Chapter 3. HTML: BASICS

Table of Contents

Objectives

At the end of this chapter you will be able to:

- Create HTML files using Notepad and run them on a Tomcat server;
- Use HTML tags to write HTML files;
- Format HTML files;

- Create lists in HTML files;
- Use anchors in HTML files;
- Use multimedia in HTML files.

3.1 Basics

3.1.1 HTML Markup

HTML pages are created by tagging textual information with HTML markup. HTML markup consists of tags, which appear inside angled brackets < and >

An example of an HTML tag is <B>, which causes text to appear in bold. <B> only notes where text should begin to appear in bold, while the tag </B> marks the end of the emboldening. Most HTML tags have a corresponding end tag, which is specified by the name of the tag preceded by the / character.

So, to create the text:

Internet Commerce is great!

The text is marked up as:

```
<B>Internet Commerce is great!</B>
```

Another example of an HTML tag is <I>, which causes text to appear in italic. In HTML 4.01, the <I> tag was used to render text in italics. However, this is not necessarily the case with HTML5. Style sheets can be used to format the text inside the <I> element. This will be demonstrated later.

Note that tags are not case-sensitive. In other words, <B> or <b> are the same tag, both specifying bold text.

3.1.2 Nesting HTML Tags

Text may be both bold and italicised. This is done by using both the <B> and <I> tags. When doing so, it is important to remember not to overlap HTML tags. In other words:

```
<B><I>Internet Commerce
     is great!</I></B>
```

is **correct**, but

```
<B><I>Internet Commerce is great!</B></I>
```

is **wrong**.

Overlapping tags is a common mistake. Although Web browsers are usually smart enough to work out what is meant, it can lead to problems. Furthermore, for an HTML page to be considered valid HTML, it must contain no overlapping tags.

To Do:

Read the section on "HTML Tags" in your textbooks.

3.1.3 Creating HTML Text using Notepad++

This section covers the creation of an HTML page. You will need a Web browser and a text editor. Use

any text editor you wish to, but the following Activity descriptions will use Notepad++. Notepad++ is a free Windows editor that also supports several programming languages. For example, you will notice that HTML keywords are highlighted in different colors.

1. Open your Web browser. This sections' goal is to create a Web document that can be opened with your browser.
2. Open Notepad++. It can be found by selecting Start, then All Programs, then Notepad++.
3. Type the following text into Notepad++: your name and the module number (CSC5003). Save this file as start.txt.
4. Now load start.txt into the browser by dragging start.txt onto your browser.
5. The browser should now display the text contained in **start.txt**. (If it does not, make sure that you have saved **start.txt** and that this is the file you are opening).
6. Once you have displayed start.txt, return to Notepad. Add the text "Internet Commerce", and save the file again.
7. Return to the Web browser and reload the document (by using either by using the Refresh or Reload toolbar buttons, or by selecting File/Open once again).
8. If you are able to see the new piece of text, you have successfully used Notepad to create your first Web page.

Activity 1: Getting started with HTML

This Activity adds HTML tags to start.txt.

1. Open your file **start.txt** in Notepad.
2. Mark up the text "Internet Commerce" so that appears in bold. Do this by placing the <B> tag in front of the text, and </B> at the end of the text, as shown below:
   ```
   <B>Internet Commerce</B>
   ```
3. Save the file as **start.html**, since it contains some HTML formatting. Save the file with this new name (using Save As). Note that saving it as **start.htm** is also accepted. Other than the obvious, the letter "L," there's not much of a difference between the two extensions. Most, if not all, web browsers and servers will treat a file with an HTM extension exactly as it would a file with an HTML extension, and vice versa[1].
4. Load **start.html** in the Web browser. **Internet Commerce** should now appear in bold.
5. Return to Notepad and add more text, some of it in bold and others in italics. (Remember <I> is the tag for italics) Save the document and reload it.

3.1.4 Standard HTML Document Structure Format

Although a number of HTML tags have been introduced that markup how text should be displayed in a browser, a correct HTML document must always include certain structural tags. These tags are<HTML>, <HEAD>, <BODY> and <TITLE>.

The <HTML> tag should be placed around the document's contents; this tells the browser that the whole document is written in HTML. Like a person, all HTML documents have only one head and one body. All the text of the HTML document should be inside either the head or the body. Roughly, the <HEAD> holds information about the document itself, and the <BODY> holds the information that should be displayed. The document's <TITLE> is given in the <HEAD>. The title is shown at the very top of the browser (i.e. in the title bar) — not in the browser window itself.

The standard structure of an HTML document is:

```
<HTML>
<HEAD>
<TITLE>Text to appear in the title bar of the browser</TITLE>
</HEAD>
<BODY>
```

[1] http://www.sightspecific.com/~mosh/www_faq/ext.html

```
        The text to appear in the main browser window.
</BODY>
</HTML>
```

This format should always be used when writing HTML documents.

Note: students are often confused about the use of the <BODY> tag, and they often include multiple body tags. This can lead to problems later on, so make sure to use only one <BODY> tag.

To Do: Read the section on HTML document structure in your textbooks.

Activity 2: Structuring your HTML document

In this Activity you will convert your file that contains a few HTML tags into a correctly structured HTML document. Open start.htm in Notepad.

1. Add the <HTML> tag on the first line of the file (before anything else).
2. Add the </HTML> end tag on the last line of the file (after everything else).
3. Add the document header by adding a <HEAD> tag on the line underneath the <HTML> tag and the </HEAD> tag on the line beneath that.
4. Between the opening and closing <HEAD> tags, add the <TITLE> and </TITLE> tags.
5. Enter the text "My first Web page" between the <TITLE> tags.
6. Underneath the </HEAD> tag, create the body of the document by entering the <BODY> tag.
7. At the bottom of the document, add the </BODY> tag just before the </HTML> tag.
8. Save the file.

