1

Introducing the Site

In what has been a relatively short life to date, the Web has grown at a tremendous pace. When I first started learning HTML, I would not have imagined that so many people would be using the Web today. Nor, while sitting at my desktop PC, would I have imagined that by now I would need to write pages that could be accessed through such a variety of devices — such as mobile phones and TV set top boxes. These new devices are quite different from the desktop PC — they have differently sized screens and different amounts of power and memory available to them, and they enable users to access information in very different ways. Given the growth of the Web and the way in which it has changed, it is hardly surprising that those of us who build Web sites might need to update our skills, and that the tools we use to get the job done also need modernizing.

This chapter describes why it is important to learn to write sites using XHTML and CSS, and why it is so important to make your sites accessible. You will also be introduced to the example site that you will be working on throughout this book.

In this chapter, you will:

- Examine some of the problems caused by traditional HTML
- Find out why it is important to separate styling from content
- Meet the example First Promotions site. This is the site that you will be updating throughout the book.
- Learn about the aims of redesigning the site
- Find out more about the benefits that you will accrue by creating accessible sites in XHTML and CSS.

By the end of the chapter, you will understand the reasons why you need to update your skills and sites, and what you will be doing to the example site in this book.

Problems with HTML

As the Web evolved, so did HTML. Several versions of HTML have been released by the W3C (the World Wide Web Consortium — the body responsible for maintaining many Web standards). The first versions of HTML enabled you to use markup to describe the structure of a document, but did not offer much control over how that document looked. By the time HTML 4 was released, it had become a much more complex language than its original version. The features that were first added to HTML over the years afforded Web designers a lot of control over the appearance of a page. For example, they enabled Web page authors to control the exact width of tables (to the pixel); the size and weight of fonts; the colors and images used as backgrounds; and so on.

The addition of these new rules that offered designers control over the appearance of pages meant that it was possible to create attractive pages that worked great on your average desktop computer. However, problems soon started to manifest:

- Various browsers developed different ways of doing the same thing, which meant sites would not look or work the same in all browsers.
- Different users had different sized screens and different screen resolutions, so pages would not look the same on all computers.
- New devices started to access the Web; it was no longer just desktop PCs, but also TV set top boxes, mobile phones, and so on, and each new device had different capabilities. For example, a page containing a table that was 700 pixels wide would not fit on a phone that had a display only 128 pixels wide.
- If you (or your boss) wanted to change some aspect of the site design, such as the colors or fonts used, every page had to be changed because this information was in the HTML for every page.
- As designers used more HTML code to control the layout of a page, the source code of the page became longer and more complicated to write. This in turn introduced greater scope for errors when authoring or editing a site.
- If you did not have 20/20 vision, it could be hard to read some of the text on the sites, and if you had severe vision impairments, you would not be able to navigate some sites at all.

As a result, some companies started to create several versions of their Web sites, with versions for different browsers and devices; they even created text-only versions of sites intended to meet disability requirements. Obviously, this means a lot of extra work, and often results in versions of the site that are not as up-to-date as others. Clearly, this solution is far from ideal.

To summarize, traditional HTML posed the following problems:

- As HTML developed, a lot of stylistic rules were introduced to the language that enabled Web page authors to control the appearance of their pages; however, these same enhancements to the language meant that the rules governing how the page should appear were mixed in with the actual content of the page, and that many pages would work as intended only on desktop PCs.
- Because the Web is such a powerful medium, it needs to be made accessible to the widest range of users as possible. That includes those with disabilities who might not be able to read, hear, or move a mouse as well as others can.

Therefore, you can see that the changes you have to make to the way you build sites merely reflect changes in the Web's popularity and how people use it. In this chapter, you will see how and why the languages for writing Web pages have changed.

Design

After HTML 4.0, the W3C (which is responsible for developing the HTML specification) decided that it was time for a large-scale overhaul in the way people write Web sites. The problems described in the preceding section indicated to the W3C that these changes needed to reflect the following simple fact:

> **Different users require different presentations of the same information.**

Three common reasons why you would need to present the information on your Web site in different ways are as follows:

- Different pieces of technology used to view the Web have different capabilities, and they are not all capable of displaying a single design adequately for all users.
- The information made available online can often be very useful in other formats, such as in print or on a projector as part of a conference.
- Some people have difficulty accessing information in a way that others might find easy; for example, those with visual impairments might not be able to read text on a computer screen.

This last point is especially important because the Web is fundamentally changing the way in which people go about their everyday lives — from being able to shop, bank, and pay bills online to the introduction of electronic voting — so you cannot exclude sections of the community who have disabilities.

> *Indeed, the Web can have a more profound effect upon the lives of people with disabilities, some of whom may find leaving their home harder than those without any disabilities. Therefore, it is essential that advances in technology include everyone.*

The W3C's solution to this problem was to strip out all the *presentational* markup that had been added to HTML over the years to control how pages looked; this is the markup such as the `<font>` and `<center>` elements, and the `bgcolor` and `color` attributes. The HTML markup that was left described the structure and semantics of a document, such as the title of a document, where paragraphs start and end, what text represents a heading, and so on. This did not mean that Web designers would lose control over how their documents appeared; rather, they would use something called a *style sheet*, which is a separate document, to indicate how the page should be displayed. Indeed, style sheets have several other advantages described throughout the chapter.

The result of this separation of style and content led to the W3C's development of two standards, which the browser manufacturers are expected to support and the Web page authors are expected to learn: *XHTML* and *CSS*. Don't let this put you off, however. In reality, both languages are very similar to the HTML you already know, which makes learning these technologies rather quick and simple.

Before looking at how the languages of the Web have changed, the following section clarifies a few key terms to ensure that your understanding of them matches mine. I then want to elaborate a little on how and why we ended up where we are today, learning these new languages, and the differences between presentational and structural markup.

Clarifying terminology

The following terms might seem obvious to you, but people often refer to the same things by slightly different names. A quick review of these definitions will ensure that you understand any concepts presented in the following discussion before you start updating your skills:

A *tag* is a set of characters surrounded by angled brackets; for example, here is a familiar *opening tag*:

```
<td>
```

Meanwhile, here is its corresponding closing tag. Anything in between these tags is used as a table cell (or *table data* — hence, the letters `td`):

```
</td>
```

An *element* refers to both an opening tag and a closing tag, plus anything between them. For example, here is a table cell element:

```
<td>234.5</td>
```

The bit between the opening tag and the closing tag is referred to as *element content*.

Elements can "carry" *attributes*, which provide further information about the element that carries them. Attributes always sit inside the opening tag of an element. For example, here is the `align` attribute:

```
<td align="right">
```

This attribute indicates that the content of this table cell should be aligned to the right.

Some elements do not have a closing tag; for example, the `<img>` element can carry several attributes, but does not have any element content — there is nothing between an opening and closing tag:

```
<img src="/images/120x80_logo.gif" alt="Acme Logo" width="120" height='80">
```

Elements that have no content are called *empty elements*. You will see in Chapter 2 that empty elements are written differently in XHTML, so you will have to start writing all of your `<img>`, `<br>`, and `<hr>` tags in a new way.

Figure 1-1 illustrates all of these concepts.

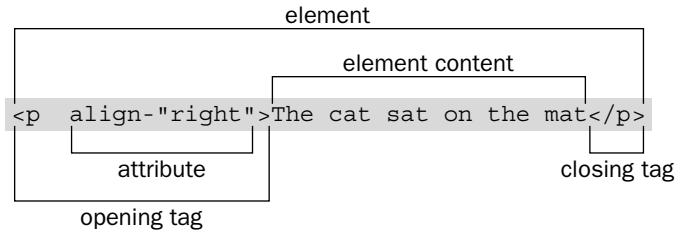

Figure 1-1

A Little Background to HTML

Removing all of the styling rules from HTML might seem like a backward step, rather than an exciting new progression of the language — you may well be wondering why we didn't have separate style sheets from the start if they are such a great idea. To understand why HTML evolved the way it did — that is, why HTML introduced all this stylistic markup such as the `<font>` elements and `bgcolor` attributes if they were only going to be removed, you need to look at how HTML developed into the language you know and use today. As is often the case, the answer appears obvious with hindsight — if only we could have foreseen that the problem would occur in the first place.

HTML was originally designed as a markup language to describe the *structure* and *semantics* of a document. The elements and attributes of HTML were supposed to indicate things such as the title of a document, what part of the text was a heading, what of the text was a paragraph, what data belonged in a table, and so on. In its earliest form, the Web was intended to transmit scientific documents so that the research community had quick and easy access to published work. Here is an example of a document that might have been generated for the scientific community (see the file `ch01_eg01.html` for the download):

```
<html>
  <head>
    <title>The Effect of the Internet on Psychological Theories of Self</title>
  </head>
  <body>
    <h1>The Effect of the Internet on Psychological Theories of Self</h1>
    <h2>Abstract</h2>
      <p>This paper looks that the role in which Internet users can adopt different
      online personae, and the effects this has on psychoanalytic theories of
      self and personal identity.</p>
      <p>While psychologists have long suggested that our concept of self should
      reflect a single, unitary, rational self, many people will adopt online
      personae that are very different than that which they display in person.</p>
      ...
  </body>
</html>
```

While this kind of document was highly practical, and enabled academics to share information with far greater ease than relying on printed journals, the presentation of these documents was rather gray and uninteresting, as you can see in Figure 1-2, which shows you what this document would look like in a browser.

