ELEC S212

Network Programming and Design

Unit 7

Internet application development

香港公開大學
THE OPEN UNIVERSITY OF HONG KONG

科技學院 School of Science and Technology

Course team

Developer:	Jacky Mak, Consultant
Designer:	Ross Vermeer, ETPU
Coordinator:	Dr Philip Tsang, OUHK
Member:	Dr Steven Choy, OUHK

External Course Assessor

Prof. Cheung Kwok-wai, The Chinese University of Hong Kong

Production

ETPU Publishing Team

The Open University of Hong Kong
Ho Man Tin, Kowloon
Hong Kong

This course material is printed on environmentally friendly paper.

Contents

Overview

In previous units you learned about the C programming language and Unix programming environment. You also learned how to apply some C programming techniques by developing some Internet applications. This unit builds on what you have learned in this course so far, and provides an introduction to Internet technologies, Internet application programming, and basic network game development.

This unit consists of seven sections. In the first and second sections, we introduce the basic features of the World Wide Web (WWW) and its underlying protocol, the Hypertext Transfer Protocol (HTTP).

In the third section, we introduce the interesting topic of writing webpages with HyperText Markup Language (HTML). Beginning with this section, you apply what you're learning by building your own website. You can practice publishing your website on the *ELEC S212* Linux server.

But HTML by itself only provides static webpages. Sometimes a website needs to respond to the user or to other outside variables. In order to make a website responsive to different needs, you need a program. In the fourth section, therefore, we introduce the concept of a script (a relatively simple program that involves few decisions and limited input that is stored in a document file). You will see that scripts can be classified as server-side scripts, which are executed on the Web server, and client-side scripts, which are included in webpages sent to the viewer's Web browser and are executed by the browser on the viewer's PC. You will also look at several server-side programming protocols such as CGI, ISAPI, ASP and SSI, and programming languages for Web programming such as C/C++, Perl, VBScript, Java, C#, and PHP.

After you gain knowledge of scripting languages, you can start building a more interactive website. The fifth topic describes JavaScript and lets you write interactive webpages. JavaScript is a very popular scripting language that can be embedded in an HTML document for adding interactive features onto webpages. JavaScript uses the processing power of client machines and contains lots of elements that you may use in writing interactive webpages. This section describes the overall model adopted by the JavaScript language and introduces some of the elements through examples.

In addition to JavaScript, you can build interactive webpages with Common Gateway Interface (CGI), introduced in the unit's sixth section. In contrast to JavaScript, CGI scripts perform the processing on the Web server. CGI scripts are only briefly described in this unit as a comparison with JavaScript.

In short, this unit:

- describes the features of the World Wide Web and its architecture;
- explains the use of programs on the Web;

- describes server-side and client-side scripting and the differences between them;
- discusses the popular programming languages used to build websites;
- outlines how to develop interactive webpages using JavaScript; and
- lists and describes the steps in CGI program development.

This unit is intended to take you five weeks (or approximately 40 hours) to complete.

The World Wide Web (WWW)

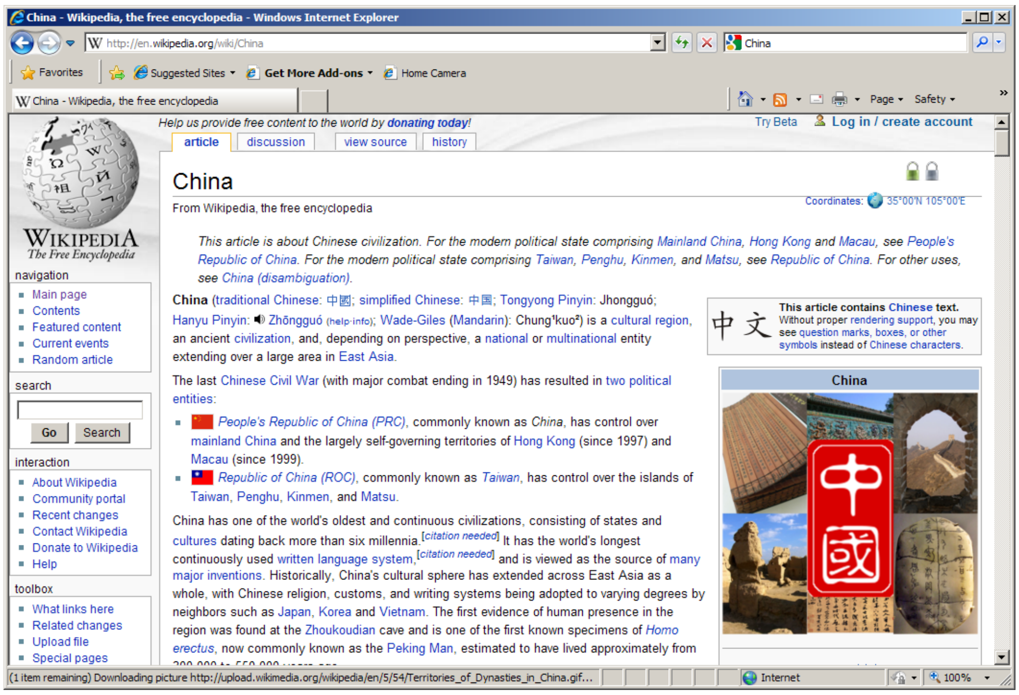
The World Wide Web (WWW), or simply the Web, should be familiar to you. It's likely that most of you use it every day. But for our purposes in this course, we need to get beyond even this daily use of the Web, and try to further look into what it really is.

What is the Web?

What do you usually use the Web for? What do you get from it? At least you read the online readings for this course, right? You may also read newspapers and get information on subjects you're interested in. So one thing the Web gives you is information; you can find information on subjects in nearly all areas: science, art, business, sports, etc.

But besides reading text, you have no doubt watched a video or listened to songs on the Web. In fact, it also provides you with all sorts of multimedia data. The information that it can provide is indeed boundless.

For example, Wikipedia (www.wikipedia.org) is a free, Web-based and collaborative multilingual encyclopedia. It contains over 13 million articles written collaboratively by volunteers around the world, and almost all of its articles can be edited by anyone who can access the Wikipedia website. If you have not visited Wikipedia before, visit its website shown in Figure 7.1 and search for any topic that interests you. You will be surprised by the wealth of information that is provided by Wikipedia!


China - Wikipedia, the free encyclopedia - Windows Internet Explorer

http://en.wikipedia.org/wiki/China

Help us provide free content to the world by ***donating today!***

Try Beta Log in / create account

article discussion view source history

China

From Wikipedia, the free encyclopedia

Coordinates: 35°00'N 105°00'E

This article is about Chinese civilization. For the modern political state comprising Mainland China, Hong Kong and Macau, see People's Republic of China. For the modern political state comprising Taiwan, Penghu, Kinmen, and Matsu, see Republic of China. For other uses, see China (disambiguation).

China (traditional Chinese: 中國; simplified Chinese: 中国; Tongyong Pinyin: Jhongguó; Hanyu Pinyin: Zhōngguó (help·info); Wade-Giles (Mandarin): Chung¹kuo²) is a cultural region, an ancient civilization, and, depending on perspective, a national or multinational entity extending over a large area in East Asia.

This article contains Chinese text. Without proper rendering support, you may see question marks, boxes, or other symbols instead of Chinese characters.

The last Chinese Civil War (with major combat ending in 1949) has resulted in two political entities:

- *People's Republic of China (PRC)*, commonly known as *China*, has control over mainland China and the largely self-governing territories of Hong Kong (since 1997) and Macau (since 1999).
- *Republic of China (ROC)*, commonly known as *Taiwan*, has control over the islands of Taiwan, Penghu, Kinmen, and Matsu.

China has one of the world's oldest and continuous civilizations, consisting of states and cultures dating back more than six millennia.[*citation needed*] It has the world's longest continuously used written language system,[*citation needed*] and is viewed as the source of many major inventions. Historically, China's cultural sphere has extended across East Asia as a whole, with Chinese religion, customs, and writing systems being adopted to varying degrees by neighbors such as Japan, Korea and Vietnam. The first evidence of human presence in the region was found at the Zhoukoudian cave and is one of the first known specimens of *Homo erectus*, now commonly known as the Peking Man, estimated to have lived approximately from

(1 item remaining) Downloading picture http://upload.wikimedia.org/wikipedia/en/5/54/Territories_of_Dynasties_in_China.gif...

Figure 7.1 Wikipedia is a free, Web-based and collaborative multilingual encyclopedia

How can there be so much diversified information on the Web? The information is not provided by a single organization or company. In fact, information is stored on computers all over the world that are connected to the Internet and act as **Web servers**. As you know, computers on the Internet are owned by different bodies, which may be organizations, companies, universities, etc., and this is the same for Web servers. Each body is responsible for the information on its own Web servers.

HyperText Markup Language (HTML)

The information provided by a website is accommodated by the webpages hosted on it. Whatever information a webpage contains, every webpage is created in **HyperText Markup Language (HTML)**. HTML holds the contents of a webpage together. The contents may include text, graphics, audio, video and even small programs (scripts) that can be executed on the **Web browser**.

HTML files that produce webpages are just text documents. As such, you can use a simple text editor to create an HTML file and then publish it on the Web. In the unit's first activity, you will use the text editor on your Windows PC to create your first webpage.

Hypertext and hyperlinks

As you know, there is a huge amount of information on the Web. But if this information is not arranged in an intelligent way, it may not be useful to you. So what information structure has the Web adopted? The Web in fact ties information together through links. When you read a webpage, you find some text that is often in different colors or underlined that shows that it is linked to other documents. The mouse pointer usually changes from an arrow to a finger when pointing to these links. When you click on these texts, the webpage it points to is retrieved and displayed automatically. Information on the Web is connected in this way. We call the link to another webpage a **hyperlink**, and the text showing the hyperlink we call **hypertext**. But you may find that pictures or other multiple objects also show hyperlinks. Instead of calling them hypertext, we call all the objects showing the hyperlink *hypermedia*.

In fact, much of the Web's value comes from its ability to link to pages and other resources (such as images, downloadable files, multimedia presentations, etc.) on either the same website or at another site. As you can go through a webpage with multiple paths through the hyperlinks, designing good webpages can be quite a challenge. Webpage authors have to think very carefully about how to navigate their readers through the pages.

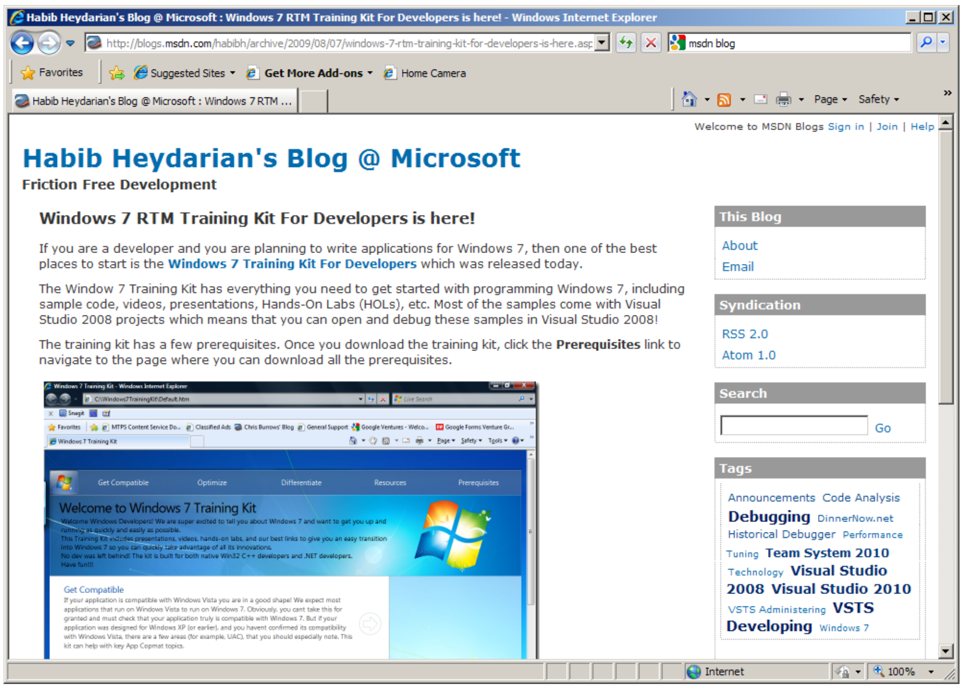
Markup

Indeed, HTML files are merely text files. If you open an HTML file with a text editor, the HTML contained will just be displayed as text. In order to view the corresponding webpage created by the HTML, you need to open the HTML file in a Web browser.

Consider the webpage shown in Figure 7.2. This page includes a heading that describes the webpage, several paragraphs of text, an image, and numerous hyperlinks (on the right) to other webpages. You may notice that different components of the page use different formatting features:

- The heading at the top of the page is larger than text in the subsequent paragraphs.
- The heading at the top of the page and the subsequent paragraphs at set at a different indentation on the left.
- Some text is in black, and some in blue.

How does the Web browser know how to display the various components as they are? In fact, the HTML contains markup that tells the browser how to display this webpage. Any text enclosed between angle brackets (`< >`) is an HTML tag (often called the *markup*). For example, a `p` within brackets (`<p>`…`</p>` tags) identifies the text in paragraphs. The markup between the `<style>` and `</style>` tags at the head of the file uses CSS (to be discussed later) to define the look and feel for various HTML elements used on this page. You embed the markup in a text file, along with text for readers to view, to tell the browser how to display your webpage.


Habib Heydarian's Blog @ Microsoft : Windows 7 RTM Training Kit For Developers is here! - Windows Internet Explorer
http://blogs.msdn.com/habibh/archive/2009/08/07/windows-7-rtm-training-kit-for-developers-is-here.asp
msdn blog
Favorites
Suggested Sites
Get More Add-ons
Home Camera
Habib Heydarian's Blog @ Microsoft : Windows 7 RTM ...
Page
Safety
Internet
100%

Welcome to MSDN Blogs Sign in | Join | Help

Habib Heydarian's Blog @ Microsoft

Friction Free Development

Windows 7 RTM Training Kit For Developers is here!

If you are a developer and you are planning to write applications for Windows 7, then one of the best places to start is the **Windows 7 Training Kit For Developers** which was released today.

The Window 7 Training Kit has everything you need to get started with programming Windows 7, including sample code, videos, presentations, Hands-On Labs (HOLs), etc. Most of the samples come with Visual Studio 2008 projects which means that you can open and debug these samples in Visual Studio 2008!

The training kit has a few prerequisites. Once you download the training kit, click the **Prerequisites** link to navigate to the page where you can download all the prerequisites.

Windows 7 Training Kit - Windows Internet Explorer

C:\Windows7TrainingKit\Default.htm

Get Compatible Optimize Differentiate Resources Prerequisites

Welcome to Windows 7 Training Kit

Welcome Windows Developers! We are super excited to tell you about Windows 7 and want to get you up and running as quickly and easily as possible.
This Training Kit includes presentations, videos, hands-on labs, and our best links to give you an easy transition into Windows 7 so you can quickly take advantage of all its innovations.
No dev was left behind! The kit is built for both native Win32 C++ developers and .NET developers.
Have fun!!!

Get Compatible

If your application is compatible with Windows Vista you are in a good shape! We expect most applications that run on Windows Vista to run on Windows 7. Obviously, you cant take this for granted and must check that your application truly is compatible with Windows 7. But if your application was designed for Windows XP (or earlier), and you havent confirmed its compatibility with Windows Vista, there are a few areas (for example, UAC), that you should especially note. This kit can help with key App Copmat topics.

