Article

Learn HTML and CSS: An Absolute Beginner's Guide

By Ian Lloyd

January 23rd 2009

Reader Rating: 9.3

So, you're ready to take the plunge and begin to learn how to build your own web pages and sites? Fantastic! We've got quite a ride ahead, so I hope you're feeling adventurous.

This information is an excerpt from my recently released book, *Build Your Own Web Site The Right Way Using HTML & CSS, 2nd edition* [1].

In the following pages, I'll show you how to set up your computer—be it PC or Mac—so that you're ready to build a site.

Then, we'll meet XHTML and walk through the details of how to structure a web page correctly. In the process, we'll play around with elements, comments, and symbols, and we'll talk about the concept of nesting. It might sound complex, but it's not! In fact, by the end of that discussion, you'll have a working web page that displays text, links, an image and even a quote!

Finally, we'll turn to the topic of Cascading Style Sheets, which we'll use to change the way elements of your web page look. We'll discuss the different ways in which you can use styles to control the way your page looks before creating our own .css file and using it—as well as tools like selectors and spans—to spruce up our web page's headings, text, and so on.

About the Author

Ian Lloyd

Ian is a UK-based senior web designer/developer who has written or co-written many web development books, including SitePoint's The Ultimate HTML Reference and Build Your Own Web Site the Right Way using HTML & CSS. Ian runs accessify.com (a web accessibility site that he started in 2002), is a regular speaker at web development conferences such as South By South West (SXSW) and works as a senior developer at UK financial services organisation Nationwide.

Ian Lloyd has written **5** articles for SitePoint with an average reader rating of **8.9**.

View all articles by Ian Lloyd...

Illustration by: Matthew Magain

Don't worry if some of these terms are unfamiliar—this excerpt, like the book itself, assumes that you have no knowledge about building web pages. The information I'll present here basically comprises the first three chapters of the book—later chapters go on to show you how to build forms, optimize graphics for the Web, publish your site, and soup it up with plug-in tools such as blogs, image galleries, and more—the Table of Contents [2] has all the details.

But now, let's start to set up your computer so that you're ready to build your first web page. Don't forget: you can grab these instructions in PDF format [3] if you'd like to save them for reading offline.

Chapter 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 are ready to go.

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 already have an Internet connection, most likely broadband (or similar). Don't worry if you have a slower connection: it won't affect any of the tasks we'll undertake in this book. It will, however, mean that some of the suggested downloads or uploads may take longer to complete, but you probably knew that already.

Note: 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 it 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 may 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**; a program that allows you to edit plain text files. 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**—that's the application you use to view web sites.
- Finally, when you're happy with your new web page, you can put it on the Internet using an **FTP client**—a utility that allows

you to transfer files across the Internet using the File Transfer Protocol. Using FTP may seem a little complicated at first, but thankfully you won't need to do it too often. We'll discuss FTP clients in detail in Chapter 8, Launching Your Web Site.

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

Windows Basic Tools

In the following section—and indeed the rest of this book—where we refer to the Windows operating system, that's really a shorthand way of saying Windows Vista (in all its confusing varieties). Any instructions and screen shots will be with Vista in mind. However, we are not going to shut out all you Windows XP users—and there are many people out there who use XP in preference to Vista, much to Microsoft's displeasure—so where instructions provided for Vista are not the same for XP, we'll cover the differences for you.

Your Text Editor: Notepad

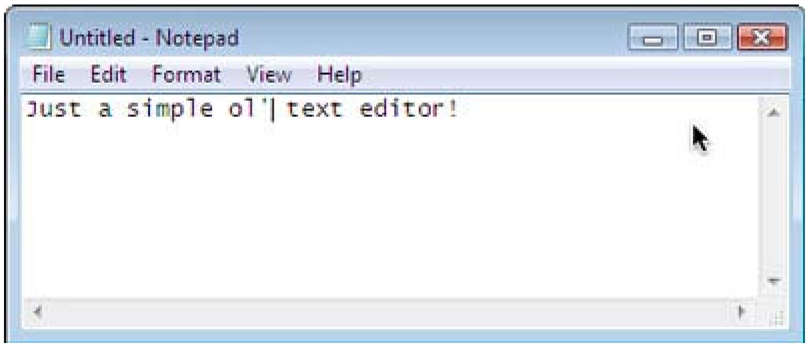
The first tool we'll consider is the text editor. Windows comes with a very simple text editor called Notepad. Many professional web designers who use complicated software packages first started out many years ago using Notepad; indeed, many professionals who have expensive pieces of software that should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

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

Tip: Shortcut to Notepad

To save yourself navigating to this location each 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.

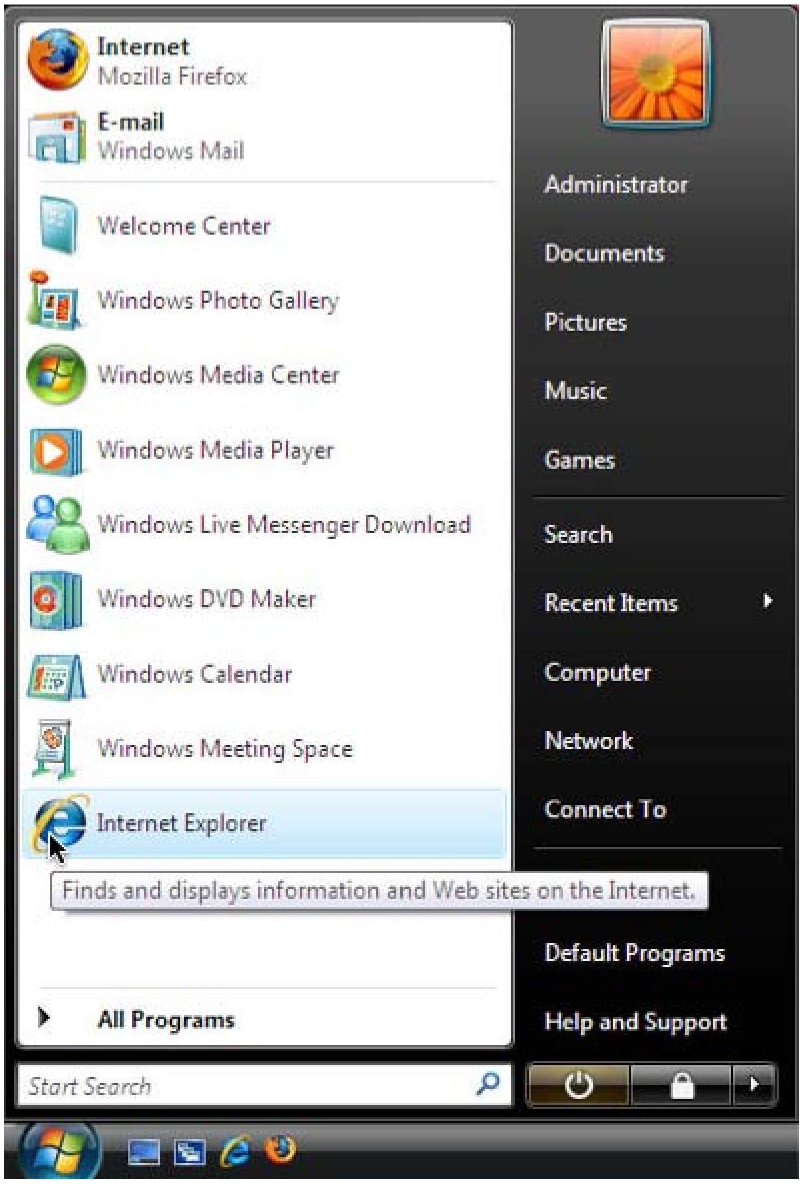
Notepad is the most basic of applications, as you can see below.



Just a simple ol' text editor!

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 (IE). Well, that's your viewer. Internet Explorer sits right there in the **Start** menu, also in the **Programs** folder (accessed via **All Programs** from the **Start** menu), in the **Quick Launch** area (bottom left of the Start menu, near the Windows logo), and a shortcut may also lurk on your desktop.

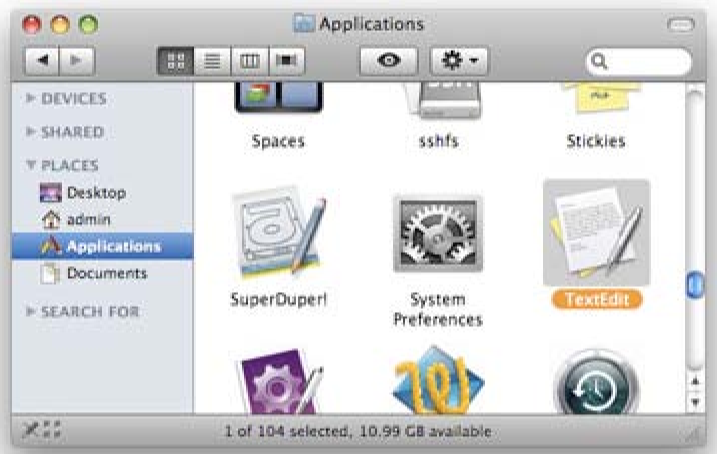


Mac OS X Basic Tools

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

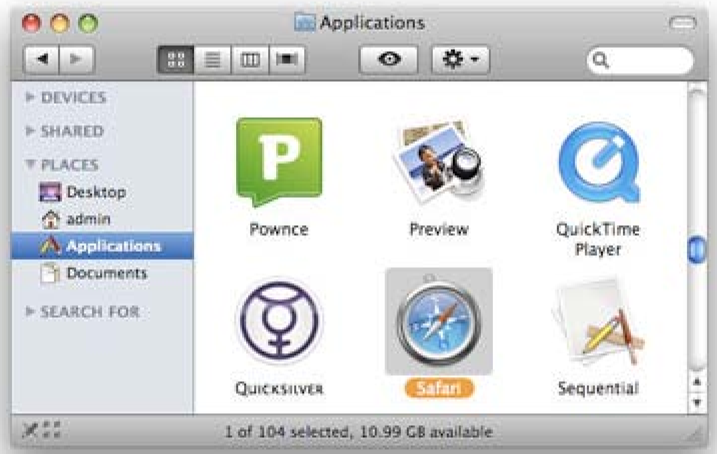
While Windows has Notepad, the Mac has TextEdit, which can be found in the **Applications** folder.



Unlike Notepad, TextEdit works as 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 adjust 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

The default browser for Mac users is Safari. You can usually find Safari in the **dock** (the dock is the bar of icons at the bottom of your screen), but you can also access it through the Applications folder.



Tip: 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. To add a program to the dock, just drag its icon from the Applications folder onto the dock, and presto! The application is now easily accessible whenever you need it.

If you are using a slightly older Mac, you may also have a copy of Internet Explorer installed. Our advice for Internet Explorer for Mac? Send it to Trash. The Mac version of IE was abandoned by Microsoft many years ago, so it's considerably outdated and is rarely supported or used in the wider world; no new Macs come with this application preinstalled. It also bears no real resemblance to its Windows counterpart, for those more comfortable using IE.

Beyond the Basic Tools

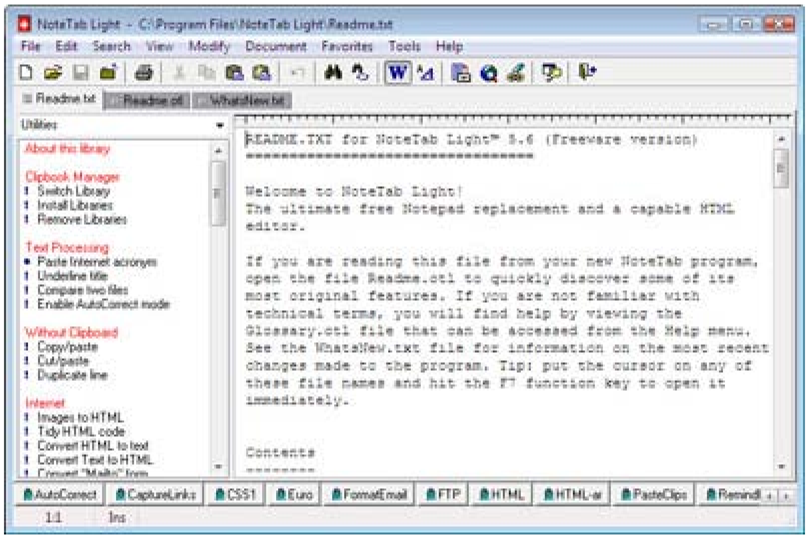
You can certainly make a good start using the tools mentioned above. However, once you're dealing with a handful of web pages and other resources, you may want to go beyond these basic tools. We'll show you how to use some slightly more advanced applications later in the book.

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. And remember, they're all free!

Windows Tools

NoteTab

NoteTab's tabbed interface lets you have many different files open simultaneously without cluttering up your screen. Files that you've opened are remembered even after you close the program and open it again later, which is very useful when you're working on a batch of files over many days. You can download the free NoteTab, or its Light version, from http://www.notetab.com/ [4].



```
README.TXT for NoteTab Light™ 5.6 (Freeware version)
====================================

Welcome to NoteTab Light!
The ultimate free Notepad replacement and a capable HTML
editor.

If you are reading this file from your new NoteTab program,
open the file Readme.otl to quickly discover some of its
most original features. If you are not familiar with
technical terms, you will find help by viewing the
Glossary.otl file that can be accessed from the Help menu.
See the WhatsNew.txt file for information on the most recent
changes made to the program. Tip: put the cursor on any of
these file names and hit the F7 function key to open it
immediately.


Contents
========
```

Firefox

As mentioned in the Preface, Firefox is a very 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 tabbed interface that helps keep your computer free from window clutter. You can download Firefox from http://www.mozilla.com/firefox/ [5]; the browser is depicted below.

