Part I

Introducing HTML, XHTML, and CSS

CHAPTER 1

Web Page Development 101

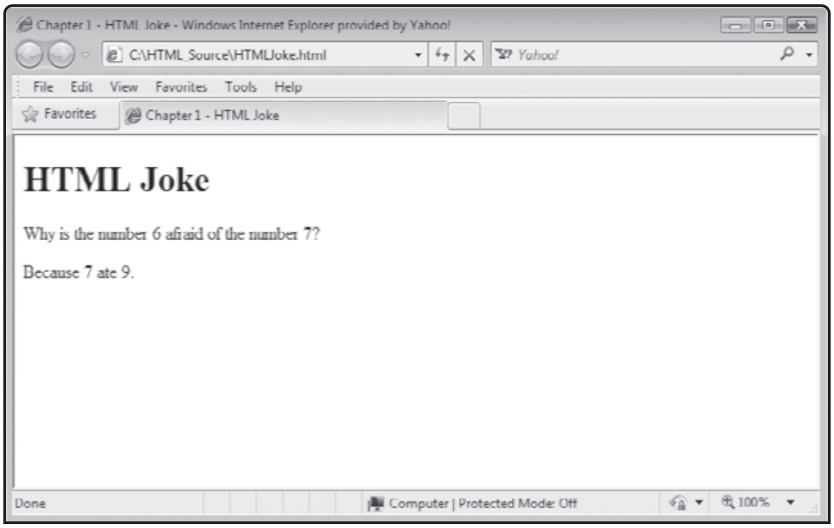
If you can surf the Internet, operate a DVD player, or create a Microsoft Word document, you too can create your own web pages. This chapter introduces the HTML and XHTML markup languages, which web developers use to develop web pages, and CSS, which is used to provide styling information that affects the appearance and presentation of web pages. You will also be introduced to the DOM (document object model), which organizes web page contents into a hierarchy allowing it to be easily referenced, and to JavaScript, which is a scripting language used to create dynamic web pages that interact with visitors. You will also learn how links are used to connect things together and will end by learning how to create your very first web page.

Specifically, you will learn:

- About the differences between HTML and XHTML markup languages
- About the role of CSS in influencing web page presentation
- About JavaScript and its role in creating interactive websites
- How links are used to tie together the pages of your website and to connect them to outside world

Project Preview: The HTML Joke Page

In this chapter and in each chapter that follows, you will learn how to create a new web project. Learning web development though hands-on exercises and instruction makes learning a lot of fun. This chapter's project is the HTML Joke Page. The web page displays a joke and its punch line when displayed, as shown in Figure 1.1.


Chapter 1 - HTML Joke - Windows Internet Explorer provided by Yahoo!
C:\HTML_Source\HTMLJoke.html
Yahoo!
File Edit View Favorites Tools Help
Favorites
Chapter 1 - HTML Joke
Done
Computer | Protected Mode: Off
100%

HTML Joke

Why is the number 6 afraid of the number 7?

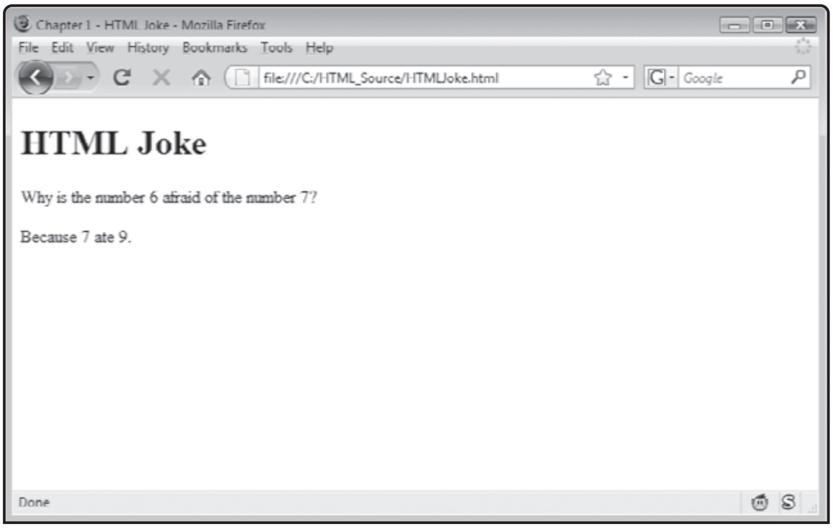
Because 7 ate 9.

Figure 1.1

Displaying the HTML Joke Page using Internet Explorer 8.

As you can see, there is not a lot to this web page, just some text that displays the joke and its punch line. Still, by learning how to create this simple web page, you will learn the basic steps involved in creating all kinds of web pages.

Properly designed web pages should display in a consistent fashion regardless of which web browser they display in. Figures 1.2 and 1.3 show how this same web page looks when displayed using different web browsers.


