Web Order Forms with JavaScript

This book offers a self-study guide to using JavaScript for automatically updated order forms on Web pages. There are now many ready-made scripts to do this kind of task on the Web, but the purpose of the book is to give the reader the opportunity to learn some JavaScript along the way.

Mike Thelwall
School of Computing and IT
University of Wolverhampton
Wulfruna Street
Wolverhampton
WV1 1SB

Code for examples and answers to selected exercises are available at:

http://www.scit.wlv.ac.uk/~cm1993/mm2005

Introduction

1.1. Overview

This section will give an introduction to the language JavaScript, explaining where JavaScript programs run and what kind of things they are capable of doing. A brief history of the language will be given and it will be compared to HTML and Java.

1.2. Objectives

In this section you will learn the following.

- Where JavaScript came from.
- How and where it works.
- How it differs from HTML and Java.
- What kind of tasks it can perform.

1.3. What is JavaScript?

JavaScript is a programming language that can be used to enhance web pages. It was created to fill the need to create pages that were not just static, but could interact with the user in some way. It was developed by Netscape and Sun Microsystems and was released in 1995, being first implemented in the Netscape Navigator 2.0 web browser. Today, the vast majority of browsers in use support one version or another of JavaScript. It is used for different types of jobs, one very common one being changing the appearance of a link or image when the mouse pointer is over it, and it is also used to check the data entered in online web forms.

JavaScript was designed to be easy to learn and use so that non-programmers would not be left out. The result is a language that tries to avoid unnecessary complications and allows the writing of short useful sections of code. You do not have to be a programmer to make good use of JavaScript, although programmers will be able to make it really sing. By the end of this course you will be able to write sections of code that will really improve the functionality of your web pages. Once you have learned the basics of the language you will also be able to go to one of the numerous JavaScript web sites and use the free source code to add pre-written applications to your pages.

There are several different versions and variants of JavaScript. The versions currently available are 1, 1.1, 1.2, 1.3 and 1.4, with version 1.3 appearing in Netscape Navigator 4.06, and version 1.4 set to appear in Navigator 5.0. Internet Explorer 5 uses most of version 1.4, but these differences are only important when using the newer features of the language, which are avoided here. JavaScript before 1.0 had a different name, LiveScript, and it is also associated with another name, ECMAScript. Netscape submitted JavaScript to the European Computer Manufacturers' Association in 1996, and the official specification of ECMAScript was the result. This defines the core functionality of the language, whilst allowing later versions to have additional capabilities.

1.4. JavaScript, HTML and Java

JavaScript is sometimes confused with HTML, and often with Java. HyperText Markup Language (HTML) is the language in which web pages are written. It is a computer language but not a programming language. This means that it can only describe web pages but cannot interact with the user. JavaScript and Java are both

programming languages and can, therefore, perform ‘cleverer’ tasks than HTML. As an example of this, HTML could create a password box in a web page but Java or JavaScript could also check its contents, for example warning the user if the password was too short. The difference between Java and JavaScript is more subtle, but they have a different set of capabilities and also operate in different ways. Java is a full-scale programming language that can only be used effectively by a trained computer programmer. It is a powerful language that was designed with Internet in mind. The original name of Java was Oak and it was developed by Sun Microsystems as an operating system for electronic devices. It did not succeed in this role, but has made a name for itself as an Internet programming language. JavaScript, in contrast to Java, is not too complicated for non-programmers to use. Another difference is that Java programs need to be translated into a machine-readable form before being sent to a web browser. As a result of this, Java programs cannot be included in the HTML of a web page, although JavaScript programs can. To use a Java program in a web page a link must be created to a separate file containing the machine-readable code. The most important difference between the two languages, however, is that Java is much more powerful, for example allowing access to files on the user’s computer. The disadvantage of this extra power is that it is a security risk: a hacker that ran a Java applet on your computer could look at all of your files and perhaps destroy them. JavaScript is not such a security risk and therefore many people who disable Java in their browser will be happy to let JavaScript programs execute.

1.5. What Can JavaScript Do?

JavaScript is very good for adding extra interactivity to web pages. Probably the most common use of JavaScript code is to change an image on a web page in response to the position of the mouse. This is sometimes called an image rollover. Many pieces of software signal to the user that the mouse is over a clickable button by altering the appearance of the clickable part in some way, and JavaScript is needed to do this in a web page. As well as changing the appearance of a web page, JavaScript can also perform spreadsheet-type calculations by processing the contents of web forms. The processing may be a simple check, such as ensuring that the user has not left a key field blank, or it could be applying a mathematical formula to the data entered. JavaScript also has access to some information about the browser, in which it is running and can use this to customise a web page for a browser. All of these jobs do not need many lines of code to accomplish. To summarise the key capabilities, JavaScript can do the following: -

- Respond to user actions such as clicking and moving the mouse.
- Recognise and react to certain attributes of the environment in which it is running for example the name and version of the browser.
- Change images on the screen, and write HTML to include in a web page, and change the contents of form fields.
- Move to a new web page and open and close new customised browser windows.
- Perform calculations with numbers and process strings of text.

1.6. Summary

- JavaScript was developed by Netscape and Sun as a Web Browser ‘scripting’ language.

- JavaScript programs normally are found in web pages and are run by the browser.
- HTML is not a programming language. Java is, and is more powerful than JavaScript, but has security risks and needs a programmer to program.
- JavaScript can add extra interactivity to a web page.

1.7. Exercises

1. Search the web for a page of general information about Java and JavaScript. Start with Internet encyclopaedia at www.webreference.com.
2. Search the web for sites that have free demonstrations of JavaScript programs and try them out. You might like to start with www.JavaScript.com.
3. Look at the web page for this section.

2. Creating Web Pages with HTML

2.1. Overview

In this section we are going to learn how to create web pages using the raw HTML, in order to be able to edit the HTML later to include a JavaScript program. There are a series of tasks throughout the text, building up to the creation of a personal web page.

2.2. Objectives

In this section you will learn the following.

- How to use a text editor to create web pages.
- The names of some of the main HTML tags for formatting text.
- How to include a link from one page to another.
- How images are embedded in web pages.

2.3. HyperText Markup Language

Web pages are written in HyperText Markup Language, known as HTML. The easiest way to create web pages is by using a special web editor or by using the 'Convert to HTML' function of another application such as a word processor. In either case the program will convert your instructions into HTML and save your web page as a HTML file. If you use the 'view source' command in your web browser when you are viewing a page then you will see the underlying HTML.

An HTML file contains not only the text to be displayed but also extra instructions that tell the browser how to format and position this text, where to put additional resources, such as images, and can also give useful information about the page. These extra instructions are in the form of tags, starting with a less than symbol and ending in a greater than symbol. An example of a tag is <B> which tells the browser to start making the text bold. Tags must follow a standard format to be recognised by a browser and in this chapter a number of the main tags will be explained.

It is possible to write web pages by typing in the text and tags in a text editor program rather than a web editor. This is more difficult and takes longer, but we will need to learn how to do this in order to write JavaScript programs

2.4. Creating a HTML File

In this section you will create your own web page with a text editor. If you are using Windows then your computer should have the basic text editor program Notepad, other computer users should also have a similar program. A text editor is a program that can create and edit a text file without adding special formats such as using a larger font size. First we will have a look at an example of a very simple web page. Here is the HTML source of the page home.htm.

```
<HTML>
<HEAD>
</HEAD>
<BODY>
Hello, this is Mike's home page!
</BODY>
```

```
</HTML>
```

And when the file home.htm is viewed in a web browser, on the screen will appear just the plain text.

Hello, this is Mike's home page!

Notice that the document uses a number of tags. The <HTML> tag indicates the start of the web page and </HTML> indicates the end. Many tags use a similar structure, with a tag name starting with a slash indicating the end of the region. The <HEAD> tag indicates the start of the head of the document, and </HEAD> indicates the end. The head contains information that will not be displayed in the browser – more about this later. At the moment the head is empty. The body of the document, starting with <BODY> and ending with </BODY>, contains the elements that will be displayed in the browser. At the moment this is just a small amount of text.

Activity
Now try using a text editor to create your own simple home page. If you will be using Notepad, this program can normally be found by clicking on the Start button, selecting Programs and then the Accessories program group. Start up Notepad and type in the following simple HTML document.

```
<HTML>
<HEAD>
</HEAD>
<BODY>
Hello, this is [your name]'s home page!
</BODY>
</HTML>
```

Now save it on the floppy disk in the A: drive with the name HOME.HTM by selecting **Save As** from the **File** menu.

Minimise Notepad by clicking on the minimise button. Start your browser and when it has loaded type A:\HOME.HTM in the location bar. When you press return the page that you have just typed in should appear in the browser.
It should appear as:

Hello, this is [your name]’s home page!

If anything else appears, you have made a mistake with your file. Switch to Notepad and try to find it.

2.5. The Document Title

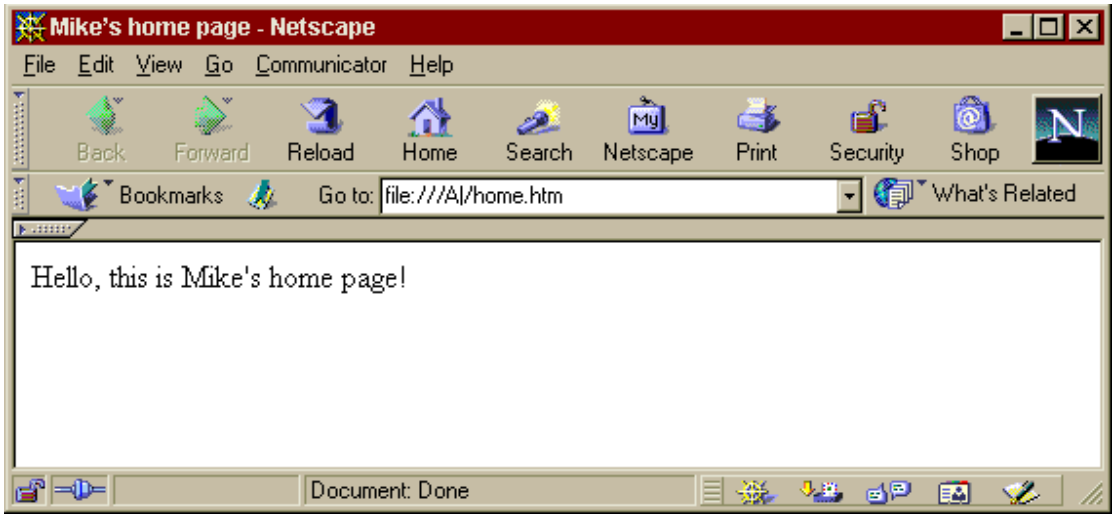
Documents are expected to have a title. The title does not get displayed in the main area of the browser window, it gets displayed in the title bar of the browser instead. Titles are very important because they are the default description of the page to be used when someone bookmarks it or adds it to the favourites list of their browser. The title goes in the *head* section because it is not displayed with the rest of the document. A title has its own start and end tags and shown in the following example.

```
<TITLE>
Page Title Goes Here
</TITLE>
```

Here is a title correctly inserted into the head of a web page, between the start of the header marker <HEAD> and the end of the header marker </HEAD>.

```
<HTML>
<HEAD>
<TITLE>
Mike's home page
</TITLE>
</HEAD>
<BODY>
Hello, this is Mike's home page!
</BODY>
</HTML>
```

This will appear on screen exactly as before.



Note that the title bar of the browser now contains the titles alongside the name of the browser.

Activity
Now have a go yourself. Minimise your browser, switch to Notepad and add the title '[your name]'s home page' to the head of your document. Save your document again with the **Save** menu option from the **File** menu. Switch to Your browser and look at the *title bar* at the very top of the screen, noticing what it says. Click on the Reload button and notice the change to the title bar. The title bar is the only thing that has changed. The main body of the document does not show your title.