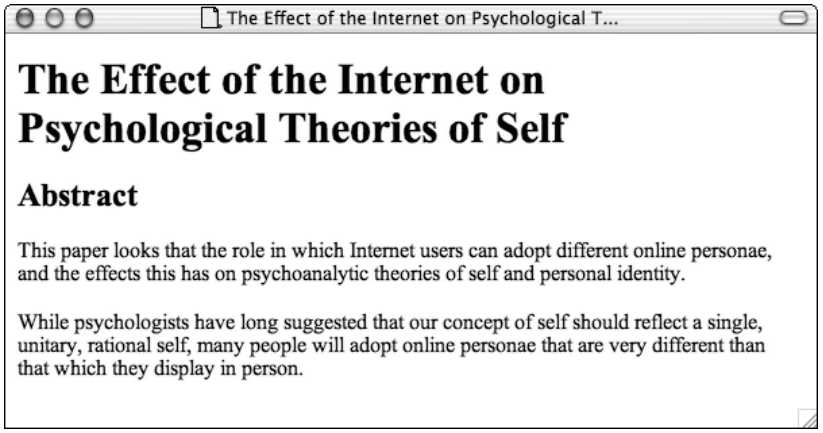

Figure 1-2

At the time, many people were quick to see the potential of the Web. These users started creating Web pages for all different kinds of purposes, from personal home pages to huge corporate sites advertising a company's products and services. It wasn't long before Web page authors wanted to control how their sites looked, and they expected the same level of control over their pages' appearance as print designers had over their creations. For example, Web designers wanted to be able to control the fonts and colors used in documents, and where their text would appear on a page. As a result, the major browser manufacturers started adding new elements and attributes to control the appearance of Web pages; both Netscape and Microsoft were desperately trying to win a greater share of the browser market by giving Web designers more control over pages shown in their browsers. The W3C also introduced many of these elements and attributes into successive versions of HTML.

This new markup is known as *presentational* or *stylistic* markup, because it affects the way that pages look. (It does not describe the structure and semantics of the document, which was the initial intention of HTML.) Prime examples of this new type of markup are the `<center>` and `<font>` elements and the `bgcolor` and `color` attributes.

Designers were quick to learn tricks that enabled them to control the layout of pages (the positioning of content on the screen) in a similar way to print designers. Two common techniques included the use of tables as layout grids for pages and the use of transparent single-pixel GIF images (commonly known as the single pixel or transparent GIF trick) to position elements.

The following example shows the same page you just met in `ch01_eg01.html`, but this time it has had stylistic markup added to control its presentation. The new file, called `ch01_eg02.html`, is much longer with the stylistic markup added to the page:

```
<html>
  <head>
    <title>The Effect of the Internet on Psychological Theories of Self</title>
  </head>
  <body bgcolor="#000000">
    <center>
    <table width="650" border="1" bordercolor="#666666" cellpadding="10"
           cellspacing="0">
```

```
        <tr>
          <td bgcolor="#999999">
            <font face="Arial, Helvetica, sans-serif" size="5" color="#FFFFFF">
              <h1>The Effect of the Internet on Psychological Theories of Self</h1>
            </font>
          </td>
        </tr>
        <tr>
          <td bgcolor="#EFEFEF">
            <font face="Arial, Helvetica, sans-serif" size="4" color="#000066">
              <h2>Abstract</h2>
            </font>
            <font face="Arial, Helvetica, sans-serif" size="2" color="#333333">
              <p>This paper looks that the role in which Internet users can adopt
              different online personae, and the effects this has on psychoanalytic
              theories of self and personal identity.</p>
              <p>While psychologists have long suggested that our concept of self
              should reflect a single, unitary, rational self, many people will adopt
              online personae that are very different than that which they display in
              person.</p>
            ...
            </font>
          </td>
        </tr>
      </table>
      </center>
    </body>
  </html>
```

Even though the layout of the document is still quite basic, the size of the page has ballooned; the number of characters (not including spaces) has almost doubled, with `ch01_eg01.html` containing 529 characters and `ch01_eg02.html` containing 928 characters. This extra complexity increases the chances of making an error, both when authoring the page in the first place and when making any alterations to the pages. Figure 1-3 shows you what `ch01_eg02.html` looks like in the browser.

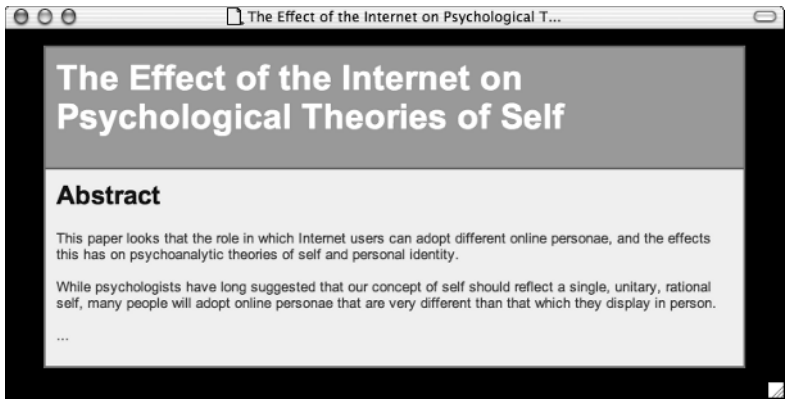

Figure 1-3

When the rules that govern the presentation of the Web page are intermingled with the actual content of the pages, changing any aspect of the presentation of your site (such as the typeface used for headings or the background color of the page) means changing *every* page of the site. When you consider that many pages are littered with `<font>` tags and a whole host of presentational attributes, it becomes even harder to find the markup you want to change. Therefore, you can see why the stylistic markup introduces more scope for error.

At the same time that the browser manufacturers were writing their browsers to understand new markup, they were also trying to make it easy for authors to write pages, so they would incorporate code that helped users display pages even if the HTML contained errors. This meant that a lot of HTML authors got into bad habits; even some of the leading authoring tools generated sloppy code. Furthermore, because the browser manufacturers were making the browsers capable of rendering pages even if they contained errors, the amount of disk space and memory required to run a browser had to increase.

None of these developments were inherently bad. Indeed, if it had not been possible to create attractive pages, and if it had not been as easy as to write Web pages as it is today, then it is unlikely that the Web would have ever become as popular as it is now. As you have already seen, however, with all of the extra stylistic markup and the tricks that designers were employing to control the layout of their sites, Web pages were becoming much larger. The markup no longer just described the structure of the page or provided information about its content; it also contained a lot of markup that was there only to control the presentation of the page for one kind of device: the desktop PC.

Browser issues

The problems associated with writing pages that contain rules governing how a page should look were exacerbated by the fact that the way in which people viewed Web sites was changing just as rapidly as the audiences were growing. As I have already hinted, advances in technologies meant the following:

- Computer processors and graphics cards improved.
- Higher-resolution screens became available.
- Larger screens fell in price and became more widespread.
- Devices other than desktop computers were accessing the Web, such as mobile phones and TV set top boxes; even refrigerators were being developed with built-in browsers.

Each of these advancements brought a new set of challenges:

- As processors and graphics cards improved, more users were encouraged to use higher screen resolutions.
- Because larger screens were becoming more widespread, more people were using these higher resolutions.
- Sites that were designed to fit on a screen at 640 × 480 resolution look smaller on a 1024 ×768 resolution screen and can therefore be harder to read at high resolutions.
- New devices often had smaller screens, lower resolution, less power available, and less memory available.

Most challenging were the new types of devices that were now accessing the Web, such as mobile phones, PDAs (personal digital assistants, such as the Blackberry and Palm Pilot), and set top boxes, because the screens could be very different from desktop PCs (not just slightly larger or with a bit better resolution). You certainly don't want to have to consider rewriting a Web site for each new kind of device that can access the Web.

The problems highlighted here are, once again, tied into the way HTML developed. If the content of the page were separated from the rules that govern how the page appears on a desktop PC, you could make the same content available to numerous devices by creating only one new set of presentational rules for each device that each page of the Web site could use (rather than rewriting every page again to be styled for a particular kind of device).

Of course, creating a new version of a site from scratch for each type of device that can access the Web is not a practical solution, and as you are about to see, you can learn a number of lessons by looking at how early mobile devices accessed the Web. It is fair to say that these events spurred the W3C's efforts to push forward the development of new versions of XHTML.

Lessons from the mobile world

When mobile phone manufacturers wanted to create Internet-enabled handsets, they knew that the HTML pages commonly available on the Web were not suitable for their devices. Not only were Web pages designed for screens much larger and more complex than theirs, but these phones had far less memory and power available to run them.

Memory was a key point because, as mentioned earlier, browsers designed to run on desktop computers had become bloated with code that enabled them to display pages even if they were not written correctly. When you consider that Netscape 7.2's system requirements are 26MB of hard disk space and 64MB of RAM, it is not hard to imagine that this is too much for your average mobile phone.

Because mobile phones presented such a different way of accessing the Web, the mobile phone manufacturers developed their own languages designed to make Web content available to their phones. This resulted in several competing languages that were designed to offer Web content on different mobile devices — examples include Wireless Markup Language (WML), which was part of the Wireless Application Protocol (WAP) group of specifications, Compact HTML, and Handheld Device Markup Language (HDML). When you look closely at all of these languages, however, you can see that they have something in common — they all act like a subset of HTML, and enable users to do the following:

- Create titles, headings, and text
- Link between pages
- Embed simple images
- Create basic tables
- Collect information from users via simple forms

What these languages lacked were the more advanced features of Web browsers, such as the capability to display complex layouts using tables and collect information using complex form controls, or the capability to display Flash movies and run powerful scripting languages.

These languages also require that the author write the page using a strict syntax — the browsers on these mobile devices were not as forgiving of sloppy coding as the major browsers on desktop computers.

By creating their own smaller languages, the mobile phone companies were able to create smaller browsers that would require less memory, eat up less power, and would therefore be able to run on their phones. With hindsight, you can see that they were all reinventing the wheel, when they could have just used a select few of the HTML elements and attributes, but this is not what happened.

