hours, or days, as the case may be, performing this most important second-to-last step before making your site live on the Internet. If you are doing a Web project for a client, have the client get involved with this final review of the site before publishing. After testing and correcting any errors you may find, you can confidently publish the site, remove the test directory from the server, and start on another Web project.



Although technically the pages are on the Internet as soon as they are uploaded to the test directory, no one else — including search engine spiders and robots — will know about the presence of these live, published files in this test directory unless you give them the specific Web address. Spiders and robots don't typically index sites unless you expressly request indexing or unless one or more of the pages is automatically indexed during a regularly scheduled crawl. For that reason, it is best to delete the test directory from the remote server as soon as possible after publishing the site.

To access the files on the Internet within the domain's test folder, enter the domain name and test directory in your browser's address bar. For example, if your domain name is `pumpkinsticks.com` and the test directory you created is called `test`, you'd enter `http://www.pumpkinsticks.com/test` in your browser's address bar. If your test folder is not set up properly within the host's server, you may see an error message that you cannot browse the directory.

After the directory is set up, be sure to correctly enter the URL of your test directory in the address bar, paying special attention to letter case, because on some host servers, entering an incorrect address can prevent you from accessing your files. For instance, if your host server only displays pages when the path uses the correct letter case, you might be able to access your files with `http://www.pumpkinsticks.com/test` but not with `http://www.pumpkinsticks.com/TEST`. If you don't automatically see your home page (`index.html`), try entering the full path to that file, as in `http://www.pumpkinsticks.com/test/index.html`.

If you are designing a site for someone else, feel free to share the test URL with your Web client. This gives both you and the client one final opportunity to review the site prior to publishing. Though some issues may come up that require a pushback of the site publication by a few hours or days, it is far wiser to have more eyes previewing the site for errors before it goes live than to publish the site with even one missing image, glaring typo, or broken link.

If you do find a coding issue that needs correcting at this stage, update the file locally first and then send the updated file to the test folder by way of FTP. Then, after clearing your browser's cache, preview the updated page again in the test directory to ensure that the error was corrected to your satisfaction. If not, continue making adjustments and uploading the file again until the page looks and functions correctly.



For times when you provide a client with the URL to the test directory, be sure to have her review the entire site online and provide written approval of the site to you when she's satisfied with the work and agrees that the project is completed. Getting a signed Project Completion form officially marks the end of the Web project so that you can submit any final invoices to the client and confidently publish the site.

With luck, this final testing phase shouldn't turn up too many issues related to accessibility or validation (though of course if you find any issues like that, you should certainly fix them). Instead your focus should be on finding and fixing any interactive problems with links, graphics, and forms, creating any custom error pages as needed, and catching any last-minute typos that might have been inadvertently overlooked.

Unfortunately, there is one thing you can't test for within a test directory, and that's the functionality of any form processing that uses scripts copied into the remote `cgi-bin` folder, such as a Join Our Mailing List form on a Contact page with a forwarding Thank You page redirect. The reason that you cannot test pages with forms within a test directory is that those pages must reside at the root level of the site to process any scripts. To test form pages, then, put a copy of those pages with test filenames (such as `testform.html` and `testthankyou.html`) at the root level of the remote server and test them there. Even though these few files will be at the root level and are technically publicly accessible to anyone with an Internet connection, no one will know about the presence of these files unless you give them the URL. When you are at the root level, you can safely test and verify the functionality of these files in a live site environment while testing the rest of the site in a virtually hidden test directory.

Keep in mind that testing your files in a test directory works well only when you have been creating your links to files and graphics using document relative links (such as `images/logo.gif`) instead of site-root relative links (like `/images/logo.gif`) or hard coded links (such as `http://www.mysite.com/images/logo.gif`). For instance, if you happen to have used site-root relative links (which are most effective for sites built with SSIs), your graphics and links may not function properly in the test directory. In other words, your images will not appear and your hyperlinks will appear to be broken. To remedy this situation, you can go back into your files, modify the links in your code, and upload the files to the test server again for testing, or you set up a special live testing environment where you can safely test your site before making it live on the Internet. For specific instructions for setting up a live testing environment for your domain, check with your host provider.

Creating Custom 401 and 404 Error Pages

When a Web server receives a request from a browser that it doesn't know how to process, it typically returns one of several error messages to the visitor's browser window. Two of the most common errors are

- ✓ **401 Unauthorized Access:** This message gets displayed when people attempt without permission to access Web pages that are password protected.

- ✓ **404 File Not Found:** This message appears when visitors click a broken hyperlink or type an incorrect Web address in the browser's address bar.

You can customize these messages to match the design of your Web site, and with a little help from your host provider, you can have these custom error pages installed and in service on your domain within 24 hours or less after transferring the site files to the remote server.

Creating the error pages

The 401 Unauthorized Access page usually includes a short message to visitors along with the refresh meta tag that redirects visitors to the home page or some other location on the Web after a specified number of seconds (see Figure 2-8).

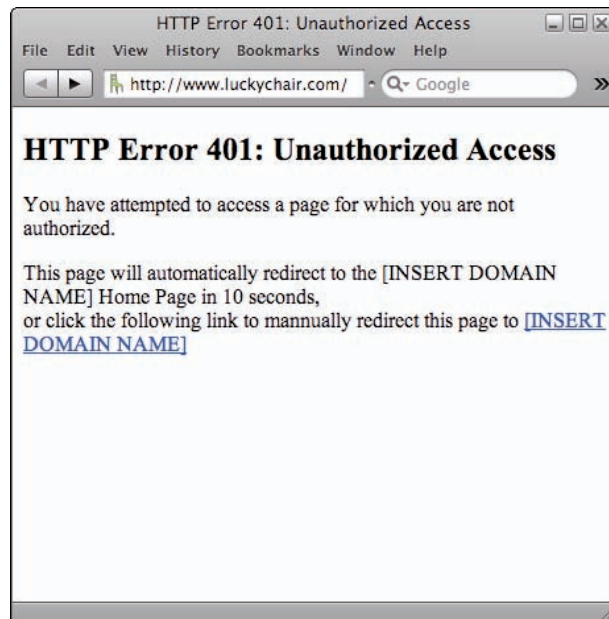


Figure 2-8: The 401 Error page is used for pages that visitors are not authorized to access.

The meta tag goes in the head of the page and should be formatted like this:

```
<meta http-equiv="refresh" content="10;URL=http://www.yourdomainname.com">
```

The message can say anything you like, but typically it says something like this:

You have attempted to access a page for which you are not authorized. This page will automatically redirect to the [INSERT DOMAIN NAME] Home Page in 10 seconds or click the following link to manually redirect this page to [INSERT DOMAIN NAME].

The 404 File Not Found message provides more space for a customized message and any additional information such as marketing and advertising, but generally should state something like this (see Figure 2-9):

The page you are seeking cannot be found:
The page you are looking for might have been removed, had its name changed, or is temporarily unavailable.

Please try the following:

- * If you typed the page address in the Address bar, make sure that it is spelled correctly.
- * Open the www.DOMAINNAME.com home page, and then look for links to the information you want.
- * Click the Back button to try another link.

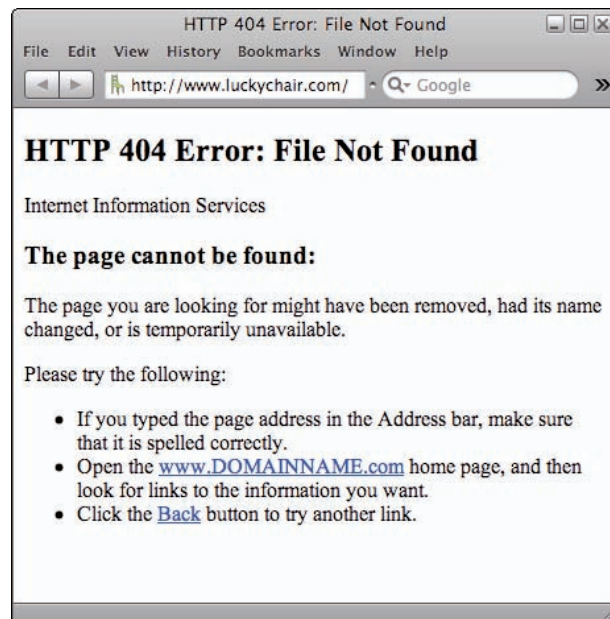


Figure 2-9: The 404 Error page is displayed when the URL cannot be found.

To customize your own 401 and 404 error pages, create two new Web pages based on existing pages on the completed site. This ensures that the new pages use the same layout and graphics as the rest of the site. Save these files with the filenames `error401.html` and `error404.html`, and then edit the content areas for both pages by using the preceding examples. If desired, add additional information to suit your site's particular needs. For example, rather than just displaying the URL of the home page on the 404 error page, why not also include links to all the pages on your site, similar to your site map page? The same technique would also work for the 401 error page. The customized content is entirely up to you, so be as creative and helpful as you can to the visitors (potential customers) that will be viewing these pages.

Editing the .htaccess file

Next, you need to edit the existing `.htaccess` file on your server (if one already exists) or create a new one. An `.htaccess` file is a hypertext file that provides directives to the server, such as password protection and serving error pages. This file needs to sit at the root level of your host directory to provide instructions to the server on how to serve up your new custom documents should either of these errors occur.

To see whether your server already has an `.htaccess` file, establish a remote connection to your server using your FTP client and take a look at the root level of your host server. If an `.htaccess` file already exists, download (get) a copy of it to your local computer so that you can make modifications to it. If you don't see an `.htaccess` file at the root level, create a new one.

To create an `.htaccess` file, follow these steps:

- 1. Open a text editor, such as Notepad or TextEdit, and type in the following two lines of code:**

```
ErrorDocument 404 /error404.html
ErrorDocument 401 /error401.html
```

- 2. Save the file as `htaccess.txt` to the root level of your local site.**
- 3. Establish a connection between your host server and your FTP client, and upload a copy of the `htaccess.txt` file to the root level of your host server. While still connected to the remote server, change the name of the `htaccess.txt` file on the remote server to `.htaccess` by removing the `.txt` extension and placing a period *before* the filename.**
- 4. Upload your customized `error401.html` and `error404.html` files to the root level of your host server.**

The final step, after creating your custom error pages and uploading them to the root level of your remote server, is to contact your host provider for further instructions on how to make these new files replace the server's default error message pages. The host provider typically needs to configure some software on the server end before the new pages will work. The provider might also request that you log on to your site's control panel or site utility and make some adjustments to your site's configuration.



To view examples of (and grab the source code for) both a 401 HTTP Protocol Unauthorized Access and a 404 HTTP Protocol File Not Found error HTML page, go to www.dummies.com/go/webdesignaio.

Publishing Your Site

To publish the site, establish an FTP client session and put a copy of the entire local site in the root level of the remote server, excluding any Template and Library folders that you may have generated to create the local version of the sites. These types of files are generated by your HTML code editor and are only necessary for site management within the editor; the files are therefore unnecessary for site functionality on the remote server. In addition to the main HTML files for your site, be sure to also upload any additional folders and files that support the functionality and presentation of the site, including images, media files, PDFs, SSI, CSS, and JavaScript.

Immediately after publishing the site, open a browser window with a live Internet connection to access the site using the domain's URL. If you see the home page when you enter the URL (such as `http://www.mydomain.com`), your site is finally *live!* Take this moment, now that all the files are in their final destination, to test all the site's pages one more time. Check any missing images, broken links, and anything else that might suddenly jump out at you as being not quite right. If you find any issues that need troubleshooting or fixing, fix them right away on your local version of the site and reupload the corrected files to the remote server as quickly as possible.

When you're absolutely, positively, 100 percent sure that the newly published site is fully functional, delete the test directory from the server. Deleting files can be as simple as selecting the directory (folder) within the FTP client session window and clicking a Delete button, or it can take a little more work. Some FTP clients require that you delete any contents inside a folder before the folder itself can be deleted. However you have to get it done with your FTP client, do it. As you'll recall, the purpose of deleting the test directory after publishing the site is to prevent any visitors, human or computer, from accessing and indexing those test pages. Then, in the event

that any of the test pages were indexed by search engines during your test period, should anyone attempt to access those URLs at a later date, your new customized 404 File Not Found error message page will display when those pages are not found by the server.

To maximize your efforts, turn to Book V, Chapter 3 to discover ways to improve search engine rankings with Search Engine Optimization and to keep your customers happy with site maintenance services.

Chapter 3: Search Engine Optimization and Site Maintenance

In This Chapter

- ✓ Understanding Search Engine Optimization (SEO)
- ✓ Seeing the benefits of ethical SEO
- ✓ Improving search engine rankings with HTML
- ✓ Submitting a URL to search engines
- ✓ Improving visibility and accessibility with a site map
- ✓ Performing site maintenance
- ✓ Keeping your site up to date

Your site is up and running — give yourself a giant pat on the back, take a deep breath, and get ready to do a few more things to make the most out of all your hard work. Although additional work is not an absolute requirement, you can apply a handful of other useful techniques to your site to make it even more search engine and visitor friendly. As an added benefit, when you do the tasks outlined in this chapter, your design services will be more valuable to your clients and can help you stand out from the competition.

This chapter explains several Search Engine Optimization (SEO) techniques that can help make your site more search engine friendly. In addition, you find out why every site should include an HTML Site Map page and how to create one. As a final note, the end of this chapter shows you how to perform regular Web site maintenance and suggests ways to help you keep your site content fresh and up to date.

[Jeanine Payer - Handcrafted Jewelry, Rings, Ne...](#)
Buy jewelry by **Jeanine Payer**. We believe jewelry is a
old world craftsmanship meets 21st century design.
[jeaninepayer.com/](#) - 13k - [Cached](#) - [Similar pages](#)

Necklaces	New Collection
Rings	Earrings
Bracelets	Holiday Picks
Men's	Objects

[More results from jeaninepayer.com »](#)

[Handcrafted Jewelry, Handmade Jewelry, Inscribed](#)
Buy handcrafted jewelry online from **Jeanine Payer**. Includes
engraved jewelry, hand inscribed jewelry, rings, bracelets, neck
[www.jeaninepayer.com/xcart/shop.php](#) - 12k - [Cached](#) - [Similar](#)

[Designer Jewelry, Jeanine Payer, Alexis Bittar, Me](#)
Tree Jewelry is an online jewelry boutique specializing in
Payer, Me & Ro, Alexis Bittar, Jes MaHarry, ...
[treejewelry.com/](#) - 27k - [Cached](#) - [Similar pages](#)

[Payer](#)

Understanding Search Engine Optimization

Search Engine Optimization (SEO) refers to any techniques applied to the code and/or content of a Web page that assists with the indexing of a site by search engines and the improvement of the site's ranking order within them. Because search engines often rely on computerized "bots" to crawl the Web and develop and maintain their indexes, many of the SEO concepts can be easily implemented by thinking strategically about the site's content, title tags, meta tags, page structure, and accessibility coding. When the site has been "optimized" for search engines, the site's URL can be submitted to search engines, which itself can help visitors find the site and improve the site's ranking within the search engine results.

Depending on your (or your client's) budget, you can do a range of things to increase both Web traffic and conversion rates. *Traffic* refers to the number of visitors to a site, while a *conversion rate* is the percentage of those Web visitors who actually complete a valid sale or other online transaction.

If you have somewhat of a budget, spending money on pay-per-click advertising can be very useful and profitable. Pay-per-click Internet advertising, or paid placement, is a method of site promotion whereby the advertiser (you) bids on the keywords or key phrases you think will be used by potential site visitors and then pays a fee anytime a visitor clicks an advertising link that leads back to the advertiser's site. Google offers the perhaps the most well-known pay-per-click service, Google AdWords, which displays your advertised URLs along the top or right edge in the Sponsored Links areas of the Google Search Results pages, like the example shown in Figure 3-1.