2.6. Paragraphs and Typesetting

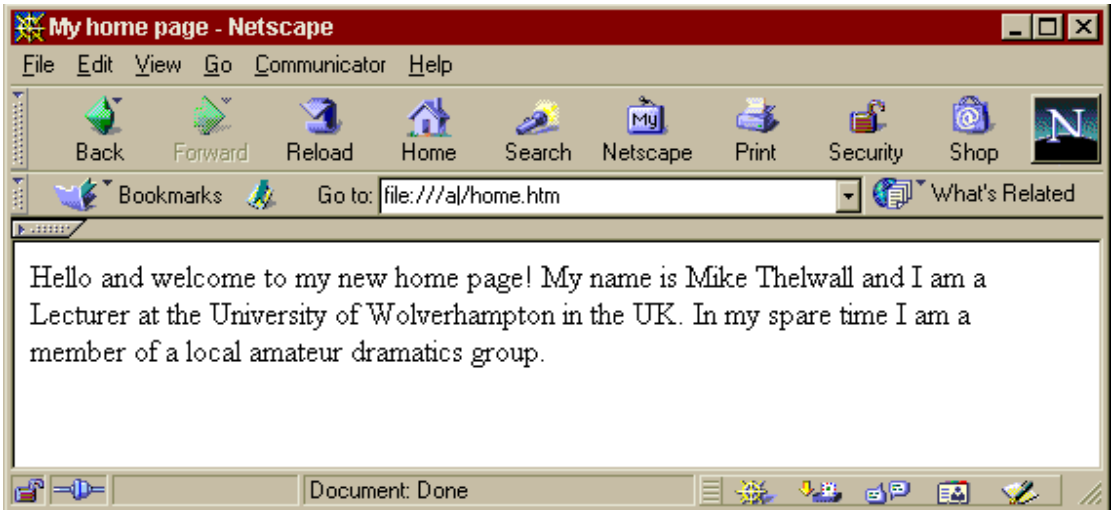
One thing that takes some getting used to with HTML is that browsers will ignore line endings in your document and realign all of the text into one big paragraph unless you tell them otherwise. Here is an example of this.

```
<HTML>
<HEAD>
<TITLE>
My home page
</TITLE>
</HEAD>
<BODY>
Hello and welcome to my new home page!

My name is Mike Thelwall and I am a Lecturer at the
University of Wolverhampton in the UK.

In my spare time I am a member of a local amateur
dramatics group.
</BODY>
</HTML>
```

When this page is loaded into a browser, all of the paragraph information is all lost, as is shown below.



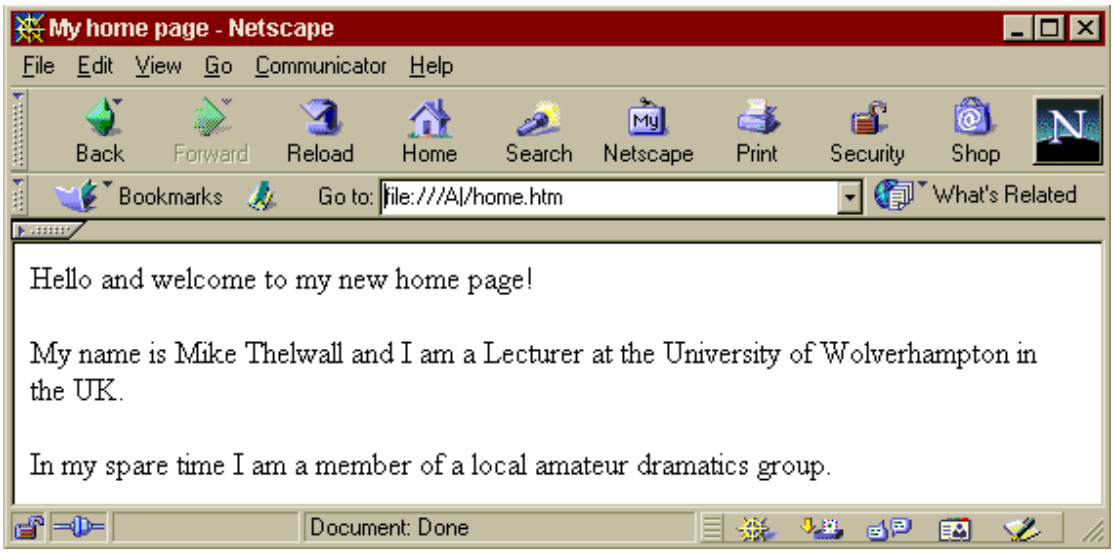
Notice that the browser has put all of the text into one big paragraph, fitting the line lengths to the screen width. The same page viewed in a smaller browser window would also be in one paragraph but with shorter lines.

There is a special tag to indicate the start of a new paragraph, which is <P>. Adding this to the start of the three paragraphs in my page will get them correctly displayed as three paragraphs.

```
<HTML>
<HEAD>
<TITLE>
My home page
</TITLE>
</HEAD>
<BODY>
<P>
```

```
Hello and welcome to my new home page!
<P>
My name is Mike Thelwall and I am a Lecturer at the
University of Wolverhampton in the UK.
<P>
In my spare time I am a member of a local amateur
dramatics group.
</BODY>
</HTML>
```

Loaded into a browser, this HTML file will now be split into three paragraphs by the <P> tags.



The reason why HTML allows the browser to decide things like the number of words to fit on each line of a paragraph is to enable people using different size browser windows and computer screens to still see all of the text in the document. It makes sense to just tell the browser where the paragraphs are and leave it to fit them on the screen as well as it can.

Activity
Now try formatting paragraphs with the paragraph tag. Switch back to Notepad and after the line <BODY> type in *three* paragraphs of information about yourself *without* using the <P> tags. Now resave the document, switch to your browser and reload it to see your new changes. Now add the <P> tags, save the file and reload in the browser to check that your paragraphs are correct. Try also changing the size of the browser window to see the effect that this has on how your document is displayed.

2.7. Document Structure

A large document usually contains headings and subheadings. You can mark your headings and subheadings with the following HTML tags.

```
<H1> Heading title </H1>      - for a level one heading
<H2> Heading title </H2>      - for a level two heading
<H3> Heading title </H3>      - for a level three heading
```

These headings will be formatted differently to the rest of the document in order to stand out. They will be on a line of their own and may be bold and in a larger font.

In the example below, the first line of the body of the HTML file has been converted from a standard paragraph into a level one heading.

```
<HTML>
<HEAD>
<TITLE>
My home page
</TITLE>
</HEAD>
<BODY>
<H1>Hello and welcome to my new home page!</H1>
<P>
My name is Mike Thelwall and I am a Lecturer at the
University of Wolverhampton in the UK.
<P>
In my spare time I am a member of a local amateur
dramatics group.
</BODY>
</HTML>
```

In my browser a level one heading is larger than normal text and also bold.

Add a heading to each paragraph in your home page file. The heading should go just above the paragraph. For example one paragraph heading could be:

```
<H1> My Job </H1>
```

View the changes and note how your browser treats headings. Level two and three headings will be less prominent than the main level one heading.

Activity
Expand your home page to several paragraphs and include headings and subheadings. You could include information on your job, your hobbies, your family and any other trivia that you don't mind others seeing. Use paragraph tags as you think appropriate. Make sure that everything that you type goes in the *body* of the document.

2.8. Text Formatting

If you do not put any formatting commands in a web page then all text, apart from headings, will be in the same font and in left-aligned paragraphs. In order to change these default settings some new tags must be learned. There are two classes of tags, those that apply to whole paragraphs at a time and those that can apply to individual words or letters.

Paragraph Formatting

There are a range of tags that can be used to specify the alignment of paragraphs and headings. The most commonly used is probably the <CENTER> tag, which, when put at the start of a paragraph causes it to be centred on the screen. This is commonly used for a main page title displayed in the browser. An end of centring tag should be put at the end of the heading or paragraph to be centred or all subsequent text may be incorrectly aligned. Here is an example of correct use.

```
<H1><CENTER>welcome to my home page!</CENTER></H1>
```

Activity
Your home page does not contain a title in its main body - the title we entered is at the title bar and separated from the rest of the document. We will add a title to the main body of the document now.

In Notepad, just below the tag <BODY> type in a title such as “My Home Page” and enter a paragraph end marker afterwards. Save it and view it in Your browser. It doesn't look right does it? A title should be in the centre of the screen, not at the left-hand side. We can tell Your browser to centre text by enclosing the text to be centred in the markers <CENTER> and </CENTER>. Note the American spelling! Do this to your title and view it to see the difference.

Note that the <CENTER> tag works on whole paragraphs at a time, it does not make sense to centre an individual word in a line, only whole lines or paragraphs. There are other tags that work on individual words or characters.

Character formatting

There are a large number of tags to change the font that is used to display text on the screen. The attributes that can be changed include the font name, its size and colour, but we shall concentrate on the simplest common changes here, bold and italic. To specify that some text is bold or italic, enclosing it in the appropriate markers <B> </B> or <I> </I> respectively.
Here is an example of some HTML employing the bold and italic tags.

```
I shall <I>quickly</I> become an <B>expert</B> html
writer.
```

When loaded into a browser, this will appear as follows.

I shall *quickly* become an **expert** html writer.

Activity
Go through your home page and use the bold and italic tags in a few places. Be warned however that too much use of bold and italic effects looks silly so they may not be appropriate in your document. There are many other character formatting effects such as those for changing colour and font size, but we are not going to try them all out here.

2.9. Adding Clickable Links

The most important property of a web page is its ability to link to other web pages through hyperlinks. A hyperlink in a web page is a picture or section of text that, when clicked, will cause a new page to be loaded. A hyperlink has two elements: the description seen in the browser, either some text or a picture; and the location of the new file to be loaded. The tag for a hyperlink is more complicated than the ones we have used previously because it has to incorporate these two components.

Hyperlinks are created with *anchor* tags. A text anchor tag has the following general form.

```
<A HREF="internet address">underlined text</A>
```

Here is an example of the HTML for a real link.

```
Here is a link to the
<A    HREF="http://www.microsoft.com/">Microsoft</A>
home page.
```

In a browser the link text will be highlighted in some way to show that it is clickable. Many browsers underline clickable links by default.

Here is a link to the Microsoft home page.

When the word Microsoft is clicked on the URL http://www.microsoft.com/ will be loaded in the browser.

Files can be specified in two ways: an *absolute address* and a *relative address*. An address starting with http:// is an absolute address and works as you would expect. Other addresses are relative addresses and the browser operates by adding them to the end of the current path. For example a page at http://www.a.com/pages/page1.htm which contained a reference to images/bob.jpg would have this interpreted as http://www.a.com/pages/images/bob.jpg. The path would have been interpreted as http://www.a.com/pages/ and the reference images/bob.jpg added to the end.

Activity
Use your browser to search for some sites that are of interest to you. Copy down their addresses from the Location bar and use these addresses to include a list of links to your favourite home pages using the anchor tag as above.

2.10. Adding Pictures and Images

HTML files are text documents and cannot contain embedded standard image files in the way that word processor documents can. Instead, any pictures to be shown on a web page must be saved to a separate file and referenced in the page using the appropriate tag, the image tag, IMG. This tag has one main argument, SRC = which should take the location of the image file to be inserted.
Here are some examples of uses of the image tag

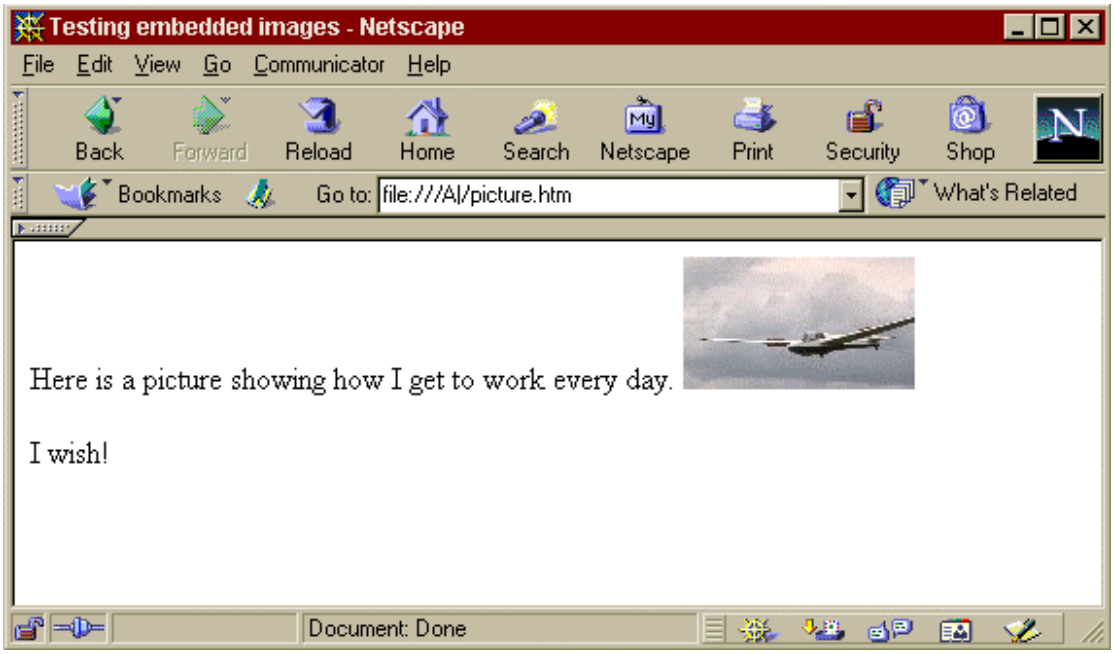
```
<IMG SRC="filename.gif">
<IMG SRC="filename.jpg">
<IMG SRC="http://www.somewhere.com/dir1/file2.jpg">
```

The actual image does not come in the HTML file, only the name of the file containing it. The filename.gif or filename.jpg must be real existing image files. Use this command to include in your home page some of the images that you have been given.