At various points in the history of the Web, you might well have thought that each new device to access the Web was going to require its own new language that reflected the capabilities of that device. This would have meant creating several different versions of each and every page for each device. If you have ever struggled to get a site to work in both Internet Explorer *and* Netscape, then this idea would probably have filled you with dread. Furthermore, if each new device required its own language, then it would have been harder for these devices to gain acceptance, because content would have to be developed especially for them if they were going to be a success.

As you will see in the last chapter of this book, the W3C's solution has made it possible for all devices, both today and in the future, to use languages based upon XHTML; but you have a way to go before you can look at working with multiple devices.

> *In reality, when it comes to developing sites for different platforms, rather than creating static sites for each device, the pages are more likely to be dynamically generated from content in a database. Therefore, as you will see in this book, it is very important to ensure that the database holds XHTML, which can be repurposed for different devices.*

Accessibility

Many Web designers got into the habit of designing sites with pixel-precision accuracy. They used tables for layout whose width was measured in pixels; they used fixed-sized fonts; they commonly relied on a lot of images to make attractive layouts. This helped to ensure that the site looked the same on different computers (or at least on different browsers on desktop computers). However, it introduced a lot of problems for anyone who had less than 20/20 vision or perfect control over their mouse.

Many computer users with visual impairments use devices known as *screen readers* to read the content of the screen to them. These devices have complicated sets of keyboard shortcuts that enable the user to operate the software without necessarily seeing what is on the screen. Screen readers are just one of the many types of devices available to users with disabilities, and in many countries it is now a legal requirement that all Web pages are accessible to those with disabilities.

Some of the problems facing users with visual impairments when visiting Web sites include the following:

- Tables are used for layout, and often screen readers process tables in a way that makes pages hard to understand, reading things to users in the wrong order.
- Poor choices of color combinations can make it very hard to read text.
- Text can be specified in fixed-size fonts, making it impossible for users to increase the size when needed.

These problems, yet again, could be solved if the rules that indicate how a document should be presented were separated from the actual content of the document (and followed accessibility guidelines). For example, users with screen readers might want to remove all visual formatting (after all, they will not *see* the page) and let their software concentrate on the actual content of the page.

When it comes to accessibility, a number of other issues affect the way you go about designing and building Web pages, and these go beyond the issue of separating style from content. That is why two chapters are dedicated to the topic after you have learned all about XHTML and CSS. Here are some of the problems that users face, which are dealt with in those chapters:

- When dealing with forms, it is not always clear what information should be entered into a form control such as a text box, or which radio button should be selected. It is therefore important to label all form controls.
- If a page contains a large header (possibly with a lot of navigation items), a user with a screen reader will have to listen to all of this information before getting to the content of a page, which could be very tedious if the information is repeated on each page. It would be better if this information could be skipped.
- If images are used to represent text that is necessary for understanding the site and these images do not make use of the `alt` attribute (to provide a text alternative for those who cannot read what the image says), then users with screen readers will not know what the text on the image said.

Learning how to build sites in XHTML and CSS is the first step in creating accessible Web sites, so you should understand these basics before you continue on and learn about the range of other design considerations specifically related to accessibility issues, which are addressed in Chapters 6 and 7.

If you create an accessible site, you also reap other benefits (beyond those of meeting accessibility laws and helping those with disabilities):

- Increase the number of potential visitors to your site (and therefore potentially increase income), because the site is accessible to people it might not have been before.
- Make the content of your site accessible to those using your site in different situations. For example, if your site can be accessed by a screen reader, it is likely that it could also be used by an emerging market of voice browsers that can be used in different situations, such as while driving or jogging, when hands cannot be used to navigate using a mouse.
- Create code that will be available to a whole host of devices other than desktop PCs.

Separating style from content

At the very core of the problems you have been introduced to up to now is that most traditional HTML pages contained rules that controlled how they were presented. There was, therefore, a simple solution: Stop authors from mixing presentational (or stylistic) markup in with the markup that dictated the structure of the document.

The idea of *separating style from content* meant taking HTML back to its roots, when markup only described the structure and semantics of a document, not how it appeared. At first, these simple HTML

documents might seem like a step backward to the days when the Web was a rather gray and drab place; but in reality, the documents can be just as visually attractive.

Rather than put the rules that govern how a document should appear in the same document that holds the real content that people read, you put those rules in a separate document known as a *style sheet*. For example, a rule in the style sheet might indicate that all level 1 headings should be written in a dark blue, size 5, Arial font.

This approach has many other advantages. For example, several documents can share the same style sheet, making it possible to create a single set of rules to be used to style every page of a site. Conversely, each document could be attached to different style sheets to make the same content appear in a different manner.

If you still need convincing, here is a summary of the key advantages of separating style rules from content:

- The content has been freed up. It can be presented in a lot of different ways for different users.
- You have simpler source documents, which will be easier to write.
- When you have simpler source documents, you are less likely to create errors when editing the documents.
- You can create the style rules once and use them for each page of your site, rather than repeat them in each page.
- It becomes much easier to maintain the style of a site, not only because it acts as a template for all pages, but because it also enables you to change a font or color across the whole site by just altering the one style sheet.
- If you want your content to be made available on different devices, you only need to write a new style sheet for each different device, rather than rewrite the whole site for each device.
- Once the browser has automatically downloaded the style sheet when it accesses the first page that uses it, it will be quicker to download subsequent pages that use the same style sheet because the browser stores a copy of the style sheet, and subsequent pages will be smaller because they do not contain style rules.

As you can see, in addition to solving the problems discussed so far, the separation of style from content brings a lot of other advantages, too.

Putting the "X" in XHTML

At the same time that the W3C decided to remove stylistic markup from HTML, they also decided that they would go one step further and rewrite HTML in a language you may well have heard of: *XML*. XML is a language that is used to write markup languages (and is therefore sometimes referred to as a *meta-markup language*). When the W3C was making these changes, XML was gaining wide acceptance in all areas of programming; it can be used on any platform because (like HTML) it uses plain text to hold the data, and it has been one of the most widely used new technologies in the past decade.

Reformulating HTML in XML would prepare the language for the next decade and beyond. Therefore, to reflect the change, rather than release HTML 5.0, the W3C decided to highlight the new family of

documents by calling it XHTML (rather like Microsoft released Windows XP instead of Windows 2003, or Macromedia released Dreamweaver MX instead of Dreamweaver 6).

> *Several languages that you may have heard of are written in XML, including Scalable Vector Graphics (SVG), MathML (a language dedicated to writing mathematical equations), Extensible Stylesheet Language Transformations (XSLT), and XML Schemas. You can also find hundreds of business-orientated markup languages written in XML.*

Making HTML compliant with the rules of XML has many advantages. You will learn more about these advantages in Chapters 2 and 8, but they include the following:

- XML requires a stricter syntax than HTML did — as you will see in Chapter 2. This in turn has other advantages:
  - Browser manufacturers can write smaller browsers that can handle the XHTML pages. These smaller browsers are ideal for small devices that do not have as much memory or power available to them as desktop computers.
  - You can perform complex operations, processing, and transformation upon the data held within the XHTML page. This means that the data is no longer solely used for visual presentation.
- Many tools and languages have been written to work with XML, all of which are then available to XHTML documents, including tools such as XSLT and Simple API for XML (SAX) processors.

In October 1999, the W3C released XHTML 1.0 — this was the reformulation of HTML in XML syntax. As you will see in Chapter 2, there are actually three versions of XHTML 1.0:

- **Strict XHTML 1.0** — which also removed all the old stylistic markup
- **Transitional XHTML 1.0** — which enabled Web page authors to continue to use the stylistic markup of HTML 4.1 while adopting XML syntax. This was largely designed to support older browsers known as *legacy* browsers.
- **Frameset XHTML 1.0** — which is used to create frameset documents.

> *Don't worry if this sounds complicated; it will all become clear in Chapter 2. Strict XHTML 1.0 is just a subset of Transitional XHTML 1.0, while Frameset XHTML 1.0 introduces only a handful of different markup to create frames. Best of all, each of the elements and attributes should be familiar to you already from HTML — after all, XHTML is just the latest incarnation of HTML.*

The Story Behind CSS

Because Strict XHTML 1.0 documents contain no presentational markup, if you want them to look visually attractive, you have to link them to a style sheet that controls how the document is to be presented.

The W3C had already created a style sheet language that would be ideal for use with Web pages long before they came up with XHTML. *CSS* or *Cascading Style Sheets* were already being used by many HTML authors to control basic aspects of the style of documents, such as fonts and background colors.

CSS1 was actually released in December 1996, while CSS2, which expanded upon CSS1, was released in May 1998. Because Web designers were already using CSS to control the appearance of Web pages, and because of its benefits, it was the obvious choice for use with XHTML.

One thing that makes CSS relatively easy to learn is that its so-called *properties*, which control how the content of an element is displayed, are very similar to the attribute names you have used in HTML. However, CSS is also a very powerful language that enables you to do much more than is possible with the basic stylistic markup of HTML. For example, CSS offers the following:

- Very fine-grained control over the presentation of a page
- Control over the layout of a document without relying on tables (as you will see in Chapter 5)
- Properties for presenting a document in paged media (what most people would call print)
- Properties for aural versions of documents — which may be used by users with visual impairments or anyone on the move who cannot look at a screen

Chapters 3, 4, and 5 ensure that you can create attractive layouts using CSS.

Using Style Sheets

To solve the problems associated with viewing a Web site in different browsers and on different devices and to make sites more accessible, you have seen that the best approach is to separate the markup that describes the structure of a document's content from the rules that indicate how it should be displayed.