If you have problems correctly formatting the file, look at the code in 3.1.4.

You are probably thinking that it looks the same as the previous document. However, if you look closely at the title bar you should see that it now displays the words "My first Web page". The main difference, however, is that the browser now has to do a lot less work to do, since the document informs it of the HTML's structure.

Activity 3: Loading your HTML file on Tomcat

The previous chapter guided you through tomcat installation. Let us launch the start.html file using the tomcat webserver. Make sure that your tomcat server has been started. Save start.html in the folder myapps that you created within the webapps folder. Load **start.html** in your browser by typing `http://localhost:8888/myapps/start.html`

3.2 HTML Formatting

3.2.1 The Browser As Formatter

As you will recall, it is the browser that actually formats the HTML document. But when it displays text, where does it put the line breaks?

The browser automatically determines the position of the line breaks. It tries to fit all of the text into the current window so that the user does not need to do any horizontal scrolling. If the browser window changes size, the browser reformats the document.

It also ignores extra spaces. If there are two spaces between words in the HTML file, the browser will display the text in exactly the same way as if there was only one. Blank lines are ignored in a similar way. The browser also tries to correct errors in incorrect HTML (such as HTML containing overlapping tags). When doing so, the browser may incorrectly interpret the HTML document, making it a wiser choice to write correct HTML. Sometimes it can be difficult to have the browser format things as you want. You will learn more tricks on how to do this as you work through the HTML units.

3.2.2 Paragraphs, Line Breaks and Preformatting

The tag
 is used to start a new line.
is a standalone tag, that means there is no closing </BR>tag. Note that
 does not place a line space between the two lines. To do that you need to use the <P> paragraph tag. Do not forget to add the end tag </p> although most browsers will display HTML correctly even if you forget the end tag. The tag <pre> defines preformatted text. The text inside a <pre> element is displayed in a fixed-width font (usually Courier), and it preserves both spaces and line breaks.

Activity 4: Paragraphs, Line Breaks and Preformatting

In this Activity you will use the <P> and
 tags to create line breaks in text. We will also demonstrate the use of <pre>.

1. Load Notepad and begin a new HTML document.
2. Enter the usual structural HTML tags. Set the title to "Formatting text".
3. Within the body type in the following text exactly as it appears below. Not how 'This is cool' has been typed. Do not use any HTML tags to format it at this stage.

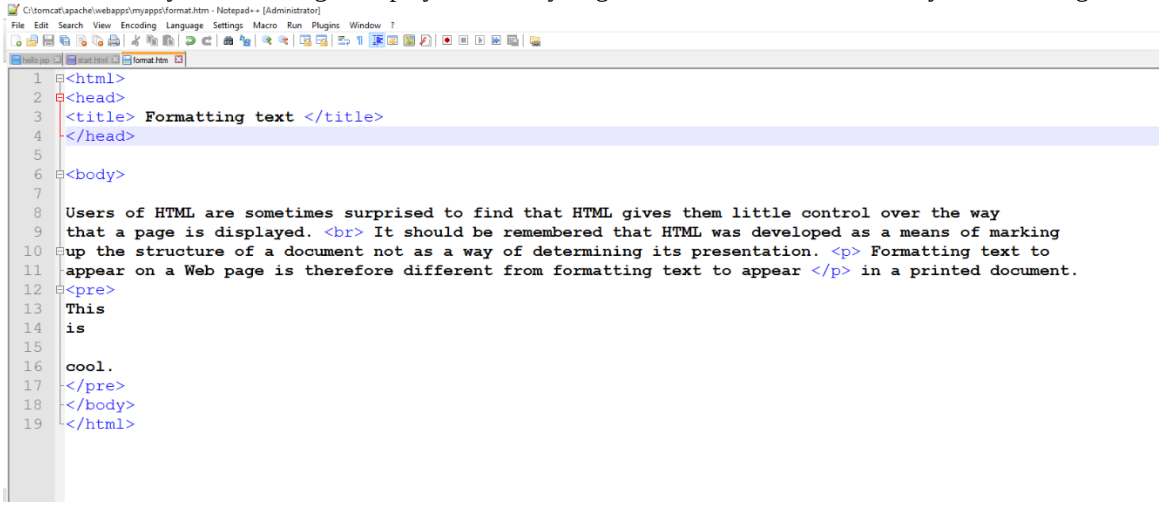
   Users of HTML are sometimes surprised to find that HTML gives them little control over the way that a page is displayed. It should be remembered that HTML was developed as a means of marking up the structure of a document not as a way of determining its presentation. Formatting text to appear on a Web page is therefore different from formatting text to appear in a printed document.
   This

   is
   Cool.
4. Save the document as **format.html** in your myapps folder and load it in your browser to view it. Note that 'This is cool' is displayed without the line breaks.
5. Resize your browser and watch how the text is reformatted to fit in the resized browser window.
6. Return to Notepad and make the changes as shown in Figure 3.1.
7. Save the file again and load it in your browser to check your HTML. Resize the browser and watch how the document is reformatted for the resized window.