Here is an example of a web page with an image.

```
<HTML>
<HEAD>
<TITLE>
Testing embedded images
</TITLE>
</HEAD>
<BODY>
<P>Here is a picture showing how I get to work every
day. <IMG SRC=”glider.jpg”>
<P>I wish!
</BODY>
</HTML>
```

And here is the page in the browser.



An important point to stress again is that the picture itself is not in the file, only a reference to it. In this case the picture is saved in the file glider.jpg which was referenced by the HTML file by *relative* addressing. It was sitting in the same place as the original file, in this case in the main directory of the floppy disk in the A: drive and therefore only the filename was needed to reference it.

Activity
Add some suitable pictures to your web page. If you do not have any picture files to use then you could search the Internet for an archive of free images that you can borrow. All images on the Internet should be assumed to be copyright protected unless otherwise stated, so please do not take images that you are unsure of. The three common formats for saving files on the Internet are png, jpeg and gif. If you have windows pictures in bitmap format (.bmp) then there you will need to find a program which will convert them to one of the other formats. Most image manipulation programs should do this. The jpeg or jpg format is most suitable for photographic images, whearas gif is better for human created images, particularly icons. The png format can cope equally well with either.

2.11. Summary

- HTML documents are made up of text and tags.
- A HTML page is often split into head and body sections.
- There are special tags such as <TITLE> for the head of the document which are not displayed in the main browser screen.
- There are tags for character formatting as well as paragraph formatting.
- The <IMG SRC=""> tag causes a picture to be inserted into your page.
- The <A HREF="">…</A> tag creates a clickable link.

2.12. Exercises

1. Complete your home page by adding images, links and extra text as you see appropriate.
2. Reorganise your home page into a set of three pages, each containing part of the information from the original page and links to the other two pages. Save all pages into the same directory and use relative links to join them.
3. Use a search engine such as www.yahoo.com or www.altavista.com to search for web pages with libraries of free images. Bookmark them for later use and as some to your pages. You could start this with a search for “free clipart”.

3. Cascading Style Sheets

3.1. Overview

This section will give an introduction to Cascading Style Sheets, a feature of HTML that gives additional control over the appearance of web pages. The most powerful way of using style sheets will be demonstrated along with some of the most useful properties.

3.2. Objectives

In this section you will learn the following.

- How to create a stylesheet file
- How to reference a stylesheet from a web page
- How to configure common text properties
- How to use the classes feature

3.3. Using Stylesheets

It is possible with HTML to choose many properties of the text on the screen, such as its colour, size and font name. The Cascading Style Sheets (CSS) feature of HTML is designed to allow these properties to be changed for all elements in all web pages on a single site, simply by editing a single file. It is, therefore, a powerful way of gaining more control over the appearance of web pages. There are several ways in which CSS can be used but we will only cover the most potent one. Style sheets are a feature of HTML 4, and were not present in earlier versions. They are part of HTML rather than JavaScript, although JavaScript programs can exploit their features.

A CSS file is a plain text file created for example with a simple editor such as Windows Notepad and saved with the extension .css, for example filename.css. It contains formatting information in the appropriate notation. Here is a simple example

```
P     {
      color : red;
      }
```

This declares that all text in paragraphs should be red instead of black. To apply a style to a web page, a reference to the stylesheet file should be included in the HEAD of the document. If the above file were called mystyle.css then this reference would be the following.

```
<LINK REL=STYLESHEET TYPE = "text/css"
       HREF = "mystyle.css">
```

The formatting effects that can be applied include the following

- color — the font colour
- background-color — the colour of the background to text
- font-size — the font size, measured in points. common sizes are 10pt and 12pt.
- font-weight — the thickness of the letters on the screen,

	these can be bold, normal, or 100, 200, … 900, where load is 700 and normal is 400.
• font-family	the name of the font used to typeset the text. Common web fonts are Verdana, Arial and Helvetica.
• text-decoration	if this is set to none for the anchor element a then clickable links will not be underlined.

The above styles can be applied to any tags in the body of the document, although on some tags, such as INPUT they will be ignored by some browsers. Here is an example of a stylesheet that changes the links to green on a yellow background, without underlining. It also makes the paragraph text red (sorry!) and the main headings blue. The font Verdana is applied to all.

```
A     {
      text-decoration : none;
      font-family : Verdana;
      color : green;
      background-color : yellow;
      }
P     {
      font-family : Verdana;
      color : red;
      }
H1    {
      font-family : Verdana;
      color : blue;
      }
```

Note that the indentation used above is optional, but the file will be easier to read if you use the tab key to indent the brackets and style commands as above.

Activity
Create a web page with a reference to a stylesheet file as explained above. Experiment with the various styles and attributes to see the different results.

Fonts

When you specify a font, it may not be used in all web browsers because some people may not have that font installed on their computer. A common way to get round this problem is to list several fonts, with the preferred ones earlier in the list. If the computer is missing one font, it will try the next. For example: -

```
font-family : Verdana, Helvetica, Arial;
```

will use Verdana on those computers that have it, and try to use Helvetica on those that don't. Any computer that has neither of these will use Arial, or, failing this, the default browser font. The font 'Verdana' was specially designed to make text in web pages easier to read, so it is worth knowing about. Here is some text typeset in Verdana. If the font is not loaded on your computer then you will see the default font instead when you apply Verdana.

Colours

When you wish to assign a colour, if you are lucky then its name can be used directly. Pre-defined colours are red, green, blue, yellow, black, white, aqua, fuchsia, gray (note the American spelling of gray), lime, maroon, navy, olive, purple, silver and teal. If you want a different colour then you can specify it directly with #012345 where the first two digits refer to the amount of green in the colour, the second two green and the last two blue. For example red = #FF0000. The digits are hexadecimal and run from 0 to 9 then A to F. If you want to try this then good luck!

Classes

Sometimes it is necessary to apply different styles to the same HTML element. If you wanted some paragraphs of text to be on a white background but others on yellow then the classes feature must be used to differentiate. To ensure that a style does not apply to all elements of a given type, a class name must be given. Here is an example.

```
P.main    {
          background-color : yellow;
          }
```

This will not apply to text in a normal paragraph <P> but will to any paragraph with a class name of main.

```
<P CLASS = "main"> I want a yellow background. </P>
```

Any class name of your own choosing can be used and they can be used for any tag. Here is a more complex example where internal links will be blue and external red.

```
A.internal     {
               color : green;
               }
A.external     {
               color : red;
               }
```

This will have the intended effect on the following section of HTML.

```
<A CLASS = "internal" HREF = "index.htm">index page</A>
<A CLASS = "external" HREF =
"http://www.JavaScript.com/">JavaScript.com</A>
```

Summary

- A stylesheet file is a text file containing a list of properties for tags.
- A stylesheet file can be referenced using the LINK tag.
- A style definition is a bracketed list of values for defined tag properties.
- Classes enable single tags to be configured with two or more separate sets of properties applicable in different places.

Exercises

1. Use stylesheets to create a web page where all the headings are red on a blue background.
2. Modify your page created for question 1 above so that the paragraph text is blue on a red background.
3. Create a new page and use stylesheets to make its links fuchsia, without underlining and with the largest possible text weight. Apply the font Helvetica to everything on the page.
4. Create a separate class of the bold tag for each named colour. Create a document that lists all of the colours and uses the bold class definitions to show each word in its own colour.

4. Basic JavaScript

4.1. Overview

In this section we are going to create some simple JavaScript programs. You will learn how to use JavaScript to perform some simple tasks, including evaluating a formula.

4.2. Objectives

In this section you will learn the following.

- How to embed a simple JavaScript program in a web page.
- How to write text to a web page as it is being loaded in the browser.
- How to create pop-up boxes.
- How to declare and use variables.

4.3. Embedding JavaScript programs in HTML

JavaScript is not part of HTML, yet it can be placed inside the HTML of a web page. How is this apparent paradox possible? The answer lies in the SCRIPT tag. This tag is part of HTML and can be used to signal that the section of text that follows is not HTML but is a set of commands from a scripting language and should therefore be sent to the appropriate script interpreter rather than the HTML parser. There is more than one scripting language that can be used in web pages, and therefore it is necessary to give the name of the language used. JavaScript programs are normally preceded by

```
<SCRIPT LANGUAGE=JavaScript>
```

and followed by

```
</SCRIPT>
```

The actual program should be inserted between these two tags. A browser that understands JavaScript will recognise the <SCRIPT> tag and use its JavaScript interpreter to interpret the program rather than its HTML interpreter. It is possible to specify the version of JavaScript that is being used, for example with the tag

```
<SCRIPT LANGUAGE="JavaScript 1.2">
```

but for our purposes it will not be necessary to do this.

JavaScript programs can be placed inside the head or the body of a web page. It is common to put complex programs in the head so that they are fully loaded before the actual document is visible in the browser. This avoids problems caused by a program being called before it has been loaded. We are going to investigate writing some very simple programs and embedding them in web pages. No attempt will be made to give exhaustive descriptions of each function covered, instead they will be introduced with examples of common ways of using them.

4.4. The function document.write()

The function document.write() writes whatever text is placed in the brackets to the web page. Plain text in the brackets will need to be enclosed in single or double quotes. For example the command

```
document.write("Hello World!")
```

would write *Hello World!* to the web page. It is actually the same as writing Hello World! in plain text in a page, but bear with it as it will be more useful later. The

function has two parts separated by a full stop. The second one, write, specifies that the text in brackets is to be written somewhere. The first part, document, specifies what part of the system is responsible for doing the writing. In programming terms, the document is an object. In JavaScript 'document' effectively refers to the current web page, and we will not need to know more than this. In fact you can treat document.write() as one indissoluble entity. Note that unlike HTML, JavaScript is CaSe sensitive so document.Write or DOCUMENT.WRITE will not work. Both words must be lower case.

In order to get our single line of code to work we have to declare that it is a JavaScript program with the HTML script tag. Here is the program again, but in a version that you can insert anywhere in the head or body of a web page.

```
<SCRIPT LANGUAGE=JavaScript>
document.write("HTML Forms in JavaScript");
</SCRIPT>
```

Notice here that there is a semi-colon at the end of the line. The semi-colon is the official indicator of the end of a JavaScript statement, but it is in fact optional if the statement is not on the same line as any other statements. Here is an example of a complete web page with an embedded script.

```
<HTML><BODY>
<SCRIPT LANGUAGE=JavaScript>
document.write("HTML with embedded JavaScript.");
</SCRIPT>
</BODY></HTML>
```

The end result of this page in the browser will be the same as loading the following plain HTML.

```
<HTML><BODY>
HTML with embedded JavaScript.
</BODY></HTML>
```

In either case the browser window will simply display the following.

HTML with embedded JavaScript.

Activity
Create a web page to display the words "I've just loaded!" using the JavaScript document.write() function.

