1

The Web, HTML, and Markup Languages

This book is about XHTML, the **eXtensible HyperText Markup Language**. It is based on HTML, the **HyperText Markup Language**, and conforms to the principles of XML (**eXtensible Markup Language**). There are a lot of definitions to introduce at the start of the book, especially for those fairly new to the Web. We will therefore leave discussions of XHTML and XML to Chapter 2, and instead discuss web basics and HTML in this chapter. (Those already familiar with the Web and HTML may skip this chapter, or quickly skim read it.)

In this chapter we will cover the following topics:

- what the Web is
- the different browsers and HTML editors
- the basics of the HTML language
- the mechanics of what goes on when you browse a web page
- what markup and parsing are

What Is the Web?

Web and Internet are two oft-confused terms and it's essential to get these two separate entities clear in your mind from the very start. The Internet is a network of linked nodes (computers) that started life as a Department of Defense project in the 1960s. It was in the era of nuclear paranoia and the DOD, concerned that their communications could be wiped out by a single nuclear strike, developed a computer network that would continue to function in the event of one or several of the routes through the network being damaged or destroyed.

So, the Internet is a network of interconnected nodes, in the same way that the subway system of a large city is a network of interconnected railway stations. The subway system is designed to carry *people* from one place to another; by comparison, the Internet is designed to carry *information* from one place to another.

While a subway system is built on a basis of steel (and other materials), the Internet uses a suite of networking protocols, known as **TCP/IP** (Transmission Control Protocol / Internet Protocol) to transfer information around the Internet. A **networking protocol** is simply a method of describing information packets so they can be sent down your telephone-, cable-, or T1-line from node to node, until it reaches its intended destination.

When the user tells the browser to go fetch a web page, the browser parcels up this instruction using a protocol called **TCP**. TCP is a transport protocol, which provides a reliable transmission format for the instruction. It ensures that the entire message is correctly packaged up for transmission (and also that it is correctly unpacked and put back together after it reaches its destination).

Before the parcels of data are sent out across the network, they need to be addressed, in the same way that you need to address a letter before you send it. So a second protocol called **Hypertext Transfer Protocol** (or **HTTP**) puts an address label on it. HTTP is the protocol used by the World Wide Web in the transfer of information from one machine to another – when you see a URL prefixed with `http://`, you know that the internet protocol being used is HTTP. All HTTP does is enable different computers to communicate with each other.

The World Wide Web on the other hand is the software, as such, that runs on the Internet. The architecture of the Web is based on a **client/server model**. The user's browser is the **client** and retrieves information held on a remote machine known as a **web server** that stores the different web pages and files being accessed. This web server can be located on a local network or could be found on the other side of the globe, it doesn't matter as the browser can use HTTP to find it. What was unique about the Web when it first appeared is that it provided a graphical rather than a text-based interface to the different services offered on the Internet.

History of the Web

The Web's origins can be said to stem from writing. A document or book has a structure, such as paragraphs and pages, and its text can contain references to other documents and books. An index helps to locate specific information within the work, and cross-referencing between sections allows us to follow a particular path through the book. The numbering of lines and verses further enhanced the ability of a reader to jump straight to a particular point of interest. Thus the basic document model for information on the web was set: pages of text, with links to other such pages, or to sections within those pages.

Although the Web as we know it today is very much a child of the nineties, the concept of a Web of electronically linked information is much earlier. For example Vanevar Bush in the 1940s, in an article entitled *As We May Think*, describes such a system, which he named the **memex**. Other visionaries include Douglas Englebart, who is credited with helping to develop the mouse and hypertext; and Ted Nelson, who coined the term **hypertext**.

For the modern Web, we must look to CERN, an international high energy physics research center near Geneva in Switzerland. In 1989 Tim Berners-Lee and Robert Caillau collaborated on ideas for a linked information system that would be accessible across the wide range of different computer systems in use at CERN. At that time many people were using TEX (a command-based document preparation system) and Postscript to prepare their documents; a few were using **SGML** (**Standard Generalized Markup Language**, discussed in *Chapter 2*). Tim realized that something much simpler was needed that would cope with all kinds of machines from 'dumb terminals' through to the high end graphical X Windows workstations. HTML (HyperText Markup Language) was conceived as a very simple solution, and matched with the HTTP network protocol. Both were to later evolve into something much more complex, but the initial simplicity made it very easy for other people to write their own browsers and servers (more of which later) and helped to propel the initial growth of the Web.

CERN launched the Web in 1991 along with a discussion list called `www-talk`. Other people thinking along the same lines soon joined and helped to grow the web by setting up Web sites and implementing browsers, such as, Cello, Viola, and MidasWWW. The break-through came when the National Center for Supercomputer Applications (NCSA) at Urbana-Champaign encouraged Marc Andreessen and Eric Bina to develop the Mosaic browser. It was later ported to PCs and Macs and became a run-away success story. The Web grew exponentially, eclipsing other Internet based information systems (such as WAIS, Hytelnet, Gopher, and UseNet).

Standardization efforts have either focussed on formalizing existing practices or have sought to define new features in advance of their widespread deployment. HTML 2.0 was developed in the Internet Engineering Task Force (IETF) to codify HTML as it was in mid-1994. HTML 3.2 was developed by the World Wide Web Consortium (W3C) to codify HTML as it was in early 1996. HTML 4.0, introduced in 1997, added new features for richer tables, forms and objects.