3.2.3 Headings, Horizontal Rules and Meta Tags

The DOCTYPE declaration defines the document type to be HTML. In HTML5 this is written as `<! DOCTYPE html>.` The `<!DOCTYPE>` declaration helps the browser to display a web page correctly. There are different document types on the web. To display a document correctly, the browser must know both type and version. The doctype declaration is not case sensitive. All cases are acceptable:

Another set of HTML tags are the headings tags. These are <H1>, <H2>, <H3>, <H4>, <H5> and <H6>. The text surrounded by the <H1> tag is displayed in a very large font size. Text surrounded by the <H2> tag is



```
1  <html>
2  <head>
3  <title> Formatting text </title>
4  </head>
5
6  <body>
7
8  Users of HTML are sometimes surprised to find that HTML gives them little control over the way
9  that a page is displayed. <br> It should be remembered that HTML was developed as a means of marking
10 up the structure of a document not as a way of determining its presentation. <p> Formatting text to
11 appear on a Web page is therefore different from formatting text to appear </p> in a printed document.
12 <pre>
13 This
14 is
15
16 cool.
17 </pre>
18 </body>
19 </html>
```

Figure 3.1: Tags for paragraph, line breaks and preformatting

displayed in a slightly smaller font size, and so on down to the <H6> heading tag. You can use these tags to provide your page with a standard outline format. For example, the page heading might be displayed using the <H1> tag, a section heading using <H2> and a sub-section heading using <H3> and so on. Use HTML headings for headings only. Don't use headings to make text BIG or **bold**. Search engines use your headings to index the structure and content of your web pages. It is important to use headings to show the document structure. Browsers automatically add some empty space (a margin) before and after each heading.

Earlier we noted that Web browsers are HTML readers. Each browser is free to interpret HTML any way it likes. Consequently, a document read in one browser might look a little different to one read in another browser. Although the HTML standard states that <H1> tags should be as big as or bigger than <H2> tags, and <H2> tags should be as big as or bigger than <H3> tags and so on, one browser might display the <H3> tag with the same font size as the <H2> tag, while another browser might display it in a smaller font size. Hence the difference in displaying the same text. In practice, these implementation questions will become an issue when you are using more complex tags. For now you can ignore this problem while writing HTML.

The **<hr>** tag creates a horizontal line in an HTML page. The <hr> element can be used to separate content.

The HTML **<meta>** element is also meta data. It can be used to define the character set, and other information about the HTML document. Other meta elements that can be used are <style> and <link>.

Activity 5: Headings

In this Activity you will set up a page heading and sub-heading for the Web page begun in Activity 5 and use the HTML headings tags to implement it.

1. Load format.htm in MS-Notepad.
2. Within the <head> tags, add `<meta charset="UTF-8">`. It does not matter whether it is below or after the <title> tag.
3. Set up the page heading "Formatting text" and place the <H1> heading tags around it, in other words, <H1>Formatting text</H1>.
4. Reload format.html in your browser. You will notice that the effect of the <H1> tag is to display the text not only in an enlarged font size but also to include extra space above and below it. So you do not need a
 or <P> tag as well.
5. Return to Notepad and use the <H2> tag to create a sub-heading for the page, "Paragraphs and line breaks".
6. Add <hr> between ‘This’ and ‘is’.
7. Reload the document in your browser to check the HTML and you should have an output like in Figure 3.2.

HTML Tip - How to View HTML Source

Have you ever seen a Web page and wondered "Hey! How did they do that?" To find out, right-click in the page and select "View Page Source" (in Chrome) or "View Source" (in IE), or similar in another browser. This will open a window containing the HTML code of the page.

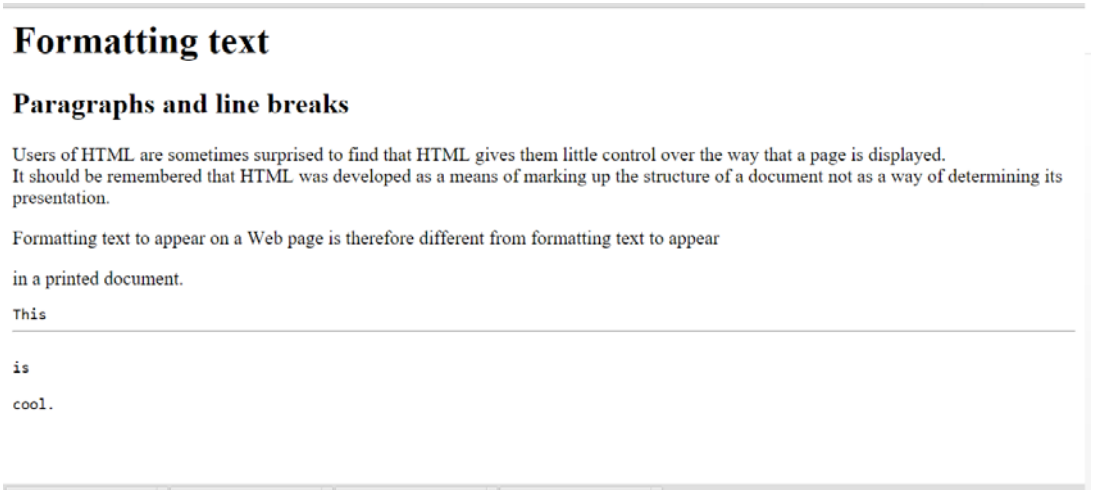


Figure 3.2: Using headings, horizontal rules and meta tags

3.3 Lists

- Apple
- Oranges
- Bananas

1. Apples
2. Oranges
3. Bananas

The two examples above are lists. The list on the left uses bullets to mark the list elements, and is known as an **unordered list**. The list on the right uses numbers to mark the list elements and is known as an **ordered list.**

HTML lists consist of a list tag and list element tags.

In an **unordered list**, the list tag is <UL> and the list element tag is <LI>. Note that although the list element end tag </LI> was optional in previous versions of HTML, it no longer is. The list end tag </UL> is also not optional.

To create an **unordered list** as in the above example, use the following HTML.

```
<UL>
<LI>Apples</LI>
<LI>Oranges</LI>
<LI>Bananas</LI>
</UL>
```

Note that it is useful to indent the <LI> tags on the page to keep track of the level of indentation. To add more list elements, add extra list element tags <LI> </LI> containing the elements within the <UL> tags.