Tip
If your example does not work, it could be a mistake on your part or the problem could be with the browser. If you get a JavaScript error message, then it is likely to be your fault. Look very carefully at what you have written and try to find the problem. If nothing appears at all, it could be that JavaScript has been disabled in your browser. You will need to find out how to enable it again. In some versions of Netscape, this can be done by ticking the enable JavaScript box in the advanced section of the preferences box, obtained by selecting Preferences from the Edit menu. [IE5]

Using document.write() to include current information in a web page

The document.write function can process more than simple lines of text, it can also read some information from the browser and echo it back into the web page. In order to do this, you need to know the official name of the source of information that you wish to use. One simple example is document.location.href, which stores the URL of the document containing the JavaScript. This can be useful if you have a web page and you wish to talk about its location, despite the fact that there may be many copies of it in different places.

```
<HTML><BODY>
<SCRIPT LANGUAGE=JavaScript>
document.write(document.location.href);
</SCRIPT>
</BODY></HTML>
```

This will put the URL version of the file name into the web page. This could produce something like:

```
file:///U|/test.htm
```

for a local web page or

```
http://www.wlv.ac.uk/test.htm
```

for a page on the Internet.

Plain text can be combined with information in a JavaScript page by joining the two with a plus sign. Here is an example.

```
<HTML><BODY>
<SCRIPT LANGUAGE=JavaScript>
document.write("I live at " + document.location.href
                    + ".");
</SCRIPT>
</BODY></HTML>
```

Here is another example illustrating the ability to get information from the browser about itself, in particular its name and version number. This showcases JavaScript's ability to create pages that will not be the same in all browsers.

```
<HTML><BODY>
<SCRIPT LANGUAGE=JavaScript>
document.write("This browser is " +
                    navigator.appName);
document.write(" version " + navigator.appVersion +
                    "." );
</SCRIPT>
</BODY></HTML>
```

This is the first program shown so far that has more than one line. JavaScript programs can have any number of commands. Each command should be on a separate line to aid the readability of the code. The lines are processed in order and so we can guarantee that the text produced by the second line will be after that produced by the fist one. The output of this will be all on one line. In a Netscape Navigator version 4.7 browser running in a Windows NT operating system it will produce the following.

This browser is Netscape version 4.7 [en] (WinNT; I).

Activity
Create a web page that includes its own location using the JavaScript document.write() function in conjunction with document.location.href.
Load the browser name and version example into your web browser to see what information it gives about itself.

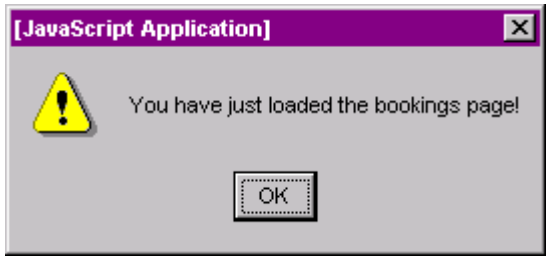
4.5. Pop-up boxes with alert()

The function alert() causes a pop up box to appear and can be used to draw the attention of the user to some important information. The example that we are going to look at here does not really show the uses of an alert box. They are often used at the end of a longer JavaScript program when some information has been processed and feedback must be given to the user. An example of a good use for an alert box is in a program to check whether a user has correctly filled in a web form. If a mistake is identified then a warning can be given with a pop-up box without having to load a separate page for it.

The same combination of text and information can be put inside the brackets as can be used with the document.write() function. Here is an example of a complete web page containing some working code.

```
<HTML><BODY>
<SCRIPT LANGUAGE=JavaScript>
alert("You have just loaded the bookings page!");
</SCRIPT>
This is the bookings page.
</BODY></HTML>
```

The above section of code produces the page containing the sentence "This is the bookings page." Before the page appears a box will pop up with the alert message.



The above example will produce a blank web page since the alert statement does not write to the browser window in the same way as document.write does.

Activity
Create a web page with a pop-up box that tells the user the name of the browser that they are using.

4.6. Variables

Like most programming languages, JavaScript allows you to use variables to store numbers and text in. A variable is a location in the computer's memory that stores some information for later use, often some text or a number. Each variable has a name used to identify it in the program. The names of these variables should be 'declared' with a var statement before they are used. Here are some examples of how the var statement can be used to declare some variables.

```
<HTML><BODY>
<SCRIPT LANGUAGE=JavaScript>
var i;
var CustomerName = "Fred Bloggs";
var StudentNumber = 34;
document.write(CustomerName)
document.write(StudentNumber)
</SCRIPT>
</BODY></HTML>
```

As these examples show, when a variable is declared it can be assigned a value and this value can be a number or text. Variables do not have to be declared at the start of the program, they can also be declared the first time that they are used.

Numbers and text can be added together with the + sign. This *adds* numbers and *concatenates* strings of text. Here are some examples of this.

```
<HTML><BODY>
<SCRIPT LANGUAGE=JavaScript>
var MyName = "Fred";
var YourName = "Jit";
var OurNames = MyName + " and " + YourName;
document.write(OurNames)
</SCRIPT>
</BODY></HTML>
```

At the end of this first example, the variable OurNames contains the string, "Fred and Jit". Here is another example with numbers.

```
<HTML><BODY>
<SCRIPT LANGUAGE=JavaScript>
var x = 3;
var y = 4;
var z;
z = x + y;
document.write(z)
</SCRIPT>
</BODY></HTML>
```

At the end of this second example, the variable z takes the value 7.

For arithmetic with numbers you can use the following table.

Operation	Add	Subtract	Multiply	Divide
Symbol to use	+	-	*	/

Round brackets can also be used in mathematical formulae. Here is another example.

```
<HTML><BODY>
<SCRIPT LANGUAGE=JavaScript>
var Height = 3;
var Width = 4;
var Length = 5;
var Volume = Height * Width * Length;
var SurfaceArea = 2 * (Height * Width + Height *
                  Length + Width * Length);
document.write("The volume of the box is " +
                  Volume + ".");
document.write("Its surface area is " +
                  SurfaceArea + ".");
</SCRIPT>
</BODY></HTML>
```

At the end of this example, the variable Volume takes the value 3 × 4 × 5 = 60. This code will print into the browser window the following line.

The volume of the box is 60. Its surface area is 94.

Activity
Create a web page that calculates the area of a rectangle using the formula
Area = height × width
Where the height is 10 and the width is 4. The answer should be displayed in a pop-up box.

4.7. Summary

- JavaScript programs can be embedded in a web page using the <SCRIPT LANGUAGE=JavaScript> and </SCRIPT> tags.
- The document.write command can be used to write HTML into a web page when it is loaded into the browser.
- The alert command causes a simple pop-up box to be displayed with a given message.
- Variables can be created and initialised with the var command. They can be used to help perform calculations.

Don't miss exercise 3 and 4 which introduce pop-up windows and the JavaScript back command!

4.8. Exercises

1. Describe the actions of the following JavaScript programs when embedded in the body of a HTML file and loaded into a browser.

```
<SCRIPT LANGUAGE=JavaScript>
```

```
document.write("Welcome to my page");
</SCRIPT>

<SCRIPT LANGUAGE=JavaScript>
alert("Welcome to my page");
</SCRIPT>

<SCRIPT LANGUAGE=JavaScript>
alert("You are viewing file: " + document.location.href);
</SCRIPT>

<SCRIPT LANGUAGE=JavaScript>
var MyName = "Charmaine";
var YourName = "Sarah";
var Story = MyName + " and " + YourName+" went to the
cinema";
document.write(Story)
</SCRIPT>

<SCRIPT LANGUAGE=JavaScript>
var TreeHeight = 44;
var TreeWidth = 3.5;
var TreePrice = 44 * TreeHeight * TreeWidth;
alert("The tree will cost " + TreePrice + " pounds.");
</SCRIPT>
```

2. Produce HTML pages with embedded JavaScript to do the following tasks. Make sure that you load your pages in a browser to check them.
   a) Display a page saying *This is my page*.
   b) Produce a blank page with an alert box displaying the message *This is a blank page*.
   c) Calculate the area of a rectangle of width 3 and length 4 and put it on the page as well as in an alert box with an appropriate message.
   d) Calculate the average of the numbers 3, 6, 7 and 11 and put the answer in an alert box with an appropriate message.

3. This is an exercise experimenting with new commands. The JavaScript command for creating a new window without any toolbars is window.open("URL"). Use this command to create a page that automatically opens another window.

4. This is another exercise experimenting with a new command. One of the JavaScript commands for going back to the previous page is history.go.back(), but it is used in place of the URL in the A HREF= command, and must be preceded by JavaScript: in the HREF argument. Here is an example.

   ```
   Go <A HREF="JavaScript:history.back()">Back</A>
   ```

   Use it command to create a page with a link back to the previous page.

5. HTML Forms

5.1. Overview

In this section we are going to learn how to create forms to capture user data from web pages. These forms will be created using the raw HTML, in order to be able to edit the HTML later to include JavaScript programs to process the form data for spreadsheet type calculations.

5.2. Objectives

In this section you will learn the following.

- How to add a form to a web page.
- The types of elements that are allowed to be included in a web form.
- What can happen to data in a form when it has been completed.

5.3. Introduction

Most web pages do not interact with the user, who is expected to look at them before clicking on a link to move to another. But some pages need to take specific information from a user to be processed in some way. In response to this need, version two of HTML introduced a new set of HTML tags that provide a means of capturing information from the mouse and the keyboard. These are the form tags. The most common place where these are seen is on a search engine. When you access a search page you are presented with a rectangular white box to type your keywords into. On some search pages you may also be able to select a range of options at the same time, such as how many hits to return per page or which area of the web to search, and these might be implemented by checkboxes, drop-down lists or one of the other possible types of form field. Forms can be processed in the browser with JavaScript, but their contents can also be sent back to a web server for storing or processing using CGI. When a search engine form is filled in with keywords these will be sent back to the server, where a program will match them against the database and produce a web page of results.

Forms are not only used for search purposes, but can also be used for a wide range of applications. Many web sites include a user feedback form or a request for information form on one of the pages.

We are going to combine forms with JavaScript to perform spreadsheet-type calculations. In a spreadsheet there is a range of cells in which data or formulae can be entered. The spreadsheet program automatically applies the formulae to the data whenever it changes, displaying the results in some other cells. We will use the text boxes of HTML forms to perform the function of spreadsheet cells and JavaScript for formulae and for some of the spreadsheet functionality that is not inherent in a web browser, such as causing the formulae to be calculated when the value of a cell changes. In this section we are going to look at HTML Forms and then later we will use JavaScript to make them active.

There is a range of different types of element that can be placed on a HTML form.

- Text Boxes
- Large text boxes
- Option buttons

- Check boxes
- Clickable buttons
- Selection lists
- Hidden text fields(!)

These can all be used by the person browsing the web page, except the hidden field, which will be discussed later. A form can be put anywhere in the boy of an HTML page. It must start with <FORM> and end with </FORM> but between these can go any HTML that you like as well as the tags for the various types of special form fields.

The syntax for a simple form is shown below

```
<FORM>
Please enter your name here: <INPUT TYPE=TEXT
NAME="YourName">
</FORM>
```

When inserted into an HTML page this will produce something like the following when viewed in a browser.

Please enter your name here: [ ]

The user will then be able to type their name into the box.

We will look now at three of the types of form fields: the input box, the option buttons and the submit button, which are the main ones that we shall be using. The syntax of the others is very similar, however, so you should not find it difficult to learn these if you need to.

5.4. The input box

The tag <INPUT TYPE=TEXT> produces a rectangular box in which the user can enter text. This is the simplest form of the command but there are various properties of the box that you can specify in the tag. The most important are NAME, SIZE and VALUE. Here are some tags with attributes set.

```
<FORM>
Please enter the following details about yourself.<BR>
Name: <INPUT TYPE=TEXT NAME="YourName" SIZE=30>
Age: <INPUT TYPE=TEXT NAME="YourAge" SIZE=2 VALUE="18">
Occupation:   <INPUT   TYPE=TEXT   NAME="YourOccupation"
SIZE=40 VALUE="Student">
</FORM>
```

This will produce the following section of a web page.

Please enter the following details about yourself.
Name: [ ] Age: [18] Occupation: [Student]

As you can see, the attributes are just listed after the INPUT TYPE=TEXT part and before the closing > of the tag. Some of the tags use quotes around them. In HTML these are strictly speaking only necessary if a value has a space or some other funny characters in.