Chapter 1 - HTML Joke - Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///C:/HTML_Source/HTMLJoke.html
Google
Done

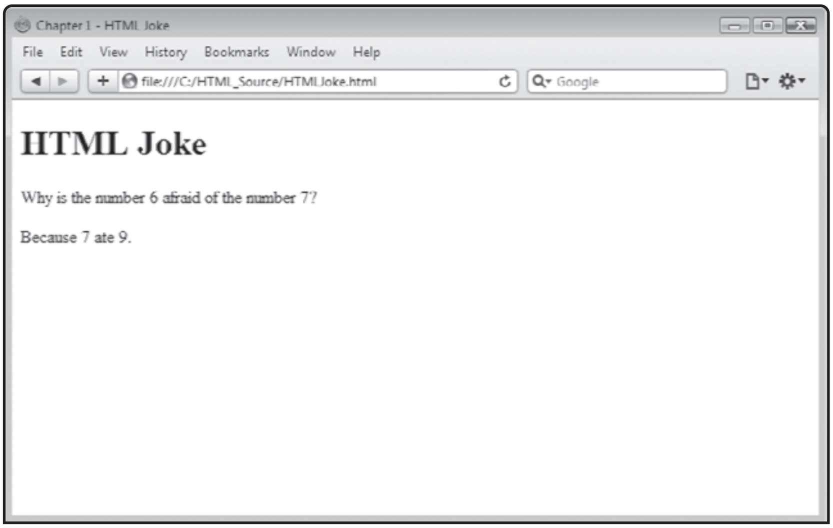
HTML Joke

Why is the number 6 afraid of the number 7?

Because 7 ate 9.

FIGURE 1.2

Displaying the HTML Joke Page using Mozilla Firefox 3.


Chapter 1 - HTML Joke
File Edit View History Bookmarks Window Help
file:///C:/HTML_Source/HTMLJoke.html
Google

HTML Joke

Why is the number 6 afraid of the number 7?

Because 7 ate 9.

FIGURE 1.3

Displaying the HTML Joke Page using Apple Safari 3.

Introducing HTML, XHTML, CSS, and Other Web Development Technologies

The development of web pages and websites involves a number of different programming languages that together allow web developers to create unique, interesting, and interactive web pages. These languages include:

- HTML
- XHTML
- CSS
- JavaScript

Each of these languages is introduced and discussed in this chapter and then reviewed in detail in later chapters. These languages have been around for years and are supported by recognized industry standards. The complementary nature of these languages produces a robust web development environment.

HINT Many people use the terms web document and web page interchangeably. In this book, a distinction is made between the two terms. The term web document is used to refer to text files containing markup code and the term web page is used to refer to the result that is displayed and made visible to web surfers when a web document is loaded and rendered by a web browser or similar device.

HTML and XHTML

HTML and XHTML are markup languages. *Markup languages* are languages that "mark up" plain text so it is formatted and displayed in a web browser in interesting and useful ways. Web pages are *rendered* (displayed) by browsers when they load a copy of your web documents (plain text files containing markup). Users load web documents by entering the web document's URL into the browser. As this book will demonstrate, most web pages are made up of nothing more than plain text files consisting of HTML or XHTML (and perhaps a little CSS).

It All Started with HTML

HTML or *Hypertext Markup Language* is a markup language that was created based on SGML or Standard Generalized Markup Language. First appearing in the early 1990s, HTML has long served as the standard markup language for the Internet. *XHTML*, otherwise known as the *Extensible Hypertext Markup Language*, is a markup language that is very similar to HTML, except that it is based on XML rather than SGML. *XML* (*Extensible Markup Language*) represents a more restrictive subset of SGML, resulting in tighter syntax that yields more consistent results when markup is rendered by browsers.

HTML was created for the purpose of describing the structure of text-based web documents. HTML suffers from a number of irregularities. XHTML was created to addresses these irregularities. XHTML has a predictable syntax. Markup, be it HTML or XHTML, is the glue that binds all of the content in your web documents together.

HINT Over the years, HTML has been enhanced a number of times. HTML 1.0 was published in 1993. HTML 2.0 made its debut in 1995. HTML 3.0 appeared in 1996 and HTML 4.0 was published in 1997. The most current version of HTML, HTML 4.01, arrived on the scene in 1999. Even as new features have been added to HTML, old features have also been deprecated. Deprecated features are marked for eventual removal from the language. As such, their usage is highly discouraged.

Along Came XHTML

XHTML 1.0 arrived on the scene in 2000 and was originally intended to be the replacement for HTML. XHTML is an update of HTML 4.01 that follows the more restrictive rules of XML. XML is an extensible language that allows you to define your own elements. However, XHTML version 1.0 does not support the definition of custom tags. Instead, you must stick with the elements defined as part of the XHTML 1.0 specification.

The changes that were made to make the leap from HTML to XHTML were primarily small ones. The primary purpose of the first version of XHTML was to adapt HTML to make it compliant with XML. As a result, the two markup languages remain very similar and the way things are done in one language is often identical to the way things are done in the other. XHTML documents are required to be well formed. To conform to XML, all XHTML elements must be written in lowercase and all elements must also be closed. HTML is a lot more lenient in both of these circumstances, allowing upper- or lower-case spelling and open elements. Unlike HTML, XHTML forces you to enclose all attribute values inside quotation marks (single or double). Although HTML sometimes imposed this requirement, it often allows you to do away with it.