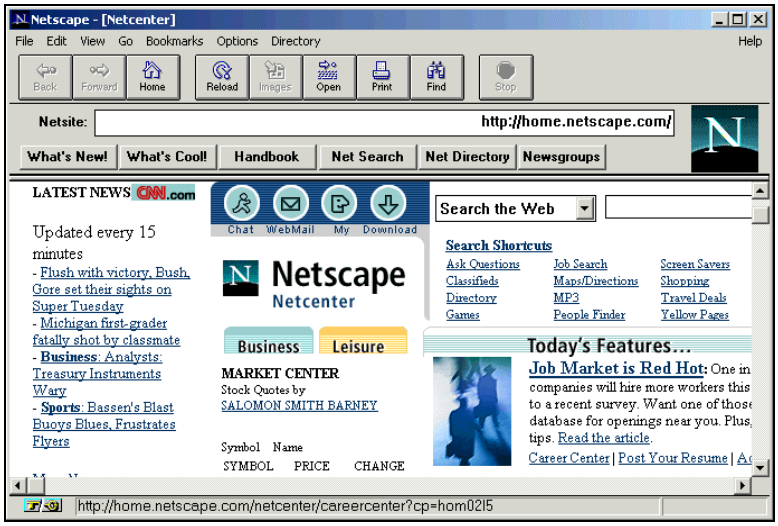
More recently, HTML 4.01 fixes a number of bugs in the HTML 4.0 specification, and provides the foundation for the XHTML specifications. The first of these is XHTML 1.0, which is the HTML 4.01 specification, but rewritten in XML. As we mentioned at the beginning of this chapter, we want to discuss XML and XHTML separately in the next chapter, so we will leave further discussion on these subjects until then. The World Wide Web Consortium is continuing work on XHTML and new specifications will be announced during 2000. You can track this work on `http://www.w3.org/MarkUp/`.

Browsers

As you can see from the history, the development of HTML standards hasn't been entirely linear. It has progressed from versions 1.0 and 2.0 of the standard straight to version 3.2 and then to 4.0, yet with no sign of a version 3.0 in between. This is down to the fast development of non-standard features in the browsers. This reached a head with version 3.0 of the standard, which was effectively obsolete before it moved out of the draft stage, and had to be amended to accommodate changes in the new version 3.0 browsers. Let's put this all into context, by giving a deeper overview of browser evolution and their changing capabilities.

A History of the Browser

We'll start with the very first web browser, which was NCSA Mosaic created by Marc Andreessen and Eric Bina. It was developed at the University of Illinois in 1993 and it changed the whole course of the Web. It allowed users for the first time to access web pages using a Graphical User Interface (GUI). Mosaic was developed for the Windows and Macintosh platforms, and it grew rapidly but was soon replaced by its follow-up Netscape Navigator. Netscape came onto the scene in 1994 with version 0.9 of their Navigator browser, when there were over 20 competing browsers already, and made a large splash. There was an HTML standard in force at the time, but Netscape forged it's popularity by creating non-standard additions to the standard, such as background images and blinking text in version 1.1 of it's browser:



By version 2.0 it was able to support Java applets and it's very own scripting language, called JavaScript. Many additions to the Navigator browsers were incorporated into the new versions of the HTML standard, as the HTML standards struggled to keep pace with the rate of innovation. However, some of these innovations ran contrary to the spirit of the standards and indeed the purpose of HTML. It's true to say that HTML was only ever meant to govern the structure of documents on the web and suddenly found itself convoluted with all manner of presentation and style tags such as `<font>` and `<frame>`. Up until then style and structure were two very different aspects of a web page. This may seem like a sweeping statement, but consider the font of the text on a page and the color of the background. They have absolutely no influence on the underlying structure and can be kept totally separate from the actual web page. In fact we will be looking at how style and content can be divided in later chapters.

It wasn't until December 1995 that Microsoft even publicly acknowledged the importance of the Internet and announced an Internet strategy and by that point Netscape already owned 75% of the market. However while Microsoft gave Netscape a huge advantage, in March 1996 they promised to deliver a whole set of Internet technologies, and later in the year they developed version 2.0 of their prototype browser, Internet Explorer. This browser though couldn't even support HTML frames and certainly didn't boast the Java support of Netscape Navigator 2.0. It wasn't until Internet Explorer 3.0, Microsoft's first browser to offer scripting capabilities and component hosting properties in web pages that Microsoft was able to seriously challenge the dominance of Netscape Navigator.

Although Microsoft had caught up in many areas, they decided to follow a different track to Netscape by integrating their browser into their Windows operating system while continuing to give it away for free. As Netscape were charging a nominal forty dollars for the licensing of Netscape Navigator, it made it very hard to compete as their main rival was giving the competition to its flagship product away for free. Worse was to follow for Netscape.

The competition for new exciting features intensified in version 4 of both browsers, as both companies competed to become the first to create a browser that allowed the user to re-arrange the contents of the page, without having to refresh the page. This new ability was dubbed Dynamic HTML, although not strictly being enabled by HTML. While Netscape released their version of their browser first, they were stymied when Microsoft's version 4 browser and it's own version of Dynamic HTML was adopted by the standards body as the standard way of enabling dynamic content in web pages. Faced with learning two widely differing interpretations of the same feature, the public opted in droves to use the Microsoft's browser and the standard Dynamic HTML. Netscape were forced to issue a humiliating public apology for not adhering to the standards and promised that the next upgrade of the browser would fall in line with the standards. Their market share slumped to 20% (`http://browserwatch.internet.com/stats/stats.html`) and has remained there, or thereabouts, ever since.

Microsoft's dominance of the browser marketplace hasn't all been smooth running, when their integration of Internet Explorer into the Windows operating system was challenged in court by the government as being an anti-competitive practice. This court case is still ongoing (as of writing) and forms a plank of the Department of Justice's case that Microsoft is a monopoly engaged in anti-competitive practices. This hasn't stopped version 5 of Internet Explorer being built into Windows 2000. IE5 offers a wide range of Dynamic HTML and XML support. Netscape tried to compete by making the source code to Navigator freely available as Open Source and encouraged an army of volunteer developers to create a new version of the browser from scratch. They've skipped a version and are now looking to ship Navigator 6 at the end of the year.

Browsers and HTML

The reason we've discussed the history at some length is because it has a large bearing on HTML – you depend on your browser's particular interpretation of HTML to render web pages in the way it's intended. So if you have a version 2 browser, then a lot of the features of the HTML 3.2 and 4.0 standards just won't be accessible to you. You have to install the latest version of Netscape's or Microsoft's (or one of the very small rival browsers such as Opera) to be able to take advantage of the latest features in the standard.