- The NAME attribute names the actual field and is useful when a program needs to refer to the contents of the field.
- The SIZE attribute specifies the width, in characters, of the text box. In the example above the width is large for occupation and name but small for age.
- The VALUE attribute specifies the initial contents of the box. If this is omitted then the box will start off empty.

Activity
Modify the above example as follows.
(a) Change the default age to 25 and the default occupation to manager.
(b) Add an extra form field for nationality with a default value of Dutch.

5.5. Option buttons

The option buttons, known as radio buttons in HTML have a structure similar to that of text buttons but go in groups. Here is an example.

```
<FORM>
Please select the option that best describes you.<BR>
<INPUT TYPE=RADIO NAME="Occupation"
Value="Student">Student<BR>
<INPUT TYPE=RADIO NAME="Occupation"
Value="Lecturer">Lecturer<BR>
<INPUT TYPE=RADIO NAME="Occupation" Value="Other">Some
other occupation<BR>
</FORM>
```

This will appear in a browser something like this:

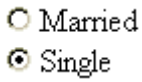
Please select the option that best describes you.
- ○ Student
- ○ Lecturer
- ○ Some other occupation

As you can see, option buttons come in groups. All members of the group must have the *same name*. Another difference between the option buttons and text fields is that you must give a value field, even though it does not appear when the page is viewed in a browser. It is good practise to put a break tag at the end of each option so that the options are all aligned together.
You can have more than one set of option buttons in a form so long as you give each group each a different NAME attribute. The optional attribute for option buttons is CHECKED. If you include this in one of the option buttons then it will start off being selected. The following option buttons could be added to the same form as the one above.

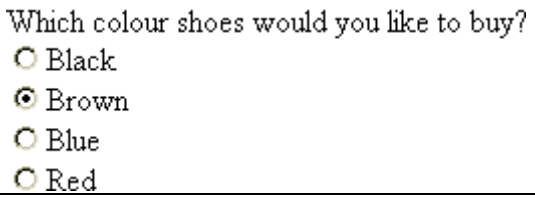
```
<FORM>
```

```
<INPUT           TYPE=RADIO           NAME="MaritalStatus"
Value="Married">Married<BR>
<INPUT   TYPE=RADIO   NAME="MaritalStatus"   Value="Single"
CHECKED>Single<BR>
</FORM>
```

These buttons should look something like the following when displayed in a web browser.



Activity
Create a new set of option buttons to look like the following. Remember to use the same name for each one.



5.6. The submit button

This is the button that will get your form processed when you click on it. The simplest syntax for a submit button is: <INPUT TYPE=SUBMIT> but you can also specify a name and a value for it as in the following example.

```
<FORM>
<INPUT TYPE=SUBMIT VALUE="Click here"
                    NAME="SubmitOrder">
</FORM>
```

This will produce the following clickable button.

The value attribute is written on the button itself, so if it were changed to “Process Order Now” then this would be displayed on the button.

Activity
Create a button with the name “Book” and caption “Place Booking Now”.

5.7. Whole forms

Forms can have any combination of valid form fields inside them. If the form is to be used to send its contents to a program on a server for processing via the CGI mechanism that search engines use then it will also have the attribute ACTION in the tag together with the web address of the program to do the processing. Later we shall include a program in the page itself to process the form using JavaScript. This could

be done with an **onSubmit** call in the submit button tag. Forms can also have NAME attributes.

```
<FORM                                    NAME="SubmitOrder"
ACTION="http://www.scit.wlv.ac.uk/~cm1993/ProcessOrder.cg
i">
```

Here is an example of a form with most of the features that we have looked at so far.

```
<FORM NAME="PersonalDetails">
Please enter the following details about yourself.<BR>
Name: <INPUT TYPE=TEXT NAME="YourName" SIZE=30>
Age: <INPUT TYPE=TEXT NAME="YourAge" SIZE=2 VALUE="18">
<P>
Please select the option that best describes your
occupation.<BR>
<INPUT          TYPE=RADIO          NAME="Occupation"
Value="Student">Student<BR>
<INPUT          TYPE=RADIO          NAME="Occupation"
Value="Lecturer">Lecturer<BR>
<INPUT  TYPE=RADIO  NAME="Occupation"  Value="Other">Some
other occupation
<P>
Please state your marital status.<BR>
<INPUT  TYPE=RADIO  NAME="MaritalStatus"  Value="Married"
CHECKED>Married<BR>
<INPUT  TYPE=RADIO  NAME="MaritalStatus"  Value="Single">
Single
<P>
<INPUT       TYPE=SUBMIT       VALUE="Send       Details"
NAME="SubmitInfo">
</FORM>
```

This will produce the following form.

Please enter the following details about yourself.
Name: [ ] Age: [18]

Please select the option that best describes you.
- ○ Student
- ○ Lecturer
- ○ Some other occupation

Please state your marital status.
- ● Married
- ○ Single

[Send Details]

5.8. What happens to the data submitted?

The data in a submitted form is normally processed by a program sitting outside the web page, on a web server. This program is often written in a complex programming language such as Perl or C. It is beyond the scope of this course to teach how to program something suitable. This is a major problem, since it means that the visitor would need to print out a form and post it in a letterbox, rather than submitting it. To get round this problem, many Internet Service Providers offer the service of a program that will take the data submitted in a form and send it to a given email address. If you are putting pages on the internet then you would need to talk to your service provider about this. Until then, you can use the program provided here to do the same job. This program is located at

```
http://www.scit.wlv.ac.uk/~cm1993/cgi/sendmail.cgi
```

This should be entered as the ACTION of the FORM command. You should also include the extra "hidden" form field element

```
<INPUT TYPE=HIDDEN NAME=mailto Value="myemail@myhost">
```

immediately after the form tag, where myemail@myhost is replaced by your email address. Here is an example of a working form set up as described.

```
<FORM
ACTION="http://www.scit.wlv.ac.uk/~cm1993/cgi/sendmail.cg
i" METHOD=POST>
<INPUT               TYPE=HIDDEN               NAME=mailto
VALUE=bill.gates@microsoft.com>
<INPUT TYPE=TEXT NAME=sample VALUE="example of a sample
field">
<INPUT TYPE=TEXT NAME=test VALUE="Another example">
<INPUT TYPE=SUBMIT VALUE="Sumbit this form">
</FORM>
```

Activity
Try out the above form, but use your own email address and check your mail to see the submitted data. When you have got this working, try using the same method to modify a form that you have created before so that it will be emailed to you.

5.9. Summary

- Forms in a web page start with the <FORM> tag, end with the </FORM> tag and can have any number of form fields in between.
- Web forms can include text fields and option buttons as well as other types of field.
- A submit button is often used to submit the data, which may be processed by JavaScript in the browser or sent to the server for processing.

5.10. Exercises

Note Use
<FORM METHOD=GET ACTION="http://www.wlv.ac.uk/~cm1993/cgi-c/query.cgi">
to see your form submitted to a program for processing via the CGI method, or use the email example above but using your own email address

2. Draw the forms that will be produced by the following HTML. Check your answer by creating pages and loading them into a browser.

```
<FORM NAME="EXERCISE_1A">
Enter your student ID number: <INPUT TYPE=TEXT NAME="ID">
</FORM>

<FORM NAME="EXERCISE_1B">
Type of Work: <INPUT TYPE=TEXT NAME="WorkType"
VALUE="Part time" SIZE=9>
</FORM>

<FORM NAME="EXERCISE_1C">
Choose a colour for your room to be painted.<BR>
<INPUT TYPE=RADIO NAME="Colour" Value="Blue">Morning
blue<BR>
<INPUT TYPE=RADIO NAME="Colour" Value="Red">Tropical
sunset<BR>
<INPUT TYPE=RADIO NAME="Colour" Value="Green">Royal
Azure<BR>
</FORM>

<FORM NAME="EXERCISE_1D">
Would you like a wake-up call for breakfast at
6.30am?<BR>
<INPUT TYPE=RADIO NAME="WakeUpCall" Value="Yes">Yes,
please<BR>
<INPUT TYPE=RADIO NAME="WakeUpCall" Value="No"
CHECKED>No, thank you<BR>
<INPUT TYPE=SUBMIT VALUE="Register"
NAME="SubmitRegistration">
</FORM>

<FORM NAME="EXERCISE_1E">
Enter your name: <INPUT TYPE=TEXT NAME="Name"
SIZE=35><BR>
Choose your breakfast menu:<BR>
<INPUT TYPE=RADIO NAME="Breakfast" Value="FullEnglish"
CHECKED>Full English<BR>
<INPUT TYPE=RADIO NAME="Breakfast"
Value="Continental">Continental<BR>
<INPUT TYPE=RADIO NAME="Breakfast" Value="None">None
<P>
Choose your pack lunch:<BR>
<INPUT TYPE=RADIO NAME="PackLunch" Value="Meat"
CHECKED>Lamb or beef sandwiches<BR>
<INPUT TYPE=RADIO NAME="PackLunch" Value="Vegetarian">Egg
mayonnaise and salad<BR>
<INPUT TYPE=RADIO NAME="PackLunch" Value="None">None
<P>
<INPUT TYPE=SUBMIT VALUE="Book Food" NAME="Food">
</FORM>
```

2. Produce HTML for forms to do the following tasks.

(a) Get the name of a customer and their address and credit card number.
(b) Get the name of a client and ask where they come from, giving the choices Wolverhampton, Birmingham, Other West Midlands and Other UK.
(c) Get the age of a car buyer and allow them to choose a primary colour for the car exterior and choose black, grey or brown for the interior. Include a submit button which should say, "Make me this car!" on it.

6. Checking the Data in Form Fields

6.1. Overview

In this section we are going to learn how to access the contents of form fields with JavaScript programs.

6.2. Objectives

In this section you will learn the following.

- How to access the contents of form fields.
- How to use the if statement to perform simple checks on the data entered in form fields.
- How to intercept the data from a form field when it has been submitted.
- How to align a set of form fields with an invisible table.

6.3. Introduction

JavaScript is a natural choice as a programming language to do some housekeeping jobs on a form. One common use is to verify the fields in a form when it is submitted and to cancel its submission if a key form is missing or incorrect. For example on a form to be filled in to request information a small JavaScript program could check that the form field for email address was not blank. JavaScript could also be used to perform calculations when a customer is ordering stock. A program could work out the total bill when the customer fills in form fields specifying how many of each stock item they are ordering.

6.4. Accessing form fields

In JavaScript you can access the current values of any of the form fields as long as you know the correct name. The names are a bit awkward but generally follow the convention

document.form_name.field_name.value

where form_name is the name you gave the form and field_name is the name you gave a field in the form. A form or a form field can be given a name by including NAME= with a name inside the tag. For example the following are valid tags.

```
<FORM NAME="Personal Details">
<INPUT TYPE=TEXT NAME="First Name" SIZE=12>
```

Here is an example of a program that accesses the value of a form field.

```
<HTML><BODY>
<FORM NAME="Age">
Please enter the following details about yourself.<BR>
Name: <INPUT TYPE=TEXT NAME="YourName" SIZE=30>
Age: <INPUT TYPE=TEXT NAME="YourAge" SIZE=2 VALUE="18">
</FORM>
<SCRIPT LANGUAGE=JavaScript>
alert("Enter age if not " + document.Age.YourAge.value);
</SCRIPT>
</BODY></HTML>
```

When loaded, this will produce a pop-up box saying, "Enter age if not 18". The form has been named *Age* and one of the form fields has been named *YourAge* and given the value 18 in the input tag and so the value of document.Age.YourAge.value is 18. Remember that all of the JavaScript is case sensitive although HTML is not so if you type in a different case for any of the letters in document.Age.YourAge.value then you will get an error in your program.
The script has to go below the form rather than in the head of the document in this case. If it is ahead of the form then the alert box will be produced before the form has been processed and so the form name will not be recognised and an error will be generated.

Here is another example of referencing a form value.

```
<HTML><BODY>
<FORM NAME="Sport">
Please answer the following questions.<BR>
What is your name? <INPUT TYPE=TEXT NAME="YourName"
SIZE=30> <BR>
What is your favourite sport? <INPUT TYPE=TEXT
NAME="favourite" SIZE=20 VALUE="football">
</FORM>
<SCRIPT LANGUAGE=JavaScript>
document.write("Most people's favourite sport is " +
document.Sport.favourite.value);
</SCRIPT>
</BODY></HTML>
```

This program will add the extra line "Most people's favourite sport is football" at the end of the document. The name of the form is Sport and the name of the favourite input field is favourite and so the reference for its value is document.Sport.favourite.value.

