2

Document Standards

This chapter explores the various options for a document foundation. CSS is a dynamic tool, in that you can use it in more than one type of document, including HTML, XHTML, and XML documents. Each type of document may have several variations, flavors, or degrees of strictness. This chapter describes what's involved in creating each type.

Document standards are something very important to the aspiring CSS web designer. Inclusion of a Document Type Declaration (explained in a moment) and a well-formed document may mean the difference between a splitting, grueling headache and a mark-up document including CSS that works as expected in all the major browsers. Chapter 1 discussed the W3C body, the group assembled to decide on web standards. This chapter examines the various documents into which you can incorporate CSS, describing what each document looks like and giving you a few very basic examples of each document in action.

The explanation of each topic in the following list is quite lengthy and can easily fill an entire book. This chapter covers only the basics, including

- Writing mark-up
- Obtaining the required web browsers
- Introduction to HTML, XML, and XHTML
- Introduction to the Document Type Declaration
- DOCTYPE sniffing and how to invoke standards mode
- Creating web documents that survive and perpetuate into the foreseeable future

Choosing Which Markup Language to Use

HTML, XHTML, and XML are all based on SGML, which stands for Standard Generalized Markup Language. SGML is the parent of tag-based languages like HTML, XHTML, and XML, although it is not limited to these three examples. SGML defines what it means to be a tag-based language. Like SGML, HTML, XHTML, and XML are acronyms for more complex names:

- **HTML:** HyperText Markup Language
- **XHTML:** eXtensible HyperText Markup Language
- **XML:** eXtensible Markup Language

Some debate exists about which markup language is best and why. Most HTML coders don't bother to follow the W3C standards, which (you may recall from Chapter 1) provide a way to create web documents that behave predictably from browser to browser and from platform to platform. Standards provide a level-playing field for development. They allow the document to be designed in such a way that it can reach the largest audience with the least amount of effort. The `<font>` tag, for example, a simple HTML tag that can set a font face, size, and color, has been *deprecated* (a programming term meaning outdated, obsolete, or slated for deletion) since 1996 (the same year as the inception of CSS level 1). Yet the `<font>` tag has lingered, bloating documents, despite the fact that CSS font controls have existed for a good many years now and are well supported by today's browsers.

Any professional website designer would tell you to prepare your documents for the future while maintaining backward-compatibility. However, that's easier said than done unless you have years of website designing experience under your belt. This book takes the middle road by using XHTML, which is, generally speaking, backward-compatible, and is also (from the standpoint of the W3C) HTML's successor and the future of HTML.

The following sections examine one example of each of these markup languages.

How to Write Markup

HTML, XHTML, and XML are best written using a plain text editor. WYSIWYG (What You See Is What You Get) editors such as Microsoft Word aren't ideally suited for mark-up because the environment is not ideal for the composition of source code. WYSIWYG programs often have features like AutoCorrection and line wrapping; a plain text editor is more appealing precisely because it does not have these automatic features. Furthermore, the more automated WYSIWYG editors are designed to write the source code for you behind the scenes, so you don't have complete control over the structure and formatting of the source code. In contrast, a plain text editor doesn't insert anything into the source code beyond what you type into the text editor.

The Windows Notepad program is one example of a text editor that is ideal for composing source code. To launch Notepad, choose Start ➪ Run and then type `Notepad` in the Open text box. You can also use Microsoft FrontPage, but FrontPage is best used in *view source* mode where you can edit the source code directly instead of via the WYSIWYG interface. The same holds true for Macromedia Dreamweaver MX.

If Notepad is just too basic for your tastes, a text editor that highlights markup and CSS syntax might suit your needs better. The following are full-featured alternative text editors for Windows:

- Crimson Editor: `www.crimsoneditor.com` (free)
- HTML-kit: `www.chami.com/html-kit` (free)

Here are some alternative text editors that work with Mac OS X:

- CreaText: `http://creatext.sourceforge.net` (free)
- BBEdit: `www.barebones.com` (shareware)

Using Windows, you must create HTML files with the .html extension. If you use Notepad, beware of Windows saving your files with a .txt extension, which does not display files properly in a browser. To ensure that your files are saved properly, choose Start➪Run and type `Explorer` (or right-click Start and choose Explore from the pop-up window) to open Windows Explorer. After Windows Explorer is open, choose Tools➪Folder Options to open the Folder Options window; click the View tab, and uncheck the Hide Extensions for Known File Types box (see Figure 2-1). Then click OK.

HTML files are not the only file type in which the document extension is important; other file types require specific extensions as well. Those file types are covered later in this chapter.

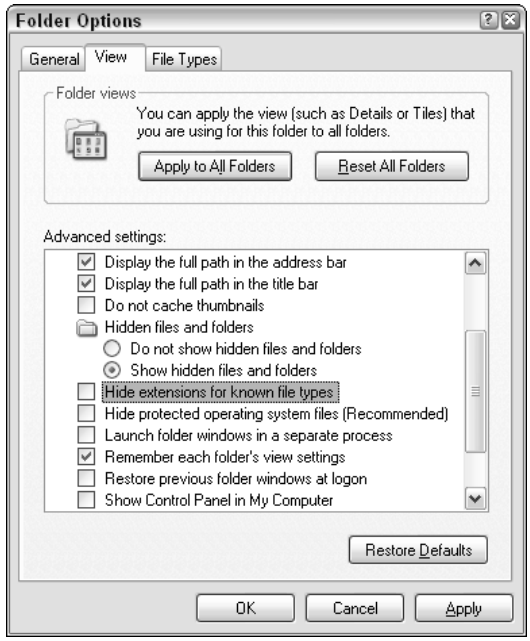

Figure 2-1

Now that you're armed with a text editor to compose your source code, the next section discusses the various browsers in use today.

Which Browser to Use

Throughout the course of the book, I jump between Mozilla 1.7, Microsoft Internet Explorer 6, and Opera 7.54. To help you fully understand each example, I recommend that you have access to all three

browsers. Some examples work on one or two of these browsers, but not all three. I use Mozilla most frequently for this book's examples because it exhibits the most complete support for CSS specifications.

The following sections outline each of these browsers and what role they play in web development.

Microsoft Internet Explorer

Internet Explorer is Microsoft's browser that comes preloaded with the Windows operating system. The current version, as of this writing, is version 6.

Microsoft discontinued development of the Internet Explorer browser as a standalone application for Windows and announced that Internet Explorer will no longer be developed for other operating systems. That translates into no more Internet Explorer for the Macintosh. Instead, Microsoft plans for Internet Explorer to be completely integrated with its next operating system, codenamed Longhorn. Internet Explorer will no longer be its own application but a smaller part of the larger operating system. Consequently, updates to Internet Explorer will only be available with operating system updates and upgrades. In the meantime, updates to the current Internet Explorer 6 browser will come in the form of service packs and updates to the Windows XP and Windows 2000 operating systems via the Windows Update mechanism provided with Windows. According to Microsoft, this will not include updates to Internet Explorer's CSS capabilities or increased support for other web standards.