Mozilla Firefox Start Page - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.google.co.uk/firefox/
Web Images Maps News Shopping more
Firefox Start
Google
Advanced Search
Preferences
Search: the web pages from the UK
Google Search
About Mozilla
Done

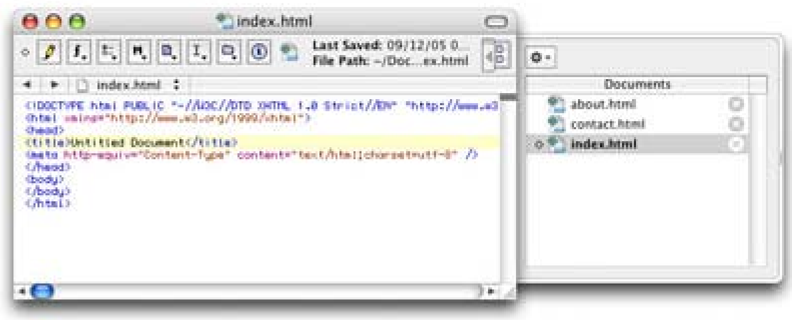
Customize Firefox to work your way with free Add-ons - add anything from weather forecasts to music players.

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, a free text editor from BareBones Software", from the BareBones Software web site [6].



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

Firefox

Firefox is popular not just among Windows users, but also with Mac users, many of whom prefer to use it instead of Safari (often because of the extra features—known as **add-ons**—that can be bolted on to the browser). A web page viewed in Firefox should display the same regardless of whether the browser is installed on a PC running Windows XP or Vista, on a Mac running OS X, or on Linux, a free, open source operating system (generally favored by highly technical people who like to tinker with their computers a lot). 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 mainly use Firefox in the examples included in this book.

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," applies. A professional image editing program, like Photoshop or Fireworks, costs hundreds of dollars. While these programs offer some excellent capabilities, we can't really recommend that you go out and pay for them unless you're 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 hope 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**—a way to construct an image by stacking 2 or more layers, one on top of the other. 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 may need to be left to download 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 [7]. If you'd rather try the lighter Photoshop Elements, trial versions are available for Windows [8] and Mac [9].

Adobe Fireworks

You can download a trial version of Fireworks from the Adobe web site [10].

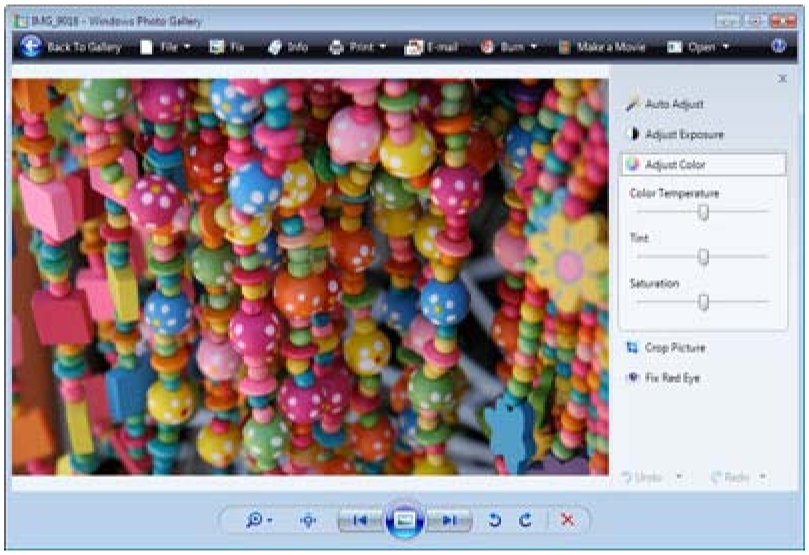
Paint Shop Pro

Paint Shop Pro is available for Windows only. To download a trial version, visit the Paint Shop Pro site, and click the Free Trial link [11].

Windows Tools

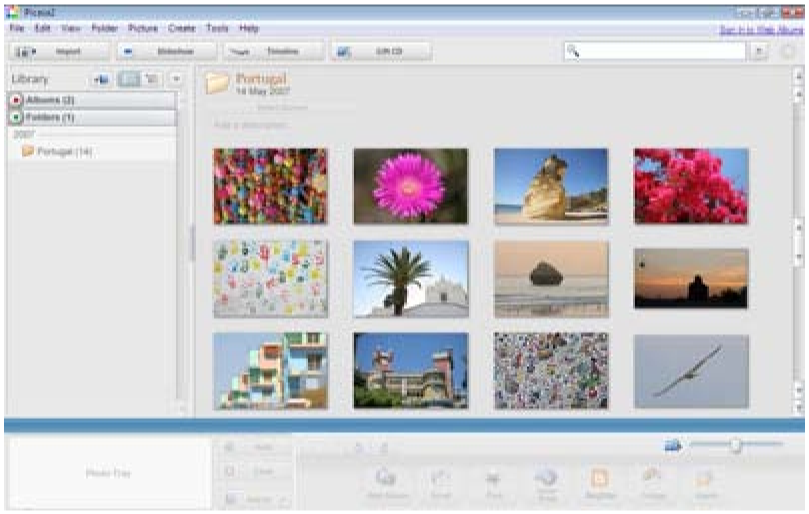
A standard Windows install has not always been blessed with image editing software. Certainly this was the case with Windows XP (although if you bought the computer as a bundle with PC, scanner, or digital camera all together, you might be lucky and find some image editing software included in the deal; scout around in your **Start > All Programs** menu to see what you can uncover).

In Windows Vista, the Photo Gallery application has seen some big improvements over its previous XP incarnation and now includes some basic, but still useful, image manipulation tools, including cropping, color, and contrast adjustment. The Photo Gallery application can be found directly in the **Start** menu.



Picasa

If you're using Vista, the tools offered in Photo Gallery may do everything that you need; if you're using XP, you'll almost certainly need to use an extra application. With that in mind, you might like to try out an excellent image management tool that Google offers for free download. 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 [12], and soon enough you'll be using this program to crop, rotate, add special effects to, and catalog the images stored on your computer. The figure below gives you an idea of the program's interface.

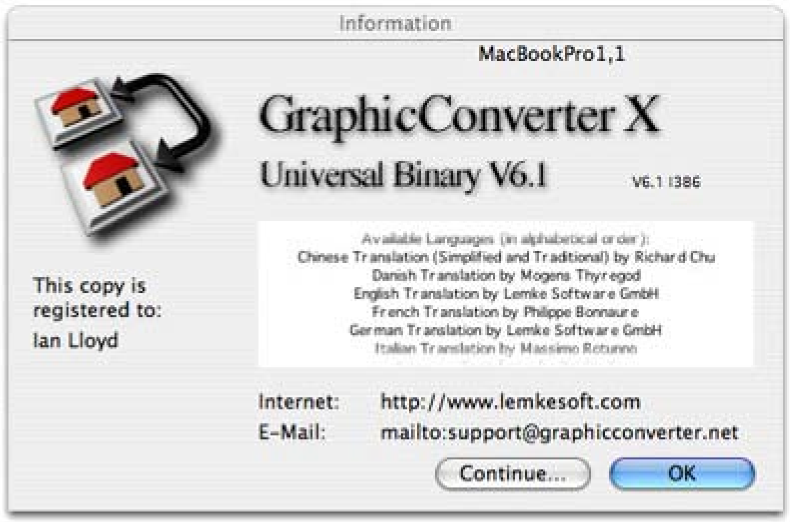


Mac OS X Tools

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 been bundled with new Macs at times, and is also available for download (you'll be encouraged to pay a modest registration fee for the software, but you can try it out for free). Although this is primarily a tool for converting graphic files, it can also be used for simple editing tasks. Using GraphicConverter, which is illustrated below, you'll be able to crop, resize, rotate, and add text to any image.



Information

MacBookPro1,1

GraphicConverter X

Universal Binary V6.1 V6.1 I386

Available Languages (in alphabetical order):
Chinese Translation (Simplified and Traditional) by Richard Chu
Danish Translation by Mogens Thyregod
English Translation by Lemke Software GmbH
French Translation by Philippe Bonnaure
German Translation by Lemke Software GmbH
Italian Translation by Massimo Rotunno

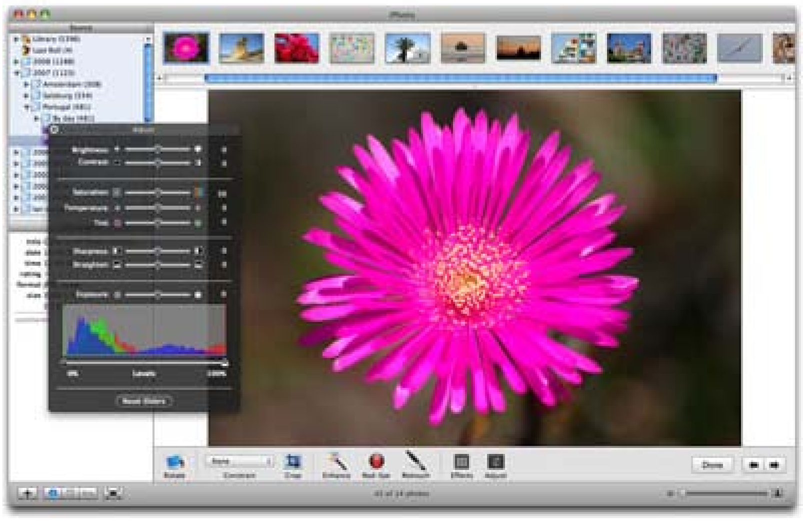
This copy is registered to:
Ian Lloyd

Internet: http://www.lemkesoft.com
E-Mail: mailto:support@graphicconverter.net

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 to be 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 iPhoto also has some very useful editing facilities that take it beyond a mere cataloging tool.

iPhoto, shown below, can be found in the Applications folder or in the dock.

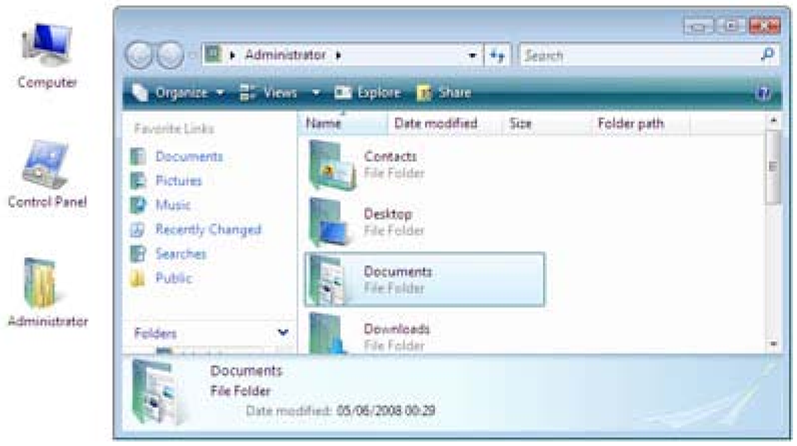


Creating a Spot for Your Web Site

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 to create 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 **Documents** folder (or the **My Documents** folder in Windows XP). The **Documents** folder can be found inside your user folder. "But what's this user folder?" I hear you cry. And a fair point too, because it won't be labeled **User** but rather it will be labeled according to the user name that you provided when you first set up Windows. In Windows Vista you'll find the user folders of all local computer account holders under **C:\Users** (in Windows XP it's under **C:\Documents and Settings**) and will have a folder name matching your user name. More conveniently though, you'll find it on your computer's desktop as shown below.

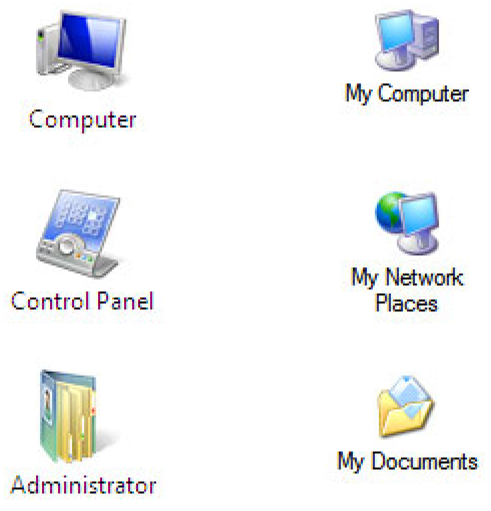


Don't worry if your user folder is not on the desktop: it's easy to get it to appear there (see the tip below for details on how to add this for Vista and XP). Double-click to open your user folder, then double-click on **Documents**, then finally create a new folder called Web by selecting **File > New > Folder**

Tip: Displaying the Users Folder in Vista and the My Documents Folder in Windows XP

Can't find your user folder on your Vista desktop? Missing your My Documents folder in XP? In an effort to clean up your desktop, you may have removed the icon by accident—it's easily done. This is how you can return the folder to your desktop:

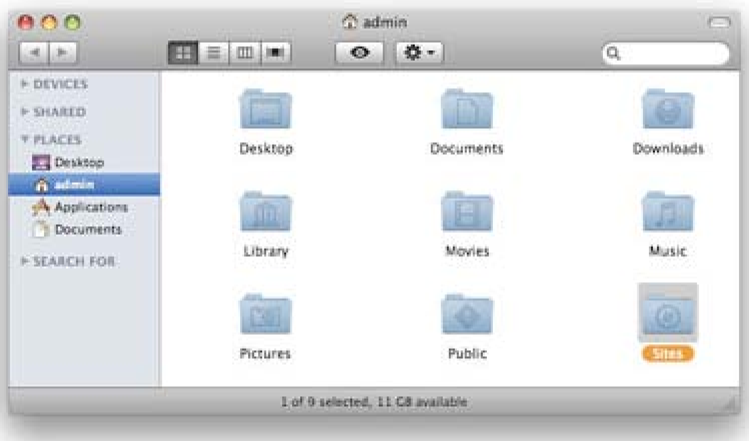
- From the **Start Menu**, select **Control Panel**.
- Select **Appearance and Personalization** (or **Appearance and Themes** in XP).
- **Vista users**: choose **Personalization** and listed in the top left, under the title **Tasks** is the option to **Change Desktop Icons**. A new dialogue box will appear: check the option entitled User's Files in the section Desktop Icons, then press **OK**. You may now also close the **Appearance and Personalization** window.
- **XP users**: select **Change the desktop background** from the list of options, then click the **Customize Desktop...** button at the bottom. Check the **My Documents** option in the dialogue box that appears, then click **OK**. Close the **Appearance and Themes** window also by pressing **OK**.
- Your user folder/My Documents folder should now be back on the desktop, as shown below.