Tip
When you edit pages which use this method of accessing form field values, the browser can ‘play up’ and make mistakes, particularly if you repeatedly load modifications of your program. It also sometimes makes mistakes when the browser is first loaded. If you get this type of problem, close the browser down, then start it up again and reload the page.

6.5. Form processing with onSubmit

The onSubmit function can start a JavaScript function when the submit button is pressed. There are several ways of doing this but only the simplest will be covered here. This way is to include onSubmit=' inside the start FORM tag, then write the program and include another quote before the end > of the form tag.

```
<FORM onSubmit='
...program goes here...
'>
```

The program assigned to the onSubmit event will be processed once the submit button is pressed. Here is an example of a complete page with a program triggered by clicking on the submit button of a form.

```
<HTML><BODY>
<FORM NAME=Age onSubmit='
     alert("Name: " + document.Age.YourName.value);
     alert("Age: " + document.Age.YourAge.value);
     return false;
'>
Please enter the following details about yourself.<BR>
Name: <INPUT TYPE=TEXT NAME="YourName" SIZE=30
          VALUE="Joe Shmo">
Age: <INPUT TYPE=TEXT NAME="YourAge" SIZE=2 VALUE="18">
<INPUT TYPE=SUBMIT VALUE="process form">
</FORM>
</BODY></HTML>
```

The above program generates a form with default values Joe Shmo and 18. When the submit button is pressed, pop-up boxes appear showing the name and age values currently in the form boxes.
What is happening here is that the onSubmit command is activated when the submit button is pressed and this calls the program in the single quotes to be processed. The program is then processed as normal before the form is submitted. The program produces two alert boxes so these will both have to be displayed before submitting the form. The line saying `return false;` on the last line of the onSubmit section actually stops the page from reloading after the submit button has been pressed. The program will still work if it is left out.

The syntax is complicated here but you can copy the example above and make any necessary alterations without having to fully understand it first time. In order to perform any really useful task with the onSubmit event programs you really need to be able to test the values of the fields in the form. For this you will need to use an if statement so it is covered briefly below with some examples of using it on forms.

Activity
Create a web form that asks for a person's name and then when they click on the submit button responds with the message "Thank you [name] for filling in the form."

6.6. Using the if statement

Sometimes your program needs to do one thing or another based on the value a customer has entered. The if statement is provided for this eventuality. The layout of the if statement is:

```
if (condition){
…Statements to execute if the condition is true…
} else {
…Statements to execute if the condition is false…
}
```

When the program is executed, the condition inside the round brackets is evaluated. If that condition is true then the first set of statements is executed. If the condition is false then the second set of statements is executed instead. Here is a simple example.

```
var x=3;
if (x > 2) {
     alert("The value of x is greater than 2.");
} else {
     alert("The value of x is not greater than 2.");
}
```

The result of this program will be an alert box saying that the value of x is greater than 2. When the program is executed, the variable x will be created first and assigned the value 3. Then on the second line of the program the condition in the if statement will be checked. Since the value of x is greater than 2, the condition x>2 is true and so the first bracketed set of statements will be executed rather than the second, so the correct alert box will be shown. If the value of x were initialised as 1 then the other alert box message would be shown. Can you see why?

Logical conditions

The logical condition in the if statement can be constructed in a variety of ways. The most common are to check whether a number or a string is equal to another or not. The following table can be used to find the appropriate symbol to use for logic conditions.

Operator	= =	!=	>	<	>=	<=
Meaning	Equals	Not equals	Greater than	Less than	Greater than or equal to	Less than or equal to

Notice that the equality check sign is a double equals sign! The single equals sign is the assignment equals. Here is an example to illustrate that point.

The statement x = 3 would assign the value 3 to the variable x but the statement x = = 3 would not change the value of x, but would be true if x was 3 and false if it wasn't.

Here are some more examples

```
if (x != 4) {
     alert("The value of x is not 4!");
} else {
     alert("x is 4.");
}

if (x >= 3) {
     document.write("x is at least 3.");
} else {
     document.write("x is less than 3.");
}
```

```
if (name == "") {
     alert("You have not entered your name!");
} else {
     alert("Your name is: " + name);
}
```

The last example illustrates that the comparisons can be performed on strings as well as numbers.

There does not have to be an "else" part of an if statement, this can be omitted as in the following example.

```
if (name == "") {
     alert("You have not entered your name!");
}
```

Examples

Here is an example of combining 'if' statements with the onSubmit event.

```
<HTML><BODY>
<FORM NAME=Age onSubmit='
alert("Name: " + document.Age.YourName.value);
if(document.Age.YourAge.value < 18){
     alert("You are under age!");
} else {
     alert("Your age is: " + document.Age.YourAge.value);
}
return false;
'>
Please enter the following details about yourself.<BR>
Name: <INPUT TYPE=TEXT NAME="YourName" SIZE=30
          VALUE="Joe Shmo">
Age: <INPUT TYPE=TEXT NAME="YourAge" SIZE=2 VALUE="18">
<INPUT TYPE=SUBMIT VALUE="process form">
</FORM>
</BODY></HTML>
```

This example works exactly like the previous example except that if the age entered is below 18 the message will be "You are under age!"

Activity
Create a web form that asks for a person's height in centimetres and then when they click on the submit button responds with the message, "I'm sorry, you are too tall to book a ride", if their height is above 150cm.

6.7. Forms in Tables

HTML has the ability to tabulate information, placing it into appropriately aligned tables. This feature was not present in the earliest versions of HTML, so the oldest browsers will not support it. Tables are not only used for tables of figures but are commonly used simply to align the elements of a web page. It is possible to switch off

the borders of the table, making it invisible. This should always be done if tables are used for alignment purposes.

Tables are very useful to align form fields and also to create a spreadsheet effect. We shall cover the basic tags for tables here, but not the advanced options.
A table starts, unsurprisingly, with the <TABLE> tag and ends with the </TABLE> tag. If you want your table to have invisible borders then the initial tag must have the option BORDER = 0, otherwise BORDER = 1 gives a single pixel width border. In a table you must specify each row and each cell, but not the columns, which are implicit.

- Each row must start with <TR> and end with </TR>
- Each cell must be in a row and start with <TD> and end with </TD>

Here are some examples to illustrate this. First is a table with two cells in two rows.

HTML	Comment
<TABLE BORDER=1>	Start of table with a 1 pixel width border
<TR>	Start of (first) row
<TD>	Start of (first) cell
Cell 1	
</TD>	End of (first) cell
<TD>	Start of (second) cell
Cell 2	
</TD>	End of (second) cell
</TR>	End of (first) row
</TABLE>	End of table

This will appear as follows.

Cell 1	Cell 2

The following example has two rows of cells.

HTML	Comment
<TABLE BORDER=1>	Start of table with a 1 pixel width border
<TR>	Start of (first) row
<TD>	Start of (the first) cell (in the first row)
Cell 1	
</TD>	End of (the first) cell (in the first row)
<TD>	Start of (the second) cell (in the first row)
Cell 2	
</TD>	End of (the second) cell (in the first row)
</TR>	End of (first) row
<TR>	Start of (second) row
<TD>	Start of (first) cell (in the second row)
Cell 3	
</TD>	End of (the first) cell (in the second row)
<TD>	Start of (the second) cell (in the second row)
Cell 4	

</TD>	End of (the second) cell (in the second row)
</TR>	End of (the second) row (in the second row)
</TABLE>	End of table

This will appear as follows.

Cell 1	Cell 2
Cell 3	Cell 4

6.8. Producing an attractive fill-in form

Many web sites include pages that request basic information from the user with a form containing multiple fields. Aligning such a form can be awkward as it will look best if the instructions and boxes are aligned with each other and the boxes are aligned with each other. This can be achieved with tables better than with other HTML tags. We will see here how to align a set of form fields attractively in the centre of the screen.

Centring a table

In order to cause a table to be centred on the screen, enclose it in center tags as follows.

HTML	Comment
<CENTER>	Begin centring
<TABLE BORDER=1>	Start of table with a 1 pixel width border
<TR>	Start of (first) row
<TD>	Start of (first) cell
Cell 1	
</TD>	End of (first) cell
<TD>	Start of (second) cell
Cell 2	
</TD>	End of (second) cell
</TR>	End of (first) row
</TABLE>	End of table
</CENTER>	End centring

This will produce a centred table.

Cell 1	Cell 2

Adding form fields to a table

Form fields can be added to a table in the same way as anywhere else in the body of a HTML document. The start and end tag of the form, <FORM> and </FORM> can be placed outside the form itself if necessary, and the fields spread out throughout the table. Here is an example of a table with a simple form field.

HTML	Comment
<FORM>	Start of the form
<TABLE BORDER=1>	Start of table with a 1 pixel width border
<TR>	Start of (first) row

<TD>	Start of (first) cell
Enter your name here	
</TD>	End of (first) cell
<TD>	Start of (second) cell
<INPUT TYPE=TEXT	
NAME=UserName>	
</TD>	End of (second) cell
</TR>	End of (first) row
</TABLE>	End of table
</FORM>	End of the form

This will produce the following simple form in a table.

If there is more than one field to fill in then the instructions will look best when they are aligned to the middle of the table. This can be achieved by instructing the cells containing them to align them to the right hand side of the cell. This can be achieved by using the ALIGN=RIGHT option in the TD cell tag, in other words changing <TD> to <TD ALIGN=RIGHT> in the appropriate cells. Here is an example.

HTML	Comment
<FORM>	Start of the form
<TABLE BORDER=1>	Start of table with a 1 pixel width border
<TR>	Start of (first) row
<TD ALIGN=RIGHT>	Start of (first) cell
Enter your name here	
</TD>	End of (first) cell
<TD>	Start of (second) cell
<INPUT TYPE=TEXT NAME=UserName>	
</TD>	End of (second) cell
</TR>	End of (first) row
<TR>	Start of (first) row
<TD ALIGN=RIGHT>	Start of (first) cell
Enter your home address here	
</TD>	End of (first) cell
<TD>	Start of (second) cell
<INPUT TYPE=TEXT NAME=Address SIZE=30>	
</TD>	End of (second) cell
</TR>	End of (first) row
</TABLE>	End of table
</FORM>	End of the form

This form will now be aligned correctly.

Completing the form design

To finish off the design of a form, the borders of the table should be switched off by changing <TABLE BORDER=1> to <TABLE BORDER = 0> and the whole table should be centred as before. Here is the above example modified in this way.

HTML	Comment
<CENTER>	Begin centring
<FORM>	Start of the form
<TABLE BORDER=0>	Start of table without a border
<TR>	Start of (first) row
<TD ALIGN=RIGHT>	Start of (first) cell
Enter your name here	
</TD>	End of (first) cell
<TD>	Start of (second) cell
<INPUT TYPE=TEXT NAME=UserName>	
</TD>	End of (second) cell
</TR>	End of (first) row
<TR>	Start of (first) row
<TD ALIGN=RIGHT>	Start of (first) cell
Enter your home address here	
</TD>	End of (first) cell
<TD>	Start of (second) cell
<INPUT TYPE=TEXT NAME=Address SIZE=30>	
</TD>	End of (second) cell
</TR>	End of (first) row
</TABLE>	End of table
</FORM>	End of the form
</CENTER>	End centring

Here is the finished form, centred on the screen.

Activity
Add a third row to the above table for a textbox with instructions: “Enter your job title here”.

6.9. Summary

- The contents of form fields can be accessed using the document.form_name.field_name.value structure.
- The if statement can be used to perform simple checks on the data entered in form fields.
- The onSubmit keyword can be used in a submit button tag to process the data in the form with a JavaScript program before it is submitted.
- Returning false from an onSubmit section of code stops the data being submitted to the server.
- Tables can be used to align forms and can have their borders switched off to help the appearance of the page.