Separating design from content suggests the following:

- You are going to have to take your HTML skills back to basics and forget about the stylistic markup that was added to later versions of HTML. The main content of a page can now be written in Strict XHTML, which contains only markup to describe the structure and semantics of a document.
- The rules governing how a page should be displayed are written in a separate document to which the XHTML page links. This document is a style sheet written in CSS.

It is hard to look at examples in depth without learning more about each of the relevant languages. However, the following example is a version of the document you met in `ch01_eg01.html` written in XHTML. It is followed by a CSS style sheet used to control presentation of the page.

As you can see in this example, even the table (which was used for layout purposes) has been removed and replaced with `<div>` elements, which act as grouping elements to which styles can be attached (`ch01_eg03.html`):

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!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 Effect of the Internet on Psychological Theories of Self</title>
```

```
    <link rel="stylesheet" type="text/css" href="ch01_eg03.css" />
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
  </head>
  <body>
    <div class="page">
      <div class="heading">
            <h1>The Effect of the Internet on Psychological Theories of Self</h1>
      </div>
      <div class="body">
            <h2>Abstract</h2>
            <p>This paper looks that the role in which Internet users can adopt
            different online personae, and the effects this has on psychoanalytic
            theories of self and personal identity.</p>
            <p>While psychologists have long suggested that our concept of self
            should reflect a single, unitary, rational self, many people will adopt
            online personae that are very different than that which they display in
            person.</p>
            ...
      </div>
    </div>
 </body>
</html>
```

The following code represents the accompanying style sheet (`ch01_eg03.css`). Do not worry if this looks complicated at first; when you start to look at the language in Chapter 3, you will notice that a lot of it reflects features you learned with HTML attributes:

```
body {
  background-color:#000000;
  font-family: arial, verdana, sans-serif;}

div.page {
  width:650px;
  border-style:solid; border-width:1px; border-color:#666666;}

div.heading {
  background-color:#999999;
  padding:10px;}

div.body {
  background-color:#EFEFEF;
  padding:10px;}

h1 {
  font-size:22pt;
  color:#000066;}
h2 {
  font-size:18pt;
  color:#000066;}
p {
  font-size:14pt;
  color:#000000;}
```

While it might seem like a hassle to write a style sheet like this for one document, it can really save time if you are creating numerous pages that use the same styles; after all, you do not have to add the presentation rules to each document that uses them.

As you go through this book, you will see in more detail how the solution of separating style from content has many advantages. Indeed, in the future, as new devices are invented that need new capabilities, this solution can be extended — rather than having to create a new languages for each new device.

Furthermore, if you design your Web pages according to the XHTML and CSS standards, it should come as a relief to you that you will also be able to view them in older browsers that were written before these standards came out.

Introducing the Sample Site

Throughout this book, you are going to be working with one example site. The site is a fictional company that sells items known as "promotional" or "corporate" gifts, such as pens, bags, note pads, and stress reliever toys that have company names or logos on them. This next section introduces you to the site, including how it is organized and coded. You will be updating the site throughout subsequent chapters.

The fictional company whose site you are working on is called First Promotions, and their catalog and price list form the main part of the site. The company does not take orders online because they need to obtain the customer's logo (in addition to the text that should appear on the merchandise) before they start on the order. This often requires working with designers to ensure that the artwork is supplied in the correct formats (it also gives the sales team the benefit of direct contact with the customer). Once the artwork has been received, the design has to be approved before anything is actually produced.

> *If you are familiar with a server-side language, such as ASP/JSP/PHP, this example should be highly relevant to you, even though the site is written in HTML. After all, each of these technologies sends HTML back to the browser, and therefore teaches you how to correctly write the code to send back to the client.*

For the purposes of this book, pretend that First Promotions built a site in the late 1990s. That is the site you are going to meet now; it certainly uses techniques that were commonplace back then, and the style could do with a makeover while you are at work on it.

You can test the site for yourself by going to `www.firstPromotions.co.uk` and selecting the option to view the original version of the site. The complete code for the original and finished versions of the site is also available for download (along with the rest of the code for this book) from `www.wrox.com`.

Figure 1-4 shows you the home page of the First Promotions site.

Figure 1-4

You can see from Figure 1-4 that the home page has a header and some navigation items underneath the logo, and then the rest of the page is divided into three columns. This is a very common layout.

As already mentioned, the main purpose of the site is to act as a catalog and price list for the company. You can see the sections of the catalog in the left-hand column — this is referred to as *category navigation*. Each of these links will take you to a product list page, which shows summary information for all of the items in that category. You can see an example of a product list page in Figure 1-5, which shows the product list page for the Bags category.

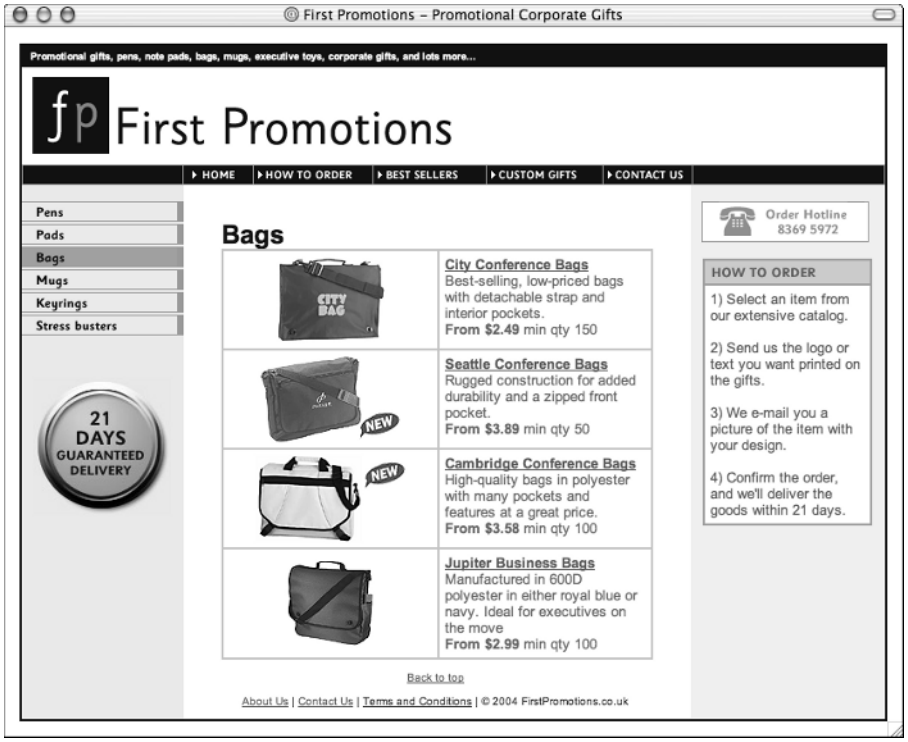

Figure 1-5

This page follows the same three-column layout that the home page does, and the list pages for all categories are identical in all but the products they contain.

When users click on the image or title of an item, they are taken to an individual product page. You can see an example of one of these product pages in Figure 1-6. As you can see from this screenshot, the product pages use a two-column layout. By removing the third column, more space is available to show the product in detail.

Across the top of each page, and at the bottom of the main column, you can see additional navigation links to pages that contain information on topics such as how to contact the company, how to place an order, information about the company, and so on. These pages all follow the same structure as the home page, with a three-column layout. The only exception is the best sellers page, which uses the same structure as the product list pages.

In summary, the site contains three main types of pages:

- Home page
- Product lists page
- Product details page

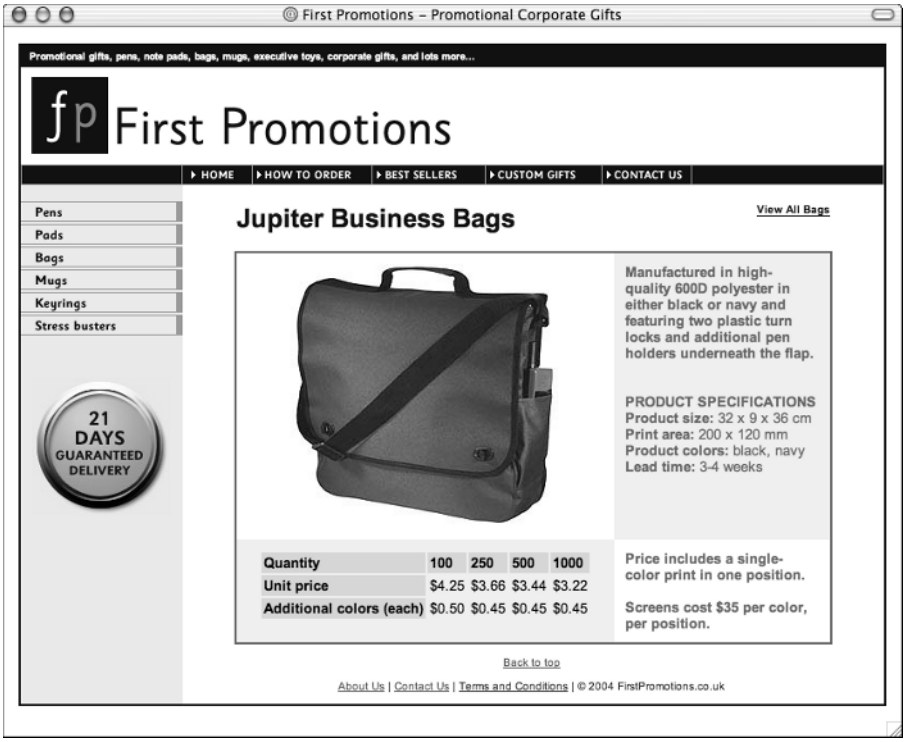

Figure 1-6

Every page follows one of these three designs. The following sections describe each one of these in a little more detail. You do not need to study every line of code in minute detail, but you should get a good idea of how the site is built because you will be coming back to it throughout the rest of the book. The code for the original version of the site is also reproduced here so that you can refer to it later as you work on the updates.

Home page

As with any site, the home page is the first that most visitors will see when they access the site. It tells users what they can expect to find on the site, and provides a base from which they can navigate the site.

Figure 1-7 shows a line drawing superimposed over the home page, indicating the location of the tables that control the layout of the page.

The whole page is written inside a single-cell table, which fixes the width of the table and draws a border around the page. Inside this table are three other tables.

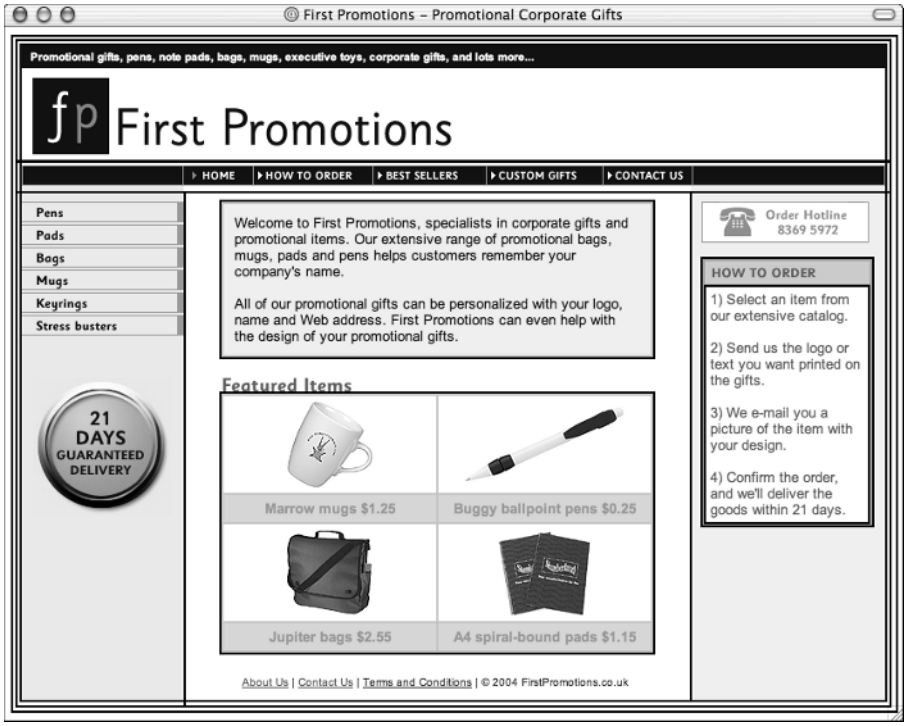

Figure 1-7

At the top you can see the first table containing the heading, with the company logo and aim of the site. Beneath this is a second table containing links to individual pages that feature other information, such as how to order products, contact details, and the best-selling items. The third table contains the main body of the page. This third table has three columns:

- Column 1 contains the navigation for different sections (and the guarantee stamp image)
- Column 2 contains the main content of the page, which is what the user came to see
- Column 3 contains additional information, such as ordering details

Here is the code for the home page (`index.html`), which starts out as you might expect:

```
<html>
<head>
  <title>First Promotions - Promotional Corporate Gifts</title>
```

The following script is generated by Macromedia Dreamweaver when you add a rollover image:

```
  <script language="JavaScript" type="text/JavaScript">
<!--
function MM_swapImgRestore() { //v3.0
  var i,x,a=document.MM_sr; for(i=0;a&&i<a.length&&(x=a[i])&&x.oSrc;i++)
  x.src=x.oSrc;
}

function MM_preloadImages() { //v3.0
  var d=document; if(d.images){ if(!d.MM_p) d.MM_p=new Array();
  var i,j=d.MM_p.length,a=MM_preloadImages.arguments; for(i=0; i<a.length; i++)
  if (a[i].indexOf("#")!=0){ d.MM_p[j]=new Image; d.MM_p[j++].src=a[i];}}
}

function MM_findObj(n, d) { //v4.01
 var p,i,x;  if(!d) d=document; if((p=n.indexOf("?"))>0&&parent.frames.length) {
  d=parent.frames[n.substring(p+1)].document; n=n.substring(0,p);}
 if(!(x=d[n])&&d.all) x=d.all[n];
 for (i=0;!x&&i<d.forms.length;i++) x=d.forms[i][n];
 for(i=0;!x&&d.layers&&i<d.layers.length;i++) x=MM_findObj(n,d.layers[i].document);
 if(!x && d.getElementById) x=d.getElementById(n); return x;
}

function MM_swapImage() { //v3.0
  var i,j=0,x,a=MM_swapImage.arguments; document.MM_sr=new Array;
  for(i=0;i<(a.length-2);i+=3)
   if ((x=MM_findObj(a[i]))!=null){document.MM_sr[j++]=x; if(!x.oSrc) x.oSrc=x.src;
        x.src=a[i+2];}
}
//-->
  </script>
</head>
```

On the `<body>` element, you can see several stylistic attributes such as the `bgcolor` and `link` attributes. The `onLoad` attribute is generated by Dreamweaver (as was the script) to preload rollover images:

```
<body bgcolor="#FFFFFF" alink="#0000CC" vlink="#003366" link="#0066CC"
  onLoad="MM_preloadImages('images/interface/nav_order_on.gif',
  'images/interface/nav_bestSellers_on.gif','images/interface/nav_custom.gif',
  'images/interface/nav_contactUs_on.gif','images/interface/nav_pens_on.gif',
  'images/nav_nav_pads_on.gif','images/interface/nav_bags.gif',
  'images/interface/nav_mugs_on.gif', 'images/interface/nav_stress_on.gif')">
```

Now you get to the content of the page, which is held within a `<center>` element to make it appear in the middle of the page. The whole page is created inside one `<table>` element that is given a `width` of 800 pixels so that the width of the page is fixed, and it is surrounded by a single-pixel border. This table has just one cell, containing the rest of the page.

Inside the single cell of the table that contains the page are three other tables: one containing the masthead (or header) for the page, the second containing the navigation, and the third containing the main body of the page. First, here is the table containing the masthead for the page:

```
<center>
<table border="1" cellpadding="0" cellspacing="0" width="800"
       bordercolor="#000066"><tr><td>

<!-- masthead -->
<table border="0" cellpadding="5" cellspacing="0" width="800">
  <tr>
    <td bgcolor="#000066">
      <font face="Arial, Helvetica, sans-serif" size="6" color="#FFFFFF">
        <b> Promotional gifts, pens, note pads, bags, mugs, executive toys,
           corporate gifts, and lots more...</b>
      </font>
    </td>
  </tr>
  <tr>
    <td><img src="images/branding/logo_400x80.gif" alt="logo" border="0"></td>
  </tr>
</table>
```

The second table contains the navigation. As you can see, the table for the navigation contains code to create rollover images on each item in the top navigation bar:

```
<!-- navigation -->
<table border="0" cellpadding="0" cellspacing="1" width="800" bgcolor="6699FF">
  <tr>
    <td width="150" bgcolor="#000066"></td>
    <td bgcolor="#000066">
      <img src="images/interface/nav_home_on.gif" width="64" height="17"border="0">
    </td>
    <td bgcolor="#000066">
      <a href="order.html" onMouseOut="MM_swapImgRestore()"
         onMouseOver="MM_swapImage('HowToOrder','',
         'images/interface/nav_order_on.gif',1)">
        <img src="images/interface/nav_order.gif" alt="How to Order"
             name="HowToOrder" width="110" height="17" border="0">
      </a>
    </td>
    <td bgcolor="#000066">
      <a href="bestSellers.html" onMouseOut="MM_swapImgRestore()"
         onMouseOver="MM_swapImage('BestSellers','',
         'images/interface/nav_bestSellers_on.gif',1)">
        <img src="images/interface/nav_bestSellers.gif" alt="Best Sellers"
             name="BestSellers" width="104" height="17" border="0">
      </a>
    </td>
    <td bgcolor="#000066">
      <a href="custom.html" onMouseOut="MM_swapImgRestore()"
        onMouseOver="MM_swapImage('CustomGifts','',
        'images/interface/nav_custom_on.gif',1)">
        <img src="images/interface/nav_custom.gif" alt="Custom Gifts"
          name="CustomGifts" width="107" height="17" border="0">
      </a>
    </td>
```

```
      <td bgcolor="#000066">
          <a href="contact.html" onMouseOut="MM_swapImgRestore()"
             onMouseOver="MM_swapImage('ContactUs','',
             'images/interface/nav_contactUs_on.gif',1)">
            <img src="images/interface/nav_contactUs.gif" alt="Contact Us"
                 name="ContactUs" width="82" height="17" border="0">
          </a>
      </td>
      <td width="180" bgcolor="#000066"></td>
    </tr>
  </table>
```

The third and final table contains the main page. This table contains one row with three table cells; there is one cell for each column in the layout.

The first column contains the product navigation, which again contains rollover images just like the top navigation. To make the code readable, I have added white space between each element; however, in the download code, there are no spaces between the navigation links and spacer images (otherwise, you can find gaps between them in Internet Explorer):