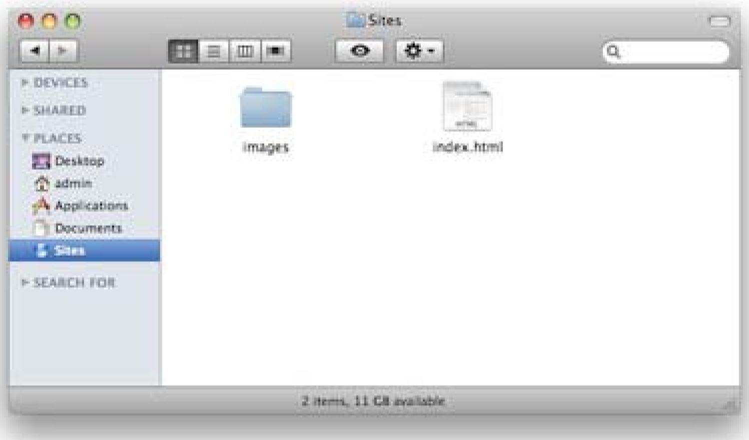


Mac OS X

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

It's easy to add the Sites folder to your sidebar for quick access: just drag the folder to the sidebar in the same way you add items to the dock.





Getting Help

Books may be a wonderful way to learn: you're sitting there with a computer running, perhaps a cup of coffee keeping your mind ticking over, 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, though there may be some therapeutic value to it!

Hopefully, you won't find yourself with 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 or want to learn more, your first stop should be the SitePoint Forums [13]. 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 forums. Whether you have questions about writing content for 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 free of charge? We 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!

Chapter 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). Elements tell the web browser what a particular item in the page is: a paragraph, a heading, a quotation, and so on. These 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, 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.

To drive the car you need to have wheels (including the steering wheel), and a place to sit. The car must also have some kind of chassis to which the bodywork can be bolted. An engine is required 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.

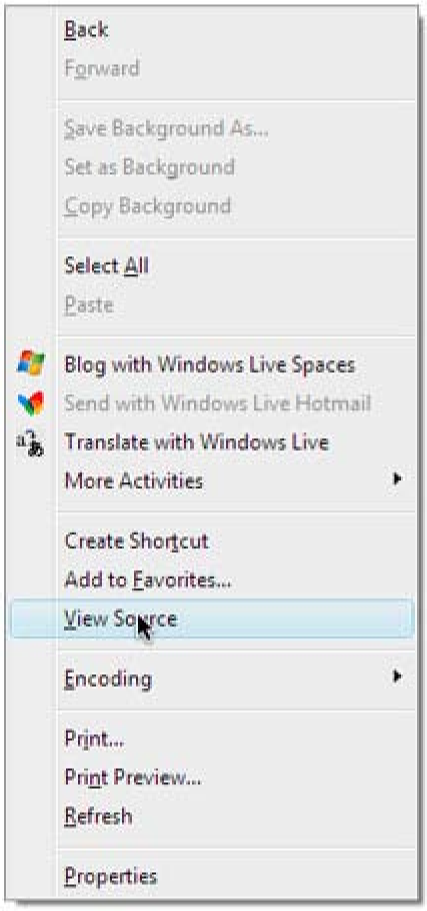
Like the car, your web page also needs to have a chassis: a basic structure upon which everything else can be built. But what does this hypothetical chassis look like? The best way to find out is to roll up our sleeves, figuratively speaking, then take a closer look at what's going on underneath the cosmetic features.

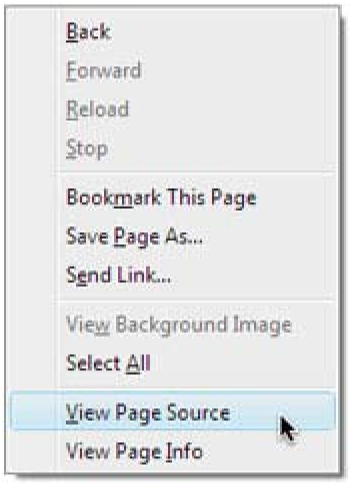
Viewing the Source

One of the great things about learning to build web pages is that we all 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 that?

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, for example the Web Standards Project's homepage. The Web Standards Project (WaSP) is a group that promotes the benefits of building your web site correctly, so you can be pretty confident that 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 a context menu similar to those shown below.





- Select **View Source**, (or **View Page Source** for Firefox) 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.

Caution: Careful Who You Trust!

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. You would have seen all of these 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
- `<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>
```

Those of you with eagle eyes may have also spotted the `<meta>` tag in the markup above. I know I haven't mentioned this yet; we'll get to it soon enough. For now be content with the knowledge that, although the <meta> tag is not part of the skeletal requirements of a web page, it serves many useful purposes, especially the provision of supporting information about the web page.

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 [14] (short for Document Type Definition). It must 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 2007 on one computer, and tried to open it on another computer that only had Word 2000 on it? Frustratingly, without some preemptive massaging when the file is saved in the first place, this doesn't work quite as expected. It fails because Word 2007 includes features that Bill Gates and his team hadn't even dreamed of in 2000, and so Microsoft needed to create a new version of its file format to cater for these new features. Just as Microsoft has many different versions of Word, so too are there different versions of HTML, most recently XHTML and HTML 5. Mercifully, the different versions of HTML have been designed so that it 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, let's take a jargon break! There are way too many abbreviations for this paragraph.

Note: HTML5—The New Kid on the Block

Note that at the time of writing, the HTML5 specification is not yet finalized and browser support for it is also incomplete (understandable, given the moving goalposts). For this reason, we'll not be covering HTML5 in this book. You should, however, be aware of its existence at the very least.

Note: Jargon Busting

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 know, 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 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 yourself (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 you won'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 the time being and I'll guide you through.

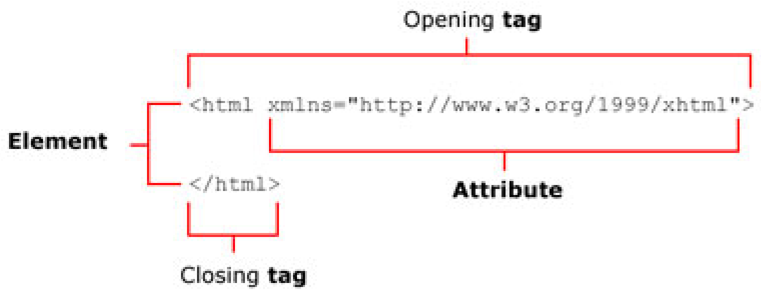
The html Element

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.
- 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 [15] element. The figure below shows what we have.



The figure below depicts the opening tag, which marks the start of the element:

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

Below this 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">
```

And there is one attribute in the opening tag:

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

Note: 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 only 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 (apart from the doctype) is kept within that outer container. 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

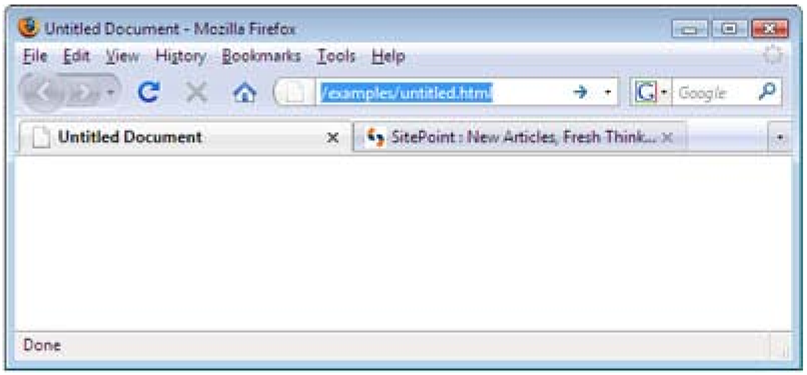
The head [16] element contains information about the page, but no information that will be displayed on the page itself. For example, it contains the title [17] element, which tells the browser what to display in its title bar (the title bar is the very top part of the browser window—the part with minimize, maximize and close buttons):

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

The title Element

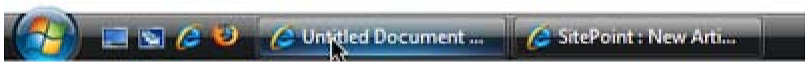
The opening `<title>` and closing `</title>` tags are wrapped around the words "Untitled Document" in the markup above. Note that the `<title>` signifies the start, while the closing `</title>` signifies the end of the title. That's how closing tags work: they have forward slashes just after the first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

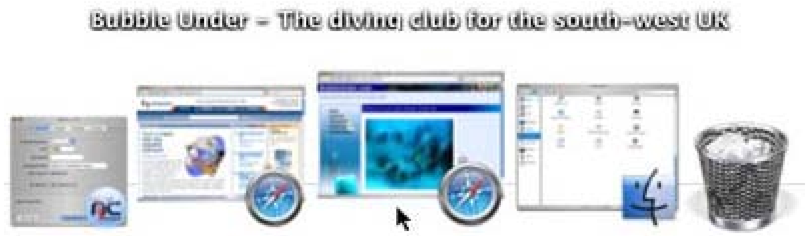
The Untitled Document title is typical of what HTML authoring software provides as a starting point when you choose to create a new web page; it's up to you to change those words. As the figure below shows, it really pays to put something meaningful as a title, and not just for the sake of those people who visit our web page.


Untitled Document - Mozilla Firefox
File Edit View History Bookmarks Tools Help
/examples/untitled.html
Untitled Document
SitePoint : New Articles, Fresh Think...
Done

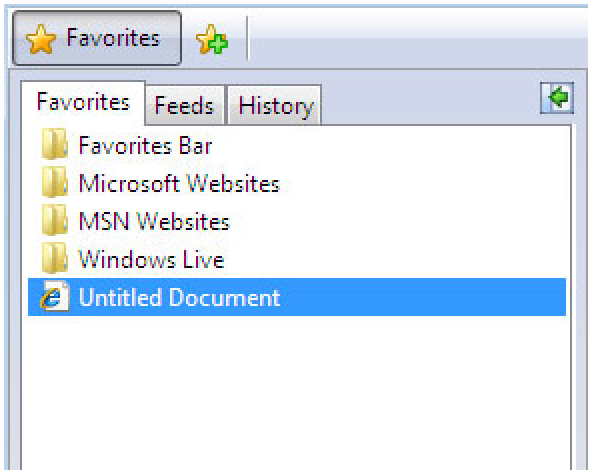
The content of the title element is also used for a number of other purposes:

- It's the name that appears in the Windows Taskbar—that strip along the bottom of your Windows desktop that show all the currently open windows—for any open document. It also appears in the dock on a Mac. When you have a few windows open, you'll appreciate those people who have made an effort to enter a descriptive title!





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



- 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:

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

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 written 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 (at least at this stage).

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. But if you are using a humble text editor and need some help, you need only remember that there is a completely searchable HTML reference, accessible at any time at SitePoint.com [18].