6.10. Exercises

3. Describe the actions of the following JavaScript programs when embedded in the body of a HTML file and loaded into a browser.

```
<HTML><BODY>
<FORM NAME="Name">
Please enter your name:
<INPUT TYPE=TEXT NAME="YourName" VALUE="Bill Gates"
SIZE=30>
</FORM>
<SCRIPT LANGUAGE=JavaScript>
alert("Enter name if not " +
                    document.Name.YourName.value);
</SCRIPT>
</BODY></HTML>
```

```
<HTML><BODY>
<FORM NAME="Phone">
Please enter the following details about yourself.<BR>
Name: <INPUT TYPE=TEXT NAME="YourName" SIZE=30>
Phone: <INPUT TYPE=TEXT NAME="YourNumber" SIZE=12
VALUE="01902321000">
</FORM>
<SCRIPT LANGUAGE=JavaScript>
alert("Enter phone number if not " +
               document.Phone.YourNumber.value);
</SCRIPT>
</BODY></HTML>
```

```
<HTML><BODY>
<FORM NAME=Cost onSubmit='
     var Price;
     Price = 44 * document.Cost.Height.value *
                    document.Cost.Width.value;
     alert("The price of that tree is £" + Price);
'>
Please enter the following details about the tree that
you would like to buy.<BR>
Height: <INPUT TYPE=TEXT NAME="Height" SIZE=4
          VALUE=100>
Trunk Width: <INPUT TYPE=TEXT NAME="Width" SIZE=2
VALUE=4>
<INPUT TYPE=SUBMIT VALUE="Calculate Price">
</FORM>
</BODY></HTML>
```

```
<HTML><BODY>
<FORM NAME=TestCost onSubmit='
```

```
     var Price;
     Price = 44 * document.TestCost.Height.value *
     document.TestCost.Width.value;
     if (Price > 100){
          alert("The tree is too expensive for you!");
     }else{
          alert("The price of that tree is £" + Price);
     }
return false;
'>
Please enter the following details about the tree that
you would like to buy. You have £100 to spend.<BR>
Height: <INPUT TYPE=TEXT NAME="Height" SIZE=4
          VALUE=100>
Trunk Width: <INPUT TYPE=TEXT NAME="Width" SIZE=2
VALUE=4>
<INPUT TYPE=SUBMIT VALUE="Calculate Price">
</FORM>
</BODY></HTML>
```

4. In the HTML page below, what is the correct JavaScript name for the name field and what is the correct name for the colour field?

```
<HTML><BODY>
<FORM NAME="Colours">
Please fill in this form.<BR>
Name: <INPUT TYPE=TEXT NAME="MyName" SIZE=30>
Favourite colour: <INPUT TYPE=TEXT NAME="Colour">
</FORM>
</BODY></HTML>
```

5. Assume that the variables a, b and c have been declared and have numerical values. Write "if" statements to do the following tasks.

   a) Show an alert box saying that a is bigger than b if it is, otherwise an alert box saying that it isn't.
   b) Write "You have enough money" to the browser if a + b >= c, otherwise write, "You do not have enough money".

4. Produce HTML for forms to perform the following tasks.
   a) A form for a person to enter their name, email address and employer, with a default employer of Wolverhampton University. Include an alert box to tell them to fill in employer if it is not Wolverhampton University by referencing the value of the employer form field in the same way as in the name and age example.
   b) Get the name of a customer and their address and credit card number and give an alert box message if their credit card number is not entered when they press the submit button.

5. Create a web page that contains a table that will look as follows. You will need to use a table with invisible borders to get the correct page layout with neatly aligned boxes.

Name

Age

Married ○ Yes ○ No

Submit form

Add some JavaScript to check that the user has filled in their name and age when they submit the form.

7. Web Applications

7.1. Overview

In this section we are going to learn how to construct and use a number of common spreadsheet-type applications in a web page. These will include calculating a bill for an online shop and calculating wages for employees. No new JavaScript features will be introduced, but most of the ones that we have already covered will be reinforced.

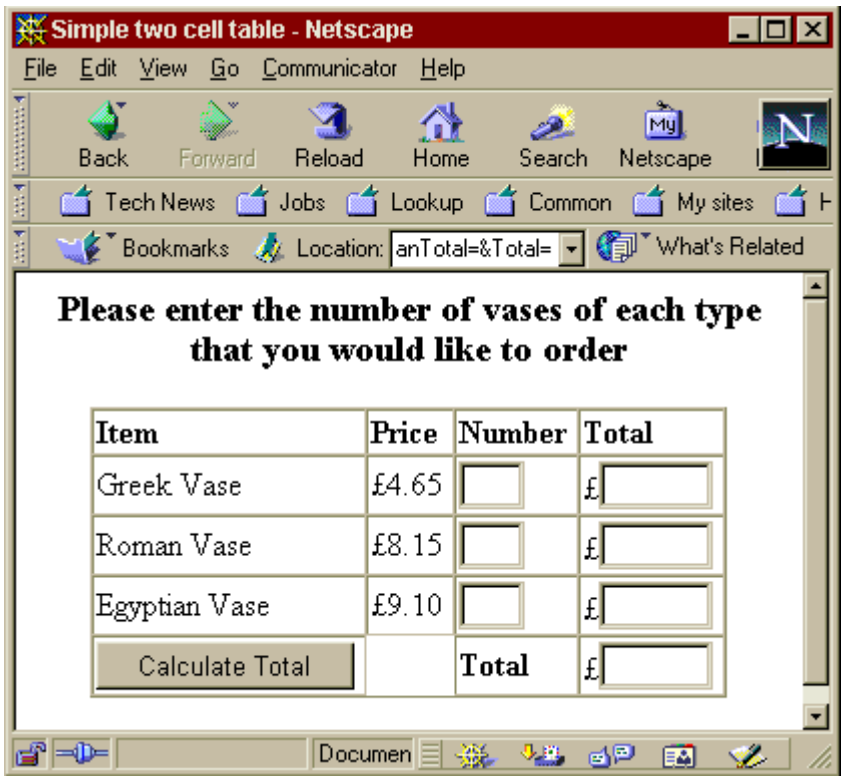
7.2. Objectives

In this section you will learn the following.

- How to automatically calculate and complete a purchases form in a web page.
- How to use the image rollover effect.

7.3. Calculating total order values

Many web pages contain an online order form. These are easy to create using a table and text boxes for customers to fill in the number of items required. Here is a typical form.



We are going to use some of the JavaScript that we have learned in order to get all of the totals filled in. This can be achieved with some simple arithmetic using the names of the boxes. In the above example the input boxes are called GreekNumber, GreekTotal, RomanNumber, RomanTotal, EgyptianNumber, EgyptianTotal and Total. The form is called Vases and so the statement to fill in the total in the second row is as follows.

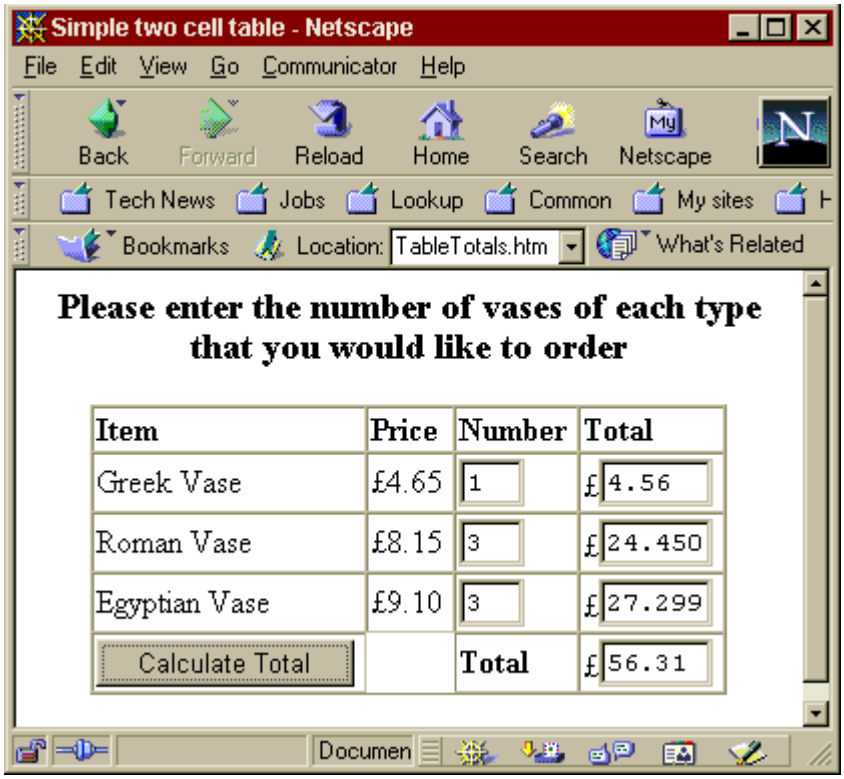
```
document.Vases.GreekTotal = 4.56 *
```

```
                                        document.Vases.GreekNumber;
```

This takes whatever number the user has entered into the third column of the table, `document.Vases.GreekNumber`, multiplies it by 4.56 and then puts the answer in the total cell, `document.Vases.GreekTotal`. This code should go in the onSubmit event of the form. To complete the total box in the bottom row of the table, the column must be added up. The JavaScript for this should be straightforward, but is not. The reason is that the addition operator '+' is the same as the string concatenation operator, and so in some browsers the numbers will not be added up, but treated as strings and concatenated instead. To get round this, use the parseFloat function to convert the value into a number. This will force the JavaScript interpreter to interpret their values as numbers. Here is the code.

```
<FORM NAME=Vases onSubmit='
      document.Vases.GreekTotal.value = 4.56 *
            document.Vases.GreekNumber.value;
      document.Vases.RomanTotal.value = 8.15 *
            document.Vases.RomanNumber.value;
      document.Vases.EgyptianTotal.value = 9.10 *
            document.Vases.EgyptianNumber.value;
      document.Vases.Total.value =
            parseFloat(document.Vases.GreekTotal.value)
          + parseFloat(document.Vases.RomanTotal.value)
          + parseFloat(document.Vases.EgyptianTotal.value);
      return false;
'>
```

Here is the code in action, with the submit button having just been pressed.



Notice that the Roman and Egyptian vase totals are annoyingly slightly incorrect. This is because the calculations have been executed in binary arithmetic, which rounds differently from decimal arithmetic. To avoid this problem, the answer can be multiplied by 100, rounded to a whole number and then divided by 100 again. The function that rounds to the nearest whole number is Math.round.

```
document.Vases.GreekTotal.value =
     Math.round(4.65 * document.Vases.GreekNumber.value
                     * 100) / 100;
```

For more complicated calculations, Math.round can be used separately from the main part by storing the answer in a variable and then rounding the value in the variable in the following statement, as illustrated below.

```
var Total = parseFloat(document.Vases.GreekTotal.value)
          + parseFloat(document.Vases.RomanTotal.value)
          +
parseFloat(document.Vases.EgyptianTotal.value);
document.Vases.Total.value =
                    Math.round(Total * 100) / 100;
```

Activity
Create the form above and add the rounding function to all *four* of the calculations.
Extra activity
If you are happy with using stylesheets and classes, create a stylesheet for your form with a special class for the end column of the table and colour its background yellow to indicate that the user should not fill it in.

Adding Local Taxes to the Total

Some types of companies normally quote prices without local taxes and only add them to the bill total. In the UK the local tax is Value Added Tax (VAT) and in the USA there are different sales taxes for each state. Other countries will have their own individual tax laws, but the tax to be applied is the one from the *selling* country, so this is not a problem.