Here's a quick summary of what version browser supports what level of features.

Version	Internet Explorer	Netscape Navigator	Opera
Version 2	Didn't support frames, Java or any script and was a pretty sparse affair.	Supports JavaScript, Java, and support for reading mail and newsgroups.	Version 2.1 was the first public release, which boasted a very small download size and multiple windows, but not much above basic HTML.

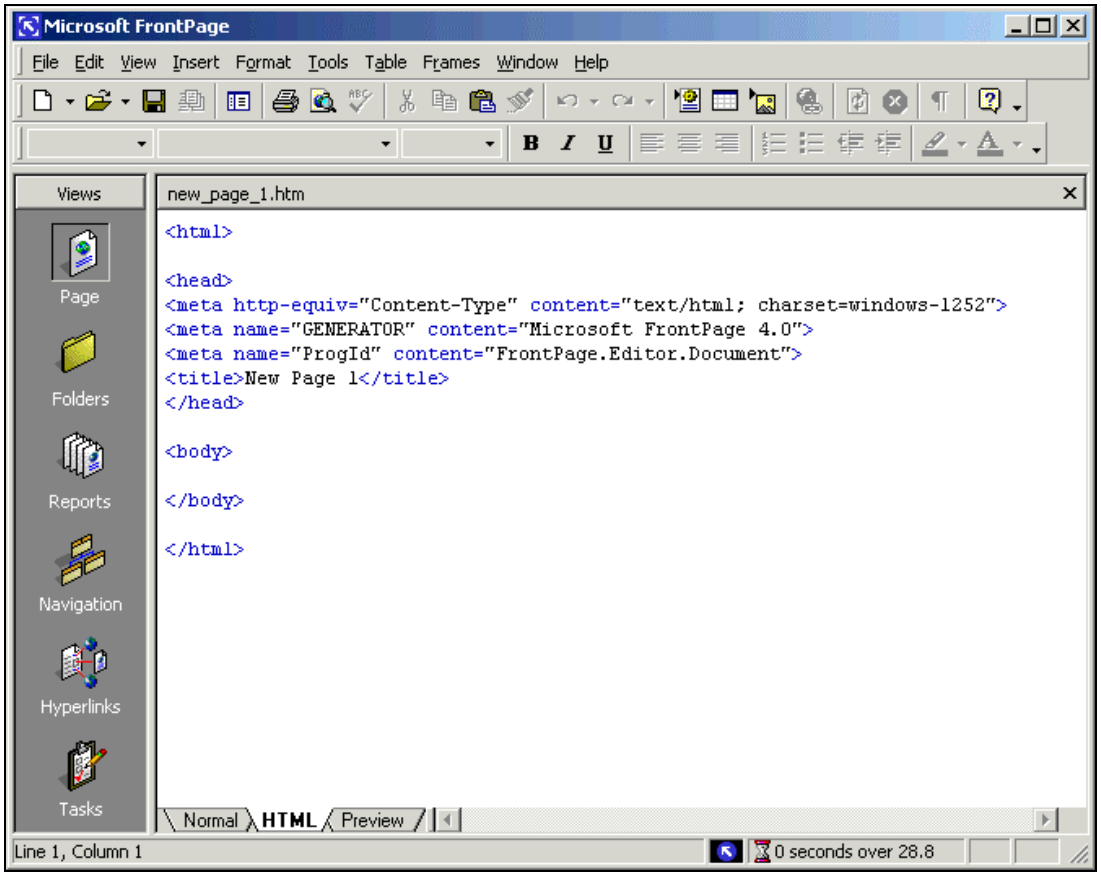
Version	Internet Explorer	Netscape Navigator	Opera
Version 3	Supports VBScript and basic style sheets and ActiveX controls. Versions 3.1, 3.2 and 3.3 were essentially bug fixes.	Short-lived version with extra abilities such as table background colors, live audio and video integration, but no style sheet support.	Added JavaScript and Netscape plug in support. Versions 3.1 and 3.2 offered bug fixes and extra security features.
Version 4	Dynamic HTML, much improved style sheets, JavaScript support (known as JScript), some HTML 4.0 features such as `<object>` tag (which allowed you to embed a multitude of components).	Limited style sheet support, Dynamic HTML supported through the non-standard `<layer>` tag, and a whole suite of mails, news, and messaging tools. Versions 4.5 and 4.65 didn't add much above better security and bug fixes. Latest version is 4.72	Version 3.5 was a new version in it's own right and added Java, style sheet, and SSL support. 3.6 is the latest version.
Version 5	Introduction of new dynamic features, behaviors and addition of XML and XSL. Most advanced browser available by some way. 5.5 is the latest release.		

Text Editors

There is a gigantic multitude of text editors on offer, some costing as much as several hundred dollars, and many given away for free. We're not going to recommend any particular page editor, but we'll list several of the most popular on offer.

Microsoft FrontPage

FrontPage 2000 comes as part of Microsoft's Office 2000 suite – it's a tool for creating and designing web pages. It offers three views of the web page. The Normal tab gives a WYSIWYG (**W**hat **Y**ou **S**ee **I**s **W**hat **Y**ou **G**et) page creation view (like the Design view in Visual InterDev), which allows you to write pages without having to code the HTML explicitly. The HTML tab allows you to write your code explicitly, and the Preview tab gives a quick view of what the page should look like in a browser:

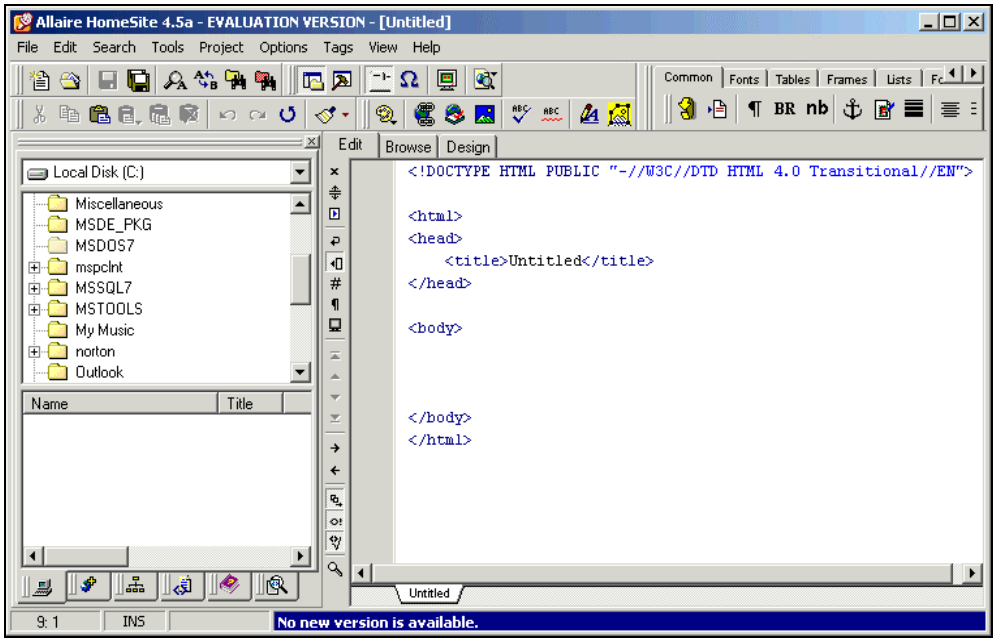


This screenshot is taken from FrontPage 2000, but you can also use Front Page 97 or 98 to edit HTML pages. The Normal and Preview tabs are able to process your HTML and give you a good idea of what your web page will look like when viewed in a browser.

One quirk of FrontPage is that it likes to 'improve' your HTML, by rearranging it. FrontPage 2000 now has a Preserve existing HTML option on the Tools | Page Options | HTML Source dialog, but older versions will still 'autoedit' your HTML for you. Beware – this window dressing can change your code and even affect the intended function of the code.

Allaire's Homesite

One of the best non-Microsoft web page editors is **Allaire's Homesite**. The evaluation copy of version 4.0.1 is currently available from their web site at `http://www.allaire.com`. The evaluation copy allows you to run the program 50 times or for 30 days –whichever elapses sooner. It features an easy-to-use interface, which allows you to keep track of your files and folders at the same time as your file contents:



In short, Homesite is a very powerful editor, and well worth a look.

There are other editors, such as Sausage Software's HotDog (`http://www.sausage.com`), SoftQuad's HotMetal Pro (`http://www.softquad.com`), or Adobe's PageMill (`http://www.adobe.com/products/`); but if you only want a completely free editor then Arachnophilia (`http://www.arachnoid.com/arachnophilia/index.html`) is very capable. They are all useful tools with which to create web pages.

Notepad

Of course if you have Windows in any shape or form, then you already have an editor. Notepad might be a time-honored text editor, but no matter how much Microsoft promote FrontPage, there will always be people who will use Notepad as their web page editor of choice. The fact that it's been free with every incarnation of Windows certainly helps sustain its popularity.

Of course, it doesn't highlight your HTML in any way, but also it doesn't auto-generate any extra code. It doesn't feature any additional functions; but, because it's so simple, Notepad is still a very popular choice. In Windows 2000, Notepad offers a GoTo feature (under the Edit menu), which allows you to move around your documents using line numbers.

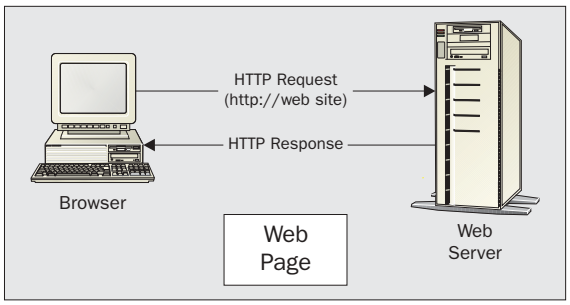
It doesn't really matter which editor you use in this book - it won't affect how you run the examples. We'll avoid any attempt to provide a tutorial on how to use any of these editorial tools – since most advanced editors have their own tutorials, and this is really beyond the scope of the book.

Now we've considered the different browsers and text editors, we'll take a look at the process involved when you start up your browser and view a web page.

What Happens When You View a Web Page

When you view a web page, you send a message from your browser to a web server, saying that you wish to view a particular page. The message passed from the browser to the web server is known as an **HTTP request**. When the web server receives this request, it checks its stores to find the appropriate page. If the web server finds the page, it parcels up the HTML contained within (using TCP), addresses these parcels to the browser (using HTTP), and sends them back across the network. If the web server cannot find the requested page, it issues a page containing an error message (in this case, the dreaded Error 404: Page Not Found) – and it parcels up and dispatches that page to the browser. The message sent from the web server to the browser is known as the **HTTP response**.

Here's an illustration of the process as we understand it so far:



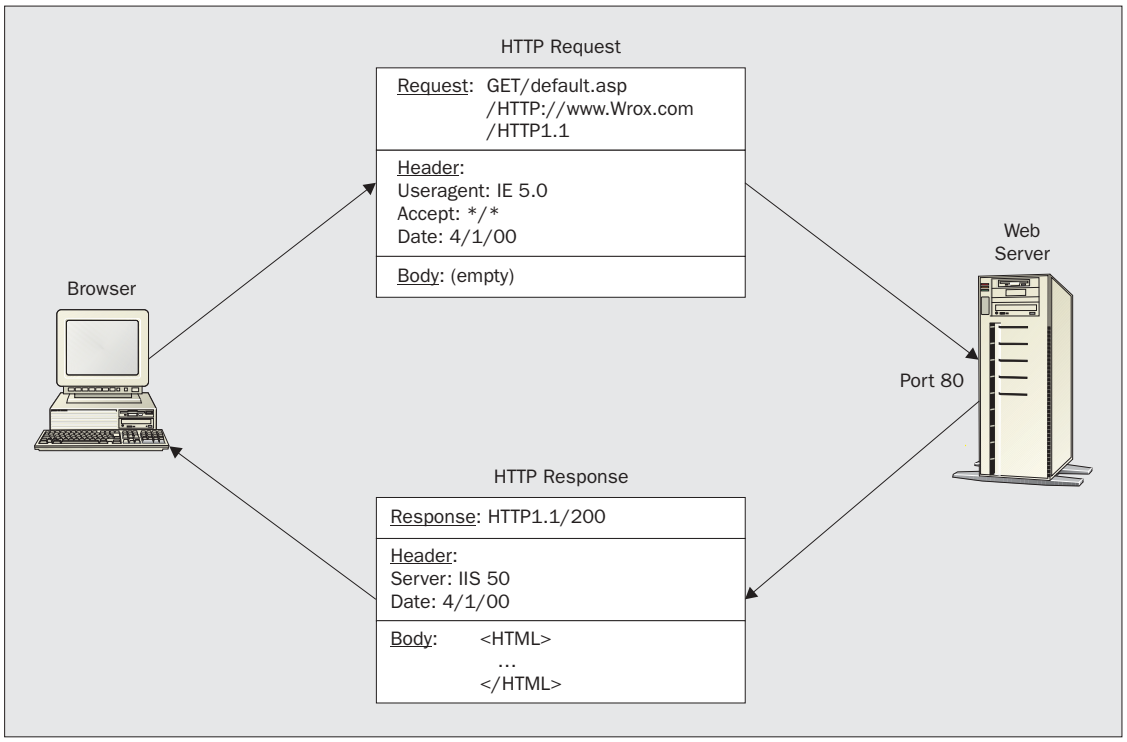
Going Deeper Into HTTP

There's still quite a lot of technical detail missing here, so let's dig further down and look more closely at exactly how HTTP works. When a request for a web page is sent to the server, this request contains more than just the desired URL. There is a lot of extra information that is sent as part of the request. This is also true of the response – the server sends extra information back to the browser. It's these different types of information that we'll look at in this section.

A lot of the information that is passed within the HTTP message is generated automatically, and the user doesn't have to deal with it directly, so you don't need to worry about transmitting such information yourself.

Every HTTP message assumes the same format (whether it's a client request or a server response). We can break this format down into three sections: the **request/response line**, the **HTTP header** and the **HTTP body**. The content of these three sections is dependent on whether the message is an HTTP request or HTTP response – so we'll take these two cases separately.

Let's illustrate our understanding of the process now:



We can see that the HTTP request and HTTP response have broadly similar structures and that there is information that is common to both, which is sent as part of the HTTP header. There are other pieces of information that can only be known to either the browser or the server, and that are also sent as part of either the request or response. In this way your request for a web page is transmitted to the web server, and then in the response the web server serves information to your browser along with the HTML code needed to create the page. Your browser then uses this information and code to construct the page, which you can view. If the browser comes across a line it doesn't understand in the HTML code, it will skip it and move to the next one. This illustrates the Web as a client/server architecture at its most basic level. We're considerably simplifying what goes on here, as you don't need to know it in any more detail.

Of course, one important point is that you don't have to go out onto the Internet or your local intranet to get your web pages from a web server, you can just as easily browse pages on your own machine. For the examples on this book this is exactly what we suggest doing. For instance, I've created a folder called XHTML on my C drive and stored the web page `example.htm` in it. I can then browse it by typing the following into the browser:

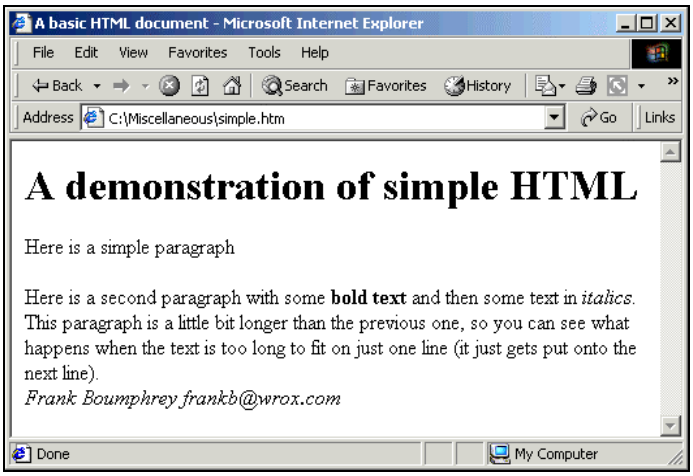
So let's move on and take a look at how we'd go about creating our web page in HTML.

HTML

HTML stands for HyperText Markup Language (we will discuss markup in the next section). Let's look at the general form of an HTML document. Type the following into your text editor and save it as `simple.htm`:

```
<html>
 <head>
  <title>A basic HTML document</title>
 </head>
 <body>
  <h1>A demonstration of simple HTML</h1>
  <p>Here is a simple paragraph
  <p>Here is a second paragraph with some <b>bold text</b> and then some text in
  <i>italics</i>. This paragraph is a little bit longer than the previous one,
  so you can see what happens when the text is too long to fit on just one line
  (it just gets put onto the next line).
  <address>Frank Boumphrey frankb@wrox.com</address>
 </body>
</html>
```

We see that the document consists of the text we wish to be displayed contained between the HTML commands or **tags** that inform the browser how the text should be displayed. If you point your browser to this file, it will be displayed as follows (here we are using Microsoft's Internet Explorer 5.0, but you can use any of the recent browsers we discussed earlier):



The general form of an HTML tag is:

```
<opening tag> text to be displayed </closing tag>
```

Both the opening and closing tags are enclosed within **angled brackets** (the name given to the 'less than' and 'greater than' symbols). Note that the closing tag differs from the opening tag only in that it contains a forward slash (`/`) before the tag name.

So, for example, if we want text to appear in **bold** we simply place that text between the HTML `<b>` and `</b>` for bold tags, as in:

```
<b>bold text</b>
```

Likewise, if we wish to place text in italics, we use the `<i>` and `</i>` for italics tags:

```
<i>italics</i>
```

As you can see, HTML is a simple language to grasp the basics of. Let's look at it in slightly more detail. Any HTML document should start with an opening `<html>` tag, and close with the matching `</html>` tag. This informs the browser that the document has been written in HTML. The HTML document contains two parts, a **head** and a **body**.