Additionally, if you are selling any products or services, conversion rates have been known to increase when a site does everything it can to easily facilitate a sale, such as having clearly labeled Add XXX Product to Cart buttons, making the checkout process easy to use, making clearly defined shipping and return policies available to visitors, and offering excellent customer service and sales support.

On the other hand, if you have no money set aside for marketing, it would be really smart for you to take a little time to make the site more search engine friendly. The more you know about SEO, the more valuable your services can be to your Web clients, whom often will want to rely on your knowledge and expertise to guide them through (or completely handle for them) the process of getting their site listed on search engines and attracting more visitors.

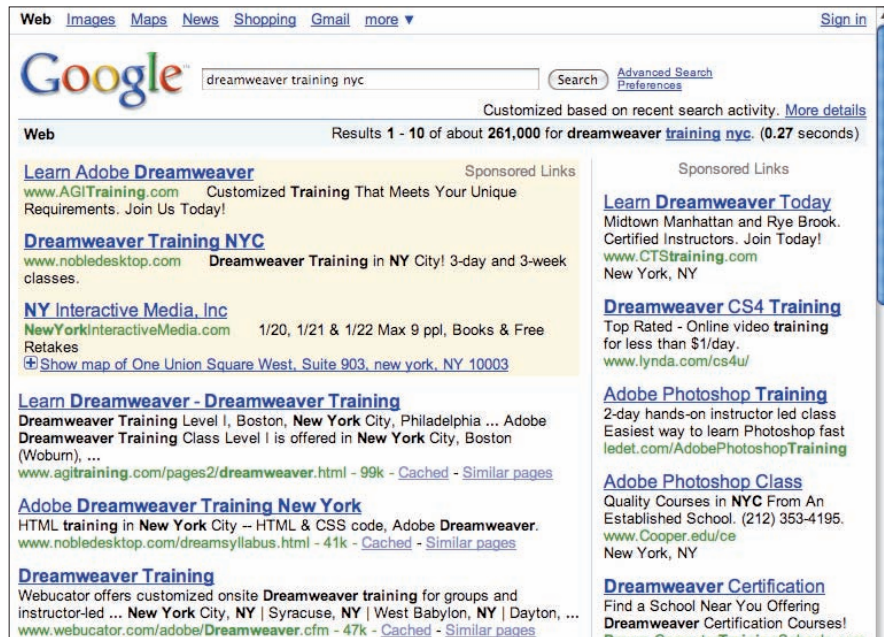


Figure 3-1: Advertise your site using Google AdWords, a pay-per-click system.

Practicing Ethical SEO Techniques

As with all things in life, there is a good way and a bad way to go about doing things, and when it comes to practicing Search Engine Optimization on your site, the same holds true. There is a good way to go about improving your search engine rankings, and then there's a bad way.

Back when the Internet was new, when there weren't so many Web sites and blogs, and search engines were something new, you could add a few different meta tags to your code and then sit back and watch as your site hit the number one or number two spot in the search engine rankings. Then, as more and more visitors began to rely on search engines that use Web robots and spiders to build their databases, a new breed of Internet scam artists was born. These so-called SEO service providers had one aim: to make money by tweaking the HTML code on a site that would trick those search engine robots and spiders into showing certain Web sites at the top of the search engine rankings. The motivation to use this type of deception was that if your site was listed on the first or second page of search engine

results, you'd get more site visitors, and therefore you'd sell more products and services. Of course, this hasn't proven to be true, but that hasn't stopped anyone from trying to cheat the system.

These sneaky tactics worked for a little while until the search engines devised a way to improve the Web robots and spiders to automatically detect and ignore sites that use common scams. As a matter of fact, some of these sneaky practices got so out of control that the search engines took even more drastic measures to ensure that their search results are as accurate and truthful as possible. As a consequence, it is no longer as simple as it once was to get good legitimate search engine rankings.

Because these questionable techniques fall into the realm of unethical practices, I don't discuss them in detail in this book. However, to educate you further about what kinds of things to avoid, the following list briefly outlines some of the more common unethical practices you should take extreme care to stay away from:

- ✔ **Keyword padding:** Do not engage in padding keywords into the meta tags or body content of Web pages that have nothing to do with the Web site's specific business.
- ✔ **Keyword listing:** Avoid listing keywords in the body of your site content when their purpose is not clear. The only legitimate place for a list of keywords is inside the keywords meta tag or when listing a site's products and services. If you do list words in this capacity, do it carefully so that it doesn't appear to be a spam-like keyword listing. A safer method would be to write copy that logically includes all the products and services in paragraph form.
- ✔ **Tag padding:** Do not use multiple versions of the same tag within the HTML code, such as duplicate meta description or title tags, to try to get more information through to search engines than you can with one tag. This, too, will be viewed as spam by search engines and will put the URL at risk.
- ✔ **Image padding:** Don't use words in your alternative (`alt`) text attribute for images that have nothing to do with the image being described.
- ✔ **Hidden text:** Don't add keyword-stuffed text to a page where the font color matches the background color of the page or other container tag that holds the text. Search engines can detect such slimy practices and treat those pages, and possibly all pages at the offending Web address, as spam.

- ✓ **Oversubmitting:** Never, ever submit a URL to any search engine, index, directory, or listing more than once in any 24-hour period. Daily submissions are also too often. Be realistic and submit only when significant changes have been made to the layout or content of a site. Besides, resubmitting your site is often unnecessary because, after your site has been indexed, the search engine periodically checks back on your site automatically.
- ✓ **Duplicate page submissions:** Do not submit pages with identical content but with different filenames. This will be viewed as spam and can put the Web address at risk.
- ✓ **Cheating:** Don't trick people into visiting a site by using inaccurate keywords, meta tags, and content, or by false advertising or unethical page redirects. Do not try to outsmart the search engines. The people who write the search programs are always on the lookout for cheaters, and if they find unethical SEO techniques on a submitted URL, they have the power to prevent the entire site from being indexed.



Be forewarned that should you choose to attempt any of these practices on your own (or a client's) Web site, you may be putting all the pages on that domain at risk of being indefinitely blacklisted by search engines. Besides, trying to trick the system is counterintuitive because most of the traffic coming to any Web site is often generated through search engines.

Be smart and do the right thing. To be a good Web designer today, you must educate yourself about the ethical ways to make a Web site visible to search engines and avoid unethical practices altogether.

Optimizing Your Site for Search Engines

Now that you know what not to do to improve search engine rankings, I now present some good, effective SEO techniques. To apply useful and ethical search engine optimization to your site, you can make several free and fee-based enhancements to your pages that can improve how the content inside them gets indexed by the Web robots and spiders that crawl the Web in search of new pages to add to their databases.

Robots and spiders, by the way, are the automated software programs that perform particular tasks, “crawl the Web” on a mission to find new Web pages, and then index the new pages in a large database. At minimum, when a new Web address is found, that page (usually the home page called `index.html`) is automatically added to that robot or spider's indexing

database. Furthermore, any hypertext links detected on that page may also be indexed automatically as part of the process. Because you have very little control over whether this indexing actually happens, you can at least prepare for the possibility of it by paying special attention to all the textual content and hyperlinks on your site, especially the home page.

To get you on the right track, the following sections describe some simple ways to improve the ranking of your Web pages in search results listings for search engines such as Google, Yahoo!, MSN, and others — all without spending any money.

Maximizing keywords

The first thing you can do to improve a Web site's search engine friendliness is to maximize your keywords. One of the things that search engine robots and spiders do is search for meaningful content within the text on pages that they crawl. This means that the sites you design need to help those bots find that content by including site-specific keywords that identify a company's products and services, especially on the home page.

Keywords can be any words or short phrases that describe the product, service, or information on the site that needs to be advertised. For instance, if your site sells hand-painted customized baby onesies and other baby items, tell the world about your products using clearly identifiable keywords that visitors might use. This can help visitors around the world using search engines to more readily find a particular Web site.

Keep in mind that the keywords within the text of the site's pages can be the same as any keywords listed in the keywords meta tag. However, because content keywords and key phrases can be integrated into the text in the body of the page, you have much more latitude for including the most popular keywords within the text that the site's target audiences are likely to use when doing a search engine search.

To evaluate your site's content and find ways to maximize keywords, look through each page on your site with the following key concepts in mind:

- 1. Verify that the text on each page — especially the home page — includes descriptive keywords and key phrases.**

For instance, if the Web site offers printing services and uses only chlorine-free 100% post-consumer recycled paper and vegetable-based inks, *Chlorine-Free 100% post-consumer recycled paper* and *vegetable-based inks* would be great key phrases to include in the page's text, as illustrated in Figure 3-2.

- 2. Find ways on every page to hyperlink any keywords or key phrases to other relevant pages on the site. If you see keywords that aren't yet linked to somewhere else on the site, add them.**

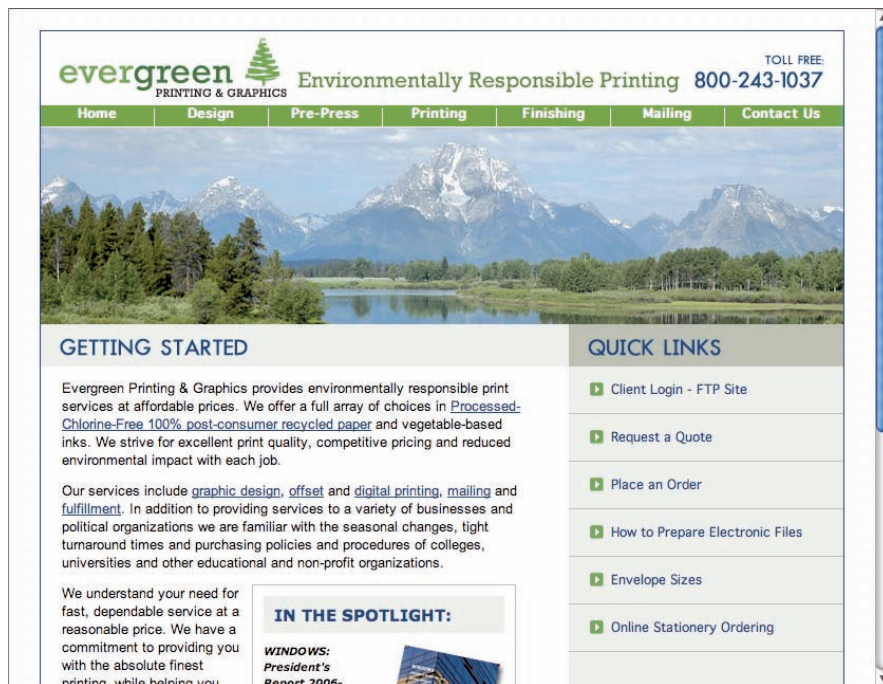


Figure 3-2: Maximize searchability by including keywords and key phrases in the text.

For instance, if the copy on a page includes the phrase, “Our services include graphic design, offset and digital printing, mailing and fulfillment,” you can easily turn the words *graphic design*, *offset*, *digital printing*, *mailing*, and *fulfillment* into hyperlinks that link directly to those pages on the site. (Refer to Figure 3-2.)

3. To emphasize certain words or phrases within the page content, mark up headlines and bylines using heading tags like `<h1>` and `<h2>` and mark up other important text with `` (bold) and `` (italic) styles.

By using these heading and emphasis tags, you’re alerting search engine robots that the content contained inside those tags is likely to be more relevant to search engine users than other content on the page.



If you’re advising Web clients about improving the content on their pages, tell them not to worry about being too conservative with the amount of content placed on every page. Many site owners erroneously think that having too much copy will scare away readers. On the contrary, some SEO guidelines recommend having a minimum of 200 words of copy on every page so that the spiders and robots have something to read and index.

When you have finished reviewing and updating your site, the content should be well marked up with heading, italic, and strong tags for emphasis; the copy should include lots of descriptive keywords and key phrases; and when applicable, those keywords and key phrases should be hyperlinked to other relevant pages on the site.

Including descriptive text and hyperlinks

Take extra care to ensure that the copy reads legibly and doesn't have keywords haphazardly thrown into it. Avoid at all costs the appearance of *keyword spamming*, which is what happens when you list a bunch of keywords in a row with no context. All the keywords need to be logically embedded in the page copy. For example, instead of listing the words *shampoo*, *conditioner*, *deep conditioner*, *hair spray*, *hair products*, *hair treatments*, *hair accessories*, and so on, write something like "At X company, we sell only the best hair products and hair treatments on the market. We offer all the best high-quality shampoos, conditioners, and deep conditioners, along with hair spray and hair accessories."

Be sure to also include hyperlinks on the keywords to each of those product categories. If you feel stuck for different ways of presenting the content on the page, it's okay to reuse some of the content and wording found elsewhere on the site — just don't go overboard. You could, for instance, use "X Company — Your Online Shop for the Finest Hair Products, Treatments, and Accessories" in the title tag and then repeat "X Company is your online shop for the finest hair products, treatments, and accessories. We sell shampoo, conditioner, deep conditioner, hair spray, and more!" somewhere within the body of the page.

Embedding object and image descriptions

Another great SEO technique that attracts the attention of search engine robots and spiders involves using the HTML `title` and `alt` attributes to describe certain page elements and images.

You can add the HTML `title` attribute to several different tags, such as hyperlinks, tables, table cells, and table rows. This attribute also improves accessibility by describing the contents found within the structural element, as in the following two code examples:

```
<a href="shippingrates.html" title="View our UPS Shipping Rates">View our UPS  
Shipping Rates</a>  
<table width="500" title="UPS Shipping Rates">
```

Similarly, the `alt` attribute describes in a few words or a short phrase what an image looks like or, when the image contains only copy, what that copy says:

```

```

In addition to making your code more standards compliant, the inclusion of these tag attributes both helps to increase search engine relevance of targeted keywords and makes it easier for visitors using screen readers, text browsers, and other assistive technologies to experience visiting the site.

Adding keyword and description meta tags

Even though you may have had the best intentions of adding these valuable tags (which you read about in Book I, Chapter 3) when you were building the site, you may have forgotten to put them into the head of all your pages or to have updated them with accurate and relevant information.

You can add several meta tags to your code to assist with making the pages search engine friendly. However, two of them in particular can really help with site optimization:

- ✓ **Description:** The `description` meta tag provides a brief description of the Web site's products and/or services in 250 characters or less, including spaces and punctuation. This description is critically important because it is this text that often gets displayed when the URL appears as part of a search engine's results listings. Therefore, whenever possible, try to include two to four keywords in the early part of the description statement. For example, the meta description for the JeaninePayer.com Web site is as follows

```
<meta name="Description" content="Handcrafted jewelry by Jeanine Payer. We believe jewelry is more than skin deep. We believe in old world craftsmanship meets 21st century design.">
```

and this same sentence appears as the site descriptor when listed in Google's search results, as shown in Figure 3-3.

- ✓ **Keywords:** The `keywords` meta tag provides a list of the seven most important keywords and key phrases (1,024 characters, including spaces and punctuation) that visitors to the site might type into a search engine to find the Web site's products and services. These keywords should be listed in order of importance and include any plural versions of keywords if visitors might search for both the singular and plural instances, such as *ipod* and *ipods*. The same goes for common misspellings of important keywords for any site, like *callender*, *callendar*, and *calander* for a site that sells calendars.