A **style** attribute can be added to an unordered list, to define the style of the marker:

Style	**Description**
list-style-type:disc	The list items will be marked with bullets (default)
list-style-type:circle	The list items will be marked with circles
list-style-type:square	The list items will be marked with squares
list-style-type:none	The list items will not be marked

For example the code below would append square bullets to the list.

```
<UL style = "list-style-type:square">
<LI>Apples</LI>
<LI>Oranges</LI>
<LI>Bananas</LI>
</UL>
```

Ordered lists are specified almost exactly the same as unordered lists, only the <OL> tag is used instead of the <UL> tag. A **type** attribute can be added to an ordered list, to define the type of the marker:

Type	Description
type="1"	The list items will be numbered with numbers (default)
type="A"	The list items will be numbered with uppercase letters
type="a"	The list items will be numbered with lowercase letters
type="I"	The list items will be numbered with uppercase roman numbers
type="i"	The list items will be numbered with lowercase roman numbers

For example,

```
<OL type = "i">
<LI>Apples</LI>
<LI>Oranges</LI>
<LI>Bananas</LI>
</OL>
```

The **description** list, is different: it has neither bullets nor numbers. The description list tag is <DL> and the list elements consist of a term and its definition. The term is marked by <DT> tags and the definition by <DD> tags. An example use definition lists is the glossary definition that appears below.

```
<DL>
<DT>HTML </DT>
<DD> Hypertext Markup Language; the format of Web documents </DD>
</DL>
```

Lists can be nested (lists inside lists). For example,

```
<OL type = "i">
<LI>Apples</LI>
<LI>Oranges</LI>
<LI>Bananas</LI>
      <ul>
      <li> small bananas </li>
      <li> big bananas </li>
      </ul>
</OL>
```

Activity 6: Lists

In this Activity you will create a series of lists to practise your HTML list-building skills.

1. Load format.html in Notepad.
2. Underneath the text, create three lists as follows:
    a. List one should be a circled bulleted (i.e. unordered) list, using square bullets, giving the days of the week.
    b. List two should be a numbered list of the months of the year. Make the numbers lowercase roman numerals.

   c. List three should be a definition list of the four seasons.
3. Save the file and view it in your Web browser to ensure that it displays as desired.
4. Reload format.html in Notepad and create a new bulleted list showing the four seasons. Within each season create a numbered sub list of the appropriate months of the year.
5. Save the file and load it in your Web browser to examine the docum

3.4 HTML Comments

Notes may be left in an HTML page to, for example, explain how or why something was done; these notes are often useful for other developers who will be working on the HTML document. These notes, called **comments**, are not displayed by the Web browser, and can only be seen in the HTML source file itself. Web developers make frequent use of HTML comments, and they can found in many Web pages (using the 'View Page Source' menu option).

An HTML comment is given as: <!--- text in the comment ---> .

Activity 7: Comments

In this Activity you will use preformatted text and HTML comments.

1. Load format.html in Notepad.
2. Place an HTML comment before the lists you created.
3. Save the file and load it in a Web browser.

3.5 Anchors

To link to another file use the <A HREF="URL">link</A> tag.

The term URL is the location of the file to be linked to. It could be on a hard or floppy disk — as in a:\filename.html or c:\my documents\week01\filename.html — on the same Web server, or on another Web server, as in http://www.fortunecity.com/username/filename.html

Activity 8: Simple hypertext links

In this Activity you will create four new Web pages: index.html, filetwo.html, filethree.html and filefour.html. You will then link them together using relative URLs.

1. Open Notepad and type in the HTML code shown below. (You may find it easier to cut and paste the code from your Web browser into Notepad rather than enter it yourself.)

```
<HTML>
<HEAD>
<TITLE>File name</TITLE>
</HEAD>
<BODY>
<h2>File name</h2>
<p>
Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam
nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam
erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci
tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo
consequat.
<P>
Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse
molestie consequat, vel illum dolore eu feugiat nulla facilisis at
vero
eros et accumsan et iusto odio dignissim qui blandit praesent
```

```
luptatum zzril delenit augue duis dolore te feugait nulla facilisi.
<P>
<A HREF="index.html">Homepage</A><BR>
<A HREF="filetwo.html">Filetwo</A> <BR>
<A HREF="filethree.html">Filethree</A> <BR>
<A HREF="filefour">Filefour</A> <BR></BODY>
</HTML>
```

2. Save this file as **index.htm**. Save the file a further three times using the file names from the list above. Each time also revise file name in the HTML title and body.
3. You should now have four unique files that link to each other. Test these files in your browser by first opening index.html.
4. Now add the following in index.htm to create a hyperlink to the University of Cape Town website.

```
<P>
<A HREF="http://www.uct.ac.za">University of Cape Town</A>
```

5. Save the file and test it in your browser.

3.5.1 Linking to Email Addresses & other Non-Web Links

The previous examples have used the HTTP protocol to inform the browser to load a Web page when you click a hyperlink. Various other protocols may be used. For example, to create a link to an email address use the 'mailto' protocol. Note that this depends on the user's email programme being correctly configured, and so may not always work. However, this feature is commonly used on the Web to contact the webmaster or to get more information from sites.

The following anchor tag creates a mail link:

```
<A HREF="mailto:username@domainname">Email user</A>
```

You can test this by providing your own email address. This was tested and works for a Gmail address.

A subject line for the email message can also be provided:

```
<A HREF="mailto:username@address.com?SUBJECT=e-mail from a friend">user</A>
```

Other protocols that can be used include: ftp://, news://, telnet:// and gopher://. These protocols often require other software besides the Web browser, and, as with emails, if the software is not correctly installed and configured the links will not work.