This Blog
About
Email

Syndication
RSS 2.0
Atom 1.0

Search
Go

Tags
Announcements Code Analysis Debugging DinnerNow.net Historical Debugger Performance Tuning Team System 2010 Technology Visual Studio 2008 Visual Studio 2010 VSTS Administering VSTS Developing Windows 7

Figure 7.2 A webpage that incorporates numerous bits of HTML and CSS markup

Web browsers and Web servers

As you've just seen, the Web browser reads the HTML contained in a webpage and uses the HTML markup within to display its content on your screen. Usually, the Web browser requests and displays webpages available from various Web servers via the Internet (and sometimes via an Intranet). The Web browsers and Web servers use the application protocol called the **Hypertext Transport Protocol** (**HTTP**) to communicate with each other. We discuss HTTP briefly in the next section.

There are many Web browsers available, and each interprets HTML in its own way. The same HTML may not look exactly the same from one browser to the next. The variations will be minor, but these minor variations can be quite annoying. Therefore, it is recommended that you adhere to standard HTML as much as possible when authoring your webpages. If you do so, your webpages will have a better chance of having the same look whether the readers will be using Microsoft Internet Explorer,[1] Mozilla Firefox[2] or Apple Safari[3] to view your pages.

Uniform Resource Locators (URLs)

Every resource (a file or document) available on the Internet resides at a unique address called a **Uniform Resource Locator** (**URL**). A URL on the Internet is analogous to the address of a building. You use a document's URL to uniquely locate a specific document on a specific server in order to access the document, just as you use an address to uniquely locate a specific building in a specific country on earth.

The following figure identifies the components of a URL that is located on the *ELEC S212* course home page on the OUHK OLE Web server:

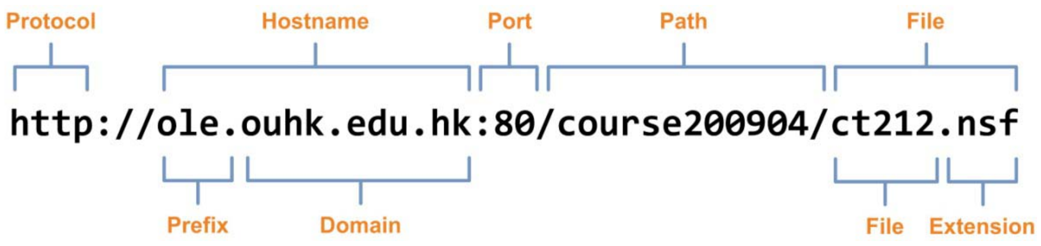


Figure 7.3 The components of a URL

1 http://www.microsoft.com/windows/internet-explorer/default.aspx

2 http://www.mozilla.com

3 http://www.apple.com/safari

Each URL component helps to define the location of a webpage or resource:

- The protocol indicates the application protocol to be used to retrieve the resource. For a webpage, the protocol is http.
- The hostname is the name of the website from which the browser will retrieve the resource. Recall from *Unit 3* that every computer connected to the Internet has a unique IP address, which is a 32-bit unsigned integer. The domain name is mapped to the actual IP address using DNS, as discussed in *Unit 3*. Many Web hostnames contain an optional domain prefix, which specifies the particular server to be accessed.
- If the optional port number is specified, the request will be made on this TCP or UDP port number (depending on the specific protocol being used). If it is not specified, then the port number will be inferred from the specified protocol. For example, the default port number is 80 for the HTTP protocol.
- The path specifies the directory on the Web server where the requested resource can be retrieved.
- The specific resource to be retrieved is identified by its file name and extension. An HTML (or XHTML) document will have an extension of `html` or `htm`. CSS files use the `css` extension, JavaScript files use `js`, JPEG images use `jpg` or `jpeg`, and so forth. A Web server can be configured to recognize these extensions and to then handle the files appropriately.

Indeed, most of the components of a URL are optional, depending on the particular server configuration. For instance, many Web servers are configured to automatically locate a default file when a directory is requested without specifying a file name. This could be the file named `index.html`, `index.php`, `default.html`, `default.aspx`, or something else, depending on the server setup. If a file with the default file name cannot be located, some servers respond with a list of files available in the specified directory.

The URL is the key instrument that allows you to build links to other resources on the Web, including other resources on your own website. You will use URLs extensively in the HTML that you author.

The following reading introduces the basic concepts in creating, designing and publishing websites. A few example websites are also used to illustrate the best practices in Web design.

Reading 7.1

Wagner, R and Mansfield, R (2007) *Creating Web Pages All-in-One Desk Reference for Dummies*, 3rd edn, Wiley Publishing, 9–37.

The architecture of the Web

The Web is a distributed client/server service in which a user using a Web browser can access a service provided by a Web server. The server, upon receiving the service request, responds with the appropriate action. Typically, the browser requests a page, and the server returns the requested page to the browser. The service is distributed because webpages are stored on many websites that are dispersed all over the world.

The browsers and servers communicate using the Hypertext Transfer Protocol (HTTP) (to be discussed briefly in the next section), which is built on top of TCP. The server listens to incoming requests on TCP port 80 (the default). As such, the operating systems of both clients and servers must support TCP/IP.

Figure 7.4 illustrates the stream of events that occur when you load a webpage in the Web browser:

1. You request a webpage by either typing its URL (for example, `http://ole.ouhk.edu.hk`) directly in the address bar of the browser, or by clicking on a link on the page.

2. The browser sends an HTTP Request to the server named in the URL and asks for the specific file. If the URL specifies a directory (not a file), it is the same as requesting the default file in that directory.

3. The server looks for the requested file and issues an HTTP response.

   a. If the page cannot be found, the server returns an error message. The message typically says ‘404 Not Found,’ although some servers may provide a more user-friendly message.

   b. If the document is found, the server retrieves the requested file and returns it to the browser.

4. The browser parses the HTML document. If the page contains images (indicated by the HTML `img` element) or other embedded objects (such as CSS files, JavaScript files and so forth), the browser contacts the server again to request each file specified in the markup.

5. The browser inserts each image in the document flow where indicated by the `img` element and displays the assembled webpage on your screen.

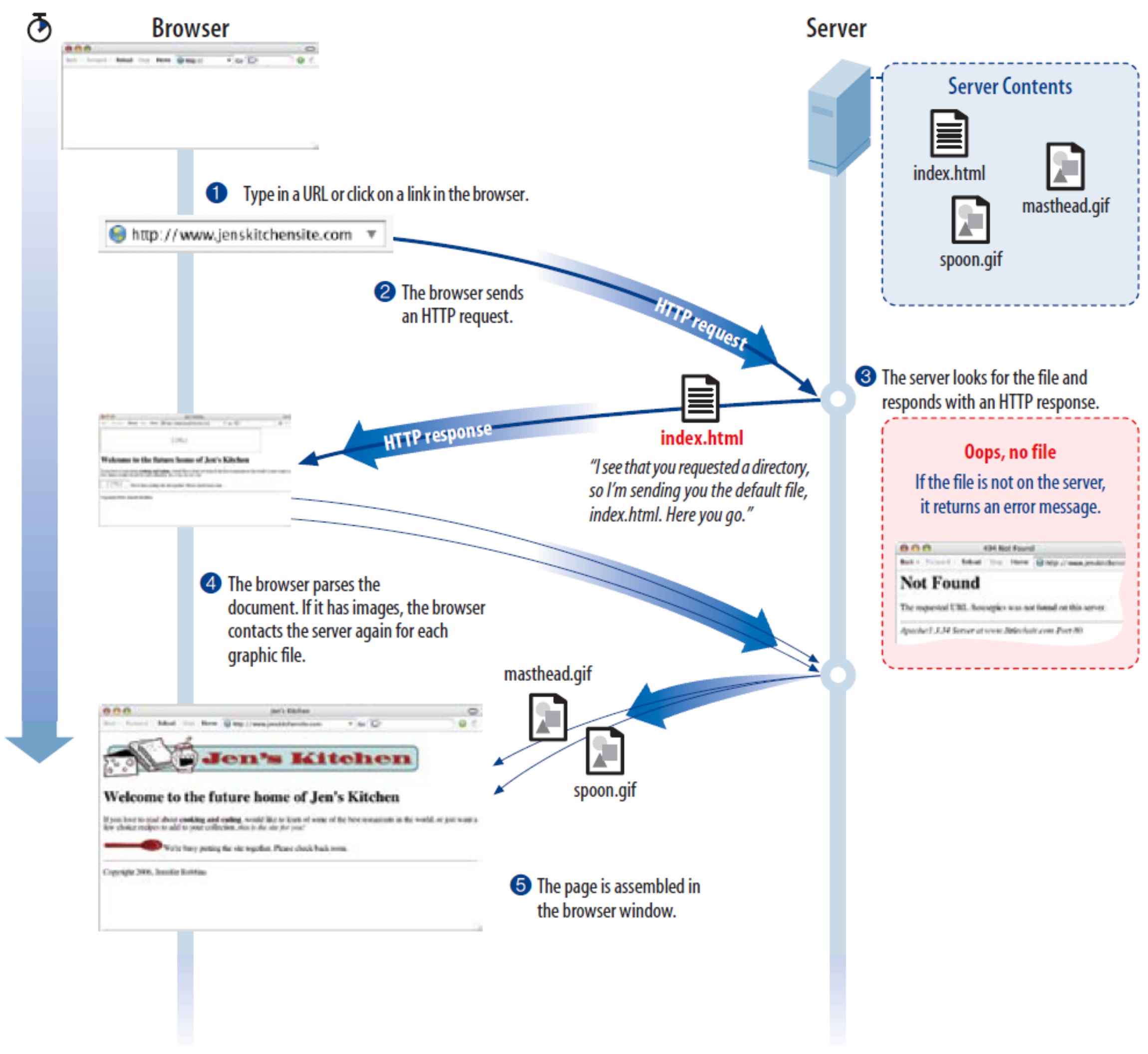

Browser
Server
Server Contents
index.html
masthead.gif
spoon.gif
1 Type in a URL or click on a link in the browser.
http://www.jenskitchensite.com
2 The browser sends an HTTP request.
HTTP request
3 The server looks for the file and responds with an HTTP response.
Oops, no file
If the file is not on the server, it returns an error message.
Not Found
HTTP response
index.html
"I see that you requested a directory, so I'm sending you the default file, index.html. Here you go."
4 The browser parses the document. If it has images, the browser contacts the server again for each graphic file.
masthead.gif
spoon.gif
Jen's Kitchen
Welcome to the future home of Jen's Kitchen
5 The page is assembled in the browser window.

Figure 7.4 How browsers display webpages (Robbins 2007, Figure 2-4)

Self-test 7.1

1 What type of model is used by the Internet, i.e. a model in which one computer stores information that is requested by one or many other computers?

2 Differentiate between a client computer and a server computer.

3 What is the difference between a webpage and a website?

Hypertext Transport Protocol (HTTP)

The **Hypertext Transfer Protocol (HTTP)** is an application-level protocol for a distributed, collaborative, hypermedia information system. It is the standard protocol for the Web that is used to obtain a webpage and also other items of information relating to a page such as an image, audio, video and so forth. The latest version of HTTP (1.1) is defined in RFC 2616.[4]

HTTP is a simple request-response protocol. A client sends a request message using a Web browser or other tool (for example, telnet) to the server's TCP port 80 (the default), and the server returns a response message. Both the request and response are simple text-based messages. Resources to be accessed by HTTP are identified using the Uniform Resource Locators (URLs) that you've already learned about.

The following Wikipedia article provides a very good introduction to HTTP. If you are interested in the details, you are recommended to read RFC 2616 thoroughly. In fact, if you want to develop your own Web browser or server, you must fully understand the protocol.

Reading 7.2 (online)

Wikipedia, 'Hypertext Transfer Protocol':

http://en.wikipedia.org/wiki/Http

Reading 7.3 (online)(optional)

RFC 2616, 'Hypertext Transfer Protocol — HTTP/1.1':

http://www.ietf.org/rfc/rfc2616.txt

[4] http://www.ietf.org/rfc/rfc2616.txt

Self-test 7.2

1 List all request methods defined in HTTP/1.1. Briefly describe each of them in one or two sentences. Which two request methods are used most often?

2 Give the status codes of the corresponding HTTP responses for the following cases:

   a The requested webpage is successfully retrieved.

   b The requested webpage cannot be found on the Web server.

   c The Web server is unable handle the request because it is currently too busy.

3 Give the HTTP/1.1 request to access the Hong Kong government website (`www.gov.hk`). Use `<CR><LF>` to denote the end of a request line.

HyperText Markup Language (HTML)

As you've seen, HTML is the predominant markup language for webpages. In this topic we'll explore this language in more depth.

A markup language is composed of a set of markup tags. HTML uses markup tags surrounded by angle brackets (<>) to define the structure of a webpage by showing heading, paragraph, bullet list, etc. HTML allows images and objects to be embedded and can be used to create interactive forms.

Some people think of HTML as a programming language, but it is not. It does not have program structures such as functions, or data structures like arrays and pointers. It does not need compiling; instead, the Web browser is responsible for interpreting HTML statements. The browser formats the webpage according to the local configurations such as its own settings, display card capability, monitor resolution and so on.

The HTML specifications are now maintained by the World Wide Web Consortium (W3C).[5] The latest version of HTML is 4.01,[6] which was published as a W3C Recommendation in 1999. Today, all major Web browsers support HTML 4.01.

In 2008, W3C published a Working Draft for the next version of HTML, HTML5.[7]

The features of HTML

HTML has the following features:

1 *It is compact* — An HTML document is just a plain text file. Even on a dial-up connection, the transfer time from the server to the client can still be very fast.

2 *It is platform-independent* — The browser is responsible for formatting the document, so you can view an HTML document on any platform that has a suitable browser.

3 *It is simple and easy to learn* — HTML tagging is simple. Everyone can learn it in a short time and then use it to put information on the Web to share.

4 *It is a free and open standard* — HTML is not owned or controlled by any company or individual. There is no license to purchase or

5 http://www.w3.org

6 http://www.w3.org/TR/html401

7 http://www.w3.org/TR/html5

specialized software required to author your own HTML documents. So anyone is free to create and publish webpages.

Although HTML defines the document structure, it does not define the document format. For example, HTML defines a text as the title of a page. However, the appearance of the text depends on the browser you use. It may be in boldface and have a larger font size, for example.

HTML does not define page layout either. In word processors like Microsoft Word, you can easily put a picture in a specific position on a page. You will find you are not able to do all this in HTML. Instead, you can use the Cascading Style Sheets (CSS) language, which is an HTML extension, to add the exact position and exact style in HTML. We do not talk about CSS in this unit.[8]

A well-known website that hosts a number of excellent tutorials and resources related to website building and Web development is w3schools.com (http://www.w3schools.com). We use the online tutorials provided on this website to supplement our discussion of HTML. To start off, read the following online reading.

Reading 7.4 (online)

w3schools.com, 'HTML Tutorial':

http://www.w3schools.com/html/

A simple website from beginning to end

In the previous section, you saw that when someone wants to look at a website, the Web browser contacts the Web server, and requests a webpage. The Web server responds by downloading the requested file to the browser, along with any other files that are needed to build the entire webpage. In creating a website, there is a two-step process:

1 creating the webpage files; and

2 publishing the website by placing these files on the server.

The following activity demonstrates how to create a webpage by using any text editor such as Notepad or WordPad. It is recommended you store your webpage files in one folder with sub-folders on your hard drive (e.g. `C:\CT212\Web`). In the next activity we demonstrate how to publish the file by using FTP, a command line program that is available on Windows PCs. You also need to change the access rights of the assigned folder for webpage storage and any sub-directory in your student account (as you learned in Lab 1.1 of Kwan 2009).