If taxes are not included already in the prices then it is not difficult to add them to the total. An extra line should be added to the table to show the tax before it is added to the total in order to help the customer to understand where the calculation has come from. Taxes are normally expressed as a percentage of the order value, for example 4%, 5% or 17.5%. In order to apply these to the total, multiply the total by the percentage and then divide the answer by 100. For VAT at 17.5% this calculation is as follows.

$$VAT = total \times \frac{17.5}{100}$$

In the above example, we put the total directly into a cell, but this time we do not want to display the total, just the VAT, so we will modify the code to save the total as a variable and then work out the tax from it.

```
var Tax = document.Vases.Total.value * 17.5 / 100;
document.Vases.Tax.value
                    = Math.round(Tax * 100) / 100;
```

The first line calculates the tax on the total by multiplying it by the percentage rate and dividing the answer by 100, storing the answer in a variable named 'Tax'. The next line uses the Math.round trick that we have used once before to ensure that the answer is displayed correctly, storing the answer in a form field named `document.Vases.Tax.value`. Of course, in order for this to work, there must be a form field with the correct name. We must therefore add two rows to the table, one for the tax and a final row to add the subtotal and the tax. Here is the complete section of code for the form.

```
<FORM NAME=Vases onSubmit='
// Calculate and round the total for Greek vases.
document.Vases.GreekTotal.value =
 Math.round(4.65   *   document.Vases.GreekNumber.value   *
100)
                                                      / 100;
// Calculate and round the total for Roman vases.
document.Vases.RomanTotal.value =
 Math.round(8.15   *   document.Vases.RomanNumber.value   *
100)
                                                      / 100;
// Calculate and round the total for Egyptian vases.
document.Vases.EgyptianTotal.value =
 Math.round(9.10 * document.Vases.EgyptianNumber.value *
                                              100) / 100;

// Add and round the subtotal for all vases.
var                      Subtotal                        =
parseFloat(document.Vases.GreekTotal.value)
          + parseFloat(document.Vases.RomanTotal.value)
          +
parseFloat(document.Vases.EgyptianTotal.value);
document.Vases.Subtotal.value =
                         Math.round(Subtotal   *   100)   /
                    100;

// Calculate and round the Tax on the Subtotal.
var Tax = document.Vases.Subtotal.value * 17.5 / 100;
document.Vases.Tax.value = Math.round(Tax * 100) / 100;

// Calculate and round the overall Total.
var Total = parseFloat(document.Vases.Subtotal.value)
          + parseFloat(document.Vases.Tax.value);
document.Vases.Total.value  =  Math.round(Total  *  100)  /
100;

// Stop the form from being submitted.
return false;
```

```
'>
```

Activity
Modify the example above to add state tax at 3.4% (instead of VAT).

Postage and Packing

In addition to tax, a product to be sent through the post often incurs an additional charge to cover the expenses related to this transaction. The cost is often known as 'Postage and Packing' or 'Shipping and Handling'. These charges are more complicated to calcualate than tax because they are not normally a simple percentage, but carry some extra conditions, such as a minimum and a maximum. Some companies may also not charge for orders over a certain amount.

In order to deal with the various contingencies that arise in these calculations, a combination of arithmetic statements and 'if' statements will be needed. We shall go through an example to illustrate this. The code that we are going to generate is going to implement the following charge rate.

> Orders will be charged at 5% of the total order value, with a minimum charge of £10, a maximum charge of £100. Orders with a total value of over £10,000 will be post-free.

The calculation for 5% of the total value is a percentage calculation, similar to that for tax above. The answer will be stored in a variable, Postage, because it will need further processing before being released to the customer by being displayed in a form field.

```
// Calculate and round the basic Postage and Packing
rate.
var Postage = document.Vases.Total.value * 5 / 100;
  Postage = Math.round(Postage * 100) / 100;
```

A statement is now needed that will increase this to £10 if it is less than £10. This is a simple use of the 'if' statement.

```
// Ensure that the Postage is not below the minimum.
if (Postage < 10) {
     Postage = 10;
}
```

A statement is now needed that will cap the postage at £100 if it is more than £100. This is another simple use of the 'if' statement.

```
// Ensure that the Postage is not above the maximum.
if (Postage > 100) {
     Postage = 100;
}
```

A statement is now needed that will set the postage to zero if the total order comes to more than £10,000.

```
// Set Postage to £0 to if the order value is over
£10,000.
if (document.Vases.Total.value > 10000) {
     Postage = 0;
}
```

The result of all these calculations can now be added to the form, using another two rows at the bottom of the table to accommodate the answers.

```
// Show the Postage in a table cell.
document.Vases.Postage.value = Postage

// Create a new overall total including the Postage.
var                    OverallTotal                        =
parseFloat(document.Vases.Postage.value)
               +  parseFloat(document.Vases.Total.value;
document.Vases.OverallTotal.value =
     Math.round(OverallTotal * 100) / 100;
```

For the above code to work, there must be two new input boxes, called OverallTotal and Postage, in two new rows at the bottom of the table.

Activity
Write the JavaScript to change the above postage and packing calculation to the one described below. Make sure that you test your program by comparing it with a range of values that you have calculated by hand.
The postage and packing charge is 10% of the order value, with a minimum charge of £5, and a maximum charge of £40.

7.4. The Image Rollover

The image rollover is a special effect on a web page when one image is changed for another when the mouse is over it. Often the two images are very similar but the second one is enhanced or raised in some way in order to indicate that the spot is clickable. It is not difficult to write JavaScript to swap images over, in fact it can be done with one line of code, but three separate things are needed.

- Give the image a name.
- Use the onMouseOver event
- Use the .src property of the image.

The image to be changed should be referenced in the HTML in the normal way, using the <IMG SRC= …> syntax, but an extra argument for a name of the image must be included. This is so that it can be referenced by JavaScript statements. The actual statement to be used is as follows.

```
document.image_name.src = "new_image.gif"
```

Here image_name is the HTML name of the old image but new_image.gif is the file name of the new image. In order to ensure that the image changing occurs only when the mouse is in the right place, it must go inside the image tag and must be labelled

with onMouseOver. This will ensure that it is triggered at the right time. Here is how it all fits together.

```
<HTML>
<BODY>
<A HREF="http://www.wlv.ac.uk",
 onMouseOver='document.flashingImage.src="IMAGE2.GIF"'>
<IMG NAME=flashingImage SRC="IMAGE1.GIF" BORDER=0></A>
</BODY>
</HTML>
```

Notice that two different types of quotes are used above, double quotes and single quotes. In JavaScript, whenever quotes are needed in the program when the program itself is in quotes, a different kind of quotes should be used or an error message will be generated.

An extra section of code is now needed to change the image back to its original version. This is very similar to the first one but must be labelled onMouseOut in order to be triggered by the mouse no longer being over the image. Here is the code.

```
<HTML>
<BODY>
<A HREF="http://www.wlv.ac.uk",
 onMouseOver='document.flashingImage.src="IMAGE2.GIF"',
 onMouseOut='document.flashingImage.src="IMAGE1.GIF"'>
<IMG NAME=flashingImage SRC="IMAGE1.GIF" BORDER=0></A>
</BODY>
</HTML>
```

> Tip – Use a painting program to lighten the colours on one image to produce a new version that seems to be lit up. This will create a good effect when the images are swapped.

This technique works very well when it is used in the right browser and with the images stored on the computer's hard drive, but it needs some altering before it is used on the Internet. There are two problems: the code will cause an error in Internet Explorer 3 and some other browsers, and the image swap will be slow when the new file has to be downloaded from the Internet. The first problem does not have a perfect solution because some old browsers used an early version of JavaScript that did not support image swapping. As a damage limitation exercise, however, the program needs to be designed so that it does not cause an error when it does not work. This is easy to achieve because the JavaScript 'object' that enables the image swapping is called document.images and so this can be used in an if statement to make execution of the image swap conditional on it being possible, as follows.

```
<HTML>
<BODY>
<A HREF="http://www.wlv.ac.uk",
 onMouseOver='
     if (document.images) {
```

```
          /*The browser supports images */
          document.flashingImage.src="IMAGE2.GIF"
     }',
 onMouseOut='
     if (document.images) {
          /*The browser supports images */
          document.flashingImage.src="IMAGE1.GIF"
     }'
>
<IMG NAME=flashingImage SRC="IMAGE1.GIF" BORDER=0></A>
</BODY>
</HTML>
```

In the above code the image swap statements are only executed if the object document.images exists, otherwise nothing happens.

It is more difficult to get round the problem of an image needing to be downloaded first before it can be used to replace another one. In order to do this, an extra section of JavaScript code must be placed in the head of the web page. This code must create variables to store the images and then download them. This ensures that when they are called for, the images have already been downloaded. Here is the code.

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE=JavaScript>
<!-- COMMENTING OUT FOR OLD BROWSERS
var FirstImage = new Image(12,44);
     FirstImage.src = "IMAGE1.GIF";
var SecondImage = new Image(12,44);
     SecondImage.src = "IMAGE2.GIF";

<!-- COMMENTING OUT FOR OLD BROWSERS -->
</SCRIPT>
</HEAD>
<BODY>
<A HREF="http://www.wlv.ac.uk",
 onMouseOver='
     if (document.images) {
          /*The browser supports images */
          document.flashingImage.src= SecondImage.src
     }',
 onMouseOut='
     if (document.images) {
          /*The browser supports images */
          document.flashingImage.src= FirstImage.src
     }'
>
<IMG NAME=flashingImage SRC="IMAGE1.GIF" BORDER=0></A>
</BODY>
</HTML>
```

The code works by creating a variable in which to store the image and then allocating an image to the variable. The variable is of a new type: an image object. When it is declared the new keyword is needed because it is an object variable, and the dimensions of the image (height and width in pixels) need to be specified so that the browser can allocate it the correct amount of storage space. When the src is set on an image variable the browser will download the image and store it. When the image is needed it will then be almost instantly available. You might not be able to tell the difference between the speed of this example and the previous one if the images are on the computer that you are using, but it will make a difference when they are published on the Internet.

> Tip – To find the dimensions of an image in a web page, view the information available about the page in the browser and select the image. The information given should include its dimensions in pixels.

1. In order to create an image rollover effect the following steps need to be taken: -

2. Obtain two images with the same dimensions that you want to swap over.
3. Find out the width and height of the images. Your image program should tell you this.
4. In the Head of the document include JavaScript to load both versions of each image.
5. In the body of the document include code to swap the images, labelled with onMouseOver and onMouseOut.
6. Protect all references to changed images with if (document images) {} conditional statements.

7.5. Summary

- Totals for purchases in a web page can be automatically tallied with some simple JavaScript statements. Additional code can be used to add postage and packing costs and VAT.
- A short section of code is all that is needed to perform an image rollover, but extra code is needed to make it smooth and error-free.

7.6. Exercises

6. Create an automatically completed form to allow orders for the following stock offered by Cynthia's Garden Supplies.

Elephant fertiliser	£4.50 a bag
Horse manure	£2.10 a bag
Hay	£1.15 a bag

7. Enhance the example above by including a delivery charge of £5 for orders of value up to £20 but free for orders over £20.
8. Create an image rollover in a web page where the image is a picture of a house that lights up when the mouse is over it.
9. Add some image rollover effects to your home page. Consider creating a 'navigation bar' that you can copy wholesale from one page to another.

8. Troubleshooting

This section is simply a list of suggestions to try if your programs do not work.

The program does not work at all.

- Check the cAsE of everything that you type. JavaScript is case sensitive and so on accidental wrong case can cause either a wrong answer to be calculated or the whole program to crash. A likely culprit is the word 'var'. Typing this as 'Var' may stop the entire program from executing. Check carefully that all JavaScript words are the correct case. In particular, check the following: var; if; for; document; Math.round; value; function; return; write.
- Check that you have saved the page in the editor after you have last changed it and then reloaded the page in the browser after the last save in the editor.
- In Netscape, a description of any errors can be found after loading the page by typing JavaScript: in the location bar.

I can't see my JavaScript when I use 'View Source'

If you have used document.write then the browser may choose to show you the result of these statements rather than the statements themselves. You will need to load the original file into an editor such as Notepad in order to see the HTML.

What is <BASE HREF=...> doing in the source of my web page?

If you have used document.write then the browser may choose to show you the result of these statements rather than the statements themselves. It may additionally put the BASE tag at the start of the document to remind itself where the document came from.

My form/half my web page has gone!

Check that you have used a <FORM> tag at the start of the form and that any quotes anywhere in the page matching, in other words that you have not forgotten the second set of quotes anywhere. If you view source in Netscape it sometimes flashes the section of text where an error has been noticed - this can be a great help.

I am using stylesheets, but my styles overflow onto the next elements

Check that you have used an end of section tag for all the elements to which you have supplied a style. For example, it normally does not matter if you do not end a paragraph with </P> but when you are using stylesheets it can make a difference.

When I click on the submit button, all my fields go blank!

This is normal. It can be cancelled by using the onSubmit event (explained in this course) and including the following line in it. This cancels the submission of the form and stops the page being reloaded.

```
return false;
```

What is NaN doing in my form fields?

NaN stands for 'Not a Number' and means that a JavaScript program has attempted to do a numerical calculation on some user input that is not a number. This might happen, for example, if the user enters £1.20 instead of 1.20 because the £ may cause JavaScript to ignore the following number.