3.5.2 Linking to Sections within Documents

Anchor tags can be used to link to a specific location within an HTML file (even within the same HTML file).

Firstly, a location must be defined using the tag: <A NAME="xxxx"></A> with xxx being the location name. A link to the location is created using the tag <A HREF="#xxx">link</A>. If the location is in another document, its file file name too must be included as well: <A HREF="URL#xxx">link</A>

Activity 9 - Linking to sections within a document

This Activity sets up links to sections within the documents created by Activity 8

1. Open Notepad and load **index.html**.
2. Copy the following text twice into the body of the file above the hyperlinks in order to create a long file. Rename section two to section three when copying it for the second time.

```
section two
<P>
Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam
nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam
```

```
erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci
tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo
consequat..
<P>
Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse
molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero
eros et accumsan et iusto odio dignissim qui blandit praesent
luptatum zzril delenit augue duis dolore te feugait nulla facilisi.
<P>
```

3. Now define the location section two by amending the text in the following way.

```
<A NAME="section_two">section two</A>
```

4. Define the location section three in the same way. Save the file.
5. Save this file as filetwo.html, overwriting the previous file.
6. Re-open index.html and add the following hyperlinks to the top of the <body> section:
    - Section Two
    - Section Three
    - File Two: Section Two
    - File Two: Section Three

7. Ensure that the links work by reloading index.htm in your Web browser.

3.5.3 Targeting Windows

In modern Web browsers, anchor tags can specify target windows using the target window attribute.

```
<A HREF="URL" TARGET="New_Window"></A>
```

This specifies where the contents of a selected hyperlink should be displayed. It is commonly used with frames or multiple browser windows, allowing the link to be opened in a specified frame or a new browser window. Windows may have names defined for them, but the underscore should not be used for the first character of any target defined in your documents. Such names are reserved for four special target names:

_blank	The browser always loads a target="_blank" linked document in a newly opened, unnamed window.
_self	This target value is the default for all <A> tags that do not specify a target. It causes the target document to be loaded and displayed in the same frame window as the source document.
_parent	This one is useful for framed sites to create navigation links back to the parent window.
_top	_top forces a break out of a framed site, or to take over the browser window. That means, for example, that if a site has be linked to from within a framed site, clicking on the link only brings up the site inside the frame. The _top target attribute forces the link to take over the entire browser window.

Activity 10 - Targeting windows

This Activity allows you to try the different target attributes. You will see how they behave by adding them to one or more of the files set up by Activities 9 and 10. For instance, create a link that loads the UCT site in a new browser window.

3.5.4 Link Appearance

It is possible to change the colour of a hyperlink to match the colour scheme of the Web page. The underlining of a hyperlink can also be modified or removed.

To Do
Read about affecting the appearance of links in your textbooks.

Activity 11: Changing the appearance of links
In this Activity you will set up a colour scheme for a Web page by using the BGCOLOR, TEXT, LINK, VLINK and ALINK attributes of the BODY tag.

1. Open Notepad and load **index.html**. Create the following colour schemes for the page using either hexadecimal values or colour names (see your textbooks):
    a. white background, black text and red hyperlinks.
    b. black background, white text and cyan hyperlinks.
    c. your own colour scheme.
2. View the pages in the browser.

3.6 Multimedia

3.6.1 Graphics
In-line images give the Web much of its visual appeal. They can also cause people using a slow Internet connection to not visit the site; so unless large in-line images are vital, they should either be avoided or alternate pages not using them provided. Small icons, however, have almost negligible transfer cost and can greatly enhance the appearance of your pages.

Use GIF or JPG files for graphics as appropriate:

- Scanned images look better as JPG files. The JPG file format uses variable compression to make the file size smaller, but too much compression causes the picture to lose detail. As a guideline, use a JPG compression level between 60 and 75 percent. JPG is always 24 bit colour or 8 bit grey scale (such as in a black and white photograph).
- GIF is useful for special effects such as transparent artwork (background transparency) and animation. It is 8 bit.

Another format called PNG combines the features JPG and GIF.

The <IMG> tag is used to embed an image into a Web document:

```
<IMG SRC="URL" ALT="Alternate Text">
```

The URL is the location of the image file. The ALT attribute specifies text to be displayed if the browser does not display the image.

In-line images are often placed inside anchors. The HTML code to create an image as a hyperlink is:

```
<A HREF="URL"><IMG SRC="URL" ALT="text"></A>
```

When an in-line image is used as a hyperlink, a border is placed around the image. This can be removed using **BORDER=0**. Similarly, a border, of any size, can be placed around a graphic using **BORDER=x**, where **x** is the width of the border in pixels.

Make use of the WIDTH and HEIGHT attributes in the IMG tag to specify the width and height of the image. This allows the browser to layout the page before all the graphics have finished downloading. The width and height can be specified either in pixels or as a percentage:

```
<IMG SRC="URL" WIDTH=x HEIGHT=x> (in pixels)
<IMG SRC="URL" WIDTH=x% HEIGHT=x%> (as a percentage)
```

It is also possible to specify how the graphic is to be placed on the page in relation to the rest of the text. This is done using the **ALIGN** attribute. By default the bottom of the image is aligned with the bottom of the text — this can be changed using **ALIGN=left|right|top|middle|bottom**. To include a large graphic that will take a long time to load, first load another (smaller) image while the second larger one loads using **LOWSRC="URL"** to specify the location of the initial image.