Material contained between `<head>` and `</head>` is *not* displayed on the browsers page. This section contains information about the document (meta-information), such as the title of the document. The text between `<title>` and `</title>` appears both on the title bar of the browser, and in the favorites or bookmarks list when you save the address of that particular site.

It is the material contained between `<body>` and `</body>` that is displayed on the web page. What we would normally think of as the title of the document is referred to as a **heading**, abbreviated simply to `h` in the HTML tags. There are a number of heading styles, the largest of which is `h1`. Thus the traditional 'title', or opening heading, is displayed using `<h1>` tags:

```
<h1>A demonstration of simple HTML</h1>
```

A paragraph of text is begun with the `<p>` for paragraph tag. In HTML, the closing `</p>` tag is optional. Thus the first paragraph in our example above is simply:

```
<p>Here is a simple paragraph
```

When HTML encounters a second `<p>` tag before encountering the matching and closing `</p>` tag for the first `<p>` tag, it logically assumes that we want to begin a new paragraph and thus, that the old paragraph has closed. Most HTML documents don't have a closing `</p>` tag, a problem discussed in terms of XHTML in Chapter 2.

Finally, we have the `<address>` tag. This normally contains the name and contact details of the Webmaster, the person responsible for writing/maintaining the web site.

Elements and Tags

Before we go any further, it is important that we define our terminology. Consider the following:

```
<h1>This is a level one heading</h1>
```

This entire item is called an **element**. `<h1>` is the **opening tag**, and `</h1>` is the **closing tag**. The text between these tags, This is a level one heading, is the **content** of the element. An element also has a **type**, named after its tag name, and so the example above is an `h1` element type. These points are illustrated diagrammatically here:

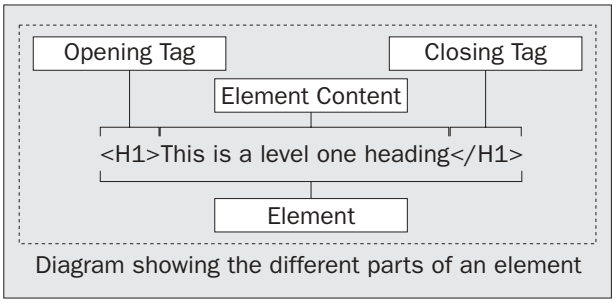


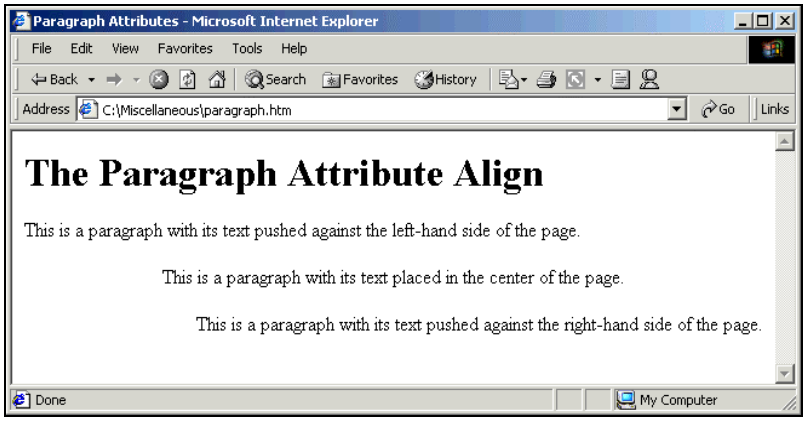
Diagram showing the different parts of an element

Tag Attributes

A tag can also possess one or more attributes, that is, additional information relating to how that element's content should be displayed. For example, the paragraph tag contains the `align` attribute, which can take one of the values `left`, `center` or `right`. Type the following into your editor and save as `attrib.htm`:

```
<html>
 <head>
  <title>Paragraph Attributes</title>
 </head>
 <body>
  <h1>The Paragraph Attribute Align</h1>

  <p align='left'> This is a paragraph with its text pushed against the left-hand
side of the page.

  <p align='center'> This is a paragraph with its text placed in the center of the
page.

  <p align='right'> This is a paragraph with its text pushed against the right-
hand side of the page.

 </body>
</html>
```

Pointing your browser to this file, you should see something similar to below:



We discuss attributes in more detail in Chapter 3.

Markup and Markup Languages

HTML is a **markup language**. You might be wondering what exactly is meant by the term 'markup'. Consider this: what we do when we apply markup (add HTML tags) to a document of any kind is to give added meaning to that document. If you highlight text in a textbook you are saying "this text deserves close attention", or if you tick an entry in your checkbook you are saying, "I have balanced this item". To use a computer term we say that the markup is semantic.

Semantic Markup

Semantic and semantics are very over worked words, but the word 'semantic' is just a derivation from the Greek word for significant. The Oxford English Dictionary defines it as 'Relating to significance or meaning'. So when we say that markup is semantic we are just saying that it has meaning to the browser.

You can often make a good guess at the meaning of a particular tag in the markup just from its name, tags such as `<font>` and `<form>` are self-explanatory, but unfortunately computers are not so smart and need more help. So, if we want a computer to read and act on our markup we have to give it an additional set of instructions on what to do. Sometimes we can build these instructions into the computer software itself; else we must provide the software with a separate set of instructions.

HTML is a markup language that is designed to be readable by both computers and humans. HTML markup is markup designed to be read by a special piece of software, the browser. All browsers have a built in knowledge as to the meaning of the markup, and is able to display the content in a standard way.