The movement from HTML to XML results in a simplified and less complicated markup language. It also means that the markup language has become compatible with all kinds of XML tools, simplifying the presentation of content on resource-constrained devices like cellular phones and other types of handheld devices.

Where We Are Now

HTML 4.01 was intended to be the final version of HTML. Despite XHTML's many improvements and wide adoption, many web developers continue to use HTML. Recognizing HTML's continued importance and the need for further improvements, W3C has resumed work on the development of a new version of HTML, HTML 5.0. This will provide web developers who support large amounts of HTML additional time to make the transition to XHTML. At the same time the W3C is also working on the next version of XHTML, XHTML 5. New web developers, however, should plan on working exclusively with XHTML. This will save the trouble of having to convert down the road.

Because HTML and XHTML are so closely related, what works in one language often works identically in the other language. As a result, web developers have taken to referring to both languages generically as (X)HTML. This text adopts this approach, using the term *(X)HTML* to refer to both languages, except where specific differences between the two languages need to be pointed out.

HTML and XHTML are sets of open standards maintained by the *World Wide Web Consortium* (*W3C*). The W3C is a non-profit organization dedicated to the development of open standards, ensuring that things on the Internet work smoothly by providing everyone with a consistent and agreed upon set of rules.

Cascading Style Sheets

In the early days of web page development, web page developers were limited to the default presentation capabilities built into HTML. However, HTML's presentation capabilities were very limited, forcing web developers to get creative. The result was often poor development techniques in which HTML elements were used in ways that were never intended. To address some of its presentation shortcomings, some of HTML's elements were enhanced to provide them with additional presentation capabilities. For example, numerous elements were given new attributes that could be used to control their spacing, color, text size, etc.

Unfortunately, the inclusion of both structure and presentation into the same markup only served to weaken both aspects. Of particular difficulty was the fact that embedding presentation directly into markup made it difficult to modify the appearance and presentation of a web page or website because altering the way things looked typically meant making changes to elements throughout web documents. The end result was markup that was more difficult to understand and update.

An answer to this challenge soon came along in the form of *Cascading Style Sheet* or *CSS*. CSS is a style sheet language that web developers use to specify the presentation of web page content. CSS is its own language with its own rules and syntax. CSS provides you with the ability to separate markup from presentation through the creation of style sheets that specify how web documents should be rendered when displayed.

CSS provides web developers with a means of applying a consistent look and feel to web pages. CSS lets you specify things like font type, color, and size as well as background styles, borders, and alignment. CSS alleviates the need for web developers to have to repeatedly configure the presentation of elements within web documents. Instead, CSS lets you define style rules once and then applies those rules to every matching page element.

CSS made its debut back in 1997. However, it got off to a relatively slow start. It took years for it to work its way into mainstream web development. Today, CSS is a widely adopted standard sponsored and maintained by the W3C. By teaching you how to control the presentation of your web pages using CSS, this book will help you to avoid making many of the mistakes that web developers have traditionally made when building new web pages and sites.

You will get an early introduction to CSS in Chapter 3 followed by a more in-depth presentation of CSS in Chapters 7 and 8. By the time you are done, you will know how to develop web documents whose content and presentation are kept separate, resulting in web documents that are more streamlined and easy to maintain and update. As a result, you'll be able to make major or minor changes that affect the look and feel of entire websites through the modification of CSS, without having to change your content and markup.

Getting Interactive with JavaScript

In order to create web pages that are truly dynamic and capable of doing more than simply displaying static content, you need to learn how to work with a client-side programming language. Many such languages are available. Of these, JavaScript is by far the most commonly used. JavaScript programs are small text-based scripts that are downloaded as part of web documents and then executed within web browsers.

HINT

A *client-side programming language* is one that executes within a browser on the user's computer as opposed to a server-side language, which executes on web servers located on the internet.

JavaScripts give you, as a web developer, the ability to create truly interactive content. You can write JavaScripts that respond to user activity such as mouse movements and key presses, or to validate user input, such as ensuring that a phone number is formatted correctly. JavaScript is an interpreted programming language. This means that scripts written in JavaScript are not precompiled (made executable) at development time. Instead, scripts are converted to an executable format only when downloaded into the web browser for execution. Because it is an interpreted language, JavaScripts tend to execute a little slower than other compiled programming languages. However, thanks to the speed of modern computers and Internet connections, this impact is hardly noticeable.

JavaScript is also an object-based programming language. As such, it views everything within an (X)HTML document and the browser as *objects*. One key feature of JavaScript that is essential to the development of interactive web pages is the ability to trigger the execution of scripts based on the occurrence of different types of events. An *event* is an action that occurs when the user interacts with your web pages. As explained in Chapter 9, events occur for all sorts of reasons, such as when visitors first access a web page or when they leave it. Events also occur when visitors click on form buttons, key input into text fields, and so on.