What does this mean for users and developers? Internet Explorer 6, released in 2001, is the last version of Internet Explorer until the release of the next Windows OS, Longhorn, currently projected for 2006. That leaves a gap of several years before developers will have access to improved support for CSS and other web standards in Microsoft's flagship browser. At the time of this writing, Internet Explorer is already aging rapidly. This means developers are likely to be stuck developing for Internet Explorer 6 for a good few years, coding around features that it does not support and dealing with bugs in the features that it does support. Internet Explorer's global market share has also been on the decline. At the time of this writing, this decline in market share is mostly among technically inclined people. People in the information technology industry have been switching more and more to Mozilla, an open-source browser with far more advanced support for web standards. Current statistics indicate that some websites receive as many as 20% of visitors using a Mozilla browser, although this figure can be much higher or lower depending on the website and its target audience. This figure should not to be taken lightly, however, as the number has been steadily rising over the course of 2004. Despite the decline in its use for some websites, Internet Explorer is still the dominant browser, garnering from 70% to 90% of the browser market share for most websites.

IE 6 provides the least support for the CSS standards discussed in this book; because of this fact, it receives the least attention. However, I couldn't very well write a successful book on website design without addressing how to design a website for the world's most popular browser. Internet Explorer's CSS bugs are well known; when and where appropriate, I discuss how to get around non-existent functionality and how to fix bugs in Internet Explorer without compromising standards. The bulk of this discussion is presented in Chapter 18 where I examine a few of Internet Explorer's most well-known CSS bugs and explore an emerging solution that actually implements CSS support in Internet Explorer 5, 5.5, and 6 for Windows.

You should use the Internet Explorer browser for some of this book's examples. If you don't currently have this browser, you can get it by going to the Internet Explorer homepage at `www.microsoft.com/ie`.

For Macintosh users, I recommend not using Internet Explorer for Mac because the capabilities and bugs of Internet Explorer for Windows and Internet Explorer for Mac are very different. Internet Explorer for the Macintosh has much better support for CSS, but it is an entirely different browser. The name may be the same, but the browsers themselves are very different. For Mac users, I recommend the Safari and Gecko browsers, which I discuss further in the coming sections. If you don't have Internet Explorer for Windows, you still can work through most exercises and examples presented in this book, with the exception of Chapter 18. That chapter is largely focused on identifying and correcting Internet Explorer for Windows bugs.

Gecko: Mozilla, Mozilla Firefox, and Netscape

Gecko was created in January 1998. At that time, Netscape announced that it was making its browser free to its users and that its browser would be *open source*, meaning that its source code would be freely available for modification and distribution. This led to the creation of Mozilla, the organization that develops and manages the Netscape code. America Online later purchased Netscape, and until July 2003, Mozilla remained a part of Netscape. In July 2003, the Mozilla Foundation was created, making Mozilla an independent, not-for-profit corporation. When the Netscape browser became open source, its rendering engine, the part of the browser software responsible for making the source code of a web page into something you can see and interact with, was given the name Gecko.

Gecko is the foundation that a whole suite of browsers relies on to do the behind-the-scenes work of rendering web pages. Gecko is included in AOL for OS X, Netscape 6, Netscape 7, Mozilla, and Mozilla Firefox browsers.

Netscape's browser market share has greatly diminished, whereas Mozilla continues to gain in popularity, occupying the number-two spot by most statistical estimates. The following table charts the relationship between Mozilla and Netscape browsers; the version relationships here show the corresponding version of Gecko shared between the two browsers.

Netscape Browser	Mozilla Browser
Netscape 7.2	Mozilla 1.7
Netscape 7.1	Mozilla 1.4
Netscape 7.0.2	Mozilla 1.0.2
Netscape 7.0.1	Mozilla 1.0.1

For instance, Netscape 7.2, which is the latest Netscape browser at the time of this writing, is based on the Gecko software inside Mozilla 1.7; and the two can be expected to behave the same as far as CSS is concerned.

The Mozilla browser is more than just a browser. It includes an e-mail client, an Internet Relay Chat (IRC) client, and a web page-authoring tool. However, this extra functionality doesn't appeal to everyone who uses a web browser. In fact, an average web user is unlikely to use anything but the browser itself. All this extra functionality also significantly increases the size of a Mozilla download. This, and other considerations, led the Mozilla foundation to create yet another browser. An application that includes only the browser was initially released under the name Phoenix, a name symbolic of a browser

rising from the ashes of the overly complicated Mozilla browser and its Netscape predecessor. It was later renamed Firebird because of threats of trademark infringement from a company already using the Phoenix name; still later, the name was changed to Firefox to avoid conflicts with another open source project named Firebird. Now, settled on the name Firefox, Mozilla Firefox also uses the Gecko rendering engine to present web pages; it can be thought of as the browser part of the Mozilla browser.

The following table shows the relationship between the Mozilla and Mozilla Firefox browsers.

Mozilla Browser	Firefox Browser
Mozilla 1.7	Firefox 0.9
Mozilla 1.6	Firefox 0.8

Firefox 0.9 and Mozilla 1.7 can be expected to behave identically where CSS and design layout is concerned.

For the purpose of this book, I make heavy use of the Mozilla 1.7 browser, although Firefox 0.9 (or a later version) is also suitable. You can obtain either browser by visiting `ww.mozilla.org`. The Mozilla browser is available for Apple, Linux, and Windows operating systems.

> **Each chapter of this book features browser-compatibility notes before a new CSS feature is introduced. These compatibility notes indicate the CSS specification that is referenced (see Appendix B for a detailed explanation) and which popular browsers you can expect to support the CSS feature (at the time of this writing). For those browsers that use the Gecko engine to present web documents — Mozilla, Netscape, Firefox, and AOL for the Mac — I list only Mozilla in the compatibility notes.**

Opera

Opera is a lesser-known, Norwegian-based company. Opera users are fewer, accounting for only a few percent market share by most statistical estimates. Again, that figure can be much higher or lower depending on a website's audience. Also be aware that Opera and Mozilla Firefox browsers can be configured to identify themselves to a website as Microsoft Internet Explorer browsers. This, of course, can distort statistical analysis. This spoofing is done because websites often create content targeting Microsoft Internet Explorer and Netscape specifically, leaving everyone else out in the cold — even though third-party browsers like Mozilla Firefox and Opera probably support the required functionality.

At the time of this writing, the current version of the Opera browser is 7.54. You can download this browser for free from `www.opera.com`. The free version of Opera includes a window that displays advertisements. The Opera browser receives some attention in this book for a few CSS features that neither Mozilla nor Internet Explorer support, but which are worth mentioning. The Opera browser is available for Windows, Macintosh, Linux, and a variety of other platforms.

KHTML: Safari and Konqueror

The last browser that I discuss is Safari, which is based on Konqueror, itself an open-source browser available for Linux operating systems. The rendering engine used in the Safari and Konqueror web

browsers is called KHTML. Safari is developed by Apple and is the browser included with Macintosh OS X operating systems. Previous to Safari, Internet Explorer and Gecko had been dominant on the Mac. Safari behaves in much the same way as Gecko.

For the purpose of this book, I note Safari compatibility when appropriate; however, none of the examples are dependent upon it. Safari is available only for Mac OS X and can be obtained from `www.apple.com/safari`. Konqueror is only available for Linux at this time; it can be found at `www.konqueror.org`.