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: 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 shown 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):

```
<!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:

1. Select File > Save As... from the menu and find the Web folder you created inside your Documents folder.
2. Enter the filename as basic.html.
3. Select UTF-8 from the Encoding drop-down list.
4. Click Save.

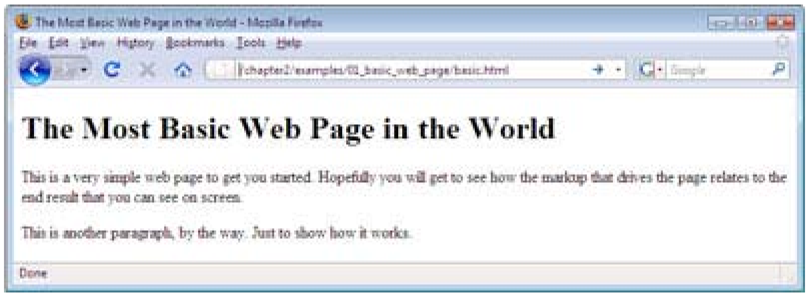
If you're using TextEdit on a Mac, first make sure that you're in plain text mode, then:

1. Select File > Save As... from the menu.
2. Find the Sites folder, enter the filename as basic.html.
3. Select Unicode (UTF-8) from the Plain Text Encoding drop-down list.
4. Click Save.
5. 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.

Caution: 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 tries to view your web site (say, a Korean friend of yours), they'll probably end up with a screen of gobbledygook. 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. The figure below shows how the page displays.


The Most Basic Web Page in the World - Mozilla Firefox
File Edit View History Bookmarks Tools Help
chapter2/examples/01_basic_web_page/basic.html
Google
Done

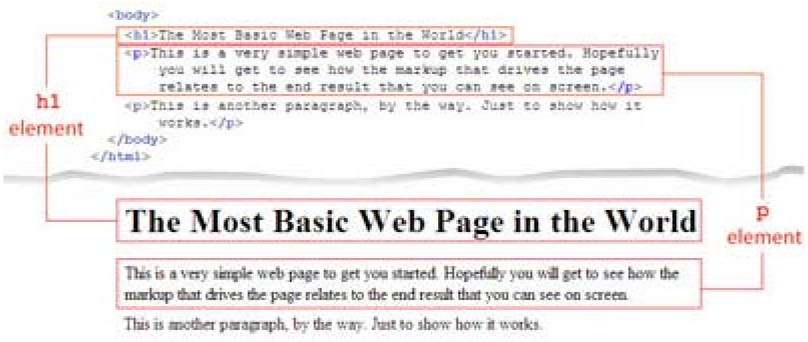
The Most Basic Web Page in the World

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.

This is another paragraph, by the way. Just to show how it works.

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



```
  <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>
```

The Most Basic Web Page in the World

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.

This is another paragraph, by the way. Just to show how it works.

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.

Important: A Case of Keeping Low

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. 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 the smaller the font size (unless you have re-styled the headings with CSS, but more of that later in this article.

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, as `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

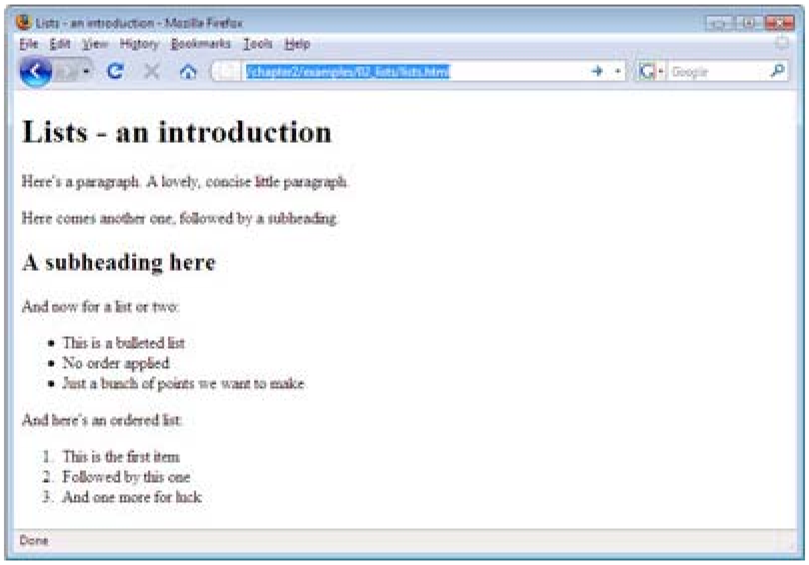
Let's imagine that you want a list on your web page. To include an *ordered list* (the HTML term for a numbered list) of items, we use the `ol` element. An *unordered list*—known as bullet points to the average person—makes use of the `ul` element. In both types of list, individual points or list items 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. Simple.

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:

```
<!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 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 creating your own web site—it really will pay to learn by doing. Even if you make mistakes, it's still a better way to learn (you'll be pleased when you can spot and fix your own errors yourself). When displayed in a browser, the above markup should look like the page shown below.

There are many, many different elements that you can use on your web page, and we'll learn 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 clear-cut: for example, what would you use `div`, `span`, or `a` elements for? Any guesses? All will be revealed in good time.


Lists - an introduction - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Done

Lists - an introduction

Here's a paragraph. A lovely, concise little paragraph.

Here comes another one, followed by a subheading.

A subheading here

And now for a list or two:

- This is a bulleted list
- No order applied
- Just a bunch of points we want to make

And here's an ordered list:

1. This is the first item
2. Followed by this one
3. And one more for luck

Commenting Your HTML

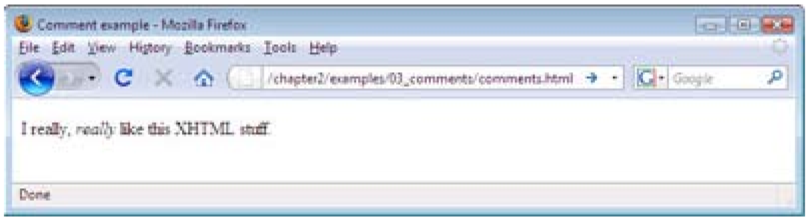
Back in the garage, you're doing a little work on your project car and, as you prepare to replace the existing tires with a new set, you notice that your hubcaps aren't bolted on: you'd stuck them to the car with nothing more than super glue. There must have been a good reason for doing that, but you 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:

```
<!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>
```

The figure below shows the page viewed on-screen.


Comment example - Mozilla Firefox
File Edit View History Bookmarks Tools Help
/chapter2/examples/03_comments/comments.html
Google
Done

I really, *really* like this XHTML stuff

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.

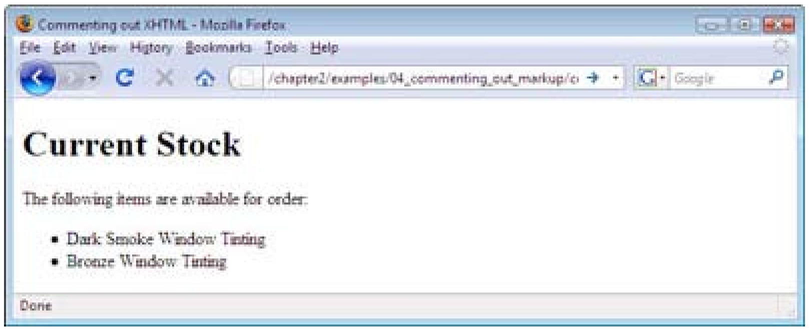
Oh, and did you spot how we snuck another new element in there? The emphasis element, denoted with the `<em>` and `</em>` tags, is used wherever ... well, do I really need to tell you? Actually, that last question was there to illustrate this point: did you notice that the word "really" appeared in italics?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

```
<!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>
```

The figure below shows how the page displays in Firefox.


Commenting out XHTML - Mozilla Firefox
File Edit View History Bookmarks Tools Help
/chapter2/examples/04_commenting_out_markup/c
Google
Done

Current Stock

The following items are available for order:

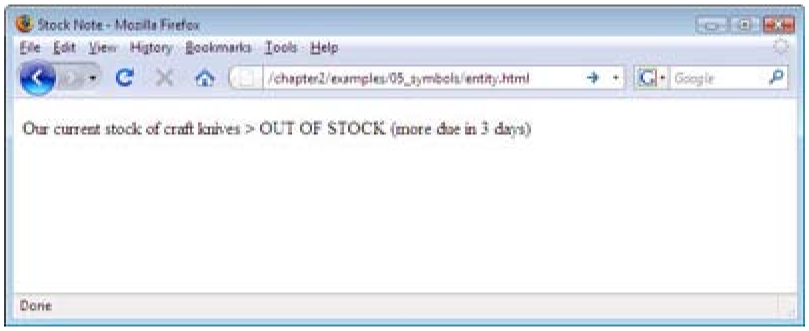
- Dark Smoke Window Tinting
- Bronze Window Tinting

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 `&gt;`—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 the figure below, "The `&gt;` entity is displayed as `>` in the browser".

```
<!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;
     OUT OF STOCK (more due in 3 days)</p>
   </body>
 </html>
```


Stock Note - Mozilla Firefox
File Edit View History Bookmarks Tools Help
/chapter2/examples/05_symbols/entity.html
Google
Done

Our current stock of craft knives > OUT OF STOCK (more due in 3 days)

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 below:

- `&gt;` >

- `&lt;` <
- `&amp;` &
- `&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 worth keeping, 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 on it (the car chassis with no bodywork or interior). 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:

```
<!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

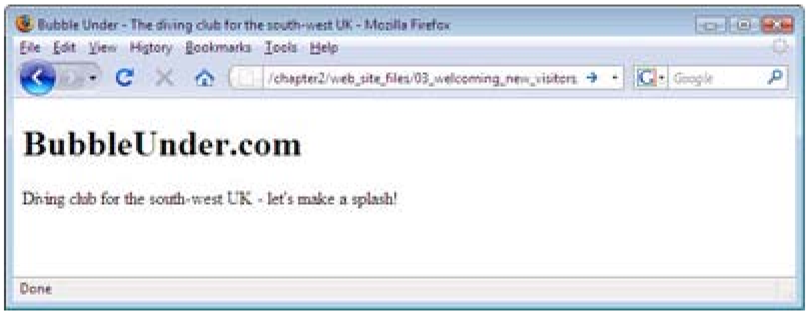
Remembering what we’ve learned so far, let’s make a few changes to this document. Have a go at the following:

- 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—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):

```
<!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>
     <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. The figure below shows what it should look like.


Bubble Under - The diving club for the south-west UK - Mozilla Firefox
File Edit View History Bookmarks Tools Help
/chapter2/web_site_files/03_welcoming_new_visitors
Google
Done

BubbleUnder.com

Diving club for the south-west UK - let's make a splash!

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:

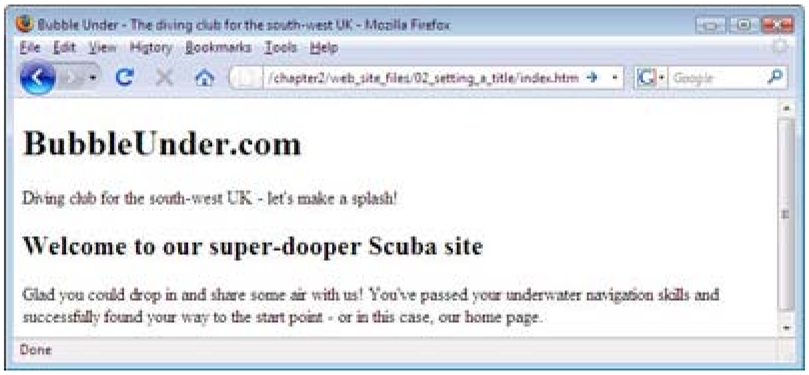
```
<body>
   <h1>BubbleUnder.com</h1>
   <p>Diving club for the south-west UK - let's make a splash!</p>
   <h2>Welcome to our super-dooper Scuba site</h2>
   <p>Glad you could drop in and share some air with us! You've
       passed your underwater navigation skills and successfully
       found your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

Apologies for the diving terminology puns, they're truly cringe-worthy!

Important: 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 below.


Bubble Under - The diving club for the south-west UK - Mozilla Firefox
File Edit View History Bookmarks Tools Help
/chapter2/web_site_files/02_setting_a_title/index.htm
Google
Done

BubbleUnder.com

Diving club for the south-west UK - let's make a splash!

Welcome to our super-dooper Scuba site

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.

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).

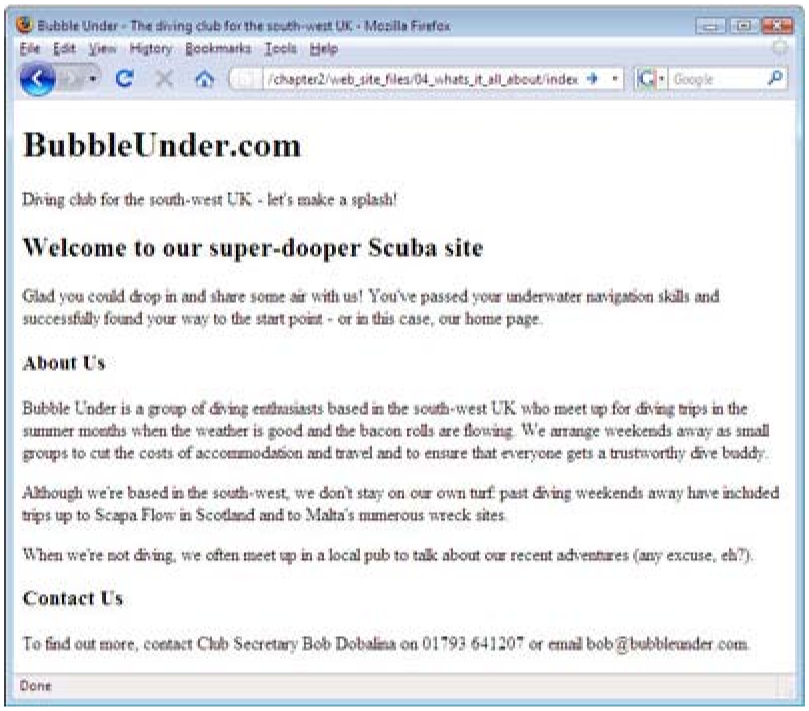
- Next, add the following text. Note that there is more than one paragraph. "Bubble Under is a group of diving enthusiasts based in the south-west UK who meet up for diving trips in the summer months when the weather is good and the bacon rolls are flowing. We arrange weekends away as small groups to 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?)."
- 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 much like the following markup in the body of your document:

```
<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>
```

You can see how our home page is shaping up below.



Bubble Under - The diving club for the south-west UK - Mozilla Firefox
File Edit View History Bookmarks Tools Help
/chapter2/web_site_files/04_whats_it_all_about/index
Google
Done

BubbleUnder.com

Diving club for the south-west UK - let's make a splash!

Welcome to our super-dooper Scuba site

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.

About Us

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 away 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?).

Contact Us

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

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.

Note: 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.

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:

```
<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>
```

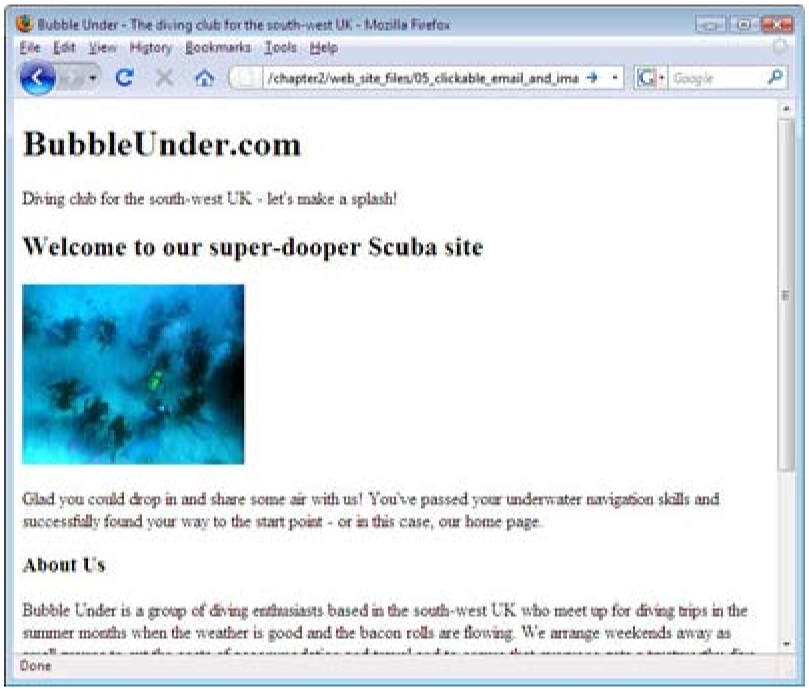
The `img` element is used to insert an image into our web page, and the attributes `src`, `alt`, `width`, and `height` describe the image that we're inserting. `src` is just the name of the image file. In this case, it's `divers-circle.jpg`, which you can grab from the code archive. `alt` is some alternative text that can be displayed in place of the image if, for some reason, it can't be displayed. This is useful for blind visitors to your site, 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. We'll cover pixels when we look into images in more detail a bit later.

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

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

```
<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 below.