```
  <!-- main page -->
  <table border="0" cellpadding="0" cellspacing="0" width="800">
    <tr>
      <td width="150" bgcolor="#D9ECFF" valign="top">
        <img src="images/interface/1px.gif" height="15" width="150">
        <a href="products/pens/index.html" onMouseOut="MM_swapImgRestore()"
            onMouseOver="MM_swapImage('Pens','',
            'images/interface/nav_pens_on.gif',1)">
          <img src="images/interface/nav_pens.gif" alt="Pens" name="Pens" width="150"
               height="19" border="0">
        </a>
        <img src="images/interface/1px.gif" height="2" width="150">
        <a href="products/pads/index.html" onMouseOut="MM_swapImgRestore()"
           onMouseOver="MM_swapImage('Pads','',
           'images/interface/nav_pads_on.gif',1)">
          <img src="images/interface/nav_pads.gif" alt="Pads" name="Pads" width="150"
               height="19" border="0">
        </a>
        <img src="images/interface/1px.gif" height="2" width="150">
        <a href="products/bags/index.html" onMouseOut="MM_swapImgRestore()"
           onMouseOver="MM_swapImage('Bags','',
           'images/interface/nav_bags_on.gif',1)">
          <img src="images/interface/nav_bags.gif" alt="Bags" name="Bags" width="150"
               height="19" border="0">
        </a>
        <img src="images/interface/1px.gif" height="2" width="150">
        <a href="mugs.html" onMouseOut="MM_swapImgRestore()"
           onMouseOver="MM_swapImage('Mugs','',
           'images/interface/nav_mugs_on.gif',1)">
          <img src="images/interface/nav_mugs.gif" alt="Mugs" name="Mugs" width="150"
            height="19" border="0">
        </a>
```

```
        <img src="images/interface/1px.gif" height="2" width="150">
        <a href="products/keyrings/index.html" onMouseOut="MM_swapImgRestore()"
           onMouseOver="MM_swapImage('Keyrings','',
           'images/interface/nav_keyrings_on.gif',1)">
          <img src="images/interface/nav_keyrings.gif" alt="Key rings"
               name="Key rings" width="150" height="19" border="0">
        </a>
        <img src="images/interface/1px.gif" height="2" width="150">
        <a href="products/stressbusters/index.html" onMouseOut="MM_swapImgRestore()"
           onMouseOver="MM_swapImage('StressBusters','',
           'images/interface/nav_stress_on.gif',1)">
          <img src="images/interface/nav_stress.gif" alt="Stress busters"
               name="StressBusters" width="150" height="19" border="0">
        </a>
        <img src="images/interface/1px.gif" height="40" width="150">
        <center>
         <img src="images/interface/guarantee.jpg" height="126" width="127">
        </center>
      </td>
```

The second column holds the main message of the page. The first thing you see in this column is an introductory paragraph telling users what the company does. This paragraph is held in its own table to make the text look like it is in a box of its own:

```
      <td width="470" bgcolor="#FFFFFF" valign="top">
        <center>
        <img src="images/interface/1px.gif" height="15" width="1">
          <table border="1" cellpadding="10" cellspacing="0" bordercolor="#C5C5C5"
                 bgcolor="#EFEFEF" width="400">
            <tr>
              <td>
                <font face="Arial, Helvetica, sans-serif" size="2" color="#000066">
                  Welcome to First Promotions, specialists in corporate gifts and
                  promotional items. Our extensive range of promotional bags,
                  mugs, pads and pens helps customers remember your company's
                  name.<br><br>

                  All of our promotional gifts can be personalized with your logo,
                  name and Web address. First Promotions can even help with
                  the design of your promotional gifts.</font>
              </td>
            </tr>
          </table>
        </center>
        <br>
```

Beneath the introductory paragraphs on the home page is a simple table containing some featured items. Note how the image that says "featured items" is indented using a single-pixel transparent GIF. These featured items link directly to pages that show the full details of the items in the catalog:

```
        <img src="images/interface/1px.gif" width="35" height="1">
        <img src="images/interface/featuredItems.gif">
        <center>
          <table border="1" cellpadding="5" cellspacing="0" bordercolor="#C5C5C5"
                 bgcolor="#D6D6D6" width="400">
            <tr>
              <td width="200" bgcolor="#FFFFFF" align="center">
                <a href="products/mugs/mug2.html">
                  <img src="products/mugs/images/mugs2_thumb.gif" border="0">
                </a>
              </td>
              <td width="200" bgcolor="#FFFFFF" align="center">
                <a href="products/pens/pen2.hml">
                  <img src="products/pens/images/pens2_thumb.gif" border="0">
                </a>
              </td>
            </tr>
            <tr>
              <td width="200" align="center">
                <font face="Arial, Helvetica, sans-serif" size="2" color="6699FF">
                  <b>Marrow mugs from $1.25</b>
                </font>
              </td>
              <td width="200" align="center">
                <font face="Arial, Helvetica, sans-serif" size="2" color="6699FF">
                  <b>Buggy ballpoint pens from $0.25</b>
                </font>
              </td>
            </tr>
            <tr>
              <td width="200" bgcolor="#FFFFFF" align="center">
                <a href="products/bags/bag4.html">
                  <img src="products/bags/images/bags4_thumb.gif" border="0">
                </a>
              </td>
              <td width="200" bgcolor="#FFFFFF" align="center">
                <a href="products/pads/pad3.html">
                  <img src="products/pads/images/pad3_thumb.gif" border="0">
                </a>
              </td>
            </tr>
            <tr>
              <td width="200" align="center">
                <font face="Arial, Helvetica, sans-serif" size="2" color="6699FF">
                  <b>Jupiter bags $2.55</b>
                </font>
              </td>
              <td width="200" align="center">
                <font face="Arial, Helvetica, sans-serif" size="2" color="6699FF">
                  <b>A4 spiral-bound pads $1.15</b>
                </font>
              </td>
            </tr>
          </table>
```

At the bottom of the center column, you can see the footer links:

```
        <font face="Arial, Helvetica, sans-serif" size="1" color="#0000CC">
         <br><br>
          <a href="about.html">About Us</a> |
          <a href="contact.html">Contact Us</a> |
          <a href="terms.html">Terms and Conditions</a> |
          &copy; 2004 FirstPromotions.co.uk
         <br><br>
        </font>
        </center>
      </td>
```

The third and final column contains the information about ordering that you can see on the right-hand side of the home page. This part of the page uses nested tables to create boxes around parts of the text:

```
      <td width="180" bgcolor="#EFEFEF" valign="top">
        <img src="images/interface/1px.gif" height="15" width="1">
        <center>
          <img src="images/interface/orderHotline.gif" width="155" height="38"> <br>
          <img src="images/interface/1px.gif" height="15" width="1">
          <table border="1" cellpadding="0" cellspacing="0" bordercolor="#FF9900"
                 width="155">
            <tr>
              <td>
                <img src="images/interface/howToOrder.gif" width="153" height="21">
              </td>
            </tr>
            <tr>
               <td bgcolor="#FFFFFF">
                 <table border="0" cellpadding="5" cellspacing="0">
                   <tr>
                     <td>
                       <font face="Arial, sans-serif" size="2" color="#904C2D">
                        1) Select an item from our extensive catalog.<br><br>
                        2) Send us the logo or text you want printed on the
                           gifts.<br><br>
                        3) We e-mail you a picture of the item with your
                           design.<br><br>
                        4) Confirm the order, and we'll deliver the goods within
                           21 days.<br>
                        </font>
                     </td>
                   </tr>
                 </table>
               </td>
             </tr>
           </table>
           <br><br>
        </center>
      </td>
    </tr>
  </table>
```

```
</td></tr></table>
</center>
</body>
</html>
```

As you can see when you browse through the site, the pages that you get to when using the top and the bottom navigation are of a very similar structure — the main content is the middle of an otherwise identical three-column layout — the only exception being the best sellers page, which follows the structure of the product list pages that you are about to meet.

Product list pages

Each of the six categories has its own folder in the directory structure of the site. Each of these folders has a file called `index.html` that contains a list of all the products in that category; hence, it is known as a product list page. This page uses a three-column layout, just like the home page, and indeed the only real difference between this page and the home page is the center column of the table that forms the main part of the page.

Let's start looking at this page inside the `<body>` tags. Because the logo and masthead, the top navigation, and the left navigation are virtually identical to those in the home page, I'll skip over them. There are only two notable differences:

- The image indicating that the user is in the Bags section of the site is "on."
- All URLs to images and links to other pages are preceded by `../../`, indicating that the relative URLs given are from the folder above the parent folder for this file.