Having discussed the browsers required for this book, I now introduce the various types of markup documents used with CSS.

Introduction to HTML

Plain old HTML is the oldest markup language. It is composed of a variety of tags. This section is intended as an HTML crash course. HTML is a book-length subject by itself. Unfortunately, I can cover only basic concepts here.

The following shows a set of HTML tags:

```
<HTML></HTML>
```

Tags begin with a left angle bracket (<), also known as the less-than symbol, followed by a tag name, and ending with a right angle bracket (>), or greater-than symbol. Most HTML tags have both an opening and closing tag; the closing tag contains a forward slash before the tag name (/>). Each HTML tag has a predefined purpose. The `<HTML>` tags in this example denote the beginning and end of an HTML document.

Single Tags and HTML Line Breaks

Some tags in HTML don't come in pairs, such as

```
<BR>
```

This is the HTML line break tag. You use it when you want to break text onto another line. Otherwise, HTML continues putting text on the same line — even if you've pressed the Return or Enter key.

In HTML, you combine tags to structure a document. Each tag tells the browser about the data contained in the document. The following example demonstrates a simple HTML table:

```
<TABLE BORDER=1 WIDTH=500>
    <TR>
        <TD>
            HEAT 1/2 OF BUTTER SLOWLY
            ADD BROWN SUGAR STIRRING UNTIL DISSOLVED
            ADD GINGER, RUM, VANILLA & REMAINING BUTTER
            TAKE OFF HEAT & STIR UNTIL ALL BUTTER IS MELTED
        </TD>
    </TR>
</TABLE>
```

In this example, each tag comes together to define a table. A table is used in HTML to show the relationship between data in much the same way a spreadsheet does. The <TABLE> tag begins the table, the <TR> tag defines a row of data and the <TD> tag defines a cell of data. Figure 2-2 shows the text contained in the <TD> element. An *element* is a term often used to refer to a *set* of HTML tags. In this usage, for example, it refers to the opening <TD> tag, the closing </TD> tag, and all the data between those two tags. If only one tag is required, as in the
 tag, that single tag constitutes the element.

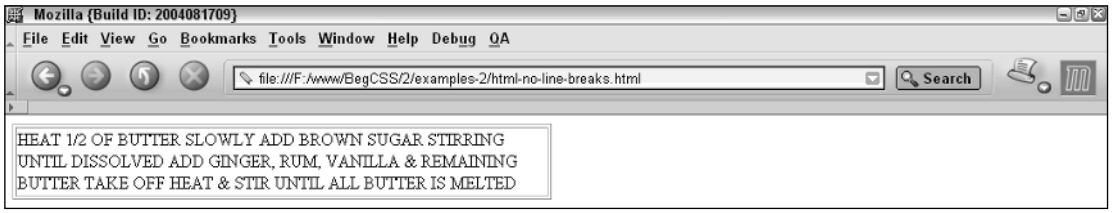

Figure 2-2

When viewed in a browser, the text between the <TD> and </TD> tags appears on one line, or the text wraps around to the next line when it reaches the end of the table cell, table, or window — whichever comes first. Here is the same code snippet with line breaks added:

```
<TABLE BORDER=1 WIDTH=500>
    <TR>
        <TD>
            HEAT 1/2 OF BUTTER SLOWLY<BR>
            ADD BROWN SUGAR STIRRING UNTIL DISSOLVED<BR>
            ADD GINGER, RUM, VANILLA & REMAINING BUTTER<BR>
            TAKE OFF HEAT & STIR UNTIL ALL BUTTER IS MELTED<BR>
        </TD>
    </TR>
</TABLE>
```

As you can see in Figure 2-3, when this code is viewed in a browser, the lines break wherever you inserted a
 tag.

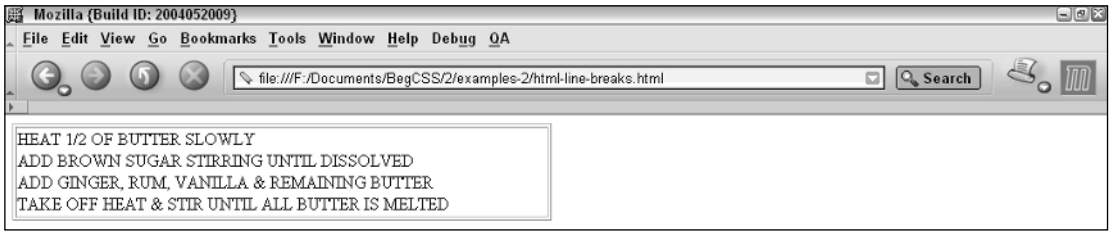

Figure 2-3

HTML Attributes

You can use HTML attributes to alter some feature offered by a specific element. For instance, you can alter features pertaining to the presentation of a particular element, or you can alter features to uniquely identify that element. In the following example, the `ID=mytable` attribute uniquely distinguishes the table from other HTML tables. The `COLSPAN=2` attribute in an HTML table tells the browser that a specific cell of table data spans two columns. The `BORDER=1` attribute tells the browser that a border should go around the table, and the `WIDTH=250` attribute tells the browser how wide the `<TABLE>` should be:

```
<TABLE ID=mytable BORDER=1 WIDTH=250>
     <TR>
          <TD COLSPAN=2>
               Some text.
          </TD>
     </TR>
     <TR>
          <TD>
               Some more text.
          </TD>
          <TD>
               Yet some more text.
          </TD>
     </TR>
</TABLE>
```

Figure 2-4 shows the result of this source code.

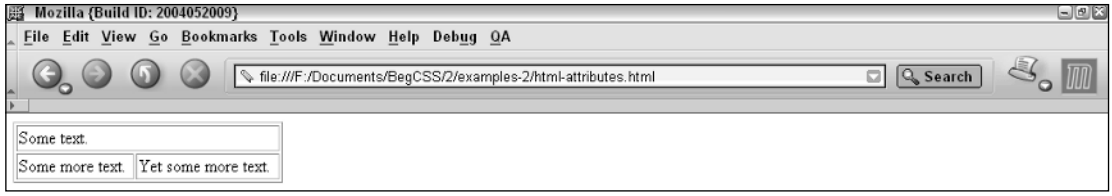

Figure 2-4

Now that I've shown you some HTML basics, work through the following Try It Out. The example uses a recipe to demonstrate HTML table-based design, which is a very common HTML design method.