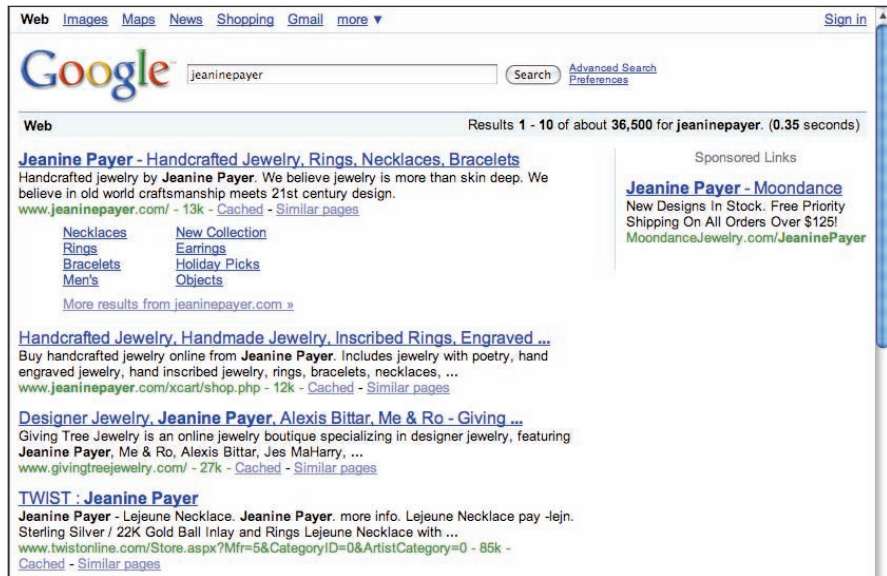


Figure 3-3: Your description meta tag appears in search engine results listings, so be sure to write a good one and include it in your code.

Keywords, unfortunately, are one of those tags that has been rendered almost completely useless by unethical SEO practitioners. Today, only one former search engine, Inktomi (which is now part of Yahoo!), still uses keywords in factoring search engine rankings. Whether you choose to include this meta tag in your code is up to you, but the prevailing thought is that as long as one search engine uses them, that's enough to warrant including the keywords meta tag in your pages.

Remember too, that meta tags don't appear anywhere on the Web page when viewed in the browser and that they are strictly used by search engine robots and spiders to index your site. The syntax for these two meta tags should be written as follows, where you fill in the specific content (shown in bold) for your particular Web site:

```
<head>
<title>Your Page Title</title>
<meta name="keywords" content="Your keywords">
<meta name="description" content="Your description">
</head>
```



One question a lot of new Web designers want to know the answer to is whether they should use the same set of keywords and description meta tags on all the pages throughout a Web site. The answer is entirely up to you. Some SEO professionals suggest using customized descriptions and keyword lists for each page of the site to improve search engine rankings for individual pages. Although this is useful, keep in mind that this takes a bit more work to manage. For one, you need to come up with the customized meta tag content for each page. For another, if you are using a template system through your HTML code editor like Dreamweaver, you need to reposition one of the closing editable area template comment tags within the HTML code so that it includes the meta tags in question. Repositioning the closing tag would make that area of the code editable within any template-based files. For example, you'd need to change your code in your template from this:

```
<!-- TemplateBeginEditable name="doctitle" -->
<title>MindworkNY.com</title>
<!-- TemplateEndEditable -->
<meta name="Keywords" content="Psychologist,Therapist,Depression,Mood
Swings,Anxiety,Stress,Counseling,New York,Carolyn Ehrlich,Mindwork,New York
City,Therapy" />
<meta name="Description" content="Mindwork was founded on the belief that those
who seek therapy have the potential for psychological and emotional growth."
/>
```

to this:

```
<!-- TemplateBeginEditable name="doctitle" -->
<title>MindworkNY.com</title>
<meta name="Keywords" content="Psychologist,Therapist,Depression,Mood
Swings,Anxiety,Stress,Counseling,New York,Carolyn Ehrlich,Mindwork,New
York City,Therapy" />
<meta name="Description" content="Mindwork was founded on the belief that those
who seek therapy have the potential for psychological and emotional growth."
/>
<!-- TemplateEndEditable -->
```

For more on how to use these and other meta tags effectively, turn to Book III, Chapter 1.

Updating bland page titles

There is nothing more boring than a page title in the browser's title bar that simply repeats the name of the page, such as *Contact*, and doesn't identify the name of the site or provide any additional clues about what can be found on the page. Title tags are like free advertising spaces that allow you to highlight the page-specific content, which in turn can help visitors find that information more quickly when researching a topic of interest through

a search engine. To take the fullest advantage of this free publicity space, be sure to add unique titles packed with keywords and key phrases to all the pages on your site.



Because your page titles do appear in the visitor's browser's title bar, as illustrated in Figure 3-4, you must pay special attention to the words you use there. Whenever possible, write complete sentences that accurately describe what can be found on each particular page.



Figure 3-4: Make the most of your page titles by writing complete, informative, keyword-rich sentences.

For example, instead of having a super-boring and unhelpful title, like this:

```
<title>About Us</title>
```

... you can use more site-specific keywords in the title to improve the chances of people finding particular pages on your site when searching for products, services, and information through a search engine, like this:

```
<title>Rockwood & Perry provides information on wine rating, shipping costs,
methods & legal states, policies, guarantee & cellaring</title>
```

As with the other meta tags, try with your title tags to include a couple of keywords or key phrases near the front part while identifying the content on the page, but be careful not to simply list keywords. Keyword listing could be misinterpreted as spamming and could blacklist the domain from being indexed by search engines. Instead, each title needs to read like an enticing, informative sentence, not a laundry list.



Page titles can be any length up to about 70 characters. If your titles exceed this suggested length, any extra characters may be truncated in the title bar of some browsers, such as “Rockwood & Perry Fine Wine & Spirits offers fine wines, advice, accessories, direct imports . . .” instead of the full title as listed in the code. Still, for the extra bump that having well-written, keyword-rich, descriptive titles can do for your site rankings, seeing truncated titles in a browser’s title bar may not be such a critical issue.

Submitting a Site to Search Engines

As soon as a new site is 100 percent complete and published on its domain, the URL can be submitted to search engines for indexing and listing. You do this by submitting the site’s URL to the search tools that the target audience is likely to use. These tools include search engines, which use robots and spiders to crawl the Web in search of new listings, and search directories, which are essentially categorized lists of sites that are sometimes compiled or edited by people instead of bots.

Hand-submitting the URL

Although it is a good idea to get the site listed on the most popular search tools as you think will assist the target in finding the site, it does not mean using some kind of submission tool or SEO service that will blast the URL, like spam, to any and all search engines around the world. Those kinds of submission techniques not only create tons of kickback spam e-mails to the submitting e-mail address, but they also rarely, if ever, increase site traffic significantly.

What’s more, some SEO submission tools and SEO services deliberately submit the URL directly to spam sites — Web pages of site listings that clog up the Internet and have no purpose — which can jeopardize the integrity of the domain when it comes to listing the site legitimately on the major search engines. The best way to get a site listed, then, is to hand-submit the URL to the major search engines and directories yourself and, when additional exposure is desired, to pay a reputable listing service like Google AdWords and Yahoo! SearchMarketing (formerly Overture Keywords) for pay-per-click advertising. These services offer paid listing options for as little as \$25 per month that can guarantee that your site is listed in the top search results or is listed in the sponsored advertising space directly above or to the right of the regular search listings.

Most search engines, search directories, and search listings (which use search engines and directories to compile their data) charge you a fee for submitting a URL with them; however, a few of them are still free. Table 3-1 lists the most popular tools for making search engine submissions.

Table 3-1 Search Engines, Directories, and Search Listings			
<i>Service Name</i>	<i>URL</i>	<i>Service Type</i>	<i>Free or Paid Service</i>
Google	www.google.com/addurl	Search engine	Free
Open Directory Project	http://dmoz.org/add.html	Search directory	Free
Yahoo!	http://search.yahoo.com/info/submit/free/request	Search directory	Free and paid service (requires registration with Yahoo!)
AOL Search	Submit URL to Google and your listing should appear in AOL's search listings	Search listing	Free
MSN Search	http://beta.search.msn.com/docs/submit.aspx	Search listing	Free
Google AdWords	http://adwords.google.com	Search engine	Paid service
Yahoo! Search Marketing	http://searchmarketing.yahoo.com/index.php Yahoo! is now a hybrid of purchased indexes: Inktomi, AltaVista, and AlltheWeb	Search directory	Paid service

Submitting your site to a search tool usually only takes a minute or two, and although you have no guarantee of the submitted URL being indexed, a submitted page is better than no submission at all. As a value-added service, you may want to offer to submit your Web clients' sites to the free search engine submission tools as part of your Web design package.

Most of the free submission tools request just the domain name and a valid e-mail address to complete the submission process. Google, for instance, only requests the entry of the home page URL, comments about the site, and an optional human submission validation field entry, as shown in Figure 3-5.

Google™ Add your URL to Google

[Home](#)
[About Google](#)
[Advertising Programs](#)
[Business Solutions](#)
[Webmaster Info](#)
[Submit Your Site](#)

Find on this site:

Share your place on the net with us.

We add and update new sites to our index each time we crawl the web, and we invite you to submit your URL here. We do not add all submitted URLs to our index, and we cannot make any predictions or guarantees about when or if they will appear.


Please enter your full URL, including the `http://` prefix. For example:
`http://www.google.com/`. You may also add comments or keywords that describe the content of your page. These are used only for our information and do not affect how your page is indexed or used by Google.

Please note: Only the top-level page from a host is necessary; you do not need to submit each individual page. Our crawler, Googlebot, will be able to find the rest. Google updates its index on a regular basis, so updated or outdated link submissions are not necessary. Dead links will 'fade out' of our index on our next crawl when we update our entire index.

URL:

Comments:

Optional: To help us distinguish between sites submitted by individuals and those automatically entered by software robots, please type the squiggly letters shown here into the box below.



Other Options

Instant Ads on Google
 Create your own targeted ads using [AdWords](#). With credit card payment, you can see your ad on Google today.

Google AdSense for Web Publishers
 Publish ads that match your content, help visitors find related products and services – and maximize your ad revenue. [Learn more.](#)

Google-Quality Site Search
 Reduce support costs, keep users on your site longer, and turn browsers into buyers with the [Google Search Appliance](#) or [Google Mini](#).

Submit Additional Content
 See all opportunities to [share your content](#) for free across Google including gadgets, product search, local business center and video.

Figure 3-5: Enter the home page URL to submit a site address to Google.

Other tools require that you choose a submission category that matches the site's product or service offerings. For example, before submitting a URL to the Open Directory Project (<http://dmoz.org>), you must use that site's search feature to locate the most appropriate category for your site, and then submit the URL to the directory from within that specific section of the directory. To illustrate, suppose you've just designed, built, and published a new Web site for a children's mystery book writer who's just finished his second novel in a series about a fictional character. What would be the best category to list this site in? One for children's books, one for children's book authors, or one that promotes the sales of the children's books?

To find out, go to <http://dmoz.org> and, in the search bar, enter the keywords *children's books*. What are some of the subcategories that appear? Are some more appropriate than others? Type another search with the keywords *children series books*. Two suitable categories for this Web site might be "Arts: Literature: Children's: Children's Series Books" and "Business: Publishing and Printing: Publishing: Books: Children." Because you may submit the site to only one category, choose wisely because you have no second chances.



If you are submitting the site for a Web client, ask the client for her opinion about which category best matches her needs before submitting the site to the search tool. As soon as you have selected the right category, select it and then click the Submit URL link at the top of the page. This brings you to a submission page where you can follow the online instructions for submitting the site.

Waiting for the site to be listed

After you make a submission, each search engine, search listing, search index, and search directory can take anywhere from one day to two months to get it listed. For instance, Google claims listings will occur within four weeks of submission, but listings can just as likely occur within a day or two. Tell your client to be patient while waiting for the listing to appear after the home page URL has been submitted. Just because a site gets published doesn't automatically mean it will be found instantly by any and every search engine. All that publishing a site means is that the content on the domain is publicly accessible to anyone with an Internet connection. To be listed, a URL must be submitted to a search engine, search listing, search directory, or search index. Tell your client that as long as the URL submitted to the search engine directs visitors to her domain, the listing should happen within a reasonable time frame, typically between one and eight weeks.



With most search engine submissions, only the home page will be indexed, rather than all the pages on the entire site. If you are interested in finding out how many of the site's pages have been indexed, check with the search engine to see whether it has any method for verifying that information. Google, for instance, can show you how many of your site's pages have been indexed when you type **site:mydomainname.com** into Google's search field. The Dummies Web site (site:dummies.com) alone boasts about 39,000 indexed pages!

Though some SEO professionals may encourage you to resubmit your URL to search engines, search listings, search directories, and search indexes daily, weekly, or any time you update even a single page on a site, understand that those search tools may treat multiple submissions like those as more of a form of computerized Internet harassment than a smart business practice. A more rational and courteous submission schedule to take for your site would be to make one initial submission shortly after the site gets published, and then make regular quarterly or slightly more frequent resubmissions when the content on the site is updated.

Giving Your Site an HTML Site Map

When you create a visual site map (as described in Book I, Chapter 3), you create a diagram of all the pages on a site, including the interconnectivity of the main pages through navigation and subnavigation. You then use that

information to help gather and define site content, as well as serve as a useful guide when generating the mock-up of the site's design. After your site has been fully built, you can use this visual site map again to help you create an HTML Site Map page, which will be added to the site on the remote host server as a tool to help visitors navigate through the site using a list of hypertext links.

The following sections explain what an HTML Site Map is, how to create one, and how to make it easily accessible to all other pages on the site.

Deciding what to include on the HTML Site Map page

In its most basic form, the HTML Site Map page contains a list of standard hypertext links to all the pages on a Web site. This list should include links to the home page, to all the main pages and subpages on the site, and to any other pages on the site that might not be accessible through the main navigation, such as a Privacy Policy or Customer Service page.

For best results, save the page with the filename `sitemap.html` and present all the hyperlinks to the site's pages in a simple list format, like the one shown in Figure 3-6.

To make this finished page accessible, you should include a link to it in the footer as well as adding a link to it within the head of the page. These two actions will help make the site more accessible to

- ✓ Visitors with disabilities using screen-reading programs or other devices
- ✓ Visitors with other browsing preferences, such as text-only browsers and browsers with JavaScript disabled
- ✓ Visitors who want to be able to go directly to any page on the site with a single click, rather than using the site's main navigation system
- ✓ Visitors who want to see at a glance all the pages on a given site and know how they're virtually organized
- ✓ Visitors browsing the site using handheld devices like a Web-enabled BlackBerry or iPhone

In addition to helping human visitors navigate the site, a site map also helps search engines locate and (with luck) index all the pages of a site. Having hyperlinks to all the pages in one location does improve the chances of the entire site getting indexed, and when that happens, the contents of that become more readily accessible to visitors, which can help increase Web traffic. This is because some search engine robots and spiders ignore the images and other graphics on a page and instead only pay attention to the

meta tags, marked-up text, and hyperlinks when indexing individual pages. Logically, then, because the Site Map page is really just a listing of hyperlinks to all the pages on a site, those pages have a higher likelihood of being indexed.

If you're still not convinced about the benefits of having an HTML Site Map page, think about this: If your site has a page for products and the products page links to six additional pages that have further information about each of those six specific products, you'd be wise to allow visitors to be able to find those specific pages from a search engine. For example, if your Web site sells custom hand-crafted silver jewelry that includes rings, necklaces, and earrings, you may be able to sell more necklaces by having a hyperlink on the Site Map page that takes visitors directly to your necklaces page. After a search engine indexes the pages listed on the Site Map page, a visitor searching for *hand-crafted silver necklaces* might more easily find your site in the search engine results listing and be able to go directly to your site's necklaces page.

