sitepoint

BUILD YOUR OWN WEB SITE THE RIGHT WAY USING HTML & CSS

BY IAN LLOYD

LEARNING HTML AND CSS HAS NEVER BEEN SO MUCH FUN!

Build Your Own Web Site The Right Way Using HTML & CSS (4 Chapter Sample)

Thank you for downloading this four-chapter sample of Ian Lloyd's book, *Build Your Own Web Site The Right Way Using HTML & CSS*, published by SitePoint.

This excerpt includes the Summary of Contents, Information about the Authors, Editors and SitePoint, Table of Contents, Preface, four chapters of the book, and the index.

We hope you find this information useful in evaluating this book.

Summary of Contents of this Excerpt

Summary of Additional Book Contents

Build Your Own Web Site The Right Way Using HTML & CSS

by Ian Lloyd

Build Your Own Web Site The Right Way Using HTML & CSS

by Ian Lloyd

Expert Reviewer: Marc A Garrett
Managing Editor: Simon Mackie
Technical Editor: Matthew Magain
Technical Director: Kevin Yank
Printing History:
First Edition: April 2006

Editor: Georgina Laidlaw
Index Editor: Bill Johncocks
Cover Design: Jess Mason
Cover Layout: Alex Walker

Notice of Rights

Notice of Liability

The author and publisher have made every effort to ensure the accuracy of the information herein. However, the information contained in this book is sold without warranty, either express or implied. Neither the authors and SitePoint Pty. Ltd., nor its dealers or distributors will be held liable for any damages to be caused either directly or indirectly by the instructions contained in this book, or by the software or hardware products described herein.

Trademark Notice

Rather than indicating every occurrence of a trademarked name as such, this book uses the names only in an editorial fashion and to the benefit of the trademark owner with no intention of infringement of the trademark.

Published by SitePoint Pty. Ltd.

424 Smith Street Collingwood
VIC Australia 3066.
Web: www.sitepoint.com
Email: business@sitepoint.com

ISBN 0-9752402-9-3
Printed and bound in the United States of America

About the Author

Ian Lloyd is a web designer and developer who believes passionately in the importance of web standards. He is a member of the Web Standards Project (http://webstandards.org/) and has spoken at several high profile web conferences, including South By Southwest in Austin, Texas, and @media in London. Unlike many of his peers, Ian does not run his own company but remains a "salary man", working for Nationwide Building Society in the UK (where he constantly harps on about web standards, accessibility and usability to anyone with a sympathetic ear!).

Ian lives in Swindon, UK, a town that is known only for two things:

- the famous "magic roundabout"—a mega roundabout that comprises five individual but joined roundabouts
- for being that place off *The Office* (thus making it second in dullness only to Slough)

That said, Ian does his best to get out of Swindon in his treasured air-cooled VW camper van (http://vwkombi.com/) when he's not glued to the laptop for one reason or another.

Ian is married to Manda, who really doesn't get all this techy stuff—or, for that matter, the Volkswagen fascination—but puts up with it regardless.

About The Expert Reviewer

Marc A. Garrett was born in Japan and raised in Stone Mountain, Georgia. After receiving a Bachelor of Arts at the University of Georgia, he earned a Juris Doctor at the American University Washington College of Law. He has co-authored two computer books, published by Macromedia Press and Apress; served as technical editor for several titles from O'Reilly and New Riders; and has written for Digital Web Magazine. Mr. Garrett specializes in web-based applications written in ColdFusion, PHP, and ASP.NET.

About The Technical Editor

Before joining the SitePoint team as a technical editor, Matthew Magain worked as a software developer for IBM and also spent several years teaching English in Japan. He is the organizer for Melbourne's Web Standards Group (http://webstandardsgroup.org/), and enjoys swimming, listening to classical jazz and drinking Coopers Pale Ale, though not all at the same time. Matthew lives with his wife Kimberley and their daughter Sophia.

About SitePoint

SitePoint specializes in publishing fun, practical and easy-to-understand content for web professionals. Visit http://www.sitepoint.com/ to access our books, newsletters, articles and community forums.

For Manda (who will, at the very least, read this page!)

Table of Contents

Preface

Congratulations on buying this book. Oh, wait a minute—perhaps you haven't yet. Perhaps you've just picked up the book in your local bookshop, and are trying to decide whether it's right for you. Why should this be the book that makes it into your shopping basket? The answer can be found in the title of the book: it's all about getting things right the first time, not learning bad habits—bad habits that you have to *un*-learn at a later date—for the purposes of getting a quick result.

Let's take a step back for a moment, and look at another skill that many people learn at some point in their lives: learning to drive. Apologies if that particular experience is *also* new to you, but stick with me. For many people, first driving lessons can be very confusing—you've got to figure out which pedals to press, in what order, and manage to get out of the driving school car park without hitting the other students. Meanwhile, other more experienced people just get into their cars, start the engine, and drive from A to B without even really thinking about what they're doing. These drivers may have picked up a few bad habits along the way, but if they learned with a proper driving instructor, the chances are that they were taught properly from the beginning, following a strict set of rules and guidelines to ensure they stay safe.

The driving instructor tells you to check your mirrors diligently, observe speed limits, and avoid cutting corners (literally as well as metaphorically!). Imagine, though, if the instructor told you not to worry about the speed limit signs, to "put your foot down" because the road is clear, or told you that the one-way sign "wasn't all that important at that time of night." It'd be a miracle if you passed your driving test, and the chances are that those bad habits would stay with you (so long as you could manage to keep your license).

Learning to build web pages can be a bit like that.

I've been designing and building web sites for around eight years now, but I can clearly remember the joy of creating my first site. Admittedly, in hindsight, it was a pretty nasty-looking web site, but it achieved the goal at the time—I had published something, and I was able to create it with the bare minimum of tools. It gave me an enormous sense of achievement, and made me want to learn more and create even better web sites.

At the time, there weren't that many books that really seemed to what I wanted, but I lapped up everything I could find, learning some tricks from books, and

getting other ideas from visiting web sites. But then I discovered that I'd been doing it all wrong. The books I had learned from had given me what later turned out to be poor advice, and the web sites I'd visited had themselves learnt from the same sources and made use of similar bad techniques. So, what had gone wrong?

In the early days of the web, when people first started properly to embrace the technology, to publish homepages, and to develop online corporate presences for their companies, they all realized fairly quickly that the medium was quite limiting. Necessity is the mother of invention, though. People began to coax out of their web pages tricks and displays that were never intended by the technologies they used, and the browsers helped them along the way by adding features that offered even more opportunities for this kind of behavior.

Numerous books have been written on the topics of web design and programming, as have many free tutorials that you can read on the 'Net. Many of them were written during those heady years, and were based on what seemed like best practices back then, but their authors were constrained by browsers that often rendered the same well-designed pages in vastly different ways. This meant that the tutorials' authors needed to resort to abusing various features of those browsers, such as using data tables to lay out pages. This certainly got many people building their first web pages, but it ensured that bad habits were ingrained at an early stage, and many people are still using these bad practices years later.

Web developers the world over have learnt bad habits (myself included) and must now try to un-learn them all. There's no longer a need for these practices—they often produce pages that are inflexible, slow to download, and difficult to maintain, but like the badly taught driver who insists on flouting the rules because it's worked for him so far, many developers find those outdated habits difficult to break.

I saw the light many years ago, and have tried to educate as many people as possible since. But for the eager beginner, those same old books are still peddling the same bad old ideas. This just *has* to stop. And it stops now.

You're not going to learn any bad habits in this book. Not one.

In this book, you'll learn the right way to do things. If there's a wrong way to do things—a way that cuts corners to save time but encourages bad techniques—we won't even tell you about it. Not even as a "by the way, you might try this…" You won't need to avert your eyes—we'll take care of that for you!

What is a Browser?

If you use Microsoft Windows, the browser is probably what you know as the "little blue e on the desktop" (shown in Figure 1), but is commonly called Internet Explorer. The majority of people don't stray beyond using this program for the purposes of viewing web pages—for many, Internet Explorer *is* the Internet.

Figure 1. Internet Explorer—the "little blue e on the desktop"

Internet Explorer (or **IE**, as we'll refer to it from now on) is the most commonly used browser, largely because Microsoft included it as part of the Windows operating system as far back as Windows 95 (this was later to come back and haunt Microsoft—it became the catalyst for a massive anti-trust trial which ruled that the company had stifled competition by bundling IE with the operating system to the exclusion of all others).

However, there are other browsers that you can use instead of IE. Currently riding an ever-increasing wave of popularity is Firefox,[1] a small, speedy program that has a number of attractive features that aren't available in IE (at the time of writing), and handles the features of some web pages better than IE can. It's also available for Windows, Mac OS X, and Linux, while the latest versions of IE are only available for Windows. The screen shots you'll see in this book were taken using Firefox on Windows XP, unless stated otherwise. We recommend that you download a copy of Firefox for the purposes of working through the exercises in this book.

Alternatively, you might like to try another browser that supports web standards well—check out BrowseHappy[2] for some more ideas.

Not interested in a different browser? Well, you can use IE, as indeed the majority of web users still do. In fact, you can be sure that everything you read here will work in all recent browsers without any real hiccups.

[1] http://www.mozilla.com/firefox/
[2] http://browsehappy.com/

Who Should Read this Book?

Does this sound like you?

- ❑ an absolute beginner—at least as far as creating web pages goes
- ❑ confident with using a computer, but not necessarily a "power user"
- ❑ someone who uses the Web a lot, enjoys other people's web sites, and would like to create your own

  Perhaps you have a hobby that you want to tell the world about, or you want to start a web site for a community that you belong to (be that a knitting circle or remote-control car racing club)

- ❑ quickly put off by the techno-babble that computer people tend to speak

  When you have conversations with your more tech-savvy relatives about a problem you're having with your computer—perhaps you've been struck down by a computer virus—do you find a lot of the jargon goes over your head?

- ❑ perhaps a little daunted about learning this new skill, but keen to learn (with some friendly hand-holding)

If any of the above descriptions strikes a chord with you, then this is the book to put in your shopping cart—we'll ease you in gently, and have you building web pages like a pro in no time!

Don't worry if you don't understand the terminology that your 15-year-old nephew (or cousin!) keeps spouting when you ask him about building web sites. I've assumed no prior knowledge of any of these terms, and I'll be guiding you all the way through the process of creating a web site from scratch. By the end of this book, you'll know how to build the site, how to secure hosting, how to promote the site, and how to keep it running once it's live.

The best part is this: what you learn in this book, you'll never have to un-learn. You'll be learning how to build sites the right way from the get-go.

What you'll Learn from this Book

By the time that you finish reading this book, and trying out the exercises contained within, you'll be able to build a complete web site—the right way—without incurring any costs for expensive software or web hosting.

Using an example web site, I'll guide you through the process of developing web pages from scratch. From these humble beginnings, great things will evolve! By the end of the book, you'll be able to create a web site that includes the following features:

- ❑ easy-to-use navigation
- ❑ a professional-looking site header
- ❑ a regularly updated news/events section
- ❑ a "Contact Us" page
- ❑ tables—the presentation of data in neatly organized grids
- ❑ attractive forms
- ❑ a simple image gallery
- ❑ a search engine that covers our site, as well as related sites
- ❑ simple statistics that you can use to see who's using your site, how they found your site (e.g. from search engines), and so on

You'll also learn how to manage your web site effectively, without it becoming a chore or getting too technical. I'll show you how you can:

- ❑ Get your own dot-com (or dot-net or dot-org) web address.
- ❑ Find somewhere to host your web site.
- ❑ Upload your files to your web site.
- ❑ Get feedback from visitors while avoiding spam emails.

How you'll Learn to Build your Web Site

This book will take you through each new topic using a step-by-step approach. It provides a mixture of examples and practical exercises that will soon have you feeling confident enough to try a little **HTML** for yourself.

HTML, Markup, CSS... Welcome to your First Bits of Jargon!

From here on in, you're going to see these phrases more and more. But what do they mean?

HTML HTML stands for HyperText Markup Language. It's the primary language that's used to create web pages, so you'll come to know it very well through the course of this book. We'll be using the latest version of HTML, which is called XHTML.

markup Imagine, if you will, that you're a newspaper editor. You've been passed a news story, but the text—from the heading through to the conclusion—is all the same size, and the headings, paragraphs, quotes, and other features of the text are not clearly indicated. It's just one big block of text. For starters, you'd probably want to emphasize the headline, maybe by displaying it in bold or italic text (or in all caps with an exclamation mark if you were working for a tabloid). As an editor, you'd probably grab a pen and start scribbling annotations on the print-out—an "h" here to signify a heading, "p"s here, there, and everywhere to show where paragraphs start and end, and "q"s to denote quotations.

This is essentially what markup is—a set of simple tags that suggest the structure of a document (this section is a heading, paragraph, a quote, etc.). We'll cover the various tags that HTML uses in detail a little later.

Markup isn't Computer Code

Markup is not the same thing as "code." Often, people incorrectly refer to markup as code, but code goes beyond the basic abilities of markup. With code, you can create programs, and do more dynamic things with your web page, while markup simply deals with the page's structure. So, if you want to impress your friends

and relatives, refer to markup rather than code. See, we told you we'd teach you good habits!

CSS CSS stands for Cascading Style Sheets. We'll be using a combination of HTML and CSS to create web sites. CSS is a language that lets you control how your web pages look, but we'll get into that in more detail later. For now, it's important that you know what the abbreviation stands for.

Building the Example Site

Each example that's presented in this book will be backed up with a sample of the markup you need to write, and a screen shot that shows how the results should look.

Each example is complete: nothing's missing. You'll see the picture build gradually, so you won't be left trying to guess how the example web site got to the point it's at. The files we'll use in all the examples are provided in a separate code archive (described in more detail in a moment).

What you Can Expect from the Example Web Site

- a fun web site project that will be built up through the chapters
- a complete web site that demonstrates all the features you're likely to need in your own web site
- all the XHTML and CSS used to progress the site through each chapter—in a single download

You can pick up the project at any point, so you don't need to worry about mistakes you might have made in previous chapters' exercises!

What this Book Won't Tell You

While it might be tempting to cram everything into one book and claim that the reader will learn everything in 24 hours, the honest truth is that this is not necessarily the right approach for everyone.

This book won't try to force-feed you everything there is to know about creating web pages; instead, it focuses on the most useful aspects that you'll find yourself using over and over again.

This book does not cover:

- ❑ JavaScript
- ❑ other programming languages, such as ASP or PHP
- ❑ advanced CSS techniques
- ❑ search engine optimization techniques

By the time you've finished this book and have had a chance to tackle your own web site, you might want to take the next steps to increasing your site-building knowledge. We'll be recommending next steps where appropriate throughout the book, and suggesting other resources that you might like to check out.

So, this is where the introductory piffle ends and the process of learning begins—learning how to build web sites the *right* way. So step this way, ladies and gentlemen…

What's in this Book?

Chapter 1: *Setting Up Shop*

In this chapter, we'll make sure that you have all the tools you're going to need to build your web site. I'll explain where you can get the right tools—all of them for free! By the end of the chapter, you'll be ready to get cracking on your first web site.

Chapter 2: *Your First Web Pages*

In this chapter, we'll learn what makes a web page. We'll explore XHTML, understand the basic requirements of every web page, and investigate the common elements that you'll see on many web pages. Then, you'll start to create pages yourself. In fact, by the end of this chapter, you'll have the beginnings of your first web site.

Chapter 3: *Adding Some Style*

In this chapter, we'll start to add a bit of polish to the web pages we created in Chapter 2. You'll learn what CSS is, and why it's a good thing, before putting it into action for yourself. As the chapter progresses, you'll see the

project web site start to take shape as we apply background and foreground colors, change the appearance of text, and make web links that have been visited look different from those that haven't.

Chapter 4: ***Shaping Up with CSS***

This chapter builds on the previous chapter's introduction to the color and text-styling abilities of CSS to reveal what CSS can do for border styles and page layouts in general. First, we'll review the full range of border effects that you can apply to elements such as headings and paragraphs. We'll experiment with dotted borders, and big, bold borders, as well as some slightly more subtle effects. In the second half of the chapter, we'll learn how it's possible to use CSS to position the elements of a web page—including blocks of navigation—anywhere on the screen.

Chapter 5: ***Picture this! Using Images on your Web Site***

As the chapter title suggests, this one's all about images. We'll discover the difference between inline images and background images, and look into the issue of making images accessible for blind or visually impaired people (blind people surf too!). We'll also learn how to adjust pictures to suit your web site using the software that we downloaded in Chapter 1. Then we get practical, putting all this knowledge together to create a photo gallery for the project site.

Chapter 6: ***Tables: Tools for Organizing Data***

In this chapter, we'll learn when tables should be used and, perhaps more importantly, when they should *not* be used. Once we've got the basics out of the way, I'll show how you can breathe life into an otherwise dull-looking table—again, using CSS—to make it much more visually appealing.

Chapter 7: ***Forms: Interacting with your Audience***

In Chapter 7, we learn all about forms—what they're used for, what's required to build a form, and what you can do with the data you collect through your form. I'll teach you what the different form elements—such as text inputs, checkboxes, and so on—do, and show you how to use CSS to make a form look more attractive. Finally—and this is something that other books may not explain—I'll show you how you can use a free web service to have the data that's entered into your form emailed to you.

Chapter 8: ***Getting your Web Site Online***

It's all well and good to build a web site for fun, but you need a way for people to see it—that's what this chapter is all about. We'll learn about hosting plans, discuss the pros and cons of using free services, and look at

the tools that you'll need in order to transfer your files from your computer at home to a web server for the world to see.

Chapter 9: ***Adding a Blog to your Web Site***
Blogging's one of the best ways to keep your web site's content fresh and ever-changing. In this chapter, you'll learn what a blog is (that's always a good start), and how you can set one up for yourself. We'll also spend some time making sure it looks consistent with the rest of your web site.

Chapter 10: ***Pimp my Site: Cool Stuff you can Add for Free***
You've heard of the MTV program *Pimp My Ride*, right? No? Well, every week, the guys on this show take an everyday car and transform it—with some well-placed and carefully executed cosmetic touches—into a real head-turner of a vehicle. And that's this chapter's aim for your web site! You'll discover that there are all kinds of tools, plugins, and add-ons that you can build into to your web site to make it even more useful for you and your visitors. Among the tools on offer we'll find site search facilities, statistics programs, and online discussion forums.

Chapter 11: ***Where to Now? What you Can Learn Next***
In this final chapter, we summarize the skills that you've learned in this book, then consider your options for learning more. I'll recommend web sites that I feel can take you to that next level, and books that really should be on your bookshelf (or rather, open on your desk next to your computer!), in an effort to ensure that you continue to learn the good stuff once you've put this book down.

Appendix A: ***XHTML Reference***
This appendix lists all of the XHTML elements that are available for use in your web pages. It's not intended to replace the official W3C reference, but it does provide detailed information about each element and how it can be used. The appendix also includes an example of how each element can be applied within markup, so you should find it to be a handy reference not just as you work through this book, but in the web projects you tackle in future.

The Book's Web Site

Located at http://www.sitepoint.com/books/html1/, the web site supporting this book will give you access to the following facilities:

The Code Archive

As you progress through the text, you'll note a number of references to the code archive. This is a downloadable ZIP archive that contains complete code for all the examples presented in the book. It also includes a copy of the Bubble Under web site, which we use as an example throughout the book.

Updates and Errata

No book is perfect, and I expect that watchful readers will be able to spot at least one or two mistakes before the end of this one. The Errata page, at http://www.sitepoint.com/books/html1/errata.php on the book's web site, will always have the latest information about known typographical and code errors, and necessary updates for new browser releases and versions of web standards.

The SitePoint Forums

If you'd like to communicate with me or anyone else on the SitePoint publishing team about this book, you should join theSitePoint Forums.[3] In fact, you should join that community even if you *don't* want to talk to us, because there are a lot of fun and experienced web designers and developers hanging out there. It's a good way to learn new stuff, get questions answered (unless you really enjoy being on the phone with some company's tech support line for a couple of hours at a time), and just have fun.

The SitePoint Newsletters

In addition to books like this one, SitePoint offers free email newsletters.

The SitePoint Tech Times covers the latest news, product releases, trends, tips, and techniques for all technical aspects of web development. The long-running *SitePoint Tribune* is a biweekly digest of the business and moneymaking aspects of the web. Whether you're a freelance developer looking for tips to score that dream contract, or a marketing major striving to keep abreast of changes to the major search engines, this is the newsletter for you. *The SitePoint Design View* is a monthly compilation of the best in web design. From new CSS layout methods to subtle

[3] http://www.sitepoint.com/forums/

Photoshop techniques, SitePoint's chief designer shares his years of experience in its pages.

Browse the archives or sign up to any of SitePoint's free newsletters at http://www.sitepoint.com/newsletter/.

Your Feedback

If you can't find your answer through the forums, or you wish to contact me for any other reason, the best place to write is `books@sitepoint.com`. We have a well-manned email support system set up to track your inquiries, and if our support staff are unable to answer your question, they send it straight to me. Suggestions for improvement as well as notices of any mistakes you may find are especially welcome.

Acknowledgements

While writing a book sometimes seems like quite a solitary process, the truth is that there are a lot of people who indirectly guide—or have guided—the hands that type the words on these pages. None of this would have been possible had I not been pointed in the direction of influential and persuasive web sites like webmonkey.com, whose CSS tutorial first made me see the light, and individuals such as Jeffrey Zeldman, Molly Holzschlag, and Eric Meyer, whose pioneering work has benefited me (and many others) greatly. However, if I were to list the names of all the people who have inspired me in way, shape, or form in the last few years, this acknowledgments section would end up looking more like an index! You folks know who you are, keep up the good work!

I would like to acknowledge the work undertaken by the Web Standards Project (of which I am also a member, albeit a pretty inactive one for the last couple of years!), and give a little shout-out to my fellow "Britpackers"—wear those Union Jack pants with pride, folks!

Thanks to all those at SitePoint who have helped me in the crafting of this book, particularly to Simon Mackie, my main point of contact and sounding board, and my expert reviewer, Marc Garrett—your comments were always on target. Thanks to Matthew Magain, who tech edited the book—I hope I didn't leave you much to correct or find fault with! And then of course there's Georgina Laidlaw, who ensured that any peculiarly British turns of phrases were removed, despite my best efforts to sneak a few in.

Finally, thanks to Manda for putting up with me when deadlines loomed and I all but shut myself off from civilization to get the chapters in on time. Social life? Oh that! I remember… At those times it seemed like it would never end, but finally we can both see the fruits of my labour. Now, if only it were something she'd actually want to read!

Conventions Used in this Book

You'll notice that we've used certain typographic and layout styles throughout this book to signify different types of information. Look out for the following:

Markup Samples

Any markup—be that HTML or CSS—will be displayed using a fixed-width font like so:

File: **webpage.html (excerpt)**

```
<h1>A perfect summer's day</h1>
<p>It was a lovely day for a walk in the park. The birds were
    singing and the kids were all back at school.</p>
```

Menus

When you need to select something from a menu, it'll be written as File > Save; this means "select the Save option from the File menu."

Tips, Notes, and Warnings

Hey, you!

Tips will give you helpful little pointers.

Ahem, Excuse me…

Notes are useful asides that are related—but not critical—to the topic at hand. Think of them as extra tidbits of information.

Make Sure you Always…

…pay attention to these important points.

Watch Out!

Warnings will highlight any gotchas that are likely to trip you up along the way.

1 Setting Up Shop

Before you dive in and start to build your web site, we need to take a little time to get your computer set up and ready for the work that lies ahead. That's what this chapter is all about: ensuring that you have all the tools you need installed and ready to go.

Tooling Up

If you were to look at the hundreds of computing books for sale in your local bookstore, you could be forgiven for thinking that you'd need to invest in a lot of different programs to build a web site. However, the reality is that most of the tools you need are probably sitting there on your computer, tucked away somewhere you wouldn't think to look for them. And if ever you don't have the tool for the job, there's almost certain to be one or more free programs available that can handle the task.

We've made the assumption that you're already on the Internet—your web site wouldn't be of much use without it! And, though it's not essential, it will also be handy if you have a broadband internet connection. Don't worry if you don't have broadband: it won't affect any of the tasks we'll undertake in this book. It will, however, mean that some of the suggested downloads may take a while to load onto your computer, but you probably knew that already.

Planning, Schmanning

At this point, it might be tempting to look at your motives for building a web site. Do you have a project plan? What objectives do you have for the site?

While you probably have some objectives, and some idea of how long you want to spend creating your site, we're going to gloss over the nitty-gritty of project planning to some extent. This is not to say that project planning isn't an important aspect to consider, but we're going to assume that because you've picked up a book entitled *Build Your Own Web Site The Right Way*, you probably want to just get right into the building part.

As this is your first web site, and will be a fairly simple one, we can overlook some of the more detailed aspects of site planning. Later, once you've learned—and moved beyond—the basics of building a site, you might feel ready to tackle a larger, more technically challenging site. When that time comes, proper planning will be a far more important aspect of the job. But now, let's gear up to build our first, simple site.

The Basic Tools you Need

As I mentioned earlier, many of the tools you'll need to build your first web site are already on your computer. So, what tools *do* you need?

- ❑ The primary—and most basic—tool that you'll need is a **text editor**. You'll use this to write your web pages.
- ❑ Once you've written a web page, you can see how it looks in a **web browser**.
- ❑ Finally, when you're happy with your new web page, you can put it on the Internet using an **FTP client**. Using FTP can be a little complicated, but thankfully you won't need to do it too often. We'll discuss FTP clients in detail in Chapter 8.

You've already got most of these programs on your computer, so let's go find them.

Windows Basic Tools

Your Text Editor: Notepad

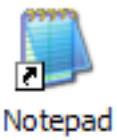
The first tool we'll consider is the text editor. Windows comes with a very simple text editor called Notepad. Many professional web designers who use complicated software packages first started out many years ago using Notepad; indeed, many professionals who have expensive pieces of software that should be time-savers still resort to using Notepad for many tasks. Why? Well, because it's so simple, little can go wrong with it. Bells and whistles are definitely not on the menu here.

You can find Notepad in the Start menu: go to Start > All Programs > Accessories.

Shortcut to Notepad

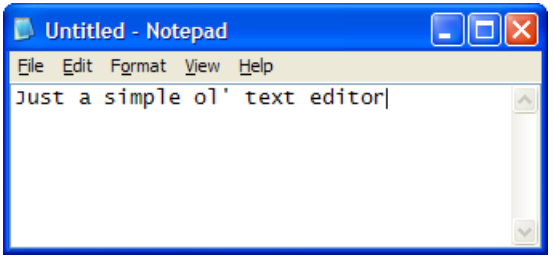
To save yourself navigating to this location each and every time you want to open Notepad, create a shortcut on your desktop. With the Start menu open to display Notepad's location, hold down the **Ctrl** key, click and hold down the mouse button, then drag the Notepad icon to your desktop. When you release the mouse button, a shortcut to the application will appear on your desktop, as in Figure 1.1.

Figure 1.1. Creating a shortcut to Notepad



Notepad is the simplest of simple applications. See Figure 1.2 for a depiction of its fancy interface (that was sarcasm, by the way).

Figure 1.2. Notepad: a contender for the world's ugliest program?



Your Web Browser: Internet Explorer

Once you've created a web page using Notepad, you'll need a way to view the results of your handiwork. You'll remember that, in the preface to this book, we mentioned Internet Explorer. Well, that's your viewer! As Figure 1.3 shows, Internet Explorer sits right there in the Programs folder, and also lurks on your desktop.

Figure 1.3. Internet Explorer: there's no hiding this baby away!

Mac OS X Basic Tools

Like Windows, the Mac operating system (specifically OS X; we won't be looking at previous versions of the Mac OS) has a number of tools that you can use straight out of the box. These tools are virtually equivalent to the Windows programs mentioned above.

Your Text Editor: TextEdit

While Windows has Notepad, the Mac has TextEdit, which can be found in the Applications folder, as Figure 1.4 illustrates.

Figure 1.4. TextEdit comes as part of Mac OS X's default installation

Unlike Notepad, TextEdit works as what we call a "rich text editor" by default, which means we can work with fonts, make text bold and italic, and so on. However, we want to work with TextEdit as a plain text editor, so you'll need to fiddle with some of TextEdit's preferences. Start TextEdit, then select TextEdit > Preferences from the menu to bring up the Preferences screen. Select Plain text within New Document Attributes, then close the Preferences screen. The next time you create a new file in TextEdit, it will be a plain text document.

Your Web Browser: Safari

Internet Explorer is also available for Mac, but was abandoned by Microsoft when Apple began to make its own web browser, Safari, so it's considerably outdated. You can usually find Safari in the dock, but you can also access it through the Applications folder, as Figure 1.5 illustrates.

Figure 1.5. Internet Explorer and Safari are available via Mac's Applications folder

Stick it in the Dock

Just as you can drag shortcuts to programs onto the Windows desktop, you can add programs to the dock in Mac OS X (the dock is the bar of icons at the bottom of your screen). To add a program to the dock, just drag it from the Applications folder onto the dock, and presto! The application is now easily accessible whenever you need it.

Beyond the Basic Tools

You can certainly make a good start using the tools mentioned above. However, once you're dealing with a handful of web pages and other resources, you may want to go beyond these basic tools. We'll show you how to use some slightly more advanced applications later in the book.