[8] If you are interested in learning CSS, we recommend you read the CSS tutorial on w3schools.com (http://www.w3schools.com/css).

Creating a webpage

A webpage is one file that contains HTML text. You can create this HTML file by using any text editor such as Notepad. Before being introduced to the syntax of HTML, look at the following example to get an idea of what an HTML document is like. In order to get you involved, this example is presented as an activity for you.

Activity 7.1

Creating a webpage with Notepad in Windows

1 Launch Notepad by clicking **Start** → **Programs** → **Accessories** → **Notepad**.

2 Type in the following (substitute `#your name#` with your name):

```
<HTML>
<HEAD>
<TITLE>My First HTML Document</TITLE>
</HEAD>
<BODY>
<!--This is a comment-->
<H1>Home Page of #your name# </H1>
<P>
Welcome!! This is the first webpage I created in the course
  CT212!
</P>
<P>
Thank you for visiting!!
</P>
</BODY>
</HTML>
```

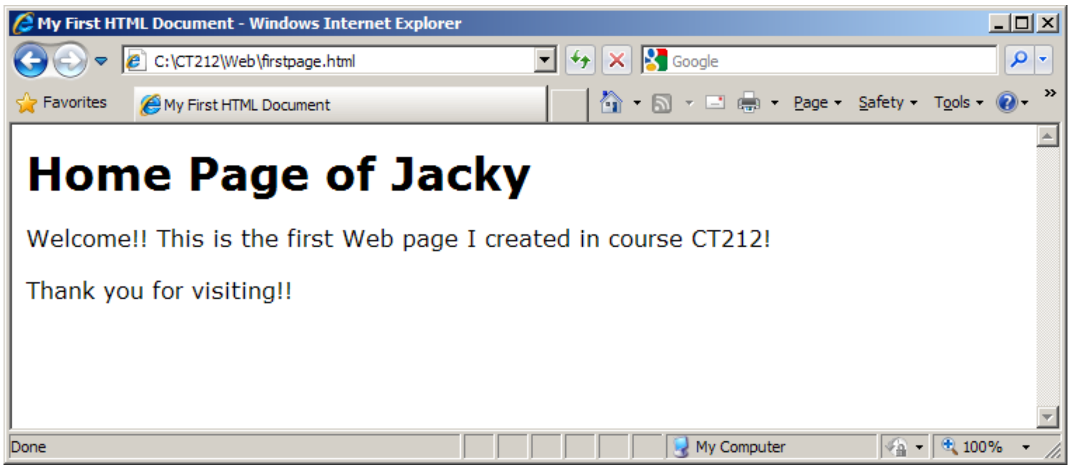
3 Save the file as `firstpage.html` in a folder of your choice, for example, `C:\CT212\Web`.

4 Start your favorite Web browser.

5 In the address bar of the Web browser, type the URL of the file just created. When you are looking at an unpublished webpage — i.e. one that is stored on your PC — the URL is simply the path of the file. In this example, the URL is `C:\CT212\Web\firstpage.html`.

Congratulations! Your webpage should display as shown in the following figure. When you publish this webpage on a Web server, others will be able see it through the Internet as well. You have successfully created a webpage!

The details of the HTML of this webpage will be described later in this section.


My First HTML Document - Windows Internet Explorer
C:\CT212\Web\firstpage.html
Google
Favorites
My First HTML Document
Page
Safety
Tools
Done
My Computer
100%

Home Page of Jacky

Welcome!! This is the first Web page I created in course CT212!

Thank you for visiting!!

Figure 7.5 Your first webpage for *ELEC S212* opened locally on your PC.

Publishing the website

When you publish a website, you copy to the server all of the pages and supplemental files needed to build the website. The best tool for transporting files from your PC to the server is FTP, which is available on both Windows and Linux. You should now do the following activity to publish your website on the *CT212* Server.

Activity 7.2

Publishing a website

1 Do Step 1 of Lab 1.8 of Kwan 2009 if you have not done so. When you finish this step, you should have a directory called '`public_html`' under your home directory. The access rights of this directory should be set to 711.

2 On your Windows PC, launch the Command Prompt by clicking **Start** → **Programs** → **Accessories** → **Command Prompt**. The Command Prompt window opens.

3 Execute the following commands in the Command Prompt window to upload your webpage to your '`public_html`' directory on the *ELEC S212* Server:

   a Change to the directory where you saved your `firstpage.html` in Activity 7.1.

   b Enter the command: `ftp labsupport.no-ip.org` to connect the *ELEC S212* FTP Server.

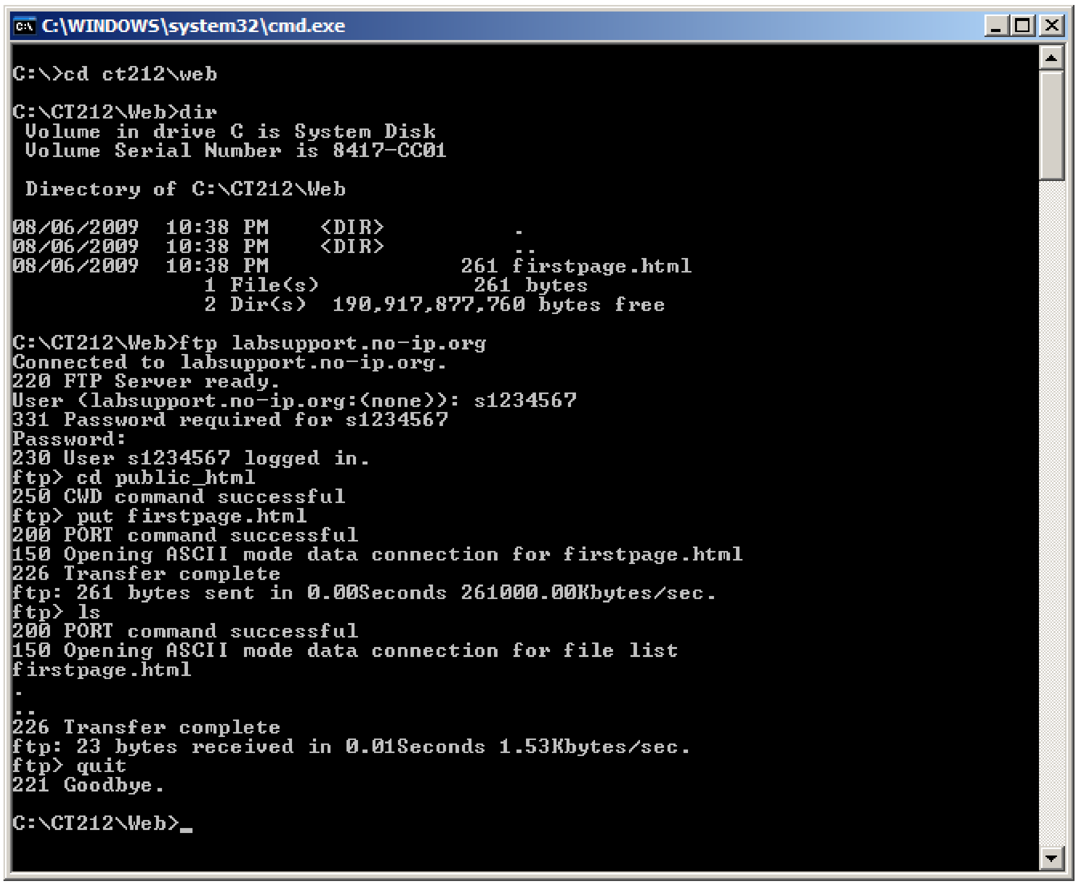
   c Supply your user name and password on the *ELEC S212* Server to log in.

d After you have logged in successfully, enter the FTP client command: `cd public_html` to change to your ‘`public_html`’ directory.

e Enter the command `put firstpage.html` to upload your webpage.

f Enter the command: `ls` to verify that the file has indeed been uploaded.

g Enter the command: `quit` to terminate your FTP session.



C:\WINDOWS\system32\cmd.exe

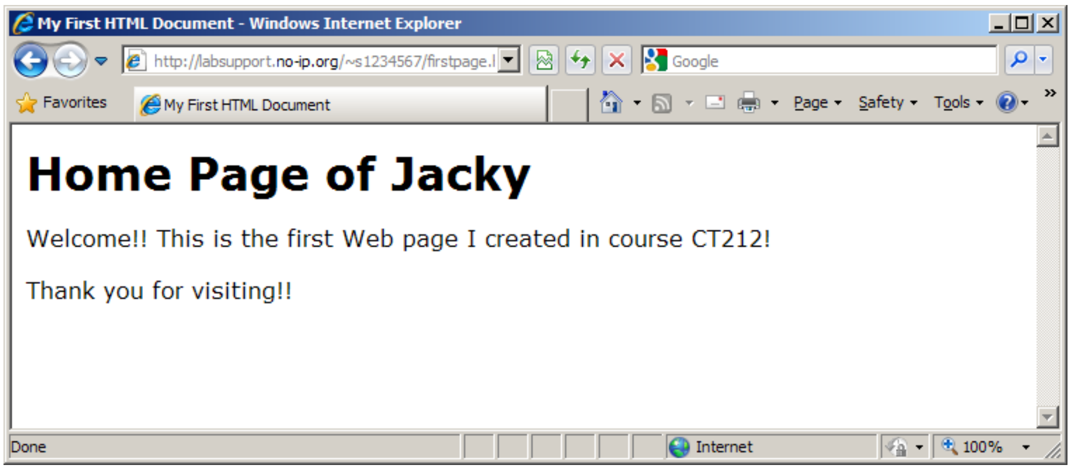
```
C:\>cd ct212\web

C:\CT212\Web>dir
 Volume in drive C is System Disk
 Volume Serial Number is 8417-CC01

 Directory of C:\CT212\Web

08/06/2009  10:38 PM    <DIR>          .
08/06/2009  10:38 PM    <DIR>          ..
08/06/2009  10:38 PM               261 firstpage.html
               1 File(s)            261 bytes
               2 Dir(s)  190,917,877,760 bytes free

C:\CT212\Web>ftp labsupport.no-ip.org
Connected to labsupport.no-ip.org.
220 FTP Server ready.
User (labsupport.no-ip.org:(none)): s1234567
331 Password required for s1234567
Password:
230 User s1234567 logged in.
ftp> cd public_html
250 CWD command successful
ftp> put firstpage.html
200 PORT command successful
150 Opening ASCII mode data connection for firstpage.html
226 Transfer complete
ftp: 261 bytes sent in 0.00Seconds 261000.00Kbytes/sec.
ftp> ls
200 PORT command successful
150 Opening ASCII mode data connection for file list
firstpage.html
.
..
226 Transfer complete
ftp: 23 bytes received in 0.01Seconds 1.53Kbytes/sec.
ftp> quit
221 Goodbye.

C:\CT212\Web>_
```

Figure 7.6 An FTP session to upload your webpage to the *ELEC S212* Server

4 On the *ELEC S212* Server, change the access rights of the `firstpage.html` file to 644. Without setting proper access rights, others cannot view your webpage on the Web.

5 On your Windows PC, launch your Web browser and visit this URL (substitute s1234567 with your own student ID):

`http://labsupport.no-ip.org/~s1234567/firstpage.html`


My First HTML Document - Windows Internet Explorer
http://labsupport.no-ip.org/~s1234567/firstpage.h
Google
Favorites
My First HTML Document
Page
Safety
Tools
Done
Internet
100%

Home Page of Jacky

Welcome!! This is the first Web page I created in course CT212!

Thank you for visiting!!

Figure 7.7 Your first webpage for *ELEC S212* published on the Web

The following reading is an introduction to HTML.

Reading 7.5 (online)

w3schools.com, 'HTML Introduction':

http://www.w3schools.com/html/html_intro.asp

Basis structure of a webpage

Step 2 in the example in Activity 7.1 shows the basic structure of an HTML document.

Tags

You should find many tags inside the document. These include `<html>`, `<head>`, `<title>`, `<body>`, `<H1>` and `<P>` and their corresponding end tags — that is, tags with a slash inside the brackets such as `</html>`. Though all of the tags shown are in pairs, we will show you some that do not need an end tag. HTML tags are not case sensitive, so `<HTML>`, `<hTmL>` and `<html>` are the same.

The tags `<html>`, `<head>`, `<body>` and their corresponding end tags define the basic structure of an HTML document. The entire content of an HTML document is enclosed by the `<html>` tag and its end tag `</html>`. HTML documents are divided into two logical parts, the head and the body, which are enclosed by `<head> </head>` and `<body> </body>` respectively. Generally, the head contains the general information of the document like the title, author, etc., and the body is the actual content.

What you can see from a Web browser is only the body part of the webpage. That means the browser will not show the information in the head part, except the information enclosed in the `<title> </title>` tags. And although the title is shown, it is not shown inside the browser window as the body, but on the title bar of the browser window. If no title is defined in the document, the URL for the document is shown.

If you need to record some notes to help yourself or other readers to understand the HTML document, you can put your comments inside the tags `<!-- -->` which will be ignored by the Web browser in interpreting the Web document.

You may create headlines with the headline tags `<H1> </H1>`. There are six levels of headline tags, from `<H1>` to `<H6>`. You can try all of them in the later activities.

`<P>` denotes the beginning of a new paragraph, and `</P>` indicates the paragraph end. The headlines and paragraph tags are used in the body part of the HTML document. Usually the extension `.html` or `.htm` is used as the file extension for HTML documents.

Free format

An HTML document can be composed in free format. That is, extra spaces and line breaks are ignored in the document. The following example shows the HTML document in Activity 7.2, with arbitrary spaces and line breaks added. The Web browser will interpret it the same way and display the same result as in Activity 7.2.

Example 7.1

```
<HTML>                                    <HEAD>
                                          <TITLE>My First
                                                 HTML
Document</TITLE></HEAD>
<BODY>
<!--This
 is a comment-->
<H1>Home Page      of #your name# </H1><P> Welcome!
! This is the
first webpage I        created in course CT212!
   </P><P>Thank you for         visiting!!</P>
                                          </BODY></HTML>
```

Formatting

In this section we look at how to format:

- paragraphs;
- pre-formatted text;
- headings and rulers; and
- lists.

Paragraphs

The browser also ignores any extra spaces and carriage returns for the text within `<p>` and `</p>` tags and then displays the text in a line with automatic wrapping. As mentioned in the above example, the `<p>` tag indicates to the browser that a new paragraph has started so that the new text can be separated from the previous text. Usually, a blank line is inserted. The `</p>` is optional because the `<p>` of the next paragraph automatically marks the end of the last paragraph.

If you want to insert a line break in a paragraph, you may use the tag `<br>`. The `<br>` tag does not require a corresponding end tag, and all existing Web browsers work fine without the corresponding end tag.

The following reading discusses HTML paragraphs.

Reading 7.6 (online)

w3schools.com, 'HTML Paragraphs':

http://www.w3schools.com/html/html_paragraphs.asp

Preformatted text

In contrast to the `<P>` tag, where extra spaces and line breaks are ignored, text enclosed by the `<PRE>` and its end tag `</PRE>` is displayed according to what is actually coded in the source. The following is an example.

```
<PRE>
There    are    spaces            between words.
This statement is separated

into two lines with a blank line in between.
</PRE>
```

The output in the browser will be:

```
There    are     spaces            between words.
This statement is separated

into two lines with a blank line in between.
```

Note, however, that the browser will still process any HTML tags inside the `<PRE>` and `</PRE>` tags.