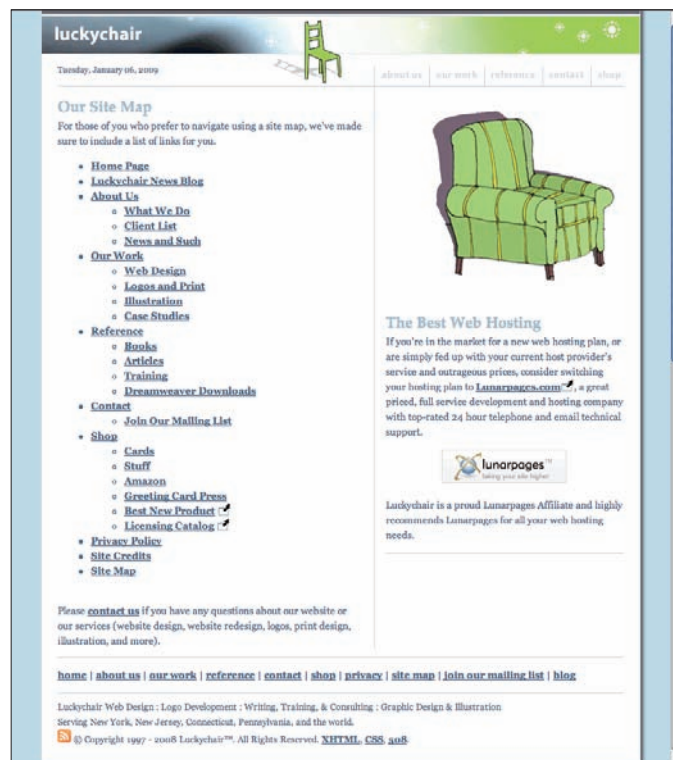


Figure 3-6: Keep your Site Map page links in a simple list format for easy navigation.

Creating a Site Map page

Creating a Site Map page in HTML is fairly straightforward. It is just a matter of inserting a set of listed hyperlinks that lead to the rest of the pages on the site.



To illustrate how to convert your visual site map diagram (which you create for your site in Book I, Chapter 3) into an HTML Site Map Web page, grab your site map or, if you want to follow along with the example, download the sample site map for Sugar Monkey in PDF format from www.dummies.com/go/webdesign101, shown in Figure 3-7; then follow these steps to create a Site Map page:

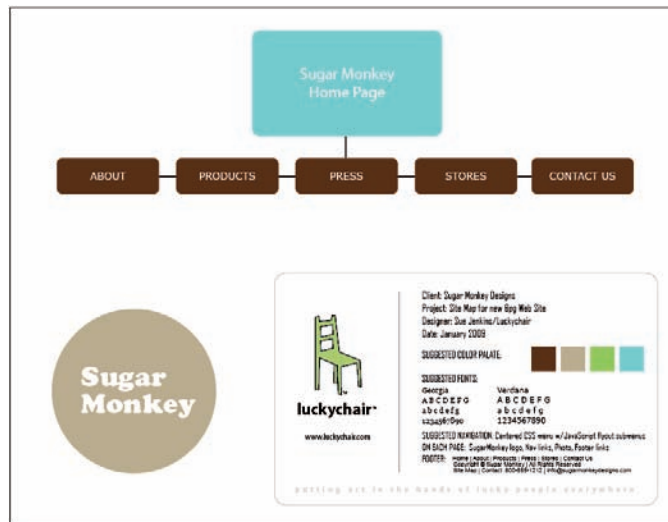


Figure 3-7: Use the architectural site map diagram to build the HTML Site Map page.

1. **Open the home page of your completed Web site in Dreamweaver or in your preferred HTML or WYSIWYG code editor.**

If you are using Dreamweaver, make sure that the Files panel lists your site as the managed site. If not, choose Site→Manage Sites to select your site from the listing.

2. **Choose File→Save As, and when the Save As dialog box opens, save a copy of the home page to the root level of your managed site with the filename `sitemap.html`.**

The new page looks identical to the home page, including all the site's design elements, logo placement, and navigation.

- 3. In the main editable area of the page, select and delete all the home page-specific content.**

This makes room for you to insert a list of hyperlinks.

- 4. Place your cursor at the top of the empty area and type in the names of all the pages on the site, including the Site Map page, with one name per line, as they appear on the visual site map.**

For example, use the page names Home, About, Products, Press, Stores, Contact Us, and Site Map.

- 5. Select all the words and convert them into an unordered list, thereby converting each page name into a list item.**

The HTML code for the Site Map page should use the unordered list tags, as shown in the following sample code:

```
<ul>
  <li>Home</li>
  <li>About</li>
  <li>Products</li>
  <li>Press</li>
  <li>Stores</li>
  <li>Contact Us</li>
  <li>Site Map</li>
</ul>
```



When you are creating your own Site Map page in HTML, be sure to also include links in this list to any pages on your site that may not be linked through the main navigation. For instance, if your site includes Privacy Policy, Terms of Service, and Customer Support pages, add links to those pages to the bottom of the list, directly above the link to the Site Map page.

- 6. If you have any subpages, add those pages to the list as a subset of the main page they should be listed under, as shown in this example:**

```
<ul>
  <li>Home</li>
  <li>About</li>
  <li>Products</li>
  <ul>
    <li>Silver</li>
    <li>Gold</li>
  </ul>
  <li>Press</li>
  <li>Stores</li>
  <li>Contact Us</li>
  <li>Site Map</li>
</ul>
```

For example, the Sugar Monkey Web site has no subpages below any of the main navigation links, so you can skip this step.

- 7. (Optional) If desired, type in a descriptive sentence beneath each bulleted item in the list to help visitors learn about what they can expect to find on each page.**

A description like this can assist visitors viewing the pages as well as help search engines locate and index each of the pages on the site.

For example, if you were to add descriptions to the Sugar Monkey Site Map HTML page, one of your list items might look like this:

```
<li>About<br>
Learn about the founders of Sugar Monkey and get the real story behind
Sugar Monkey's success.</li>
```

- 8. Convert each of the page names in the list into hyperlinks to their respective pages on the site, making sure that each link includes its own title attribute.**

For instance, if the about list item became a link to the About page, your code might look like this:

```
<li><a href="about.html" title="Learn About Sugar Monkey">About</a><br>
Learn about the founders of Sugar Monkey and get the real story behind
Sugar Monkey's success.</li>
```

- 9. Save your changes, close the `sitemap.html` file, and upload a copy of this file to the site's remote host server.**

Test the page as soon as possible and correct any typos or coding errors. If you make any changes, reupload the file to the remote host server.

No one will know about this page until you add a few links to make the page accessible to site visitors.

Making the site map accessible

After you have finished building your new HTML Site Map page, you need to tweak your site a little bit to make that page easily accessible to all the other pages on the site. Specifically, the Site Map page should be linked in two key areas of a Web page:

- ✓ **Footer:** The first place to add the Site Map link is in the footer of every page, next to the other footer links to all the main pages of the site, as shown in Figure 3-8.
- ✓ **Head of HTML code:** The second place to add a link to the Site Map page is within the head of your HTML code on every page of the site. If you are using a template on your site, you should be able to add this link to the template, and your HTML editor should update all the template-based pages with this link when you save the changes to the template file.



Figure 3-8: Include a site map link in the footer of every page.

The link itself includes both the `rel` tag attribute, which specifically defines the link as a site map link, and the `href` attribute, which allows you to specify the source (location) and filename of your Site Map page. Place the link code anywhere between the page's opening and closing `<head>` tags:

```
<head>
  <link rel="Site Map" href="sitemap.html">
</head>
```

By adding this link to the head, you not only make the page friendlier to search engines but you also make the page, and thus the entire site, more accessible to people using methods other than browsers to access the pages on your site.

Site Map pages are also especially useful for larger sites that rely on graphics-heavy JavaScript, DHTML, or Flash navigation menus, all of which cannot be easily accessed by certain screen-reading programs, text-only browsers, other assistive technologies, or Web robots or spiders. This isn't to say that you should never use JavaScript or other methods for your navigation. Rather, when you do use them, be sure you also include a link to the Site Map page in the footer of every page. In addition, when the navigation uses JavaScript, be sure to also include `<noscript>` tags somewhere on your page (perhaps at the bottom of the body area) for visitors who have JavaScript disabled or use devices that can't read JavaScript. The following example shows how you might add a `<noscript>` tag to your code:

```
<noscript>
<p>This site uses JavaScript for the main navigation. To access the pages
  without JavaScript, please use the links on our <a href="sitemap.html">Site
  Map</a> page.</p>
</noscript>
```

To further enhance the usefulness of your Site Map Web page, apply any of or all the following suggestions to your pages:

- ✓ On larger sites, in addition to including a descriptive sentence below each link in the list, consider adding a sentence below each main navigation area to describe the contents to be found within that section.
- ✓ Include the `title` attribute in every hypertext link on the site map, as in the following code example:

```
<li><a href="freemp3s.html" title="Download Free MP3s">Download Free MP3s</a></li>
```

- ✓ Add the `tabindex` tag attribute (normally used for form fields, but they can also be used to add a tab index number to image maps and hyperlinks) inside each hypertext link to help visitors with disabilities access those links with a single keystroke, such as

```
<li><a href="freemp3s.html" title="Download Free MP3s" tabindex="7">Download Free MP3s</a></li>
```

- ✓ Add a site search feature to your site to further assist visitors in finding exactly what they are looking for. Google has a nice free custom search engine tool you can use (www.google.com/coop/cse), or you can find several other scripts online, for money and for free, that you can use to add the search functionality to your site.
- ✓ Consider adding context-specific navigational hypertext breadcrumb links, such as Home → Services → Search Marketing, to the tops of all the content areas on your site to show visitors where they are hierarchically within the site and provide visitors with an alternative method of navigation, as illustrated in Figure 3-9.

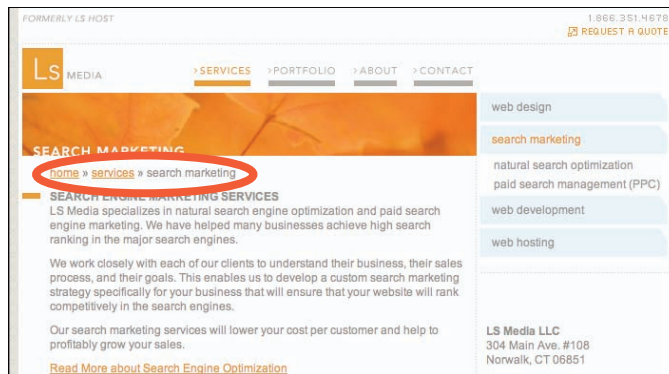


Figure 3-9: Breadcrumb links show visitors which page they are on relative to other pages on a site and provide an alternative method of navigation.

Keeping the Site Relevant

Finally, you've put your site online, the SEO stuff is all taken care of, you're happy to be finished, and there appears to be nothing else to do with this particular project for the time being. Now is the time to celebrate, relax, and forget about the site for a little while . . . that is, until it is time to make some changes. Routine site maintenance is a must for any Web site, so before you get to that stage of running a Web site, take some time to plan out a method for performing your site updates. For instance, you may want to schedule updates at regular intervals rather than just haphazardly jumping in to make changes anytime they are needed. In addition to site maintenance, you need to think about how often to add new content to the site to keep it fresh and appealing to visitors.

Performing site maintenance

No matter how big or small your Web site happens to be, at some point, there will come a time when you need to perform some site maintenance. The changes may be as minor as updating the copyright year in the footer after January 1, as complex as adding a new page to the site and having to update all the site's navigation links to accommodate the new page, or as major as a full Web site redesign to give your site a fresh appearance.

Depending on the breadth of the site maintenance task before you, you can either leave the site online or take the site "temporarily down" while you upload the changed files to the remote host server:

- ✓ **Online site updates:** When your site changes are fairly minor, work on a local copy of the site and make your changes offline. Test your pages in a temporary directory on the host server to verify that the changes are accurate, and then upload all the updated files — including any images, media files, and external CSS, JavaScript, and SSI files — to the remote host server. In most instances, the uploads such as these will not significantly disrupt visitor traffic.
- ✓ **Offline site updates:** When your site changes are substantial, you can still work on a local copy of the site and make your changes offline. When you are finished, test your updated pages in a temporary directory on the host server to verify that the changes are accurate. However, to avoid the possibility of negatively impacting your visitors' experience on the site during the transferring of all the updated files from your local computer to the root level of the remote host server, temporarily disable the site.

To temporarily take your site offline so that you can transfer updated files to the server without affecting visitors, do one of the following:

- ✓ Work with your host so that you can upload your updated site to an active server where you can do all your testing. Then change the DNS record so that instead of *mywebsite.com* pointing to the old server at 123.123.123.123, it now points to 123.123.123.124, where your new content is ready to rock.
- ✓ Create and upload a special Under Maintenance Web page to the remote host server and then install a 301 Redirect script on your server so that access to any page on the site during the maintenance period automatically redirects visitors to the maintenance page. The 301 Redirect can be implemented in a variety of ways, depending on your skill level and coding preferences. For a tutorial on all the different ways you can create a 301 Redirect on your site, visit www.webconfs.com/how-to-redirect-a-webpage.php.

Scheduling site updates

Some site updates may come with regular frequency, such as adding new files that can be downloaded by visitors each week or updating a monthly schedule of events. Other changes, by contrast, may come at less frequent intervals, which is often the case when you're maintaining a Web site for a client, family member, or friend.

To help you manage your time and make the most out of every updating session, you may want to limit site updates to a particular day of the week. For instance, by making Monday your site update day, you can combine multiple small tasks into one updating session. If you have multiple Web clients, this kind of system can also help you establish some ground rules with your client so that change requests aren't coming in all week long. Instead, your clients can take on the responsibility of compiling all their change requests into a single document that they can then send to you, say, each Friday afternoon. If your clients have less frequent site update needs, you can have them send in any change requests by a particular date each month, such as on the 1st or 15th.

Adding new content regularly

On the Internet, the well-ranked and popular sites are the ones that keep all their content fresh and up to date.

At a minimum, you should plan to make regular updates to the home page. If you have more time and are devoted to fine-tuning your site to attract your target audience and accommodate their specific needs, you may choose to make daily, weekly, biweekly, or monthly updates to both the home page and the other pages on the site.

In addition to updating content within your pages, you can do plenty more things to keep your content current, including the following:

- ✓ If you will be making regular updates, invite your site's visitors back at regular intervals and encourage them to participate in any special promotions, discounts, sweepstakes, and opinion polls.
- ✓ Regularly check your site statistics. Most host providers include some kind of site stat tool with your hosting plan that includes information like which URLs and IP addresses are visiting your site, their country of origin, and the entry and exit page URLs.
- ✓ Anytime you make updates to your site, check all the internal and external links. Some of the links to external links may have changed or otherwise been removed from the Internet. When you find any broken or dead-end links, update them immediately or remove them.
- ✓ Periodically update your site's design. Change the color scheme or some of the photos, or put a new face on your site. Showcase your talents and keep visitors coming back for more.
- ✓ Revisit your page titles and meta tags and update any that no longer accurately describe the contents of the pages and your site.
- ✓ If your site lists a calendar or event schedule, make sure that it gets updated in a timely manner. People don't want to know about a class they can no longer sign up for or read about a product or service that isn't offered anymore.
- ✓ Got a new product or service? Shout about it on your Web site. Devote an area on the home page and either integrate the new item(s) into your site's other pages or create a new dedicated What's New page. This can help encourage visitors to return to the site more often.
- ✓ If your site sells products or services, ask yourself whether you can do anything to improve the ready-to-purchase and checkout cycles. If needed, add additional informational steps to help your visitors complete their purchases.
- ✓ Sites that get a lot of e-mails and phone calls that ask many of the same questions can often benefit from adding a FAQs page to the site. The FAQs page can list both the questions and the answers to these important customer inquiries.
- ✓ How about adding a blog to your site? Blogs can increase visitor traffic by posting information that is of interest to the target audience.
- ✓ If you have anything else on your site that is no longer relevant or is out of date, take it off the site. Taking care of your site includes making sure that your content is timely and relevant.