Bubble Under - The diving club for the south-west UK - Mozilla Firefox
File Edit View History Bookmarks Tools Help
/chapter2/web_site_files/05_clickable_email_and_ima
Google
Done

BubbleUnder.com

Diving club for the south-west UK - let's make a splash!

Welcome to our super-dooper Scuba site

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.

About Us

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

Adding Structure

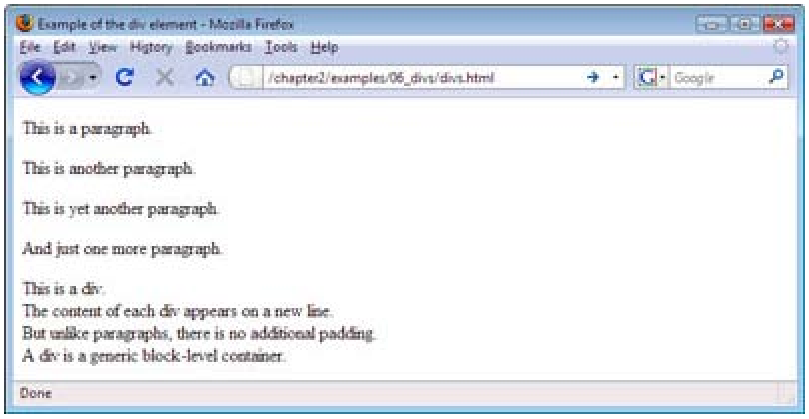
Paragraphs? No problem. Headings? You've got them under your belt. In fact, you're now familiar with the basic structure of a web page. The small selection of tags that we've discussed so far are fairly easy to remember, as their purposes are obvious (remember: `p` = paragraph). But what on earth is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

When you place content inside a `div`, 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. Unlike a `p` element, the `div` does not add any margins or padding. Compare the following:

```
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
<p>This is yet another paragraph.</p>
<p>And just one more paragraph.</p>

<div>This is a div.</div>
<div>The content of each div appears on a new line.</div>
<div>But unlike paragraphs, there is no additional padding.</div>
<div>A div is a generic block-level container.</div>
```

The difference can be seen below.


Example of the div element - Mozilla Firefox
File Edit View History Bookmarks Tools Help
/chapter2/examples/06_divs/divs.html
Google
Done

This is a paragraph.

This is another paragraph.

This is yet another paragraph.

And just one more paragraph.

This is a div.
The content of each div appears on a new line.
But unlike paragraphs, there is no additional padding.
A div is a generic block-level container.

The purpose of a `div` is to divide (hence the abbreviation 'div') up a web page into distinct sections—a basic structural framework with no styling—whereas `p` should be used to create a paragraph of text.

Important: Use Elements as Intended

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

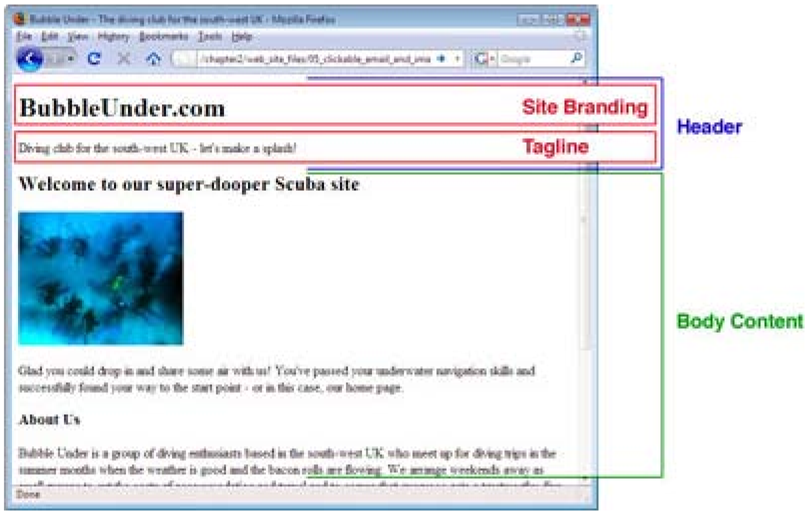
Rather than leaving the paragraph tags as they are, you might decide to have something like this:

```
<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. We have:

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



BubbleUnder.com

Diving club for the south-west UK - let's make a splash!

Welcome to our super-dooper Scuba site

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.

About Us

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

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, "Noting distinct sections in the basic web page". 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.

Important: No Sharing `id`s

`id` attributes are used in XHTML to uniquely 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.

Note: `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:

Example 2.12. `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 `div`s to the parts of the content that we've identified as "site branding" and "tag line."

Nesting Explained

We already know that `div`s can contain paragraphs, but a `div` can also contain a number of other `div`s. 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. Nesting elements can help to logically group sections of a web page together, just as you might do in the real world by placing a selection of small boxes containing similar items inside a larger box.

```
<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, “Noting distinct sections in the basic web page” shows, some nesting is taking place: the “site branding” and “tag line” `div`s are nested inside the “header” `div`.

The Sectioned Page: All Divided Up

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

```
<!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>
```

Tip: 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 just you—who's looking at the source markup. It does not affect how the browser interprets or displays the web page.*

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 at the same time, 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 you didn't buy this book to learn how to create a one-page web site, did you?

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 `div`s (`/chapter2/website_files/06_adding_structure_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 in lowercase only. 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`.
- Rename the other one as `about.html`.

Tip: 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. *Vista users, select **Organize > Folder and Search Options...**; Windows XP users, select **Tools > 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. This time, we're removing the introductory spiel and the "About Us" section. (If you're not sure you've got it right, keep reading: we'll show the altered markup in a moment.)

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:

`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>
```

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>
```

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` in a web browser 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.

- 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.

Note: 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 (for example, email bob@bubbleunder.com). 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" when "Submit an image," or "Notify us of your change of address" more than suffices. Never use "Click here" links—it really is bad linking practice and is discouraged for usability and accessibility reasons [19].

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 main body content starts.

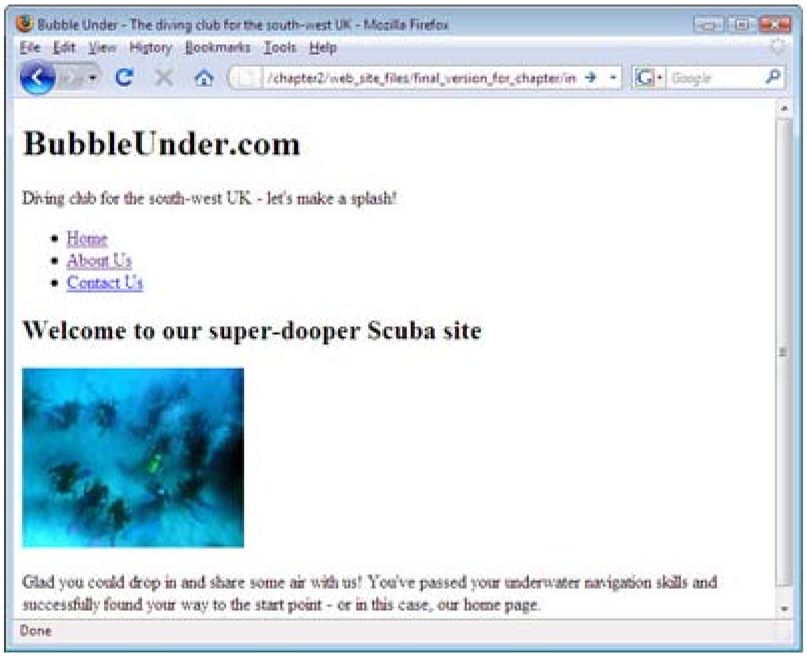
In the code below, the navigation block appears in position on the homepage:

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 below.


Bubble Under - The diving club for the south-west UK - Mozilla Firefox
File Edit View History Bookmarks Tools Help
/chapter2/web_site_files/final_version_for_chapter/in
Google
Done

BubbleUnder.com

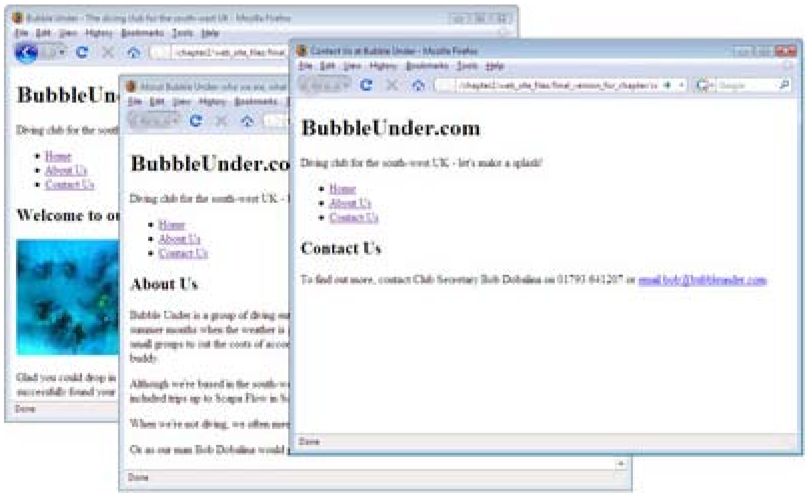
Diving club for the south-west UK - let's make a splash!

- Home
- About Us
- Contact Us

Welcome to our super-dooper Scuba site

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.

Add the block of 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. This is a landmark: you're now the creator of a working, navigable web site. Let's now discuss a few more XHTML elements that you can add to your pages.



BubbleUnder.com

Diving club for the south-west UK - let's make a splash!

- Home
- About Us
- Contact Us

Contact Us

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

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 are the lines:

"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` using a `blockquote` element; here's the markup you'll need:

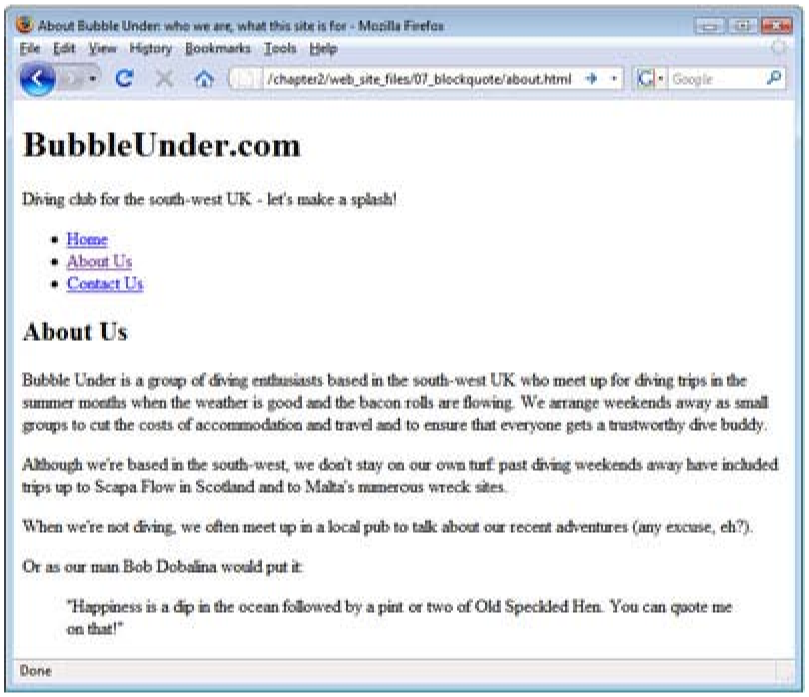
`about.html` (excerpt)

```
<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 (fictional) Club Secretary, Bob Dobalina:

```
<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.



About Bubble Under: who we are, what this site is for - Mozilla Firefox
File Edit View History Bookmarks Tools Help
/chapter2/web_site_files/07_blockquote/about.html
Google

BubbleUnder.com

Diving club for the south-west UK - let's make a splash!

- Home
- About Us
- Contact Us

About Us

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 away 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?).

Or as our man Bob Dobalina would put it:

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

Done

Note: Displaying `blockquote`s

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, namely CSS.