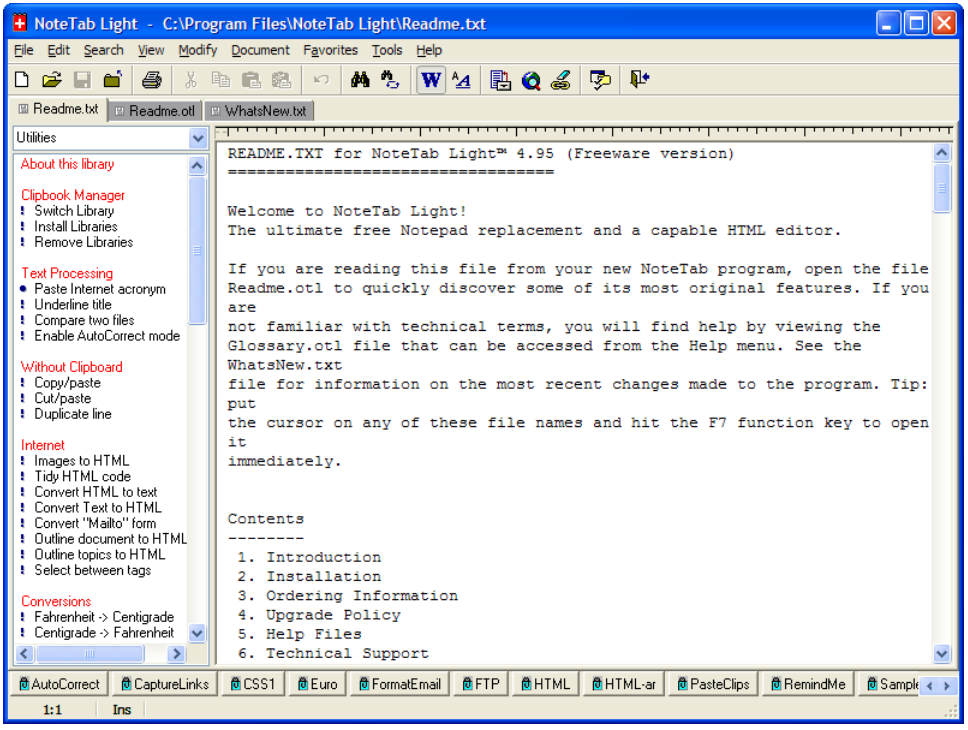
Countless other text editors and web browsers are available for download, and many of them are free. Obviously, we don't have time to describe each and every one of them, so I've settled on a few options that have worked for me in the past, and which you might like to download and have at your disposal.

Windows Tools

NoteTab

NoteTab's tabbed interface lets you have many different files open simultaneously without cluttering up your screen, as Figure 1.6 illustrates. Files that you've opened are remembered even after you close the program and open it again later (very useful when you're working on a batch of files over many days, for instance). You can download the free NoteTab, or its Light version, from http://www.notetab.com/.

Figure 1.6. NoteTab Light's tabbed interface



Firefox

Firefox is a popular alternative to Internet Explorer, and, as we proceed through this book, it will be our browser of choice for a number of reasons. As with NoteTab, Firefox offers a nice tabbed interface that keeps your computer free from window clutter. You can download Firefox from http://www.mozilla.com/firefox/; the browser is depicted in Figure 1.7.

Figure 1.7. Firefox—this critter is worth hunting down

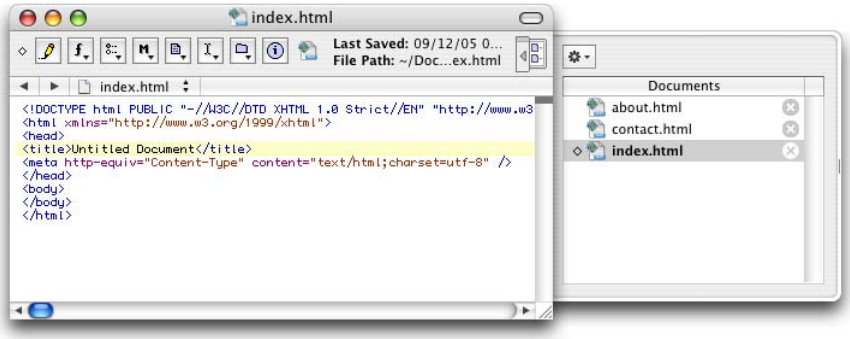
Mac OS X Tools

It is true that there are fewer free programs available for the Mac operating system than there are for Windows. However, there are a few programs that you might like to consider as you move beyond the basics.

TextWrangler

TextWrangler is a free, simple text editor made by BareBones Software. As with NoteTab for Windows, TextWrangler can tidy up your workspace by allowing several text files to be open for editing at the same time (the documents are listed in a pull-out "drawer" to one side of the interface, rather than in tabs). You can download TextWrangler, which is shown in Figure 1.8, from the BareBones Software web site.[1]

Figure 1.8. TextWrangler, a free text editor from BareBones Software



Firefox

Firefox is gaining popularity not just among Windows users, but among Mac users too! A web page viewed in Firefox should display the same regardless of whether the browser is installed on a PC running Windows XP, on a Mac running OS X, or on Linux (an operating system favored by people who really, *really* hate Microsoft products with a vengeance). The predictability of Firefox is a welcome change from the bad old days of endless browser competition, and is one very good reason why we will primarily use Firefox in the examples included in this book.

[1] http://www.barebones.com/products/textwrangler/

Not Just Text, Text, Text

You can build an entire web site using just the tools mentioned above, but it won't be the sexiest site on the Web. The missing element here is images: so far, the programs we've mentioned are used to manipulate plain text or view web pages. If your web site is going to be visually appealing, you'll need to be able to create and manipulate images, either from scratch, using photos you've taken, or using images that you have the legal right to use on your web site.

Unfortunately, when it comes to image editing software, that old saying—"You get what you pay for"—is alive and well. A professional image editing program, like Photoshop or Fireworks, costs hundreds of dollars. While these programs offer some excellent capabilities, we cannot really recommend that you go out and pay for them unless you are *absolutely sure* that they're right for you. If you already have a copy of one of these, or a similar image editing program, by all means use it and experiment with it. Programs like Paint Shop Pro or Photoshop Elements (a cut-down version of Photoshop) are more reasonably priced. However, for the purposes of this book, we'll look only at tools that are free to download, and offer enough functionality to give you an idea of what's possible.

Keep an eye open for free image editors that are included on disks attached to the covers of internet, computing, and design magazines. Software vendors often give away older versions of their software in the hopes that users might be tempted to upgrade to a new version at a later date. Look out for Paint Shop Pro, or any image editor that supports "layers." While we'll keep our image editing fairly simple throughout this book, it's certainly worth keeping an eye open for free (and full-featured) image editing software, as these offers will not always be available.

Taking the Big Boys for a Spin

The most commonly used image editing packages are available for trial download. They are large downloads (hundreds of megabytes) and will probably need to be downloaded overnight, even on a broadband connection.

These trial versions are typically available for 30 days' use; after that time, you can decide whether you want to pay for the full software, or stop using the program. However, those 30 days might just be enough time for you to use the software while you work through this book.

Adobe Photoshop	A trial of the latest version of Photoshop is available for download.[2] If you'd rather try the lighter Photoshop Elements, trial versions are available for Windows[3] and Mac.[4]
Macromedia Fireworks	You can download a trial version of Fireworks from the Macromedia web site.[5]
Paint Shop Pro	Paint Shop Pro is available for Windows only. To download a trial version, visit the Paint Shop Pro site,[6] and click the "Try" link on the right.

Windows Tools

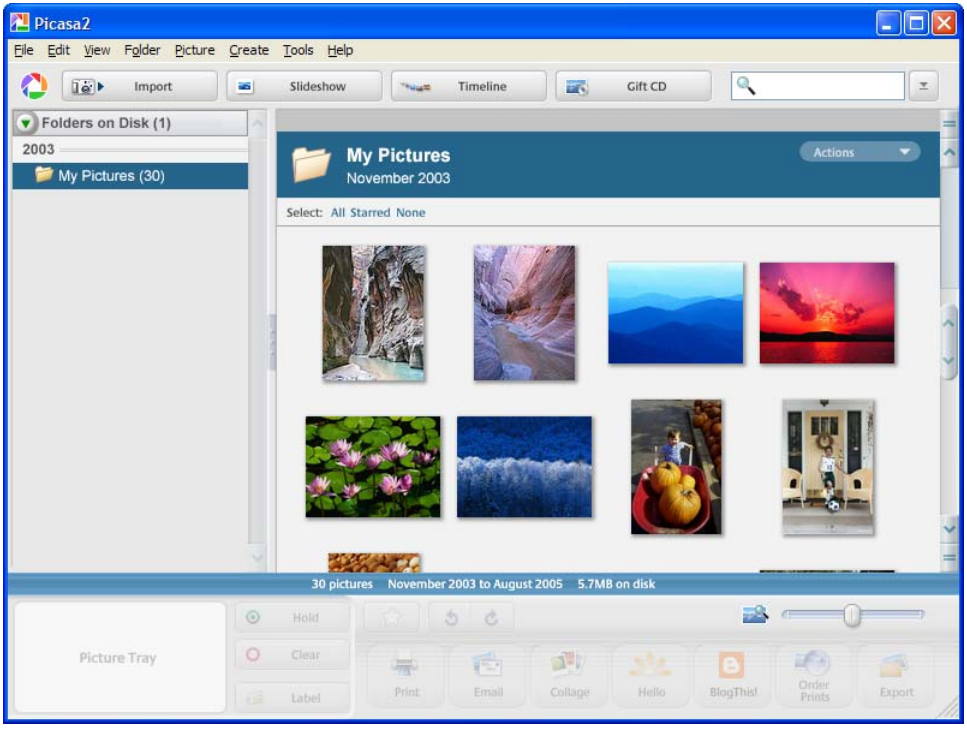
A standard Windows install is not exactly blessed with image editing software (although you might be lucky enough to have bought a PC, scanner, or digital camera that came bundled with some image editing software; scout around in your Start > All Programs menu to see what you can uncover). As a general rule, you'll have to look elsewhere for useful image manipulation software.

Picasa

Though the Windows default image editing and management tools are distinctly lackluster, the good news is that there's an excellent program just begging to be downloaded and used, and it's brought to you by those good people at Google. The program is called Picasa, and it's extremely well equipped to handle most of the tasks that you're likely to encounter as you manage imagery for your web site. Download a copy from the Picasa web site,[7] and soon enough you'll be using this program to crop, rotate, add special effects to, and catalogue the images stored on your computer. Figure 1.9 gives you an idea of the program's interface.

[2] http://www.adobe.com/products/photoshop/tryout.html
[3] http://www.adobe.com/products/photoshopelwin/
[4] http://www.adobe.com/products/photoshopelmac/
[5] http://www.macromedia.com/go/tryfireworks/
[6] http://www.corel.com/paintshoppro/
[7] http://www.google.com/picasa/

Figure 1.9. Picasa, Google's free, fully-featured photo and image editing and management tool



Mac OS X Tools

The Mac has a reputation for being favored by designers and creative types, and the platform makes many tools available to the budding artist. However, they usually come at a price, and often that price is higher than those of the Windows equivalents! So, what free software can we use on the Mac, assuming that we want something more permanent than a 30-day trial version of Photoshop or Fireworks?

GraphicConverter

GraphicConverter has much greater capabilities than its name suggests. It's bundled with most newer Macs, and is also available for download[8] (you'll be encouraged to pay a modest registration fee for the software, but you can try it out for free). Sure, this is primarily a tool for converting graphic files, but it can also be used for simple editing tasks. Using GraphicConverter, which is illustrated in Figure 1.10, you'll be able to crop, resize, rotate, and add text to any image.

Figure 1.10. GraphicConverter does a lot more than simply convert graphics!

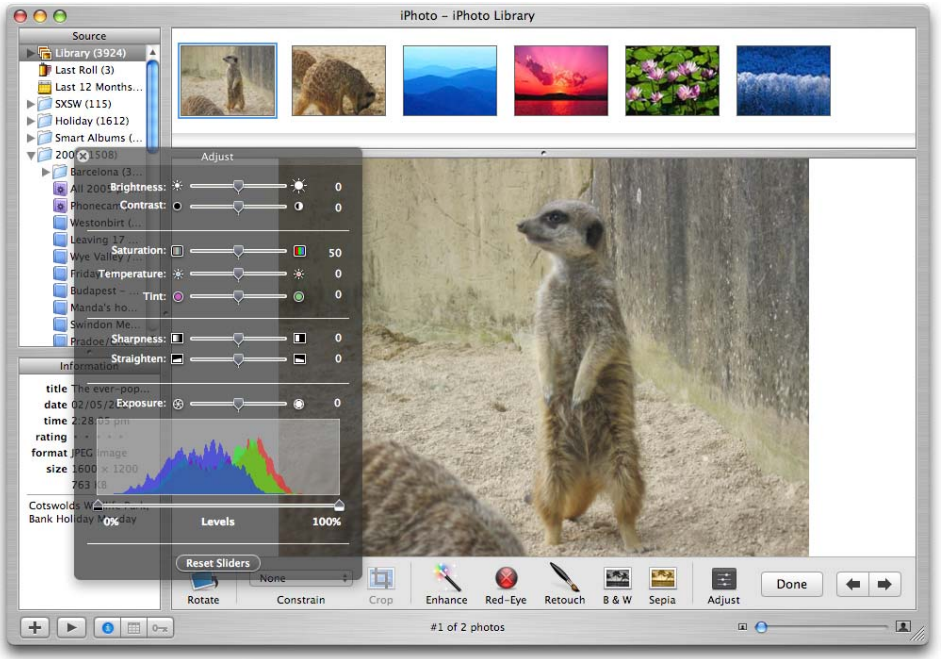
iPhoto

Also included with Mac OS X is a program that probably needs no introduction to the experienced Mac user: iPhoto. This excellent program is not intended as a fully-featured image editor; it's really designed for managing and viewing large numbers of photos stored on a computer. It's great for organizing photo albums, but with recent updates, iPhoto has been given some editing facilities that have promoted it beyond a mere cataloguing tool. Unfortunately, the latest version of iPhoto is not available as a free download, and must be purchased as part of Apple's iLife suite of programs.

[8] http://www.graphicconverter.net/

iPhoto, pictured in Figure 1.11, can be found in the Applications folder, or in the dock.

Figure 1.11. The iPhoto 5 image adjustment tools



Creating a Spot for your Web Site

So far, we've looked at some of the tools that you'll need to create your web site. We've looked at programs that are readily available, and where you can find them on your computer. And for cases in which the free tools that came with your computer are not up to the job, we've suggested other programs that you can download and use. The next task we must tick off our to-do list before we go any further is creating a space for your web site on the hard drive.

Windows

The easiest and most logical place to keep your web site files is in a dedicated folder within the My Documents folder. There should be a My Documents folder on your desktop, but don't worry if it's not there: it's easy to get it to appear

there (see the tip below for details). Double-click on My Documents, then create a new folder called Web by selecting File > New > Folder.

Displaying the "My Documents" Folder

Lost your "My Documents" folder? It's easily done: in an effort to clean up your desktop, you may have removed the icon by accident. You can return the folder to your desktop as follows:

- From the Start Menu, select Control Panel.
- Select Appearance and Themes, then select Display (if you don't see Appearance and Themes, but do see Display, double click on that). A Display Properties window will appear.
- Select the Desktop tab along the top, then click the Customize Desktop... button at the bottom.
- You'll be presented with the Desktop Items dialog, which will include a set of Desktop icons checkboxes that allow you to select the icons you'd like to see on your desktop. Check the My Documents checkbox, then click OK on both open dialogs.
- Your My Documents folder should now be back on the desktop, as shown in Figure 1.12.

Figure 1.12. The My Documents folder displaying on the desktop in XP

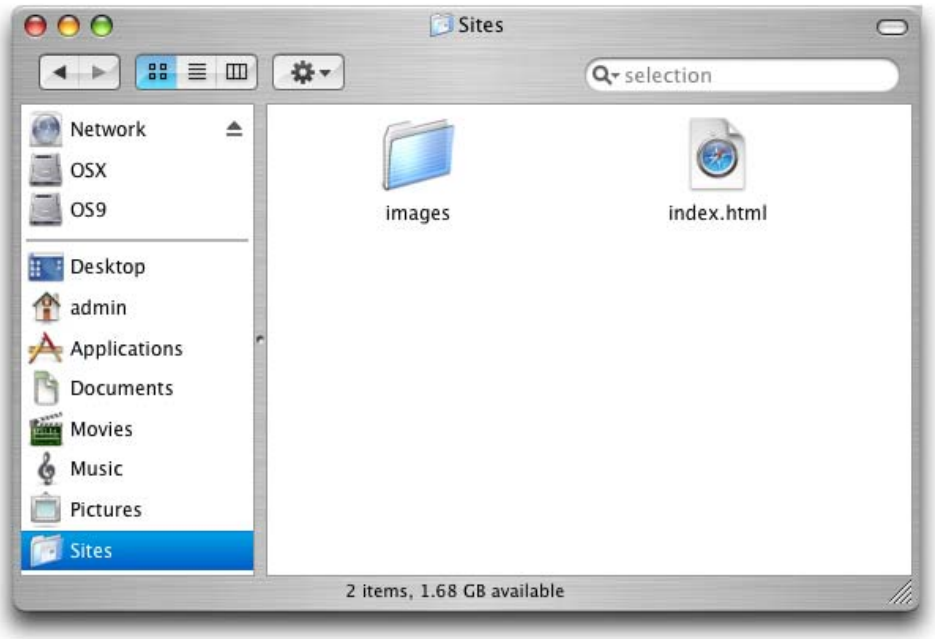
Mac OS X

In Mac OS X, there's already a handy place for you to store your web site files: the Sites folder shown in Figure 1.13. Open your home directory (from Finder, select Go > Home), and there it is!

Figure 1.13. The Sites folder displaying in the Mac OS X home directory

It's easy to add the Sites folder to your sidebar (which you can see in Figure 1.14) for quick access: just drag the folder to the sidebar in the same way you add items to the dock.

Figure 1.14. The Sites folder appearing in the sidebar



Getting Help

Books may be a wonderful way to learn: you're sitting there with a computer running, a cup of coffee keeping your mind ticking over (you've got one there, right? A coffee, that is, not a mind), and a bookmark signifying your progress to date. Great. But what if you don't understand something in the book? What do you do next? Shouting at the book won't help (well, it may be therapeutic, but you won't get a response)!

Hopefully, you won't find yourself asking too many questions as you work through this book, but if you're the curious type—or a quick learner—you might want to go beyond what we're going to teach you here.

Whether you're getting stuck on something, or you want to learn more, your first stop should be the SitePoint Forums.[9] It will only take a few moments to register, and once you've done so, you can log in and ask questions in a range of different

[9] http://www.sitepoint.com/forums/

forums. Whether you have questions about writing your web site, you need marketing tips, or you're facing a few tricky graphic design issues, the hundreds of experts who contribute to and moderate these pages every day will be happy to help out.

Register at SitePoint's forums today; then, when we recommend further reading or research, you'll be good to go. Oh, and did we mention that all this friendly, helpful advice is 100% free of charge? I thought that might encourage you!

Summary

Believe it or not, we've now got everything we need to build our own web site—and all without spending a cent! Not only do we have the basic tools—our text editor (Notepad or TextEdit) and our web browser (Internet Explorer or Safari)—but we've also looked at some alternatives to these.

We've reviewed some simple and freely available image editing programs that can help us spruce up our sites: Picasa for Windows, and GraphicConverter and iPhoto for Mac. Finally, we mentioned some more capable—and more expensive—options, such as Photoshop and Paint Shop Pro.

Now we've got the tools, let's learn how to use them!

2 Your First Web Pages

A wise man once said that a journey of a thousand miles begins with a single step. In this chapter, you'll take that first metaphorical step on your journey towards web site enlightenment: you'll create your first web page. By the end of the chapter, you'll have duplicated that first page to form the beginnings of a multi-page web site.

Nice to Meet you, XHTML

In the preface to this book, we touched briefly on what XHTML is. In this chapter, we'll learn the basics of XHTML, periodically previewing our progress in a browser, and steadily building up our knowledge of various XHTML **elements**. Elements are the basic building blocks of XHTML; they tell the web browser what a particular item in the page is: a paragraph, a heading, a quotation, and so on. Elements contain all the information that the browser requires, as we'll soon see.

Anatomy of a Web Page

In the preface, we said that learning XHTML was like taking a driving lesson. To take that analogy a step further, you can imagine a web page as being the car in which you're learning to drive. There are some things that are essential to the process of driving; others are mere fashion items. For instance, the interior might

have two seats, or it may have four or six; it might have black leather trim, fluffy dice, or faux leopard-skin door panels. But, while you may dearly love those door panels, they're not essential to driving.

You can't drive the car unless you have a steering wheel to hold onto, four wheels (at a minimum!), and a place to sit. The car must also have some kind of chassis to which the bodywork can be bolted. An engine is needed to power the car, as is bodywork to which your (nonessential, but spiffy) trim can be attached. Anything less, and all you have is a collection of attractive—but useless!—spare parts.

Like the car, your web page also needs to have a chassis: a basic structure upon which everything else can be built. But what does this hypothetical "chassis" look like? The best way to find out is to get down on our hands and knees (again, figuratively speaking) and take a close look under the hood.

Viewing the Source

One of the great things about learning to build web pages is that you and I have the ability to view the source code of other people's web pages. You can learn a lot by simply taking a peek at how someone else's web page was built … but how do you do it?

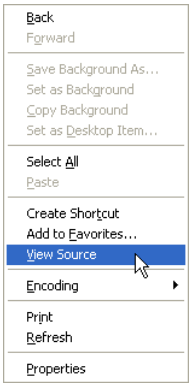
Although every browser uses slightly different terminology, the variations in the ways different browsers let us view web page code are so small that the process doesn't need to be spelled out for every browser. Here's the technique you'd use to view a web page's source in IE:

- Bring up a page in your browser. (I'd suggest the Web Standards Project's homepage.[1] The Web Standards Project (WaSP) is a group that promotes the benefits of building your web site correctly, so you can be pretty sure they've got it right!)

- Position your cursor somewhere on the page (other than over an image), and right-click (Ctrl-click on a Mac). You should be presented with the context menu shown in Figure 2.1.

[1] http://webstandards.org/

Figure 2.1. Selecting the View Source command after right-clicking on a web page



- Select View Source, and a new window will appear, displaying all of the page's underlying markup.

At this point, we're not going to analyze the markup that you're looking at, but this is one of those tricks that's really useful to know from the beginning. A note of warning, though: most web pages don't use best-practice techniques, so avoid looking at a page's source unless the web site in question is mentioned in this book as being a good example.

Basic Requirements of a Web Page

As we've already discussed, in any web page, there are some basic must-have items (all of which you could pick out if you scanned through the markup that appeared when you tried to "view source" a moment ago):

- a doctype
- an `<html>` tag
- a `<head>` tag
- a `<title>` tag
- a `<body>` tag

These requirements make up the basic skeleton of a web page. It's the chassis of your car with some unpainted bodywork, but no wheels or seats. A car enthusiast would call it a "project"—a solid foundation that needs a little extra work to turn it in to something usable. The same goes for a web page. Here's what these requirements look like when they're combined in a basic web page:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Untitled Document</title>
    <meta http-equiv="Content-Type"
        content="text/html; charset=utf-8" />
  </head>
  <body>
  </body>
</html>
```

The markup above is the most basic web page you'll see here. It contains practically no content of any value (at least, as far as someone who looks at it in a browser is concerned), but it's crucial that you understand what this markup means. Let's delve a little deeper.

The Doctype

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

This is known as the **doctype** (short for Document Type Declaration). It must *absolutely* be the first item on a web page, appearing even before any spacing or carriage returns.

Have you ever taken a document you wrote in Microsoft Word 2003 on one computer, and tried to open it on another computer that ran Word 97? Frustratingly, without some pre-emptive massaging when the file is saved in the first place, this just doesn't work. It fails because Word 2003 includes features that Bill Gates and his team hadn't even dreamed of in 1997, and Microsoft needed to create a new version of its file format to cater to these new features. Just as Microsoft has many different versions of Word, so too are there different versions of HTML, the latest of which is XHTML. Mercifully, the different versions of HTML have been designed so that this language doesn't suffer the same incompatibility gremlins as Word, but it's still important to identify the version of HTML that you're using. This is where the doctype comes in. The doctype's job

is to specify which version of HTML the browser should expect to see. The browser uses this information to decide how it should render items on the screen.

The doctype above states that we're using XHTML 1.0 Strict, and includes a **URL** to which the browser can refer: this URL points to the **W3C**'s specification for XHTML 1.0 Strict. Got all that? Okay: jargon break! There are too many abbreviations for this paragraph!

Jargon Busting 101

URL

URL stands for Uniform Resource Locator. It's what some (admittedly more geeky) people refer to when they talk about a web site's address. URL is definitely a useful term to learn, though, because it's becoming more and more common.

W3C

W3C is an abbreviation of the name World Wide Web Consortium, a group of smart people spread across the globe who, collectively, come up with proposals for the ways in which computing and markup languages that are used on the Web should be written. The W3C defines the rules, suggests usage, then publishes the agreed documentation for reference by interested parties, be they web site creators like your good self (once you're done with this book, that is), or software developers who are building the programs that need to understand these languages (such as browsers or authoring software).

The W3C documents are the starting point, and indeed everything in this book is based on the original documents. But, trust me: you don't want to look at any W3C documents for a long time yet. They're just plain scary for us mere mortals without Computer Science degrees. Just stick with this book for time being—I'll guide you through!

The `html` Element

So, the doctype has told the browser to expect a certain version of HTML. What comes next? Some HTML!

An XHTML document is built using elements. Remember, elements are the bricks that create the structures that hold a web page together. But what exactly *is* an element? What does an element look like, and what is its purpose?

- An XHTML element starts and ends with **tags**—the **opening tag** and the **closing tag**.[2]
- A tag consists of an opening angled bracket (<), some text, and a closing bracket (>).
- Inside a tag, there is a **tag name**; there may also be one or more **attributes**.

Let's take a look at the first element in the page: the `html` element. Figure 2.2 shows what we have.

Figure 2.2. Constituents of a typical XHTML element

Figure 2.2 depicts the opening tag, which marks the start of the element:

```
<html xmlns="http://www.w3.org/1999/xhtml">
```

Below it, we see the closing tag, which marks its end (and occurs right at the end of the document):

```
</html>
```

Here's that line again, with the tag name in bold:

```
<html xmlns="http://www.w3.org/1999/xhtml">
```

[2]Like any good rule, there are exceptions to this: empty elements, such as `meta`, use special empty tags. We'll take a look at empty tags soon.

And there is one **attribute** in the opening tag:

```
<html xmlns="http://www.w3.org/1999/xhtml">
```

What's an Attribute?

HTML elements can have a range of different **attributes**; the available attributes vary depending on which element you're dealing with. Each attribute is made up of a **name** and a **value**, and these are always written as *name*="*value*". Some attributes are optional, while others are compulsory, but together they give the browser important information that the element wouldn't offer otherwise. For example, the image element (which we'll learn about soon) has a compulsory "image source" attribute, the value of which gives the filename of the image. Attributes appear in the opening tag of any given element. We'll see more attributes crop up as we work our way through this project, and, at least initially, I'll be making sure to point them out, so that you're familiar with them.

Back to the purpose of the `html` element. This is the outermost "container" of our web page; everything else is kept within that outer container—there are no exceptions! Let's peel off that outer layer and take a peek at the contents inside.

There are two major sections inside the `html` element: the head and the body. It's not going to be difficult to remember the order in which those items should appear, unless you happen to enjoy doing headstands.

The `head` Element

The `head` element contains information *about* the page, but no information that will be displayed on the page itself. For example, it contains the `title` element, which tells the browser what to display in its title bar:

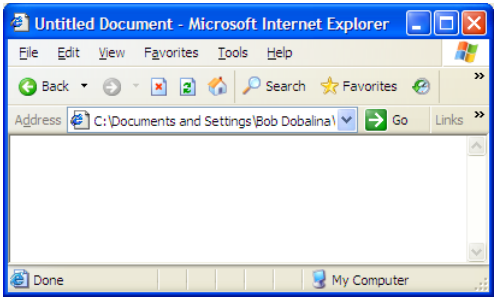
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Untitled Document</title>
    <meta http-equiv="Content-Type"
        content="text/html; charset=utf-8" />
  </head>
  <body>
  </body>
</html>
```

The `title` Element

The opening `<title>` and closing `</title>` tags are wrapped around the words "Untitled Document" in the markup above. Note that the `<title>` signifies the start, while the closing `</title>` signifies the end of the title. That's how closing tags work: they have forward slashes just after the angle bracket.

The "Untitled Document" title is typical of what HTML authoring software provides as a starting point when you choose to create a new web page; it's up to you to change those words. However, if you're creating a web page from scratch in a text editor (like Notepad), you will, I hope, remember to type something a little more useful. But just what is this `title`? Time for a screenshot: check out Figure 2.3.

Figure 2.3. "Untitled Document"—not a very helpful title



It really would pay dividends to put something useful in there, and not just for the sake of those people who visit our web page. The content of the `title` element is also used for a number of other purposes:

- It's the name that appears in the Windows Taskbar for any open document, as shown in Figure 2.4. It also appears in the dock on a Mac, as Figure 2.5 illustrates. When you have a few windows open, you'll appreciate those people who have made an effort to enter a descriptive `title`!

Figure 2.4. The `title` appearing in the Windows Taskbar

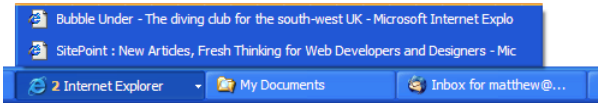
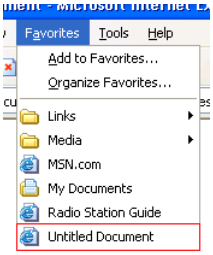


Figure 2.5. The `title` displaying in the Mac dock

- If users decide to add the page to their bookmarks (or favorites), the `title` will be used to name the bookmark, as Figure 2.6 illustrates.

Figure 2.6. An untitled document saved to IE's favorites



- Your `title` element is used heavily by search engines to ascertain what your page contains, and what information about it should be displayed in the search results. Just for fun, and to see how many people forget to type in a useful `title`, try searching for the phrase "Untitled Document" in the search engine of your choice.