Each Web site warrants its own update schedule. Consider your own Web site's needs and create a custom calendar of things to remember to help you stay on top of site maintenance and content changes each month. For a nice example of a "Calendar for Website Maintenance and Submission,"

like the one shown in Figure 3-10, as well as a good list of things to remember when performing your site updates, visit the Lorelle VanFossen's WordPress Web site at <http://lorelle.wordpress.com/2005/09/27/website-development-make-a-schedule-and-calendar>.

Calendar for Website Maintenance and Submission			
January	February	March	April
<ul style="list-style-type: none"> • Site Submission • Check Site Statistics 	<ul style="list-style-type: none"> • Check Link Popularity • Verify Links 	<ul style="list-style-type: none"> • Check Site Statistics 	<ul style="list-style-type: none"> • Site Submission • Add New Content
May	June	July	August
<ul style="list-style-type: none"> • Check Site Statistics • Update Headings and Tags 	<ul style="list-style-type: none"> • Check Link Popularity • Check Tags 	<ul style="list-style-type: none"> • Site Submission • Check Site Statistics 	<ul style="list-style-type: none"> • Verify Links • Add New Content
September	October	November	December
<ul style="list-style-type: none"> • Check Site Statistics 	<ul style="list-style-type: none"> • Site Submission • Check Link Popularity • Add New Content 	<ul style="list-style-type: none"> • Check Site Statistics 	<ul style="list-style-type: none"> • Review Web Standards and Update

Figure 3-10: Making a schedule for your Web site can help you stay on top of necessary site maintenance tasks.

Moving on

Congratulations! You've done it! You have published your first Web site. Although that feels fantastic in one sense, finishing a project can sometimes be a little anticlimactic too. You've put in so much work to design and build it and now, suddenly, you must let it go and move on. If you have done an admirable job, there is good chance that while you're working on your new Web projects, the client will often come back to you to make minor (and sometimes major) adjustments to the site. Some clients need regular weekly or monthly site maintenance, whereas others might only contact you sporadically for small edits like a change of address or to modify the copyright year in the footer.



The more responsive and friendly you are in making site updates for your existing clients, the more likely those clients will continue working with you and be inclined to recommend your services to their friends, family,

colleagues, and business associates. To help generate more business from your existing clients, consider sending out your own periodic e-mail newsletters or making personal phone calls once a month or so to check in and say hello. Personal contact like this really means a lot in this fast-paced, impersonal, need-it-now society. To really stand out from the competition, send your client a handwritten congratulations card with a note saying how much you enjoyed working with him or her and how pleased you are with the launch of the new site.

Another thing you should do after finishing each new project is add information about the project to your own Web site. Some designers simply include a small screen shot graphic of the completed site's home page with a link to it and the name of the client's site. Others, like the good folks over at 420Creative.com, whose Portfolio page is shown in Figure 3-11, combine a thumbnail listing with more in-depth case studies that detail the entire process from start to finish. (If you don't have your own Web site yet, you've just found your next project to work on.)

When word begins to spread about your services, hopefully you'll be able to spend less time and money on marketing. Until then, it's up to you to find new clients and build your portfolio. The more projects you have under your belt, the more you can feel confident about your skills and possibly charge more for your services. Best of luck!

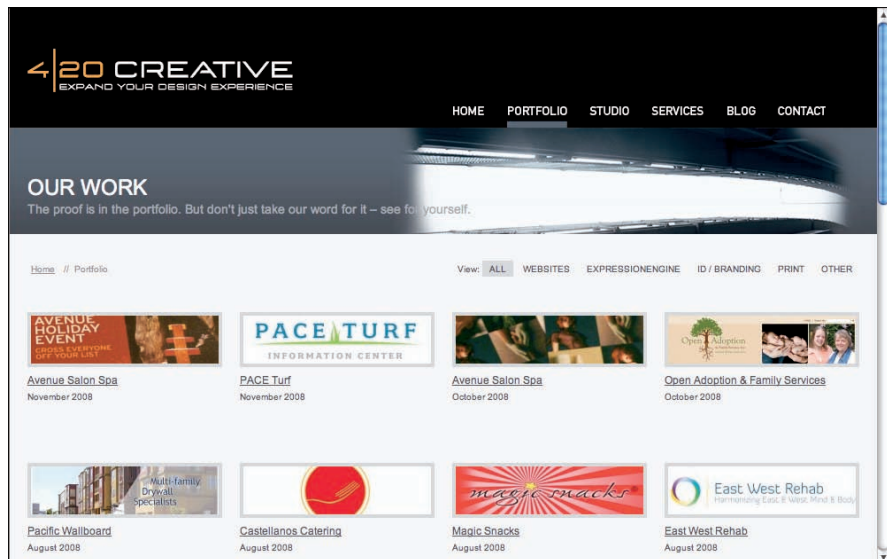


Figure 3-11: Showcase your talent with a portfolio page that contains a screen shot of all your completed Web projects.

Index

Symbols and Numerics

- (•) bullet, 159, 366
- { curly braces, 274
- (-) dash, 366
- (|) vertical line, 366
- 404 File Not Found page, creating, 588, 589–590
- 401 Unauthorized Access page, creating, 587–590
- 14–15-inch monitor, 130
- 19 inch monitor, 130
- 96 ppi, setting resolution to, 181
- 17 inch monitor, 130
- 17–19-inch monitor, 130
- 30-inch monitor, 130
- 301 redirect, 213
- 24-inch monitor, 130
- 20-inch monitor, 130
- 20-inch+ monitor, 130

A

- About page, deciding whether to use, 64
- above the fold, designing, 160–163
- accessibility
 - coding for, 245–247
 - with CSS (Cascading Style Sheets), 272
 - described, 135
 - HTML Site Map, 613–615
 - in layouts, creating, 334
 - standards
 - overview, 503–504
 - Section 508, 504–508
- accesskey attribute, 247–248
- Actionscript, 364
- activating your registered domain name, 555

- active hyperlinks, 293
- Adobe Fireworks CS4 How-Tos: 100 Essential Techniques* (Babbage), 186
- advanced combinators, 293
- A-heads. *See* header graphics
- align attribute, 228–229
- alignment
 - of contents in table cell, 257
 - of graphics, 228–229
 - of tables, 256
 - of text HTML, 222–223
- all media type, 279
- alt attribute, 226–227, 246
- anchor links, 241–243
- <a> anchor tag, 230
- antialiasing, 164–165
- <applet> tag, 218
- Arial fonts, 141
- artists, hiring freelance, 76–77
- artwork for image of site, 15–16
- attributes
 - accesskey, 247, 248
 - align, 228–229
 - alt, 226–227, 246
 - alternate text attribute in an image tag, 245
 - border, 225, 227
 - cellpadding, 258–260
 - cellspacing, 258–260
 - href, 230
 - id, 243–244, 246, 254
 - nowrap, 260
 - overview, 96–97
 - tabindex, 248
 - table title attributes, 246
 - target, 231–232, 247
 - title, 246
- aural media type, 279

B

- Babbage, Jim (*Adobe Fireworks CS4 How-Tos: 100 Essential Techniques*), 186
- background color for tables, 261–263
- background images, 173
- background style (CSS), 307–310
- <basefont> tag, 218
- BBEdit, 93
- behaviors, 377
- Berners-Lee, Tim (W3C), 485
- biography, 20
- Birley, Shane (*Blogging For Dummies*), 33
- bit depth, 46
- block boxes, 302
- block style (CSS), 310–312
- blocking out parts of the page in mock-ups, 157–160
- blogger, 33
- Blogging For Dummies* (Gardner & Birley), 33
- BlogHarbor, 34
- blogosphere, 33
- blogs
 - advantages of, 36–37
 - content-relevant ads on, 37
 - disadvantages of, 36–37
 - including a blog on your Web site, 25–26
 - overview, 32–33
 - profile, adding a, 35–36
 - tools for creating, 34
 - used to attract visitors, 32–37
- <body> tag, 97–98, 208–209
- bold text, 221–222
- border attribute, 225, 227

- border color for tables, 261–263
- border style (CSS), 314–315
- borders
 - graphics, 225, 227
 - tables, 257–258
- Box model (CSS)
 - absolute positioning for
 - block elements, 304
 - block boxes, 302
 - customizing, 303–304
 - floating position for block elements, 304
 - inline boxes, 303
 - normal position for block elements, 304
 - overview, 301–302
 - positioning blocks, 304
- box style (CSS), 313–314
- Braille media type, 279
- brand
 - above the fold location for, 161
 - described, 148
 - information, providing, 69
 - positioning, 148–149
- Bravenet Site Search, 22
- brochureware Web sites, 12
- Browser Compatibility tool (Dreamweaver), 527–528
- browsers
 - compatibility, checking, 527–528
 - interface component used for FTP (File Transfer Protocol), 574
- building phase
 - described, 10
 - performing code cleanup during, 516
- building Web sites
 - with SSIs (Server-Side Includes), 469–480
 - with templates, 460–468

C

- Cascading Style Sheets. *See* CSS
- cellpadding attribute, 258–260
- cellspacing attribute, 258–260
- <center> tag, 218
- CGI (Common Gateway Interface), 398
- check box fields in Web forms, 409–410
- Clean Up HTML/XHTML command (Dreamweaver), 520–521
- Clean Up Word feature (Dreamweaver), 518–519
- cleaning up your code, 515–524
- clear property, 313
- ClickTracks, 48
- client
 - preferences of, suggesting the opposite of, 128
 - working with client to make design choices, 125–126
 - writing, getting the client's approval in, 170, 463
- Clients page, deciding whether to use a, 65
- clip property, 320–321
- CMS (Content Management System), 120
- CMYK (Cyan, Magenta, Yellow, and Black) colors, 177–178
- code
 - cleaning up your code
 - building phase, performing code cleanup during, 516

- DTD, converting syntax by, 521–524
- finding and replacing errors, 516–517
- formatting, removing unwanted, 518–519
- HTML syntax, applying
 - consistent, 520–521
 - overview, 515–516
 - source formatting, applying, 521
 - spelling, checking the, 517–518
 - XHTML syntax, applying
 - consistent, 520–521
- fixing common code errors
 - browser compatibility, checking, 527–528
 - with built-in testing tools and reports, 525–526
 - links, verifying internal and external, 529–530
 - meta tags, checking, 525
 - overview, 524–526
 - page titles, checking, 525
 - site reports, generating, 530–531
 - validating your markup, 526–527
- hand coding, 214–216, 217
- HTML and CSS markup validation
 - direct input, validation by, 533
 - failing acceptably, 538–539
 - with free online validation tools, 532–536
- icons, adding validation, 540–541
- noncompliant code, fixing, 536–538
- overview, 531
- proof of validation, obtaining, 540–541

- retesting, 538
- upload, validation by, 532
- URL, validation by, 532
- with W3C Markup validator, 533–536
- meta tags, 83
- overview, 217–219
- validation process
 - advantages of using, 510
 - cleaning up your code, 515–524
 - fixing common code errors, 524–531
 - HTML and CSS markup validation, 531–541
 - overview, 510
- code editors, 93–94
- CoffeeCup Free FTP, 574
- collecting and using visitor data, 26
- color
 - CMYK (Cyan, Magenta, Yellow, and Black) colors, 177–178
 - contrasting foreground and background colors, 247
 - conversion tools, 111
 - hexadecimal, 110–114
 - matte, 199, 200, 201
 - in mock-ups, 160
 - mode for Web graphics, 177–178
 - out of gamut, 178–180
 - palette, choosing a, 138–140
 - Pantone color, finding hexadecimal equivalent of, 111–114
 - reduction algorithm for optimization of Web graphics, 197
 - RGB (Red, Green, and Blue) colors, 110, 177–178
 - values for custom hyperlinks, 293–294
 - Web-safe, 108–109
- Color Cache, 140
- color gamut warnings, 178–180
- Color Picker, 179–180
- Color Wheel Pro, 140
- comments
 - adding, 389
 - tags in Dreamweaver templates, 467
- Common Gateway Interface (CGI), 398
- compound styles, 292–294
- contact information, 17
- Contact page, what to include on, 64–65
- content
 - added to cells, 252–253
 - described, 61
 - gathering, 62–63
 - graphics
 - artists, hiring freelance, 76–77
 - copyright issues, 78
 - overview, 73–75
 - stock images, licensing, 77–80
 - lists, 268–269
 - meta-tag data, choosing, 80–84
 - page titles, choosing, 80–82
 - requirements for, defining, 62–70
- text
 - copywriter, hiring a, 75–76
 - overview, 73–75
 - vision for the site, questions to help determine, 62–70
- wireframes
 - for every page element, 70
 - for a particular page element, 71
- Content - Language meta tag, 212
- Content - Type meta tag, 213
- Content Management System (CMS), 26, 120
- content of site
 - determining, 16–24
 - marketing and sales content, 18–24
 - minimum requirements to include on a site, 16–18
 - overview, 16
- content-relevant ads on blogs, 29
- contests and sweepstakes used for attracting visitors, 40–41
- contract phase, 10, 43
- conversion rate, 594
- conversion tools, 111
- Convert Syntax command (Dreamweaver), 522
- cookies, 17–18
- copyright issues for graphics, 78
- creativity in navigation systems, 147
- cross-platform fonts, 141, 142
- CS4 Web Premium (Adobe), 105
- CSS Beauty, 331
- CSS (Cascading Style Sheets)
 - accessibility with, 272
 - advantages of, 271–272
 - anatomy of a style, 273–274
 - background style, 307–310
 - block style, 310–312
 - border style, 314–315
 - box model, 301–304
 - box style, 313–314

- CSS (Cascading Style Sheets) (*continued*)
 - combining different types of, 276–277
 - comments, adding, 389
 - compliance information, 23
 - declarations, 274
 - descendant selector style, 343
 - Dreamweaver, linking CSS with, 284–286
 - extension style, 321–322
 - external, 276
 - faster page download time with, 272
 - footers, styling, 322–324
 - formatting
 - advantages of, 500–501
 - examples of, 501–503
 - HTML formatting compared with, 499–500
 - overview, 498–499
 - headers, styling, 322
 - images, styling, 327–328
 - inline, 275
 - internal, 275
 - labeling content for CSS markup, 243–245
 - layers, styling, 329
 - layers-based layout
 - creating, 340–344
 - Dreamweaver templates for, 356–358
 - styling, 344–349
 - linked style sheets, 276
 - linking external CSS to a page, 277–279
 - list menus, 364
 - list navigation menus
 - creating, 385–390
 - overview, 383–385
 - list style, 315–317, 324–326
 - markup, labeling content for, 243–245
 - markup validation
 - direct input, validation by, 533
 - failing acceptably, 538–539
 - with free online validation tools, 532–536
 - icons, adding validation, 540–541
 - noncompliant code, fixing, 536–538
 - overview, 531
 - proof of validation, obtaining, 540–541
 - retesting, 538
 - upload, validation by, 532
 - URL, validation by, 532
 - validation process, 531–541
 - with W3C Markup validator, 533–536
 - master CSS file
 - creating, 295–300
 - overview, 294
 - media types, 279–284
 - online resources for, 329–331
 - overview, 271–272
 - paragraphs, styling, 322
 - positioning style, 317–321
 - print media CSS, adding, 182
 - selectors
 - advanced combinators, 293
 - compound styles, 292–294
 - custom class styles, 286–288
 - custom hyperlinks, 292–294
 - ID style, 290–292
 - multiple selectors, 292
 - overview, 274
 - tag redefine style, 288–290
 - types of, 286
 - site maintenance with, 272
 - style management with, 272
 - tables, styling, 324, 326–327
 - type style, 305–307
 - as Web standard, 272–273
 - .css file extension, 276
 - CSS Vault, 331
 - CSS Zen Garden, 330–331, 501–503
 - { } curly braces, 274
 - cursor property, 321
 - custom class styles, 286–288
 - custom hyperlinks
 - color values for, 293–294
 - overview, 292
 - custom server-based search tool, 21
 - customer service
 - of hosting plans, 557
 - overview, 23
 - CuteFTP, 574
 - Cyan, Magenta, Yellow, and Black (CMYK) colors, 177–178
 - Cyberduck, 574
-
- ## D
-
- daily blog entries, 457
 - daily interactive content, 452–457
 - daily tip or news item, 452–456
 - daily word game, 457
 - data encryption, 396–398
 - data transfer and page views, 559