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>
```

So what do we do if something is both a quotation and a citation? The `blockquote` element has a `cite` attribute for this very purpose:

```
<blockquote cite="http://www.petermoore.net/sftb/chapter1.htm">
 <p>It didn't take long for a daily routine to form: when they
     left for work in the morning I'd still be in bed. And when
     they came home they'd find me sitting on the sofa, drinking
     beer and watching TV soaps.</p>
</blockquote>
```

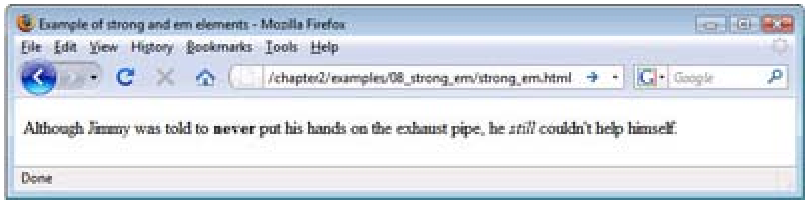
We're not using the `cite` element (or the `cite` attribute) in the diving web site, but you may find them useful in your own web site projects.

`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. The figure below 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>
```


Example of strong and em elements - Mozilla Firefox
File Edit View History Bookmarks Tools Help
/chapter2/examples/08_strong_em/strong_em.html
Google
Done

Although Jimmy was told to **never** put his hands on the exhaust pipe, he *still* couldn't help himself.

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:

```
<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>
```

Important: 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 effect should be achieved later). Be careful in your use of* `br`.

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

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.

Chapter 3. Adding Some Style

Earlier in this article, 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 *that* impressed. 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 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 the appearance of elements on the page: the size, style, and color of text, 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 jumps 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."

Note: But Wait a Minute: Inline Styles?

If you have dabbled with CSS before you may be thinking at this stage, "But this isn't the right way to do it," to which I say "Just wait a short while—all will be explained soon." We just need to run through these basics first before approaching the best way of doing this.

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
   jumps 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 be messy. There are cleverer ways to apply styling, as we'll see very soon.

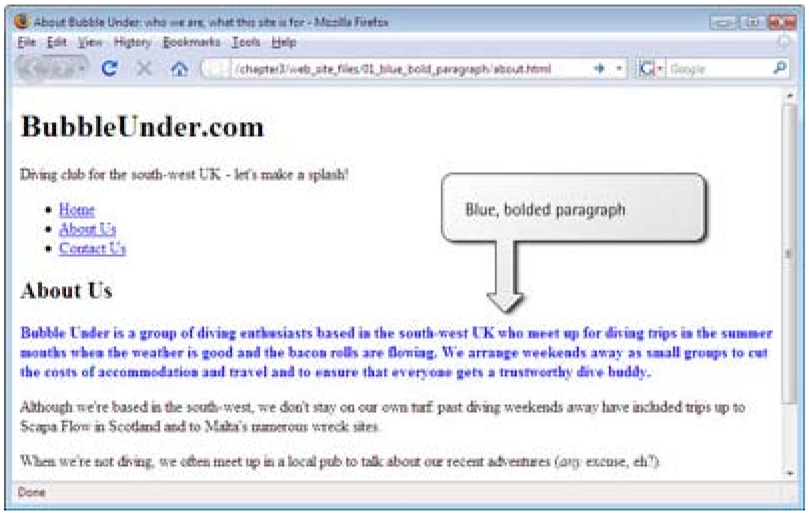
Adding Inline Styles

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 your markup looks like that shown here, save `about.html` and take a look at it in your browser. It should appear like the page shown below.


About Bubble Under: who we are, what this site is for - Mozilla Firefox
File Edit View History Bookmarks Tools Help
/chapter3/web_site_files/01_blue_bold_paragraph/about.html
Google
Blue, bolded paragraph
Done

BubbleUnder.com

Diving club for the south-west UK - let's make a splash!

- Home
- About Us
- Contact Us

About Us

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 away have included trips up to Scapa Flow in Scotland and to Malta's numerous wreck sites.

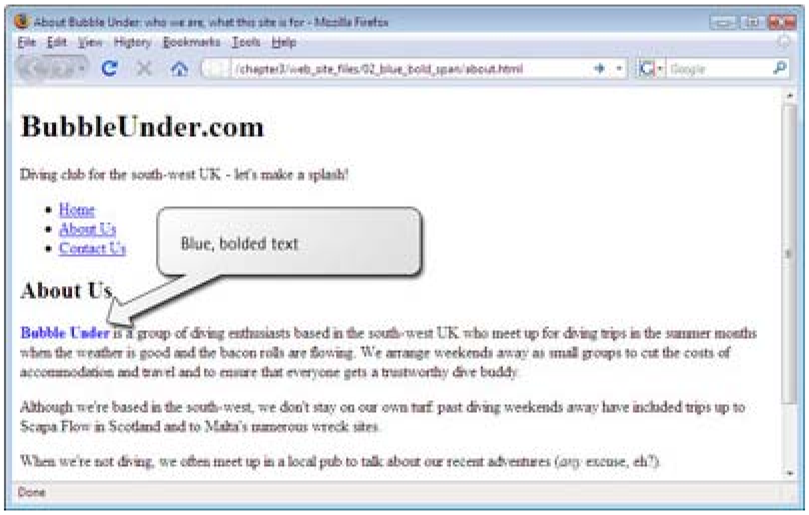
When we're not diving, we often meet up in a local pub to talk about our recent adventures (*any* excuse, eh?).

The span Element

You can easily color a whole paragraph like this, but more often than not, you'll want to pick out just 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 a section to which you want to apply a style.[4] 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
   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 below.


About Bubble Under: who we are, what this site is for - Mozilla Firefox
File Edit View History Bookmarks Tools Help
/chapter3/web_site_files/02_blue_bold_span/about.html
Google
Blue, bolded text
Done

BubbleUnder.com

Diving club for the south-west UK - let's make a splash!

- Home
- About Us
- Contact Us

About Us

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 away 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?).

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 jumps 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">jumps</span> over the lazy dog.</p>
```

Embedded Styles

Inline styles are a simple, quick way to apply some CSS effects 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 sheet* is a section you add to the start of a web page 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 elements 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 instructs the browser to apply all the declarations between the curly braces to certain elements. The selector, curly braces, and declarations combine to form 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:

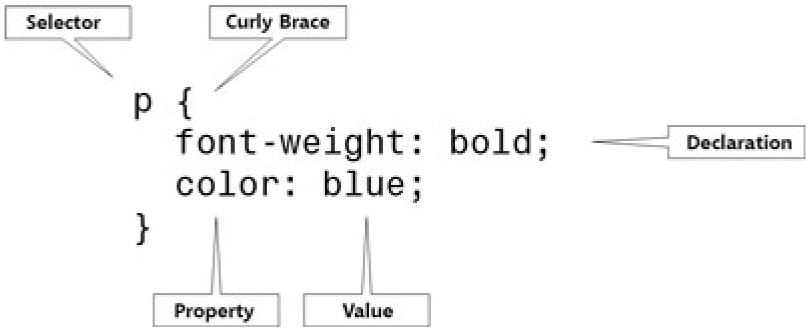


Figure 3.3. The anatomy of a rule
The anatomy of a rule

Why Embedded Styles Are Better than Inline Styles

In the example provided in Figure 3.3, “The anatomy of a rule”, 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 paragraphs text to red, you need only to change it in the style sheet at the top of the page:

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

That’s efficiency for you. 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 was 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 were slow to adopt CSS-based design in the past). Some browsers can choke on a few of the more advanced CSS techniques, but, by and large, you can style your web pages using 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:

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 perform exactly the same function: 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 using 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:

```
<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 earlier in this article: it does not have separate start and end tags. `link` is a complete element in its own right, and ends using 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. Go ahead, give it a try.

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 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 use Times New Roman as the default—it’s 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:

```
/*
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 the following:

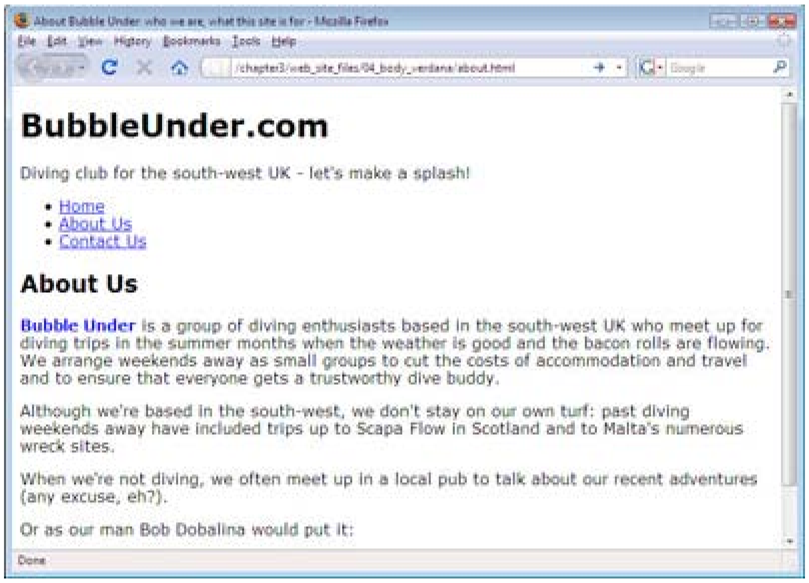
```
body {
 font-family: Verdana, Helvetica, Arial, sans-serif;
}
```

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. The figure below shows the appearance of our newly-styled homepage.


About Bubble Under: who we are, what this site is for - Mozilla Firefox
File Edit View History Bookmarks Tools Help
/chapter3/web_site_files/04_body_verdana/about.html
Google
Done

BubbleUnder.com

Diving club for the south-west UK - let's make a splash!

- Home
- About Us
- Contact Us

About Us

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 away 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?).

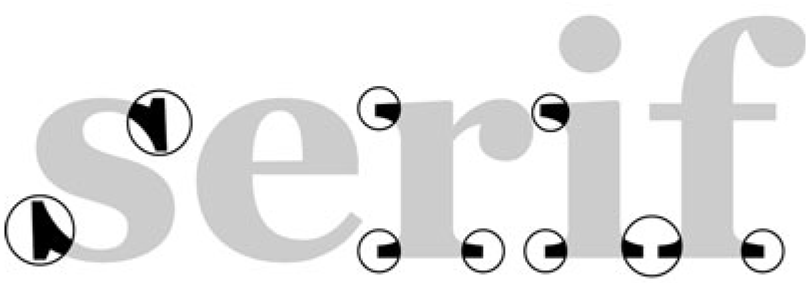
Or as our man Bob Dobalina would put it:

Tip: Sans-serif Fonts: Better for On-screen Viewing

A serif font is one that has all those little flourishes at the ends of each letter. These flourishes, which are shown below, "Highlighting the serifs of a serif font (Georgia)", 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 (from French, translating as "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`*.*



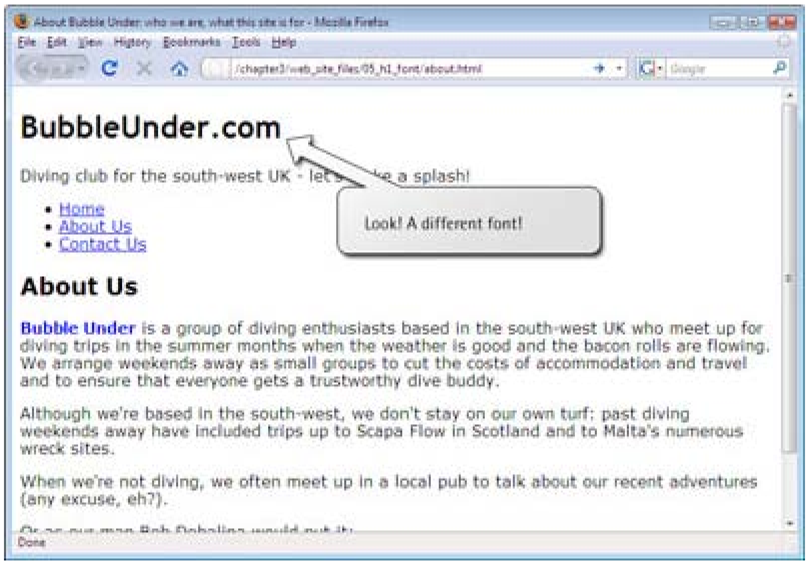
Stylish Headings

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.


About Bubble Under: who we are, what this site is for - Mozilla Firefox
File Edit View History Bookmarks Tools Help
/chapter3/web_site_files/05_h1_font/about.html
Look! A different font!
Done

BubbleUnder.com

Diving club for the south-west UK - let's make a splash!

- Home
- About Us
- Contact Us

About Us

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 away 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?).

Note: Some Font Names Deserve Quotes

In the code 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 inside 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.

Tip: 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. Caching is when your browser accesses files previously saved to the hard drive when you visit a given web page, rather than downloading new files each time. For example, you enter the URL, and the browser pulls the page stored in its cache. This speeds up the process of displaying a web page that has been loaded before. Unfortunately, your cache can soon become out-of-date, and when that happens, the page you visit might not display the most recent data.

This happens most frequently with images, but it can also occur using CSS files. The good news is that you have control over your browser's cache settings. Therefore, the amount of space the cache takes up on your hard disk before cached content is replaced with newer data can be adjusted. You can poke around your browser's settings for terms 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.

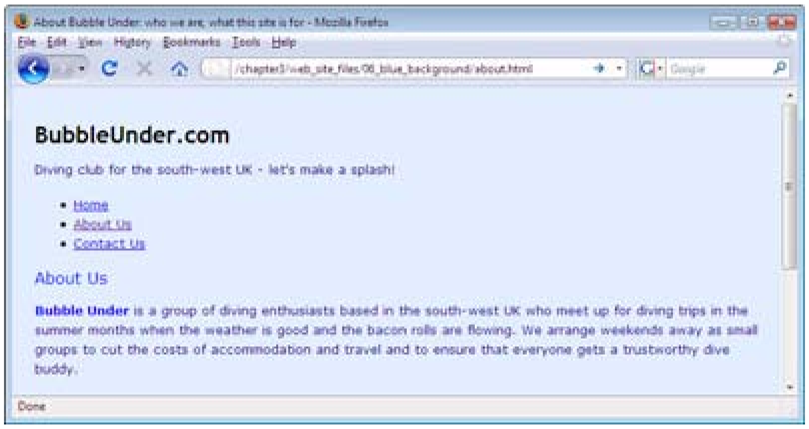
To stop the caching problem and force the browser to retrieve the most up-to-date version of your CSS file, simply hold down the ***Shift*** *key and click on the Refresh (or Reload) icon on your browser's toolbar.*

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):

```
/*
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;
}

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

Save the CSS file, then click **Reload** (or **Refresh**) in your browser. Hopefully, you'll be looking at a page like the one shown in Figure 3.7, "Applying subtle changes to the CSS that affects the display of the font".