Introducing the Document Object Model

Web documents are made up of many different types of elements. As you will learn, web browsers organize these elements using something called the *document object model* or *DOM*. The DOM provides the ability for scripts and CSS to access and update the content and style of web documents.

DOM Basics

The DOM is provided by the browser. The DOM provides a means through which JavaScripts can access and interact with web document content. By programmatically manipulating DOM objects, web developers can dynamically update web pages. Every time a browser loads a web document, it renders an HTML page. At the same time, the browser creates a DOM tree map in memory specifying all of the elements in the web document.

NOTE

Back at the end of the 1990s, web browsers provided inconsistent support for the DOM, often putting their own unique spin on things. As a result, web browsers did not always display content the same way. In 1998, W3C published the first DOM standard. In April 2004, DOM 3 was published. DOM 3 is still the current standard. All modern web browsers support DOM 3. As a result, different web browsers now render much more consistent results when displaying web pages.

Within the DOM, objects are organized in a hierarchy. At the top of this hierarchy is the `document` object. The `document` object ties together all of the elements in the document into a tree-like structure. Every element defined in a web document is represented by its own object in the DOM. The result is a collection of related objects with parent, child, and sibling relationships. Using a scripting language like JavaScript, you can use these relationships as a means of navigating the DOM tree. Further, you can use methods and properties provided by these objects to interact with and control them, allowing you to do things like make elements appear and disappear. You can also use CSS and the DOM to modify web documents by changing things like their color, size, and so on.

Navigating the DOM Tree

As already noted, the `document` object sits at the top of the DOM tree as depicted in Figure 1.4. The `document` object provides programmatic access to all of the objects that make up a web document.

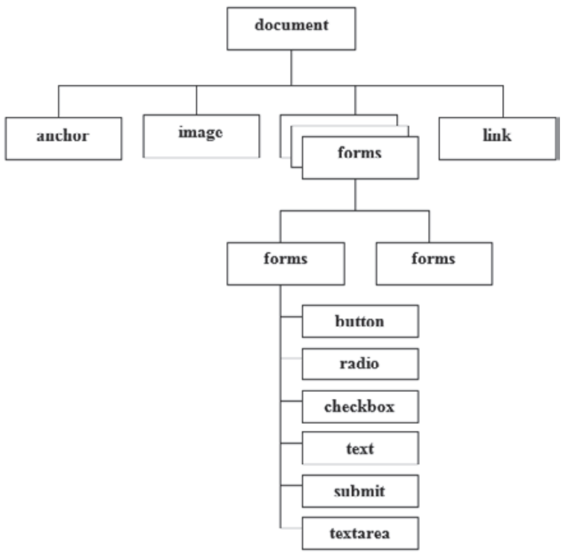


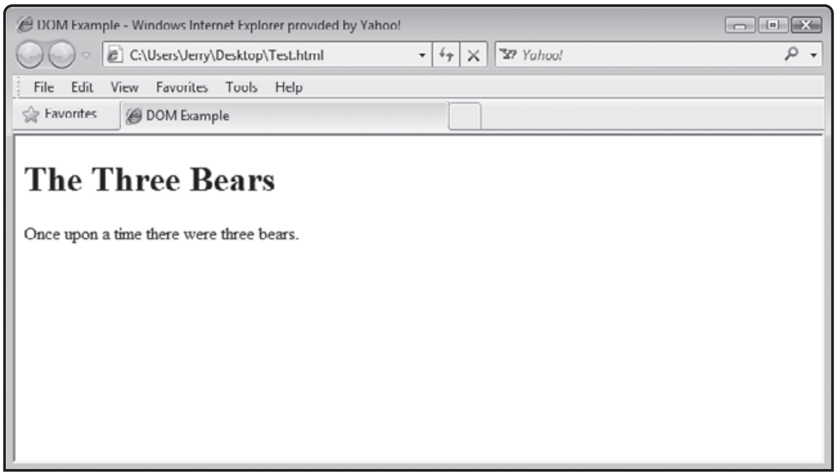
FIGURE 1.4

The `document` object provides easy access to all of the elements located on a web document.

The `document` object resides at the top of the tree. Underneath it are all of the other objects that make up the web document. As an example of how web documents are mapped out by the DOM, take a look at the following web document.

```
<!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" lang="en" xml:lang="en">

  <head>
    <title>DOM Example</title>
  </head>

  <body>
    <h1>The Three Bears</h1>
    <p>Once upon a time there were three bears.</p>
  </body>

</html>
```

This small XHTML page contains a title element, a level 1 heading (`h1`), and a paragraph (`p`) element. Figure 1.5 shows the web page that is displayed when this web document is loaded and rendered by Internet Explorer.


DOM Example - Windows Internet Explorer provided by Yahoo!
C:\Users\Jerry\Desktop\Test.html
Yahoo!
File Edit View Favorites Tools Help
Favorites
DOM Example

The Three Bears

Once upon a time there were three bears.