- database
 - described, 24
 - need for, determining, 26–27
 - uses for, 24–25
 - Davidson, Mike (sIFR fonts), 143
 - declarations
 - CSS (Cascading Style Sheets), 274
 - DOCTYPEs (DTDs), 490
 - dedicated IP address, 559
 - deep menus, 362–363
 - definition (DTDs), 490
 - Delorie, 514
 - dependent files, transferring, 582
 - deprecated tags, 218–219
 - descendant selector style (CSS), 343
 - Description meta tag, 212, 601
 - design phase, 10
 - design theme, 125–129
 - “Designer’s Guide to Photoshop DVD” (Jenkins), 186
 - DHTML (Dynamic HyperText Markup Language), 365
 - DHTML menus, 365
 - Diallo, Amadou (photographer), 147
 - <dir> tag, 218
 - direct input, validation by, 533
 - display property, 311–312
 - display resolution, 46
 - dithering, 198–199
 - DNS transfer, 555
 - DOCTYPEs (DTDs)
 - adding, 209–211
 - choosing, 210
 - converting syntax by, 521–524
 - included in Web code, 485
 - overview, 209–210, 489
 - document relative paths, 477
 - domain names
 - availability of, checking, 550–553
 - finding a domain name for your client, 549
 - overview, 545–547
 - registering
 - activating your registered domain name, 555
 - with domain registrar, 553–554
 - with host provider, 554–555
 - overview, 553
 - selecting, 546–553
 - using domain name generators, 549–550
 - domain pointer, 560
 - DomainsBot, 552–553
 - dots per inch (dpi), 180
 - downloading files, 582
 - Dreamweaver (Adobe)
 - behaviors, 377
 - Browser Compatibility tool, 527–528
 - Clean Up HTML/XHTML command, 520–521
 - Clean Up Word feature, 518–519
 - Convert Syntax command, 522
 - DOCTYPE, adding, 493–495
 - image maps added to graphics, 440–441
 - Link Checker tool, 529–530
 - linking CSS with, 284–286
 - multimedia files, adding, 444–447
 - multipart rollover effects, creating, 427–432
 - overview, 95
 - remote FTP connection, setting up a, 576–579
 - rollovers in, 377–379
 - sound, adding, 450–452
 - Spry Menu Bar widget, 379–380
 - templates
 - comment tags in, 467
 - creating, 464–465
 - creating a template with editable regions, 465–467
 - preparing a page to become a template, 462–463
 - template-based page, creating a, 467–468
 - used to transfer files, 582–585
 - Dreamweaver CS4 All-in-One For Dummies* (Jenkins), 462
 - DTDs. *See* DOCTYPEs
 - duplicate page
 - submissions, avoiding, 597
 - Dynamic HyperText Markup Language (DHTML), 365
-
- ## E
-
- e-commerce component, deciding whether to use a, 66
 - tag, 97, 221–222
 - e-mail
 - HTML e-mail or newsletter, building a, 351–356
 - hyperlinks for, 237–238
 - spam, protecting your e-mail addresses from, 238–239
 - embossed media type, 279

- e-newsletters used for attracting visitors, 28–30
- errors, finding and replacing, 516–517
- ethical techniques for Search Engine Optimization (SEO), 595–597
- Evans, Brian (form-processing script creator), 398
- event handler scripts, 372
- Events page, deciding whether to use a, 66
- expandable layout, 132–137
- Expression Web (Microsoft), 95
- eXtensible HyperText Markup Language. *See* XHTML
- eXtensible Markup Language (XML), 489
- extension style (CSS), 321–322

F

- failing acceptably, 538–539
- FAQs (frequently asked questions), 21
- faster page download time with CSS, 272
- federal privacy laws for Web forms, 392
- Fetch, 574
- file extensions, 100–101
- file formats for Web graphics
 - choosing, 195–196
 - comparison of, 192
 - GIF (Graphic Interchange Format), 193–196
 - JPG (Joint Photographic Experts Group) format, 194–196
 - overview, 191–193
 - PNG (Portable Network Graphics) format, 195–196
 - file size for Web graphics, 182
- File Transfer Protocol. *See* FTP
- filenames, choosing, 100–101
- FileZilla, 574
- filter property, 321–322
- final site testing, performing, 585–587
- FindLegalForms Privacy Policy, 17–18
- Terms of Use Agreement, 22
- finishing the project, 619–620
- Fireworks (Adobe) for optimization of Web graphics, 184
- outputting rollovers in, 374–377
- overview, 105
- slicing Web graphics, 187–191
- 508 compliance information, 23
- fixed-width layout, 132–137
- Flash menus, 364–365
- FlashFXP, 574
- fluid design, 132
- font sets, 141–142
- tag, 218
- font-family property, 305
- fonts
 - Arial, 141
 - cross-platform, 141, 142
 - Georgia, 141
 - graphics containing text, fonts in, 140–141
 - layout and design decisions, 140–142
 - list of safe-to-use HTML fonts, 141
 - Macintosh computers used to view, 142
 - in mock-ups, 159
 - sIFR, 143
 - text, 140–142
 - Verdana, 141
- font-size property, 306
- font-style property, 306
- font-variant property, 307
- font-weight property, 306
- footer
 - accessibility coding, 247
 - overview, 18
 - styling, 322–324
- form input tag labels, 248
- <form> tags, 400
- formatting HTML (HyperText Markup Language), 96–97
- lists, 268–269
- removing unwanted, 518–519
- tables
 - alignment of contents in table cell, 257
 - alignment of table, 256
 - background color, 261–263
 - border color, 261–263
 - borders, 257–258
 - cellpadding attribute, 258–260
 - cellspacing attribute, 258–260
 - headers, 260
 - height of table, 254
 - id attribute, adding, 254
 - nesting tables, 264
 - nowrap attribute, 260
 - overview, 253
 - splitting and merging table cells, 260–261


- tiling background
 - images, 263
 - width of table, 254–256
- forms
 - check box fields in, 409–410
 - creating, 401–412
 - deciding what visitor information to collect on, 392–394
 - federal privacy laws for, 392
 - graphics used in place of buttons in, 412
 - hidden fields in, 405
 - individual form fields, adding, 406–412
 - list/menu fields in, 407–408
 - overview, 392–394
 - radio button fields in, 410–411
 - relevant information, collecting only, 393
 - security issues
 - data encryption, using, 396–398
 - overview, 394
 - shared SSL, using, 396
 - SSL digital security certificates, using, 394–396
 - third-party services used for credit card processing, 396
 - simplifying collection of information on, 393–394
 - Spry validation
 - adding Spry validation fields to a form, 417–419
 - overview, 416
 - widgets for, 416–417
 - structure of
 - creating, 401–405
 - overview, 400–401
 - text fields in, 406–407
 - Thank You page, 397–399
 - validating
 - adding a Validate Form behavior to an existing form, 413–416
 - overview, 412–413
 - Spry validation, 416–419
 - testing the validation, 416, 419–421
 - 404 File Not Found page, creating, 588–590
 - 401 Unauthorized Access page, creating, 587–590
 - 14–15-inch monitor, 130
 - The Free Dictionary Web site, 457
 - freeware
 - code editors, 93
 - color picker, 111
 - graphics software applications, 105
 - online image map editor, 439
 - online validation tools, 532–536
 - for optimization of Web graphics, 184
 - frequently asked questions (FAQs), 21
 - FTP Client, 574
 - FTP (File Transfer Protocol)
 - browser interface
 - component used for, 574
 - choosing program for, 572–574
 - dependent files, transferring, 582
 - downloading files, 582
 - Dreamweaver used to transfer files, 582–585
 - integrated software
 - program for, 573
 - Internet control panel
 - used for, 573
 - overview, 572
 - putting files on the remote server, 582
 - remote connection, setting up a, 575–579
 - stand-alone software
 - program for, 572–574
 - uploading files, 582
 - Fugu SFTP, 574
 - FusionBot Free Package, 22

G

 - Gardner, Susannah (*Blogging For Dummies*), 33
 - general site navigation
 - element of wireframes, 70
 - general steps for organizing site content, 85–86
 - general usability of navigation system, 361
 - Georgia fonts, 141
 - GIF settings, 197–200
 - global hyperlinks, 230–231
 - “Going to Print” (Meyer), 182
 - GoLive (Adobe), 95
 - Google AdWords, 594, 595
 - Google Analytics, 48
 - Google Checkout, 116
 - Google Site Search, 21–22, 453
 - graphics
 - above the fold location for information about products/services/benefits, 161
 - additional Web graphics, creating, 171–173

- graphics (*continued*)
 - CMYK (Cyan, Magenta, Yellow, and Black) colors, 177–178
 - color gamut warnings, 178–180
 - color mode for, 177–178
 - containing text, 140–141
 - copyright, 78
 - dithering, 198–199
 - elements to appear on every page, determining, 68–69
 - file formats for
 - choosing, 195–196
 - comparison of, 192
 - GIF (Graphic Interchange Format), 193–196
 - JPG (Joint Photographic Experts Group) format, 194–196
 - overview, 191–193
 - PNG (Portable Network Graphics) format, 195–196
 - file size for, 182
 - fonts in, 140–141
 - gathering
 - artists, hiring freelance, 76–77
 - copyright issues, 78
 - overview, 73–75
 - stock images, licensing, 77–80
 - HTML, 223–229
 - hyperlinks, 234–237
 - ICC profile, 201
 - image maps, adding, 440–441
 - Internet connection
 - speeds and, 198
 - jaggies, 201
 - lossy compression format, 198
 - matte color, 199–201
 - optimization of
 - color reduction
 - algorithm for, 197
 - Fireworks used for, 184
 - freeware for, 184
 - GIF settings, 197–200
 - guidelines for, 197
 - ImageReady used for, 184
 - JPG settings, 200–201
 - naming conventions for, 202
 - output options for, 201–204
 - overview, 183
 - PNG-8 settings, 197–200
 - PNG-24 settings, 200
 - previewing, 187
 - program for, choosing, 107, 183–184
 - Save for Web & Devices (Photoshop & Illustrator) used for, 184–186
 - settings for, 196–201
 - overview, 176–182, 223–224
 - page size for, 182
 - print graphics, 176–177
 - printable, 181–182
 - programs, 102–107
 - raster programs for, 176
 - resolution, 180–182
 - RGB (Red, Green, and Blue) colors, 177–178
 - slicing
 - drag and release used for, 187
 - in Fireworks, 187–191
 - guides, creating slices from, 187–189
 - in Illustrator, 187–191
 - in ImageReady, 187–191
 - overview, 186–187
 - in Photoshop, 187–191
 - Slice tool used for, 187–189
 - styling, 327–328
 - transparency settings, 199, 200
 - unit of measure for, 182
 - used in place of buttons in Web forms, 412
 - vector programs for, 176
 - graphics software
 - applications
 - cost comparison chart for, 106–107
 - freeware, 105
 - graphics programs, 102–107
 - overview, 102
 - raster (bitmap) programs, 102–107
 - shareware, 105
 - vector programs, 102–107
 - Web graphic optimization programs, 107
 - greek text, 71–72
 - grid system, designing layouts using, 150–151
 - Grigg, Kurt (Comet Trail script), 425
 - GUI (graphical user interface), 148–152
 - guides, creating slices from, 187–189
-
- ## H
- <h1> tag, 220–221
 - <h2> tag, 220–221
 - <h3> tag, 220–221
 - <h4> tag, 220–221
 - <h5> tag, 220–221
 - <h6> tag, 220–221
 - hand coding your pages, 214–217
 - handheld media type, 279
 - hand-submitting the URL to search engines, 605–608
 - hard drive space, 559

- Harris, Andy (*HTML, XHTML, and CSS All-in-One Desk Reference For Dummies*), 182
- `<head>` tag, 97, 208–209
- header graphics, 171–172, 220–221, 260, 322
- heading tags, 220–221
- height of table, 254
- height property
- box style (CSS), 313
 - positioning style (CSS), 318
- hexadecimal values for colors, 110–114
- hidden fields in Web forms, 405
- hidden text, avoiding, 596
- home page
- information on, 16–17
 - keywords on, 17
 - mock-ups for, 157
 - naming, 101
- host provider, registering
- domain names with, 554–555
- hosting plans
- cancellation policy of, 557
 - customer service of, 557
 - data transfer and page views, 559
 - dedicated IP address, 559
 - domain pointer, 560
 - evaluating hosting plan packages, 557–560
 - features of, 558–560
 - hard drive space, 559
 - overview, 555
 - referrals for, 557
 - researching, 555–557
 - site access of, 557
 - uptime, 559
 - user accounts, 559
 - Web-based statistics, 559
- hotspots, 442–443
- hover hyperlinks, 293
- `href` attribute, 230
- `.htaccess` file, editing, 590–591
- HTML (HyperText Markup Language). *See also* HTML tags
- accessibility coding
- access keys, 247
 - alternate text attribute in an image tag, 245
 - contrasting foreground and background colors, 247
- footer links, 247
- form input tag labels, 248
- hyperlink targets, 247
- link tags, 246
- long description, 246
- object labels, 246
- overview, 245
- site map page, 247
- tab index, 248
- table title attributes, 246
- title attribute for hyperlinks, 246
- title tag, 245
- attributes, 96–97, 217
- basic HTML, setting up
- `<body>` tag, 208–209
 - coding pages by hand, 214–216
 - DOCTYPE, adding a, 209–211
 - `<head>` tag, 208–209
 - `<html>` tag, 208–209
 - meta tags, adding, 211–214
 - overview, 208–209
 - page title, adding a, 209
- building a Web page, 98–100
- coding your pages
- by hand, 214–217
 - overview, 217–219
 - using code editor, 217
- compliance information, 23
- CSS markup, labeling
- content for, 243–245
- deprecated tags, 218–219
- described, 92
- DOCTYPEs (DTDs), 491–492
- e-mail or newsletter, building a, 351–356
- file extensions, 100–101
- files, adding media-dependent style sheets to your, 282–284
- formatting, 96–97
- graphics
- alignment, 228–229
 - alternative text used for, 226–227
 - borders for, 225, 227
 - organizing, 223
 - overview, 223–224
 - padding, 227–228
 - sizing, 225
 - textual description of, 224
- `<head>` tag, 97
- hyperlinks
- for e-mail, 237–238
 - (frames only) window or parent, opening link in, 233
 - global, 230–231
 - for graphics, 234–237
 - image map links, 239–240
 - local, 230–231
 - locations for opening, 231–233
 - named anchor links, 241–243
 - named window or frame, opening link in, 232
 - new window, opening link in, 232
 - overview, 229–230
 - same window, opening link in, 232