`meta` Elements

Inside the `head` element in our simple example, we can see a `meta` element, which is shown in bold below:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Untitled Document</title>
    <meta http-equiv="Content-Type"
        content="text/html; charset=utf-8" />
  </head>
```

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

`meta` elements can be used in a web page for many different reasons. Some are used to provide additional information that's not displayed on-screen to the browser or to search engines; for instance, the name of the page's author, or a copyright notice, might be included in `meta` elements. In the example above, the `meta` tag tells the browser which **character set** to use (specifically, UTF-8, which includes the characters needed for web pages in just about any language).

There are many different uses for `meta` elements, but most of them will make no discernible difference to the way your page looks, and as such, won't be of much interest to you.

Self-closing Elements

The `meta` element is an example of a **self-closing element** (or an **empty element**). Unlike `title`, the `meta` element needn't contain anything, so we could write it as follows:

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

XHTML contains a number of empty elements, and the boffins who put together XHTML decided that writing all those closing tags would get annoying pretty quickly, so they decided to use self-closing tags: tags that end with `/>`. So our `meta` example becomes:

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

The Memory Game: Remembering Difficult Markup

If you're thinking that the doctype and `meta` elements are difficult to remember, and you're wondering how on earth people commit them to memory, don't worry: most people don't! Even the most hardened and world-weary coders would have difficulty remembering these elements exactly, so most do the same thing—they copy from a source they know to be correct (most likely from their last project or piece of work). You'll probably do the same as you work with project files for this book.

Fully-fledged web development programs, such as Dreamweaver, will normally take care of these difficult parts of coding.

Other head Elements

Other items, such as CSS markup and JavaScript code, can appear in the `head`element, but we'll discuss these as we need them.

The body Element

Finally, we get to the place where it all happens! The `body` element of the page contains almost everything that you see on the screen, including headings, paragraphs, images, any navigation that's required, and footers that sit at the bottom of the web page.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Untitled Document</title>
    <meta http-equiv="Content-Type"
        content="text/html; charset=utf-8" />
  </head>
  <body>
  </body>
</html>
```

The Most Basic Web Page in the World

Actually, that heading's a bit of a misnomer: we've already showed you the most basic page (the one without any content). However, to start to appreciate how everything fits together, you really need to see a simple page with some actual content on it. Let's have a go at it, shall we?

Open your text editor and type the following into a new, empty document (or grab the file from the code archive if you don't feel like typing it out—I understand completely!).

File: **basic.html**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>The Most Basic Web Page in the World</title>
    <meta http-equiv="Content-Type"
```

```
        content="text/html; charset=utf-8" />
  </head>
  <body>
    <h1>The Most Basic Web Page in the World</h1>
    <p>This is a very simple web page to get you started.
        Hopefully you will get to see how the markup that drives
        the page relates to the end result that you can see on
        screen.</p>
    <p>This is another paragraph, by the way. Just to show how it
        works.</p>
  </body>
</html>
```

Once you've typed it out, save it as `basic.html`.

If you're using Notepad, select File > Save As... from the menu and find the Web folder you created inside My Documents. Enter the filename as **`index.html`**, select UTF-8 from the Encoding drop-down list, and click Save.

If you're using TextEdit on a Mac, first make sure that you're in plain text mode, then select File > Save As... from the menu. Find the Sites folder, enter **`index.html`**, select Unicode (UTF-8) from the Plain Text Encoding drop-down list, and click Save. TextEdit will warn you that you're saving a plain text file with an extension other than `.txt`, and offer to append `.txt` to the end of your filename. We want to save this file with an `.html` extension, so click the Don't Append button, and your file will be saved.

The Importance of UTF-8

If you neglect to select UTF-8 when saving your files, it's likely that you won't notice much of a difference. However, when someone else comes along to view your web site (say, a Korean friend of yours), they'll probably end up with a screen of garbage. Why? Because their computer is set up to read Korean text, and yours is set up to create English text. UTF-8 can handle just about any language there is (including some quite obscure ones!) and most computers can read it, so UTF-8 is always a safe bet.

Next, using Windows Explorer or Finder, locate the file that you just saved, and double-click to open it in your browser. Figure 2.7 shows how the page displays.

Figure 2.7. Displaying a basic page

Analyzing the Web Page

We've introduced two new elements to our simple page: a heading element, and a couple of paragraph elements, denoted by the `<h1>` tag and `<p>` tags, respectively.

Figure 2.8. Comparing the source markup with the view presented in the browser

Do you see how the markup you've typed out relates to what you can see in the browser? Figure 2.8 shows a direct comparison of the document displays.

The opening <h1> and closing </h1> tags are wrapped around the words "The Most Basic Web Page in the World," making that the main heading for the page. In the same way, the p elements contain the text in the two paragraphs.

There's another point to note: the tags are all lowercase. All of our attribute names will be in lowercase, too. Many older HTML documents include tags and attributes in uppercase, but this isn't allowed in XHTML.

Headings and Document Hierarchy

In the example above, we use an h1 element to show a major heading. If we wanted to include a subheading beneath this heading, we would use the h2 element. There are no prizes for guessing that a subheading under an h2 would use an h3 element, and so on, until we get to h6. The lower the heading level, the lesser its importance and, as a rule, the smaller (or less prominent) the font.

With headings, an important (and commonsense) practice is to ensure that they do not jump out of sequence. In other words, you should start from level one, and work your way down through the levels in numerical order. You can jump back up from a lower-level heading to a higher one, provided that the content under the higher-level heading to which you've jumped does not refer to concepts that are addressed under the lower-level heading. It may be useful to visualize your headings as a list:

- First Major Heading
  - First Subheading
  - Second Subheading
    - A Sub-subheading
- Another Major Heading
  - Another Subheading

Here's the XHTML view of the example shown above:

```
<h1>First Major Heading</h1>
<h2>First Subheading</h2>
<h2>Second Subheading</h2>
<h3>A Sub-subheading</h3>
```

```
<h1>Another Major Heading</h1>
<h2>Another Subheading</h2>
```

Paragraphs

Of course, no one wants to read a document that contains only headings—you need to put some text in there. The element we use to deal with blocks of text is the `p` element. It's not difficult to remember: `p` is for paragraph! That's just as well, because you'll almost certainly find yourself using this element more than any other. And that's the beauty of XHTML: most elements that you use frequently are either very obvious, or easy to remember once you're introduced to them.

For People who Love Lists

Let's imagine that you want a list on your web page. To include an **ordered list** of items, we use the `ol` element. An **unordered list**—called "bullet points" by the average person—makes use of the `ul` element. In both types of list, individual points or list items are specified using the `li` element. So we use `ol` for an ordered list, `ul` for an unordered list, and `li` for a list item. Easy!

To see this markup in action, type the following into a new text document, save it as `lists.html`, and view it in the browser by double-clicking on the saved file's icon.

File: **lists.html**

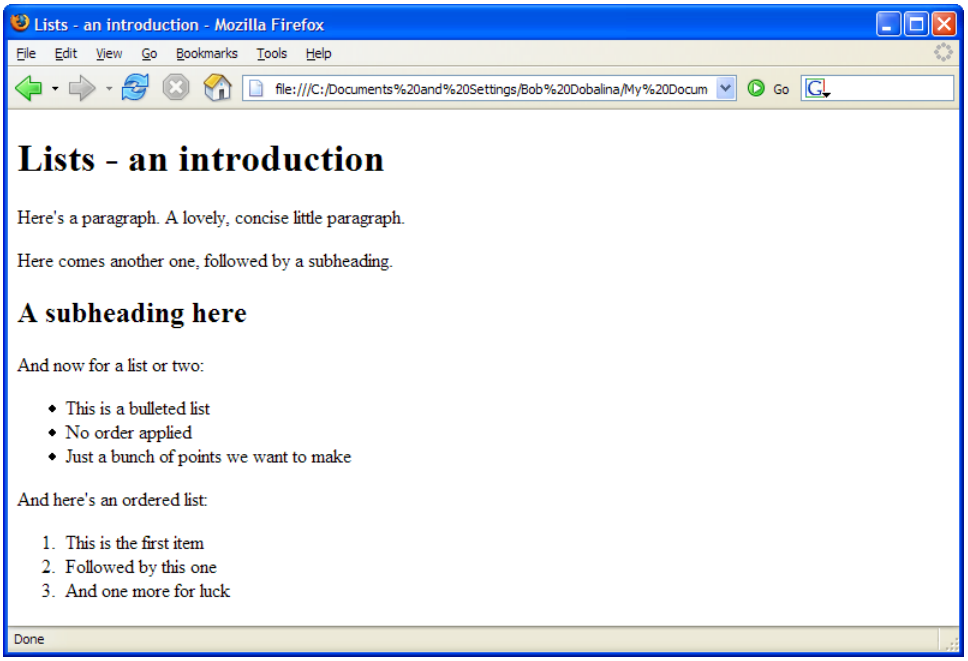
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Lists - an introduction</title>
    <meta http-equiv="Content-Type"
        content="text/html; charset=utf-8" />
  </head>
  <body>
    <h1>Lists - an introduction </h1>
    <p>Here's a paragraph. A lovely, concise little paragraph.</p>
    <p>Here comes another one, followed by a subheading.</p>
    <h2>A subheading here</h2>
    <p>And now for a list or two:</p>
    <ul>
      <li>This is a bulleted list</li>
      <li>No order applied</li>
```

```
      <li>Just a bunch of points we want to make</li>
    </ul>
    <p>And here's an ordered list:</p>
    <ol>
      <li>This is the first item</li>
      <li>Followed by this one</li>
      <li>And one more for luck</li>
    </ol>
  </body>
</html>
```

How does it look to you? Did you type it all out? Remember, if it seems like a lot of hassle to type out the examples, you can find all the markup in the code archive, as I explained in the preface. However, bear in mind that simply copying and pasting markup, then saving and running it, doesn't really give you a feel for what's happening—it really will pay to learn by doing. Even if you make mistakes, it's still a better way to learn (you'll be happy when you can spot your own errors and fix them for yourself). When displayed in a browser, the above markup should look like the page shown in Figure 2.9.

There are many, many different elements that you can use on your web page, and we'll meet more of them as our web site development progresses. As well as the more obvious elements that you'll come across, there are others that are not immediately obvious: what would you use `div`, `span`, or `a` elements for? Any guesses? Don't worry, all will be revealed in good time.

Figure 2.9. Using unordered and ordered lists to organize information



Commenting your Web Pages

Back in the garage, you're doing a little work on your project car and, as you prepare to replace the existing tires with a new set of low-profile whitewalls, you notice that your hubcaps aren't bolted on: you'd stuck them to the car with super glue. There must have been a good reason for doing that, but you can't remember what it was. The trouble is, if you had a reason to attach the hubcaps that way before, surely you should do it the same way again. Wouldn't it be great if you'd left yourself a note when you first did it, explaining why you used super glue instead of bolts? Then again, your car wouldn't look very nice with notes stuck all over it. What a quandary!

When you're creating a web site, you may find yourself in a similar situation. You might build a site, then not touch it again for six months. Then, when you revisit the work, you might find yourself going through the all-too-familiar head-scratching routine. Fortunately, there *is* a solution!

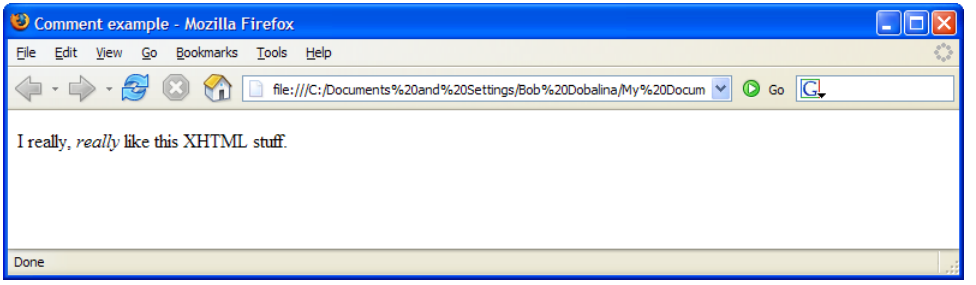
XHTML—like most programming and markup languages—allows you to use **comments**. Comments are perfect for making notes about something you've done and, though they're included within your code, comments do not affect the on-screen display. Here's an example of a comment:

File: **comments.html**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Comment example</title>
    <meta http-equiv="Content-Type"
        content="text/html; charset=utf-8" />
  </head>
  <body>
    <p>I really, <em>really</em> like this XHTML stuff.</p>
    <!-- Added emphasis using the em element. Handy one, that! -->
  </body>
</html>
```

Figure 2.10 shows the page viewed on-screen.

Figure 2.10. The comment remains hidden in the on-screen display



Comments must start with `<!--`, after which you're free to type whatever you like as a "note to self." Well, you're free to type *almost* anything: you cannot type double dashes. Why not? Because that's a signal that the comment is about to end—the `-->` part.

Oh, and did you spot how we snuck another new element in there? The emphasis element, denoted with the `<em>` and `</em>` tags, is used wherever … well, do I *really* need to tell you? Actually, that last question was kind of rhetorical. It was there to make a point: did you notice that the word "really" appeared in italics?

Read that part to yourself now, and listen to the way it "sounds" in your head. Now you know when to use the `em` element!

Using Comments to Hide Markup from Browsers Temporarily

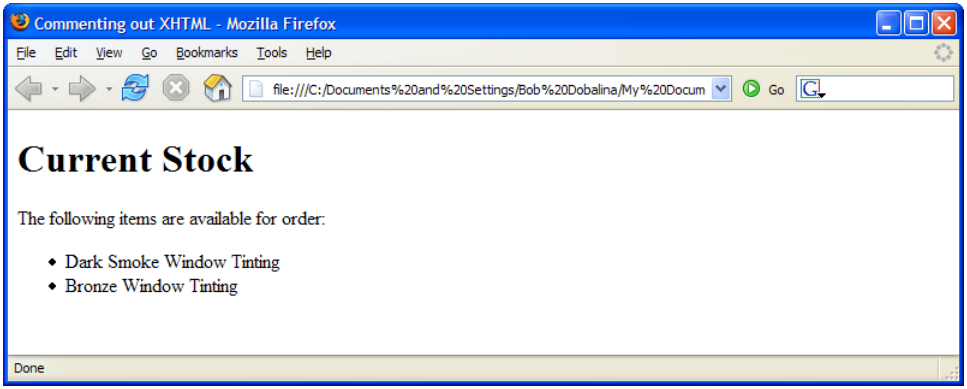
There is no limit to the amount of information you can put into a comment, and this is why comments are often used to hide a section of a web page temporarily. Commenting may be preferable to deleting content, particularly if you want to put that information back into the web page at a later date (if it's in a comment, you won't have to re-type it). This is often called "commenting out" markup. Here's an example:

File: **commentout.html**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Commenting out XHTML</title>
    <meta http-equiv="Content-Type"
        content="text/html; charset=utf-8" />
  </head>
  <body>
    <h1>Current Stock</h1>
    <p>The following items are available for order:</p>
    <ul>
      <li>Dark Smoke Window Tinting</li>
      <li>Bronze Window Tinting</li>
      <!-- <li>Spray mount</li>
      <li>Craft knife (pack of 5)</li> -->
    </ul>
  </body>
</html>
```

Figure 2.11 shows how the page displays in Firefox.

Figure 2.11. The final, commented list items are not displayed



Remember, you write a comment like this: `<!--Your comment here followed by the comment closer, two dashes and a right-angled bracket-->`.

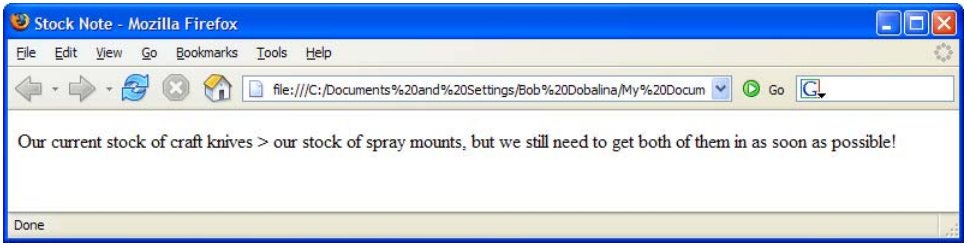
Symbols

Occasionally, you may need to include the greater-than (>) or less-than (<) symbols in the text of your web pages. The problem is that these symbols are also used to denote tags in XHTML. So, what can we do? Thankfully, we can use special little codes called **entities** in our text instead of these symbols. The entity for the greater-than symbol is `>`; we can substitute it for the greater-than symbol in our text, as shown in the following simple example. The result of this markup is shown in Figure 2.12.

File: **entity.html**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Stock Note</title>
    <meta http-equiv="Content-Type"
        content="text/html; charset=utf-8" />
  </head>
  <body>
    <p>Our current stock of craft knives &gt; our stock of spray
        mounts, but we still need to get both of them in as soon
        as possible!</p>
  </body>
</html>
```

Figure 2.12. The `>` entity is displayed as > in the browser



Many different entities are available for a wide range of symbols, most of which don't appear on your keyboard. They all start with an ampersand (`&`) and end with a semicolon. Some of the most common are shown in Table 2.1.

Table 2.1. Some common entities

Entity	Symbol
`>`	>
`<`	<
`&`	&
`&pound;`	£
`&copy;`	©
`&trade;`	™

Diving into our Web Site

So far, we've looked at some very basic web pages as a way to ease you into the process of writing your own XHTML markup. Perhaps you've typed them up and tried them out, or maybe you've pulled the pages from the code archive and run them in your browser. Perhaps you've even tried experimenting for yourself—it's good to have a play around. None of the examples shown so far are keepers, though. You won't need to use any of these pages, but you will be using the ideas that we've introduced in them as you start to develop the fictitious project we'll complete in the course of this book: a web site for a local diving club.

The diving club comprises a group of local enthusiasts, and the web site will provide a way for club members to:

- ❑ share photos from previous dive trips
- ❑ stay informed about upcoming dive trips
- ❑ provide information about ad-hoc meet-ups
- ❑ read other members' dive reports and write-ups
- ❑ announce club news

The site also has the following goals:

- ❑ to help attract new members
- ❑ to provide links to other diving-related web sites
- ❑ to provide a convenient way to search for general diving-related information

The site's audience may not be enormous, but the regular visitors and club members are very keen to be involved. It's a fun site that people will want to come back to again and again, and it's a good project to work on. But it doesn't exist yet! You're going to start building it right now. Let's start with our first page: the site's home page.

The Homepage: the Starting Point for all Web Sites

At the very beginning of this chapter, we looked at a basic web page with nothing in it (the car chassis, with no bodywork or interior, remember?). You saved the file as `basic.html`. Open that file in your text editor now, and strip out the following:

- ❑ the text contained within the opening `<title>` and closing `</title>` tags
- ❑ all the content between the opening `<body>` and closing `</body>` tags

Save the file as `index.html`.

Here's the markup you should have in front of you now:

File: **index.html**

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

Let's start building this web site, shall we?

Setting a Title

Remembering what we've learned so far, let's make a few changes to this document. Have a go at the following:

- ❑ Change the title of the page to read "Bubble Under – The diving club for the south-west UK."
- ❑ Add a heading to the page—a level one heading (hint, hint)—that reads "BubbleUnder.com."
- ❑ Immediately after the heading, add a paragraph that reads, "Diving club for the south-west UK — let's make a splash!" (This is your basic, marketing-type tag line, folks!)

Once you make these changes, your markup should look something like this (the changes are shown in bold):

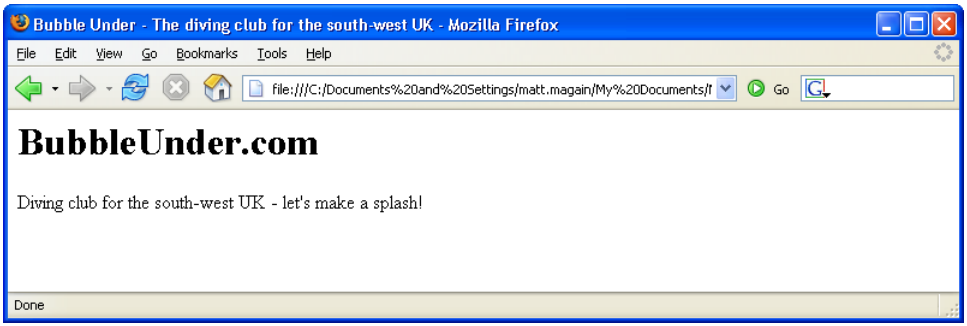
File: **index.html**

```
<!DOCTYPE html "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Bubble Under - The diving club for the south-west
        UK</title>
    <meta http-equiv="Content-Type"
        content="text/html; charset=utf-8" />
  </head>
  <body>
```

```
    <h1>BubbleUnder.com</h1>
    <p>Diving club for the south-west UK - let's make a
        splash!</p>
  </body>
</html>
```

Save the page, then double-click on the file to open it in your chosen browser. Figure 2.13 shows what it should look like.

Figure 2.13. Displaying our work on the homepage



Welcoming New Visitors

Now, let's expand upon our tag line a little. We'll add a welcoming subheading—a second level heading—to the page, along with an introductory paragraph:

File: **index.html (excerpt)**

```
<body>
  <h1>BubbleUnder.com</h1>
  <p>Diving club for the south-west UK - let's make a splash!</p>
  <h2>Welcome to our super-dooper Scuba site</h2>
  <p>Glad you could drop in and share some air with us! You've
      passed your underwater navigation skills and successfully
      found your way to the start point - or in this case, our
      home page.</p>
</body>
```

Apologies to anyone who doesn't get my hilarious puns on diving terminology. Come to think of it, apologies to those who do … they're truly cringe-worthy!

Hey! Where'd it All Go?

You'll notice that we didn't repeat the markup for the entire page in the above example. Why? Because paper costs money, trees are beautiful, and you didn't buy this book for weight-training purposes. In short, we won't repeat all the markup all the time; instead, we'll focus on the parts that have changed, or have been added to. And remember: if you think you've missed something, don't worry. You can find all of the examples in the book's code archive.

Once you've added the subheading and the paragraph that follows it, save your page once more, then take another look at it in your browser (either hit the refresh/reload button in your browser, or double-click on the file icon in the location at which you saved it). You should be looking at something like the display shown in Figure 2.14.

Figure 2.14. The homepage taking shape

So, the homepage reads a lot like many other homepages at this stage: it has some basic introductory text to welcome visitors, but not much more. But what exactly is the site about? Or, to be more precise, what will it be about once it's built?

What's it All About?

Notice that, despite our inclusion of a couple of headings and a couple of paragraphs, there is little to suggest what this site is about. All visitors know so far is that the site's about diving. Let's add some more explanatory text to the page, along with some contact information:

- ❑ Beneath the content you already have on the page, add another heading: this time, make it a level three heading that reads, "About Us" (remember to include both the opening and closing tags for the heading element).
- ❑ Next, add the following text. Note that there is more than one paragraph.

> Bubble Under is a group of diving enthusiasts based in the south-west UK who meet up for diving trips in the summer months when the weather is good and the bacon rolls are flowing. We arrange weekends away as small groups to cut the costs of accommodation and travel, and to ensure that everyone gets a trustworthy dive buddy.
>
> Although we're based in the south-west, we don't stay on our own turf: past diving weekends have included trips up to Scapa Flow in Scotland and to Malta's numerous wreck sites.
>
> When we're not diving, we often meet up in a local pub to talk about our recent adventures (any excuse, eh?).

Save yourself Some Trouble

If you don't feel like typing out all this content, you can paraphrase, or copy it from the code archive. I've deliberately chosen to put a realistic amount of content on the page, so that you can see the effect of several paragraphs on our display.

- ❑ Next, add a Contact Us section, again, signified by a level three heading.
- ❑ Finally, add some simple contact details as follows:

> To find out more, contact Club Secretary Bob Dobalina on 01793 641207 or email bob@bubbleunder.com.

So, just to recap, we suggested using different heading levels to signify the importance of the different sections and paragraphs within the document. With that in mind, you should have something like the markup below in the body of your document:

File: **index.html (excerpt)**

```
<h1>BubbleUnder.com</h1>
<p>Diving club for the south-west UK - let's make a splash!</p>
```

```
<h2>Welcome to our super-dooper Scuba site</h2>
<p>Glad you could drop in and share some air with us! You've
    passed your underwater navigation skills and successfully
    found your way to the start point - or in this case, our home
    page.</p>
<h3>About Us</h3>
<p>Bubble Under is a group of diving enthusiasts based in the
    south-west UK who meet up for diving trips in the summer
    months when the weather is good and the bacon rolls are
    flowing. We arrange weekends away as small groups to cut the
    costs of accommodation and travel and to ensure that everyone
    gets a trustworthy dive buddy.</p>
<p>Although we're based in the south-west, we don't stay on our
    own turf: past diving weekends away have included trips up to
    Scapa Flow in Scotland and to Malta's numerous wreck
    sites.</p>
<p>When we're not diving, we often meet up in a local pub
    to talk about our recent adventures (any excuse, eh?).</p>
<h3>Contact Us</h3>
<p>To find out more, contact Club Secretary Bob Dobalina on
    01793 641207 or email bob@bubbleunder.com.</p>
```

Clickable Email Links

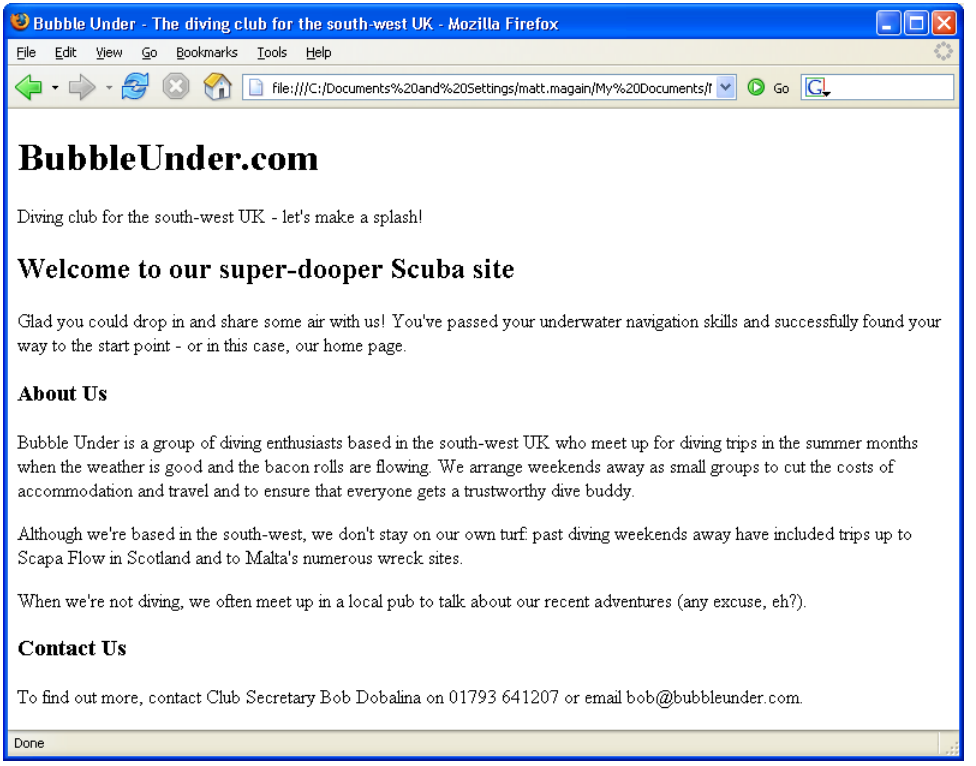
It's all well and good to put an email address on the page, but it's hardly perfect. To use this address, a site visitor would need to copy and paste the address into an email message. Surely there's a simpler way? There certainly is:

```
<p>To find out more, contact Club Secretary Bob Dobalina
    on 01793 641207 or <a
    href="mailto:bob@bubbleunder.com.">email
    bob@bubbleunder.com</a>.</p>
```

This clickable email link uses the `a` element, which is used to create links on web pages (this will be explained later in this chapter). The `mailto:` prefix tells the browser that the link needs to be treated as an email address (that is, the email program should be opened for this link). The content that follows the `mailto:` section should be a valid email address in the format *`username@domain`*.

Add this to the web page now, save it, then refresh the view in your browser. Try clicking on the underlined text: it should open your email program automatically, and display an email form in which the To: address is already completed.

Figure 2.15. Viewing `index.html`



It's still not very exciting, is it? Trust me, we'll get there! The important thing to focus on at this stage is what the content of your site should comprise, and how it might be structured. We haven't gone into great detail about document structure yet, other than to discuss the use of different levels of headings, but we'll be looking at this in more detail later in this chapter. In the next chapter, we'll see how you can begin to **style** your document—that is, change the font, color, letter spacing and more—but for now, let's concentrate on the content and structure.

The page so far seems a little boring, doesn't it? Let's sharpen it up a little. We can only keep looking at a page of black and white for so long—let's insert an image into the document. Here's how the `img` element is applied within the context of the page's markup:

File: **index.html (excerpt)**

```
<h2>Welcome to our super-dooper Scuba site</h2>
<p><img src="divers-circle.jpg" width="200" height="162"
    alt="A circle of divers practice their skills" /></p>
<p>Glad you could drop in and share some air with us! You've
    passed your underwater navigation skills and successfully
    found your way to the start point - or in this case, our home
    page.</p>
```

Whoa, hold up there. Let's talk about all this `img` stuff (don't worry, it's pretty simple). The `img` element is used to insert an image into our web page, and the attributes `src`, `alt`, `width`, and `height` describe the image that we're inserting. `src` is just the name of the image file. In this case, it's `divers-circle.jpg`, which you can grab from the code archive. `alt` is some alternative text that can be displayed in place of the image if, for some reason, it can't be displayed. This is useful for visitors to your site who may be blind (they have web browsers, too!), search engines, and users of slow Internet connections. `width` and `height` should be pretty obvious: they give the width and height of the image, measured in pixels. Don't worry if you don't know what a pixel is; we'll look into images in more detail a bit later.

Figure 2.16. Divers pausing in a circle

Go and grab `divers-circle.jpg` from the code archive, and put it into your web site's folder. The image is shown in Figure 2.16.

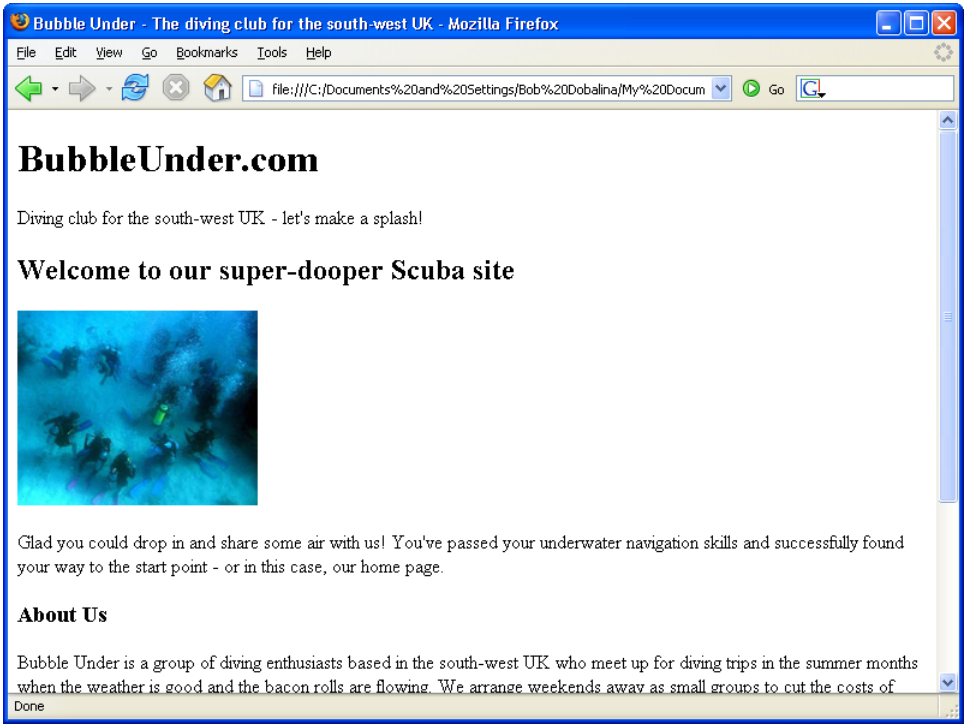
Open `index.html` in your text editor and add the following markup just after the level two heading (`h2`):

File: **`index.html` (excerpt)**

```
<p><img src="divers-circle.jpg" width="200" height="162"
    alt="A circle of divers practice their skills" /></p>
```

Save the changes, then view the homepage in your browser. It should look like the display shown in Figure 2.17.

Figure 2.17. Displaying an image on the homepage



Adding Structure

Paragraphs? No problem. Headings? You've got them under your belt. In fact, you're now familiar with the basic structure of a web page. The small selection of tags that we've discussed so far are fairly easy to remember, as their purposes are obvious (remember: `p` = paragraph). But what the heck is a `div`?

A `div` is used to divide up a web page and, in doing so, to provide a definite structure that can be used to great effect when combined with CSS.

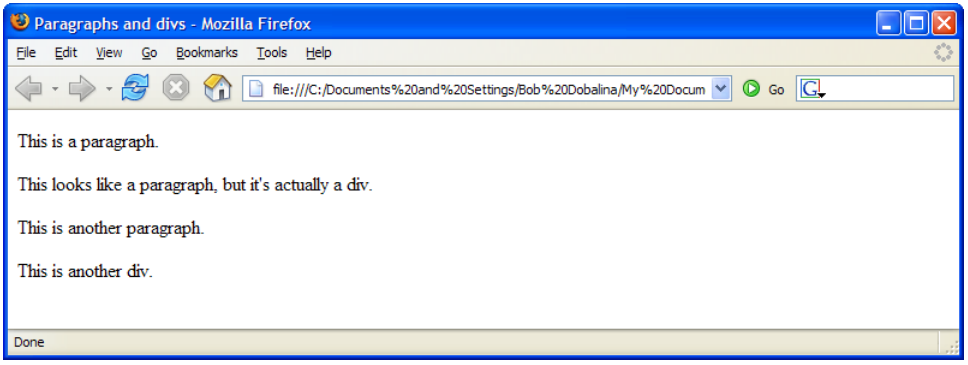
The strange thing about a `div` is that, normally, it has no effect on the styling of the text it contains, except for the fact that it adds a break before and after the contained text. Consider the following markup:

```
<p>This is a paragraph.</p>
<p>So is this.</p>
```

If you were to change those paragraph tags to `div`s, there would be very little obvious change in the page's appearance when you viewed it in a browser (as Figure 2.18 shows).