FIGURE 1.5

An example of the web document once it has been rendered by Internet Explorer.

The browser's internal view of the web document's content is quite different from what the web surfers see. Figure 1.6 provides a graphical representation of the DOM tree that the browser will create in memory when it loads the web document.

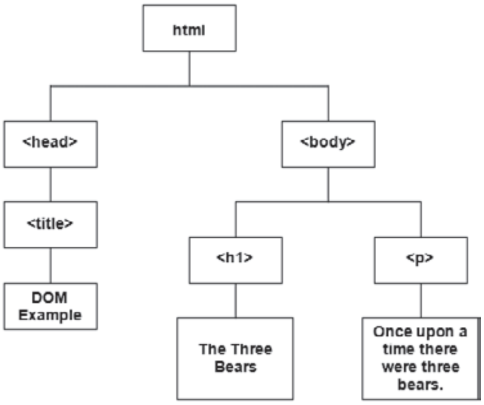


FIGURE 1.6

A graphic representation of the DOM tree that the browser will produce in memory as it renders the web document.

NOTE

As you will learn later in this book, you can assign an optional `id` attribute to the elements that make up your web documents. You can then use each element's assigned `id` to programmatically interact with the element or to alter its presentation by applying one or more CSS rules to it. To learn more about the DOM, visit www.w3c.org/DOM.

Understanding How Things Get Done on the World Wide Web

People use many different types of devices to connect to the Internet. Most people surf the Internet using their computer and a web browser like Microsoft Internet Explorer. A *web browser* is a software application that processes HTML and XHTML documents and renders web pages based on the contents of those documents. The following list identifies a number of today's most commonly used web browsers.

- **Internet Explorer.** Windows browser (www.microsoft.com/windows/Internet-explorer/).
- **Safari.** Mac OS X browser (www.apple.com/safari/).
- **Firefox.** Windows, Mac OS X, and Linux browser (www.firefox.com).
- **Opera.** Windows, Mac OS X, and Linux browser (www.opera.com).
- **Deepnet Explorer.** Windows browser (www.deepnetexplorer.com).
- **Konqueror.** Linux browser (www.konqueror.org).
- **Camino.** Mac OX S browser (caminobrowser.org).
- **Google Chrome.** Windows browser (www.google.com/chrome).

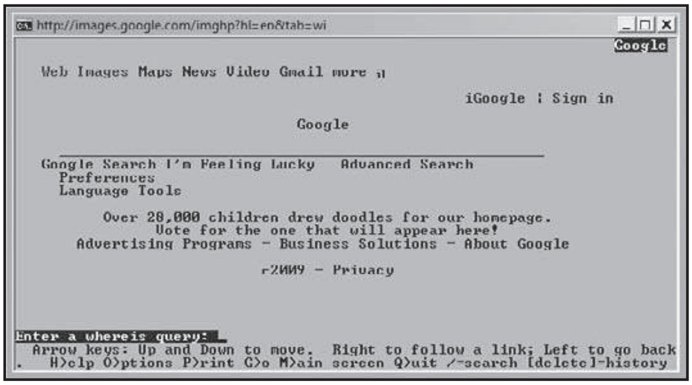
HINT

Despite all of the standardization that has occurred in recent years with HTML, XHTML, CSS, and JavaScript, there are still small differences in the way that browsers render content. As a result, it is important that you test your web pages using at least two or three of the browsers listed above.

There are, however, many other ways that people connect to the Internet, including things like Internet-enabled cell phones, specialized text-to-speech devices that assist visually challenged people who surf the Internet. Regardless of which software program or device people are using, they all have one thing in common: they are designed to process (X)HTML documents and to render web pages based on the content provided in those documents.

HINT

Some of the people that visit your website may not use any of the above browsers. These users might have visual problems and surf the Internet using special text-to-voice applications or they may use browsers like Lynx. Shown in Figure 1.7, Lynx only supports text-based browsing. Text-only browsing allows for really fast browsing.



```
http://images.google.com/imghp?hl=en&tab=wi
                                                                Google
   Web Images Maps News Video Gmail more »
                                                   iGoogle | Sign in
                                  Google
   ____________________________________________________________
   Google Search I'm Feeling Lucky   Advanced Search
     Preferences
     Language Tools

         Over 20,000 children drew doodles for our homepage.
               Vote for the one that will appear here!
     Advertising Programs - Business Solutions - About Google

                              ©2009 - Privacy

Enter a whereis query:
  Arrow keys: Up and Down to move.  Right to follow a link; Left to go back
     H)elp O)ptions P)rint G)o M)ain screen Q)uit /=search [delete]=history
```

FIGURE 1.7

Using the Lynx browser to visit `www.google.com`.

Most websites depend heavily on the use of graphics to help deliver their message. This can create a range of challenges for web developers. Fortunately, as you will learn in Chapter 5, there are ways of providing content specifically designed to service the needs of these types of users.