- HTML (HyperText Markup Language) (*continued*)
 - list of safe-to-use HTML fonts, 141
 - marking text as, 140
 - markup validation
 - direct input, validation by, 533
 - failing acceptably, 538–539
 - with free online validation tools, 532–536
 - icons, adding validation, 540–541
 - noncompliant code, fixing, 536–538
 - overview, 531
 - proof of validation, obtaining, 540–541
 - retesting, 538
 - upload, validation by, 532
 - URL, validation by, 532
 - validation process, 531–541
 - with W3C Markup validator, 533–536
 - page content
 - graphics, adding, 223–229
 - overview, 219
 - text, adding, 219–223
 - rules for, 497–498
 - semantics, 217–218
 - structure of, 95–97
 - syntax, applying
 - consistent, 520–521
 - text
 - adding, 219–223
 - alignment, 222–223
 - bold, 221–222
 - headings, 220–221
 - italic, 221–222
 - title tag, 98
 - writing semantic, 495–498
 - XHTML compared, 95–96, 497–498
 - HTML Site Map
 - accessibility, 613–615
 - creating, 611–613
 - what to include on, 609–610
 - HTML tags
 - <a> tag, 230
 - <applet> tag, 218
 - <basefont> tag, 218
 - <body> tag, 97–98, 208–209
 - <center> tag, 218
 - deprecated tags, 218–219
 - <dir> tag, 218
 - tag, 97, 221–222
 - tag, 218
 - <form> tags, 400
 - <h1> tag, 220–221
 - <h2> tag, 220–221
 - <h3> tag, 220–221
 - <h4> tag, 220–221
 - <h5> tag, 220–221
 - <h6> tag, 220–221
 - <head> tag, 97, 208–209
 - heading tags, 220–221
 - <html> tag, 208–209
 - tag, 223–224
 - <isindex> tag, 218
 - <menu> tag, 218
 - overview, 96–97
 - <p> tag, 97
 - <s> tag, 218
 - <strike> tag, 218
 - tag, 97, 221–222
 - <title> tag, 98
 - <u> tag, 218
 - HTML, XHTML, and CSS All-in-One Desk Reference For Dummies* (Harris & McCulloh), 182
 - hyperlinks
 - accessibility coding, 247
 - active, 293
 - custom
 - color values for, 293–294
 - overview, 292
 - for e-mail, 237–238
 - (frames only) window or parent, opening link in, 233
 - global, 230–231
 - for graphics, 234–237
 - hover, 293
 - image map links, 239–240
 - local, 230–231
 - locations for opening, 231–233
 - named anchor links, 241–243
 - named window or frame, opening link in, 232
 - new window, opening link in, 232
 - normal, 293
 - null links, 437
 - overview, 229–230
 - same window, opening link in, 232
 - Search Engine Optimization (SEO), 600
 - verifying internal and external, 529–530
 - visited, 293
 - HyperText Markup Language. *See* HTML
-
- 
- ICC profile, 201
 - icons, adding validation, 540–541
 - id attribute, 243–244, 246, 254
 - ID style selectors (CSS), 290–292

ideal site visitor, 52–55
 Illustrator (Adobe)
 overview, 102–103
 slicing Web graphics,
 187–191
 image maps
 complex image maps,
 building, 441–443
 free online image map
 editor, 439
 graphic, adding an image
 map to a, 440–441
 hotspots, 442–443
 hyperlinks, 239–240
 overview, 439
 image of site, 15–16
 image padding, avoiding,
 596
 ImageReady (Adobe)
 overview, 107
 slicing Web graphics,
 187–191
 used for optimization of
 Web graphics, 184
 tag, 223–224
 industry-specific pages,
 deciding whether to
 use, 67
 inline boxes, 303
 inline CSS (Cascading Style
 Sheets), 275
 Inman, Shaun (IFR font
 system), 143
 intentionally adding an
 error to mock-ups, 169
 interactivity
 daily interactive content,
 452–457
 Google Site Search, 453
 image maps, 439–443
 with JavaScript, 424–426
 multimedia files, 444–452
 with multipart rollover
 effects, 426–432
 new browser window,
 432–439

internal CSS (Cascading
 Style Sheets), 275
 Internet usage statistics,
 45–48
 intranet, 132
 <isindex> tag, 218
 italic text, 221–222

J

jaggies, 201
 Java applets menus, 365
 JavaScript
 converting a daily tip
 JavaScript into an
 external .js file,
 455–456
 described, 47
 free resources for, 425
 interactivity with,
 424–426
 multitier menus, 364
 percentage of Internet
 users leaving
 JavaScript enabled in
 their browsers,
 checking, 47
 W3Schools tutorial for,
 426
 Jenkins, Sue
 “Designer’s Guide to
 Photoshop DVD,” 186
 *Dreamweaver CS4 All-in-
 One For Dummies*, 462
 JPG settings, 200–201

K

keyword spamming, 600
 keywords
 described, 17
 on home page, 17
 limited power of, 83
 meta tags for, 83

Search Engine
 Optimization (SEO)
 described, 598
 maximizing, 598–600
 meta tags for, adding,
 601–603
 searches for similar
 companies,
 performing, 48–51
 Keywords meta tag, 212,
 601

L

languages, deciding if text
 needs to appear in
 multiple, 67–68
 layers
 overview, 337–338
 styling, 329
 layouts
 accessible layouts,
 creating, 333
 checklist, 150–151
 design decisions
 color palette, choosing
 a, 138–140
 expandable layout,
 132–137
 fixed-width layout,
 132–137
 fonts, choosing, 140–142
 overview, 129–130
 printing the layout,
 choosing a method for,
 137
 size for site, choosing,
 130–132
 layers-based layouts
 adding a layer to a page,
 338–339
 advantages of, 336
 CSS layers-based layout,
 creating, 340–344
 CSS layers-based layout,
 Dreamweaver
 templates for, 356–358

layouts (*continued*)

- CSS layers-based layout, styling, 344–349
- online resources for, 356–358
- overview, 336, 338
- tables-based layouts compared, 336
- options for text navigation menus, 366–367
- standards-compliant layouts, creating, 333–334
- tables-based layouts
 - advantages of, 349–350
 - HTML e-mail or newsletter, building a, 351–356
 - overview, 349–351
- LCD monitors, resolutions for, 130
- Link Checker tool (Dreamweaver), 529–530
- links. *See* hyperlinks
- liquid design, 132
- list/menu fields in Web forms, 407–408
- lists
 - content, adding, 268–269
 - creating, 266–268
 - formatting, 268–269
 - nested, 266–268
 - ordered, 264–265
 - overview, 264
 - styling, 324–326
 - types of, 264–265
 - unordered, 264–265
- local hyperlinks, 230–231
- lossy compression format, 195, 198
- LZW (Lempel-Ziv-Welch) lossless compression, 193

M

- Macintosh computers used to view fonts, 142
- MailChimp Resource Center, 30, 356
- marketing and sales content of site
 - biography, 20
 - company information, 19
 - CSS compliance information, 23
 - customer service (help), 23
 - 508 compliance information, 23
 - frequently asked questions (FAQs), 21
 - HTML compliance information, 23
 - news and press information, 20
 - overview, 118–19
 - portfolio, 21
 - product/service information, 20
 - RSS feeds, 24
 - shopping cart, 22–23
 - site credits, 23
 - site search, 21–22
 - terms of service, 22
 - video and podcasts, 20–21
 - XHTML compliance information, 23
- markup language, 92. *See also* HTML
- Mascheck, Adam (free online image map editor), 439–440
- master CSS file, 294–300
- matte color, 199–201
- Max Design, 331
- Max Design Web Standards Checklist, 488

- McCulloh, Chris (*HTML, XHTML, and CSS All-in-One Desk Reference For Dummies*), 182
- media types (CSS), 279–284
- <menu> tag, 218
- merchant account, 118
- meta tags
 - adding, 211–214, 601–603
 - checking, 525
 - coding, 83
 - Content - Language meta tag, 212
 - Content - Type meta tag, 213
 - data, choosing, 80–81, 82–84
 - for description, 82–83
 - Description meta tag, 212
 - for keywords, 83
 - Keywords meta tag, 212
 - location for, 211
 - overview, 82, 84, 211
 - refresh meta tag, 213
 - revised meta tag, 213
 - robots meta tag, 212
 - using, 213–214
 - viewing, 49
- Meyer, Eric (“Going to Print”), 182
- minimum requirements to include on a site, 16–18
- mock-ups
 - above the fold, designing, 160–163
 - additional Web graphics, creating, 171–173
 - background images in, 173
 - blocking out parts of the page, 157–160
 - changes, making, 170
 - checklist for, 167
 - color in, 160
 - creating, 157–166
 - described, 157

design unification as
 reason for creating, 154
 easy modification as
 reason for creating, 154
 fonts in, 159
 header graphics in,
 171–172
 for home page, 157
 including navigation links
 in rollover state,
 167–168
 intentionally adding an
 error to, 169
 navigation links in, 159
 presentation of, 169–171
 reasons for creating,
 154–155
 reviewing, 166–167
 revisions, limiting number
 of, 155
 rollover graphics in, 172
 satisfaction as reason for
 creating, 155
 site map as starting point
 for, 155–156
 squint test for, 165
 subnavigation, showing,
 167–168
 tips for designing, 162
 unifying the layout with
 design elements,
 163–166
 visual representation as
 reason for creating, 154
 written client approval on
 design, importance of,
 170
 monthly processing fees,
 118
 Morris, Tee (*Podcasting For
 Dummies*), 21
 mouseover state, 167
 multimedia files, 444–452
 multipart rollover effects,
 426–432
 multitier menus, 363

N

named anchor links,
 241–243
 named window or frame,
 opening link in, 232
 naming home page, 101
 navigation system
 assessing the navigational
 needs of your site,
 360–361
 choosing, 364–366
 creativity in, 147
 CSS list navigation menus,
 383–390
 deep menus, 362–363
 DHTML menus, 365
 examples of, 144
 expandability of, 361
 Flash menus, 364–365
 forms (jump) menus, 364
 general usability of, 361
 Java applets menus, 365
 JavaScript multitier
 menus, 364
 location of, 145
 main navigation links, 360
 multitier menus, 363
 multitier Spry menus in
 Dreamweaver
 creating, 380–383
 overview, 379–380
 non-navigational links,
 360
 overview, 142–144, 361
 researching, 145
 rollover button graphic
 navigation menu
 building rollovers,
 372–373
 Dreamweaver, creating
 rollovers in, 377–379
 event handler scripts in,
 372
 Fireworks, outputting
 rollovers in, 374–377
 overview, 371
 preload scripts in, 372
 rollover scripts in, 372
 rollover menus, 364
 single-tier menus, 363
 source code for, 145
 style of, 145
 submenus in, 146
 subnavigation links, 360
 target audience of, 361
 text navigation menus,
 366–371
 text-only menus, 364
 tree-style menus, 365
 usability of, 142
 Web application menus,
 365
 wide menus, 362
 nested lists, 266–268
 nested templates, 461
 nesting tables, 264
 netiquette, 29, 31
 Network Solutions, 550–551
 new browser window
 Dreamweaver used to add
 pop-up window,
 436–439
 hand-coding script to
 launch pop-up window,
 434–436
 launching, 432–439
 when to launch, 432–434
 new content, regularly
 adding, 617–619
 new window, opening link
 in, 232
 news and press information
 overview, 12
 releases page, deciding
 whether to use a
 news/press, 66
 19-inch monitor, 130
 96 ppi, setting resolution
 to, 181

noncompliant code, fixing, 536–538
non-navigational links, 360
nowrap attribute, 260
null links, 437

O

object descriptions, embedding, 600–601
object labels, 246
offline site updates, 616–617
“Online Contest or Illegal Lottery?” (Rothkin), 40–41
online resources
 for CSS (Cascading Style Sheets), 329–331
 for Internet usage statistics, 45–47
 for layers-based layouts, 356–358
online site updates, 616
ordered lists, 264–265
output options for optimization of Web graphics, 201–204
overflow property, 320
oversubmitting, avoiding, 597

P

<p> tag, 97
padding graphics, 227–228
padding property, 313
page content (HTML), 223–229
page headers. *See* header graphics
page layout programs, 103–104
page titles
 adding, 209

 checking, 525
 choosing, 80–82
page-break-before/-after property, 321
Pantone color, finding hexadecimal equivalent of, 111–114
paragraphs, styling, 322
parsing, 470
paths
 adjusting paths in an SSI file from document relative to site-root relative, 478–479
 document relative, 477
 edited to work with SSIs, 478–479
 site-root relative, 477–478
PayPal, 22, 115–116
pay-per-click Internet advertising, 594
perspective of visitors, 58–59
phases of a Web project, 10–11
Photoshop (Adobe)
 overview, 102–103
 slicing Web graphics, 187–191
pixels per inch (ppi), 180–181
placeholder page
 customization of creating the page, 563–566
 overview, 562
 styling the page, 566–568
described, 545
 overview, 560–562
 uploading, 568–570
placement (left, top, right, bottom) property
 positioning style (CSS), 320
planning phase, 10

PNG-8 format, 195, 197–200
PNG-24 format, 195, 200
Podcasting For Dummies (Morris, Tomasi, & Terra), 21
podcasts, 20–21
polls and calculators used for attracting visitors, 37–40
position property, 317–318
positioning blocks, 304
positioning style (CSS), 317–321
prelaunch testing
 on multiple browsers, 511–514
 on multiple platforms, 511–514
 overview, 510–511
 testing a page, 514–515
 with third-party testing tools, 515
 Web-testing checklist, 511
print graphics, 176–177
print media type, 280
printable Web graphics, 181–182
printing the layout, choosing a method for, 137
privacy policy, 17–18
Products page, deciding whether to use a, 65–66
products/services Web site database, 18
professional portfolio, purpose of site as, 12
profile, adding a, 35–36
programmers
 dynamic content needs determining the hiring of, 119–120
 finding, 121–122
 when to hire, 119–121

projection media type, 280

publishing your site

- final site testing, performing, 585–587
- 404 File Not Found page, creating, 588–590
- 401 Unauthorized Access page, creating, 587–590

FTP (File Transfer Protocol)

- browser interface component used for, 574
- choosing program for, 572–574
- dependent files, transferring, 582
- downloading files, 582
- Dreamweaver used to transfer files, 582–585
- integrated software program for, 573
- Internet control panel used for, 573
- overview, 572
- putting files on the remote server, 582
- remote connection, setting up a, 575–579
- stand-alone software program for, 572–574
- uploading files, 582
- .htaccess file, editing, 590–591
- overview, 591–592
- test directory
 - creating, 580–581
 - deleting, 591–592
 - overview, 579–580
- purpose of site, 11–14

Q

questions

- for determining vision for the site, 62–70
- to help define ideal site visitor, 53–55

QuickMenu, 145

QuickTime Player, 450, 451

R

radio button fields in Web forms, 410–411

raster (bitmap) programs. *See also* Photoshop (Adobe)

- overview, 102–107
- for Web graphics, 176

RealPlayer, 450–451

referrals for hosting plans, 557

refresh meta tag, 213

registering domain names

- activating your registered domain name, 555
- with domain registrar, 553–554
- with host provider, 554–555
- overview, 553

relevancy of site, 616–619

remote connection, setting up a, 575–579

repeating regions in templates, 461

research on your competition

- evaluating competitors' sites, 51–52
- industry-related associations and organizations as source for, 52

keyword searches for similar companies, performing, 48–51

meta tags, viewing, 49

overview, 48

search engine used for, 49–51

summary of facts, writing out a, 52

resolution

- for LCD monitors, 130
- 96 ppi, setting resolution to, 181
- Web graphics, 180–182

resources

- for JavaScript, 425
- online resources
 - for CSS (Cascading Style Sheets), 329–331
 - for Internet usage statistics, 45–47
 - for layers-based layouts, 356–358
 - for Web standards, 488
- revised meta tag, 213

revisions, limiting number of, 155

RGB (Red, Green, and Blue) colors, 110, 177–178

rights-managed (RM) images, 77–80

robots, 597

robots meta tag, 212

rollover button graphic navigation menus, 371–379

rollover buttons, 47

rollover graphics, 172

rollover menus, 364

rollover scripts, 372

rollover state, 167

rollover text-based navigation bar, creating a, 367–371

Rothkin, Ira (“Online Contest or Illegal Lottery?”), 40–41
royalty-free (rf) images, 77–80
RSS feeds, 24, 67