If not specified, the browser expects to find the image in the same folder as the web page. However, it is common to store images in a sub-folder. You must then include the folder name in the src attribute. For example,

```
<img src="/images/html5.gif" alt="HTML5 Icon">
```

Activity 12: Graphics

In this Activity you will use the IMG tag and its attributes to create a Web page that includes graphic elements.

1. Open Notepad and begin a new HTML document by entering the main structural tags.
2. Save this file as **image.html.**
3. Now find an image or icon to use in your page. You can save one from any Web page by right-clicking the image and selecting Save As. Save the image in the same directory as **image.html**.
4. This image is going to be embed into the document. Save the file after each instruction and view it in your browser to see the affect.
5. Embed the graphic image in the document using the <IMG SRC="URL"> tag.
6. Include some alternative text to be used <IMG SRC="URL" alt="text">
7. If you can find out the width and height dimensions of the image, include these, otherwise try resizing the image as <IMG SRC="URL" alt="text" width=50 height=50>.
8. Turn the image into a hyperlink by linking it to a page of your choice:
   `<A HREF="URL"> <IMG SRC="URL" alt="text" width=50 height=50></A>`
9. Turn the border that appears around the image off.
   `<IMG SRC="URL" alt="text" width=50 height=50 border=0>.`
10. Enter a short paragraph of text.
11. Align the image so that appears on the right side of the Web page.
    `<IMG SRC="URL" alt="text" width=50 height=50 border=0 align="right">`

3.6.2 Objects

This section described how to incorporate other objects objects - such as multimedia and Java applets - into your Web pages:

- The <EMBED SRC="URL"> tag is used to embed media such as background sound. For example,
  `<EMBED SRC="music.avi" WIDTH=320 HEIGHT=200 autostart=true loop=3>` inserts an AVI movie object into a page, scales it to 320 x 200, starts playing the video, and loops it three times.
- Java applets can be embedded using the <APPLET> tag. For example,
  `<APPLET CODE=clock.class codebase=http://www.server.com/classes/ height=100 width=100> </ APPLET>` loads a Java applet from a remote Web server.

The <OBJECT> tag is used by multimedia software publishers like Macromedia and Digital Workshop (among others) to embed their programmes into Web browsers. Originally used by Microsoft for its Active-X technology, Netscape has adapted it to some degree. You probably will not be writing code using this tag. Programmes, like Flash and Director, generate the code for you to copy and paste into your HTML files. Some programmes, like Dreamweaver, which is a Web authoring tool, produce sophisticated code snippets based upon the designer's input parameters.

The following is an example that Macromedia Director produced for a Shockwave movie. Notice how the EMBED tag is inside the OBJECT tag.

```
<object classid="clsid:166B1BCA-3F9C-11CF-8075-444553540000"
codebase="http://download.macromedia.com/pub/shockwave/cabs/director/sw.cab
#version=7,0,0,0" width="150" height="100">

<param name="src" value="welcome.dcr">
<embed src="welcome.dcr"
pluginspage="http://www.macromedia.com/shockwave/download/" width="150"
height="100">

</embed>
</object>
```

3.6.3 ImageMap

An imagemap is a graphic that has certain areas on it defined as hyperlinks. These areas can do whatever normal HTML links do — email, ftp, gopher, etc. Two very popular uses for image maps are navigation bars and thumbnail sheets.

There are two types of imagemaps: **client-side** imagemaps and **server-side** imagemaps.A **server-side** imagemap requires the imagemap information to be saved within a separate file stored on a server and accessed through a CGI script. This type of imagemap is far more complicated to set up, and is not supported by all servers. Server-side imagemap behaviour varies from system to system, even among different systems using the same server. A server-side imagemap shows mouse co-ordinates at the bottom of the screen.

A **client-side** imagemap (CSIM) requires the imagemap information to be stored within the HTML document. A client-side image map shows the actual URL in the status bar message at the bottom of the browser window. Client-side image maps store the hyperlink information in the HTML document, not in a separate map file as do server-side image maps.

When the user clicks a hotspot in the image, the associated URL is sent directly to the server. This makes client-side image maps faster than server-side image maps because the server does not need to interpret where the user clicked. Client-side image maps are supported by all modern browsers.

3.7 Writing Good HTML

This final section is concerned with the production of good HTML, as well as with the importance of testing your Web documents before going 'live'.

There are a number of online validation services that check the validity of a Web page. The World Wide Web Consortium's HTML Validator , for example, checks HTML documents for compliance with W3C HTML Recommendations and other HTML standards.
If there are mistakes in the HTML document, the validator will list the problems — but if it is correct, it provides a message similar to:

```
Valid HTML 4.0
No errors found!
Congratulations, this document validates as HTML 4.0 Transitional!
```

Certain HTML editors or authoring tools have built-in HTML checkers or validators; these are sometimes useful, but are not always accurate. Sometimes they notice 'errors' you have intentionally entered, but mostly they can be of great help, particularly when producing complex Web pages. The more sophisticated tools are able to adjust to the level of the target browser, as well as fine tune options such as listing missing ALT attributes inside IMG tags.

Activity 13: Validating HTML (Optional)

This Activity examines online HTML Validation Services

1. Look at the at W3C's Validator [http://validator.w3.org] and submit two or three of your Web pages. What happens?
   Note that you need to give the validators an URL. In other words, your document must exist somewhere on the WWW for it to be validated. You could search for a free hosting service, or place the page on the personal server space most ISPs provide for their clients. Obtaining an URL is not trivial, but it will allow you to validate your code.
2. Try at least one other online validation service and compare the results. Which is the easiest to use? What features of the services do you like or dislike?
3. Discuss your results with the other students in the course in the Discussion Forum.

3.8 Discussion and Answers

3.8.1 Discussion Topics

At an early stage of learning HTML it is often the little problems that provoke discussion. So you may want to spend some time online discussing the problems you have encountered with your fellow students or tutors.