HTML Markup

HTML is just one markup language of a set of many markup languages created by a meta-language (a template for creating languages) called **SGML** (Standard Generalized Markup Language). SGML creates these languages via sets of rules known as **DTD**s (Document Type Definitions). Other members of this set include a language called DocBook, TEI, and MIL-STD, all of which are used in specific but limited situations. The main elements in HTML's popularity are first its simplicity to learn, because markup languages don't have tricky programming structures to learn such as you may find in Visual Basic, Java or C++; second, HTML can be automatically generated by many third party editors so you don't need to learn any HTML to create web pages; and third, that when a browser comes across a broken line of HTML it doesn't throw out an error like most other programming languages would, it just skips the broken code and carries on to the next line. Oh, and it's also down to the phenomenal growth of the Web itself!

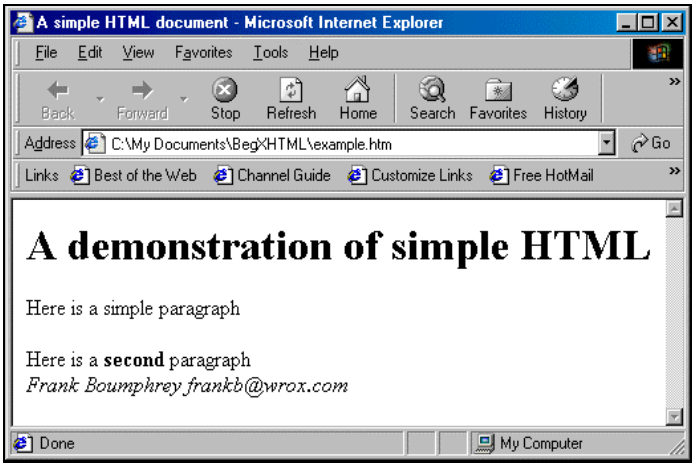
The concept of any SGML based language is quite simple. An SGML based language, such as HTML, is characterized by a series of tags taken from a dictionary of tags for the language. The browser contains a parser which can determine which tags form part of this dictionary. Probably the best way to explain this is to look at an example, so lets look at a simple HTML document right now.

Try It Out – A simple HTML document

1. Type the following into your text editor and call it `example.htm`:

```
<html>
<head>
<title>A simple HTML document</title>
</head>
<body>
<h1>A demonstration of simple HTML</h1>
<p>Here is a simple paragraph
<p>Here is a <b>second</b> paragraph
<address>Frank Boumphrey frankb@wrox.com</address>
</body>
</html>
```

2. You will notice that we have a series of simple sentences enclosed by tags. As we mentioned previously HTML is designed to be read by a piece of software called an HTML browser, so if I load this document into a browser, (in this case IE5) this is what I see:



Let's have a quick look at what is going on in the 'mind' of the browser. The following description is of course somewhat simplified.

How It Works: Parsing

Here is a brief description of how a browser may process the above document in order to achieve this display. This whole process is known as **parsing**. Parsing a document is akin to applying the rules of English grammar to a sentence. Immediately you'd know that the following sentence "The rain in Spain mainly on the plain" is missing a pretty vital ingredient, namely a verb. You know to expect a verb after a noun. The process of parsing in a browser is applying the same check to the HTML language. Of course a sentence that meets the laws of grammar doesn't have to make any sense, "The plain in Spain falls mainly on the rain", is grammatically perfect but doesn't make any sense. The parsing that the browser does then is an entirely structural check on the code provided. The parsing process in a browser works as follows:

1. The browser 'reads' the document from the beginning to the end looking for tags. The tags with the format <[tagname]> are called opening tags, and the tags with the format </[tagname]> are called closing tags.

2. Every time the browser comes across a "<" it knows it is going to be reading an opening tag, and every time it comes across a "</" it knows it is going to be reading a closing tag.

3. As soon as the browser sees a "<" it looks to see where the next ">" is. The text in between these characters is the tag's content. The first word of this content is the tag name, and is also the name of the element type.

4. Once the browser has read the tag name, the browser determines if it recognizes the tag name. If the browser recognizes the tag name, the browser decides what to do with the content between the opening tag and the closing tag.

5. The browser has a dictionary of all the tags that the browser supports, and this dictionary contains the details of how to process (display) the contents of the tag. If the browser can't identify a particular tag then the browser still needs to display the contents of the tag in some kind of default manner, normally this is text which is displayed in default font without any style. Again we are going to keep things really simple at this stage, but this strategy allows for both backward compatibility of the browser when new tags are created, and also protects against programming 'typos' disastrously altering the content of the document display.
6. When it comes across `<html>` it knows that this is the beginning of an HTML document.
7. When it comes across the `<head>` tag, it knows that this information contained from this point is generic information about the header of the document, such as the title.
8. When it comes across `<title>` it goes looking for a closing tag called `</title>`. It knows that it has to display the text between the two tags in the title bar of the browser.
9. When it comes across the `</head>` tag, it knows that that is the end of the information about the document header.
10. When it sees a `<body>` tag, it knows to expect the 'meat' of the document, the text that you intend to display in the browser window.
11. When it comes across an `<h1>` tag it knows that it is dealing with a level one heading(there are six level of headings each getting progressively smaller so that six is the smallest), and that the text between the opening and the closing `</h1>` tag must be displayed in the type of heading 1, of which the default is a roughly size 24 font in bold in most browsers.
12. When it sees a `<p>` tag it knows that it has to begin a new paragraph.
13. You will notice in the second paragraph that one of the words has been put between a pair of `<b></b>` tags. This is an instruction to the browser to render the text between the tags, the elements content, in a **bold** font.
14. The text between the `<address> </address>` is the address of the webmaster. The browser displays this in italics.
15. It comes across the closing `</body>` tag which is a precursor to the end of the document.
16. Finally the browser comes across the closing `</html>` tag. It knows that it has read to the end of the document.

What we have done here is really quite simple. We have taken a document and marked up its text so that the software (an HTML browser) can display it on a computer screen.

Structural, Stylistic, and Descriptive Markup

We mentioned earlier in this chapter that HTML was only meant to define document structure but has ended up also defining document presentation as well. As a result there are three types of HTML tags. The simple example above displayed the three chief kinds of markup:

1. **Structural markup**. This is markup that lays out the structure of a document. Paragraphs (`<p>`) and headings (`<h1>`) fall in to this category.