Any text to be centered is enclosed between the tags `<CENTER>` and `</CENTER>`. The `<CENTER>` tag may be used to centre text, images, headings etc.

Headings and rulers

As mentioned, there are six levels of heading, from `<H1>` to `<H6>` with `<H1>` the largest. Headings are typically displayed in larger and bolder fonts than normal body text.

The tag `<HR>` will create a horizontal ruler across the page. There are four attributes for this tag: `SIZE`, `WIDTH`, `ALIGN` and `NOSHADE`. Table 7.1 gives a summary of these attributes.

Table 7.1 Attributes for the `<HR>` tag

Attributes	**Usages**	**Examples**
`SIZE`	Specifies the thickness of a rule.	`<HR SIZE="10">`

`WIDTH`	Specifies how far a rule stretches across the page; you can specify according to the number of pixels or as a percentage of the screen width.	`<HR WIDTH="20">`
		`<HR WIDTH="50%">`

`ALIGN`	Specifies the alignment of a rule: left, centre or right; this attribute should be used with WIDTH.	`<HR WIDTH=30% ALIGN="RIGHT>`
`NOSHADE`	Specifies whether a rule appears solid or engraved.	`<HR NOSHADE>`

You may use all of these attributes at once in the same `<HR>` tag.

The following reading is about HTML headings.

Reading 7.7 (online)

w3schools.com, ‘HTML Headings’:

http://www.w3schools.com/html/html_headings.asp

Lists

You can list items in a webpage with any of the three types of list: unnumbered, numbered and definition. The list can be nested to any level.

An unnumbered list is specified by the `<UL>` tag, and each list item is denoted by the `<LI>` tag. The following is an example:

```
<UL>
<LI> Unit1: Network Basics
<LI> Unit2: Internetwork Architecture
<LI> Unit3: Network Protocols for the Internet
</UL>
```

The output in the browser will be:

```
•  Unit1: Network Basics
•  Unit2: Internetwork Architecture
•  Unit3: Network Protocols for the Internet
```

A numbered list, or ordered list, is specified by the `<OL>` tag. Each list item is also denoted by the `<LI>` tag. The following shows an example:

```
<OL>
<LI> Unit1: Network Basics
<LI> Unit2: Internetwork Architecture
<LI> Unit3: Network Protocols for the Internet
</OL>
```

The output in the browser will be:

```
1  Unit1: Network Basics
2  Unit2: Internetwork Architecture
3  Unit3: Network Protocols for the Internet
```

There are also different attributes for the above tags. Table 7.2 gives a summary.

Table 7.2 Attributes for the `<UL>`, `<OL>` and `<LI>` tag

Tags	Attributes	Usages	Values
`<UL>`	`TYPE`	Specifies the appearance of the bullet.	`CIRCLE`, `SQUARE`, `DISC` in which `DISC` is the default value.
`<OL>`	`TYPE`	Specifies the type of numbering.	`a` (lowercase letters), `A` (uppercase letters), `i` (lowercase Roman numerals), `I` (uppercase Roman numerals), `1` (Arabic numerals) — default value.
	`START`	Specifies the starting number of the list.	Any integer.
`<LI>`	`TYPE`	Specifies type for each list item; same effect as the `TYPE` attribute for both `<UL>` and `<OL>` tags.	Same as the above `TYPE` attribute for both `<UL>` and `<OL>` tags.
	`START`	Specifies the starting number of a list item; same effect as the `START` attribute of the `<OL>` tag.	Same as the above `START` attribute for `<OL>` tag.
	`VALUE`	Specifies the number of a list item; only available in `<OL>` tag.	Any integer.

You can create a list that looks like the glossary by using the definition list. The `<DL> </DT>` tags are used to denote such a list. Between these two tags, the `<DT>` tag, defining a term and the `<DD>` tag, defining the content, appear alternately. Web browsers generally format the definition on a new line and indent it.

```
<DL>
<DT> HTML
<DD> HyperText Markup Language
<DT> HTTP
<DD> HyperText Transfer Protocol
<DT> WWW
<DD> World Wide Web
</DL>
```

The output in the browser will be:

```
HTML
HyperText Markup Language
HTTP
HyperText Transfer Protocol
WWW
World Wide Web
```

The following reading discusses HTML lists.

Reading 7.8 (online)

w3schools.com, ‘HTML Lists’:

http://www.w3schools.com/html/html_lists.asp

Character formatting

You can format characters in two ways: using either the physical style or the logical style. The tags used in these two styles are summarized in Tables 7.3 and 7.4.

Table 7.3 Tags for character formatting in physical style

Tags	**Usages**
`<B>`	Boldface
`<I>`	Italic
`<TT>`	Typewriter text — fixed width font
`<FONT>`	Specifies the font size with the SIZE attribute; specify the font color with the COLOR attribute

The physical style

In the physical style, you can directly specify the typefaces of the characters such as italic and boldface. For the `SIZE` attribute of `FONT`, you can specify an absolute font size by writing down the actual font size value, for example, `<FONT SIZE="5">`. Or you can specify a font size relative to the base font size, which is set to 3, by using the plus or minus sign, for example, `<FONT SIZE="+2">`. Font size ranges from 1 to 7.

For the `COLOR` attribute of `FONT`, you can specify text color by name, e.g. `<FONT COLOR="YELLOW">` or by the RGB value, e.g. `<FONT COLOR="#4070E1">`. (RGB stands for red, green and blue.) Using the `#` sign, each color is specified by a two-digit hexadecimal number with each digit ranging from 0 to F. For example, to specify red, the hexadecimal number FF0000 is used. In this number: The first two digits FF define 100% of red, the second two digits 00 define 0% of green and the third two digits 00 define 0% of blue.

There is additional tag `<BODY>` that allows a user to specify global coloring of the webpage. Attributes of this tag include:

`BGCOLOR` — Specifies the background color of the document.

`TEXT` — Specifies the color of the text in the document.

`LINK` — Specifies the color of the hypertext links in the document.

`ALINK` — Specifies the color of the link when activated by your mouse.

`VLINK` — Specifies the color of the visited hypertext links in the document.

For example, to create a webpage with a background color of green normal text in red and links in blue is specified with the following attributes to the tag `<BODY>: <BODY BGCOLOR="#00FF00" TEXT="red" LINK="#0000FF">`. You can also provide a background image for a document. The selected image is tiled across the document, and then the text of the document is written over the image(s). The background image is achieved by adding a background attribute to the `BODY` markup tag. For example: `<BODY BACKGROUND="backgrd.jpg">`.

The logical style

In the logical style, you specify the character formats according to their functions, such as the text for emphasis, strong emphasis, etc. The Web browser will format these logical styles according to its own settings. The typical typeface for each style is shown in Table 7.4. These tags should be associated with their end tags in order to specify the range of text they have effects on.

Table 7.4 Tags for character formatting in logical style

Tags	**Usages**
`<CITE>`	For titles of books, films, etc.; typically displayed in italics.
`<CODE>`	For computer code; displayed in a fixed-width font.
`<SAMP>`	For a sequence of literal characters; displayed in a fixed-width font.
`<BLOCKQUOTE>`	For making notes; typically indented.
`<DFN>`	For the definition of a word; typically displayed in italics.
`<EM>`	For emphasis; typically displayed in italics.
`<KBD>`	For user keyword entry; typically displayed in plain fixed-width font.
`<STRONG>`	For strong emphasis; typically displayed in bold.
`<VAR>`	For a variable in which you will replace the variable with specific information; typically displayed in italics.

The following reading discusses HTML formatting.

Reading 7.9 (online)

w3schools.com, ‘HTML Formatting’:

http://www.w3schools.com/html/html_formatting.asp

Escape sequences

Some special and reserved characters of HTML, such as `<` and `>`, cannot directly be typed in the code; otherwise, the browser may misinterpret it. You then have to use the escape sequence to indicate it. Some commonly used escape sequences are shown in Table 7.5. You may refer to the section ‘Using HTML special characters’ in the Smith reading for a more detailed description.

Table 7.5 HTML escape sequences

Escape sequences	**Usages**
`&lt;`	The escape sequence for <
`&gt;`	The escape sequence for >
`&quot;`	The escape sequence for "
`&amp;`	The escape sequence for &
`&nbsp;`	The escape sequence for a space; this forces the browser to display a space character.

Linking

The power of HTML comes from its hyperlinks, as you’ve seen how links can allow relevant information to be tied together and retrieved easily.

Creating hyperlinks

To add a hyperlink to a text or an image, you can use the `<A>` tag. See the following example.

Example 7.2

```
<A HREF="NextPage.htm">Link to Next Page</A>

<A HREF="http://www.ouhk.edu.hk/default.htm">
Microsoft Home</A>
```

The HREF attribute of the `<A>` tag specifies the location that the hyperlink should jump to. The text enclosed by the `<A>` and `</A>` tags forms the hyperlink. When a user clicks on the text, the Web browser will download the specified HTML document and display it. In the above example, the first and second statements will display the file `NextPage.htm` and `default.htm` on the website `www.ouhk.edu.hk` respectively, upon clicking the mouse.

You can specify a location in two formats: relative path and absolute path. In the above example, relative path is used in the first statement, and absolute path in the second statement.

For a relative path, the browser will get the file `NextPage.htm` from the same directory as the current document. If the file `NextPage.htm` is under the directory `dir1`, which is a subdirectory of the current directory, the location should be specified as `dir1/next.htm` and the statement is:

```
<A HREF="dir1/NextPage.htm"> Link to Next Page</A>
```

You can also use the `..` to specify the parent directory of the current document, for example, if the file `NextPage.htm` is put in the parent directory of the current document. The following statement should be used:

```
<A HREF="../NextPage.htm"> Link to Next Page</A>
```

For the absolute path, a URL is used to specify the location of files on the same or other Web servers. The following is another example:

```
<A HREF="http://www.microsoft.com">MicrosoftHome</A>
```

You can also navigate a browser to a new point in the same document by using the `NAME` attribute of the `<A>` tag.

Hyperlinks for sending email

If you want to create a hyperlink that will call up the mail program for sending email when you click the mouse, you can use the following statement:

```
<A HREF="mailto:webmaster@plbpc011.ouhk.edu.hk">Mail
To Webmaster</A>
```

You may find that the `<A>` tag and its `HREF` attribute are also used here. But the value for the `HREF` attribute is now an email address preceded by the keyword: `mailto` and a colon.

Automatic redirect

When surfing the Web, you may have found that on downloading a webpage, you are automatically redirected to another location. The most common examples are situations in which a site has moved to a new Web server. The Web master usually redirects all the viewers from the old server to the new one. In such cases you can use the following statement:

```
<META HTTP-EQUIV="Refresh" CONTENT="0;
URL="www.ouhk.edu.hk"> Webmaster</A>
```

This statement should be put inside the head part of the HTML document. The `<META>` tag is used by the document to describe information about itself. The `CONTENT` attribute specifies the waiting time in seconds before retrieving the document specified by the `URL` attribute. Both relative and absolute paths can be used to specify the document location for the `URL` attribute. In the above example, 0 seconds is specified for the `CONTENT` attribute, so anyone who opens this HTML document will be immediately redirected to the homepage of the Open University, which is `www.ouhk.edu.hk`.

The following reading discusses HTML links.

Reading 7.10 (online)

w3schools.com, 'HTML Links':

http://www.w3schools.com/html/html_links.asp

Handling images

You can insert images into a webpage by using the `<IMG>` tag. There is no end tag required for `<IMG>`. Some attributes for `<IMG>` are shown in Table 7.6.

Table 7.6 Attributes for the <IMG> tag

Attributes	Usages	Examples

`SRC`	Specifies the image file to be displayed; similar to the `HREF` attribute of the `<A>` tag, both relative and absolute paths may be used here.	`<IMG SRC="../image.gif">`
		`<IMG SRC="http://www.microsoft.com/image.gif">`

`ALT`	Specifies text to be displayed when the image cannot be displayed.	`<IMG SRC="image.gif" ALT="first image">`
`ALIGN`	Specifies the alignment of the image; values such as `TOP`, `BOTTOM`, `MIDDLE`, `LEFT` and `RIGHT` can be used.	`<IMG SRC="image.gif" ALIGN="LEFT">`
`HEIGHT`	Specifies the height of the image in pixels.	`<IMG SRC="image.gif" HEIGHT="100" WIDTH="150">`
`WIDTH`	Specifies the width of the image in pixels.	`<IMG SRC="image.gif" HEIGHT="100" WIDTH="150">`
`BORDER`	Defines the width of a border around the image.	`<IMG SRC="image.gif" BORDER="5">`

The following reading discusses HTML images.

Reading 7.11 (online)

w3schools.com, ‘HTML Images’:

http://www.w3schools.com/html/html_images.asp

Creating tables

A table is created using the `<TABLE>` markup tag. The simplest table consists of a single data cell. A table is generated using a group of table tags, as listed in Table 7.7.

Table 7.7 Tags that are used by tables in HTML

Escape sequences	**Usages**
`<TABLE></TABLE>`	Beginning and end of the table.
`<TR></TR>`	Beginning and end of one row of a table.
`<TD></TD>`	Beginning and end of one cell in one row of a table.
`<CAPTION></CAPTION>`	Title of the table written above the table (optional).
`<TH></TH>`	Volume heading over one column in a table (optional).

The following reading discusses HTML tables.

Reading 7.12 (online)

w3schools.com, 'HTML Tables':

http://www.w3schools.com/html/html_tables.asp

Forms

You have likely encountered many HTML forms when surfing the Web, such as sign-ups, subscriptions, and surveys and evaluations. The `<FORM>` tag and its end tag `</FORM>` are used to define a form.

A form is used to get user input and submit it to the server. A program in the Web server is responsible for processing the received data, for example storing the information in the database. The elements that can be defined in a form for capturing data are shown in Table 7.8. The `<INPUT>` and the `<SELECT>` tags are used to define various elements.

Table 7.8 HTML form elements

Escape sequences	**Usages**
One line text box	`<INPUT TYPE="text" Name="name" SIZE="20" VALUE="Your name">`

Multiple lines of text box	`<TEXTAREA NAME="feedback" ROWS=5 COLS="20">`
	`Here is my comment:`
	`</TEXTAREA>`

Radio button	`<INPUT TYPE="radio" NAME="sex" VALUE="M">Male<BR>`
	`<INPUT TYPE="radio" NAME="sex" VALUE="F">Female`

Check box	`<INPUT TYPE="checkbox" NAME="ISO>ISO 9002 Certified`

Pick list	`<SELECT NAME="Media">`
	`<OPTION SELECTED>Hard disk`
	`<OPTION>Floppy disk`
	`</SELECT>`

Hidden field	`<INPUT TYPE="hidden" NAME="id" VALUE="123456">`
Submit button	`<INPUT TYPE="submit" VALUE="Send">`
Reset button	`<INPUT TYPE="reset" VALUE="Reset">`

The following reading discusses HTML forms.

Reading 7.13 (online)

w3schools.com, ‘HTML Forms’:

http://www.w3schools.com/html/html_forms.asp

The following website provides a very comprehensive HTML and CSS online reference with lots of examples. You are highly recommended to use it as a reference when authoring your webpages.

Reading 7.14 (online)(optional)

HTML Ref:

http://www.htmlref.com

Programming for the Web

In the previous sections you learned how to build a website using HTML. The result produces a straightforward website whereby each page is a simple rendering of hypertext. Each time users access the site, they see the same pages with no changes, unless the site developer edits the original pages.

These days, this type of static webpage cannot satisfy the commercial environment on the Internet. Doing so requires a website responsive to customers' online requests, or one that processes orders in real time. This requires the use of programs called scripts or macros (a macro is a list of instructions that performs a task and can make decisions based on input and produce output).