LINKING EVERYTHING TOGETHER

In order to access anything on the Internet, you must know its *URL* or *Uniform Resource Locator*. For example, to access Microsoft's website you would enter a URL of www.microsoft.com in your browser's URL field and press Enter. Every website has one. In fact, every page of every website has a URL as well. URLs are defined using the syntax shown in Figure 1.8.

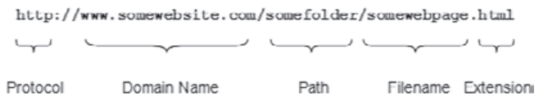


FIGURE 1.8

A URL is made up of multiple components that together identify the location of a resource on the web.

As you can see in Figure 1.8, every URL consists of the following components.

- **Protocol.** A set of rules that governs communications and the exchange of data between computers over a network. You will specify *http* (*HyperText Transfer Protocol*), which is a protocol that governs the transmission of hypertext-encoded data between computers on a network.
- **Domain Name.** Identifies the website where the specified web page or resources resides.

- **Path.** A hierarchical list of folders in which a web page of file resides.
- **Filename.** The name of the file to be retrieved and loaded.
- **Extension.** The file's file extension, typically .htm or .html.

URLs begin by specifying the http protocol, which is separated from the remaining text in the URL by the :// characters. Domain Name specifies the name of the website where the web page or file resides. Just like your computer, web servers can store files in a hierarchical collection of folders. Path specifies the folder in which the document is stored. The / character is used to separate the domain name from the path. The / character is also used to delineate every folder specified in the path. The path is followed by another / and then the name of the file to be retrieved, followed by a dot and the file's file extension, which in the case of web pages is either .htm or .html. Specifying the correct extension is important, since a website may have files that share the same name but different file extensions.

HINT

You may use either .htm or .html as the file extension of your web pages. The three-character .htm file extension dates back to the days of MS-DOS when all files had to have three character file extensions. You may use either file extension you want. Throughout this book the more familiar .html extension will be used.

URLs are essential to the operation of the web. They provide an easy, intuitive way of specifying where things reside. URLs are also used in the construction of links, providing the foundation for navigation between web pages. Path information in URLs can be specified using either absolute or relative values, both of which are discussed in the sections that follow.

Working with Absolute Paths

An *absolute URL* is one that specifies a text string that contains a fully qualified path that identifies the exact location of a resource on the Internet. An absolute URL specifies the protocol, domain name, complete path showing all folders as well as a filename and extension. Absolute URLs are used when creating links to external web pages that exist outside of your website. For example, the following string is an example of a typical absolute URL.

```
http://www.apple.com/education/it-professionals/index.html
```

Here, an absolute path of /education/it-professionals/ shows the exact location within which the index.html web page resides at the www.apple.com domain.

HINT

Most web servers are configured to automatically load a default web page when a URL specifies a folder instead of a filename. More often than not, the default web page is named `index.html`. So, if you wanted, you could rewrite the previous URL as shown below and the web server will serve up the same web page.

http://www.apple.com/education/it-professionals/

Relative Paths

A *relative URL* is one that specifies the location of a file relative to the location of the current web page. A relative URL allows you to point to a file location by specifying only its path and file information. Relative URLs allow you to shorten paths. For example, if you need to create a link to a file that resides in the same folder as the web page, all you have to do is specify the filename and extension of the other file as demonstrated here:

```
help.html
```

TRICK

Another advantage of working with relative URLs is that they can make the movement of websites from one server to another a lot easier. All you would have to do is copy all of the files that make up your website from one web server to another and as long as you used relative URLs to connect your web pages, everything should still work. No changes required.

If on the other hand, you need to refer to a file that resides in a parent folder, then use a relative URL, type `../` followed by the name of the file.

```
../help.html
```

If the file being referenced is, say, three folders up in the folder hierarchy, you do as demonstrated here:

```
../../../help.html
```

If it's easier, you can specify the location of a file starting from the location of the website's *root* or top-most folder. Just enter a / followed by the path to the folder where the file resides, as demonstrated here:

```
/projects/helpfiles/help.html
```

HINT

When a string indicating a path begins with the / character, that opening / character always represents the root folder.

Working with an (X)HTML Editor

The development of web documents differs from the development of other documents like word publishing and spreadsheets in one key way: to create and view the results of a web document in its final rendered form, you must work with two different applications. First, you must have a text or code editor with which you can create the text file that makes up your (X)HTML document. Once you have created and edited your web document, you must then use a web browser to view the resulting web page.

If after reviewing your work you decide to make additional changes, all you have to do is re-edit the text of the document's file and save it. Once complete, you can click on the web browser's refresh button to update the display of the resulting web page. In no time at all, you will quickly master the process and find the transition between editor and browser to be second nature.

HINT

There are plenty of high-end web editors available. Some of these editors include features like color-coding, automatic indentation, as well as access to prewritten code snippets. Some editors even come with built-in browsers of their own, saving you the hassle of having to work with two separate applications. I recommend that you stick with your computer's default text editor for now. This keeps things simple and allows you to focus on the fundamentals of web page development. Once you have mastered this, you can always upgrade to a higher-end editor and take advantage of all its bells and whistles.