S

-
- <s> tag, 218
same window, opening link in, 232
Save for Web & Devices (Photoshop & Illustrator) used for optimization of Web graphics, 184–186
saving Web files, 100–101
Scalable Inman Flash Replacement (sIFR) fonts, 143
scheduling site updates, 617
screen media type, 280
Search Engine Optimization (SEO)
cheating, avoiding, 597
conversion rate, 594
descriptive text, including, 600
duplicate page submissions, avoiding, 597
ethical techniques for, 595–597
hidden text, avoiding, 596
hyperlinks, including, 600
image descriptions, embedding, 600–601
image padding, avoiding, 596
keywords
described, 598
maximizing, 598–600
meta tags for, adding, 601–603
meta tags, adding
keyword and description, 601–603
object descriptions, embedding, 600–601
oversubmitting, avoiding, 597
overview, 594
robots, 597
spiders, 597
submitting a site to search engines
hand-submitting the URL, 605–608
overview, 605
tools for, 606
waiting for site to be listed, 608
tag padding, avoiding, 596
titles, updating bland page, 603–605
traffic, 594
searching with Google, 453
Section 508, 504–508
Secure Sockets Layer. *See* SSL
security
shopping cart transactions, 118–119
spam, protecting your e-mail addresses from, 238–239
SSL (Secure Sockets Layer), 118–119
Web forms
data encryption, using, 396–398
overview, 394
shared SSL, using, 396
SSL digital security certificates, using, 394–396
third-party services used for credit card processing, 396
selectors (CSS)
advanced combinators, 293
compound styles, 292–294
custom class styles, 286–288
custom hyperlinks, 292–294
ID style, 290–292
multiple selectors, 292
overview, 274
tag redefine style, 288–290
types of, 286
selling products, purpose of site as, 12
semantics, 217–218
SEO. *See* Search Engine Optimization
separate CSS files for different media types, creating, 280–282
Server-Side Includes. *See* SSIs
Services page, deciding whether to use a, 66
services, purpose of site to market, 12
setup and application fees, 118
17-inch monitor, 130
17–19-inch monitor, 130
SGML (Standard Generalized Markup Language), 489
shared SSL, using, 396
shareware, 105
Shea, Dave (CSS Zen Garden), 330
shopping carts, 22–23, 114–119
sIFR fonts, 143
Simple Mail Transfer Protocol (SMTP), 399
site credits, 23
site launch phase, 10

- site maintenance
 - with CSS (Cascading Style Sheets), 272
 - performing, 616–617
 - site maps
 - accessibility coding, 247
 - building, 87–90
 - HTML Site Map
 - accessibility, 613–615
 - creating, 611–613
 - what to include on, 609–610
 - overview, 18, 87–88, 608–609
 - reviewing, 90
 - as starting point for mock-ups, 155–156
 - steps for creating, 88–90
 - working from a, 155–156
 - site reports, generating, 530–531
 - site search, 21–22
 - site-root relative paths, 477–478
 - Slice tool, 187–189
 - slicing, 186–191
 - slide shows, creating, 447–450
 - small- to medium-sized
 - business, statement of purpose for, 14
 - smaller Web site projects, organizing site content for, 86
 - SmartFTP, 574
 - SMTP (Simple Mail Transfer Protocol), 399
 - sound, adding, 450–452
 - source code for navigation systems, 145
 - source formatting, applying, 521
 - sources for stock images, 78–80
 - spam, protecting your
 - e-mail addresses from, 238–239
 - spelling, checking the, 517–518
 - spiders, 597
 - splitting and merging table cells, 260–261
 - Spry Menu Bar widget (Dreamweaver), 379–380
 - Spry validation
 - adding Spry validation fields to a form, 417–419
 - overview, 416
 - widgets for, 416–417
 - squint test for mock-ups, 165
 - SSIs (Server-Side Includes)
 - creating SSIs, 472–475
 - editing an SSI file, 471
 - “file” used in SSI link, 475
 - guidelines for using SSIs, 471–472
 - including an SSI file inside a page, 469–470
 - including SSIs, 472–475
 - overview, 469
 - paths edited to work with SSIs, 478–479
 - templates compared, 479–480
 - testing SSIs, 475
 - “virtual” used in SSI link, 472, 475
 - when to use, 480
 - SSL (Secure Sockets Layer)
 - overview, 118–119
 - in Web forms, 394–396
 - Standard Generalized Markup Language (SGML), 489
 - standards-compliant code, importance of writing, 484–485
 - standards-compliant
 - layouts, creating, 333–334
 - statement of purpose, 13–14
 - stickiness of Web site, 21, 33
 - stock images, 77–80
 - `<strike>` tag, 218
 - `` tag, 97, 221–222
 - style management with CSS (Cascading Style Sheets), 272
 - style property, 314
 - submenus in navigation systems, 146
 - submitting a site to search engines
 - overview, 605
 - Search Engine Optimization (SEO)
 - hand-submitting the URL, 605–608
 - overview, 605
 - tools for, 606
 - waiting for site to be listed, 608
 - subnavigation in mock-ups, showing, 167–168
 - subnavigation links, 360
 - summary of facts, writing out a, 52
 - sweepstakes used for attracting visitors, 32–33
 - SWOP (standard Web offset press), 178
-
- ## T
- tab index, 248
 - tabindex attribute, 248
 - table cells layout option for text navigation menus, 367
 - table title attributes, 246

- tables
 - content added to cells, 252–253
 - formatting
 - alignment of contents in table cell, 257
 - alignment of table, 256
 - background color, 261–263
 - border color, 261–263
 - borders, 257–258
 - cellpadding attribute, 258–260
 - cellspacing attribute, 258–260
 - headers, 260
 - height of table, 254
 - id attribute, adding, 254
 - nesting tables, 264
 - nowrap attribute, 260
 - overview, 253
 - splitting and merging table cells, 260–261
 - tiling background images, 263
 - width of table, 254–256
 - overview, 249–252
 - structure of, 251–252
 - styling, 324, 326–327
- tables-based layouts
 - advantages of, 349–350
 - HTML e-mail or newsletter, building a, 351–356
 - layers-based layouts
 - compared with, 336
 - overview, 349–351
- tag pairs, 96
- tags in HTML. *See* HTML tags
- target attribute, 231–232, 247
- target audience
 - defining
 - competition, researching your, 48–52
 - Internet usage statistics, gathering, 45–48
 - market research, performing informal, 44–45
 - overview, 44
 - described, 44
 - of navigation system, 361
 - visual profile, 55
- templates
 - building Web sites with
 - creating a template, 464–465
 - creating a template with editable regions, 465–467
 - Dreamweaver templates, 461–462
 - nested templates, 461
 - optional editable regions, 461
 - overview, 460–461
 - preparing a page to become a template, 462–463
 - repeating regions, 461
 - SSIs compared, 479–480
 - template-based page, creating a, 467–468
 - when to use, 480
 - overview, 460–461
- terms of service, 22
- Terra, Evo (*Podcasting For Dummies*), 21
- test directory
 - creating, 580–581
 - deleting, 591–592
 - overview, 579–580
- testing
 - prelaunch
 - on multiple browsers, 511–514
 - on multiple platforms, 511–514
 - overview, 510–511
 - testing a page, 514–515
 - with third-party testing tools, 515
 - Web-testing checklist, 511
 - SSIs (Server-Side Includes), 475
 - validation, 416, 419–421
- testing phase, 10
- text
 - above the fold location for information about products/services/benefits, 161
 - alternative text used for graphics, 226–227
 - antialiasing, 164–165
 - bold text, 221–222
 - elements to appear on every page, determining, 68–69
 - fields in Web forms, 406–407
 - fonts, choosing, 140–142
 - gathering content
 - copywriter, hiring a, 75–76
 - overview, 73–75
 - graphics containing, 140–141
 - greeking, 71–72
 - HTML
 - adding, 219–223
 - alignment, 222–223
 - bold, 221–222
 - headings, 220–221
 - italic, 221–222
 - marking text as, 140
 - text navigation menus
 - horizontal list layout option for, 366
 - layout options for, 366–367
 - overview, 366
 - rollover text-based navigation bar, creating a, 367–371

table cells layout option
for, 367

vertical list layout option
for, 366

text-align property, 311

Thank You page, 397–399

third-party
prelaunch testing with
third-party testing
tools, 515

services used for credit-
card processing, 396

shopping carts, 116–117

30-inch monitor, 130

301 redirect, 213

tiling background images,
263

title attribute, 246

<title> tag, 98, 245

titles, updating bland page,
603–605

Tomasi, Chuck (*Podcasting
For Dummies*), 21

traffic, 594

transparency settings,
199–200

tree-style menus, 365

tty media type, 280

tv media type, 280

24-inch monitor, 130

20-inch monitor, 130

20-inch+ monitor, 130

type style (CSS), 305–307

TypePad, 34

U

<u> tag, 218

unordered lists, 264–265

updating content, 18

uptime, 559

URL, validation by, 532

usability of navigation
systems, 142

user accounts, 559

V

validating
Web forms
adding a Validate Form
behavior to an existing
form, 413–416

overview, 412–413

Spry validation, 416–419

testing the validation,
416, 419–421

your markup, 526–527

validation process
advantages of using, 510

cleaning up your code,
515–524

fixing common code
errors, 524–531

HTML and CSS markup
validation, 531–541

overview, 510

Validator (Dreamweaver),
526–527

vector programs. *See also*
Illustrator (Adobe)

overview, 102–107

for Web graphics, 176

Verdana fonts, 141

VeriSign, 119

(|) vertical line, 366

vertical-align property, 311

video and podcasts, 12–13

visibility property, 318–319

vision for the site, 62–70

visited hyperlinks, 293

visitors
attracting
blogs used for, 32–37

contests and
sweepstakes used for,
40–41

e-newsletters used for,
28–30

free tips and articles
used for, 31–32

overview, 27–28

polls and calculators
used for, 37–40

data on
collecting and using, 26

deciding what visitor
information to collect
on Web forms, 392–394

visual editors, 94–95

W

Wabi Sabi aesthetic, 147

waiting for site to be listed,
608

W3C Markup validator,
533–536

W3C Tutorial, 330

W3C Web site, 486–488

W3C (World Wide Web
Consortium), 243, 272,
333, 485–486, 485–488

W3C's CSS, 330

Web address
domain name section of,
547

extension section of,
547–548

parts of, 547

protocol section of, 547

www section of, 547

Web application menus,
365

Web editors
building a Web page,
98–100

code editors, 93–94

HTML structure,
understanding, 95–97

overview, 92

saving Web files, 100–101

selecting, 93–95

visual editors, 94–95

Web page structure, 97–98

Web sites, building. *See*
building Web sites

- Web standards
 - accessibility standards
 - overview, 503–504
 - Section 508, 504–508
 - CSS (Cascading Style Sheets) as, 272–273
 - CSS formatting
 - advantages of, 500–501
 - examples of, 501–503
 - HTML formatting
 - compared with, 499–500
 - overview, 498–499
 - presentation separate from content in, 500
 - DOCTYPEs (DTDs)
 - declaration, 490
 - definition, 490
 - Dreamweaver, adding a DOCTYPE in, 493–495
 - HTML DOCTYPEs, 491–492
 - included in Web code, 485
 - overview, 489
 - selecting, 489–493
 - XHTML DOCTYPEs, 492–493
 - HTML, writing semantic, 495–498
 - Max Design Web Standards Checklist, 488
 - overview, 484
 - resources for, 488
 - standards-compliant code, importance of writing, 484–485
 - uniform methods of coding HTML and XHTML, 485
 - W3C recommendations
 - for, 485–486
 - XHTML, writing semantic, 495–498
 - Web-safe color, 108–109
 - Weinman, Lynda (Web-safe color palette), 108
 - What We Do page, deciding whether to use a, 66
 - white-space property, 311
 - width of table, 254–256
 - width property
 - border style (CSS), 314
 - box style (CSS), 313
 - positioning style (CSS), 318
 - Windows Media Player, 450, 452
 - wireframes
 - advantages of, 72
 - content for a particular page element, 71
 - content that appears on every page element, 70
 - creating, 70–73
 - dynamic functionality element of, 71
 - educating client on use of, 72
 - general site navigation element of, 70
 - interactive components element of, 70
 - overview, 70–71
 - WordPress, 34, 36
 - word-spacing property, 310
 - World Wide Web Consortium (W3C), 243, 272, 333, 485–488
 - Wright, Matt (form-processing script creator), 398
 - writing, getting the client's approval in, 170, 463
 - W3Schools, 45, 47
 - W3Schools Tutorial, 330
 - WS_FTP, 574
 - www section of Web address, 547
 - WYSIWYG (what you see is what you get), 94
-
- ## X
-
- XHTML (eXtensible HyperText Markup Language)
 - compliance information, 23
 - described, 95–96
 - DOCTYPEs, 492–493
 - HTML compared, 95–96, 497–498
 - rules for, 497–498
 - syntax, applying
 - consistent, 520–521
 - writing semantic, 495–498
 - XML (eXtensible Markup Language), 489
-
- ## Y
-
- Yummy FTP, 574
-
- ## Z
-
- Zeldman, Jeffrey (Web site), 488–489
 - Z-index property, 319

Everything you need to know to create dazzling Web designs is in one of these minibooks

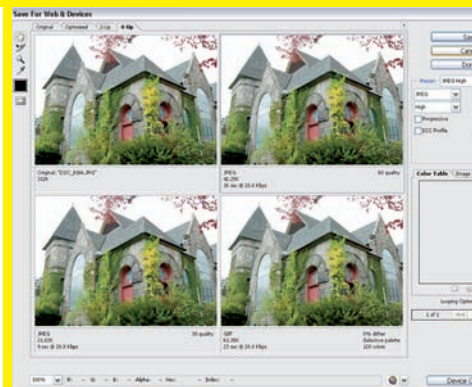
A Web designer is a graphic designer, creative organizer, visual communicator, markup language technologist, and cutting-edge trendsetter, all in one. This All-in-One guide helps you wear all those hats without losing your head! Learn to lay the groundwork, follow design rules, test your site, register a domain name, and more. Ready to get started?

- **Start here** — Book I covers planning, defining your target audience, choosing the right software, and more
- **Design it** — Book II acquaints you with HTML, CSS, and JavaScript®, plus how to choose a layout and optimize graphics
- **Build it** — Book III gets down to nuts and bolts: putting text, images, hyperlinks, and multimedia files together, organizing content, and building navigation systems
- **Does it work?** — Book IV teaches you how to test and validate so everyone can enjoy your site
- **Maintenance and more** — Book V helps you get your site online and keep it current



Open the book and find:

- How to choose a Web editor and graphics program
- Tips for attracting visitors
- Why and how to slice up graphics
- How to make your site accessible to the widest possible audience
- What a layers-based layout is
- How to use Dreamweaver® templates
- Ethical SEO techniques and how to use them
- When to ask for help from a pro



Go to **dummies.com**®
for more!

For Dummies®
A Branded Imprint of
WILEY

\$44.99 US / \$53.99 CN / £28.99 UK

ISBN 978-0-470-41796-6



Sue Jenkins is a professional designer with experience in Web sites, print media, logo design, and illustration. She is the author of several books and teaches Adobe software; has created training videos on Dreamweaver, Illustrator, and Photoshop; and operates Luckychair, a Web and graphic design studio (www.luckychair.com).