Beginning with this section, you will learn how to create webpages in which content and layout can be modified using built-in programs. This section describes two kinds of Web programs: server-side scripts and client-side scripts. It then introduces the protocols and languages used on the server side. Finally, it introduces the use of scripting on both the client-side and the server-side.

Server-side scripting and client-side scripting

In the previous section, you learned how a webpage displays information to the user. Once you enter a site's URL, the Web browser requests the page, and the Web server sends it back to the browser. There are two kinds of programs executed on the Internet: server-side scripts and client-side scripts.

A **server-side script** or server-side program is a program executed on the server. For example, imagine you want to buy a book from amazon.com. Once you enter the site's URL, the Web browser requests the page, and the Web server sends it back to the browser. You can enter the subject and click **Go**! When the server receives your input, it passes the information to a program that is part of the website. The program is executed on the server. It is used to search a database for the book and inserts that list into the webpage, which is sent to your browser. The following figure illustrates the process.

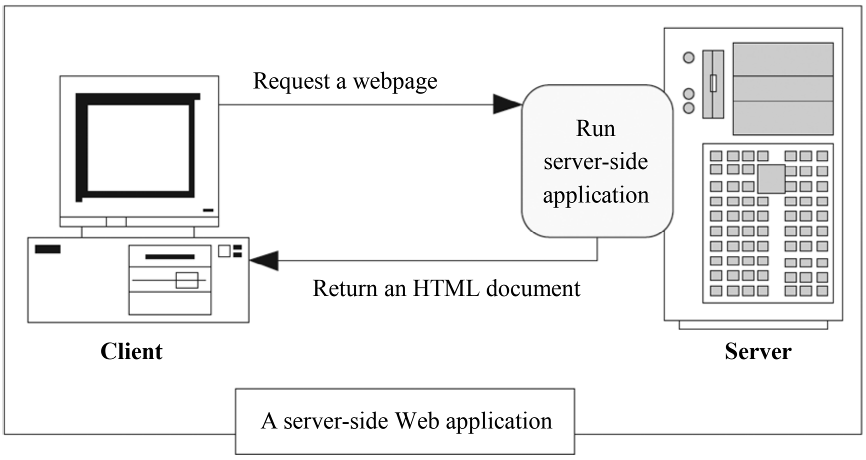


Figure 7.8 A server-side script (MacDonald 2007, Figure 1-3)

The following figure shows the alternative program, the **client-side script**. The script is included in the webpage sent to the Web browser, and is executed by the browser on the PC. The script customizes the HTML for the browser before the browser displays the page. A script executed on the PC is called a client-side program or client-side script.

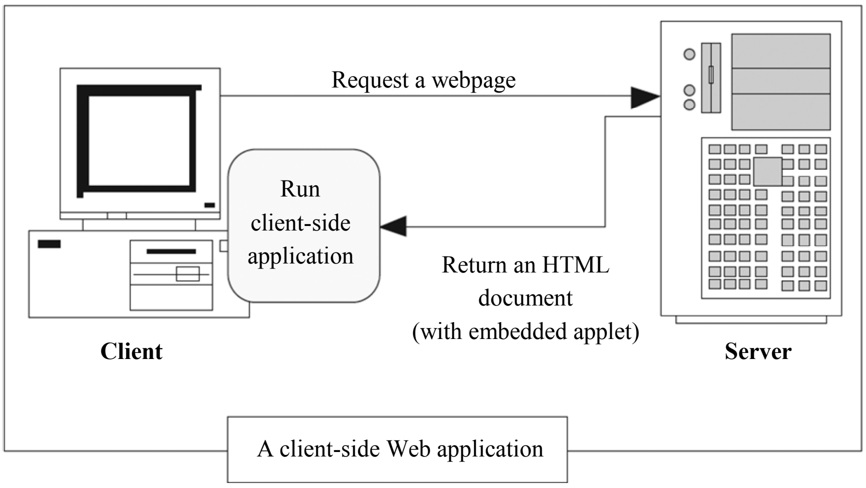

Request a webpage
Run client-side application
Return an HTML document (with embedded applet)
Client
Server
A client-side Web application

Figure 7.9 A client-side script (MacDonald 2007, Figure 1-3)

Client-side programs solve many problems. Computing is distributed over the Web so that no one server is overloaded with handling programming requests. Webpages containing client-side programs can be tested locally without first uploading them to the Web server. A client-side program is likely to be more responsive to the user, because the user does not have to wait for data to be sent over the Internet to the Web server.

Client-side programs can never completely replace server-side programs, however. For example, if you need to run a search form or process a purchase order, it must be run from a central server, because the server will most likely contain the database needed to complete those operations.

Self-test 7.3

1 What is the difference between a dynamic webpage and a static webpage?

2 What software executes a client-side script?

3 What are the differences between client-side and server-side scripting?

Both server-side scripts and client-side scripts can be contained in a webpage. The server-side script is executed on the server, and the client-side script is executed by the browser. When a webpage reaches the client, there is no server-side script in the page, because the server-side script has already been executed and has substituted the resulting HTML tags and text in the page in place of the script. Client-side script is generally executed before the browser displays the final HTML page.

A survey of programming languages used for Web programming

A programming language is a set of commands and arguments that have a predetermined meaning to the software that is executing the program. Because programming on the Internet has become popular, there has already been some evolution of languages. This section introduces the protocols and languages used on the server-side.

Server-side programming protocols

As you learned in previous section, server-side programs are interpreted and executed by the Web server. There are a variety of types of Web server available in the market, e.g., Apache,[9] Internet Information Services,[10] nginx,[11] etc. Programmers may know in advance which kind of Web server will be interpreting and executing their programs. Therefore, standards are needed that specify how a server recognizes a program, and how the server and the program interact. The standards are referred to as programming protocols. Here we introduce some popular programming protocols:

- Common Gateway Interface (CGI),
- Internet Server Application Programming Interface (ISAPI),
- Active Server Page (ASP), and
- Server-Side Includes (SSI).

[9] http://httpd.apache.org

[10] http://www.iis.net

[11] http://nginx.net

Common Gateway Interface (CGI)

Common Gateway Interface (CGI) is the original — and still very popular — way for a program or script to interact with a Web server. It was initially developed for use on UNIX computers. Today, all of the popular Web servers on both Windows and UNIX support GCI. The following figure demonstrates how the CGI protocol works.

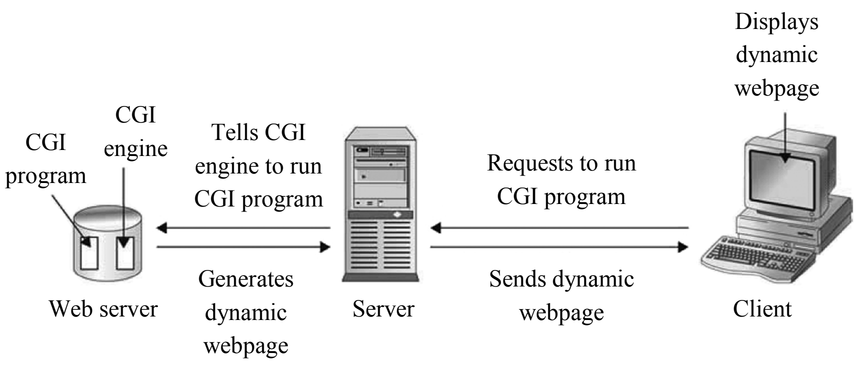


Figure 7.10 How a CGI program works (Keogh 2005, Figure 1-2)

The basic concept of CGI is that the users submit information, generally via a form, from their browser to the server. The server then calls the CGI application. The CGI application then sends a response to the server, and the server sends the CGI response back to the user's browser as an HTML page.

Internet Server Application Programming Interface (ISAPI)

Internet Server Application Programming Interface (ISAPI) was originally developed by Microsoft for use on Windows NT servers that were using Microsoft's Web server software, Internet Information Services (IIS). The ISAPI protocol makes use of Microsoft's dynamic linked library concept. A dynamic linked library (DLL) is an object or a program that is executed by another program to perform a specific task.

As many have realized, CGI isn't very efficient. ISAPI was developed to fulfill the deficiencies of CGI. One of the main reasons that ISAPI is more efficient than CGI is that ISAPI remains memory-resident. CGI, on the other hand, unloads after it's called. Each time the CGI executable is called, the server must load the CGI process back into memory, whereas ISAPI will already be in memory and can therefore serve the request more quickly. For large-scale sites such as EBay, ISAPI has been a savior. If EBay were to run on CGI, the burden on their servers would be nearly doubled, resulting in crashing and poor performance, and thus resulting in lost business. Another great benefit of ISAPI is the flexibility of the process. Like CGI executables, the user can use the compiler's embedded functions and the Windows API to access the operating environment of the server.

However, ISAPI is more difficult for developers to use than CGI, because the program they build must relate to the Web server as a DLL rather than as a stand-alone program. A DLL is more restricted by design and is written specifically for the Web server that uses it.

Active Server Pages (ASP) and ASP.NET

Active Server Pages (ASP) technology was developed by Microsoft to handle server-side programming more easily and powerfully. It allows you to write a webpage with one or more scripts on the page. When the server receives a request for an ASP file, it processes server-side scripts contained in the file to build the webpage that is sent to the browser. In addition to server-side scripts, ASP files can contain HTML (including related client-side scripts) as well as calls to COM components that perform a variety of tasks, such as connecting to a database or processing business logic. COM (Microsoft's Component Object Model) is a Microsoft standard that defines how software components from different vendors can work together.

The way ASP works is similar to the CGI technology that we discussed previously. The browser requests the webpage containing the ASP script. The Web server recognizes the webpage contains an ASP script, because the file extension of the webpage is .asp. The Web server executes the scripts that have access to the system resources on the website, such as a database. After the script is completed, the webpage is sent to the browser.

In fact, ASP has been superseded by Microsoft's new Web application framework, **ASP.NET**. ASP.NET was first released in 2002 as part of the Microsoft .NET Framework. ASP.NET is built on the **Common Language Runtime** (**CLR**), allowing programmers to write ASP.NET code using any supported .NET language, including C#, Visual Basic .NET, JavaScript, COBOL, PERL, Python, Ruby, Eiffel and many others. In contrast, classic ASP supports only a few scripting languages such as VBScript and JScript (Microsoft's JavaScript variant).

Server-Side Includes (SSI)

CGI is useful for processing form data and generating dynamic output, but it does require some programming knowledge. Generating even a small HTML document with a CGI program requires writing a complete program. Not only is this a bit more complex than writing a standard HTML document, but it also requires significant resources to start a CGI program each time the document is requested. Many documents contain a large amount of static content and a relatively small amount of dynamic content. Therefore, rather than using a heavyweight solution like CGI, you might want to consider using server-parsed HTML — otherwise known as **server-side includes** (**SSIs**).

SSIs allow you to place special server-side tags into an HTML document that are processed by the server. SSI tags can execute other programs or display data in your HTML files. When the document is requested, the server parses the file and processes these tags. Dynamic data is inserted into the HTML document in place of the tags.

Programming languages for Web programming

There are a variety of programming languages that can use either the CGI or ISAPI specifications. Classic ASP can only be used by two scripting languages: VBScript and JavaScript. SSI is limited to a few HTML tags.

There are several other popular languages used for Web programming.

- **Perl**

  Perl has been described as a compiled scripting language. It combines elements of C with some UNIX scripting and text manipulation languages into a more complete language. Perl's advocates claim it is easy to use, especially compared to Java. One of the motivations in developing Perl was to provide an alternative to C for tasks that were a little too difficult for an existing UNIX tool, or where performance was a problem. C was seen as unnecessarily low level, and C development too time-consuming. Perl tries to delay the need to program at a lower level. It's been very successful, especially among system administrators.

- **C/C++**

  You learned about the C programming language in *Unit 5*, so details are not repeated here. C++ is an updated evolution of C that uses an object-oriented approach to programming. An object is anything that is addressed by the program as an entity with properties, attributes and rules that the program must follow in order to use the object.

- **Java**

  Java is the leading contender for a full-feature programming language targeted at Internet applications. It advantages are: *familiarity* (derived from C++); *platform independence* (it will run on any platform which implements the Java Virtual Machine); and *performance* (byte-code compiled faster than fully interpreted).

  Java is aggressively distributed and promoted by Sun Microsystems, which developed it, and, evidently, sees it as a way to loosen Microsoft's and Intel's grip on the computer platform. All major Web browsers now support the Java VM, and Java applets can be found on many websites.[12] The ability to deliver a platform independent application, or, more correctly, an OS-independent application, is greatly appealing to developers, who spend a large portion of their resources developing and maintaining versions of their products for the different hardware/software platform combinations. With Java, one set of sources, and, even more important, one byte compiled executable, can be delivered for all

[12] However, the use of Java applets has been diminishing in recent years, partly due to the extremely popular Adobe Flash. Rich Internet applications (RIAs) such as Adobe Flash and Microsoft Silverlight have taken away almost all market shares of Java applets. In fact, nowadays it is definitely easier to spot an Adobe Flash application than a Java applet running on a website.

hardware/software platforms. Although interpretation of byte-compiled program is slower than execution of a native executable, the claim is made that, once interpreted, the resulting executable is of comparable performance, which means Java apps could be interpreted once and the result cached locally, and thereafter executed optimally.

However, the most common complaint is that Java is not simple; it's basically a slimmed down, cleaned up C++, with a big GUI library. C++ programming is not described by most as 'simple,' and Java programming is not much simpler, especially compared to HTML, or some other languages put forward as its competition. Java is the market leader at the moment, so it is the obvious target.

- **Visual Basic Script (VBScript)**

  VBScript is a subset of Visual Basic, Microsoft's popular visual programming language, with no GUI building capability and with access to other applications via OLE. VBScript source code is embedded in HTML, and downloaded to the client in the HTML file, where it is compiled and executed in association with its runtime libraries. VBScript has the advantage over other programming languages because there are many Visual Basic programmers who already know Visual Basic, so learning VBScript is almost effortless.

- **JavaScript**

  JavaScript is a scripting language used to enable programmatic access to objects within other applications. It is primarily used in the form of client-side JavaScript for the development of dynamic websites.

  JavaScript is a dialect of the ECMAScript[13] standard and is characterized as a dynamic, weakly typed, prototype-based language with first-class functions. JavaScript was influenced by many languages and was designed to look like Java, but it was intended to be easier for non-programmers to work with. JavaScript differs from Java in that it is interpreted, rather than compiled, only supports certain built-in objects and user-defined functions, rather than full support for user-defined classes with inheritance and methods,

  JavaScript is integrated with HTML, rather than invoked from HTML files. It is meant to extend HTML to be more of a full programming language but retaining HTML's ease of use. The principal criticism of Java programming is that it is much more complex than HTML programming — i.e. it's more like C++ programming — and therefore it's not as accessible to users as HTML. This is a problem that JavaScript attempts to address.

[13] ECMAScript is a scripting language, standardized by ECMA International in the ECMA-262 specification and ISO/IEC 16262. The language is widely used on the Web, especially in the form of its three best-known dialects, JavaScript, ActionScript and JScript.

- **C#**

  C# (pronounced 'C Sharp') is a multi-paradigm programming language encompassing imperative, functional, generic, object-oriented (class-based), and component-oriented programming disciplines. It was developed by Microsoft within the .NET initiative and later approved as a standard by ECMA (ECMA-334) and ISO (ISO/IEC 23270).

  C# is one of the programming languages designed for the Common Language Infrastructure. It has an object-oriented syntax based on C++ and greatly resembles that of Java. C# is one of the two primarily used languages in ASP.NET (The other one is Visual Basic .NET).