Try It Out — Creating an HTML Document

Example 2-1. Follow these steps to create an HTML table:

1. In your favorite text editor, type the following HTML:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
    <HTML>
        <HEAD>
            <TITLE>Hummus</TITLE>
        </HEAD>
        <BODY>
            <TABLE BORDER=0 WIDTH=500>
```

```
<THEAD>
    <TR>
        <TH COLSPAN=4 BGCOLOR=#000000>
            <FONT COLOR=#FFFFFF>
                Hummus
            </FONT>
        </TH>
    </TR>
    <TR>
        <TH BGCOLOR=#000000>
            <FONT COLOR=#FFFFFF>
                #
            </FONT>
        </TH>
        <TH BGCOLOR=#000000>
            <FONT COLOR=#FFFFFF>
                Measurement
            </FONT>
        </TH>
        <TH BGCOLOR=#000000>
            <FONT COLOR=#FFFFFF>
                Product
            </FONT>
        </TH>
        <TH BGCOLOR=#000000>
            <FONT COLOR=#FFFFFF>
                Instructions
            </FONT>
        </TH>
    </TR>
</THEAD>
<TBODY>
    <TR>
        <TD>1</TD>
        <TD>#10 can</TD>
        <TD>CARBONZO BEANS</TD>
        <TD>DRAIN 1/2 CAN</TD>
    </TR>
    <TR>
        <TD>15</TD>
        <TD>CLOVES</TD>
        <TD COLSPAN=2>FRESH GARLIC</TD>
    </TR>
    <TR>
        <TD>2</TD>
        <TD>TEASPOONS</TD>
        <TD COLSPAN=2>SALT</TD>
    </TR>
    <TR>
        <TD>8</TD>
        <TD>TABLESPOONS</TD>
        <TD COLSPAN=2>REAL LEMON JUICE</TD>
    </TR>
    <TR>
        <TD>2</TD>
```

```
                        <TD>CUPS</TD>
                        <TD COLSPAN=2>TAHINI</TD>
                    </TR>
                    <TR>
                        <TD>2</TD>
                        <TD>TABLESPOONS</TD>
                        <TD>PARSLEY</TD>
                        <TD>CHOPPED</TD>
                    </TR>
                </TBODY>
                <TFOOT>
                    <TR>
                        <TD COLSPAN=4 BGCOLOR=#808080>
                            <FONT COLOR=#FFFFFF>
                                BLEND IN FOOD PROCESSOR
                            </FONT>
                        </TD>
                    </TR>
                </TFOOT>
            </TABLE>
        </BODY>
    </HTML>
```

2. Save the file as example2-1.html. In your browser of choice, you should see the output shown in Figure 2-5.

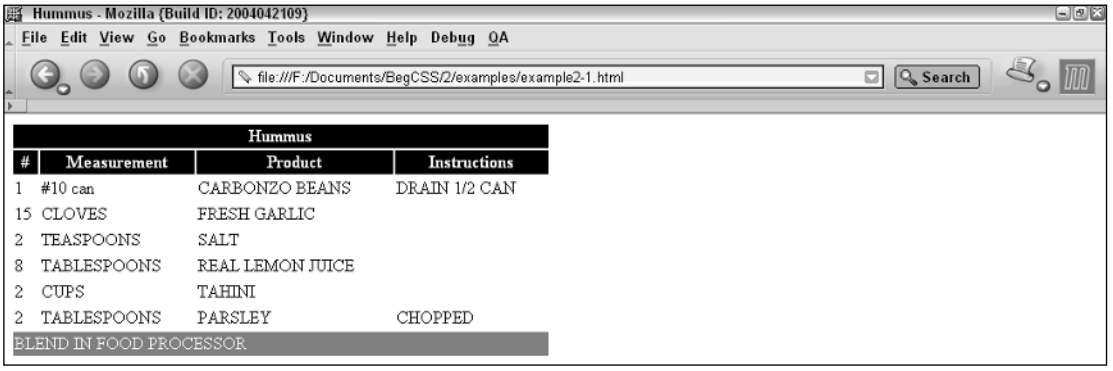

Figure 2-5

How It Works

Nothing fancy here. Notice that the text between the `<TITLE>` and `</TITLE>` tags now appears in the title bar of your web browser's window, and the browser window now contains the text between the `<BODY>` and `</BODY>` tags in the form of a simple recipe card, organized in a table format. So, HTML gives a document some basic structure and gives the browser a predefined way of organizing the data. Presentational markup includes the `<FONT>` tag and the `BGCOLOR` attribute. The `<FONT>` tag is used in markup to set the face, size, and color of a font, and the `BGCOLOR` attribute allows a solid color to be specified for the background of an element using a hexadecimal notation that I explain in more detail in Chapter 4. This book demonstrates why CSS is a much better solution for presentation. Can you imagine having to update 26 table cells just to change the font or background color of each individual cell? With

CSS, the presentational aspects of a document require only a few lines in a style sheet, which can eliminate redundancy. The old way of writing HTML is time-consuming and inefficient on multiple levels.

XML

I discuss XML next because it is, in many ways (especially with its rigid rules), the opposite of HTML. I could spend the entire book discussing XML, but I'm going to cover only the very basic stuff here. Unlike HTML, every XML tag absolutely must have both an opening and closing tag. XML is also case-sensitive, so `title` isn't the same thing as `TITLE`. In a nutshell, XML uses tags and attributes to describe what they enclose:

```
<movies>
     <movie>Big Fish</movie>
     <movie>Lord of the Rings</movie>
</movies>
```

Although having the tags describe what they enclose is one of the primary benefits of XML, it's not required. The following is equally valid XML:

```
<asdfgh>
     <lkj>Big Fish</lkj>
     <lkj>Lord of the Rings</lkj>
</asdfgh>
```

XML is useful not only for web presentation, but for an entire suite of applications, including complex programming languages, databases, and many other applications. But again, going into all the deep, dark details of XML is beyond the scope of this book.

To reiterate, the important differences between XML and HTML are primarily the following:

- XML is case-sensitive; HTML is not.
- XML absolutely requires both an opening and closing tag on all elements or the use of a special shortcut syntax in which only one tag is required. (This shortcut syntax allows both the opening and closing tag to be written as one tag.) Conversely, HTML does not always require both an opening and closing tag.
- Unlike HTML, XML tags do not have a predefined meaning. The author (not the browser) creates the meaning of each tag, which makes XML dynamic and adaptable to more than one application.
- XML documents are opened with an XML declaration instead of with an `<HTML>` tag.

You can find out more about applying CSS to an XML document in Chapter 17, or you might consider buying Wrox's Beginning XML, 3rd Edition

Now that you know what an XML document is, the following Try It Out demonstrates how the recipe card used in the preceding Try It Out example is formatted when written in XML.

Try It Out An XML Document

Example 2-2. Follow these steps to create a simple XML document.

1. Continuing the recipe example, fire up your favorite text editor and enter the follow XML:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<recipe>
    <title>Hummus</title>
    <ingredients>
        <ingredient>
            <quantity>1</quantity>
            <measurement>#10 can</measurement>
            <product>CARBONZO BEANS</product>
            <instructions>DRAIN 1/2 CAN</instructions>
        </ingredient>
        <ingredient>
            <quantity>15</quantity>
            <measurement>CLOVES</measurement>
            <product>FRESH GARLIC</product>
        </ingredient>
         <ingredient>
            <quantity>2</quantity>
            <measurement>TEASPOONS</measurement>
            <product>SALT</product>
        </ingredient>
        <ingredient>
            <quantity>8</quantity>
            <measurement>TABLESPOONS</measurement>
            <product>REAL LEMON JUICE</product>
        </ingredient>
        <ingredient>
            <quantity>2</quantity>
            <measurement>CUPS</measurement>
            <product>TAHINI</product>
        </ingredient>
        <ingredient>
            <quantity>2</quantity>
            <measurement>TABLESPOONS</measurement>
            <product>PARSELY</product>
            <instructions>CHOPPED</instructions>
        </ingredient>
    </ingredients>
    <directions>
         BLEND IN FOOD PROCESSOR
    </directions>
</recipe>
```