About Bubble Under: who we are, what this site is for - Mozilla Firefox
File Edit View History Bookmarks Tools Help
/chapter3/web_site_files/06_blue_background/about.html
Google
Done

BubbleUnder.com

Diving club for the south-west UK - let's make a splash!

- Home
- About Us
- Contact Us

About Us

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.

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:

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

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 such as `navy`, `blue`, `red`, `yellow`, and so on. These are easy to remember and spell, but you can be limited 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 the code for it:

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

It's hardly intuitive, is it? This obscure-looking reference (`#e2edff`) translates to a light shade of blue. You could not, as a beginner, begin to guess that this would be a light blue. Thankfully there are numerous tools on the Web that let you choose a color from a chart (often called a *color picker*), then give you the code to match. Take a look at some of these tools,[5] and you'll soon be able to find the hexadecimal numbers you need to create your ideal color schemes.

Note: 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…
```

Eh? What's happening here? Well, as you can see, after we reach 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, 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 42, can be represented in hexadecimal. And Table 3.1, "Decimal to Hexadecimal Conversion" proves it!

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 amounts of red, green, and blue to create a single dot on its screen.[6] 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 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—which 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.

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, Shaping Up Using CSS.

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 won't take up too much space on your screen!

Now, to the paragraph styles:

```
p {
 font-size: small;
 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 appropriate color of navy.

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

```
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. As we've already seen—there are many ways to skin a cat using 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.

We've also introduced a new rule 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!).

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

The `h2` element also receives a minor makeover:

```
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 (and 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`, or `green`) or hexadecimal color specifications such as `#ff0000`.

font-family

This property takes a list of font names, 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 using 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`

relative font sizes

- a percentage (e.g. `140%`)
- em units (e.g. `1.2em`; `1em` is equal to the height of the M font character)

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`

Tip: Backing It Up!

Before you experiment using 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.

Recap: the Style Story so Far

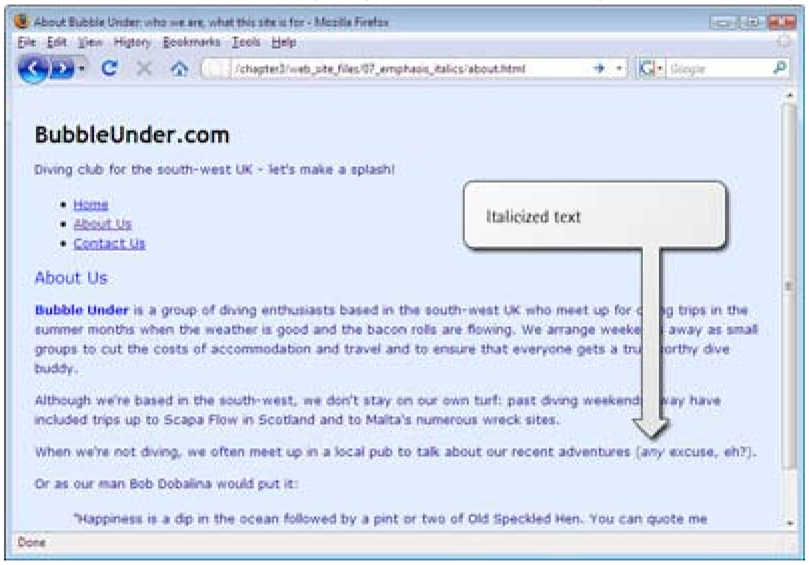
Let's allow ourselves a moment to reflect. Our site now boasts a CSS file using a selection of attractive styles. We're in the enviable position of being able to change the site at a whim by altering just that one CSS file. Let's try styling some more of the elements on our web pages.

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: `<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 below. As you can see, emphasis elements appear in italics by default. We're going to use CSS to change that default style.


About Bubble Under: who we are, what this site is for - Mozilla Firefox
File Edit View History Bookmarks Tools Help
/chapter3/web_site_files/07_emphasis_italics/about.html
Google
Italicized text
Done

BubbleUnder.com

Diving club for the south-west UK - let's make a splash!

- Home
- About Us
- Contact Us

About Us

Bubble Under is a group of diving enthusiasts based in the south-west UK who meet up for d g trips in the summer months when the weather is good and the bacon rolls are flowing. We arrange weeke away as small groups to cut the costs of accommodation and travel and to ensure that everyone gets a tru orthy dive buddy.

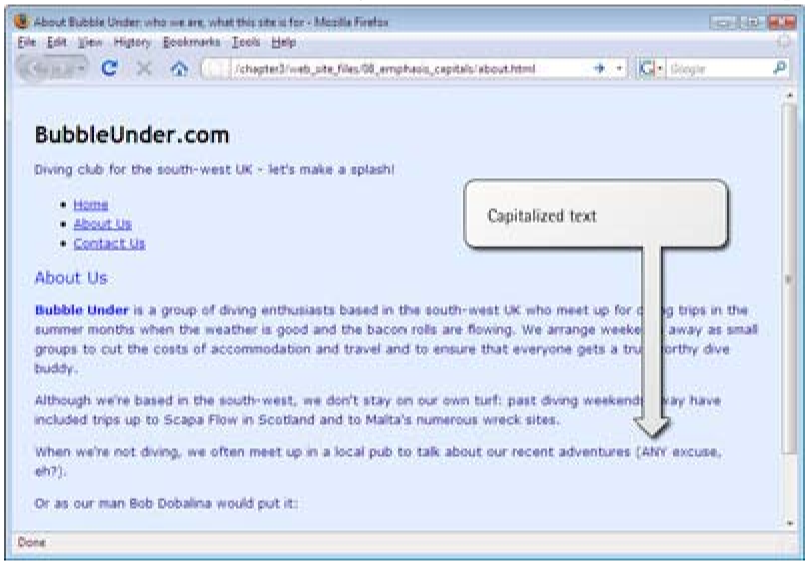
Although we're based in the south-west, we don't stay on our own turf: past diving weekend way have included trips up to Scapa Flow in Scotland and to Malta's numerous wreck sites.

When we're not diving, we often meet up in a local pub to talk about our recent adventures (*any* excuse, eh?).

Or as our man Bob Dobalina would put it:

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

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


About Bubble Under: who we are, what this site is for - Mozilla Firefox
File Edit View History Bookmarks Tools Help
/chapter3/web_site_files/08_emphasis_capitals/about.html
Google
Capitalized text
Done

BubbleUnder.com

Diving club for the south-west UK - let's make a splash!

- Home
- About Us
- Contact Us

About Us

Bubble Under is a group of diving enthusiasts based in the south-west UK who meet up for d g trips in the summer months when the weather is good and the bacon rolls are flowing. We arrange weeke away as small groups to cut the costs of accommodation and travel and to ensure that everyone gets a tru orthy dive buddy.

Although we're based in the south-west, we don't stay on our own turf: past diving weekend way have included trips up to Scapa Flow in Scotland and to Malta's numerous wreck sites.

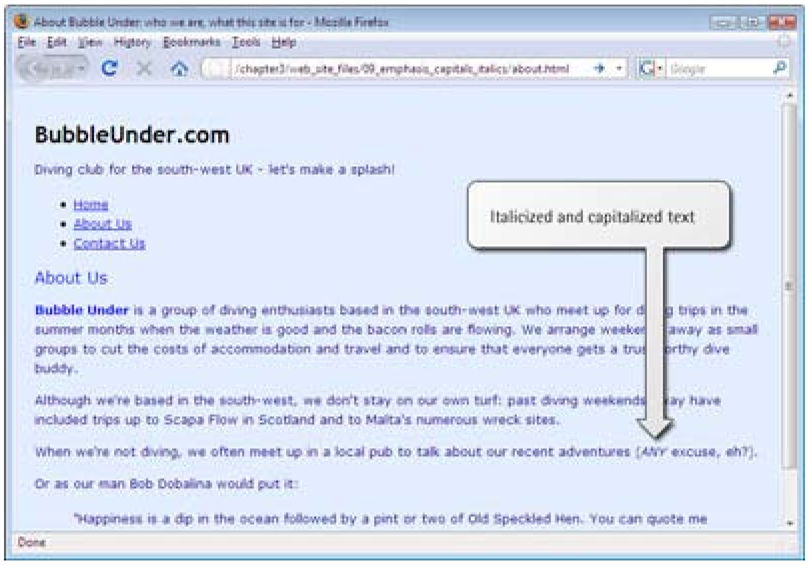
When we're not diving, we often meet up in a local pub to talk about our recent adventures (ANY excuse, eh?).

Or as our man Bob Dobalina would put it:

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 redefining the styles is a good idea? Aren't italics a suitable style for emphasis? They probably are. As they say in the Spider-Man film, "With great power comes great responsibility," so be sure to exercise caution. Just because you *can* change a default style doesn't 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 below:

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


About Bubble Under: who we are, what this site is for - Mozilla Firefox
File Edit View History Bookmarks Tools Help
/chapter3/web_site_files/09_emphasis_capitals_italics/about.html
Google
Italicized and capitalized text
Done

BubbleUnder.com

Diving club for the south-west UK - let's make a splash!

- Home
- About Us
- Contact Us

About Us

Bubble Under is a group of diving enthusiasts based in the south-west UK who meet up for d g trips in the summer months when the weather is good and the bacon rolls are flowing. We arrange weeke away as small groups to cut the costs of accommodation and travel and to ensure that everyone gets a tru orthy dive buddy.

Although we're based in the south-west, we don't stay on our own turf: past diving weekends ay 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?).

Or as our man Bob Dobalina would put it:

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

Tip: Emphasis or 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 text 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, use CSS.*

Why is this important? It might not seem a big deal as you look at the italicized text in your web browser. But imagine if you were blind, and you used software that read web pages aloud to you, instead of displaying them on the screen. This program (called a screen reader) might read text marked up with an `em` *element using slight emphasis, and text marked up with* `strong` *in a more powerful voice (though this, of course, depends on the screen reader being used). 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 not treat them in any special way—thus potentially losing the meaning that you were trying to convey. A search engine (e.g. Google, Yahoo) may also place more importance on a user's search terms that appear within a* `strong` *element than a* `b` *element (although the search engine companies never give anything solid away about how their search algorithms work!).*

One other presentational tag that you might see others use, but should never copy, is the `u` *element. Wrap this around some text and needless underlining occurs that only serves to confuse users. This is because in web pages, underlined text normally signifies a link—which the* `u` *element most definitely isn't!*

Looking at Elements in Context

Here's a riddle for you: 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 big, solid pen. If, however, you're an office worker, you'd opt for the sheep being the larger of the two—after all, you'd find it a lot easier to pick up a pen and flip it around your fingers.

Context can change a situation quite drastically, and we can use context 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:

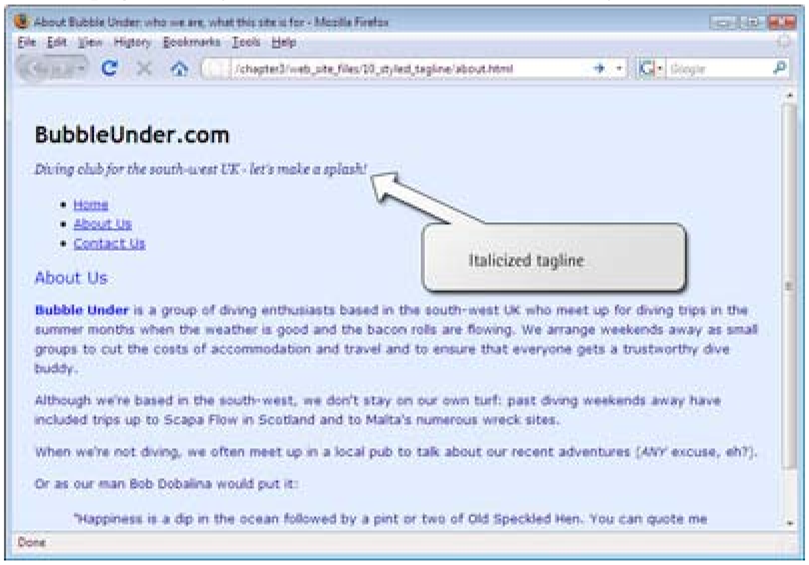
```
body {
 font-family: Verdana, Helvetica, Arial, sans-serif;
}

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:

`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`—can be useful. 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: `#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 similar to the one shown below. 

About Bubble Under: who we are, what this site is for - Mozilla Firefox

/chapter3/web_site_files/10_styled_tagline/about.html

BubbleUnder.com

Diving club for the south-west UK - let's make a splash!

- Home
- About Us
- Contact Us

About Us

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 away 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?).

Or as our man Bob Dobalina would put it:

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

Done

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."

Note: 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, for the body text.
- `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 repetitive. Plus, once you have a lot of styles on the page, it is 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;
}
```

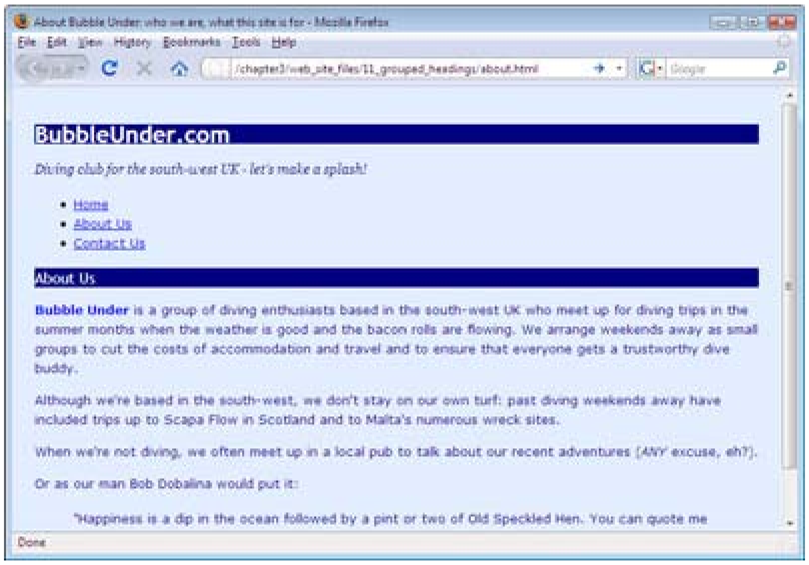
Translation: if the element is a level one heading, a level two heading, or a level three heading, set the text to yellow and the background to black.