- **PHP**

  PHP is an extremely popular general-purpose scripting language especially suited for producing dynamic webpages. It generally runs on a Web server, which is configured to take PHP code as input and create webpage content as output. It can be deployed on most Web servers and on almost every operating system and platform free of charge.

Self-test 7.4

1 On what operating system platform was the CGI model first developed?

2 What is the difference between a DLL and a regular program in relation to how the program or DLL is stored in memory?

3 What language was first used to write CGI scripts?

4 Small programs that are downloaded to a browser for handling special multimedia effects are normally written using what language?

JavaScript programming

In order to enhance interaction with readers after downloading an HTML document to the client machine, client-side scripting is introduced into HTML documents to handle events triggered by the readers. For instance, when you click on a submit button in an HTML form, with client-side scripting, the client machine can validate the data you entered and immediately prompt you for any errors. Compared with pure HTML code, an HTML form is always submitted to the server for processing, and the server is responsible for validating the data input and prompting for any incorrect data. Client-side scripting empowers Web browsing with the following features:

1 Interactive applications can be built within an HTML document.

2 Since the processing is done on the client machine, no data are sent over the network to the Web server, so responses can be much faster. This may also reduce the load on stressed Web servers.

JavaScript is the most popular client-side scripting language on the Web. It was developed by Netscape and was initially used only in Netscape browsers. Over the years, however, JavaScript has moved towards formal acceptance as an international standard, and now all major Web browsers support it.

A simple example of JavaScript

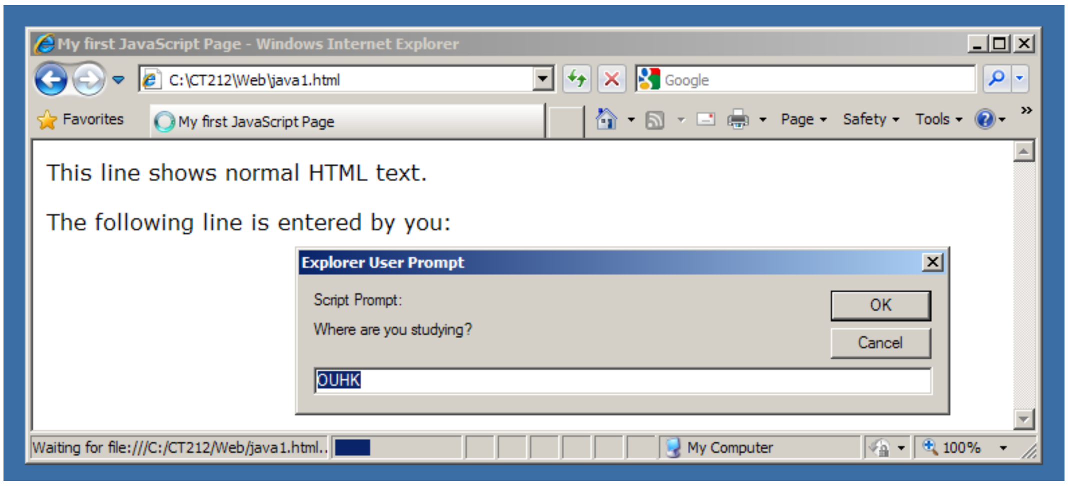
Again, the best way to begin with a new topic is with a simple example. The following shows a sample HTML page with JavaScript embedded. This is presented in the following activity.

Activity 7.3

1 Type in the following HTML document. Save it with the name ‘`java1.htm`’:

```
<HTML>
<HEAD>
<title>My first JavaScript Page</title>
</HEAD>
<BODY>
<P>This line shows normal HTML text.</P>
<SCRIPT LANGUAGE="JavaScript">
document.write("The following line is entered by you:");
document.write(prompt("Where are you studying?", "OUHK"));
</SCRIPT>
<P>This line shows normal HTML text again.</P>
</BODY>
</HTML>
```

2 Open the document with your Web browser.



This line shows normal HTML text.

The following line is entered by you:

Script Prompt:

Where are you studying?

OUHK

Figure 7.11 Your first JavaScript webpage

3 You will be prompted with a dialog box. Just press the **OK** button. What do you get?

4 Reload the document. This time, type 'Open University of Hong Kong' in the dialog box. What do you get?

If you just press the **OK** button without entering anything into the dialog box, you will probably get the following:

```
This line shows normal HTML text.
The following line is entered by you: OUHK
This line shows normal HTML text again.
```

If you enter the text as indicated in step 4 of the above activity, you will get:

```
This line shows normal HTML text.
The following line is entered by you: Open University of Hong Kong
This line shows normal HTML text again.
```

You see that you can enter data into the webpages interactively. You may also find the above example is like a simple HTML document, with the following additional statements:

```
<SCRIPT LANGUAGE="JavaScript">
document.write("The following line is entered by you:");
document.write(prompt("Where are you studying?", "OUHK"));
</SCRIPT>
```

To insert scripting code into an HTML document, a new tag `<SCRIPT>` is used. You have to specify the language type in the `<SCRIPT>` tag. In this example, it's JavaScript. Everything between the tag `<SCRIPT>` and the end tag `</SCRIPT>` is JavaScript code. The `document.write()` is one of the most important functions in JavaScript programming. `document.write()` is used to write something in the actual document, and it takes a string as its argument. So, the text 'The following line is entered by you:' can appear on the browser. `window.prompt()` is another function that calls up a dialog box to get user input.

The following reading introduces JavaScript.

Reading 7.15 (online)

w3schools.com, 'JavaScript':

http://www.w3schools.com/js/default.asp

http://www.w3schools.com/js/js_intro.asp

http://www.w3schools.com/js/js_whereto.asp

w3schools.com, 'JavaScript Tutorial':

http://www.w3schools.com/js/

Non-JavaScript browsers

Nowadays, all major Web browsers support JavaScript. Occasionally, however, some users might turn off the JavaScript support feature on their browsers (either by mistake or intentionally, for whatever reason). It is also possible that a user is using an odd browser that really does not support JavaScript. In such cases, these non-JavaScript browsers may display the HTML document in Activity 7.3 as:

```
This line shows normal HTML text.
document.write("The following line is entered by you:");
document.write(prompt("Where are you studying?","OUHK"));
This line shows normal HTML text again.
```

You may wonder why the `<SCRIPT>` and `</SCRIPT>` tags are not displayed. The reason is that the browser ignores any tags that it cannot interpret and follows up to format the remaining elements. Without interpreting the `<SCRIPT>` tag, the statements starting with `document.write` become normal text in the view of non-JavaScript browsers. In order to maintain the correct rendering of the HTML documents in non-JavaScript browsers, the document is changed as follows:

Example 7.3

```
<HTML>
<HEAD>
<TITLE>My first JavaScript Page</TITLE>
```

```
</HEAD>
<BODY>
<P>This line shows normal HTML text.</P>
<SCRIPT LANGUAGE="JavaScript">
<!--
document.write("The following line is entered by you:")
document.write(prompt("Where are you studying?","OUHK"))
//-->
</SCRIPT>
<P>This line shows normal HTML text again.</P>
</BODY>
</HTML>
```

A pair of HTML comment tags — `<!-- and -->` — is used to encapsulate the scripting code. Non-JavaScript browsers will not display them. The '`//`' is the comment tag of the JavaScript code. You should note that the browsers enabled with JavaScript already know this trick, so they can correctly interpret the document in the above example.

Handling events

As mentioned, JavaScript makes the webpages interactive. You may wonder how this can be achieved. The most important channel is through events.

But what are 'events'? Generally, you may consider events to be the results of user actions. For example, if a user clicks a button, a 'click' event occurs. If a user moves the mouse pointer across a hyperlink, a 'mouse-over' event occurs. It seems that every user action on the webpage can cause an event. This isn't wrong. However, you may only want to react to the events you are interested in, so you have to define event handlers to do this. Also, JavaScript only allows you to define event handlers on some of the webpage elements such as form buttons, hyperlinks, etc. An event handler is usually defined through the attributes of the HTML tags. The following is an example of an event handler of the form button.

Example 7.4

A form button is defined in the following code. Apart from the standard HTML code, there is a new attribute `onclick` for the `<INPUT>` tag, which defines the event handler. The event handler `onclick` will be triggered whenever a user clicks on the **Click Here** button. `alert` is a function that calls up an alert window showing the message contained in its argument.

```
<FORM>
<INPUT TYPE="button" VALUE="Click Here"
onclick="alert('You have clicked on the button!')">
</FORM>
```

JavaScript provides many different event handlers for you to write your interactive webpages. Some of them are shown in Table 7.9. For the full list of events that JavaScript supports, you may refer to the online reading on the following page.

Table 7.9 JavaScript event handler examples

Events	**Applies to**	**Occurs when**	**Event handlers**
Abort	Images	User aborts the loading of an image (e.g., by clicking a link or clicking the Stop button).	`onabort`
Change	Text fields, text areas, select lists	User changes value of element.	`onchange`
Click	Buttons, radio buttons, checkboxes, submit buttons, reset buttons, links	User clicks form element or link.	`onclick`
Key Press	Documents, images, links, text areas	User presses or holds down a key.	`onkeypress`
Load	Document body	User loads the page in the browser.	`onload`
Mouse Down	Documents, buttons, links	User depresses a mouse button.	`onmousedown`
Mouse Over	Links	User moves cursor over a link.	`onmouseover`
Select	Text fields, text areas	User selects form element's input field.	`onselect`

The following reading discusses JavaScript events.

Reading 7.16 (online)

w3schools.com, 'JavaScript events':

http://www.w3schools.com/js/js_events.asp

Functions

Very often, a function is called to give responses in handling an event. There are many built-in functions in JavaScript that you may use, like `document.write`, `prompt` and `alert`, introduced above. We do not describe these functions one by one; instead, you'll learn some of them through examples. These functions are associated with elements on the webpage, discussed in detail in the next section.

As with most programming language, JavaScript enables you to write your own functions for repeated tasks. Let's look at an example.

Example 7.5

```
<HTML>
<HEAD>
<TITLE>Writing Functions in JavaScript</TITLE>
</HEAD>
<BODY>
<SCRIPT language="JavaScript">
<!-- hide the script from old browsers
function myFunction(){
  document.write("Hello! I am using JavaScript!<br />")
}
myFunction()
myFunction()
myFunction()
// close the hiding-->
</SCRIPT>
</BODY>
</HTML>
```

The output of the above codes will be:

```
Hello! I am using JavaScript!
Hello! I am using JavaScript!
Hello! I am using JavaScript!
```

A function can definitely help you write clear codes. You may find that the definition of a function in JavaScript is very similar to that in the C language, except that the semicolons between statements are optional. You can also use variables, arithmetic operators like `+`, `-`, `*`, `/` and control flow statements like `if..else` and for in JavaScript.

In JavaScript, it is not necessary to declare what kind of value the variable will hold. The C language requires you to do this like:

```
int i,j;
char k = "a";
```

But now the value that a variable holds determines its data type, and later on the variable may also be assigned with values in different type. So you can create variables wherever you need them, without declaring them in the very beginning. To define the scope of a variable, so that local variable will not interfere with global variables if there is a duplicate variable name, the keyword `var` is typically used for the first appearance of a variable such as:

```
var i = 10
var k = "a"
```

In addition to variable declarations, you can consider most of the syntax for the operators and control flow statements to be the same as that in C, so we do not repeat the descriptions here. To further explore these areas, work through the following online readings.

The following reading discusses various sections of JavaScript.

Reading 7.17 (online)

w3schools.com, 'JavaScript':

Statements:	http://www.w3schools.com/js/js_statements.asp
Variables:	http://www.w3schools.com/js/js_variables.asp
Operators:	http://www.w3schools.com/js/js_operators.asp
Comparisons:	http://www.w3schools.com/js/js_comparisons.asp
If...Else:	http://www.w3schools.com/js/js_if_else.asp
Switch:	http://www.w3schools.com/js/js_switch.asp
Functions:	http://www.w3schools.com/js/js_functions.asp
For Loop:	http://www.w3schools.com/js/js_loop_for.asp
While Loop:	http://www.w3schools.com/js/js_loop_while.asp
Break and Continue:	http://www.w3schools.com/js/js_break.asp

Example 7.6

```
<HTML>
<HEAD>
<SCRIPT language="JavaScript">
<!--hide the script from old browser
function calculation(){
  var x = 1;
  var y = 2;
  var result = x + y;
  alert(result);
}
// close the hiding">
</SCRIPT>
</HEAD>
<BODY>
<FORM>
<INPUT TYPE="button" VALUE="1+2="
OnClick="calculation()">
</FORM>
</BODY>
</HTML>
```

The output of the above webpage is shown in the following figure:

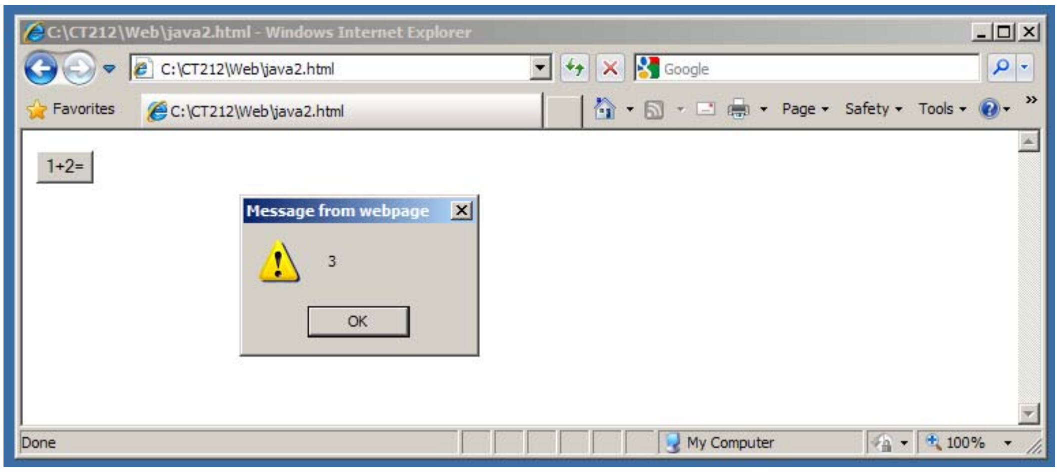


Figure 7.12 An example of the onclick event handler

We have trapped the `onclick` event handler of the form button to run the function `calculation()`. Inside the calculation, we are using the variables `x`, `y` and `result`.

Activity 7.4

Write a JavaScript program with a function so that when the user clicks on a form button, it prompts the user to enter two strings, concatenate the two strings, and display the result in an alert window.

You can get user input by using the prompt function as in the following:

```
var x = prompt("Please enter a string:", "default");
```

where `default` is the default value for the input.

Note that there is feedback for this activity provided at the end of the unit.

The document object model (DOM)

As mentioned, built-in functions of JavaScript are associated with the dynamic elements on the webpage. JavaScript arranges all of the elements on a webpage in a hierarchy. This is called the **document object model** (**DOM**), which includes the elements of the Web browser, those inside an HTML document, elements inside an HTML form, etc.

To develop interactive webpages, you have to understand the object model and use the properties and functions associated with the objects to create our desired effects. Let's examine a standard HTML document and see how it fits into the object model.