Creating a Simple Web Page

This chapter has used the term *document* frequently to refer to plain text files containing (X) HTML statements. Documents, when rendered by web browsers, are used as the basis for creating web pages, made up of the content outlined in documents. All XHTML Strict 1.0 documents share a common format that must be strictly adhered to. This format consists of the `DOCTYPE` element, which is followed by the `html` element and its contents.

```
<!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" lang="en" xml:lang="en">

  <head>

  </head>

  <body>
  </body>

</html>
```

HINT

Note the extra use of blank spaces and lines in the previous example. Their usage has no effect on the resulting document other than to improve its readability for the developer. When rendered by a web browser, this extra white space is ignored. Also note the use of indentation for some of the document's elements, which is used strictly for the purpose of visually formatting and organizing the hierarchical relationship of elements to one another.

Lastly, note that the `DOCTYPE` element, located at the beginning of the document, though written on two lines, is actually a single statement. It was spread out over two lines to improve its presentation within this book. To show that the second part of the statement is related to the first part, the second part was indented two spaces.

The `html` element serves as the document's *root element* and acts as a container in which all other elements are stored. Specifically, the `html` element contains the `head` and `body` elements. The `head` element contains other elements that are used to provide information about the document and its contents. The `body` element contains all of the document's contents, which when rendered by the web browser is presented as web page content. Every XHTML document that you create will follow this same format, varying only if you decide to work with a different version of XHTML.

TRAP

Aside from the `DOCTYPE` element, every element in an (X)HTML document must be placed inside the opening `<html>` and closing `</html>` tags that make up the `html` element. Otherwise, the document is regarded as invalid resulting in unpredictable results.

If, however, you decide to work with HTML documents, both the `DOCTYPE` and opening `html` element will vary slightly, as demonstrated here:

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

<html>

  <head>

  </head>
  <body>

  </body>

</html>
```

In this example, a document adhering to HTML 4.01 version strict has been laid out.

Back to the HTML Joke Page

It is now time to turn your attention to the development of this chapter's project, the HTML Joke page. When loaded into the web browser, this document, named HTMLJoke.html, will present visitors with a web page that displays the text for a humorous joke along with the text for that joke's punch line.

Since this book has yet to introduce you to the intricacies of (X)HTML development, don't worry if you do not fully understand what each individual statement is doing. For now, keep your attention on the overall process being followed. Things will become clear as you make your way through this book. By the time you have finished this book, web pages like the HTML Joke page will seem quite elementary to you.

Designing the Application

To help keep things simple, the development of this application will be performed in three steps, as outlined here:

1. Create a new HTML document.
2. Develop the document's markup.
3. Load and test the HTML page.

Although this web project is relatively simple, its development will walk you through the basics steps involved in creating and testing most web pages. As long as you take your time and follow along carefully with all of the instructions that are provided, you should not have any trouble creating and then testing your own copy of this web document.

HINT

In order to be able to share your web pages with the world, you will have to find a web host. Advice on how to find a web host is provided in Chapter 2. For now, you will learn how to test the execution of your web pages, locally on your own computer.

Step 1: Creating a New HTML Document

The first step in the creation of this project is the creation of an empty text file. Begin by opening your preferred code or text editor; Microsoft Notepad will work just fine on Windows and TextEdit will do on Mac OS X. Of course, you can use any editor that can save its output as a text file. Create and save a new, empty text file. Name the file HTMLJokes.html and save it.

HINT

To keep things simple and make your web pages easy to access and maintain, consider creating a dedicated folder to store them in. For example, when developing the web documents for this book, a folder named HTML_Source was created on the computer's local C: drive. You'll see that reflected in the URL field of a number of the figures shown in this and other chapters.

Step 2: Developing the Document's Markup

The next step in creating the HTML Joke project is to develop the web document's markup. Begin by adding the following elements to the HTMLJokes.html document.

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

<html>

  <head>

  </head>

  <body>

  </body>

</html>
```

These statements are identical to those for the HTML template that you were introduced to earlier in this chapter. These elements including the `DOCTYPE` element, which specifies the version of HTML being used, and the elements needed to outline the document's `html`, `head`, and `body` sections. At this point you have supplied everything needed to create a valid, well-formed HTML page. If you were to save and load this document into your browser you would see a blank web page. Of course, to create a web page of value, you must add content to it. In the case of the HTML Joke project, this includes the addition of both `head` and `body` elements.

Modifying the head Section

To complete the document's `head` section, modify it by embedding the `meta` and `title` elements as shown here:

```
<head>
  <meta http-equiv="Content-type" content="text/html; charset=UTF-8">
  <title>Chapter 1 - HTML Joke</title>
</head>
```

The `meta` element shown here is used to specify the content type and character set used by the document. The `title` element is used to specify a text string that will be displayed in the web browser's titlebar. You will learn more about how to work with both of these elements in Chapter 3.