```
<center>
<table border="1" cellpadding="0" cellspacing="0" width="800"
       bordercolor="#000066">
  <tr><td>

<table border="0" cellpadding="5" cellspacing="0" width="800">
<!-- LOGO AND MASTHEAD HERE -->
</table>

<table border="0" cellpadding="0" cellspacing="1" width="800" bgcolor="6699FF">
<!-- TOP NAVIGATION GOES HERE -->
</table>

<table border="0" cellpadding="0" cellspacing="0" width="800">
  <tr>
    <td width="150" bgcolor="#D9ECFF" valign="top">
<!-- LEFT NAVIGATION GOES HERE -->
    </td>
```

The real changes are in the middle column in the third table (which is the table that holds the main content of the page that changes). First, there are a couple of transparent single-pixel GIFs that position the heading in the desired place. The heading indicates which category the user is accessing — in this case, `Bags`:

```
      <td width="470" bgcolor="#FFFFFF" valign="top">
        <center>
          <img src="../../images/interface/1px.gif" height="15" width="1">
        </center><br>
        <img src="../../images/interface/1px.gif" height="1" width="35">
        <font face="Arial, Helvetica, sans-serif" size="5" color="#000066">
          <b>Bags</b>
        </font><br>
```

The main part of the product list page is a table that shows the different products available in the category. There is one row for each product, with an image of that product in the left-hand cell and a description of that product in the right-hand cell. Both the image and the title link to the page that contains details about that product:

```
        <center>
          <table border="1" cellpadding="5" cellspacing="0" bordercolor="#C5C5C5"
                 bgcolor="#D6D6D6" width="400">
            <tr>
              <td width="200" bgcolor="#FFFFFF" align="center">
                <a href="bag1.html">
                  <img src="images/bags1_thumb.gif" border="0">
                </a>
              </td>
              <td width="200" bgcolor="#FFFFFF" valign="top">
                <font face="Arial, Helvetica, sans-serif" size="2" color="#666666">
                  <b><a href="bag1.html">City Conference Bags</a></b><br>
                    Best-selling, low-priced bags with detachable strap and
                      interior pockets.<br>
                  <b>From $2.49</b> min qty 150
                </font>
              </td>
            </tr>
            <tr>
              <td width="200" bgcolor="#FFFFFF" align="center">
                <a href="bag2.html">
                  <img src="images/bags2_thumb.gif" border="0">
                </a>
              </td>
              <td width="200" bgcolor="#FFFFFF" valign="top">
                <font face="Arial, Helvetica, sans-serif" size="2" color="#666666">
                <b><a href="bag2.html">Seattle Conference Bags</a></b><br>
                  Rugged construction for added durability and a zipped front
                    pocket.<br>
                  <b>From $3.89</b> min qty 50
                </font>
              </td>
            </tr>
            <tr>
              <td width="200" bgcolor="#FFFFFF" align="center">
                <a href="bag3.html">
                  <img src="images/bags3_thumb.gif" border="0">
                </a>
              </td>
```

```
            <td width="200" bgcolor="#FFFFFF" valign="top">
                <font face="Arial, Helvetica, sans-serif" size="2" color="#666666">
                  <b><a href="bag3.html">Cambridge Conference Bags</a></b><br>
                    High-quality bags in polyester with many pockets and features
                      at a great price.<br>
                   <b>From $3.58</b> min qty 100
                </font>
              </td>
            </tr>
            <tr>
              <td width="200" bgcolor="#FFFFFF" align="center">
                <a href="bag4.html">
                  <img src="images/bags4_thumb.gif" border="0">
                </a>
              </td>
              <td width="200" bgcolor="#FFFFFF" valign="top">
                <font face="Arial, Helvetica, sans-serif" size="2" color="#666666">
                  <b><a href="bag4.html">Jupiter Business Bags</a></b><br>
                    Manufactured in 600D polyester in either royal blue or navy.
                      Ideal for executives on the move<br>
                  <b>From $2.99</b> min qty 100
                </font>
              </td>
            </tr>
          </table>
```

As with all of the pages, this column contains some footer links at the bottom:

```
        <font face="Arial, Helvetica, sans-serif" size="1" color="#0000CC">
        <!-- FOOTER LINKS GO HERE -->
        </font>
        </center>
      </td>
```

The rest of the page, including the right-hand column, has exactly the same content as the home page, so I won't repeat that here:

```
      <td width="180" bgcolor="#EFEFEF" valign="top">
      <!-- RIGHT COLUMN GOES HERE -->
      </td>
    </tr>
  </table>

  </td></tr></table>
  </center>
```

Remember that there are six product categories — pens, bags, mugs, pads, keyrings, and stress busters. Each category has its own folder in the directory structure. In the folder for each category is a file called `index.html`, which is the product list for that product category.

Product details page

When users click on any of the items on the product list (or the featured items on the home page), they will be taken to an individual product page containing the detailed information for that product, along with prices for the products.

Because more information needs to go on the product details pages than the other pages, you have only two columns in the main part of this page. Figure 1-8 shows the product details page for the Jupiter bag, with lines superimposed over the locations where tables control the layout.

Figure 1-8

As before, the entire page is contained inside a table that controls the width of the page and draws a border around the edge of it. Inside this table are three more tables, the first containing the masthead, the second the navigation, and the third the main body of the page. This third table contains only two cells, one for the left navigation and the one for the product details.

Now have a look at the code behind this page (`bag4.html`). The start of the page is the same as both the home page (except that the URLs start with `../../`) and the product list pages, so I'll start in the body of the page. As with the other pages, a table with a single cell holds the entire content of the page. Inside this cell are tables for the masthead and navigation:

```
<center>

<table border="1" cellpadding="0" cellspacing="0" width="800"
       bordercolor="#000066"><tr><td>

<table border="0" cellpadding="5" cellspacing="0" width="800">
  <!-- MASTHEAD -->
</table>

<table border="0" cellpadding="0" cellspacing="1" width="800" bgcolor="6699FF">
  <!-- TOP NAVIGATION -->
</table>
```

Next is the third table, which is the one that creates the columns in the main part of the page. As with the other pages, the first column (which is held in the first cell of this table) contains the navigation — you don't need to look at this again.

The second column is a lot wider at 650 pixels and provides a lot more room to display the details of the selected product:

```
<!-- main page -->
<table border="0" cellpadding="0" cellspacing="0" width="800">
  <tr>
    <td width="150" bgcolor="#D9ECFF" valign="top">
      <!-- LEFT NAVIGATION -->
    </td>
    <td width="650" bgcolor="#FFFFFF" valign="top">
```

Now you are in the cell that represents the main product information part of the page. This cell contains two tables — the first holds the title of the product and the link to all of the bags:

```
      <br>
      <center>
      <table border="0" cellpadding="0" cellspacing="0" width="550">
        <tr>
         <td>
           <font face="Arial, Helvetica, sans-serif" size="5" color="#000066">
             <b>Jupiter Business Bags</b>
           </font>
         </td>
         <td align="right" valign="top">
           <a href="/products/bags/index.html">
             <font face="Arial, Helvetica, sans-serif" size="1" color="#000066">
               <b>View All Bags</b>
             </font>
           </a>
         </td>
        </tr>
     </table><br>
```

Although it may not look like it, the second table contains just one cell. That is because it is a wrapper table, and it in turn contains another table. This single-celled table is used to create a line around the information it contains and is made up of two rows and two columns:

- Row 1, column 1 contains the image of the bag.
- Row 1, column 2 contains the description of the bag.
- Row 2, column 1 contains the prices.
- Row 2, column 2 contains printing information.

Let's start with the first row, which contains the image and description and is fairly straightforward:

```
<table border="1" bordercolor="#666666" cellpadding="0"
       cellspacing="0" width="550"><tr><td>

   <table border="0" cellpadding="10" cellspacing="0" bgcolor="#FFFFFF"
          width="550">
     <tr>
      <td width="350" bgcolor="#FFFFFF" align="center" valign="top">
        <img src="images/bags4.jpg" border="0">
      </td>
      <td width="200" bgcolor="#EFEFEF" valign="top">
        <font face="Arial, Helvetica, sans-serif" size="2" color="#666666">
          <b>Manufactured in high-quality 600D polyester in either black or
             navy and featuring two plastic turn locks and additional pen
             holders underneath the flap.</b>
          <br><br><br>
          <b>PRODUCT SPECIFICATIONS</b><br>
          <b>Product size:</b> 32 x 9 x 36 cm<br>
          <b>Print area:</b> 200 x 120 mm<br>
          <b>Product colors:</b> black, navy<br>
          <b>Lead time:</b> 3-4 weeks <br><br>
        </font>
      </td>
     </tr>
```

The following code shows the second row. The first column of row two contains more nested tables that control the layout of the prices for the products. The second column of row two just contains the printing information:

```
      <tr>
        <td bgcolor="#EFEFEF" align="center" valign="top">
          <table border="0" cellpadding="2" cellspacing="2">
            <tr>
              <td bgcolor="#D6D6D6">
                <font face="Arial, sans-serif" size="2"><b>Quantity</b></font>
              </td>
              <td bgcolor="#D6D6D6">
                <font face="Arial, sans-serif" size="2"><b>100</b></font>
              </td>
```

```
                <td bgcolor="#D6D6D6">
                  <font face="Arial, sans-serif" size="2"><b>250</b></font>
                </td>
                <td bgcolor="#D6D6D6">
                  <font face="Arial, sans-serif" size="2"><b>500</b></font>
                </td>
                <td bgcolor="#D6D6D6">
                  <font face="Arial, sans-serif" size="2"><b>1000</b></font>
                </td>
              </tr>
              <tr>
                <td bgcolor="#D6D6D6">
                  <font face="Arial, sans-serif" size="2"><b>Unit price</b></font>
                </td>
                <td><font face="Arial, sans-serif" size="2">$4.25</font></td>
                <td><font face="Arial, sans-serif" size="2">$3.66</font></td>
                <td><font face="Arial, sans-serif" size="2">$3.44</font></td>
                <td>
                  <font face="Arial, Helvetica, sans-serif" size="2">$3.22</font>
                </td>
              </tr>
              <tr>
                <td bgcolor="#D6D6D6">
                  <font face="Arial, sans-serif" size="2">
                    <b>Additional colors (each)</b>
                  </font>
                </td>
                <td><font face="Arial, sans-serif" size="2">$0.50</font></td>
                 <td><font face="Arial, sans-serif" size="2">$0.45</font></td>
                 <td><font face="Arial, sans-serif" size="2">$0.45</font></td>
                 <td><font face="Arial, sans-serif" size="2">$0.45</font></td>
               </tr>
             </table>
           </td>
           <td width="200" bgcolor="#FFFFFF" valign="top">
             <font face="Arial, Helvetica, sans-serif" size="2" color="#666666">
               <b>Price includes a single-color print in one position.<br><br>
               Screens cost £35 per color, per position.</b>
             </font>
           </td>
         </tr>
       </table>
     </td></tr></table>
```

Remember that this page has only two columns, so the bottom of the page finishes at the end of this cell, which still contains the footer links:

```
    <font face="Arial, Helvetica, sans-serif" size="1" color="#0000CC">
    <!-- FOOTER LINKS -->
    </font>
    </center>
  </td>
```

```
            </tr>
          </table>
        </td>
      </tr>
    </table>

    </td></tr></table>
    </center>
```

At this point you have seen the structure of every page on this site. Only the content of the pages differs, but it would probably help if you also took a look at the directory structure.

Site structure

Having looked at how all of the pages are written for the First Promotions site, it is helpful to see how the directory structure is set up. Figure 1-9 shows the corresponding folder structure for the site.

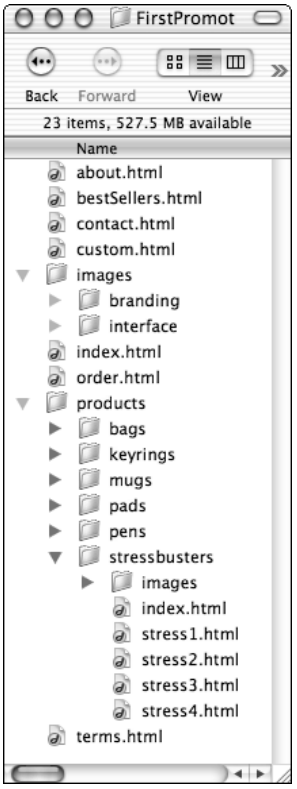

Figure 1-9

The main home page is called `index.html`, and it sits in the root directory, along with an `images` folder, a `products` folder, and the following pages:

Filename	Purpose
`aboutUs.html`	Introduces information about the company
`bestSellers.html`	Contains details about the best-selling items
`contact.html`	Contact information to get in touch with the company
`custom.html`	How to place orders for custom items that are not in the catalog
`order.html`	How to order items from the catalog
`terms.html`	Terms and conditions of using the site

Inside the `images` folder are two more folders. The first one is called `branding` and contains images such as the logo; the second folder is called `interface` and contains images used to create the user interface, such as the navigation buttons.

Inside the `products` folder are six subfolders — one for each of the six different product categories that First Promotions offers: pens, bags, mugs, pads, keyrings, and stress busters. Each of these folders contains the following:

- An `index.html` page that provides an overview of all the products in that category
- A folder called `images` that contains the photographs of the products in that category (there is a thumbnail and full-size copy of each image)
- A page for each product that provides more details about the product and pricing information

Before you continue with subsequent chapters, it would be helpful to either try the site out at `www.FirstPromotions.co.uk` or to download a copy from `www.wrox.com` along with the rest of the source code for this book and get it running on your own computer. This will help you in the coming pages as you look at the site in more detail.

Updating the site

Throughout the book, you will be making changes to the site as you learn each topic. By the end of the book, you will have a new site that is XHTML compliant, whose style and layout is controlled by CSS, and which meets the W3C and Section 508 accessibility guidelines. The design will also be more modern. Figure 1-10 shows a screenshot of the home page that you will end up with after all your efforts.

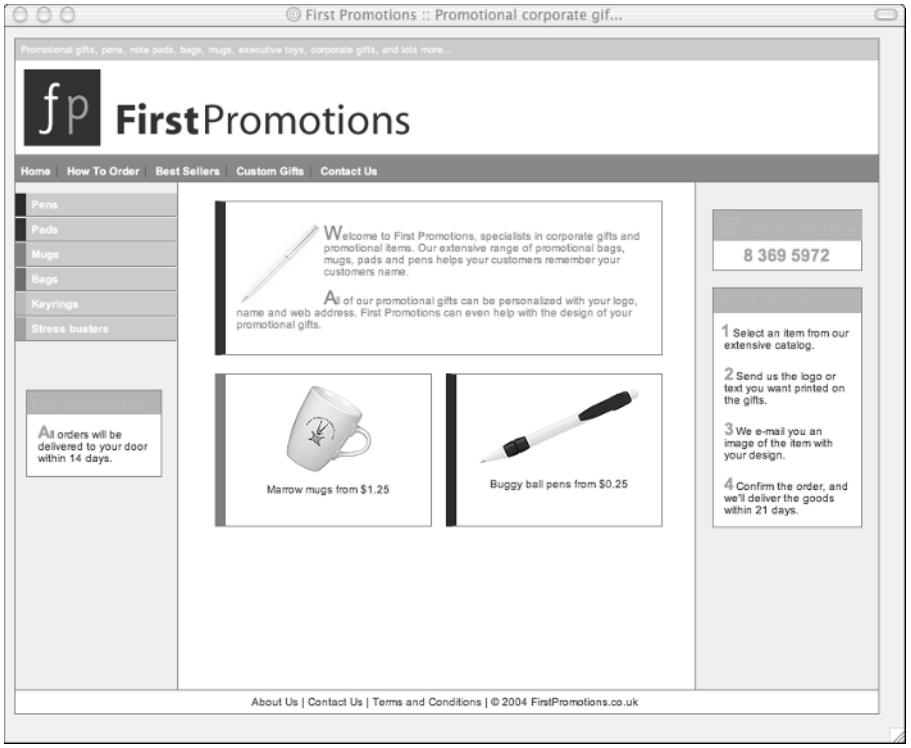

Figure 1-10

Your updating of the site will take the following course throughout the book:

1. While you look at XHTML (and the differences between HTML and XHTML), you will take the HTML structure that you have just been introduced to and change it to make the site XHTML-compliant. By the end of Chapter 2, all of the code will use XHTML syntax, and the presentational markup in the site will have been removed. As a result, the site will look a lot less styled, but that is okay because you will make it more attractive again in the chapters on CSS. The XHTML pages should also be easier to maintain because they will not be cluttered with presentational markup.

   Having learned how to write XHTML pages, you will be able to write content that can be served to devices that use XHTML browsers and don't necessarily support all of the features of HTML. In addition, because XHTML is written in XML, all of your pages could be used in conjunction with any of the existing XML tools, such as XSLT, DOM, and SAX-aware processors.

2. Chapters 3 and 4 will show you how to use CSS to control the presentation of the site, and by the end of Chapter 4, the site will look much as it does now, but it will use a CSS style sheet, rather than presentational markup, to control its appearance. This achieves a separation of style from content, and makes the updating and maintenance of styles much easier. Then, should you want to change a color or font across the whole site, you can do so by making an alteration to the one style sheet. Because the presentational rules are contained in the one style sheet, rather

than being repeated in every page (and because browsers keep a copy of this style sheet locally once it has been downloaded), it should make the size of the files in your site smaller, which will enable the site to download faster. The use of style sheets to control presentation also means that you can attach different style sheets to the same XHTML documents to present them in different ways for different purposes and devices.

3. In Chapter 5, you will see how the layout of the pages can be controlled by CSS. Rather than relying on tables to create the masthead and navigation at the top of the page, and using cells to create columns in the main body of the page, these sections of the site will be rebuilt contained within `<div>` elements, which are then positioned using CSS. Using CSS to control page layout means that tables are used only for tabular data, which is what they were originally intended to contain, and that you can change the layout of your entire site simply by altering the one style sheet. It should make your XHTML pages simpler and help make your site more accessible.

4. Chapters 6 and 7 look at the topic of accessibility in greater detail, and throughout these chapters you will learn techniques to ensure that your site meets the requirements of the W3C and Section 508 guidelines. Once you have created an XHTML version of the site that uses CSS to control presentation and layout, you are already a long way towards creating an accessible site; however, you need to be aware of many more issues, which are covered in these chapters. You will be looking at different aspects of a site and the guidelines that cover each aspect; for example, providing text alternatives to nontext content (such as images, video, and audio) so that visually impaired users can still use the site to creating accessible forms and tables. You will also be introduced to some tools that you can use to help ensure that your site meets accessibility requirements.

5. Finally, in Chapter 8 you will see how XHTML is looking toward the future and evolving to cope with the plethora of new devices that can access the Internet. In this chapter, you will learn that the move to XHTML will provide you with a solid foundation for devising pages for all kinds of new devices. You will even see a sample of what a site can look like on a mobile phone.

The code for the final version of the site is available in the code download along with the rest of the code samples for the book. You can also see it online by visiting `www.FirstPromotions.co.uk` and selecting the option to view the final site.

Summary

In this chapter, you have learned both how the Web has changed over time and how Web page authors must adapt skills to reflect those changes. You have seen how, as technology advanced, the size and resolutions of the monitors used when accessing the Web improved, and how the variety of devices capable of viewing Web pages ballooned. This means that you can no longer create one design for a Web page and expect it to work on all devices that access the Internet.

Although the successive versions of HTML introduced all sorts of elements and attributes that helped Web page authors control the appearance of their pages, it is now necessary to stop using that stylistic or presentational markup in the same document as the content you expect people to read. Instead, you should be separating the markup that describes the structure of a document from the stylistic rules that indicate how a Web page should be displayed. The presentation rules should be placed in a separate style sheet written using a language called Cascading Style Sheets, or CSS for short.

HTML has therefore been replaced with a new language that has removed the stylistic markup. It also has a slightly refined syntax because the successor to HTML has been written in a language called XML; hence the name has changed to XHTML. The reformulation of HTML in XML helps ensure that the most widely used language on the Web today will remain just as popular for many years to come, even if new devices with different capabilities start to emerge. This topic is discussed in more detail in Chapter 8.

Finally, you have seen that you have to make numerous changes to the way you design your Web pages if you are going to meet legal requirements that expect all Web pages to be accessible to those with disabilities. Learning to write sites in XHTML and CSS is the first step along the road to building accessible Web sites, although you will find a lot more tips and techniques you need to learn from Chapters 6 and 7 to create truly accessible sites.