Example 7.7

```
<HTML>
<HEAD>
<TITLE>Your Name</TITLE>
</HEAD>
<BODY>
<P ALIGN="left"><IMG SRC="image0.gif" WIDTH="100"
   HEIGHT="100"></P>
<P><IMG SRC="image1.gif" WIDTH="500" HEIGHT="69"></P>
<P>&nbsp;</P>
<FORM NAME="MyForm">
<P>Your Name: <INPUT TYPE="text" NAME="YourName" size="20"></P>
<P>Password: <INPUT Type="text" NAME="Password" SIZE="20"></P>
<P><INPUT TYPE="submit" VALUE="Submit">&nbsp;
<INPUT TYPE="reset" NAME="B2"></P>
</FORM>
<P>&nbsp;</P>
<P><A HREF="home.htm">Back to Home Page</A></P>
</BODY>
</HTML>
```

The following figure shows the output of this document:

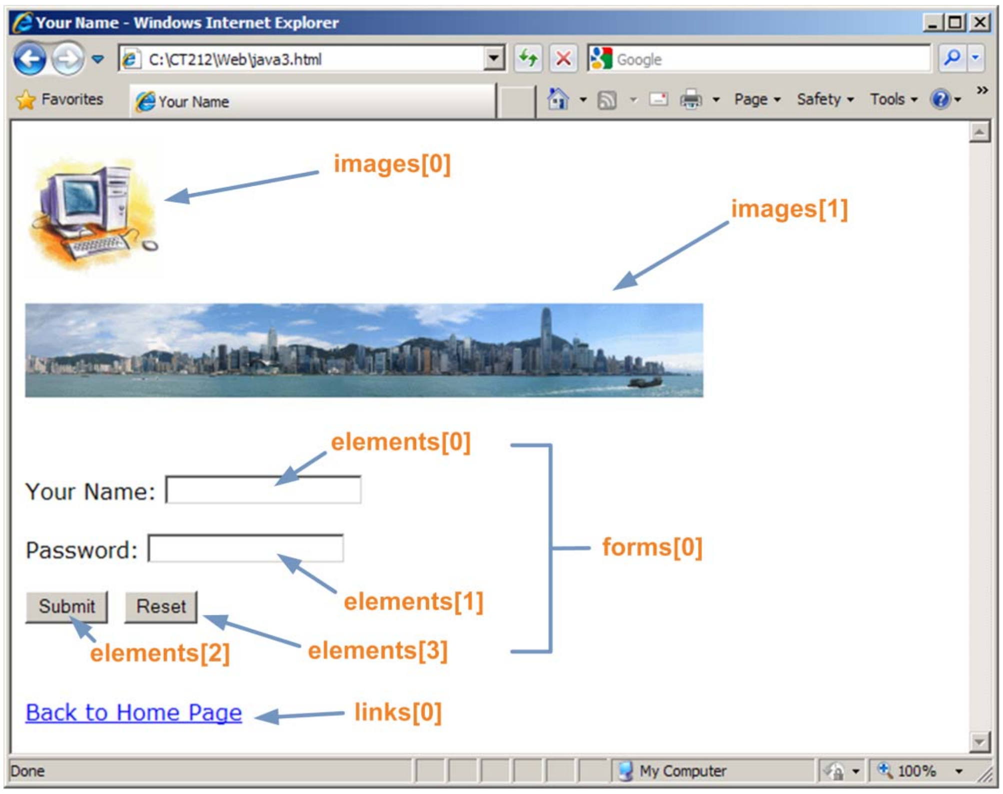


Figure 7.13 Example of HTML DOM objects

In the above example, we have two images, one form with two text fields and two buttons, and one hyperlink. All of these are objects in the JavaScript object model. From the JavaScript's object model, the browser window is a window object that contains many elements such as the status bar. Inside the window, we load an HTML document. This document is regarded as a document object, which is a very important object in JavaScript for providing interactive effects. Those shown in the above example belong to the document object. In fact, all HTML objects are properties of the document object. The following diagram illustrates the hierarchy of the JavaScript object model.

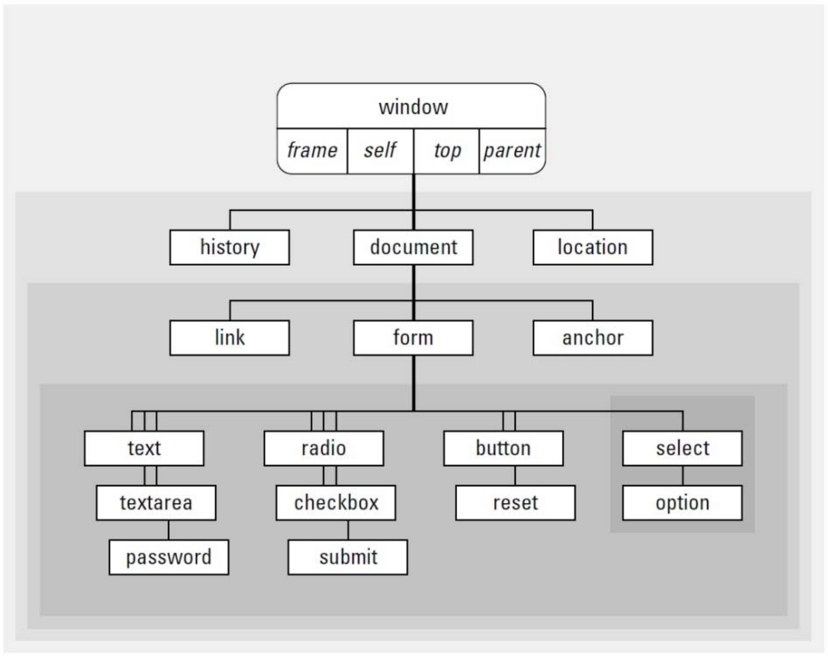


Figure 7.14 The basic DOM objects (Goodman 2007, Figure 14-1)

In order to get information about different objects and manipulate them, we must know how to access the objects in the hierarchy. The method of access is from the top to the bottom, first from the browser window, then the document, and then the elements in the document, and finally each part of the element, if the element is composed of different parts.

The first image in the document is represented through `images[0]` in the above example. We can gain access to this object in JavaScript with `window.document.images[0]`. In general, we can omit the window object, since there must be only one window in a browser. The accessing path can be written as `document.images[0]`. Similarly, if you want to know what a user has entered into the first text field of the form which is represented by `elements[0]` in the above example, you must think about how to access this object. Again we start from the top of the hierarchy. As it is an element of the form with name `forms[0]`, you can access the first text field through `document.forms[0].elements[0].`

After determining how to access the object, you come to the problem of recognizing the text the user has entered. Each object is associated with its own properties and methods. The properties of an object are JavaScript variables that the object used to hold required information for its processing. The methods of an object are JavaScript functions that the object used to manipulate and carry out its processing. We do not introduce the properties and methods of each object one by one; instead, you will learn some of them through examples. Here, we introduce the value property of the text field so you can access the data inside the first text field of the form through `document.forms[0].elements[0].value`.

If a document contains a number of elements, it is very difficult to address each one through the number only. Also, this is very hard for maintenance. You can instead give a different name to each element. For example, the form in the above example is given the name `myForm` and its first text field the name `YourName`. Now you can access the data in first text element through `document.myForm.YourName.value`. This will make it much easier for the developer to understand and maintain the JavaScript code.

Our focus is on the `document` object. But under the window object you will also find the `location` and `history` objects. These two objects do not depend on the HTML code. The `location` object represents the address of the loaded HTML document. So if the current document is the URL: `www.ouhk.edu.hk`, the `href` property of the `location` object, that is, `location.href` is equal to this address. If you assign a new value to the `location.href`, the browser will automatically load that document immediately. This is an important trick in redirecting users to different URLs. The history object has properties representing URLs that have been previously visited.

The following reading discusses more sections of JavaScript.

Reading 7.18 (online)

w3schools.com, 'JavaScript':

HTML DOM: http://www.w3schools.com/jsref/default.asp

The following example shows you a simple JavaScript calculator available at the URL `www.javascriptsource.com`. There are many interesting JavaScript programs there that you can download and try.

Example 7.8

```
<HTML>
<HEAD>
<TITLE>Simple Calculator</TITLE>
</HEAD>
<BODY>
<CENTER>
<FORM NAME="Calc">
<TABLE BORDER=4>
<TR>
<TD>
<INPUT TYPE="text" NAME="Input" Size="16">
<br>
</TD>
</TR>
<TR>
```

```
<TD>
<INPUT TYPE="button" NAME="one" VALUE=" 1 "
       onclick="Calc.Input.value += '1'">
<INPUT TYPE="button" NAME="two" VALUE=" 2 "
       onclick="Calc.Input.value += '2'">
<INPUT TYPE="button" NAME="three" VALUE=" 3 "
       onclick="Calc.Input.value += '3'">
<INPUT TYPE="button" NAME="plus" VALUE=" + "
       onclick="Calc.Input.value +='+'">
<br>
<INPUT TYPE="button" NAME="four" VALUE=" 4 "
       onclick="Calc.Input.value += '4'">
<INPUT TYPE="button" NAME="five" VALUE=" 5 "
       onclick="Calc.Input.value += '5'">
<INPUT TYPE="button" NAME="six" VALUE=" 6 "
       onclick="Calc.Input.value += '6'">
<INPUT TYPE="button" NAME="minus" VALUE="-"
       onclick="Calc.Input.value += '-'">
<br>
<INPUT TYPE="button" NAME="seven" VALUE="7"
       onclick="Calc.Input.value += '7'">
<INPUT TYPE="button" NAME="eight" VALUE="8"
       onclick="Calc.Input.value += '8'">
<INPUT TYPE="button" NAME="nine" VALUE="9"
       onclick="Calc.Input.value += '9'">
<INPUT TYPE="button" NAME="times" VALUE="x"
       onclick="Calc.Input.value += '*'">
<br>
<INPUT TYPE="button" NAME="clear" VALUE="c"
       onclick="Calc.Input.value = ''">
<INPUT TYPE="button" NAME="zero" VALUE="0"
       onclick ="Calc.Input.value += '0'">
<INPUT TYPE="button" NAME="DoIt" VALUE="="
       onclick ="Calc.Input.value = eval(Calc.Input.value)">
<INPUT TYPE="button" NAME="div" VALUE=" / "
       onclick ="Calc.Input.value += '/'">
<br>
</TD>
</TR>
</TABLE>
</FORM>
</CENTER>
</BODY>
</HTML>
```

The following figure shows the output of this document:

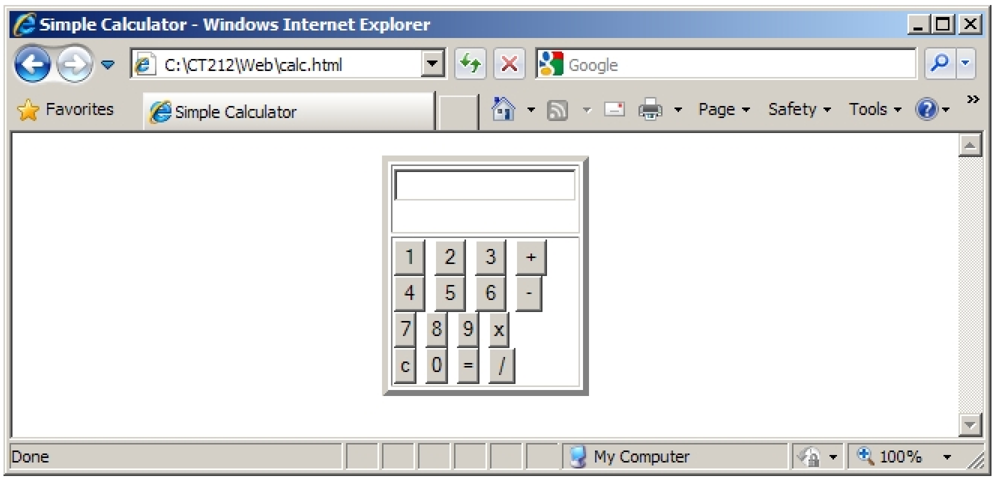


Figure 7.15 A simple calculator written in JavaScript

The above code is simple and easy to understand. Actually, nothing should be new to you, except the function `eval()` which gets a simple arithmetic expression as input, evaluates it and returns the result. `Calc.Input.value` is the value held by the text field Input of the form `Calc`. When you click on the calculator buttons other than the `=` sign, the expression on the text field is updated. After you enter your arithmetic expression, you click on the `=` sign button which calls the function `eval()` for calculation and returns the result.

Activity 7.5

Based on the above calculator example, write a simpler calculator that consists of a text field with the name `Expr` and a button named `Calculate`. Users should be able to type their arithmetic expression in the text field. After entering the expression, they should be able to click on the button and see the results shown in the alert window.

There is feedback for this activity provided at the end of the unit.

Controlling the browser window

When surfing the Web you may have encountered a new browser window that suddenly opens when you browse some webpages. The newly-opened browser window may have customized configurations such as no menu bar, no toolbar, no status bar, etc. However these are not the configurations of your browser.

The new browser window shows that the HTML document loaded can control the browser window settings. The 'magic' of this kind of interaction is done through the JavaScript window object. You can open a new window to load a new document. Take a look at this example.

Example 7.9

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
<!--
function openWin(){
  myWin = open("newpage.htm");
}
//-->
</SCRIPT>
</HEAD>
<BODY>
<FORM>
<INPUT TYPE="button" VALUE="Open new window" onclick="openWin()">
</FORM>
</BODY>
</HTML>
```

The function `open()` helps you load an HTML document into a new window. So, after you click on the **Open new window** button, you will get a new window loaded with the HTML document `newpage.htm`, provided that this document exists. (If you try the above code, remember to substitute `newpage.htm` with an HTML document that exists on your computer.) You can control the appearance of the new window by changing the open function as follows:

```
myWin = open("newpage.htm", "newWindow", "width=400,
               height=300,status=no,toolbar=no,menubar=0")
```

Here you have specified the size of the window to be 400 × 300 pixels and no status bar, toolbar and menu bar for the window. The second parameter '`newWindow`' is the name of the newly opened window. You may specify the `TARGET` of the `<A>` tag as the name of the window like:

```
<A HREF="newpage2.htm" TARGET="newWindow">Click Here</a>
```

When the user clicks on this hyperlink, the HTML document is loaded in a new window with configurations as `newWindow`.

The counterpart for the `open()` function is the `close()` function, which closes a window on calling. Both of these functions are methods of the `window` object. Normally, we have to write `window.open()` and `window.close()` instead of `open()` and `close()` according to the object model hierarchy. However, the `window` object is an exception, as stated above. You don't have to write window if you want to call a method of the window object. Table 7.10 summarizes the properties of the window that you can control.

Table 7.10 Properties of the Window object

Events	**Values**
directories	yes | no
height	number of pixels
location	yes | no
menubar	yes | no
resizable	yes | no
scrollbars	yes | no
status	yes | no
toolbar	yes | no
width	number of pixels

The status bar of the browser window normally shows the current URL. Have you ever noticed that? Some of you may find that the status bar can show values other than this. In fact the status property of the window object holds this value, and the default is the current URL. You can assign a new value to the status property, making the status bar display something new. Consider the code shown in the following example. When you click on the **Write!** button, the status bar will change to display the message ‘Hello! How are you?’. When you click on the **Erase!** button, the status bar will be cleared.

Example 7.10

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
<!--
function statbar(txt){
  window.status=txt;
}
//-->
</SCRIPT>
</HEAD>
<BODY>
<FORM>
<INPUT TYPE="button" NAME="write" VALUE="Write!"
       onclick="statbar('Hello! How are you? ');">
<INPUT TYPE="button" NAME="erase" VALUE="Erase!"
       onclick="statbar('');">
</FORM>
</BODY>
</HTML>
```

Common Gateway Interface (CGI) scripts

As you've seen, in addition to JavaScript, Common Gateway Interface (CGI) scripts enable Web developers to write interactive webpages. But in contrast to JavaScript, which is a client-side scripting language, CGI allows the Web client to execute programs on the Web server. The server then returns the results to the client.