2. Save the file as example2-2.xml. Figure 2-6 shows what the XML document looks like when loaded into a browser.

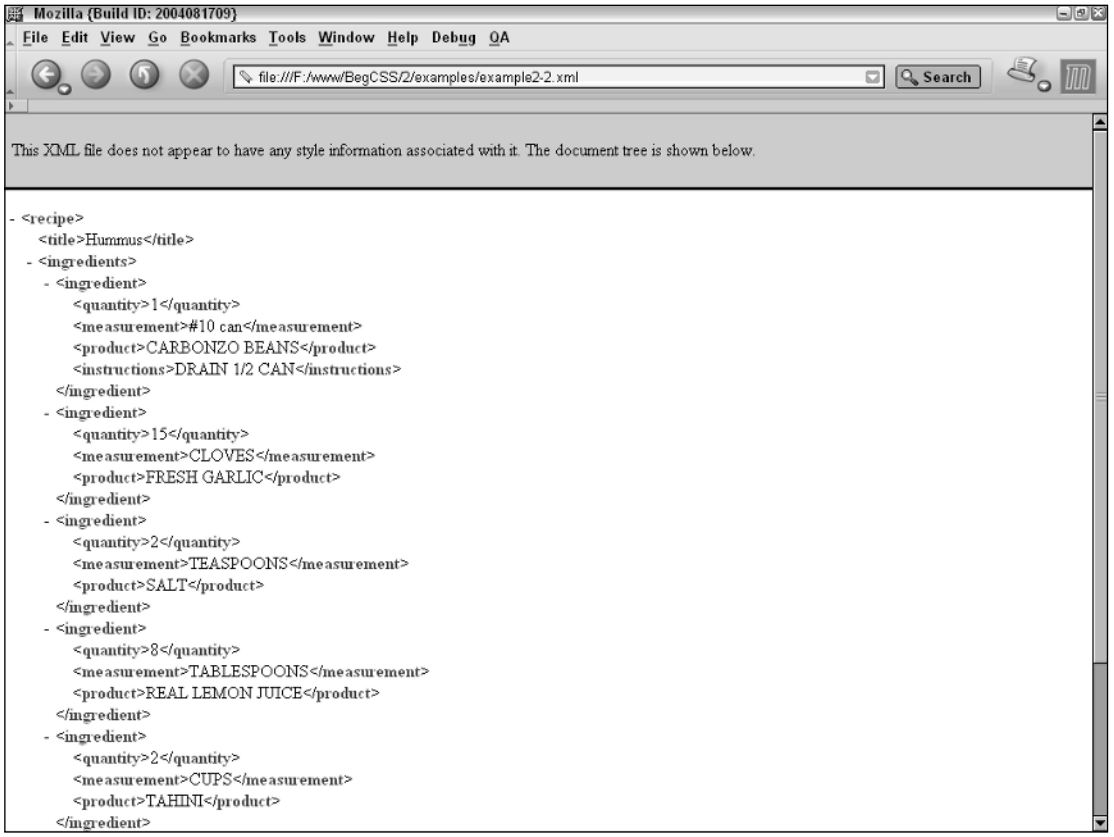

Figure 2-6

How It Works

The browser has no information about how to present the document, so the browser shows a tree of the XML source code. Often, the browser also provides a way to expand and collapse portions of the XML source code tree by displaying plus and minus signs that act as navigation buttons for the XML source code.

XML not only structures data; it also describes data. (Although this isn't required, it is one of XML's advantages.)

Chapter 17 explores in greater detail how to style XML documents, as well as offers more discussion on its background, structure, and uses. For now, simply note that XML is a more advanced method of structuring data that is applicable to more than one medium, not just end-user presentation.

XHTML

XHTML, the marriage of HTML and XML, is the document structure on which this book focuses. Put more simply, XHTML is a reformulation of HTML that conforms to the strict rules of XML syntax. For the purpose of this book, the XML part of XHTML is ignored, primarily because without XML, XHTML is the same as HTML except that it must conform to a stricter set of rules. Without XML, the examples are less confusing and more straightforward. Furthermore, and perhaps most important, Internet Explorer 6 — the world's most popular browser — does not currently properly support XHTML documents that contain both HTML and XML. Because I ignore the XML piece of XHTML, the examples presented in this book can work in the largest percentage of browsers.

Even though I have ignored the XML portion of XHTML, the use of XHTML-compliant HTML allows the documents presented in this book to be forward-compatible. That way, when XHTML support is improved and becomes more widespread, integration of XML becomes effortless.

XHTML provides the same predefined tags as HTML, while offering the user the extensibility of XML. XHTML differs from HTML in the following ways:

- XHTML, like XML, is case-sensitive, so you must write all HTML tags in all lowercase letters to achieve their associated HTML meaning.
- All attribute values must be enclosed in quotes.
- No attribute minimization (described shortly) is allowed.
- All tags must have both an opening and closing tag.

This set of rules allows XHTML to take on the extensibility of XML. The following sections discuss each rule in greater depth.

XHTML Is Case-Sensitive

The HTML part of XHTML basically includes the same tags with the same functionality used in HTML. More tags in HTML are used than in XHTML; for the most part, however, the two languages include the same tags with the same functionality.

One of the major differences between XHTML and HTML is that HTML is a very loose language; it doesn't care how a tag is written. For instance, `<HTML>` and `<html>` is the same tag in HTML. However, XML is case-sensitive, so when it came to marrying XML with HTML, certain aspects of HTML had to be made more rigid. In XHTML, `<HTML>` and `<html>` are not the same tag. In XHTML, all HTML tags must appear in all lowercase letters to achieve their associated HTML meaning.

All Attribute Values Must Be Enclosed in Quotes

Just as HTML is not case-sensitive, it is also very forgiving of attribute values not enclosed within quotation marks. The following is an example of an attribute value not enclosed within quotations:

```
<TD COLSPAN=2>
```

This attribute value causes an HTML table cell to span across two columns instead of just one. The following is an XHTML acceptable equivalent:

```
<td colspan="2">
```

The tag and attributes must be written in lowercase, and the value of the attribute must be enclosed within quotation marks. You may use either single or double quotation marks, as shown in the following example:

```
<td colspan='2'>
```

This all begs the question, "What happens if I don't follow the syntax rules?" The simple answer is that the document ceases to be valid XHTML. The complex answer is that when XML is added to an XHTML document and the XHTML document is not well formed, the browser refuses to display the document and displays an error indicating what went awry. I go into further detail about these errors in Chapter 17.

Well formed describes an XML document that strictly follows the syntax rules outlined here.