```
<p>This is a paragraph.</p>
<div>This looks like a paragraph, but it's actually a div.</div>
<p>This is another paragraph.</p>
<div>This is another div.</div>
```

Figure 2.18. Paragraphs displaying similarly to `div`s



"But they look exactly the same! What's the difference between using a `div` and a `p`?", you may protest. Well, the purpose of a `div` is to divide up a web page

into distinct sections, adding structural meaning, whereas `p` should be used to create a paragraph of text.

Use Elements as Intended

Never use an XHTML element for a purpose for which it was not intended. This really is a golden rule!

Rather than leaving the paragraph tags as they are, you might decide to have something like this:

```
<div>
  <p>This is a paragraph inside a div.</p>
  <p>So is this.</p>
</div>
```

You can have as many paragraphs as you like inside that `div` element, but note that you cannot place `div` elements inside paragraphs. Think of a `div` as a container that's used to group related items together, and you can't go wrong.

If we look at our homepage in the browser, it's possible to identify areas that have certain purposes. These are listed below, and depicted in Figure 2.19. We have:

- a header area that contains:
  - the site name
  - a tag line
- an area of body content

Figure 2.19 shows how the different segments of content can be carved up into distinct areas based on the purposes of those segments.

Take the homepage we've been working on (`index.html`) and, in your text editor of choice, add `<div>` and `</div>` tags around the sections suggested in Figure 2.19. While you're adding those `div`s, add an `id` attribute to each, appropriately allocating the names `header`, `sitebranding`, `tagline`, and `bodycontent`. Remember that attribute names should be written in lowercase, and their values should be contained within quotation marks.

Figure 2.19. Noting distinct sections in the basic web page

No Sharing `id`s!

`id` attributes are used in XHTML to identify elements, so no two elements should share the same `id` value. You can use these `id`s later, when you're dealing with elements via CSS or JavaScript.

`h1`, `header`, and `head`

An `id` attribute set to `header` should not be confused with headings on the page (`h1`, `h2`, and so on); nor is it the same as the `head` of your HTML page. The `id=` attribute could just as easily have been named `topstuff` or `pageheader`. It doesn't matter, so long as the attribute name describes the *purpose* of that page section to a fellow human being (or to yourself 12 months after you devised it, and have forgotten what you were thinking at the time!).

To get you started, I've done a little work on the first part of the page. In the snippet below, that section has been changed to a `div` with an `id` attribute:

File: **`index.html` (excerpt)**

```
<div id="header">
  <h1>BubbleUnder.com</h1>
  <p>Diving club for the south-west UK - let's make a splash!</p>
</div> <!-- end of header div -->
```

Now, try doing the same: apply `divs` to the parts of the content that we've identified as "site branding" and "tag line."

Nesting Explained

We already know that `divs` can contain paragraphs, but a `div` can also contain a number of other `divs`. This is called **nesting**. It's not tricky, it's just a matter of putting one div inside the other, and making sure you get your closing tags right.

```
<div id="outer">
  <div id="nested1">
    <p>A paragraph inside the first nested div.</p>
  </div>
  <div id="nested2">
    <p>A paragraph inside the second nested div.</p>
  </div>
</div>
```

As Figure 2.19 shows, some nesting is taking place: the "site branding" and "tag line" `divs` are nested inside the "header" `div`.

The Sectioned Page: all Divided Up

All things being well, your XHTML should now look like this:

File: **index.html**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Bubble Under - The diving club for the south-west
        UK</title>
    <meta http-equiv="Content-Type"
        content="text/html; charset=utf-8" />
  </head>
  <body>
    <div id="header">
      <div id="sitebranding">
        <h1>BubbleUnder.com</h1>
      </div>
      <div id="tagline">
        <p>Diving club for the south-west UK - let's make
            a splash!</p>
```

```
      </div>
    </div> <!-- end of header div -->
    <div id="bodycontent">
      <h2>Welcome to our super-dooper Scuba site</h2>
      <p><img src="divers-circle.jpg" width="200" height="162"
          alt="A circle of divers practice their skills" /></p>
      <p>Glad you could drop in and share some air with us! You've
          passed your underwater navigation skills and
          successfully found your way to the start point - or in
          this case, our home page.</p>
      <h3>About Us</h3>
      <p>Bubble Under is a group of diving enthusiasts based in
          the south-west UK who meet up for diving trips in the
          summer months when the weather is good and the bacon
          rolls are flowing. We arrange weekends away as small
          groups to cut the costs of accommodation and travel and
          to ensure that everyone gets a trustworthy dive
          buddy.</p>
      <p>Although we're based in the south-west, we don't stay on
          our own turf: past diving weekends away have included
          trips up to Scapa Flow in Scotland and to Malta's
          numerous wreck sites.</p>
      <p>When we're not diving, we often meet up in a local pub
          to talk about our recent adventures (any excuse,
          eh?).</p>
      <h3>Contact Us</h3>
      <p>To find out more, contact Club Secretary Bob Dobalina on
          01793 641207 or
         <a href="mailto:bob@bubbleunder.com">email
         bob@bubbleunder.com</a>.</p>
    </div> <!-- end of bodycontent div -->
  </body>
</html>
```

Indenting your Markup

It's a good idea to "indent" your markup when nesting elements on a web page, as is demonstrated with the items inside the `div` section above. Indenting your code can help resolve problems later, as you can more clearly see which items sit inside other items. Note that indenting is only really useful for the person—perhaps you!—who's looking at the source markup. It does not affect how the browser interprets or displays the web page.[3]

[3]The one exception to this is the `pre` element. Pre is short for pre-formatted, and any text marked up with this element appears on the screen exactly as it appears in the source; in other words, carriage

Notice that, in the markup above, comments appear after some of the closing `div` tags. These comments are optional, but again, commenting is a good habit to get into as it helps you fix problems later. Often, it's not possible to view your opening and closing `<div>` tags in the same window, as they're wrapped around large blocks of XHTML. If you have several nested `<div>` tags, you might see something like this at the end of your markup:

```
    </div>
  </div>
</div>
```

In such cases, you might find it difficult to work out which `div` is being closed off at each point. It may not yet be apparent why this is important or useful, but once we start using CSS to style our pages, errors in the XHTML can have an impact. Adding some comments here and there can really help you debug later.

```
    </div> <!-- end of inner div -->
  </div> <!-- end of nested div -->
</div> <!-- end of outer div -->
```

How does the web page look? Well, we're not going to include a screen shot this time, because adding those `div` elements should make no visual difference at all. The changes we just made are structural ones that we'll build on later.

Show a Little Restraint

Don't go overboard adding `div`s. Some people can get carried away as they section off the page, with `<div>` tags appearing all over the place. Overly enthusiastic use of the `div` can result in a condition that has become known as "`div`-itis." Be careful not to litter your markup with superfluous `<div>` tags just because you can.

Splitting Up the Page

We've been making good progress on our fictitious site … but is a web site really a web site when it contains only one page? Just as the question, "Can you have a sentence with just one word?" can be answered with a one-word sentence ("Yes"), so too can the question about our one-page web site. But it's not really ideal, is it? You didn't buy this book to learn how to create one page, did you?

returns, spaces, and any tabs that you've included will be honored. The `pre` element is usually used to show code examples.

Let's take a look at how we can split the page we've been working on into separate entities, and how these pages relate to each other.

First, let's just ensure that your page is in good shape before we go forward. The page should reflect the markup shown in the last large block presented in the previous section (after we added the `<div>` tags). If not, go to the code archive and grab the version that contains the `divs` (`/chapter2/website_files/05_with_divs/index.html`). Save it as `index.html` in your web site's folder (if you see a prompt that asks whether you want to overwrite the existing file, click Yes).

Got that file ready? Let's break it into three pages. First, make two copies of the file:

- ❑ Click on the `index.html` icon in Windows Explorer or Finder.
- ❑ To copy the file, select Edit > Copy.
- ❑ To paste a copy in the same location, select Edit > Paste.
- ❑ Repeat the process once more.

You should now have three HTML files in the folder that holds your web site files. The `index.html` file should stay as it is for the time being, but take a moment to rename the other two (select each file in turn, choosing File > Rename, if you're using Windows; Mac users, simply select the file by clicking on it, then hit **Return** to edit the filename).

- ❑ Rename one file as `contact.html` (all lowercase).
- ❑ Rename the other one as `about.html` (all lowercase).

Where's my File Extension?

If your filename appears as just `index` in Windows Explorer, your system is currently set up to hide extensions for files that Windows recognizes. To make the extensions visible, follow these simple steps:

1. Launch Windows Explorer.
2. For Windows ME/2000/XP, select Tools > Folder Options... (on Windows 98, this is View > Folder Options...).
3. Select the View tab.

4. In the Advanced Settings group, make sure that Hide extensions for known file types does not have a tick next to it.

We have three identical copies of our XHTML page. Now, we need to edit the content of these pages so that each page includes only the content that's relevant to that page.

To open an existing file in Notepad, select File > Open..., and in the window that appears, change Files of type to All Files. Now, when you go to your Web folder, you'll see that all the files in that folder are available for opening.

Opening a file in TextEdit is a similar process. Select File > Open... to open a file, but make sure that Ignore rich text commands is checked.

In your text editor, open each page in turn, and edit them as follows (remembering to save your changes to each before you open the next file):

`index.html` Delete the "About Us" and "Contact Us" sections (both the headings and the paragraphs that follow them), ensuring that the rest of the markup remains untouched. Be careful not to delete the `<div>` and `</div>` tags that enclose the body content.

`about.html` Delete the introductory spiel (the level two heading and associated paragraphs, including the image) and remove the "Contact Us" section (including the heading and paragraphs).

`contact.html` You should be getting the hang of this now (if you're not sure you've got it right, keep reading: we'll show the altered markup in a moment). This time, we're removing the introductory spiel and the "About Us" section.

Now, each of the three files contains the content that suits its respective filename, but a further change is required for the two newly created files. Open `about.html` in your text editor and make the following amendments:

- Change the contents of the `title` element to read "About BubbleUnder.com: who we are; what this site is for."
- Change the level three heading `<h3>About Us</h3>` to a level two heading. In the process of editing our original homepage, we've lost one of our heading levels. Previously, the "About Us" and "Contact Us" headings were marked up as level three headings that sat under the level two "Welcome" heading.

It's not good practice to skip heading levels—an `h2` following `h1` is preferable to an `h3` following an `h1`.

Next, open `contact.html` in your text editor and make the following changes:

- ❑ Amend the contents of the `title` element to read, "Contact Us at Bubble Under."
- ❑ Change the level three heading to a level two heading, as you did for `about.html`.

If everything has gone to plan, you should have three files named `index.html`, `about.html`, and `contact.html`.

The markup for each should be as follows:

File: **index.html**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Bubble Under - The diving club for the south-west
        UK</title>
    <meta http-equiv="Content-Type"
        content="text/html; charset=utf-8" />
  </head>
  <body>
    <div id="header">
      <div id="sitebranding">
        <h1>BubbleUnder.com</h1>
      </div>
      <div id="tagline">
        <p>Diving club for the south-west UK - let's make a
            splash!</p>
      </div>
    </div> <!-- end of header div -->
    <div id="bodycontent">
      <h2>Welcome to our super-dooper Scuba site</h2>
      <p><img src="divers-circle.jpg"
          alt="A circle of divers practice their skills"
          width="200" height="162" /></p>
      <p>Glad you could drop in and share some air with us! You've
          passed your underwater navigation skills and
          successfully found your way to the start point - or in
          this case, our home page.</p>
```

```
    </div> <!-- end of bodycontent div -->
  </body>
</html>
```

File: **about.html**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>About Bubble Under: who we are, what this site is
        for</title>
    <meta http-equiv="Content-Type"
        content="text/html; charset=utf-8" />
  </head>
  <body>
    <div id="header">
      <div id="sitebranding">
        <h1>BubbleUnder.com</h1>
      </div>
      <div id="tagline">
        <p>Diving club for the south-west UK - let's make a
            splash!</p>
      </div>
    </div> <!-- end of header div -->
    <div id="bodycontent">
      <h2>About Us</h2>
      <p>Bubble Under is a group of diving enthusiasts based in
          the south-west UK who meet up for diving trips in the
          summer months when the weather is good and the bacon
          rolls are flowing. We arrange weekends away as small
          groups to cut the costs of accommodation and travel and
          to ensure that everyone gets a trustworthy dive
          buddy.</p>
      <p>Although we're based in the south-west, we don't stay on
          our own turf: past diving weekends away have included
          trips up to Scapa Flow in Scotland and to Malta's
          numerous wreck sites.</p>
      <p>When we're not diving, we often meet up in a local pub
          to talk about our recent adventures (any excuse,
          eh?).</p>
    </div> <!-- end of bodycontent div -->
  </body>
</html>
```

File: **contact.html**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Contact Us at Bubble Under</title>
    <meta http-equiv="Content-Type"
        content="text/html; charset=utf-8" />
  </head>
  <body>
    <div id="header">
      <div id="sitebranding">
        <h1>BubbleUnder.com</h1>
      </div>
      <div id="tagline">
        <p>Diving club for the south-west UK - let's make a
            splash!</p>
      </div>
    </div> <!-- end of header div -->
    <div id="bodycontent">
      <h2>Contact Us</h2>
      <p>To find out more, contact Club Secretary Bob Dobalina on
          01793 641207 or <a
          href="mailto:bob@bubbleunder.com">email
          bob@bubbleunder.com</a>.</p>
    </div> <!-- end of bodycontent div -->
  </body>
</html>
```

Linking Between our New Pages

We've successfully created a three-page web site, but there's a small problem: there are no links between the pages! Try for yourself: open `index.html` and take a look at the display. How will you get from one page to another?

To enable site visitors to move around, we need to add navigation. Navigation relies on **anchors**, which are more commonly referred to as links. The XHTML for an anchor, or link, is as follows:

```
<a href="filename.html">Link text here</a>
```

The `a` element might not be intuitive (it stands for "anchor"), but you'll get to know this one very quickly: it's what the Web is built on! There are almost as

many links in use on the Web as there are Elvis impersonators in Las Vegas. *Almost*.

- The `a` element contains the **link text** that will be clicked (which, by default, appears on the screen as blue, underlined text).
- The `href` attribute refers to the URL to which you're linking (be that a file stored locally on your computer, or a page on a live web site). Unfortunately, again, `href` is not immediately memorable (it stands for "hypertext reference"), but you'll use it so often that you'll soon remember it.

Don't Click Here!

The link text—the words inside the anchor element, which appear underlined on the screen—should be a neat summary of that link's purpose (to "email bob@bubbleunder.com," for example). All too often, you'll see people asking you to "Click here to submit an image," or "Click here to notify us of your change of address." Never use "Click here" links—it really is bad linking practice and is discouraged for usability and accessibility reasons.[4]

Let's create a simple navigation menu that you can drop into your pages. Our navigation is just a list of three links. Here's the markup:

```
<ul>
  <li><a href="index.html">Home</a></li>
  <li><a href="about.html">About Us</a></li>
  <li><a href="contact.html">Contact Us</a></li>
</ul>
```

We'll place all of this inside a `div`, so we can quickly and easily see what this block of XHTML represents.

```
<div id="navigation">
  <ul>
    <li><a href="index.html">Home</a></li>
    <li><a href="about.html">About Us</a></li>
    <li><a href="contact.html">Contact Us</a></li>
  </ul>
</div> <!-- end of navigation div -->
```

Now, we just need to paste this markup into an appropriate place on each of our pages. A good position would be just after the header has finished, before the

[4] *Why 'Click here' is bad linking practice* [http://www.cs.tut.fi/~jkorpela/www/click.html], Jukka Korpela.

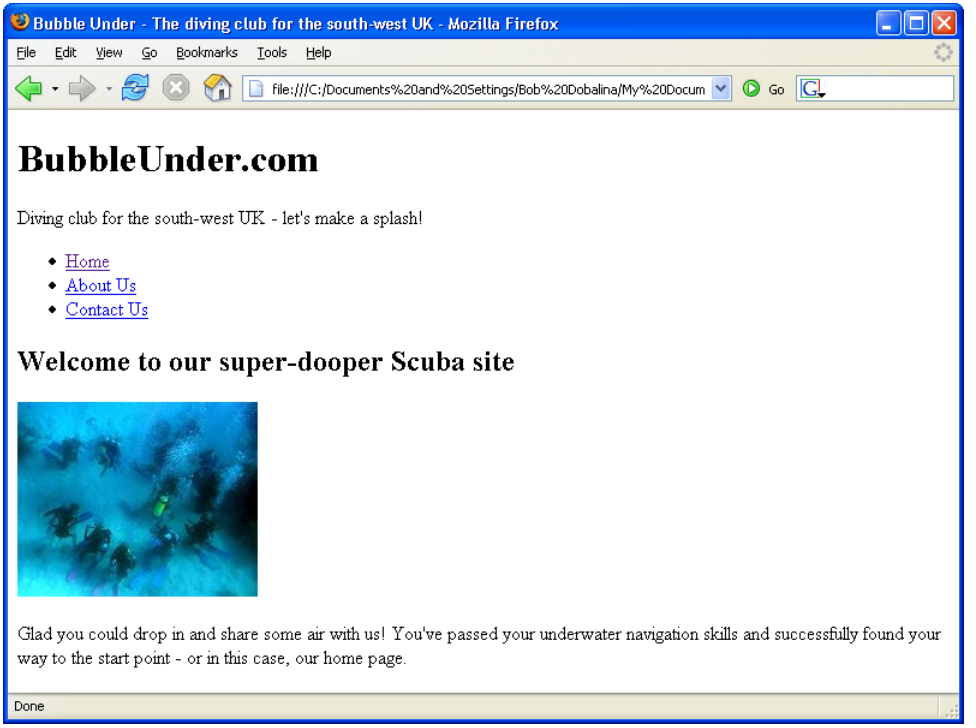
main body content starts. In the code below, the navigation block appears in position on the homepage:

File: **index.html**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Bubble Under - The diving club for the south-west
        UK</title>
    <meta http-equiv="Content-Type" content="text/html;
        charset=utf-8" />
  </head>
  <body>
    <div id="header">
      <div id="sitebranding">
        <h1>BubbleUnder.com</h1>
      </div>
      <div id="tagline">
        <p>Diving club for the south-west UK - let's make a
            splash!</p>
      </div>
    </div> <!-- end of header div -->
    <div id="navigation">
      <ul>
        <li><a href="index.html">Home</a></li>
        <li><a href="about.html">About Us</a></li>
        <li><a href="contact.html">Contact Us</a></li>
      </ul>
    </div> <!-- end of navigation div -->
    <div id="bodycontent">
      <h2>Welcome to our super-dooper Scuba site</h2>
      <p><img src="divers-circle.jpg" width="200" height="162"
          alt="A circle of divers practice their skills" /></p>
      <p>Glad you could drop in and share some air with us!
          You've passed your underwater navigation skills and
          successfully found your way to the start point - or in
          this case, our home page.</p>
    </div> <!-- end of bodycontent div -->
  </body>
</html>
```

You should now be looking at a page like the one shown in Figure 2.20.

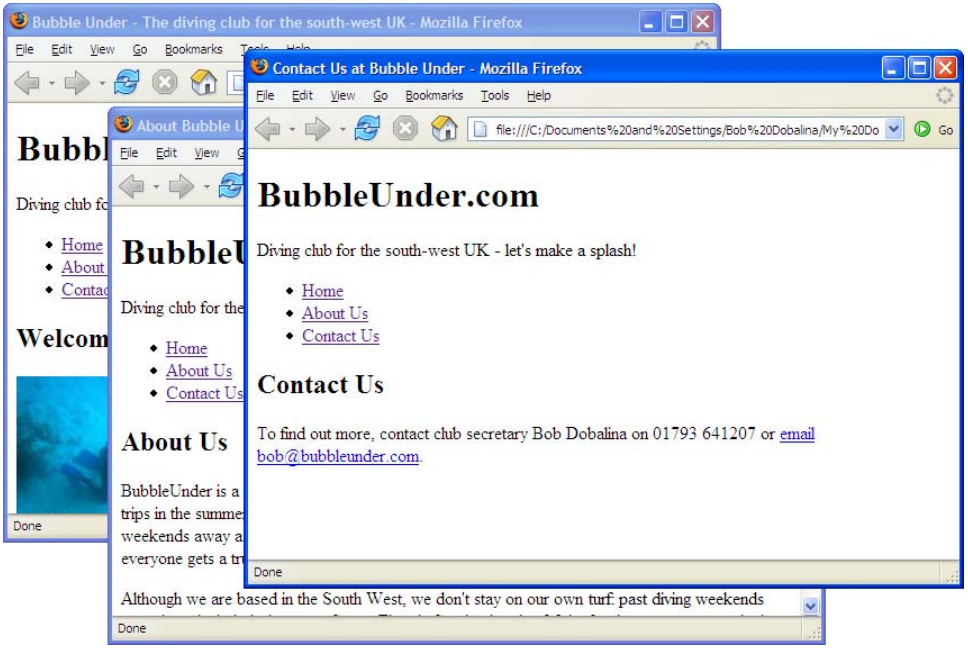
Figure 2.20. Displaying simple navigation on the page



Add the links to `contact.html` and `about.html`, then try clicking on the links that you've just added. It should be possible to flick between all three pages, as shown in Figure 2.21.

This is a landmark: you're now the creator of a working, navigable web site! Don't crack open the bubbly just yet, though: first, let's discuss a few more XHTML elements that you can add to your pages.

Figure 2.21. Navigating the web site



The `blockquote` (Who Said That?)

We're going to add a sound-bite—well, a written quote, to be precise—to the About Us page. Here's the line:

> "Happiness is a dip in the ocean followed by a pint or two of Old Speckled Hen. You can quote me on that!"

We'll add the quote after the final paragraph in `about.html`; here's the markup you'll need:

```
<blockquote>
  <p>"Happiness is a dip in the ocean followed by a pint or two of
      Old Speckled Hen. You can quote me on that!"</p>
</blockquote>
```

Or is it? Who's doing the talking? Well, it's our dear (though fictional) Club Secretary, Bob Dobalina.

File: **about.html (excerpt)**

```
<p>Or as our man Bob Dobalina would put it:</p>
<blockquote>
  <p>"Happiness is a dip in the ocean followed by a pint or two of
      Old Speckled Hen. You can quote me on that!"</p>
</blockquote>
```

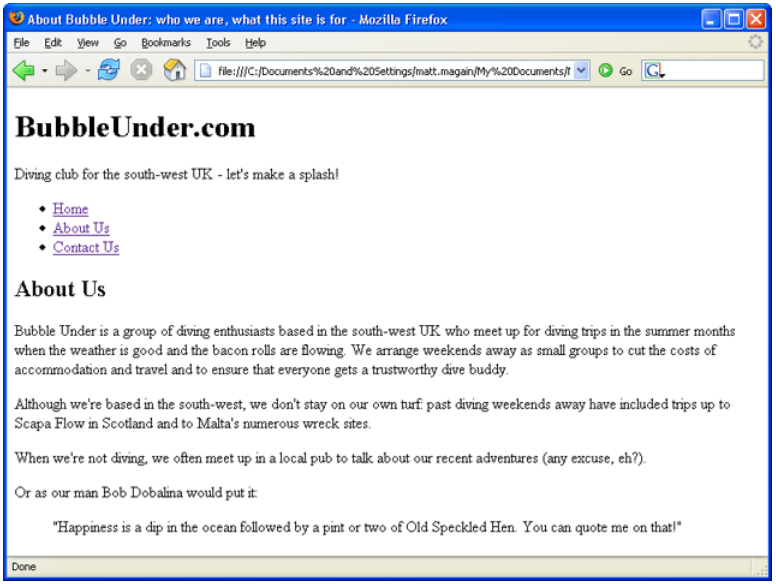
The quotation can contain as many paragraphs as you like, as long as each one starts and ends correctly, and the opening `<blockquote>` tag is closed off properly.

Displaying `blockquotes`

In most browsers, your use of `blockquote` will see the quoted text indented in the page display. This effect can be overridden if it's not to your taste, but that's something we'll cover in a later chapter. On the flip side, you should *never* use the `blockquote` element for the purposes of indenting text. This is very poor form. Only use `blockquote` for its intended purpose: to present a quotation. There are other, better ways to create visual indentations.

Figure 2.22 shows how the `blockquote` above will appear on the page.

Figure 2.22. Displaying a quotation on the page



The `cite` Element

If the quote to which you've referred is written elsewhere—in a magazine, for instance, or a book, or even your own web site—you can add some information to communicate the quote's source. One way is to use the `cite` element. A citation, by default, will style the text in italics. Here's how the markup would look for a citation:

```
<p>I remember reading <cite>Salem's Lot</cite> by Stephen King as
    a child, and being very scared of the dark for days after.</p>
```

If the citation includes a web page to which you can refer, this information can also be displayed. Strangely, the `cite` element has an optional `cite` attribute. Weird. How did *that* one get through the approval process? Anyway, here's how it looks:

```
<p>One of my favorite travel writers is Pete Moore. I particularly
    liked <cite cite="http://www.petermoore.net/">Swahili For The
    Broken Hearted</cite>.</p>
```

We're not using the `cite` element in the diving web site, but you may find it useful in your own web site projects.

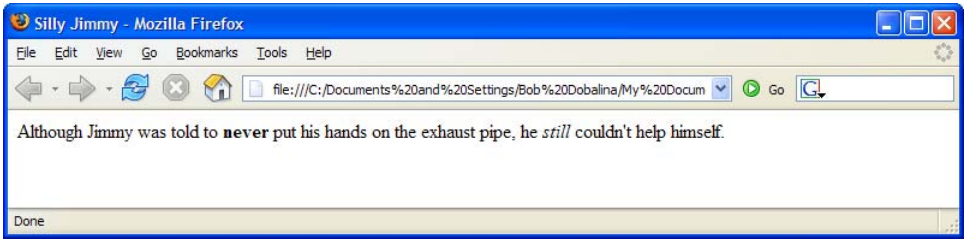
`strong` and `em`

We mentioned the `em` element earlier in this chapter. It's a fairly straightforward element to remember. If you can imagine yourself adding some kind of inflection as you say a word, then emphasis is probably what you need. If you're looking to strike a slightly more forceful tone, then you should consider "going in `strong`."

By default, adding `em` will style text in italics, while using `strong` makes the text bold. You can combine the two if you want, but usually, one or the other will suffice. The examples below should help you understand what these elements are used for. Figure 2.23 shows how they appear in the browser.

```
<p>Although Jimmy was told to <strong>never</strong> put his hands
    on the exhaust pipe, he <em>still</em> couldn't help
    himself.</p>
```

Figure 2.23. Displaying different emphasis styles in the browser



Taking a Break

The chapter's almost at an end, so why take a break? Well, this is just an excuse for a headline pun! We have one more element to look at: the break element.