Tip: Comma = “Or”

You can think of the commas in CSS selectors (like the one above) as the word “or.”

Let’s try grouping some styles in our project site. We don’t have any `h3` headings yet, but we will soon.

- Edit your CSS file (`style1.css`) by adding the following to the bottom of it: `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. You should be looking at a page like the one shown in Figure 3.12, “Displaying the changed heading styles”.



About Bubble Under: who we are, what this site is for - Mozilla Firefox

File Edit View History Bookmarks Tools Help

/chapter3/web_site_files/11_grouped_headings/about.html

BubbleUnder.com

Diving club for the south-west UK - let's make a splash!

- Home
- About Us
- Contact Us

About Us

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 away 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?).

Or as our man Bob Dobalina would put it:

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

Done

That CSS really does kill several birds with one stone (figuratively speaking, of course; I did say no animals would be harmed!). 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.

Tip: 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 sea animals, and so on?” That’s a fair question, but it’s important to remember with 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`. It’s not quite right, 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.

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:

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

...

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!

Recapping Our Progress

Time for another breather. What have we learned? 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 see that those links were underlined, which remains the case today. By default, a browser uses the following color scheme for links:

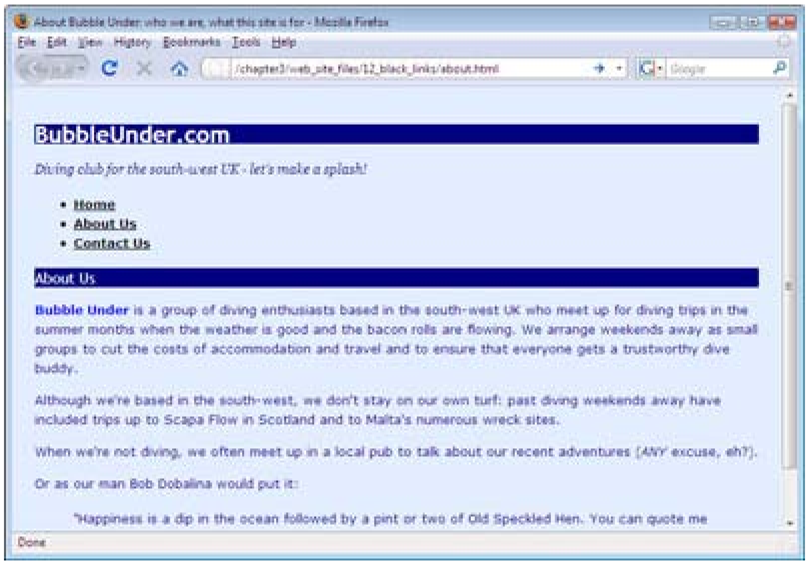
- **blue**: an unvisited link
- **purple**: a link to a web page that you've previously visited
- **red**: an active link (one you're clicking on; you may have noticed links flash red momentarily when you initially click on them)

This color scheme isn't to everyone's taste, but it's what we're stuck with for 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, then save it and see how it affects your web pages—Figure 3.13, “Styling all the links in our navigation to bold and black” illustrates this.



About Bubble Under: who we are, what this site is for - Mozilla Firefox
File Edit View History Bookmarks Tools Help
/chapter3/web_site_files/12_black_links/about.html
Google
Done

BubbleUnder.com

Diving club for the south-west UK - let's make a splash!

- **Home**
- **About Us**
- **Contact Us**

About Us

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 away 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?).

Or as our man Bob Dobalina would put it:

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

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 these link states using *pseudo-classes*, which sounds complicated but really is fairly straightforward. You can think of a pseudo-class as being like an internal class the browser automatically applies to the link while it’s in a certain state. 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 adding a colon (`:`) and the pseudo-classes `link`, `visited`, `hover`, and `active`. Adding pseudo-classes to your style sheet means the browser applies the rule when the element is in the state specified by the pseudo-class.

Tip: Getting Your Link States in Order

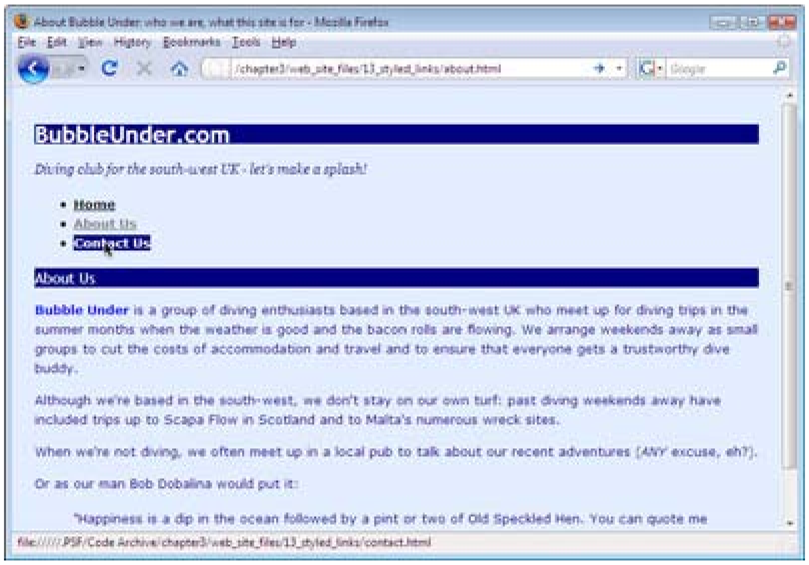
Browsers usually aren't fussy about the order in which you specify rules in your CSS file, but links should always be specified in the order shown above: `link`, `visited`, `hover`, and `active`. Try to remember the letters LVHA. The more cynical users might find it easier to remember this mnemonic with the phrase, "Love? Ha!" We can thank Jeffrey Zeldman for that little gem. (Designing With Web Standards, Jeffrey Zeldman, New Riders.)

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

- 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.

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

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.


About Bubble Under: who we are, what this site is for - Mozilla Firefox
File Edit View History Bookmarks Tools Help
/chapter3/web_site_files/13_styled_links/about.html
Google

BubbleUnder.com

Diving club for the south-west UK - let's make a splash!

- Home
- About Us
- Contact Us

About Us

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 away 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?).

Or as our man Bob Dobalina would put it:

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

file:///////PSF/Code Archive/chapter3/web_site_files/13_styled_links/contact.html

Tip: 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 under Browsing History that reads Delete. Click on this and it will bring up a Delete Browsing History dialog with more options, as shown in Figure 3.15, "Clearing the history in IE displays unvisited link styles again"; make sure that the History checkbox has been ticked, then click the Delete button on that dialog. Afterwards, reload the web page. Any links you may have visited will now appear as unvisited. (The process for clearing history in older versions is a little easier—just look for the button on the Internet Options dialogue that says Clear History).*

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

In order to make use of the style once it has been added to your style sheet, 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:

`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, it could look 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 earlier).

Tip: `class` vs `id`

So far, we've looked at both `class` *selectors (which involve periods: ".") and* `id` *selectors (which involve pound or hash 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.*

Tip: 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! Take a look:*

```
.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!*

Tip: 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. Determining which rule is the most specific is a little bit complex, but understandable. In our style sheet, the specificity is easy to determine: the `.fun` *rule is applied to the* `blockquote` *element and properties are inherited by the* `p` *elements, but property values are only inherited in the absence of any other declaration. We have another color declaration in our project site—the one that we created at the start of the chapter that states that all paragraphs should be navy-colored:*

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

The rule with the element selector `p` *has a greater specificity for the* `p` *elements because the selector specifically targets* `p` *elements, whereas the* `.fun` *rule does not. Imagine if, however, we added another rule like this:*

```
.fun p {
 color: green;
}
```

The effect would be this: all paragraph text would be navy, except for the paragraphs inside elements with the `class fun`*, which would be green. This is because the* `.fun p` *selector is more specific for those paragraphs. 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 be 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 great depth in the SitePoint CSS Reference [20] if you'd like to explore it further.

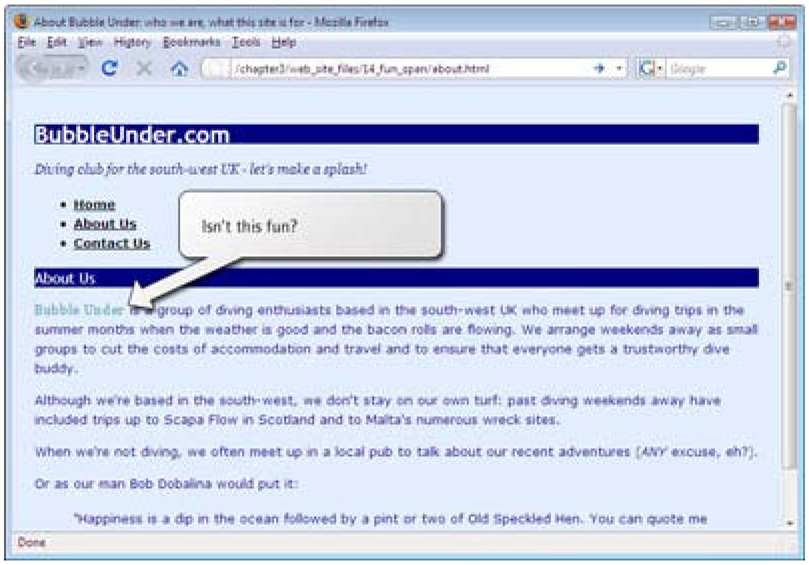
Styling Partial Text Using `span`

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 Us" page as suggested in the above XHTML code. If you save the changes and check them in your browser (remember to hit Reload), your page should look like the one shown below.


About Bubble Under: who we are, what this site is for - Mozilla Firefox
File Edit View History Bookmarks Tools Help
/chapter3/web_site_files/14_fun_span/about.html
Google
Isn't this fun?
Done

BubbleUnder.com

Diving club for the south-west UK - let's make a splash!

- Home
- About Us
- Contact Us

About Us

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 away 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?).

Or as our man Bob Dobalina would put it:

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

Note: Don't Throw (Needless) `span`s into the Works

The `span` element is nearly always used with a class attribute. There's 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.

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, it's advisable not to do 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! For that, you'll need your own copy of /http://www.sitepoint.com/books/html1//#lt#*Build Your Own Web Site The Right Way Using HTML & CSS* [21]. After the next chapter, on CSS, we explore the tasks of using images on your web site, the appropriate use of tables, building forms that promote user interaction, publishing your site to the Web, adding a blog to your site, and hooking up to ready-made tools like statistics packages, search functions, and discussion forums that you can apply to your site. As I mentioned, the book comes with a complete code archive, so you can apply the files we use in the book straight into your own sites (makes life easy, eh?).

Check out the book's Table of Contents [22] for the specifics. I look forward to seeing your site on the Web soon!

Back to SitePoint.com

[1] http://www.sitepoint.com/books/html2/
[2] http://www.sitepoint.com/books/html2/toc.php
[3] http://www.sitepoint.com/books/html2/samplechapters.php
[4] http://www.notetab.com/
[5] http://www.mozilla.com/firefox/
[6] http://www.barebones.com/products/textwrangler/
[7] http://www.adobe.com/products/photoshop/
[8] http://www.adobe.com/products/photoshopelwin/
[9] http://www.adobe.com/products/photoshopelmac/
[10] http://www.adobe.com/products/fireworks/
[11] http://www.corel.com/paintshoppro/
[12] http://picasa.google.com/download/
[13] http://www.sitepoint.com/forums/
[14] http://reference.sitepoint.com/html/doctypes/

[15] http://reference.sitepoint.com/html/html/
[16] http://reference.sitepoint.com/html/head/
[17] http://reference.sitepoint.com/html/title/
[18] http://reference.sitepoint.com/html/
[19] http://www.cs.tut.fi/~jkorpela/www/click.html
[20] http://reference.sitepoint.com/css/specificity/
[21] http://www.sitepoint.com/books/html1//#lt#*Build Your Own Web Site The Right Way Using HTML & CSS*
[22] http://www.sitepoint.com/books/html2/toc.php