In particular, you might want to discuss the different format of HTML tags. Some (like bold) have end markers, some (like the line break) do not have end markers, and some (like anchors and images) take arguments. Why do different tags have different formats? What other tags might there be? Is there anything that cannot be done in a tag?

If you managed to validate your code discuss the usability of the Validating Services you investigated as part of Activity 13. In particular, you should:

1. Compare the results obtained from the different online validation services.
2. Discuss their ease of use and their functionality.
3. And finally, you should take an online vote on your preferred online validation service.

3.8.2 Discussion of Activity 2

Your HTML document should look something like the one below:

```
<html>
<head>
<title> My first Web page </title>
</head>

<body>
Your name
<i> csc5003 </i>
<b>Internet Commerce</b>

</body>
</html>
```

3.8.3 Discussion of Activity 4 and 5

Your solution should look like what is shown in Figure 3.1.

3.8.4 Discussion of Activity 6 - Lists

The code to create the lists should look something like the one below. The HTML of particular interest is shown in bold.

```
<UL TYPE=square>
```

```
<LI>Monday</LI>
<LI>Tuesday</LI>
<LI>Wednesday</LI>
<LI>Thursday</LI>
<LI>Friday</LI>
<LI>Saturday</LI>
<LI>Sunday</LI>
<UL>
<OL TYPE=i>
<LI>January</LI>
<LI>February</LI>
<LI>March</LI>
<LI>April</LI>
<LI>May</LI>
<LI>June</LI>
<LI>July</LI>
<LI>August</LI>
<LI>September</LI>
<LI>October</LI>
<LI>November</LI>
<LI>December</LI>
</OL>
<DL>
<DT>Spring
<DD>First season of the year: March - May
<DT>Summer
<DD>Second season of the year: June - August
<DT>Autumn
<DD>Third season of the year: September - November
<DT>Winter
<DD>Fourth season of the year: December - February
</DL>
<UL>
<LI>Spring</LI>
<OL START=3>
<LI>March</LI>
<LI>April</LI>
<LI>May</LI>
</OL>
<LI>Summer</LI>
<OL START=6>
<LI>June</LI>
<LI>July</LI>
<LI>August</LI>
</OL>
<LI>Autumn</LI>
<OL START=9>
<LI>September</LI>
<LI>October</LI>
<LI>November</LI>
</OL>
<LI>Winter</LI>
<OL START=12>
<LI>December</LI>
<LI VALUE=1>January</LI>
<LI>February</LI>
</OL>
</UL>
```

3.8.5 Discussion of Activity 7 – Comments

You should include the following tag:

```
<!--- Creating lists --->
```

3.8.6 Discussion of Activity 9 – Linking to sections within a document

The link to the specified sections should look as below:

```
<P>
<A HREF="#section_two">Section two</A><BR>
<A HREF="#section_three">Section three</A> <BR>
<A HREF="filetwo.html#section_two">Filetwo-Sec2</A> <BR>
<A HREF="filetwo.html#section_three">Filetwo-Sec3</A> <BR>
```

3.8.7 Discussion of Activity 11 – Changing appearance of links

The code to generate the colour schemes is shown below: white background, black text and red hyperlinks

```
<BODY BGCOLOR="#FFFFFF" TEXT="#000000" LINK="#FF000000">
or
<BODY BGCOLOR="white" TEXT="black" LINK="red">
<BODY BGCOLOR="#000000" TEXT="#FFFFFF" LINK="#00FFFF">
or
<BODY BGCOLOR="black" TEXT="white" LINK="cyan">
```

Chapter 4. HTML Tables

Table of Contents

Objectives

At the end of this unit you will be able to:

- explain why tables are used to organize and display data;
- construct a table using HTML5 tags;
- insert data into the relevant table cells;
- add colour to particular table cells;
- discuss the advantages and disadvantages of fixed and flexible Web page design;
- implement a table in a flexible Web page using relative measurements;
- implement a table in a fixed Web page using pixel measurements.

4.1 Introduction to Tables

4.1.1 What is a Table?

A table is a grid organized into columns and rows, much like a spreadsheet. An example table is shown below. This table consists of sixteen cells organized into rows and columns. But before beginning to use tables in website design, we should consider the role that they fill. Working with tables in HTML5 has become more powerful due to the new HTML5 table tags and other elements available in HTML5.

1	2	3	4
5	6	7	8
9 13	10 14	11 15	12 16

4.1.2 Why do We Use Tables?

Tables were initially developed as a method to organize and display data in columns and rows. This chapter discusses such tables. However, tables later became a tool for Web page layout, and as such provide a possible solution for structured navigation. Frames may also be used to provide structured navigation. However, the use of tables over frames is preferred for this purpose, as earlier Web browsers (e.g. Netscape ver.1.0) do not support frames.

To Do

Read up about tables in your textbooks.

4.1.3 Creating a Data Table

Work through Activity 1 in order to understand how tables are created. Bear in mind that rarely is anything achieved which satisfies all of the stated requirements in the first pass. The key to developing perfect Web pages relies on that old adage: "*Learn from your mistakes!*"

Therefore, as long as a start is made, and mistakes are seen as a learning experience, then the design process will eventually succeed.

Please feel free to experiment at any time. If you make mistakes but manage to correct them, take encouragement from this.

Activity 1: Creating a Table

The objective of this Activity is to create a timetable for CSC5003 students to be displayed on a Web page as shown below:

CSC503 timetable

	Monday	Tuesday	Wednesday	Thursday	Friday
6-7pm	look at website	free	Implementation	free	free
7-8pm	take some notes	free	Implementation	free	free

1. Begin a new Web page in your text editor. The header is shown below. When entering the text, try to spot the deliberate mistake and correct it as necessary.

```
<HTML>
  <HEAD>
  <TITLE>
  HTML Table Design
  </HEAD>
</TITLE>
<BODY>
</BODY >
</HTML >
```

   The correct code is given at the chapter's end.