The break element (`br`) basically replicates what happens when you hit the carriage return on an old typewriter. To create a paragraph on an old typewriter, you'd hit **Enter** twice to give the necessary spacing. In XHTML, the fact that you're marking up a paragraph with `<p>` and `</p>` tags means the spacing is worked out for you automatically. However, if you just want to signify the start of a new line, rather than a new paragraph, the element you need is `br`, as demonstrated in this limerick:[5]

```
<p>The limerick packs laughs anatomical,<br />
Into space that is quite economical.<br />
But the good ones I've seen,<br />
So seldom are clean,<br />
And the clean ones so seldom are comical.</p>
```

Avoid Multiple Breaks

It's all too easy to resort to using multiple breaks in a web page to achieve a visual effect. If you find yourself doing this, something's wrong: you almost certainly need to look for a more suitable technique (we'll look at how this visual affect should be achieved later). Be careful in your use of `br`.

Note that `br` is an empty element, just like `meta` and `img`, so it's written as `<br />`.

[5] http://en.wikipedia.org/wiki/Limerick_(poetry)

Summary

Wow—what a great start we've made! In this chapter, you've built a single web page gradually into three linked pages. You've become familiar with the most commonly used XHTML tags, as well as some of the less common ones that you can apply to your web pages. But, somehow, despite all your efforts, the web pages are still looking a little on the bland side. We're going to fix that very soon: in the next chapter, we'll start to add some splashes of color, and make the site look a little more like a fun diving site and less like a boring old Word document.

3

Adding Some Style

In Chapter 1 and Chapter 2, we stepped through the process of setting up your computer so that we could develop web sites, and pulled together the beginnings of a web site with which you could impress your friends and family! The trouble is, when you came to show off your fledgling site to your nearest and dearest, they *weren't* impressed!

"That's nice dear," said your better half, then carried on doing the dishes. Even Rover (normally your biggest fan), was singularly unimpressed, and decided it would be better to curl up in a warm corner until you stopped waving the laptop in his face. What have you done wrong?

The answer is: nothing. It's true that the web site *may* look a little bland at present, but the underlying structure on which it's built is rock-solid. To return to our automotive analogy, you now have a perfect chassis and some decent bodywork, and, while your car's not making people turn their heads just yet, it's only a matter of time. Just let them see what you can do with this rolling shell!

In this chapter, we'll begin the process of adding that lick of paint to your otherwise plain site. The tool for the job is **Cascading Style Sheets—CSS** to those in the know (or with limited typing abilities). Let's take a look at what CSS can do for you.

What is CSS?

As this chapter revolves almost exclusively around CSS, it's probably a good idea to begin with a basic discussion of what CSS is, and why you should use it. As we've already mentioned, CSS stands for Cascading Style Sheets, but that's too much of a mouthful for most people—we'll stick with the abbreviation!

CSS is a language that allows you to change elements such as text size, color, bolding, background colors, border styles, and colors—even the position of elements on the page. Let's take a look at some CSS in action; we'll start by learning about **inline styles**.

Inline Styles

If you're familiar with Microsoft Word (or a similar word processing package), you may well have created your fair share of flyers, advertisements, or personal newsletters (as well as the more mundane letters to the local authorities and so on). In doing so, you've probably used text formatting options to color certain parts of your text. It's as simple as highlighting the words you want to change, then clicking on a color in a drop-down palette. The same effect can be achieved in XHTML using a little bit of inline CSS. This is what it looks like:

```
<p style="color: red">The quick brown fox jumped over the lazy
    dog.</p>
```

In the example above, we use a `style` attribute inside the opening `<p>` tag. Applying a style to a specific XHTML element in this way is known as using an "inline style."

The `style` attribute can contain one or more **declarations** between its quotation marks. A declaration is made up of two parts: a **property**, and a **value** for that property. In the example above, the declaration is `color: red` (`color` being the property and `red` being its value).

If you wanted to, you could add another declaration to the example above. For instance, as well as having the text display in red, you might want it to appear in a bold typeface. The property that controls this effect is `font-weight`; it can have a range of different values, but mostly you'll use `normal` or `bold`. As you might expect, you'd use the following markup to make the paragraph red and bold:

```
<p style="color: red; font-weight: bold">The quick brown fox
    jumped over the lazy dog.</p>
```

Notice that a semicolon separates the two declarations. You could carry on adding styles in this way, but beware: this approach can get messy! There are cleverer ways to apply styling, as we'll see very soon.

Adding Inline Styles

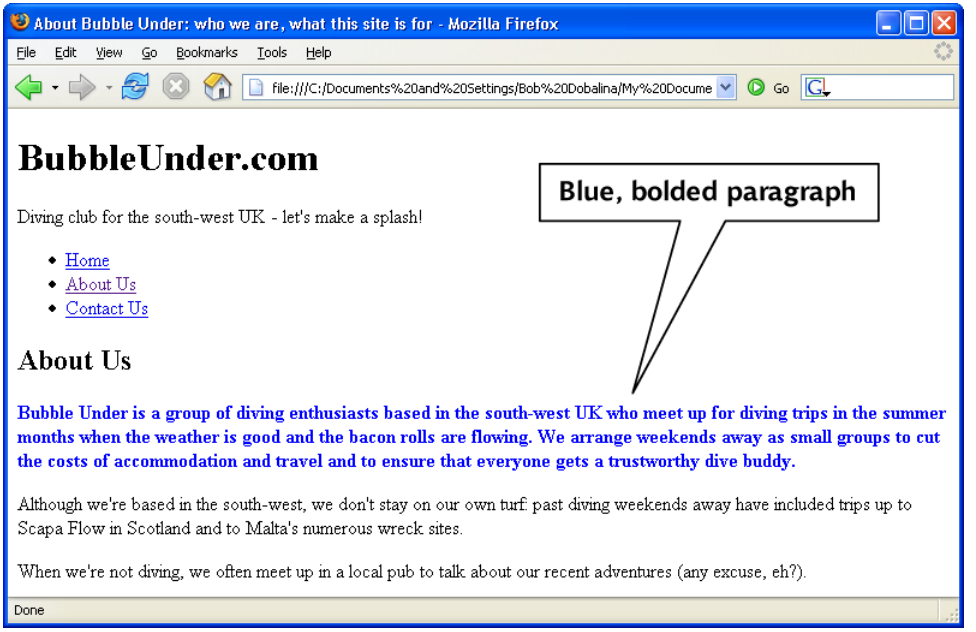
Open `about.html` in your text editor, and add an inline style. We want to make the text in the first paragraph after the "About Us" heading bold and blue. Refer to the previous example as you create the style.

Does the markup for your paragraph look like this?

```
<p style="color: blue; font-weight: bold;">Bubble Under is a group
    of diving enthusiasts based in the south-west UK who meet up
    for diving trips in the summer months when the weather is good
    and the bacon rolls are flowing. We arrange weekends away as
    small groups to cut the costs of accommodation and travel and
    to ensure that everyone gets a trustworthy dive buddy.</p>
```

If the markup *does* look like that shown here, save `about.html` and take a look at it in your browser. It should appear like the page shown in Figure 3.1.

Figure 3.1. Content displaying with blue and bold styles



The span Element

You can easily color a whole paragraph like this, but more often than not, you'll want to pick out just a couple of specific words to highlight within a paragraph. You can do this using a `span` element, which can be wrapped around any content you like. Unlike `p`, which means paragraph, or `blockquote`, which signifies a quotation, `span` has no "meaning." A `span` is little more than a tool for highlighting the start and end of something to which you want to apply a style.[1] Instead of making that whole paragraph blue, we might want just the first two words, "Bubble Under," to be blue and bold. Here's how we can use the `span` element to achieve this:

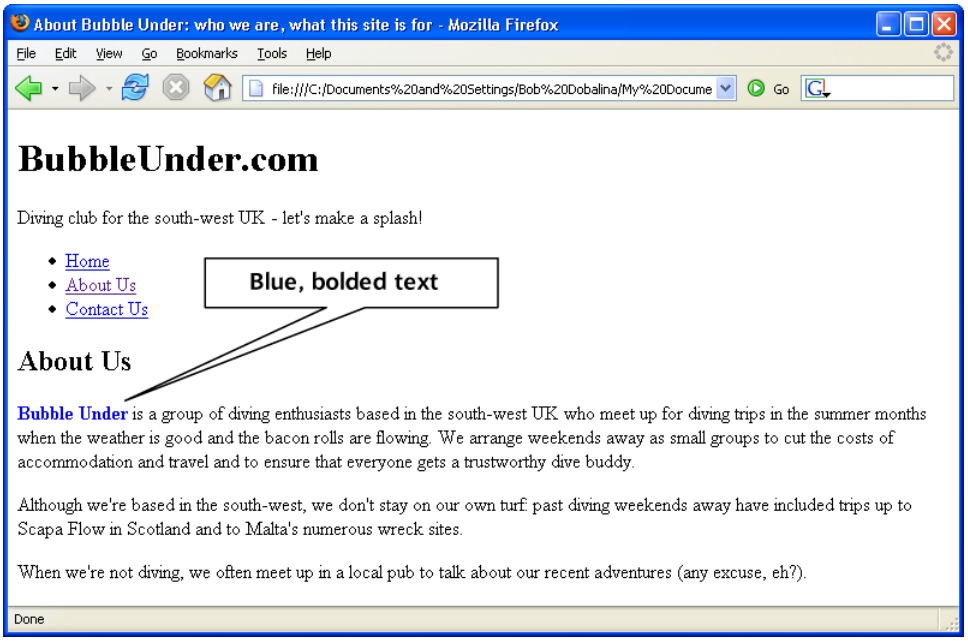
```
<p><span style="color: blue; font-weight: bold;">Bubble
    Under</span> is a group of diving enthusiasts based in the
    south-west UK who meet up for diving trips in the summer
    months when the weather is good and the bacon rolls are
```

[1] Applying a `span` also gives you the ability to do some other clever things to your web page with scripting, but for our purposes, its scope is limited to the things it allows you to do with CSS.

```
    flowing. We arrange weekends away as small groups to cut the
    costs of accommodation and travel and to ensure that everyone
    gets a trustworthy dive buddy.</p>
```

When we view that markup in a browser, we see the display shown in Figure 3.2.

Figure 3.2. Using the span element to pick out specific words for styling



Let's take a quick look at other ways that we can apply inline styles (don't worry, this isn't part of our project site; feel free to experiment).

```
<p style="font-style: italic">The quick brown fox jumped over the
    lazy dog.</p>
```

Not surprisingly, that CSS declaration will italicize all the text in the paragraph. Here's another example, in which `span` is used to highlight specific words:

```
<p>The quick brown fox <span style="font-style: italic;
    font-weight: bold">jumped</span> over the lazy dog.</p>
```

Embedded Styles

Inline styles are a simple, quick way to apply some "makeup" to specific sections of a document, but this is not the best method of styling a page. Wouldn't it be better if you could set styles in just one place, rather than having to type them out every time you wanted to use them?

Embedded style sheets are a logical step up. An embedded style allows you to add to the start of a web page a section that sets out all the styles that will be used on that page. To do this, you need to use the `style` element inside the `head`:

```
<head>
  <title>Bubble Under - The diving club for the south-west
      UK</title>
  <meta http-equiv="Content-Type"
      content="text/html; charset=utf-8" />
  <style type="text/css">
    p {
      font-weight: bold;
    }
  </style>
</head>
```

In the markup shown above, we've moved the inline style into an embedded style sheet. The embedded style sheet starts with a `<style type="text/css">` tag and, predictably, ends with a `</style>` tag. The actual style declarations are enclosed in a set of **curly braces**: `{` and `}`. The `p` that appears before the first curly brace tells the browser what the style rules are for; in this case, we're making the text inside every `p` element bold. The `p` is called the **selector**, and it's a great tool for quickly and easily changing the appearance of lots of elements on your page. The selector, curly braces, and declarations combine to make up what's called a **rule**.

In this case, our style sheet contains one rule: "Please style all the paragraphs on this page so that the text appears in a bold font."

If we wanted to, we could add more declarations to our rule. For instance, if we wanted to make the text bold and blue, we'd add the declaration `color: blue` to our rule:

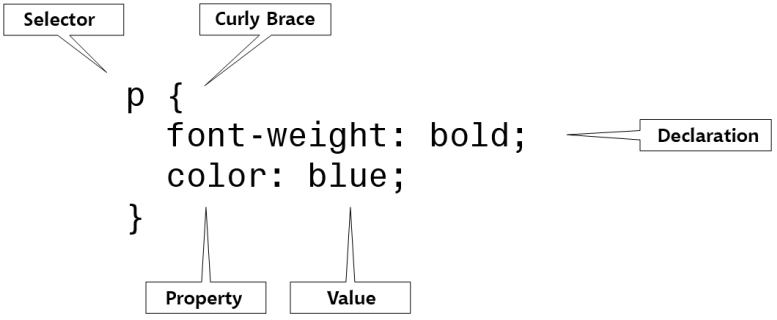
```
<style type="text/css">
  p {
    font-weight: bold;
```

```
    color: blue;
  }
</style>
```

Jargon Break

Okay, okay. There's been an awful lot of jargon so far. Let's recap: Figure 3.3 brings the theory into focus.

Figure 3.3. The anatomy of a rule



Why Embedded Styles are Better than Inline Styles

In the example provided in Figure 3.3, text in all paragraphs will display in bold, blue type. This is useful because it saves you having to type `<p style="font-weight: bold; color: blue">` every time you start a new paragraph—a clear benefit over inline styles. If you wanted to change the color of all paragraph text to red, you need only to change it in the style sheet at the top of the page. That's efficiency for you!

```
<style type="text/css">
  p {
    font-weight: bold;
    color: red;
  }
</style>
```

For this reason, an embedded style sheet is a marked improvement over inline styles. But what if you have a web site that comprises many pages? If you want

to make your changes across the whole site, using embedded style sheets in the way I demonstrated above is still not quite the perfect solution. Why not? Because, to make a site-wide change, you'd have to edit the embedded style sheet on every single page of that site. The best solution is to use an **external style sheet**.

External Style Sheets

Why External Style Sheets are Better than Embedded Styles

An external style sheet provides a location in which you can place styles that can be applied on all of your web pages. This is where the true power of CSS lies, and it's for this reason that we haven't spent too much time applying inline styles or embedded styles to our diving club project site.

The Bad Old Days

In the past, or The Bad Old Days, as we'll call them, people would create web sites on a page-by-page basis, and style them on a page-by-page basis using all manner of nasty elements of which I dare not even speak! Sometimes, these sites grew beyond the webmaster's wildest imagination. "Fantastic," thought Mr or Mrs Webmaster. "My web site now has over 200 pages! Soon I'll be bigger than Microsoft." A few months later, the webmaster decided to redesign the web site and realized, with considerable horror, that he or she would have to alter each and every single web page in order to redesign the site in a consistent manner. Every page needed 20 or more different tweaks, and each tweak had to be applied consistently to every page of the site. Inevitably, some pages were missed, and eventually the redesign plan got unceremoniously dropped. In short, the ugly web site remained ugly for a whole lot longer before dying a nasty death through sheer negligence (indeed, there are many such "legacy" documents littered around the Web today). This need not be the case, though.

CSS gives you the power to set styling rules in one place. When you want to make changes to your web site, you make changes in that one place, and your whole web site changes automatically to reflect those new styles.

Happy Days! CSS Support is Here!

The good news is that the large majority of web browsers in use today offer excellent support for CSS (though this has not always been the case, which is why some people have been slow to adopt CSS-based design). Some browsers can choke on a few of the more advanced CSS techniques, but, by and large, you can style your web pages with CSS and be confident that what you see on your screen is the same view that 99.5% of your intended audience will see.

Creating an External CSS File

If you are to make use of all the benefits that an external style sheet can provide, you'll first need to create a CSS file that can be shared among all the pages of your web site. Open your text editor and enter the following:

File: **style1.css**

```
/*
CSS for Bubble Under site
*/

p {
  font-weight: bold;
  color: blue;
}
```

Save the file in the same folder as your HTML files, naming it `style1.css`; you can save a CSS file in the same way you saved your HTML files.

Note that the first few lines we typed into our CSS file will not actually do anything. Like HTML, CSS allows you to add comments. It's a shame that the tags for HTML comments are different from those for CSS comments, but they do exactly the same thing: they allow you to make notes about your work without affecting the on-screen display. In CSS, a comment starts with a `/*` and ends with a `*/`; the browser ignores anything in between. Above, we used the comment simply to make a note about the purpose of the file, namely that it is the CSS for the Bubble Under site. We've also added a rule to turn the type in all our paragraphs bold and blue.

Linking CSS to a Web Page

Before your CSS can have any effect, you need to link it to the web page, or pages, to which you want the styles to apply. To do this, you need to add a `link`

element to the `head` of each and every web page that will be styled with CSS. Our site contains just three pages at the moment, so this will be nice and easy. The `link` element simply "links" a file to the page on which the element appears; in this case, the linked file is a style sheet.

Below, the new line appears in the context of the homepage:

File: **`index.html` (excerpt)**

```
<head>
  <title>Bubble Under - The diving club for the south-west
      UK</title>
  <meta http-equiv="Content-Type"
      content="text/html; charset=utf-8" />
  <link href="style1.css" rel="stylesheet" type="text/css" />
</head>
```

Let's take a look at what the markup means.

The `href` attribute tells the web browser where the style sheet file (`style1.css`) can be found, in the same way that the `href` attribute is used in an anchor to point to the destination file (e.g. `<a href="home.htm">Home</a>`).

The `rel="stylesheet"` and `type="text/css"` parts of the link tag tell the browser what kind of file is being linked to, and how the browser should handle the content. You should always include these important attributes when linking to a `.css` file.

Empty Element Alert!

The `link` element is another of those special empty elements we discussed in Chapter 2: it has no start or end tags. `link` is a complete element in its own right, and ends with the space and forward slash required by XHTML.

Now that we know how to link our pages to our CSS file, let's try it out on our project site:

- Open each of your web pages—`index.html`, `about.html`, and `contact.html`—in your text editor. Add the following line just before the closing `</head>` tag in each of those files.

  ```
  <link href="style1.css" rel="stylesheet" type="text/css" />
  ```

- Be sure to save each page. Then, try opening each one in your web browser.

All of your paragraphs should now display in bold, blue text. If so, congratulations—you've now linked one style sheet to three separate pages. If you change the color specified in your `.css` file from blue to red, you should see that change reflected across your pages the next time you open them.

Now, using blue, bold text might be a good way to make sure your style sheets are correctly linked, but it's not necessarily the design effect we want to use. Remove the `p` rule from your style sheet, but leave the comment, and let's start building our style sheet for real.

Starting to Build our Style Sheet

The style sheet is ready to be used: it's saved in the right location, and all of your web pages (all three—count 'em!) are linked to it correctly. All we need to do, then, is set some styles!

One of the first changes that people often make to a web site's default styling is to alter the font (or typeface) that's used. On Windows, most browsers will opt for Times New Roman—the font that has been used in all the screen shots we've seen so far. For many people, though, it's a little bit dull, probably because this font is used more than any other. It's very easy to change fonts using CSS's `font-family` property. The best place to use this is within the `body` element, as shown below.

File: **style1.css**

```
/*
CSS for Bubble Under site
*/

body {
  font-family: Verdana;
}
```

Here, I've chosen to use the Verdana font. It's applied to the `body` element because `body` contains every element that you will see on the web page. The nature of the way in which CSS is applied means that every element contained in the `body` element will take on the same font (unless another font is specified for a given element or elements within `body`—but more on that a little later).

Great: Verdana it is! But … what if some people who view your site don't have Verdana installed on their computers? Hmm, that's a tricky one. The short answer is that the browser will make its best guess about which font it should use instead,

but we don't have to make the browser do *all* the guesswork. The `font-family` property allows us to enter multiple fonts in the order in which we'd prefer them to be used. So, we could type something like this:

```
body {
  font-family: Verdana, Helvetica, Arial, sans-serif;
}
```

This translates as: "Please style everything in the body of my web page so that the text appears as Verdana. Failing that, please try using Helvetica and, failing that, Arial. If none of the above are installed, just use whatever sans-serif font is available."

We'll use this selection of fonts in our diving site, so let's open the style sheet file and play around with some CSS.

- ❑ Type the above CSS into `style1.css`.
- ❑ Save the file, then open the homepage (`index.html`) in your browser.

If everything went to plan, your web page (all three of them, actually) should display slightly differently than they did before. Figure 3.4 shows the appearance of our newly-styled homepage.

Sans-serif Fonts: Better for On-screen Viewing

A serif font is one that has all those fancy extensions and flourishes at the ends of each letter. These "flourishes," which are shown in Figure 3.5, are known as **serifs**. They're great for reading printed material, as they give a little shape to the words, making them easier to read.

However, on the screen, serif fonts can become a little messy, especially when they're used for smaller type—there simply aren't enough pixels on the screen to do these little flourishes justice. For this reason, you'll notice that many web sites use **sans-serif** fonts (French words that literally mean "without serif") when the font size is set quite small.

Note that when you refer to a sans-serif font in CSS, you must hyphenate the two words, i.e. `sans-serif`.

Figure 3.4. A font change in the style sheet affects the body of our web pages

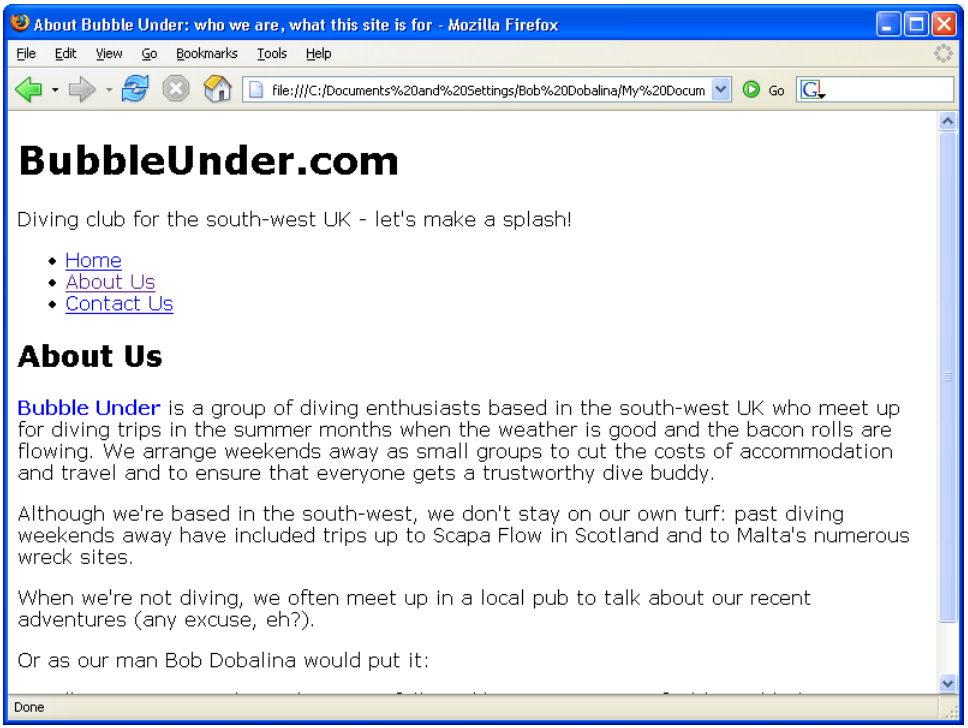


Figure 3.5. Highlighting the serifs of a serif font (Georgia)

Stylish Headings

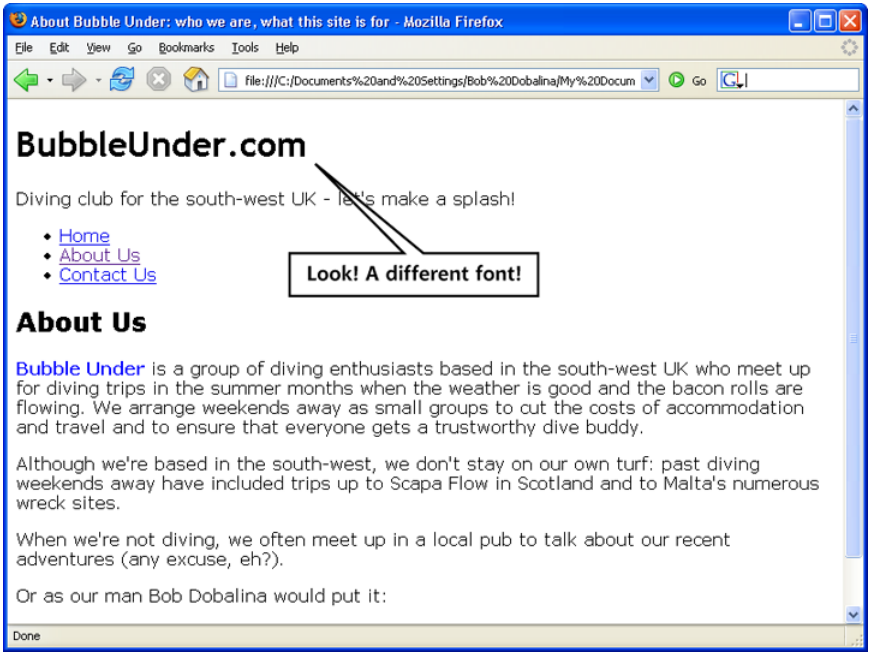
The first element that we'll style is our level 1 headings, denoted by the `h1` element. Let's add some rules to our CSS file to see what's possible when it comes to those headings. In your text editor, add the following to `style1.css`:

```
h1 {
  font-family: "Trebuchet MS", Arial, Helvetica, sans-serif;
}
```

Save the CSS file and refresh your view of the homepage in your browser. Can you see what's changed? All the first-level headings now display in the Trebuchet MS font, while everything else displays in Verdana.

The font we've chosen is another sans-serif font, but it's different enough to provide plenty of contrast with the paragraphs, as Figure 3.6 illustrates.

Figure 3.6. `h1` headings displaying in one font (Trebuchet MS) while other text displays in another (Verdana)



Some Font Names Deserve Quotes

In the example above, "Trebuchet MS" appeared in quotation marks. You don't need to bother wrapping quote marks around font names, unless the font comprises several words, such as "Courier New" or "Times New Roman." A single-word font name, such as Arial or Verdana, does not need to be wrapped in double quotes.

Have a quick look around all three pages of the web site and you'll see that your new styles have been applied to all your web pages. Let's go a step (or two) further.

What's Going On? Nothing's Changed!

If you try refreshing your browser's view of a page and nothing appears to change, first check that you saved the changes you made to the CSS file. If you have saved the altered file, check that you typed the CSS exactly as described. If you did, you may be experiencing a **caching** problem with your browser.

Web browsers "cache" some content. What this means is that, to save having to download files each and every time you visit a given web page, your browser uses versions of the files that it saved to the hard drive previously. This reduces the time it takes to display a web page that has been loaded once before. Unfortunately, your cache can get out of date, and when that happens, the page you visit (i.e. you enter the URL, and the browser pulls the page stored in its cache) might not display the most recent data.

This happens most frequently with images, but it can also occur with CSS files. The good news is that you have control over your browser's cache settings and the amount of space the cache takes up on your hard disk before previously cached content gets replaced with newer data. You can poke around your browser's settings for words like "Cache" or "Temporary Internet Files" to change these settings; however, most users opt to leave their caches to the default settings.

If you're positive that you've made the necessary changes to your CSS file (and saved them) correctly, you may need to "force-reload" the CSS file in your browser.

To stop the caching problem and force the browser to get the most up-to-date version of your CSS file, simply hold down the **Shift** key and click on the Refresh (or reload) icon on your browser's toolbar.

A Mixture of New Styles

Let's change the look of the site a little more: we'll add more styles to the body, and change the look of the navigation. Copy the CSS below into your `style1.css` file (or copy it from the book's code archive):

File: **style1.css**

```
/*
CSS for Bubble Under site
*/

body {
  font-family: Verdana, Helvetica, Arial, sans-serif;
  background-color: #e2edff;
  line-height: 125%;
  padding: 15px;
}

h1 {
  font-family: "Trebuchet MS", Helvetica, Arial, sans-serif;
  font-size: x-large;
}

li {
  font-size: small;
}

h2 {
  color: blue;
  font-size: medium;
  font-weight: normal;
}

li {
  font-size: small;
}

p {
  font-size: small;
  color: navy;
}
```

Save the CSS file, then click Reload in your browser. Hopefully, you'll be looking at a page like the one shown in Figure 3.7.

Figure 3.7. Applying subtle changes to the CSS that affects font display

A New Look in a Flash!

We've introduced quite a few new style declarations here. Let's examine a few of them in the order in which they appear in the CSS file.

Example 3.5. `style1.css` (excerpt)

```
body {
  font-family: Verdana, Helvetica, Arial, sans-serif;
  background-color: #e2edff;
  padding: 15px;
  line-height: 125%;
}
```

The `background-color` property can be applied to most elements on a web page, and there are many different ways in which you can specify the color itself. One is to use recognized color names[2] such as `navy`, `blue`, `red`, `yellow`, and so on. These are easy to remember and spell, but you are limited somewhat by the range. Another way of referencing colors is to use a **hexadecimal** color specification. Yes, you're right: that *does* sound a little scary. I mean, just look at it:

[2] http://www.w3schools.com/html/html_colornames.asp

```
background-color: #e2edff;
```

It's hardly intuitive, is it? This obscure-looking reference translates to a light shade of blue. You could not, as a beginner, begin to guess that this would be a light blue, but thankfully there are numerous tools on the Web that let you choose a color from a chart, then give you the code to match. Take a look at some of these tools,[3] and you'll soon be able to find the hexadecimal numbers you need to create your ideal color schemes.

What the Heck's Hex?

Colors in HTML are often written as a hexadecimal color specification. You might remember the hexadecimal counting system from your high school math class. Or maybe you were asleep up the back of the room. Never mind. Hexadecimal is that weird counting system that goes up to 16 instead of 10; the one you thought you'd never have any practical use for. Well, you do now!

That's right: when you count in hexadecimal, there are not ten, but **16 digits**. The hexadecimal sequence looks like this:

```
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, 10, 11,
12, 13...
```

Eh? What's happening here? Well, as you can see, after we get to 9, instead of going straight to 10 (as we do when counting in decimal) we go through A, B, C, D, E, and F before we finally hit 10. That gives us six extra digits to use when we count! Sound confusing? Well, it is. But as it so happens, computers can count in hexadecimal far better than humans can.

The key here is that all of those numbers that we know and love in the decimal system, like 2,748, 15,000,000 and … er … 69, can be represented in hexadecimal. Look, Table 3.1 proves it!

[3] A good selection of links to color scheme tools is available at http://www.clagnut.com/blog/260/.