No Attribute Minimization Allowed

Scattered throughout HTML are a few oddball attributes that can be written with only one word. This syntax is referred to as *attribute minimization*. For example, suppose you're writing an HTML form with a select field or a field that shows a drop-down box. The following code specifies the option to allow the preselection of a value that appears in the drop-down list:

```
<SELECT NAME='some_field' SIZE='5'>
     <OPTION VALUE='1' SELECTED>Some selection</OPTION>
     <OPTION VALUE='2'>Another selection</OPTION>
</SELECT>
```

The `SELECTED` syntax used in the `<OPTION>` element is an example of attribute minimization. In XHTML this is not allowed. The following code is an XHTML-acceptable equivalent:

```
<select name='some_field' size='5'>
     <option value='1' selected='selected'>Some selection</option>
     <option value='2'>Another selection</option>
</select>
```

For backward compatibility, you generally write the `SELECTED` attribute as `selected="selected"`. In more general terms, wherever there is attribute minimization, write `attribute="attribute"`. This method works best in all browsers, including the older ones.

All Elements Require Both an Opening and Closing Tag

Like XML, XHTML requires both an opening and closing tag for every element, which means that the simple HTML line break, `<BR>`, is unacceptable in XHTML. Likewise, the `<IMG>` tag, which is used to insert an image into a document, is also not acceptable in XHTML. The same is true for a plethora of other tags.

XHTML allows you to write the HTML
 tag in two ways. The first is by using both opening and closing tags:

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

However, this presents a problem with backward compatibility. Modern browsers don't ignore the closing </br> tag, meaning that browsers treat </br> like another line break instead of as a closing tag. Therefore, instead of adding a closing tag, the W3C devised a shortcut syntax: a way to write the opening and closing tags as a single tag, as shown in the following:

```
<br/>
```

This time, the tag includes a forward slash before the closing angle bracket; the forward slash is used to close the tag. This is a shorthand way to write two tags when only one tag is technically required, as is the case for line breaks, images, and other special elements requiring no closing tag. This syntax allows the rule (that all elements requiring both an opening and closing tag) to be honored. This method presents a problem for older browsers, although modern browsers correctly interpret this tag. Older browsers ignore the tag altogether. For the tag to work on all browsers, include a single space between the end of the tag text and the forward slash, like this:

```
<br />
```

The single space allows the XHTML single slash syntax to work in older browsers. Either the
 syntax (no space) or the
 (including a space) are perfectly valid ways of writing the break tag.

Apply this shortcut syntax to any tag that doesn't require two tags, such as the image tag (<img />), the HTML form input (<input />), and so on.

The following section outlines why XHTML is important and why this book focuses on XHTML as the delivery method for web-based content.

Why XHTML Is the Future

XHTML is what HTML should have been from the very beginning. The following list details what makes XHTML so exciting to web developers.

- **XHTML is a cleaner representation of HTML.** The language is not bloated with presentational markup like the <font> tag and other HTML features that exist purely for presentation.
- **XHTML's rules are consistent.** You always know what to expect when viewing an XHTML page. Its rules define everything so that every XHTML page contains lowercase tags and attributes and attribute values are always quoted.
- **XHTML is far more dynamic than HTML.** XML capabilities make XHTML a better fit for complex projects.
- **XHTML documents download faster.** You can design XHTML files to be as much as two-thirds smaller in size when compared to presentation-oriented HTML documents. Smaller file sizes save hard disk space as well as expensive bandwidth.

- **XHTML files are easier to manage and update on multiple levels.** XHTML documents use less markup to structure the data of a document, so updating and managing XHTML becomes that much easier. When combined with CSS, you can centralize presentation into a single file. This functionality saves the maintainer the headache of editing each individual XHTML file. CSS also allows you to alter the entire look and feel of a website with a single file. Because you can link externally to CSS files (discussed in Chapter 3), a browser can cache them. This means you avoid re-download of every page requested by the visitor, which again saves expensive bandwidth.

Now that you are familiar with XHTML documents, it's time to revisit the recipe card example. The following Try It Out builds on Example 2-1, but it rewrites the original HTML document in XHTML, including all XHTML-acceptable syntax.

Try It Out An XHTML Document

Example 2-3. Follow these steps to rewrite the original HTML recipe document in XHTML.

1. Enter the following XHTML code in your text editor.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
    <html>
        <head>
            <title>Hummus</title>
            <style type='text/css' media='all'>
                table {
                    border-width: 0px;
                    width: 500px;
                }
                th {
                    background: black;
                    color: white;
                }
                tfoot td {
                    background: #808080;
                    color: white;
                }
            </style>
        </head>
        <body>
            <table>
                <thead>
                    <tr>
                        <th colspan='4'>
                            Hummus
                        </th>
                    </tr>
                    <tr>
                        <th>
                            #
                        </th>
                        <th>
                            Measurement
```