Specifying Document Content

The web page's content is provided by the document's `body` section. It consists of a level 1 heading and two paragraphs that display a joke and the joke's accompanying headline. To update the document's `body` section with the elements required to tell the joke, modify the document's `body` section as shown here:

```
<body>
  <h1>HTML Joke</h1>
  <p>Why is the number 6 afraid of the number 7?</p>
  <p>Because 7 ate 9.</p>
</body>
```

Without getting into the specifics now, the `h1` element displays a heading identifying the joke. Then two `p` elements are used to display the requisite text content.

The Finished HTML Document

Once you have modified the document's `body` section, your copy of the web document is complete. The following example shows what the document looks like once it has been completely assembled.

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

<html>

  <head>
    <meta http-equiv="Content-type" content="text/html; charset=UTF-8">
    <title>Chapter 1 - HTML Joke</title>
  </head>

  <body>
    <h1>HTML Joke</h1>
    <p>Why is the number 6 afraid of the number 7?</p>
    <p>Because 7 ate 9.</p>
  </body>

</html>
```

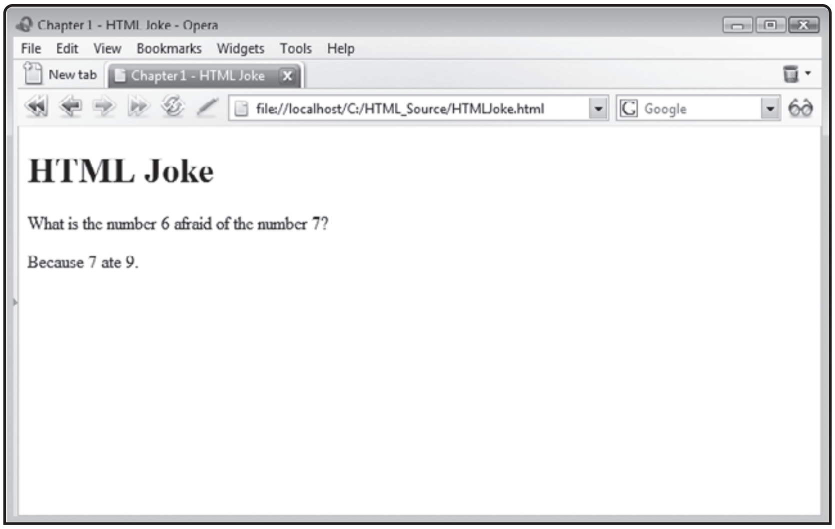
Although this book's primary focus is on the use of XHTML, its first project was done using HTML to demonstrate the similarities of the two markup languages. To convert this HTML document to an XHTML document, all you have to do is to modify the document's `DOCTYPE` and `html` elements as shown here in bold:

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

<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">

<head>
   <meta http-equiv="Content-type" content="text/xhtml; charset=UTF-8" />
    <title>Chapter 1 - XHTML Joke</title>
  </head>

  <body>
    <h1>XHTML Joke</h1>
    <p>Why is the number 6 afraid of the number 7?</p>
    <p>Because 7 ate 9.</p>
  </body>

</html>
```

Step 3: Loading and Testing the Web Page

As you are about to see, you can easily load and display your web document from your own computer using your web browser, without having to upload it to the Internet. This makes testing your document quick and easy. To do so, open your preferred web browser and then execute the following procedure.

1. Click on the browser's File menu and select the Open File command. This displays a standard file open dialog.
2. Using the dialog window, navigate to the folder where you stored the web page and select it.
3. Click on the Open button. The browser will then load and display the document, as demonstrated in Figure 1.9.


Chapter 1 - HTML Joke - Opera
File Edit View Bookmarks Widgets Tools Help
New tab
Chapter 1 - HTML Joke
file://localhost/C:/HTML_Source/HTMLJoke.html
Google

HTML Joke

What is the number 6 afraid of the number 7?

Because 7 ate 9.

FIGURE 1.9

An example of how the HTML Joke page looks when loaded using the Opera Web browser.

TRICK You can also load and test the HTMLJokes.html file by opening the folder in which you stored it and then double-clicking on it.

Well, that's it. As you can see, the steps required to create a basic web page are very straightforward. More complex web projects, involving multiple web pages and internal and external CSS and JavaScript files, require additional files and their development is a little more complicated, as you will see in later chapters.

SUMMARY

This chapter provided an introduction to HTML and XHTML, the markup languages used to develop web documents, and to Cascading Style Sheets or CSS, a web development language used to affect the appearance and presentation of web pages. You were also introduced to the document object model or DOM. Using the DOM, web developers are able to programmatically interact with and further control the presentation of web pages. You also learned how links are used to tie things together on and between web documents. On top of all this, you learned how to create your first HMTL and XHTML documents.

Challenges

1. **As currently written, the HTML Joke document displays a web page with a single joke shown on it as well as the joke's punch line. To make the page more interesting, consider adding additional jokes to the web document.**
2. **Try modifying the document title by changing the contents of its `title` element, as shown here, to a test string of your own choosing.**

```
<title>Chapter 1 - HTML Joke</title>
```

3. **Replicate these changes in the XHTML version of this web page.**