Table 3.1. Decimal to hexadecimal conversion

Decimal	Hexadecimal
7	7
15	F
2,748	ABC
15,000,000	E4E1C0
69	45

When a color is expressed as a hexadecimal number, such as `ff0000`, that number actually comprises three values that are joined together. The values represent the proportions of red (the `ff` part), green (the first two zeros), and blue (the second two zeros) that are mixed to create the specified color. Those three primary colors can be combined to display any color on the screen, similar to the way a television set uses different intensities of red, green, and blue to create a single dot on its screen. In this example, `ff` is the value for red, while the green and blue values are zero. It may not surprise you, then, to learn that `#ff0000` will give you the color red.

The `padding` property is used to provide space between the outside edge of the element in question and the content that sits inside it. Because we're referring to the `body` element, you can think of the outside edge as being the top, bottom, and sides of the browser's viewport (that being the part of the browser where the web page is viewable, not including any of the browser's tool bars, menus, or scroll bars). We'll take a look at `padding` in more detail in Chapter 4.

The value we've given to this property specifies how much space must exist between the edge of the viewport and the content. In this case, we've specified `15px`, or 15 pixels. We mentioned pixels before, when we specified the size of an image, but what is a pixel? Basically, one pixel is one of the tiny dots that make up what you see on the computer screen. The screen itself is made up of hundreds of thousands of these pixels, so a 15-pixel border isn't going to take up too much space on your screen!

The `line-height` property is an interesting one. By increasing that value (we used 125% in our example), you can increase the space between lines of text—something that can greatly increase legibility. Try tweaking this value, save your CSS file, and see how the new value affects the text on your web page.

Now, to the paragraph styles:

File: **style1.css (excerpt)**

```
p {
  font-family: Times, "Times New Roman", serif;
  color: navy;
}
```

We've already shown that it's possible to change the color of text in a paragraph; now, we're going to settle on the sensible (and appropriate!) color of navy.

Let's see what's changed with the list item style:

File: **style1.css (excerpt)**

```
li {
  font-size: small;
}
```

The size of the list items has changed ever so slightly through our application of the `font-size` property. Here, we've decided to set the font size using the `small` keyword, but we could just as easily have used the percentage or pixel methods we've already seen—there are many ways to skin a cat with CSS! Font size keywords range from `xx-small` to `xx-large` and offer a quick way to style text. Unfortunately, different browsers implement font size keywords slightly differently, and unfortunately you can't be guaranteed that an `xx-large` font will render at the same size in all browsers. However, unless you're extremely worried about precise sizing, these keywords make a good starting point.[4]

Next, we introduced a new rule, this time for the `h1` element (the main heading on our web pages, which displays the site name) and, once again, used a `font-size` property to specify the size of the text (extra large is the answer!).

File: **style1.css (excerpt)**

```
h1 {
  font-size: x-large;
}
```

The `h2` element also gets a minor makeover:

[4]For more reasons than we have space to discuss, text sizing in CSS is a topic that causes heated debate in some circles. As you become familiar with CSS, you may want to learn more about the other text-sizing techniques that it offers. A good place to start would be SitePoint's CSS discussion forum at http://www.sitepoint.com/launch/cssforum/.

File: **style1.css (excerpt)**

```
h2 {
  color: blue;
  font-size: medium;
  font-weight: normal;
}
```

Browsers usually display headings in bold type, but we can have them display in standard type by giving the `font-weight` property a value of `normal`.

A Beginner's Palette of Styling Options

We've looked at some examples of styles that can be applied to your web pages through CSS, but the examples we've seen have been a mixed bag (deliberately so). There are so many more from which you can pick and choose—too many possibilities, in fact, for us to be able to list them all here. However, this section lists some of the basic properties and values with which you might like to experiment. Feel free to try any of these in your CSS file. Note that we'll be adding to this list in subsequent chapters; it's by no means exhaustive!

`color`
`background-color`

As we've seen, both of these properties can take color keywords (e.g. `red`, `blue`, `green`, or `yellow`) or hexadecimal color specifications such as `#ff0000`.

`font-family`

This property takes a list of fonts, containing any fonts you choose in order of preference. Be sure to provide options that users are likely to have on their computers (e.g. Arial, Verdana, etc.). This list should end with one of the "generic" CSS fonts such as `serif` or `sans-serif`, which any browser that supports CSS will recognize.

`font-size`

This property can be any one of the following:

font size keywords

- `xx-small`
- `x-small`
- `small`
- `medium`

- `large`
- `x-large`
- `xx-large`

a relative unit
such as a percentage (e.g. `140%`)

fixed font sizes

- pixels (e.g. `20px`)
- points (e.g. `12pt`, as you may be used to using in Microsoft Word)

Fixed font sizes are not always a great idea, as they cannot easily be scaled up or down to suit the reader's needs. Relative font sizes are definitely the preferred option.

`font-weight`	`bold` or `normal`
`font-style`	`normal` or `italic`
`text-decoration`	`none`, `underline`, `overline`, or `line-through`

Backing it Up!

Before you experiment with the CSS properties above, it might be an idea to make a backup of your CSS file, just in case you run into difficulties. Remember that you can download all the examples used in this chapter from the code archive if you accidentally mangle your CSS file. If this happens, don't worry! It's all part of the learning process, and you can be sure that no animals will be harmed in the process. Only a handful of poor CSS selectors will be mistreated. Shed no tears!

Recap: the Style Story so Far

Let's allow ourselves a moment to reflect. Our site now boasts a CSS file with a selection of attractive styles. We're in the enviable position of being able to change the site at a whim by altering just that one CSS file. Let's try styling some more of the elements on our web pages.

Changing the Emphasis

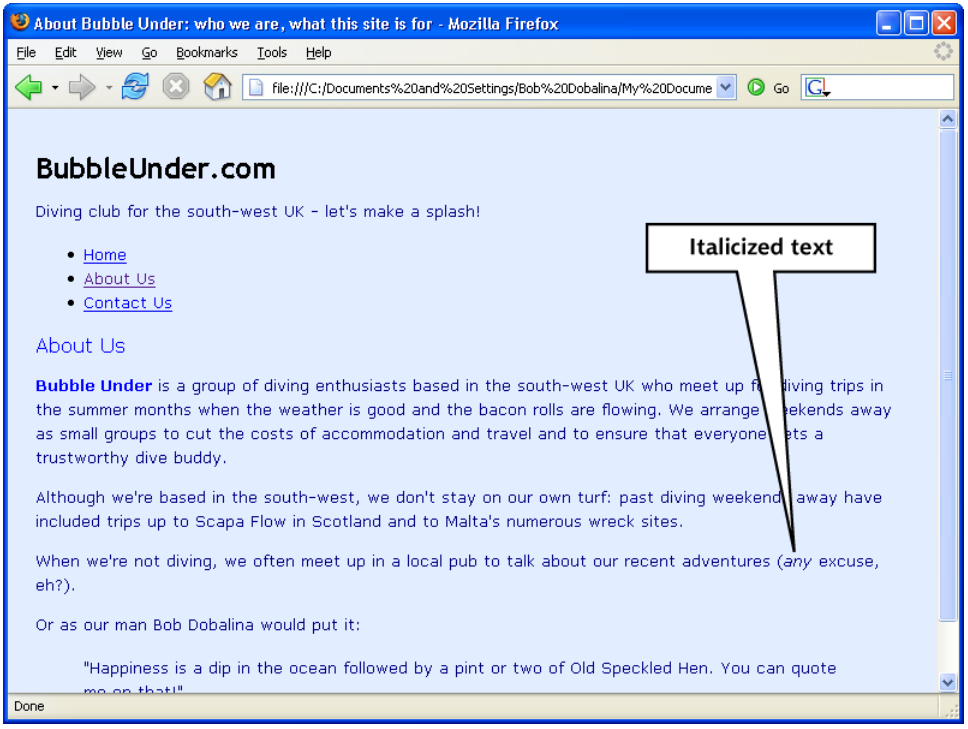
- ❑ Open `about.html` in your text editor.
- ❑ Find the paragraph about meeting up in a local pub and add an emphasis element as shown here:

File: **about.html (excerpt)**

```
<p>And when we're not diving, we often meet up in a local pub
    to talk about our recent adventures (<em>any</em> excuse,
    eh?).</p>
```

- ❑ Save the page, then view it in your web browser; it should appear as shown in Figure 3.8. As you can see, emphasis elements appear in italics by default. We're going to use CSS to change that default style.

Figure 3.8. Using emphasis to set type to italics by default



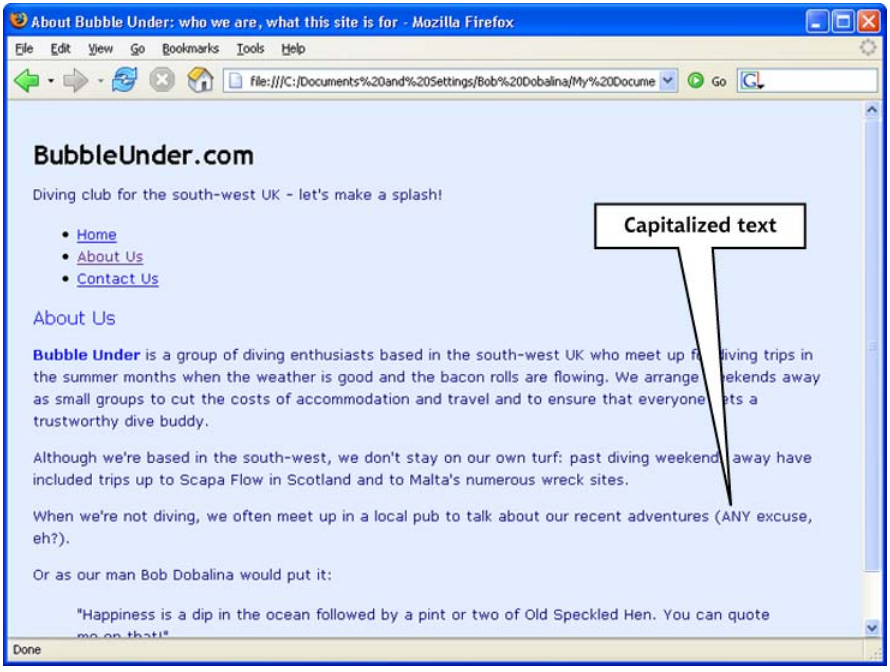
❑ Open `style1.css` (if you haven't already opened it for editing) and add the following rule below the others:

File: **`style1.css` (excerpt)**

```
em {
  font-style: normal;
  text-transform: uppercase;
}
```

❑ Save the CSS file, then refresh your browser's view of the About Us page. Does your page look like Figure 3.9?

Figure 3.9. The emphasis appearing as capitalized text instead of italics



Now, whenever you add an `em` element to any web page of your site (assuming that page is linked to `style1.css`), the emphasized text will appear in capital letters, not italics. But this raises an interesting point: when should you override a browser's default style for one of your own choosing? Presumably, the default

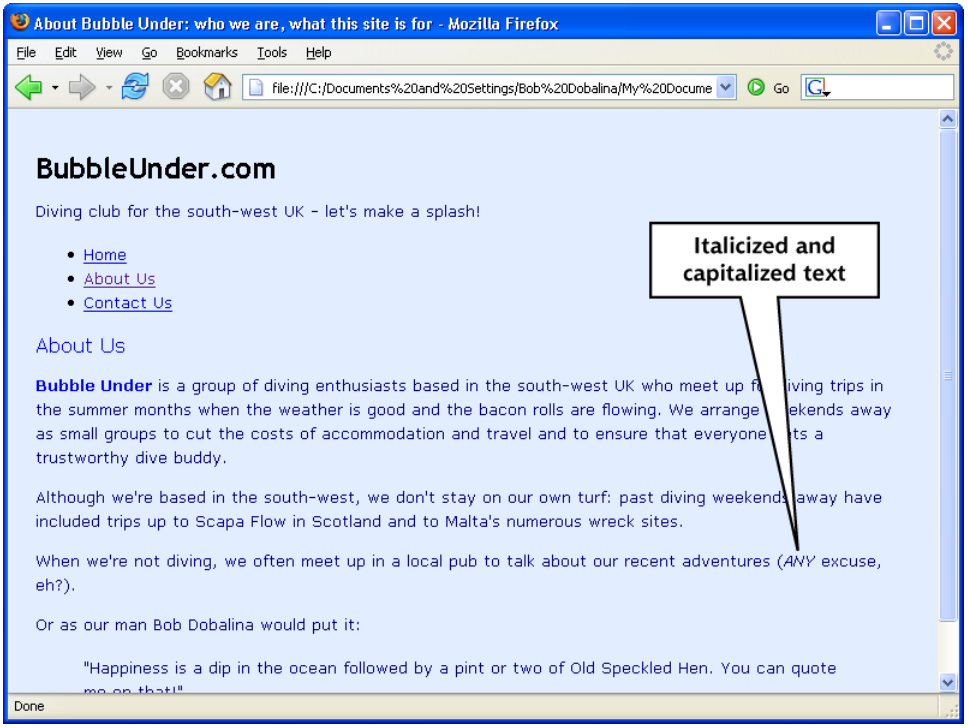
styles that browsers use were selected carefully; how can you be sure that your redefinition of the styles is a good idea? Weren't italics a suitable style for emphasis? They probably were. As Spiderman's creators say, "With great power comes great responsibility," so be sure to exercise caution. Just because you *can* change a default style does not always mean you *should*.

Perhaps a compromise is in order. Let's change the emphasis so that it's still italic, but also appears in uppercase letters. All we need to do is remove the `font-style` declaration; the `em` element will then revert to its default italicized appearance, as depicted in Figure 3.10.

File: **style1.css (excerpt)**

```
em {
  text-transform: uppercase;
}
```

Figure 3.10. Emphasis displaying as uppercase italics



Emphasis vs Italics

You might well be asking yourself, "If I want an italic font, can't I use an italic element?" In fact, HTML provides an `i` element for just this purpose, but its use isn't recommended. Why not? Well, marking something up as `i` says nothing about its meaning; `i` only communicates how it should be presented on the screen. Such elements are referred to as **presentational** HTML elements, and they should be avoided. Likewise, the `b` element (for bold), another old HTML element, should not be used. The preferred option is to use `strong` or, if you just want to display headings in bold, to use CSS.

Why is this important? It mightn't seem important to you, as you look at the italicized text in your web browser. But imagine you were a blind user of software that read web pages aloud to you, instead of displaying them on the screen. Such a program (called a **screen reader**) would read text marked up with an `em` element with slight emphasis, and text marked up with `strong` in a slow, powerful voice … but what would it do with text marked up with `i` or `b`? Well, these elements say nothing about the meaning of the text, so it would ignore them. The same is true of many programs that analyze web pages, such as web indexing programs run by Yahoo! and Google—`em` and `strong` tags tell these programs that the words inside them are important, whereas `i` and `b` tell them nothing.

One other presentational tag that you might see others use, but should *never* copy, is the `u` element. Wrap this around some text and you get needless underlining that only serves to confuse users (because in web pages, underlined text normally signifies a link—something that the `u` element most definitely does not).

Looking at Elements in Context

Pop quiz: which of these items is bigger? A pen or a sheep? Well, the answer is either, depending on the context. If you were a farmer, you'd swear that the pen is bigger. After all, you spend many hours a week rounding up herds of sheep into a nice big pen. If, however, you're an office worker, you'd opt for the sheep being the larger of the two—you'd find it a lot easier to pick up a pen and flip it around your fingers.

Context can change things quite drastically, and context is something that can be used to our advantage in CSS. We can style an element in a number of different ways, depending on its position. Let's head back to our example site for another lesson. Don't be sheepish, now!

Currently, we have styled paragraphs so that they appear in a navy blue, sans-serif font (Verdana, specifically), as does almost everything else on the page:

File: **style1.css (excerpt)**

```
body {
  font-family: Verdana, Helvetica, Arial, sans-serif;
}

p {
   font-size: small;
   color: navy;
}
```

This is all well and good, but there's one paragraph on our site that's a little different than the others in terms of its purpose. Can you spot which one it is? It's our first paragraph, the one in the tag line. Here's the XHTML for that section:

File: **index.html (excerpt)**

```
<div id="tagline">
  <p>Diving club for the south-west UK - let's make a splash!</p>
</div>
```

It's different because it's not really part of the document content and, as such, it might benefit from some different styling. The fact that this particular paragraph is contained within a specific `div` element—which has an `id` attribute of `tagline`—is something that we can put to great use. Because it's contained within its own `div`, we can set a rule for this paragraph and this paragraph only.

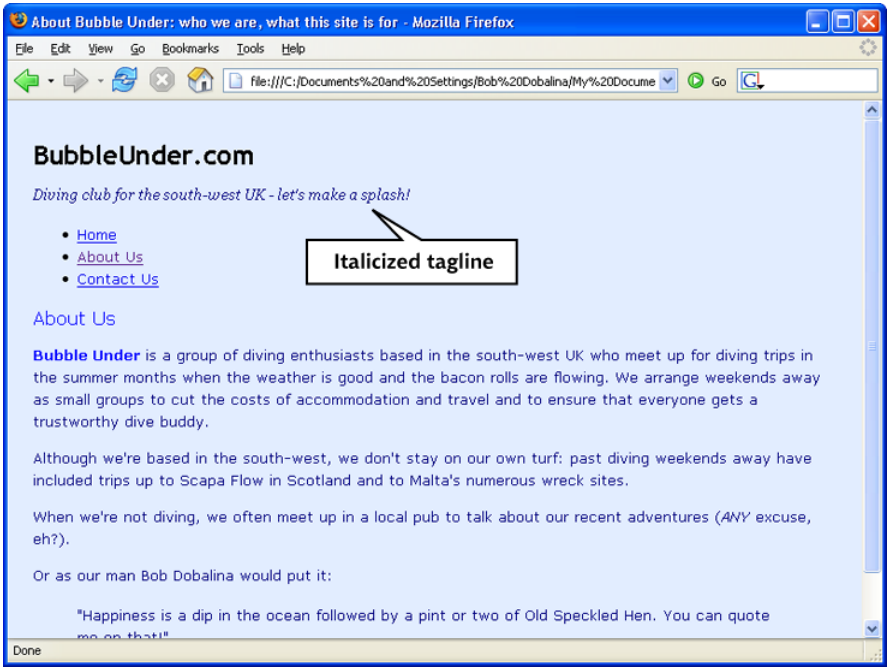
- ❑ Open the CSS file for editing, and add the following after the first paragraph rule:

  File: **style1.css (excerpt)**

  ```
  #tagline p {
    font-style: italic;
    font-family: Georgia, Times, serif;
  }
  ```

- ❑ Save the file, then refresh the About Us page (or any of the three, for that matter) in your browser. Your page should now look something like the one shown in Figure 3.11.

Figure 3.11. The tag line appearing in italics



What's happening here? Perhaps a CSS-to-English translation is required. This CSS rule means, "For any paragraph element that occurs inside an element that has an `id` of `tagline`, set the text to italics and the font to Georgia, Times, or some other serif font if you don't have either of those."

Getting a Positive ID

The `#` notation in the CSS refers to an element with a specific `id` attribute—in this case, `tagline`. We'll learn more about selecting `id`s and manipulating them in subsequent chapters.

Contextual Selectors

`#tagline p` is known as a **contextual selector**. Here are some other examples (with their English translations):

- ```
  #navigation a {
    text-decoration: none;
  }
  ```

  Translation: for any link found inside the navigation area (an element with an `id` of `navigation`), remove any decoration on that text; in other words, remove the underline (any other links on the page will remain underlined).

- ```
  #footer p {
    line-height: 150%;
  }
  ```

  Translation: set the vertical height between lines of text contained in paragraphs inside the footer area (e.g. a `div` element with an `id` of `footer`) to 150%. This would override the browser default of 100%, or other line-height values that might be set, for example, on the body.

- ```
  h1 strong {
    color: red;
  }
  ```

  Translation: for any text inside a level one heading that's marked up as `strong`, set the color to red (any other instance of `strong` on the page will not be set to red).

- ```
  h2 a {
    text-decoration: none;
  }
  ```

  Translation: don't underline the text of any link inside a level two heading (the default setting underlines all links, so any other links on the page will remain underlined).

Grouping Styles

If you want to apply the same style to different elements on a web page, you don't have to repeat yourself. For example, let's say that you want to set heading levels one through three in yellow text with a black background. Perhaps you'd do this:

```
h1 {
  color: yellow;
  background-color: black;
}
```

```
h2 {
  color: yellow;
  background-color: black;
}

h3 {
  color: yellow;
  background-color: black;
}
```

That's very messy, and once you have a lot of styles on the page, it gets even more difficult to maintain. Wouldn't it be great if you could reduce some of that work? You can! Here's how:

```
h1, h2, h3 {
  color: yellow;
  background-color: black;
}
```

Translation: if the element is a level one heading, a level two heading, or a level three heading, set the text to yellow and the background to black.

Comma = "Or"

Think of the commas in the selector above as the word "or."

Let's try grouping some styles in our project site. We don't have any `h3` headings yet, but they're coming:

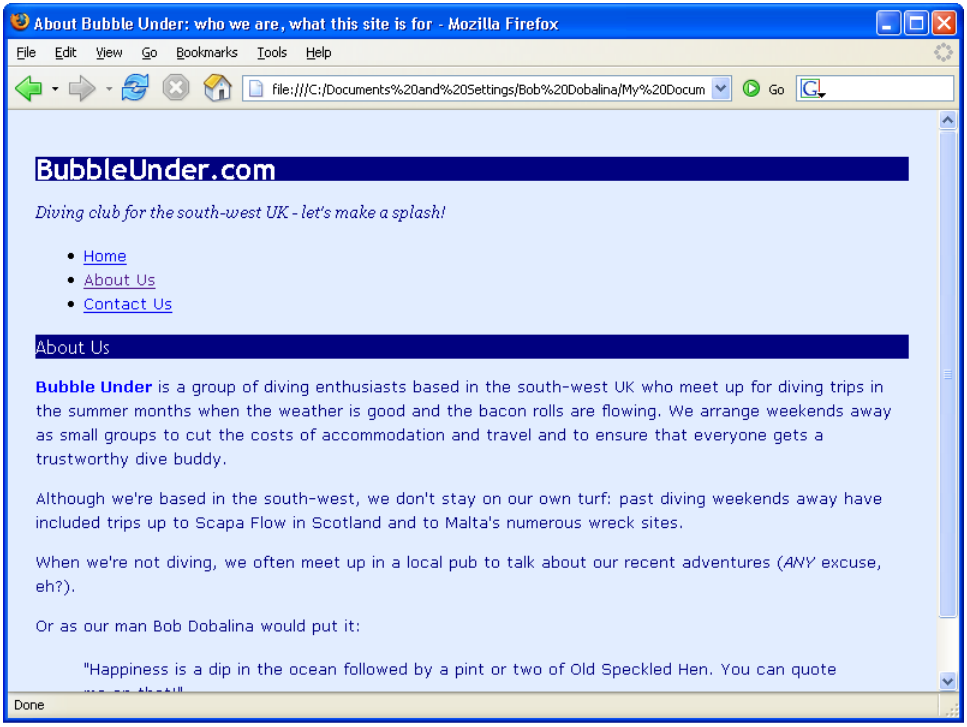
- ❑ Edit your CSS file (`style1.css`) by adding the following to the bottom of it:

File: **`style1.css` (excerpt)**

```
h1, h2, h3 {
  font-family: "Trebuchet MS", Helvetica, Arial, sans-serif;
  background-color: navy;
  color: white;
}
```

- ❑ Save the file, then refresh the "About Us" page in your browser (assuming it's still open from your last exercise). You should be looking at a page like the one shown in Figure 3.12.

Figure 3.12. Displaying the changed heading styles



That CSS really does kill several birds with one stone (and I said no animals would be harmed! Apologies to ornithologists). Now, not only do you have the convenience of being able to style many pages from one central location (your CSS file), but you have the added convenience of being able to style many elements in one go. Your CSS file becomes easier to manage and—a nice little side-benefit—smaller, and therefore quicker to download.

But something interesting is happening in our CSS file: it appears that we may have a conflict in our rules. Or have we?

Which Rule Wins?

When we added the grouped declaration for the headings, we changed some styles that we'd set previously. A look at the source shows that the level two heading, `h2`, has been set to be blue *and* white in different places in our style sheet:

File: **style1.css (excerpt)**

```
h2 {
  color: blue;
  font-size: medium;
  font-weight: normal;
}

em {
  text-transform: uppercase;
}

h1, h2, h3 {
  font-family: "Trebuchet MS", Helvetica, Arial, sans-serif;
  background-color: navy;
  color: white;
}
```

Because the declaration specifying that the `h2` should be white comes later, it has overridden the earlier one. It doesn't matter if you've defined an `h2` to be blue 100 times through your style sheet; if the last definition says it should be white, then white it will be!

Filenames for your Style Sheets

Although we've been working with `style1.css` for some time, you may be wondering why we named the file this way. The name is deliberate. You might want to add another style to your web site at a later date, and numbering is a basic way to keep track of the styles you can apply to a site.

You might be thinking, "Why not name it something like `marine.css` because it uses marine colors, references to under-sea animals, and so on?" That's a fair question, but the important thing to note about CSS is that you can always change the styles later, and your naming convention might, at a later date, bear no relevance to the styles a file contains. For example, you can edit `marine.css` such that all the colors in your web site are changed to ochres, browns, and sandy yellows. This ability to change the web site's design in one action is the whole point of CSS! With the new design, your web site might have an earthy/desert feel to it, yet you could still have 200 or more pages referring to a style sheet by the filename of `marine.css`. Something's not right there, is it? This is why I've chosen an abstract name for the CSS file, and I recommend that you do the same for the web sites you develop.

Recapping our Progress

Time for another breather. What have we learnt? Well, we've learned some more styles that you can apply in CSS, we've seen how you can style certain elements depending on their context, and more recently, we've discussed how you can group elements that need to be styled in the same way. There's one thing that we've touched on only briefly, yet it demands more attention because it's so fundamental to the way the Web functions. That topic is links.

Styling Links

Links are everywhere on the Web: they truly are the basis of everything you see online. Nowadays, we're used to seeing highly decorative web pages adorned by a wealth of different images and features. Take a step back in time, though, and you'll find that the World Wide Web was little more than a collection of linked documents. Go back to the earliest browsers and you'll find that those links were underlined, which remains the case today. By default, a browser uses the following color scheme for links:

blue an unvisited link

purple a link to a web page that you've previously visited

red an active link (one you're clicking on, but, for whatever reason, the next page hasn't quite appeared; we don't see this much in these days of widespread broadband usage, because pages load much faster than they used to!)

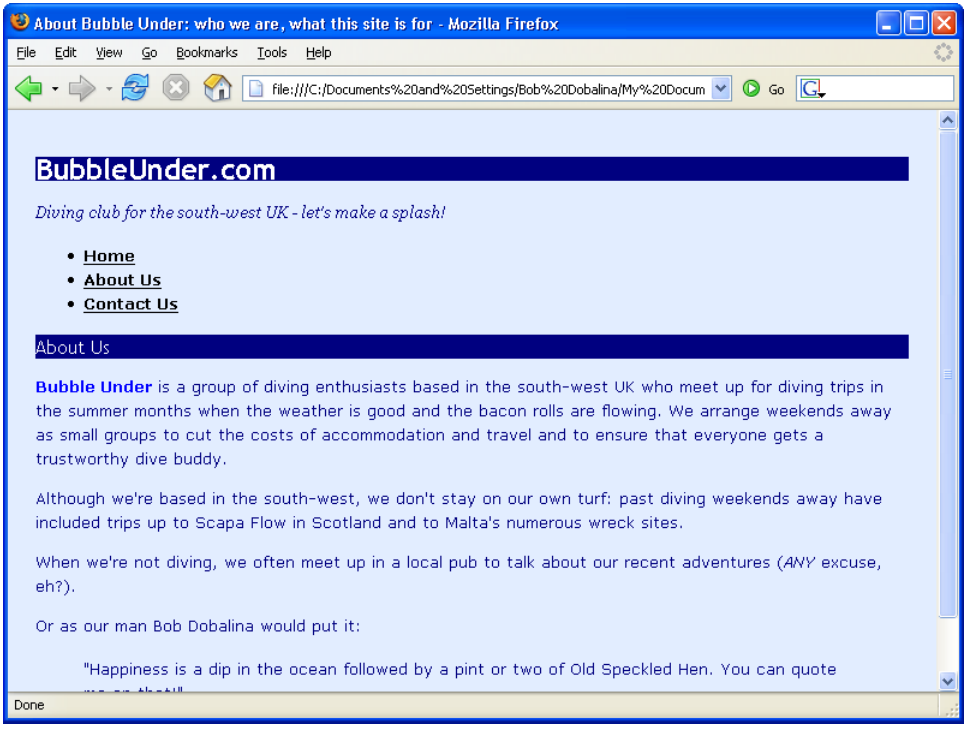
This color scheme isn't to everyone's taste, but it's what we're stuck with now. At least, it's what we *would* be stuck with if we couldn't use CSS to redefine those colors.

At its most basic, a CSS style for links might look like this:

```
a {
  font-weight: bold;
  color: black;
}
```

Now, instead of being blue and having a normal font weight, your links appear in bold, black type. Try adding that to your `style1.css` file; save it and see how it affects your web pages—Figure 3.13 illustrates.

Figure 3.13. Styling all the links in our navigation to bold and black



Link States

As I mentioned previously, there are different types of links (unvisited, visited, active) that you'll come across on a web page. There's one other state that I haven't mentioned, but it's one with which you're probably familiar: the **hover** state (which occurs when you pass your cursor over the link). In CSS, you can change the styling of all of these link states using something that sounds complicated but is really fairly straightforward: **pseudo-classes**. Here is some CSS that shows the color/style scheme for the different link states:

```
a {
  font-weight: bold;
}

a:link {
```

```
  color: black;
}

a:visited {
  color: gray;
}

a:hover {
  text-decoration: none;
  color: white;
  background-color: navy;
}

a:active {
  color: aqua;
  background-color: navy;
}
```

The different states are addressed within the CSS through the use of the `a` element selector, and by applying (with the aid of a colon) the pseudo-classes of `link`, `visited`, `hover`, and `active`.