```
                        </th>
                        <th>
                            Product
                        </th>
                        <th>
                            Instructions
                        </th>
                    </tr>
                </thead>
                <tbody>
                    <tr>
                        <td>1</td>
                        <td>#10 can</td>
                        <td>CARBONZO BEANS</td>
                        <td>DRAIN 1/2 CAN</td>
                    </tr>
                    <tr>
                        <td>15</td>
                        <td>CLOVES</td>
                        <td colspan='2'>FRESH GARLIC</td>
                    </tr>
                    <tr>
                        <td>2</td>
                        <td>TEASPOONS</td>
                        <td colspan='2'>SALT</td>
                    </tr>
                    <tr>
                        <td>8</td>
                        <td>TABLESPOONS</td>
                        <td colspan='2'>REAL LEMON JUICE</td>
                    </tr>
                    <tr>
                        <td>2</td>
                        <td>CUPS</td>
                        <td colspan='2'>TAHINI</td>
                    </tr>
                    <tr>
                        <td>2</td>
                        <td>TABLESPOONS</td>
                        <td>PARSLEY</td>
                        <td>CHOPPED</td>
                    </tr>
                </tbody>
                <tfoot>
                    <tr>
                        <td colspan='4'>
                             BLEND IN FOOD PROCESSOR
                        </td>
                    </tr>
                </tfoot>
            </table>
        </body>
    </html>
```

2. Save the file as example2-3.html. Figure 2-7 shows the output of this source code in the browser.

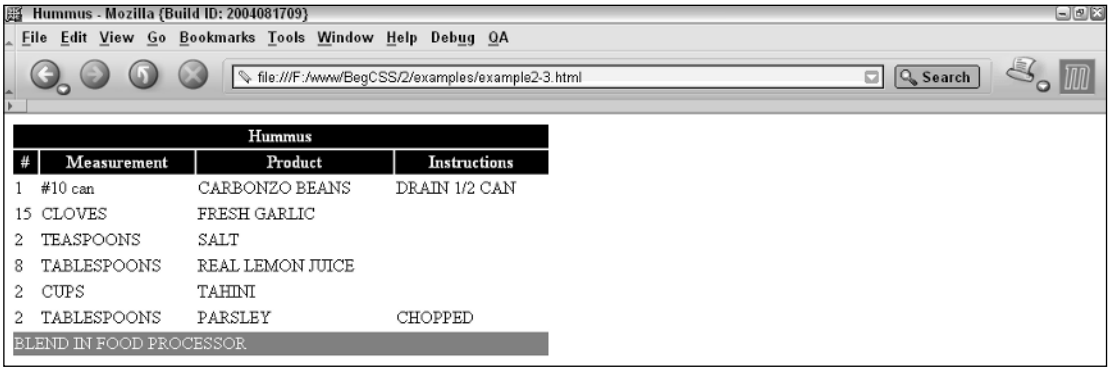

Figure 2-7

How It Works

Figure 2-7 shows that this output is identical to what's shown in Figure 2-5. This example demonstrates that XHTML is a little more rigid than HTML in terms of its syntax and structure. In this example, I remove the deprecated, presentational HTML that I used in Example 2-1 and replace it with the cleaner, more concise syntax of XHTML and CSS.

CSS brings a great deal of flexibility to the table by centralizing the document's presentation in a single place instead of scattering it all over the document, as was the case in Example 2-1. XHTML extends HTML by offering the flexibility of XML along with the predefined tags of HTML. Although this example does not demonstrate XML within an XHTML document, the subtle differences between XHTML and HTML are clear. The presentational aspects of Example 2-1 — the `BGCOLOR`, `BORDER`, and `WIDTH` attributes, and the `<FONT>` tag — are now all declared using a cascading style sheet that's embedded directly in the document. The tags and attributes are all written in lowercase letters, and the attribute values are enclosed in quotations. The HTML line break tag is written with the shortcut syntax for closing a tag: `<br />` instead of `<BR>`.

Thanks to my good friends at JT's Island Grill and Gallery in Chokoloskee, Florida, for the delicious recipe.

The Document Type Declaration

The Document Type Declaration (commonly abbreviated DTD) is a single line of code that appears before the opening `<html>` tag in HTML and XHTML documents. It's a rather complicated topic; however, it is a crucial component of writing standards-compliant XHTML documents with CSS. The Document Type Declaration looks like this:

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

The inclusion of this code, more or less announces to the browser what type of document it can expect. It tells the browser about the markup syntax used in the document and provides some rules for its behavior. The example shown here is used to indicate to the browser that the document is an XHTML document that strictly follows the W3C recommendation of what XHTML 1.0 is. For introductory purposes, this is a bit of an over-simplification of the DTD. Understanding the inner components of what it is and does, however, is not essential knowledge for designing web pages with XHTML and CSS.

The inclusion of the DTD is an important component of designing standards-oriented XHTML documents. But in relation to CSS, the DTD has a more important effect on designing web pages: DOCTYPE sniffing.

DOCTYPE Sniffing

DOCTYPE sniffing, also known as the DOCTYPE switch, is a process used by modern browsers to select a rendering mode based on the inclusion or absence of a DTD. In Internet Explorer 6, the DTD is used to invoke one of two rendering modes. The first rendering mode is known as quirks mode, and the second rendering mode is known as standards-compliant mode. Essentially this behavior makes Internet Explorer 6 two browsers in one. Standards-compliant mode is used to make web pages that follow the W3C standards to the letter. Quirks mode relies on nonstandard methods to render web pages that were created before the W3C standards were implemented and came into widespread use. Quirks mode exists as a way to provide backward-compatibility for the millions of websites in existence today that were either created before W3C standards were implemented, or for websites whose authors are not yet standards savvy.

The two rendering modes have very significant differences in Internet Explorer 6. The choice of rendering mode may have an adverse effect on authoring CSS-enabled documents because Microsoft has made some CSS features available only when Internet Explorer 6 is in standards-compliant mode.

Other browsers like Mozilla, Opera, and Safari also have standards-compliant and quirks modes. In fact, these browser introduce an additional rendering mode known as *almost standards mode*. Appendix D provides a detailed table of which DTDs invoke which mode in which browser. However, even though the less popular browsers also provide these rendering modes, the differences between these modes is not nearly as significant as Internet Explorer 6's, which is why I have focused on Internet Explorer 6 here.

Throughout this book, all the examples invoke standards-compliant mode in all browsers. The following Try It Out demonstrates how Internet Explorer renders a web page in quirks mode. After this example, I provide an example of how Internet Explorer renders content in standards-compliant mode, and I highlight the differences between the two.

Try It Out — Experimenting with Quirks Mode and Standards Mode

Example 2-4. To demonstrate the nuances in rendering your web page according to these two rendering modes in Internet Explorer 6, follow these steps.

1. Using your favorite text editor, type the following HTML document:

```
<html>
    <head>
        <title> Doctype Sniffing </title>
        <style type='text/css' media='all'>
```

```
            table {
                margin: auto;
            }
            td {
                background: green;
                margin: 10%;
                border: 5px solid black;
                padding: 10%;
                width: 100%;
                color: black;
                font-size: 200%;
                text-align: center;
            }
            div {
                background: green;
                border: 5px solid black;
                font-size: 200%;
                padding: 1%;
            }
            #div1 {
                float: left;
                width: 30%;
            }
            #div2 {
                margin-left: 34%;
                width: auto;
            }
        </style>
    </head>
    <body>
        <table>
            <tr>
                <td>
                    Some content.
                </td>
            </tr>
        </table>
        <div id='div1'>
            This div smells funny.
        </div>
        <div id='div2'>
            This div has a <em>strange</em> and rather pleasing odor about it.
        </div>
    </body>
</html>
```

2. Save the file as example2-4.html.

3. Using the example2-4.html file, add the following DTD before the opening `<html>` tag to see how this example renders according to W3C standards:

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

4. Save the modified file as example2-5.html.
5. Open Internet Explorer and load example2-4.html in one browser window and example2-5.html in another.

Figure 2-8 shows example2-4.html in quirks mode in Internet Explorer.

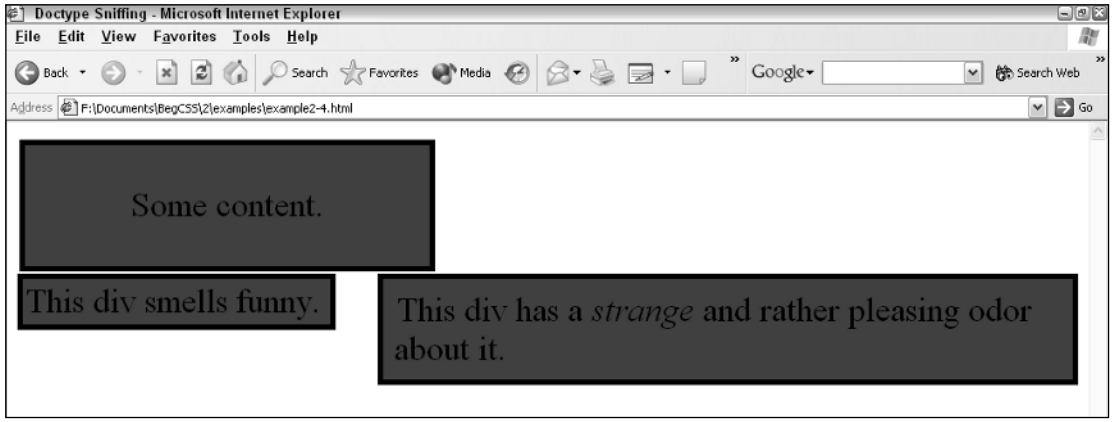

Figure 2-8

Figure 2-9 shows example2-5.html in standards-compliant mode in Internet Explorer.

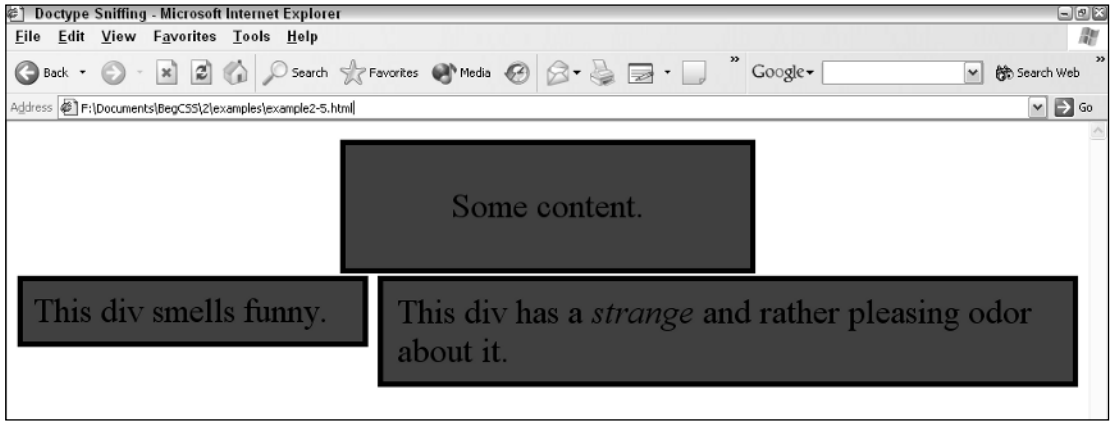

Figure 2-9

How It Works

When compared side-by-side, the effects of the DTD become obvious. In example2-4.html, the table is not centered, but it is centered in the rendering of example2-5.html. Furthermore, the `<div>` elements do not have the same dimensions. These two examples demonstrate a few of the very important differences between quirks mode and standards-compliant mode. You do not encounter any of these differences

between these two modes when you view the same two pages using the Mozilla browser. This indicates to you that the quirks mode of one browser isn't the same as the quirks mode of another browser. Furthermore, through this example it becomes obvious that using Internet Explorer in quirks mode limits the scope of available CSS functionality.

Microsoft itself provides a detailed accounting for these inconsistencies in available CSS functionality. These are documented for each CSS feature at the following URL:

```
http://msdn.microsoft.com/library/default.asp?url=/workshop/author/css
/reference/attributes.asp
```

In summary, for CSS design, cross-browser consistency is dependent upon the browser's interpretation of the W3C standards. Using the standards-compliant mode provided by the browser is one way to avoid inconsistency. The following section outlines some of the possible DTDs that are available.

Common Document Type Declarations

The following sections summarize the most common DTDs in use and include a brief description of what the `DOCTYPE` is used for. The examples in this book primarily use the XHTML 1.0 Strict `DOCTYPE`.

HTML 4.01 Transitional

HTML transitional is the loosest document type. It allows the use of deprecated HTML, such as the `<font>` tag, which was one feature of HTML that necessitated the creation of CSS in the first place. The HTML 4.01 Transitional `DOCTYPE` is not case-sensitive. For example, `<HTML>` is the same as `<html>`. This principle applies to all HTML 4.01 `DOCTYPE`s. When this `DOCTYPE` appears with a URL to the DTD, it invokes the standards mode:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

If the URL does not appear in the DTD, as in the following example, the `DOCTYPE` invokes quirks mode:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" >
```

In fact, if the DTD URL is incorrect in any way, the `DOCTYPE` invokes quirks mode instead of standards mode.

HTML 4.01 Frameset

HTML frameset is the same declaration as HTML transitional, but with the inclusion of *framesets*. Framesets enable the browser window to be partitioned (or separated) into more than one window, allowing the author to display and reference more than one HTML page at once. HTML frameset also allows the use of inline frames, which enables you to include external HTML documents inside another HTML document. You specify HTML frameset like this:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
"http://www.w3.org/TR/html4/frameset.dtd">
```

Like HTML 4.01 transitional, this DOCTYPE also invokes standards mode. If it appears without the DTD URL or with an invalid URL, it invokes quirks mode.

HTML 4.01 Strict

HTML strict (theoretically, at least) does not allow the use of deprecated HTML. Like the example of the XHTML strict Document Type Declaration that I discussed earlier in this chapter, the HTML 4.01 Strict DTD implies that the document strictly follows the W3C's definition of what HTML 4.01 is. You specify HTML strict like this:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```

This DOCTYPE invokes standards mode regardless of whether it appears with the DTD URL. But an incorrect URL may invoke quirks mode in some browsers.

XHTML 1.0 Transitional

Similar to HTML 4.01 Transitional, XHTML transitional implies that the markup is not perfect. You specify XHTML transitional like this:

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

All XHTML DOCTYPEs invoke standards mode regardless of whether a DTD URL appears in the declaration.

XHTML 1.0 Frameset

Similar to HTML 4.01 frameset, XHTML frameset is essentially XHTML transitional but allows the use of frames. Note, however, not including a frameset DOCTYPE generally has no effect on frame usage. You specify the XHTML frameset DOCTYPE like this:

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

As stated previously, this XHTML DOCTYPE invokes standards mode.

XHTML 1.0 Strict

XHTML 1.0 Strict is the DOCTYPE appearing in all of this book's examples. It implies that the document conforms to the more rigid XHTML syntax rules and does not allow deprecated HTML tags and attributes. The browser ignores deprecated HTML tags or attributes if they appear in an XHTML document with a strict DOCTYPE. You specify XHTML strict like this:

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

Again, XHTML strict invokes standards mode.

Summary

CSS is designed to work with a variety of documents. Although not limited to HTML, XHTML, and XML, you find CSS most commonly used in those types of documents. In this chapter you learned that the following:

- Internet Explorer 6, Gecko, Opera, and KHTML browsers make up the majority of browsers in use today, with Internet Explorer 6 being the world's most popular browser.
- Markup languages are used to structure data.
- HTML is the oldest markup language and allows the loosest syntax.
- XML is the extreme opposite of HTML in terms of syntactical strictness in that it is very rigid and very defined.
- XHTML is the middle road and the chosen delivery medium for most of this book's examples.
- XHTML is stricter than HTML in terms of its basic syntactical rules.
- XHTML is case-sensitive, and you must write all its HTML tags in lowercase letters and all attribute values must be enclosed within quotation marks.
- XHTML does not allow attribute minimization.
- All XHTML tags must contain both an opening and a closing tag, although XHTML provides a shortcut syntax that allows the opening and closing tags to be written together as a single tag.
- The Document Type Declaration is a line of code that appears before the opening `<html>` tag; it provides the browser with some basic information about the document's syntax.
- DOCTYPE sniffing (the DOCTYPE switch) is used to select one of two rendering modes in Internet Explorer 6, and one of three rendering modes in all other browsers.
- Internet Explorer 6 quirks mode limits the scope of available CSS functionality, which results in huge inconsistencies between Internet Explorer 6 quirks mode and Internet Explorer 6 standards-compliant mode.

Chapter 3 begins a discussion of basic CSS syntax.

Exercises

1. Using a structure similar to that of the recipe paradigm, create HTML, XML, and XHTML documents listing your favorite movies. Include information for the lead actor or actress, the title, and the genre.
2. Explain the differences between quirks mode and standards mode.