CGI scripts are just normal executable programs or system script files like UNIX shell scripts. Compiled programs may be written in any programming language, like C and C++. Interpreted scripts may be written in any scripting language such as Perl or UNIX shell script, provided the system running the Web server supports it.

The following points should be noted in using these two types of CGI application:

- Compiled programs can be executed much faster than interpreted scripts. For applications with many user accesses, compiled programs are preferred.
- Scripts can be developed and modified more easily. They can be modified directly and take effect immediately without the need for recompilation.
- A script doesn't need to keep its source files separately.

Frequently, CGI scripts are placed in the `cgi-bin` directory of the Web server. However, the Webmaster may modify this path. Therefore, you may find that your ability to write interactive webpages on a Web server using CGI scripts depends on the policy of the website. Given this restriction, does it influence your answer to the question of which method — i.e. JavaScript or CGI scripts — is a better tool for you in writing interactive webpages?

To invoke a CGI program, you can create a hyperlink to the CGI program with the `HREF` attribute of the `<A>` tag. The following is an example:

```
<A HREF="http://www.myserver.com/cgi-bin/mycgi.cgi">
```

CGI program files usually have the extension `cgi`, but this is not required.

CGI methods

In JavaScript, the script gets user input through the properties of the objects. If you are using CGI, a similar question arises: How does the CGI program on the server receive the necessary data from the client for processing? There are two methods: GET and POST. The developer has to determine which method his CGI program will use in receiving client data at the time the program is developing. A program that uses the GET method gets the data from a pre-defined environment variable. A program that uses the POST method gets the data from the standard input device.

Frequently, an HTML form is used to collect user input for the CGI programs. The ACTION attribute of the `<FORM>` tag indicates which method the client will use in order to pass data to the server.

The GET method

Using the GET method, a CGI program retrieves the data from the environment variable `QUERY_STRING` and then decodes it for further processing. The `QUERY_STRING` is defined as anything which follows the first `?` in the URL. For example, consider the following URL:

```
http://www.myserver.com/cgi-
bin/mycgi.cgi?firstname=bill&lastname=gate
```

The data '`firstname=bill&lastname=gate`' is then put in the environment variable `QUERY_STRING`. The CGI program that receives this input decodes the string into the original values and then executes them.

Some special input characters sent to CGI script are encoded by the sequence `%HH`, where H is a hexadecimal digit. A space is a special character that is encoded by the plus '`+`' sign. Your CGI program has to decode the `%HH` and the `+` sign back to its actual meaning before further processing the string.

Table 7.11 Special characters in QUERY_STRING.

Input characters	**Encoded characters**
space	`+`
%	`%25`
=	`%3D`
&	`%38`
Line feed	`%0A`
Carriage return	`%0D`

POST method

For the POST method, the CGI program will retrieve the encoded form input from the standard input device; that is, `stdin` in UNIX. But there is a problem: the program does not know the end of the data, as no End-Of- File (EOF) mark is sent to the CGI program. To solve this problem, the CGI program has to use a pre-defined environment variable with the name `CONTENT_LENGTH` to determine the length of data that should be read from the standard input device.

Sending the results back

To return the results to the client, the CGI program has to specify the document type. The reason is that the result after processing may be of different types like an image file, an HTML document, or an audio clip. It may also be a link to another location. To do this, the CGI program places a short header on the output. This header is written in ASCII text. There should be a single blank line between the actual content and the header.

To specify the document type, the first line of the header should contain the keyword '`Content-type`' to indicate the MIME type of the document.

The following is an example of outputting an HTML file through a CGI program.

Example 7.11

```
Content-type: text/html

<HTML>
<HEAD>
<TITLE>CGI Output</TITLE>
</HEAD>
<BODY>
This is output from a CGI program.
</BODY>
</HTML>
```

If the result is a redirection to another location, the second line should contain the keyword '`Location`' to specify the URL of the new document. The following is an example of this:

Example 7.12

```
Content-type: text/html
Location: http://www.myserver.com/newpage.htm

<HTML>
<HEAD>
<TITLE>DUMMY</TITLE>
</HEAD>
<BODY>
```

```
This part should not be shown in the client. The
client will be redirected to the newpage.htm.
</BODY>
</HTML>
```

The value for Location should be a valid URL.

A simple CGI script

The following shows a simple CGI script written in UNIX shell script. The script uses the GET method, and returns the original data that the client has sent to the server in the form of an HTML document.

Example 7.13

```
#!/bin/sh
echo Content-type: text/html
echo
echo "<HTML>"
echo "<HEAD>"
echo "</HEAD>"
echo "<BODY>"
echo "<P>Here is the data submitted through GET
method:"
echo "<PRE>"
echo QUERY_STRING = $QUERY_STRING
echo "</PRE>"
echo "</BODY>"echo "</HTML>"
```

The following reading presents a very clear description of the basics of CGI programming with many good examples and exercises. It describes how to execute a CGI script file, decode data sent to a CGI script, record details about the current viewer of the webpage and send information to the script by using POST and GET methods. You can also try the exercises shown in the reading.

Reading 7.19 (online)

CGI Scripts, 'An interactive introduction to HTML and CGI scripts on the WWW':

http://www.cem.brighton.ac.uk/staff/mas/mas/courses/html/html.html

Summary

In this unit we briefly described the basic features of the World Wide Web. Since the main purpose of this unit was to help you gain basic programming skills for the Web, we did not go into the details of the underlying mechanisms. One thing you should remember, however, is taht the Web uses Hypertext Transfer Protocol, which is an application layer protocol like FTP, SMTP etc. running over TCP/IP. HTTP also uses the client-server model. The Web browser you use to surf the Web is the client program.

After this introduction, we immediately proceeded to explore the HyperText Markup Language, which is the language used in order to write webpages. HTML provides tags and directives for you to define the structure of a webpage. But you should have noted that it does *not* define the format of the document. The Web browser that interprets the document does this job. HTML does not define the page layout of the document either. To do this, you should write Web documents with the Cascading Style Sheets (CSS) language, an extension of HTML.

You covered the basic elements of HTML and should now be able to write your own webpages. You should also know the limitations of HTML — you can only write webpages with static information. Therefore, you have learned to write a simple program (script) with few decisions and limited input that is stored in a webpage. It can be classified as server-side script, which executes the program on the server. A client-side script is included in the webpage that is sent to the Web browser and is executed by the browser on the PC. Several server-side programming protocols such as CGI, ISAPI, ASP and SSI were introduced. Several programming languages such as Perl, C and C++, Java, VBScript, JavaScript, C# and PHP used for Web programming were discussed.

We focused on the two popular and simple scripting languages on the Internet: JavaScript and CGI. You can write interactive webpages by using either scripting language. JavaScript can be embedded in HTML documents. It can take on the information entered by the users, perform processing and then give output. One thing you should remember is that the processing is performed on the client machine. In contrast, the CGI programs process the user input on the Web server and provide output in the form of an HTML document. The output can then be displayed on the Web browser.

The Web is developing very quickly, so it is not possible to cover all of the skills for Web programming here. However, this unit can serve as a starting point for you to explore more in this area.

Suggested answers to self-tests and activities

Self-test 7.1

1 A client/server model.

2 A client computer makes requests for data from another computer. A server is the computer with the data that receives and responds to requests.

3 A website is a collection of webpages.

Self-test 7.2

1 Eight request methods are defined in HTTP/1.1:

- HEAD — asks for the response identical to the one that would correspond to a GET request, but without the response body. This is useful for retrieving meta-information written in response headers, without having to transport the entire content.
- GET — requests a representation of the specified resource. Note that GET should not be used for operations that cause side-effects, such as using it for taking actions in Web applications.
- POST — submits data to be processed (e.g., from an HTML form) to the identified resource. The data is included in the body of the request. This may result in the creation of a new resource, or the updates of existing resources, or both.
- PUT — uploads a representation of the specified resource.
- DELETE — deletes the specified resource.
- TRACE — echoes back the received request so that a client can see what intermediate servers are adding or changing in the request.
- OPTIONS — returns the HTTP methods that the server supports for a specified URL. This can be used to check the functionality of a Web server by requesting '*' instead of a specific resource.
- CONNECT — converts the request connection to a transparent TCP/IP tunnel, usually to facilitate SSL-encrypted communication (HTTPS) through an unencrypted HTTP proxy.

Of these eight request methods, GET and POST are used most often. Indeed, HTTP servers are required to implement at least the GET and POST methods.

2 a 200 OK

  b 404 Not Found

  c 503 Service Unavailable

3 The request is composed of three lines. The third line is empty and contains only the carriage return and line feed pair (`<CR><LF>`).

```
GET / HTTP/1.1<CR><LF>
Host: www.gov.hk<CR><LF>
<CR><LF>
```

Self-test 7.3

1 Dynamic webpages can change their content in response to external criteria.

2 The Web browser.

3

Server-side scripting	**Client-side scripting**
Scripts are executed on the server.	Scripts are executed on the browser.
Scripts can access resources (database, files, etc) on the server directly.	Script cannot access resources on the server directly.
Scripts are hidden to the viewer.	Users can easily view the code in the scripts.
Independent of browsers.	Dependent on browsers.
Requires more computational and memory resources on the server.	Requires no extra resources on the server.

Self-test 7.4

1 UNIX.

2 DLL's remain in memory waiting to be used.

3 PERL.

4 Java applets.

Activity 7.4

```
<HTML>
<HEAD>
<SCRIPT language="JavaScript">
<!--hide the script from old browser
function show_message(){
  var str1 = prompt("Please enter a string: ",
"default");
  var str2 = prompt("Please enter another string: ",
"default");
  var res_str = str1 + str2;
  alert(res_str);
}
// close the hiding-->
</SCRIPT>
</HEAD>
<BODY>
<FORM>
<INPUT TYPE="button" VALUE="Show Message"
                     onclick="show_message()">
</FORM>
</BODY>
</HTML>
```

Activity 7.5

```
<HTML>
<HEAD>
<TITLE>Your Calculator</TITLE>
</HEAD>
<BODY>
<FORM Name="Myform">
<P><INPUT TYPE="text" Name="Expr"></P>
<P>
  <INPUT TYPE="button" VALUE="Calculate"
         onclick="alert(eval(Myform.Expr.value))">
</P>
</FORM>
</BODY>
</HTML>
```

Glossary

Active Server Pages (ASP) — Microsoft's first server-side script engine for dynamically-generated webpages. Also known as Classic ASP, it has now been superseded by ASP.NET.

ASP.NET — a Web application framework developed and marketed by Microsoft to allow programmers to build dynamic websites, Web applications and Web services. It was first released in January 2002 with version 1.0 of the .NET Framework, and is the successor to ASP. ASP.NET is built on the Common Language Runtime (CLR), allowing programmers to write ASP.NET code using any supported .NET language.

client-side scripting — a technology to generate dynamic webpages by including scripts (typically JavaScript) in the webpages sent to the viewing browsers. The scripts are executed on the viewing browsers, not the Web server.

Common Gateway Interface (CGI) — a standard protocol for interfacing external application software with an information server, commonly a Web server.

document object model (DOM) — a cross-platform and language-independent convention for representing and interacting with objects in HTML, XHTML and XML documents. Objects under the DOM (also sometimes called 'Elements') may be specified and addressed according to the syntax and rules of the programming language used to manipulate them. The rules for programming and interacting with the DOM are specified in the DOM Application Programming Interface (API).

hyperlink — a reference in a document to an external or internal piece of information. Hyperlinks are the basic building block of hypertexts.

hypertext — text, displayed on a computer, with references (hyperlinks) to other text that the reader can immediately access, usually by a mouse click or keypress sequence. Apart from running text, hypertext can contain tables, images and other presentational devices. Other means of interaction may also be present, such as a bubble with text appearing when the mouse hovers over a particular area, a video clip starting, or a form to complete and submit.

HyperText Markup Language (HTML) — the predominant markup language for creating webpages. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, etc., as well as for links, quotations, and other items. It allows images and objects to be embedded and can be used to create interactive forms. The current version is 4.01.

Hypertext Transfer Protocol (HTTP) — an application-level protocol for distributed, collaborative, hypermedia information systems. Its use for retrieving inter-linked resources led to the establishment of the World Wide Web. The current version is 1.1, and it is specified in RFC 2161.

Internet Server Application Programming Interface (ISAPI) — an N-tier API of Internet Information Services (IIS), Microsoft's collection of Windows-based Web server services. The most prominent application of IIS and ISAPI is Microsoft's Web server. The ISAPI has also been implemented by Apache's mod_isapi module so that server-side Web applications written for Microsoft's IIS can be used with Apache as well.

Java — a full-feature programming language from Sun Microsystems that is targeted at Internet applications. Java is used in a wide variety of computing platforms from embedded devices and mobile phones on the low end, to enterprise servers and supercomputers on the high end. Java is nearly ubiquitous in mobile phones, Web servers and enterprise applications. And while less common on desktop computers, Java applets are often used to provide improved functionality while browsing the World Wide Web.

JavaScript — a scripting language used to enable programmatic access to objects within other applications. It is primarily used in the form of client-side JavaScript for the development of dynamic websites. JavaScript is a dialect of the ECMAScript standard and is characterized as a dynamic, weakly typed, prototype-based language with first-class functions.

PHP — an extremely popular general-purpose scripting language especially suited for producing dynamic webpages. It generally runs on a Web server, which is configured to take PHP code as input and create webpage content as output. It can be deployed on most Web servers and on almost every operating system and platform free of charge.

server-side scripting — a Web server technology in which a user's request is fulfilled by running a script directly on the Web server to generate dynamic webpages. It is usually used to provide interactive websites that interface to databases or other data stores.

server side includes (SSI) — a simple server-side scripting language used almost exclusively for the Web. Its primary use is including the contents of one file into another one dynamically when the latter is served by a Web server.

Uniform Resource Locator (URL) — a type of Uniform Resource Identifier (URI) that specifies where an identified resource is available, and the mechanism for retrieving it. A URL is commonly referred to as a Web address.

Visual Basic Script (VBScript) — a subset of Visual Basic, Microsoft's popular visual programming language, with no GUI building capability and with access to other applications via OLE. VBScript source code is embedded in HTML, and downloaded to the client in the HTML file, where it is compiled and executed in association with its runtime libraries.

Web browser — a software application for retrieving, presenting, and traversing information resources on the World Wide Web. The major Web browsers include Microsoft Internet Explorer, Mozilla Firefox, Apple Safari, Google Chrome and Opera.

Web server — the software that is responsible for accepting HTTP requests from clients (usually Web browsers), and serving them HTTP responses along with optional data contents, which are usually webpages such as HTML documents and linked objects (images, etc.). The computer that runs the software described above is also called a Web server.

References

Goodman, D (2007) *JavaScript Bible*, 6th edn, Wiley.

Keogh, J (2005) *ASP.NET 2.0 Demystified*, McGraw-Hill.

Kwan, R, et al. (2009) *A Practical Approach to Internet Programming and Multimedia Technologies*, The Open University Press.

MacDonald, M (2007) *Beginning ASP.NET 3.5 in C# 2008: From Novice to Professional*, 2nd edn, Apress.

Robbins, J N (2007) *Learning Web Design*, 3rd edn, O'Reilly.

Online references

http://www.htmlref.com

http://www.javascriptsource.com

http://www.petesqbsite.com/sections/tutorials/tuts/fdlgame_design_process.html

http://www.w3schools.com

http://www.wikipedia.org