Getting your Link States in Order

Browsers usually aren't fussy about the order in which you specify rules in your CSS file, but links should always be specified in the order shown above: `link`, `visited`, `hover`, and `active`. Try to remember the letters LVHA. The more bitter of us might find it easier to remember this mnemonic with the phrase, "Love? Ha!" We can thank Jeffrey Zeldman for that little gem.[5]

Let's change the styles for different link states in our project site:

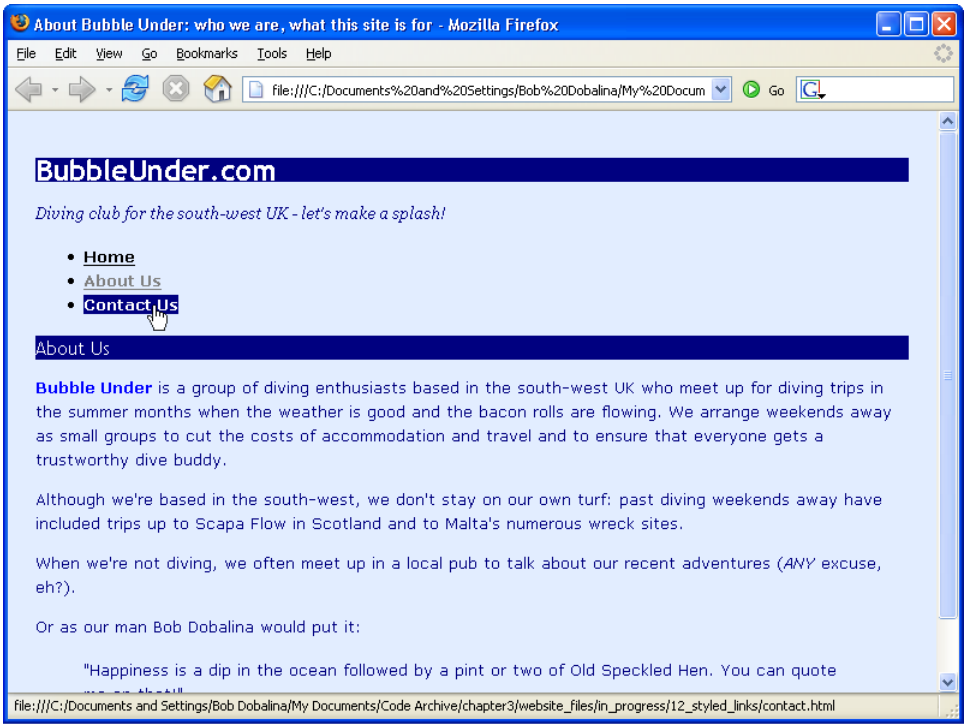
- ❑ If it's not already open, open the project site's CSS file (`style1.css`), and add the above CSS at the bottom of the file.
- ❑ Save the CSS file.
- ❑ Open any of the three web pages in your browser (or hit Reload) to see how the styled links display.

Figure 3.14 shows the three different link states: the home link is unvisited, the link to the About Us page shows that it has been visited previously (shown in

[5] *Designing With Web Standards*, Jeffrey Zeldman, New Riders.

gray), and the link to the Contact Us page is being hovered over by the user's cursor.

Figure 3.14. Styling three different link states using CSS



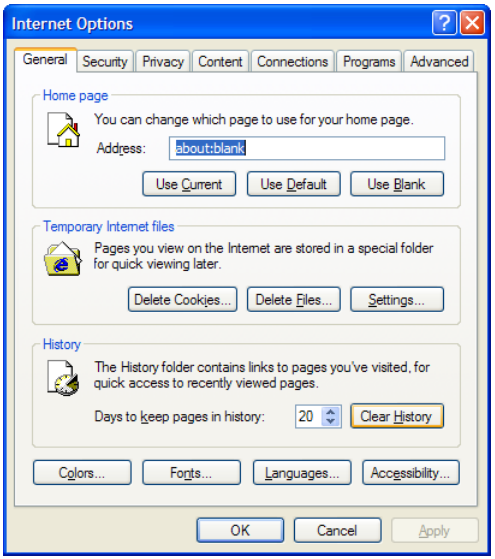
Feel free to experiment in the CSS file with the different foreground and background colors, and other text formatting styles that were detailed in the table earlier in this chapter.

Clearing your History

Your browser automatically stores a certain amount of your browsing history, and uses this information to decide whether a link has been visited or not (and, hence, how the link should be displayed). If you're building a site and testing links, you might want to check how an unvisited link looks but, because of your browsing history, they may all show as having been visited. This is almost certainly the case with our three-page project site—the links in your navigation list are probably all gray. To reset this, you can clear your browser's history. In IE, select Tools > Internet Options. You'll see a button

that reads Clear History, as shown in Figure 3.15; click it, then reload the web page. Any links you may have visited will now appear as unvisited.

Figure 3.15. Clearing the history in IE displays unvisited link styles again



Other browsers have similar options, which may be found in locations such as Tools > Options or Preferences > Privacy. I won't list all the different methods for deleting your history from various browsers here, but if you rummage around, you should be able to find them without too much difficulty.

Class Selectors

To date, we've discussed the ways in which we can style various elements, such as paragraphs and headings; we've also seen how we can style elements in specific areas of the page using the `id` attribute. However, implementing broad-brush styles, such as coloring the text in all `p` elements navy, is very much a blanket approach to design. What if you want some of those paragraphs (or any elements, for that matter) to look a little different than the rest? **Class selectors** are the answer.

A class selector lets you define a style that can be used over and over again to style many different elements. So, for example, let's say you wanted to make some parts of your text stand out—to make them look slightly more appealing or fun than other parts of the document. You could do so in your CSS like this:

```
.fun {
  color: #339999;
  font-family: Georgia, Times, serif;
  letter-spacing: 0.05em;
}
```

Here, we've created a style rule for a class called "fun." The fact that it's a class selector is denoted by the period at the beginning of the class name. We've slipped another property into this rule: `letter-spacing` defines the space between each of the letters. We've set a spacing of `0.05em` here. `1em` is the height of the M character in any font, so `0.05em` is 5% of that height. It doesn't sound like much of a difference, but when it comes to typography, subtle changes are usually more effective than extreme ones.

In order to make use of the style, all you need to do is add the `class="fun"` attribute to an element:

```
<p class="fun">A man walks into a bar; you would've thought he'd
    see it coming!</p>
```

Let's apply some classes to our project site. First, we'll need to add the style rule shown above to the style sheet we're working on:

- ❑ Open `style1.css` and add the CSS from the above block to the bottom of that file.
- ❑ Save `style1.css`, then open `about.html`.
- ❑ Find the paragraph that's contained inside the `blockquote` element.
- ❑ Add the `class="fun"` attribute to the paragraph's opening tag.

This is how your markup should look right now:

File: **about.html (excerpt)**

```
<blockquote>
  <p class="fun">"Happiness is a dip in the ocean followed by a
      pint or two of Old Speckled Hen. You can quote me on
```

```
      that!"</p>
</blockquote>
```

Note that the `class` attribute was applied at the paragraph level. If there were a few paragraphs in our man Bob's quotation, you'd end up with something like this:

```
<blockquote>
  <p class="fun">"Happiness is a dip in the ocean followed by a
      pint or two of Old Speckled Hen. You can quote me
      on that!</p>
  <p class="fun">"Join us for a weekend away at some of our
      favorite dive spots and you'll soon be making new
      friends.</p>
  <p class="fun">"Anyway, about time I got on with some
      <em>proper</em> work!"</p>
</blockquote>
```

There's a lot of repetition in there. Surely there's a tidier way to apply this style? There sure is!

```
<blockquote class="fun">
  <p>"Happiness is a dip in the ocean followed by a pint or two of
      Old Speckled Hen. You can quote me on that!</p>
  <p>"Join us for a weekend away at some of our favorite dive
      spots and you'll soon be making new friends.</p>
  <p>"Anyway, about time I got on with some <em>proper</em>
      work!"</p>
</blockquote>
```

In this example, we apply that `class` of `fun` to the `blockquote` element, so everything contained in that element inherits the style of the parent container. This saves us from having to apply these different classes all over our pages (an affliction that has become known as class-itis—a not-too-distant relation of div-itis, which we discussed in Chapter 2).

`class` vs `id`

So far, we've looked at both `class` selectors (which involve periods) and `id` selectors (which involve pound signs). Are you confused by them? It's true that these selectors are similar, but there is one important difference: *a specific id can only be applied to one XHTML element*. So, for example, on any web page, there can only be one element with an `id` of `mainnavigation`, and only one with an `id` of `header`. A class, on the other hand, can appear as many times as required.

Limiting Classes to Specific Elements

Imagine you want to italicise any `blockquote` element that has a `class` attribute with the value `fun`, but not other elements with that class value. Think it sounds tricky? Not with CSS!

```
.fun {
  font-family: Georgia, Times, serif;
  color: #339999;
  letter-spacing: 0.05em;
}

blockquote.fun {
  font-style: italic;
}
```

Now, any text inside a pair of `<blockquote class="fun">` and `</blockquote>` tags will appear in italics.

By prefixing our normal class selector with an element name, we're telling the browser to apply the following declarations to that element-and-class combination only. It's as simple as *`element.class`*, but make sure you don't leave any spaces!

Specifically Speaking

Those with an eagle eye will have noticed that not all of the `fun` styles in the previous example are actually applied to the quotation. The `font-family` and `letter-spacing` declarations take effect, but the color change does not! The reason for this can be explained with the concept of **specificity**.

Specificity simply means the rule that is the most specific is the one that is applied. "Most specific" really means "most deeply-nested in the document's structure." In our markup, the `p` element appears inside the `blockquote` element, so any style applied to the `p` is deemed to be the more specific of the two and, therefore, will always win out. We have such a rule in our project site—the one that states that all paragraphs should be navy-colored—so this is the one that takes effect:

File: **`style1.css` (excerpt)**

```
p {
  color: navy;
}
```

Note that, unlike the conflicting rules we encountered in the section called "Which Rule Wins?", this battle between style rules has no relation to the order in which they appear in the style sheet.

Specificity can get confusing, so don't lose too much sleep over it—for now, it's enough just to be aware of the concept, as this may be the reason why one of your styles doesn't take effect when you're convinced it should. Specificity is covered in greater depth in the SitePoint book *HTML Utopia: Designing Without Tables Using CSS*,[6] if you'd like to explore it further.

Styling Partial Text Using span

So, a class can be applied in many different places—perhaps to a specific paragraph, or to a block of several paragraphs contained in a `blockquote`, or to a `div` that holds many different types of content. But what would you do if you wanted to apply the style to a very small section of text—maybe just a couple of words, or even just a couple of letters, within a paragraph? For this, once again, you can use the `span` element.

Earlier in this chapter I showed how you could use the `span` element in conjunction with inline styles to pick out and style specific words within a paragraph. The exact same technique can be used with classes: we simply place an opening `<span>` tag at the point at which we want the styling to begin, and the closing `</span>` tag at the point at which the styling should end. The advantage of this technique over the inline style demonstrated earlier is, of course, that with this technique, the style is defined in a single location, so you could potentially add the "fun" class to many different elements on many different pages with a minimum of hassle. When you decide that you want to have a different kind of fun (so to speak), you need only change your style sheet (`style1.css`) for that new style to be reflected across your site.

```
<p><span class="fun">Bubble Under</span> is a group of diving
    enthusiasts based in the south-west UK who meet up for diving
    trips in the summer months when the weather is good and the
    bacon rolls are flowing. We arrange weekends away as small
    groups to cut the costs of accommodation and travel and to
    ensure that everyone gets a trustworthy dive buddy.</p>
```

Try applying the span element to your "about" page as suggested in the above code. If you save the changes and check them in your browser (remember to hit Reload), your page should look like the one shown in Figure 3.16.

[6] http://sitepoint.com/books/css1/

Figure 3.16. Applying the "fun" class to two specific words

Don't Throw (Needless) spans into the Works

The `span` element is nearly always used with a class attribute. There is not normally a good reason to apply a `span` element to your XHTML on its own, although you may see some web sites that do so.

Before you apply a `span` to any given element on your web page, take a moment to think about whether there's another element that's better suited to the task. For example, you should not use something like this:

```
<p>Do it <span class="shouty">now</span>!</p>
```

A more appropriate choice would be to use the `strong` element:

```
<p>Do it <strong>now</strong>!</p>
```

Think of the meaning of what you're writing, and aim for an XHTML element that suits the purpose. Other examples might be `em`, `cite`, and `blockquote`.

Summary

It's been another busy chapter, but my, how our site's blossoming! A chapter or two ago, we hadn't even built a web page, but now we're at the stage where we know how to apply a (virtual) lick of paint to any type of XHTML element on a page, to a specific section of a web page depending on its `id`, or to arbitrary portions of a page—sometimes in several different places—using class selectors.

The web site is starting to look a little more colorful, but the layout is still fairly basic. In the next chapter, we'll look at how it's possible to change the layout of elements on a page—their position, shape, size, and more—using CSS. Styling text? Been there, done that. Let's move to the next level!

10 Pimp my Site: Cool Stuff you can Add for Free

Your web site looks great, everything seems to be going tickety-boo, and maybe you've even got regular news updates happening thanks to a group of budding bloggers who contribute regularly. Your work's done, right? Well, no. There's always something else to do!

When you first set up your web site, you probably had a good idea of the audience you were building it for (at least, I hope you did), and you may well have catered admirably to that audience. But within a couple of weeks of launching your web site and promoting it to the world (using some of the suggestions I made in Chapter 8), you'll start to receive emails from people you don't know asking questions about the site that you really hadn't expected:

> "Can you tell me where I can get my air regulators serviced in North Devon?"
>
> "I can't find details of your training courses—do you offer any?"
>
> "My name is Abdul Akinbobola and I am the son of the recently deposed president of Burkino Faso and…"

Okay, so that last message has nothing to do with your web site, but trust me, you'll certainly get emails like this! The point I'm making is that no matter what sort of planning you've undertaken, people beyond your expected audience will

find your web site one way or the other, and you'll likely need to be able to cater to them, too. This is where you should consider some add-ons to your site—extras that will:

- let you *discover how people are arriving at your web site* (e.g., through a search on Google, or via a referral—or link—from another web site in which you've promoted your own site)
- reveal *which search terms people used* to get to your web site, and provide some statistics about the most common ones
- let the visitor *search the contents of your web site* (rather than click around the navigation in the hope of finding what they need)
- allow the visitor to *search a group of related web sites* from the comfort of your web site
- provide a way for you to *manage a list of your favorite web sites* related to the topic in your own web site, and provide them as a links resource for others
- let your visitors become part of your web site community by *providing a discussion forum*

All of these goals can be achieved using free services, and in this chapter, I'm going to provide step-by-step instructions to help you add these services and truly pimp your site![1]

Getting the Low-down on your Visitors

How can you be absolutely sure that what you've got on your web site is the right content for your audience? Well, the truth is that you can't know for sure—everyone's different, after all, and each person's needs are unique. However, you can get some indication about whether your web site is serving the audience's needs through some simple statistics.

Some hosting companies will provide statistics software as part of your hosting package, so be sure to check. If your package includes a statistics service, you can safely skip over this next section—instead, make use of the tools your host has

[1] For those who don't get the reference, Pimp My Site is my little pun based on the MTV show Pimp My Ride, [http://www.mtv.com/onair/dyn/pimp_my_ride/series.jhtml] in which old, neglected cars are renovated (or "pimped") for their owners.

already provided. However, most free hosting services, and many of the cheaper hosting plans, will not provide statistics for you. It's up to us to fill that gap.

Choosing a Statistics Service

As with a number of services I've mentioned elsewhere in this book, there are two ways that you could introduce a statistics service to your site:

- You could install and configure a statistics service on the web server that hosts your site. The web server keeps detailed records of every visit to your web site: it records the time of the visit, which pages were viewed, which browsers were used, how visitors found the site, and much, much more. There are many programs you can install onto your web server that will produce easy-to-read graphs based on this data. Installing this software is no easy feat, though, and is not really something I'd recommend for beginners.

- Thankfully, the second option is much easier—you can sign up for a third-party solution that collects and stores the data on your behalf. All you're required to do is add a link to an image or script file (hosted by the service provider) into your web pages.

Many third-party statistics services are available, but to narrow things down a little, I advise you to look for one that offers the following features:

list of referring web sites (recent referrers and totals)
: This information will tell you how your visitors found your web site.

number of visitors
: You should be able to view a count of the numbers of visitors your site receives each day and each month, as well as the total number of visitors who have stopped by since the site launched.

information about your visitors' computer setups
: This data will tell you whether your visitors are using PCs or Macs, which browsers they're using, and so on.

Any information beyond that is probably overkill for a small-scale web site (as you can probably imagine, too many statistics can end up muddying the waters—there's a lot to be said for simplicity!). Some of the hosted services you

might want to consider using include StatCounter,[2] Extreme Tracking,[3] and AdFreeStats.[4]

In the past, I've always used the Extreme Tracking service—it's easy to set up and does its job. However, I'm going to break with my own habits and show you the process for setting up an account with StatCounter. Why? Because the reporting looks good, there's lots of additional help on each report page that will lead you to discover more, and—a real selling point for me—the service generates standards-based markup for its visitor tracking code (most services are not all that thoughtful, and it would be a shame to undo the good work you've completed on your web site so far by introducing invalid markup).

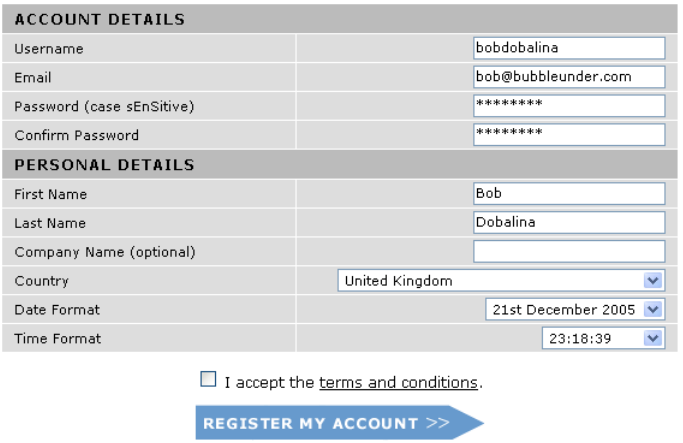
Registering an Account with StatCounter

There are just a few forms to fill in—soon, you'll be ready to add a statistics service to your site! Here's what you need to do:

- ❑ Head over to StatCounter's home page[5] and click on the Register Now link.
- ❑ Fill in your account and personal details into the registration page shown in Figure 10.1. Be sure to make your username and password easy to remember.

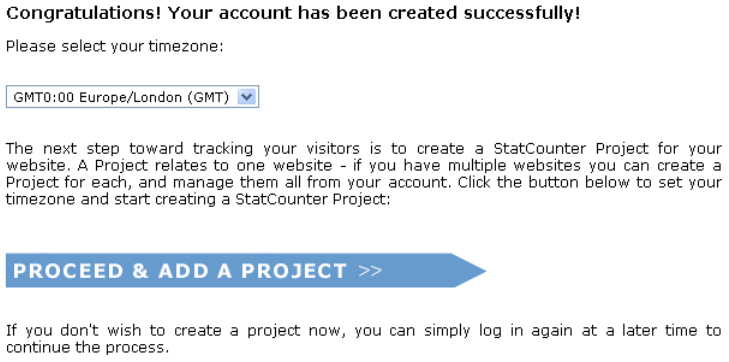
[2] http://www.statcounter.com/
[3] http://www.extreme-dm.com/tracking/
[4] http://www.addfreestats.com/
[5] http://www.statcounter.com/

Figure 10.1. StatCounter's registration page



- You'll be asked to select your time zone before proceeding to the project setup process—Figure 10.2 shows how it's done. A StatCounter Project is a single web site whose statistics are collected by StatCounter. You can collect statistics for multiple web sites, but each web site must have its own separate project.

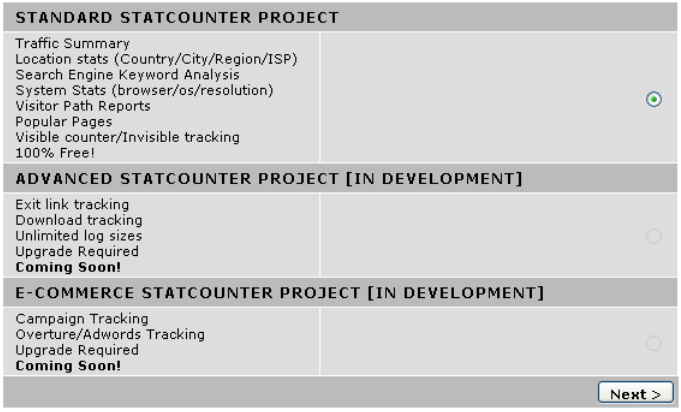
Figure 10.2. Selecting a time zone



- The first step in setting up a project is to select the type of project you want to create. At the time of writing, only one project type is available—the Standard StatCounter Project—but that's all we need anyway. Select the

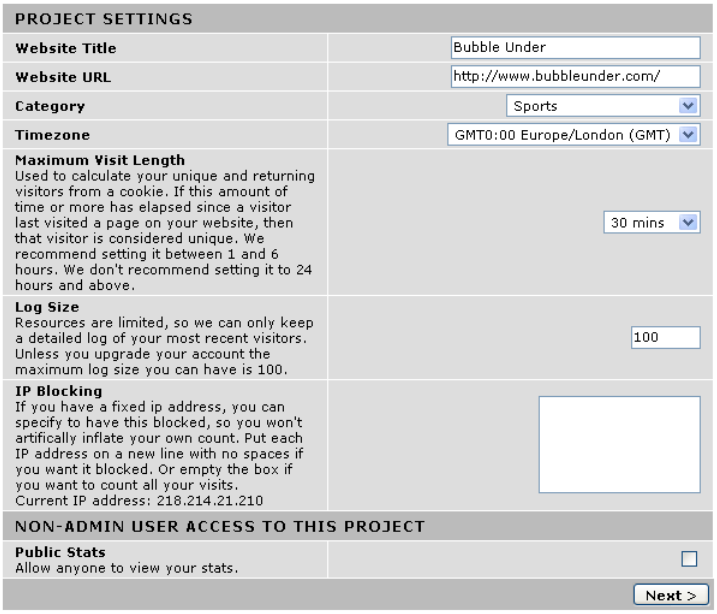
Standard StatCounter Project, as shown in Figure 10.3, and click the Next > button.

Figure 10.3. Only a standard StatCounter project is available



- On the project setup page illustrated in Figure 10.4, you'll need to provide details such as the web site's name, the web address (URL), the site's category, and so on. You'll also be asked whether you want these statistics to be available to others. The choice is yours, but personally, I'd choose to hide the statistics, just because the link tends to look a little ugly on the page. Fill in the necessary details, then press the Next > button.

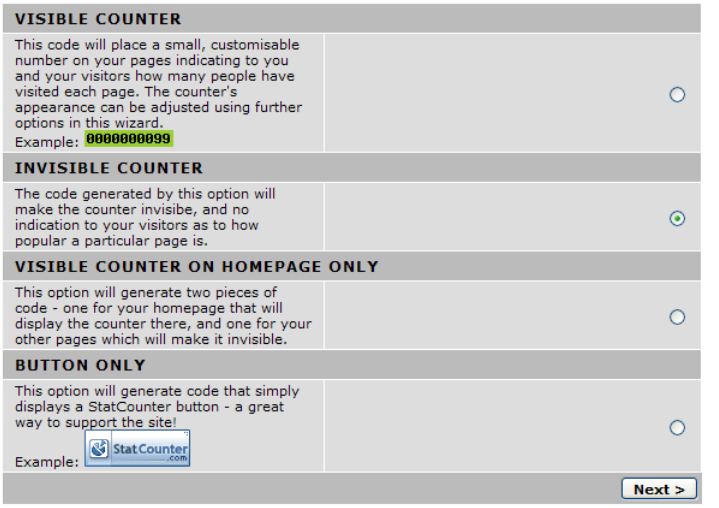
Figure 10.4. Selecting your project settings



- You'll receive a confirmation message to tell you that your project has been created. The next step is to get the XHTML that you need to put onto your web site. Click Configure & Install Code to continue.

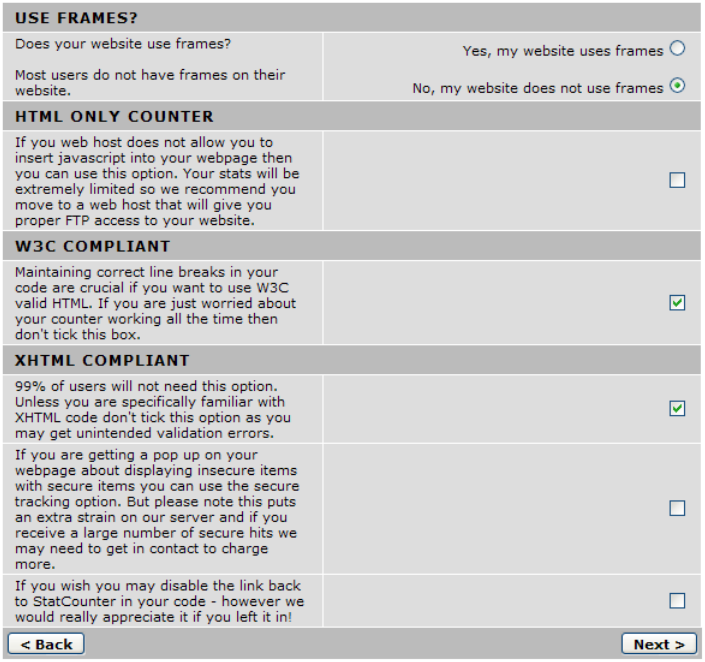
- As Figure 10.5 indicates, you'll be asked to choose a display style for the counter—do you want people to see how many visitors you've had, or do you want to keep that information hidden? Once again, I'd choose to hide it (showing your visitor numbers is *so* 1998!). Besides, until your web site has been running for a while, the visitor figures may be quite low, and there's no need to advertise the fact. Make your choice, then continue once again.

Figure 10.5. Selecting the counter type



- Next, you're asked about the make-up of your web site through the form shown in Figure 10.6. Do you use frames? If you've followed my advice so far, the answer should be No, unless you're using a web forwarding service. This page also gives you the option of making the counter W3C- and XHTML-compliant. Ignore the overly fussy warnings about compatibility and check both options—we don't need to worry about these warnings because everything we've covered in this book follows W3C recommendations.

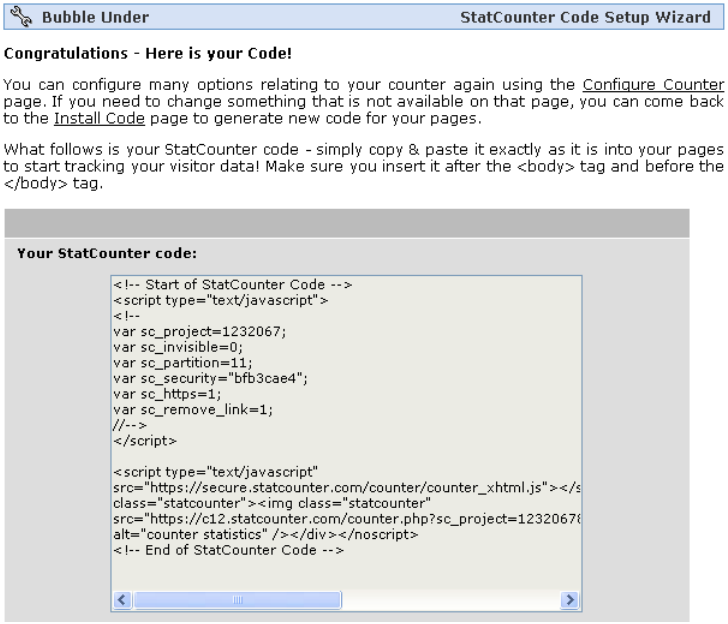
Figure 10.6. Settings relating to your web site's technical make-up



- The next page asks you if you use a (by which is meant a simple web page builder offered by the likes of AOL, or a program like Dreamweaver or FrontPage). Answer No and carry on.

- On the next page, depicted in Figure 10.7, you'll be given the code that you need to insert into your web pages so that StatCounter can keep track of your site's traffic.

Figure 10.7. The final markup is easily copied and pasted into your web pages



That's it for the sign-up process. Now all you need to do is put the generated code into your web pages.

Adding the Statistics Code to your Web Pages

Your statistics code should look something like this (though specific details relating to your account will differ):

```
<!-- Start of StatCounter Code -->
<script type="text/javascript">
<!--
var sc_project=1143403;
var sc_invisible=1;
var sc_partition=10;
var sc_security="bb4d77e9";
//-->
</script>
```

```
<script type="text/javascript"
    src="http://www.statcounter.com/counter/counter_xhtml.js">
</script>
<noscript>
  <div class="statcounter">
    <a class="statcounter" href="http://www.statcounter.com/">
      <img class="statcounter" src="http://c11.statcounter.com/cou
nter.php?sc_project=1143403&java=0&security=bb4d77e9&invisible=1"
alt="web statistics" />
    </a>
  </div>
</noscript>
<!-- End of StatCounter Code -->
```

To get yourself up and running, simply paste this code in your web page just before the closing `</body>` tag (because we've chosen a tracker that's invisible, we needn't concern ourselves with its position on the page). Here's how it looks in the context of the "About" page (which I've truncated somewhat):

File: **about.html (excerpt)**

```
    <p>
      Or as our man Bob Dobalina would put it:</p>
      <blockquote>
        <p class="fun">"Happiness is a dip in the ocean followed
            by a pint or two of Old Speckled Hen. You can quote
            me on that!"</p>
      </blockquote>
    </div> <!-- end of bodycontent div -->
    <!-- Start of StatCounter Code -->
    <script type="text/javascript">
      <!--
      var sc_project=1143403;
      var sc_invisible=1;
      var sc_partition=10;
      var sc_security="bb4d77e9";
      //-->
    </script>
    <script type="text/javascript"
        src="http://www.statcounter.com/counter/counter_xhtml.js">
    </script>
    <noscript>
      <div class="statcounter">
        <a class="statcounter" href="http://www.statcounter.com/">
          <img class="statcounter" src="http://c11.statcounter.com
/counter.php?sc_project=1143403&java=0&security=bb4d77e9&invisible
=1" alt="web statistics" />
```

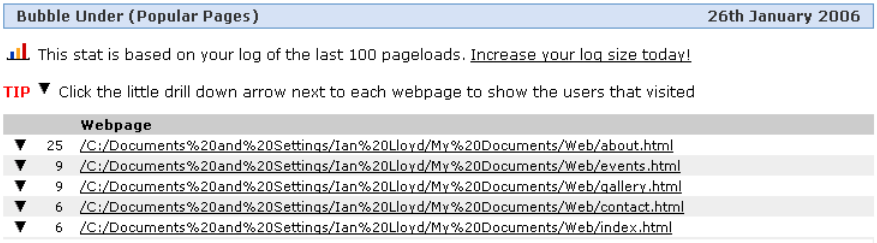
```
        </a>
      </div>
    </noscript>
    <!-- End of StatCounter Code -->
  </body>
</html>
```

Add the code to all of the pages in your web site, and save them all. If you've created a Blogger template, you'll need to update that with the statistics code, too.