2. **Stylistic markup**. Quite a number of the HTML (and XHTML) tags are for styling the display. The Bold tag (`<b>`) falls into this category. As we will see later, this kind of markup is not encouraged because of the problems it causes when documents are rendered in non-visual media.

3. **Descriptive (semantic) markup**. This is also known as semantic markup, but descriptive is a better term. Descriptive markup describes the nature of the content of the element. `<title>` and `<address>` are the two examples above.

Structural Markup

Structural markup is always useful, and indeed is the *raison d'être* for XHTML. Structural markup gives a document's content form and integrity.

Stylistic Markup

Stylistic markup depicts the style in which the element content should be displayed. Stylistic markup however is a double-edged sword, and is really only applicable to documents designed to be shown on the screen or to be printed. Even then the design principals for the 'postcard' shaped screen are much different to that of the 'letter' shaped page. Furthermore, the different resolutions of screen and page means that a font that looks good in one media could look terrible in another! Professional designers will want to produce two different designs for these two different media. And of course they will want a totally different design for voice browser, or a Braille reader. For this reason, the styling of a document is much better left to a style sheet. We will cover document styling in great detail later in this book.

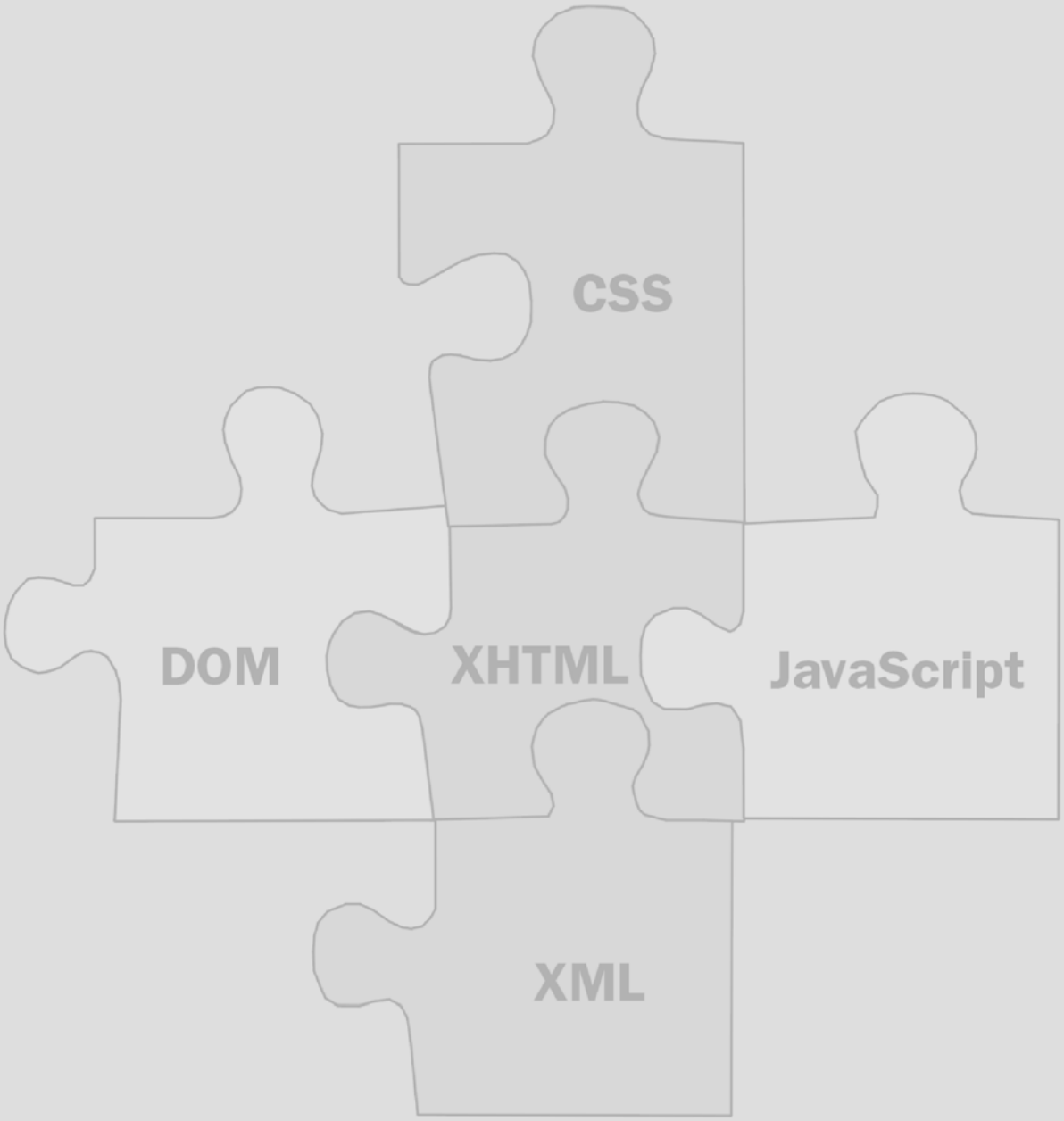
Some stylistic markup has gone into such widespread use that it will be almost impossible to eliminate it. I know that I find myself using it without thinking. However as stated above, for reasons of accessibility and cross-media compatibility it is always better to use descriptive markup.

Descriptive Markup

The trend in web markup is to go to more and more markup of a **descriptive** nature. One of the reasons for this is that it makes searching through the myriad of documents that are on the web much easier. This is one of the main reasons that XML was introduced as a markup language.

Summary

In this chapter we have introduced the World Wide Web and Internet and made clear the difference between the two terms. We looked at how the Web works, before turning our attention to some common browsers and text editors. There are a number of companies, which offer access to this Web, and most modern computers come equipped with software that allows you to surf the Web once connected to it. To view a web page, one simply types in that pages `http` web address into your browser. Web pages, up until now, have been written in HTML, a markup language the basics of which we have covered in this chapter. However, there are problems with this, which have led to the development of a new Web language called XHTML. This is based both on HTML and XML, topics we discuss next.


CSS
DOM
XHTML
JavaScript
XML