Now, try moving around the site, clicking on the links in the navigation, and generally replicating the behavior of a "real" visitor to your site. Also, try refreshing one of the web pages a few times—what we're doing here is building up a bit of dummy traffic to test things out.[6]

Next, log in to the StatCounter web site (using the details you provided at registration). Look for the little icon that looks like a bar chart—this is your way in to the statistics display. Click the icon and you'll see a chart that shows a summary of statistics. Perhaps you'd like to see which are the most popular pages on your web site? Take a look at the navigation options and you'll see a link to "Popular Pages"—that looks like a sensible choice. The resulting display should look something like Figure 10.8.

Figure 10.8. Pages ranked by popularity



You can see from above that the system was able to track popular pages stored on my hard drive (a web site address would be prefixed with http://), and my

[6]Note that you must be online for this exercise to work. Although you are only making changes to files on your hard drive at this point, the way this statistics system—and most others of this nature—works is that it requests a file (usually a graphic) from the server, and appends to the filename a bunch of data that links the file to your account. That request is what causes the system to keep a record, so if you don't have an Internet connection, it won't work.

brief clicking around the site has revealed that the "About Us" page was my most popular choice.

There are many other useful statistics here—too many to go into in more detail, in fact. My advice is to sign up, apply the statistics code, and upload your amended pages to your web server. Then, simply leave it for a few days before logging back in to check the statistics. By that time you might have enough data to see some patterns beginning to form.

Getting a Bigger Log File

You can see in Figure 10.8 that the logs created by StatCounter are based on the last 100 page loads. On a popular web site, it won't take long to reach that number. This doesn't mean that, once you reach 100 page loads, the system stops reporting; instead, newer information effectively shunts the older information into oblivion. It is possible to increase the log size, but predictably, this will come at a cost. If you decide that the information you're getting is not rich enough, you might want either to try increasing the log size or trying out one of the other services I mentioned earlier—the process for setting them up will be almost identical to the process I explained here.

What to Look for—a Summary

The most revealing statistics that will be available to you through StatCounter include:

Exit pages: From which pages are people leaving your site to go somewhere else?

If you can see a pattern forming, you can focus on these problem pages and try to work out what it is that's "turning people off." Is the page too wordy? Is it loading too slowly?

Browser and System Stats: What tools do people use to access your web site?

Knowledge is power. If you learn that 99% of your visitors are using one kind of web browser, you might not spend quite so much time worrying about display issues for the other 1% of users. This kind of information helps you prioritize any bugs you might need to fix.

Came From: Through which web pages do visitors arrive at your site?

If another site has linked to your web site, and a user follows that link to your web site, that information will be recorded on the Came From page. It's good to be aware of other web sites that have linked to you (if for no other

reason than to give you an ego boost!), and why they've linked to you—it's easy enough to take a look at the "referring site" from these reports.[7]

A Search Tool for your Site

This one's a cinch! We'll have you set up in minutes. In times gone by, I would have suggested registering for a service such as Atomz,[8] then configuring it precisely to fit your needs. However, now I use Google almost exclusively and find the search results to be very good. If only it were possible to use Google to search just your web site… Hang on—you can! And it's a walk in the park to set up!

Here's the basic markup you'll need to get Google to provide search results based on the content of your web site only:

```
<!-- SiteSearch Google -->
<form method="get" action="http://www.google.com/search">
<label for="q">Search:</label>
<input id="q" name="q" size="20" maxlength="255" value=""
    type="text" />
<input name="domains" value="http://www.bubbleunder.com/"
    type="hidden" />
<input name="sitesearch" value="http://www.bubbleunder.com/"
    checked="checked" id="mysite" type="radio" />
<label for="mysite">Just this site</label>
<input name="sitesearch" value="" id="www" type="radio" />
<label for="www">WWW</label>
<input name="btnG" value="Go" type="submit" />
</form>
<!-- SiteSearch Google -->
```

All you need to do is change the parts in bold so that they match your web site's address. It's so easy!

Here's that same code implemented the Bubble Under web site (at least, on a portion of the events page):

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

[7] Sometimes, web site managers talk about "checking their referrer logs." This is what that term means—looking through the lists of sites who have sent traffic to your web site, including search engines (reviewing the search phrases users entered into search engines to find your site).
[8] http://www.atomz.com/

```
<head>
<title>Forthcoming club diving events and trips with Bubble
    Under</title>
<meta http-equiv="Content-Type"
    content="text/html; charset=utf-8" />
<link href="style1.css" rel="stylesheet" type="text/css"
    media="screen" />
</head>
<body>
<div id="header">
  <div id="sitebranding">
    <h1>BubbleUnder.com</h1>
  </div>
  <div id="tagline">
    <p>Diving club for the south-west UK - let's make a
        splash!</p>
  </div>
<!-- SiteSearch Google -->
<div id="search">
<form method="get" action="http://www.google.com/search">
<label for="q">Search:</label>
<input id="q" name="q" size="20" maxlength="255" value=""
    type="text" />
<input name="domains" value="http://www.bubbleunder.com/"
    type="hidden" />
<input name="sitesearch" value="http://www.bubbleunder.com/"
    checked="checked" id="mysite" type="radio" />
<label for="mysite">Just this site</label>
<input name="sitesearch" value="" id="www" type="radio" />
<label for="www">WWW</label>
<input name="btnG" value="Go" type="submit" />
</form>
</div>
<!-- SiteSearch Google -->
</div>
<!-- end of header div -->
...
```

Note that we need to position the search form in an appropriate location, and format the text somewhat. I've used CSS to achieve this, using absolute positioning to place the search box in the top, right-hand corner of the page. To do so, I wrapped a `div` around the form and gave it an `id`. That way, I have some way of referencing the form in the CSS, as shown below:

```
#search {
  position: absolute;
```

```
  top: 77px;
  right: 10px;
  font-size: x-small;
  font-weight: bold;
}
```

Figure 10.9 shows how the search box looks on the page itself.

Figure 10.9. The Bubble Under site with integrated search

Google Scoped Search Limitations

Using Google's service in this way is certainly easy, but you should be aware of this solution's limitations:

- Google will only show search results if it *knows* about your web site—and it will only know about it if you've submitted your web site's address[9] to

[9]You can notify Google of your web site's existence at http://www.google.com/addurl/.

Google in the past (and Google has indexed it), or Google has found your web site by following a link from some other site.

- The search results may not be completely up-to-date. If you make changes to your site, then upload those changes, Google may not recognize that a change has been made for days or even weeks—it really depends on when the search engine re-indexes your site.

- The search results cannot be customized. The results page will look like a standard Google search results page, but the linked search results will all be pages from your web site (aside from sponsored links). However, people are familiar with Google, so this view of the search results also has its benefits.

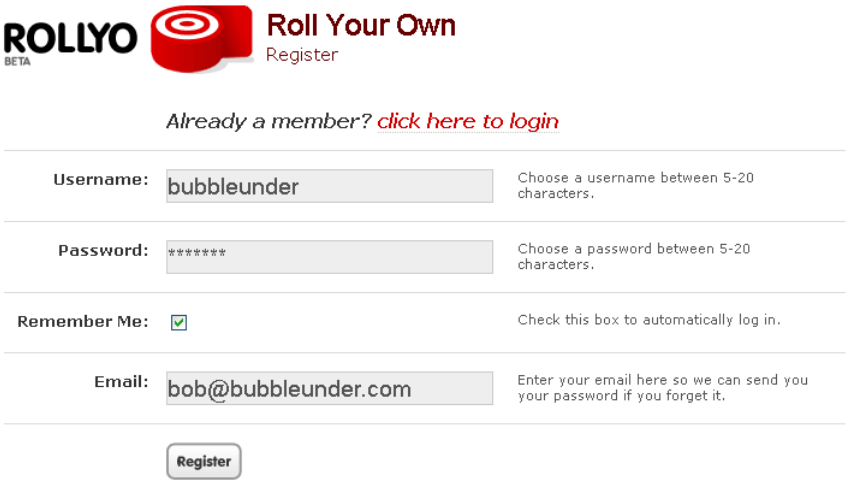
Searching By Genre

At the time of writing, a new service called Rollyo (a "roll-your-own" search engine) had just been launched. What this service allows you to do is to create a custom search interface—one that lets you pick and choose what web sites you want to search. So, if you're the caring, sharing type, why not try out Rollyo?

Note that these screenshots were taken just after the Beta launch, and as such, they may differ slightly from the final version.

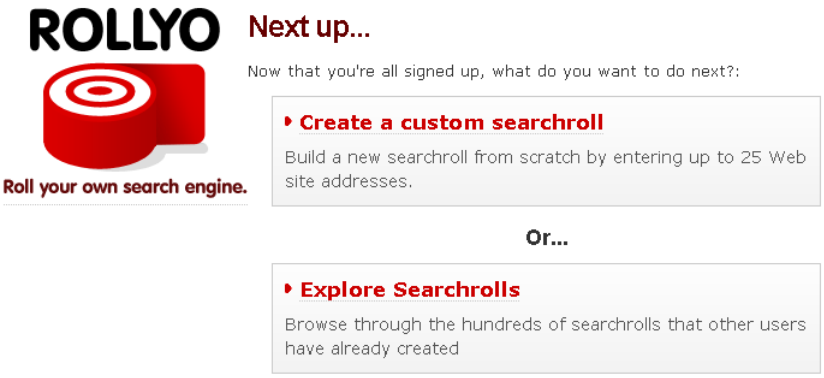
- Click on the Register link and complete the scant details requested of you by the registration page shown in Figure 10.10.

Figure 10.10. The Rollyo registration screen



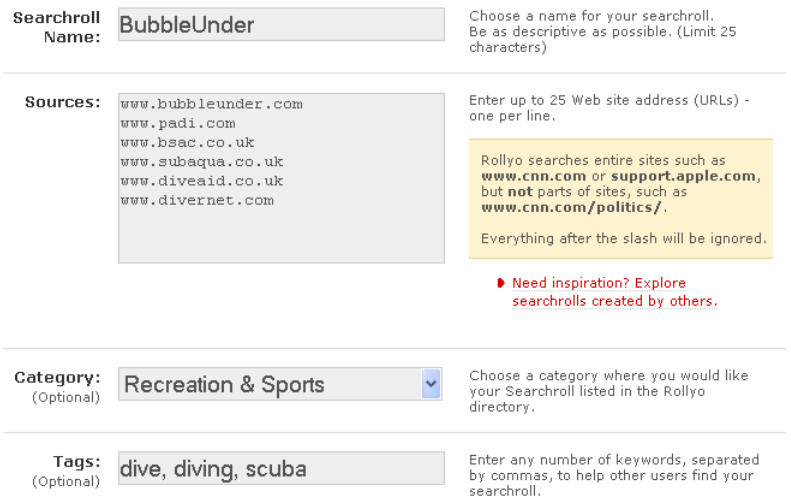
- Next, you'll be asked for profile information, and you'll see a big red arrow with the words, Skip this for now written on it. You know what to do!
- On the following page, select the Create a custom searchroll link, as illustrated in Figure 10.11.

Figure 10.11. Choose Create a custom searchroll



- Next, in the page shown in Figure 10.12, you're asked to provide names for your "Searchroll"—a list of web sites that you want to include—and also to identify "tags" (keywords that describe your search facility's purpose) that you'd like to assign to your searchroll.

Figure 10.12. Adding sites that you want to search



With that done, you can create your searchroll. At the time of writing, Rollyo had not yet provided a simple method for grabbing the source code required to place the search functionality on your own web site. However, it's possible to view the source of a web page, take what you need, then adapt it slightly. I've done just that to show how Rollyo could be used on any web site. Here's the source code for the Rollyo search I created for Bubble Under:

```
<!-- SiteSearch Rollyo -->
<div id="search">
  <form id="searchform" name="searchform"
      action="http://www.rollyo.com/search.html" method="get">
  <input type="input" name="q" value="" id="search-box" /> in
      <select id="searchmenu" name="sid">
<option value="6170">Bubble Under</option>
<option value="web">The web</option>
</select>
<input type="submit" value="Search" />
```

```
</form>
<!-- SiteSearch Rollyo -->
```

Figure 10.13 shows how the search functionality displays on the web page (I placed it in the absolutely positioned div, the same position we gave the Google search box in the earlier example).

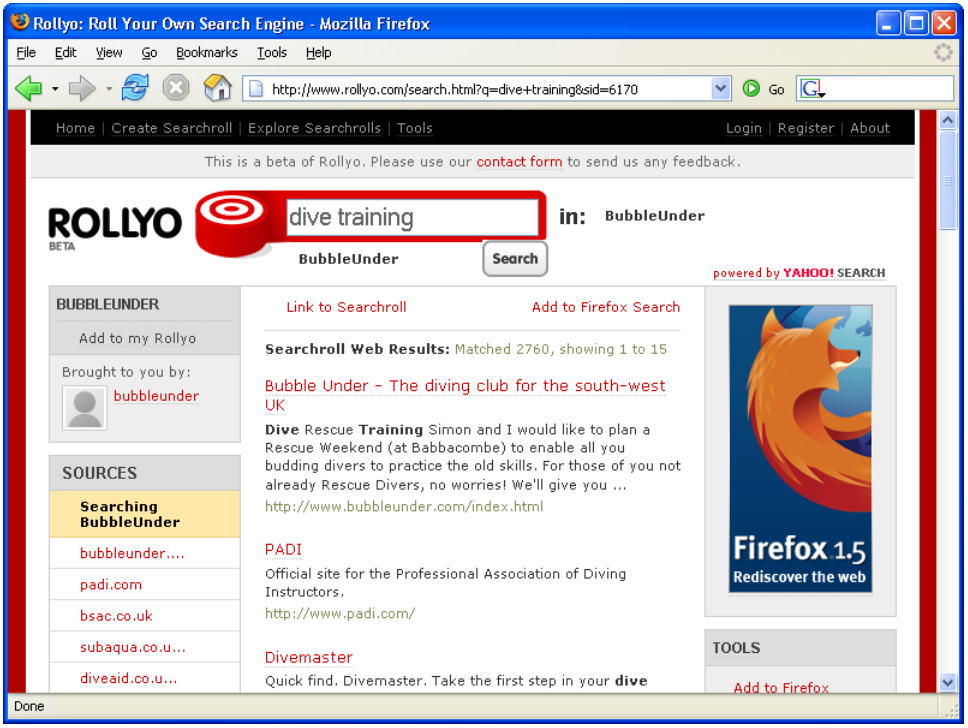
Figure 10.13. Adding a Rollyo search to the Bubble Under web site

The next step is to submit your site to Yahoo! for indexing,[10] if you haven't already. As with most search engines, it may take days or weeks before Yahoo! visits your site and adds your pages to its database. Your Rollyo searchroll will still work during this time, it just won't include your site in the results it displays.

Finally, Figure 10.14 depicts the search results displayed on the Rollyo web site.

[10] http://search.yahoo.com/info/submit.html

Figure 10.14. Displaying the search results on Rollyo



This is a novel way of adding search functionality to your site—but it's one that could potentially send people in another direction! However, if your web site is more of a fun venture, than focused on making cold hard cash, this could be the ethical way to go. Believe me when I say it'll be good for your karma—you'll see!

Caution: Contents May have Shifted in Transit

As I mentioned at the beginning of this section, Rollyo was new at the time of writing (in fact, it was still in "beta" phase—it hadn't been released to the world at large). As such, the service may have changed a little, or even a lot, by the time you read this—heaven forbid, it may even have disappeared altogether! The latter is unlikely (because it's a great service), but please be aware that you may need to adapt some of the steps above if Rollyo's service does change slightly.

Adding a Blogroll to your Web Site

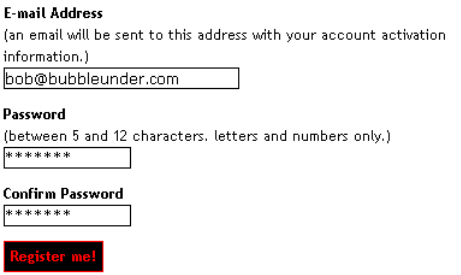
What's a **blogroll**? A blogroll[11] is a list of links that's included on your web site (or blog), and which is easily updated from a central point. A blogroll is often a better option than creating a specific page on your site just for links (which can easily turn stale), and provides you and your visitors with an easy route to fresh content on other web sites.

A range of tools are available for this purpose, but perhaps the simplest one to try is Blogrolling. Here's the process for setting up a Blogroll (once again, I've used the Bubble Under web site as an example).

Signing Up for a Blogroll

- ❑ Go to the Blogrolling[12] web site.
- ❑ Follow the link to "Create Account"—you'll be asked for an email address and password, as Figure 10.15 shows.

Figure 10.15. Setting up a Blogrolling account

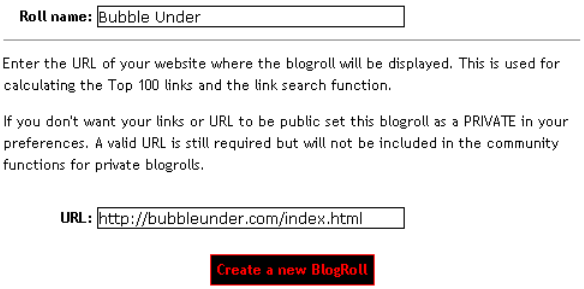


Once you've submitted the form, Blogrolling will send you an email. Open it and click on the activation link.

- ❑ Once you've activated your account, you can create a new Blogroll (in fact, you can create several different Blogrolls with just one account).

[11] http://en.wikipedia.org/wiki/Blogroll
[12] http://www.blogrolling.com/

Figure 10.16. Creating a new Blogroll



Simply enter a name and the web address of the page on which you'll be displaying the links, as shown in Figure 10.16.

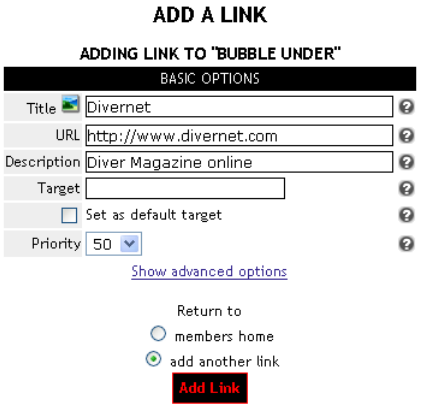
- Most of the tools you'll need to manage your blogroll links are available from the Blogrolling homepage.[13] Figure 10.17 shows the Blogrolling "control panel."

Figure 10.17. All your Blogrolling options are available here

- Select Add Links. You'll be presented with a small form that looks like the one in Figure 10.18.

[13] By "homepage," I mean the page that's linked to by the word Home in the left-hand navigation menu that you see when you're logged in to Blogrolling. I don't mean the Blogrolling web site's main homepage.

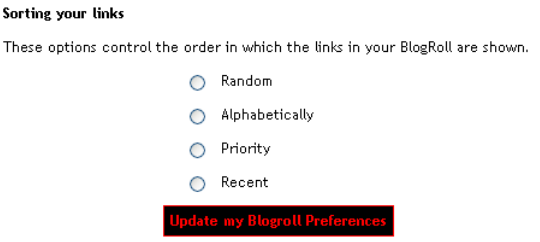
Figure 10.18. Adding a link to your Blogroll



You could keep on adding links to this list until you run out of interesting web site addresses, but bear in mind that if you add too many links, your web page is going to start to resemble Yahoo! gone mad.

- Once you've added links to your favorite web sites, you need to do just one more thing: determine the order in which you'd like the links to display on your web site. Select Preferences and scroll down to the section that says Sorting your links, as illustrated in Figure 10.19.

Figure 10.19. Setting the order in which your links appear on the page



- Now all you need to do is get the code required for this feature to function on your web site. Go back to the Control Panel's homepage and click on the Get Code link. You should see something like this:

```
<script language="javascript" type="text/javascript"
    src="http://rpc.blogrolling.com/display.php?r=ad7cab20d092c
2ee809d009e26ffcc11"></script>
```

Integrating the Blogroll with your Web Site

Usually, people add blogrolls to their sites' homepages, where it's often tucked away to one side. I'm going to do the same with the Bubble Under blogroll.

As with the other sections on your web site, it's a good idea to wrap the blogroll in a `div` with an `id`. This approach will allow you to reference the `div` in a style sheet, and makes it easy to find the blogroll when you look through the markup at a later date. Here's the markup for the Bubble Under home page (`index.html`), once the blogroll has been added:

File: **`index.html` (excerpt)**

```
<div id="navigation">
<ul>
  <li><a href="index.html">Home</a></li>
  <li><a href="about.html">About Us</a></li>
  <li><a href="events.html">Club Events</a></li>
  <li><a href="contact.html">Contact Us</a></li>
  <li><a href="gallery.html">Image Gallery</a></li>
  </ul>
</div>
<div id="bodycontent">
<div id="blogroll">
<h2>Recently updated Diving Sites</h2>
<script language="javascript" type="text/javascript"
      src="http://rpc.blogrolling.com/display.php?r=ad7cab20d092c2
ee809d009e26ffcc11"></script>
</div>
<h2>Welcome to our super-dooper Scuba site</h2>
<p><img src="divers-circle.jpg" alt="A circle of divers practice
    their skills" width="200" height="162" /></p>
<p>Glad you could drop in and share some air with us! You've
    passed your underwater navigation skills and successfully
    found your way to the start point - or in this case, our
    home page. </p>
```

Without any further intervention, your list may look a little on the dull side. Let's suppose you wanted to style your blogroll. This code will do the trick:

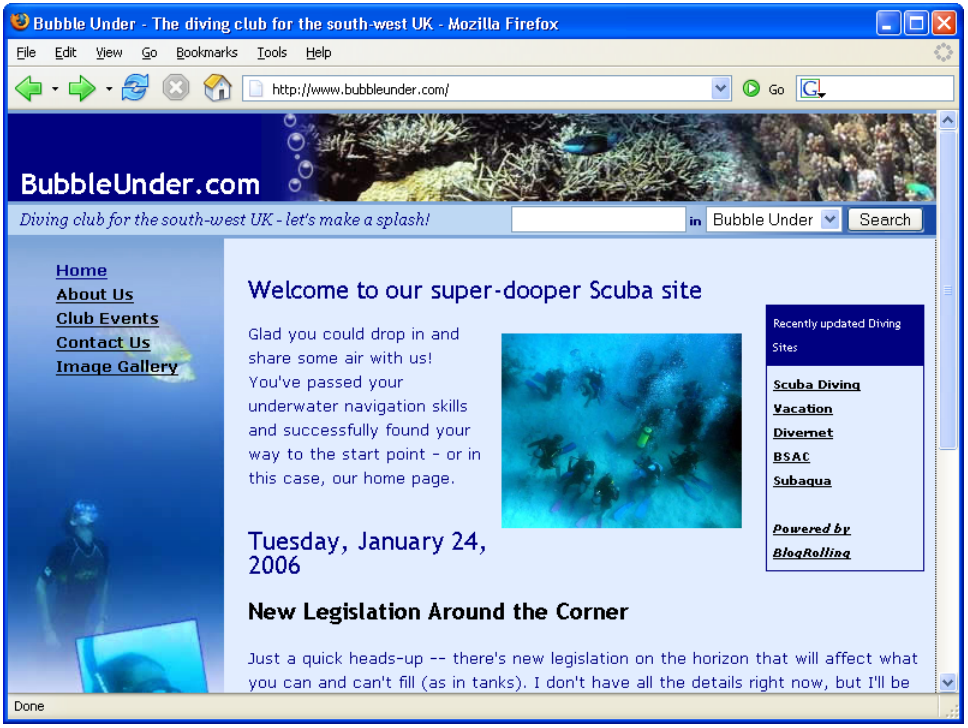
```
#blogroll {
  float: right;
  margin-top: 55px;
  margin-left: 10px;
  margin-bottom: 10px;
  margin-right: 10px;
  width: 130px;
  border: 1px solid #006;
  font-size: x-small;
}

#blogroll h2 {
  font-size: x-small;
  background: #006;
  color: #fff;
  padding: 5px;
  margin: 0;
}

#blogroll div {
  padding: 5px;
}
```

Note that I've used contextual selectors (remember them from Chapter 3 and Chapter 4?) to style the heading as well as the links themselves. Although you can't see it from the code extract that Blogrolling provided to us (all that's visible is a link to a JavaScript file), each link that we add is wrapped in a `div`. I've added a little padding to each `div` inside the `blogroll` container, like so:

```
#blogroll div {
  padding: 5px;
}
```

Also note that I've floated the `blogroll div`, and given it a specific width. This will place the list on the right-hand side of the screen (to the right of the previously floated feature image). Figure 10.20 shows how all this fits together on the page.

Figure 10.20. The blogroll in place



We've added a blogroll! Now, any time you add a good web site to your bookmarks, if it is related to your own web site's theme, you might consider adding it to your blogroll, too. All you need to do is head over to the Blogrolling site, log in, and add a link as we did above.

Discussion Forums

Arguably one of the best ways to create a virtual community around your web site—and to ensure that people come back time and time again—is to provide a chat forum. There is one small problem with this, though—forums aren't particularly straightforward to set up, and once you start to get regular posts appearing, you'll face the issue of moderating the forum's content. Will you moderate it

yourself? Will you just let things take their own course?[14] Will you empower other regular visitors to moderate the forums?

Most of the fully featured (and free) forum software products that you could use have some prerequisites that basically rule out my covering them in this book. For example, many packages require your hosting company to support PHP (a "scripting language") and to make a MySQL database available to you. At this stage, this is probably still something of a foreign language to you.

A simple solution for creating a community is to use Yahoo! Groups.[15] Signing up for the service is as simple as registering for an email account with Yahoo! Then, you can invite people to take part by registering for the discussion group themselves.

The beauty of using Yahoo! Groups is that many people will already be familiar with the mechanics of the Yahoo! services, and they may already have sign-in details that they can use (from their Yahoo! mail accounts). Also, the service is provided by a company that you know and can trust and, as such, it's unlikely to disappear overnight—taking your new virtual community with it.

The downside of using Yahoo! Groups is that this option takes users away from your web site. The best that you can do is to provide a link from your web site to the group that you set up. From that point forward, your community is entirely in the hands of Yahoo! (capable though they are), which could also present problems if you ever wanted to move the forum content to another location.

Summary

In this chapter I've shown that no matter how much work you've put into building your web site, there's always something else that you could do to add a finishing touch or two. I've focused on some add-ons that, I believe, really do improve a web site and encourage you—as well as others visiting the site—to use it more. However, I would like to sound one note of warning at this point, and it is this:

Know when to stop!

[14] The short answer to that one is: no! If you give people the option to run wild, they may well do just that, which could even get you into hot water, legally speaking. For example, someone could slander someone else on your forum, or link to copyrighted material for others to download, and you as the web site owner could be responsible for those people's actions. Moderation is a *great* idea.
[15] http://groups.yahoo.com/

There are many web sites still kicking around today that were built in the mid-to-late 90s, when the motto seemed to be "the more flashing/spinning/bouncing widgets on the page, the better the web site." Thankfully, as the art of web design has matured, people have come to realize that "less is more." Please bear this in mind when adding features. Otherwise, before you know it, the "bells and whistles" will have taken over completely!

Don't Be too Reliant on Third-party Services

There's one other reason not to go overboard adding to your web site features provided by a third party. If that third party's web server is running slowly for some reason, and your web page is trying to access something from that provider's server, your web page may appear to load slowly. Also, you need to be careful about "putting all your eggs into one basket." If the service you use is free, be prepared for the eventuality that one day, the owners may decide to close the service down, or charge for it. Would your web site be able to function properly if this happened?

At this point, your web site should have all the features that it needs. You've followed the advice I've given in this book, and you have a well-formed, standards-based web site that you can be proud of. If you've enjoyed designing and building your web pages, you might want to improve your skills even further. Instead of me leaving you to fend for yourself at this point, I'm going to make some suggestions about where you can go to get the skills you need, because, frankly, there are an awful lot of bad web sites and books out there, and you could easily pick one up by mistake. The last thing I want is for you to undo any of the good work you've done so far! So, let's continue on to the final chapter, in which we'll explore the possibilities that now lie before you.

What's Next?

If you've enjoyed these chapters from *Build Your Own Web Site The Right Way Using HTML & CSS*, why not order yourself a copy?

Build Your Own Web Site The Right Way Using HTML & CSS is the most easy-to-understand, fun, and comprehensive guide to learning the building blocks of web development—HTML and CSS. In these sample chapters, you've so far learned the basics of HTML, and played with some CSS—and you've seen the end of the book where you get to add some really cool stuff to your site, like a search facility. In the rest of the book, you'll get a much better grounding in more complex CSS (like using CSS for positioning), you'll get up to speed with other important web development concepts, like accessibility, and you won't pick up any bad habits along the way!

Join guru web developer, Ian Lloyd, as you build a complete, fully-featured web site together.

Build Your Own Web Site The Right Way Using HTML & CSS includes download access to all of the best practice code samples and graphics used in the book—so you can follow along without any retyping, and easily adapt the code for use on your own web sites.

In the rest of the book, you'll:

- Layout your web site using CSS
- Create images and other graphics
- Use tables correctly
- Build forms to interact with your visitors
- Find a web host
- Get your site online
- Make sure your site is accessible to those with disabilities
- Promote your site
- Add a blog to your site to make updating it easy
- Get a complete XHTML reference, complete with examples
- And a whole lot more!

On top of that, order direct from sitepoint.com and you'll receive a free 17" x 24" poster of your choice!

Order now and get it delivered to your doorstep!

Index

C

D

E

F

G

H

I

J

T

U

V

W

X

Y