2. Save your file as tab_ex1.html

3. The next stage is to open the table. To open and close a table, use respectively the <TABLE> and </TABLE> tags within the document's BODY.

```
<HTML>
<HEAD>
<TITLE>
  HTML Table
  Design
</HEAD>
</TITLE>
<BODY>
<TABLE>
</TABLE>
</BODY >
</HTML >
```

4. Save the file and load it in your browser. At first you will notice a blank screen as the table is not visible. A border and rows may be added to make the table visible. If you do not specify a border for the table, it will be displayed without borders. When adding a border, its size can be defined in pixels, for example: <TABLE border=10 style= “width: 80%” >. Notice the use of the width attribute to set the table to a width of 80% of the screen's size (this can also be defined in pixels). However, it is worth noting that the border attribute is on its way out of the HTML standard! It is better to use CSS by first creating a <style> tag within the <head> tag then leave using only the style attribute within the table tag. ‘td’ stands for ‘tabular data’ and ‘th’ stands for ‘tabular header’.

```
<style>
table, th, td {
    border: 1px
solid black;
}
</style>
…..
<TABLE style=
“width: 80%” >
```

5. The <TR> tag is used to add rows. Each row is composed of several data cells. Their dimensions can be defined using width and height attributes: <TD width=25% height=20 bgcolor="darkred"> Notice that the cell's colour can also be defined. Try to create the table below before you look at the solution code under Discussion and Answers at the end of the chapter.

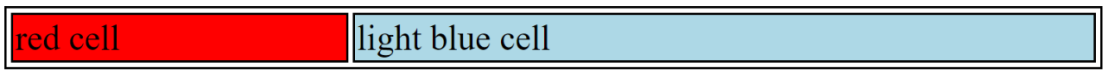


Figure 5.1: Table with one row and two columns

6. Reopen the file tab_ex1.html in your text editor and make the following amendments to <TABLE> and <tr> tags. Note the <CENTER> tag centers the table horizontally and it also centers the text within each cell in the row.

```
<TABLE style= "width: 80%" align = "center">

<tr align = "center">
```

7. Save this file as tab_ex2.html and view it in your browser. It should look as below.

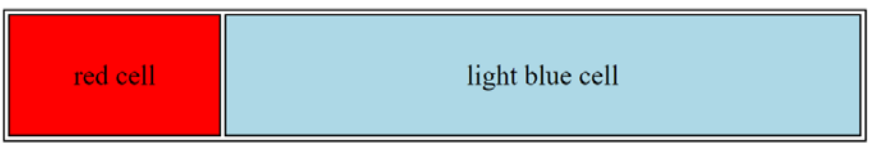


Figure 5.2: Table with centered text

8. We can see that the text is still not given any specific font. HTML <FONT> tag is deprecated in version 4.0, onwards (hence it is not supported in HTML5) and now all fonts are set by using CSS. Try to assign the Comic Sans MS font by making the following addition to the style section. Save the file as tab_ex4.html.

```
font-family: Comic Sans MS;
```

This sets all the text in in each cell to have the same font. What if you want to have different fonts in each cell? To do this, you can use the <p style > tag within each <TD> tag. Modify your <TD> tags to the following:

```
<TD width=25% height=70 bgcolor="red">  <p style="font-family: verdana">red
cell </p> </td>
<TD width=75% bgcolor="lightblue">   <p style="font-family: Comic Sans MS">
light blue cell </p></td>
```

9. To add a caption to a table use the <caption> tag within the <table> body. This caption appears on top of the table. Add the caption “Tabling” to your table thus:

```
<caption> Tabling </caption>
```

10. Save the file as tab_ex3.html and view it in your browser. It should look as below.

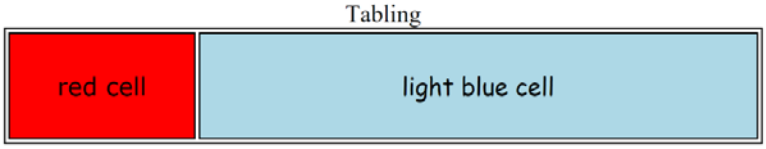


Figure 5.3: Table with caption and text with different font

11. In order to meet the objective of this Activity — that is, to create a timetable for CSC5003 — use the HTML code in the next page. Save this as tab_ex4.html. One extra HTML tag needs to be introduced: the TH tag, which inserts a table header cell. It is similar to the TD tag and has the same attributes (i.e. align, bgcolor, height etc.). However, TH is used to set the cell's text apart from the rest of the table's text, usually setting it bold and slightly larger. Now that you have completed Activity 1, you should have a good idea of how to create a basic data table.

```
<HTML>
<HEAD>
<TITLE>
    HTML Table Design
</TITLE>

<style>
table, th, td {
    border: 1px solid black;
    }
</style>

</HEAD>
<BODY>

    <TABLE style= "width: 80%" align = "center">
    <caption> CSC503 timetable </caption>
    <tr >
        <td width=50%> </td>
        <th width = 150> Monday </th>
        <th width = 150> Tuesday</th>
        <th width = 150> Wednesday </th>
        <th width = 150> Thursday</th>
        <th width = 150> Friday</th>
    </tr>
    <tr >
        <td > 6-7pm </td>
        <td > Look at website</td>
        <td > free </td>
        <td > Implementation </td>
        <td > free </td>
        <td > free </td>
    </tr>
    <tr >
        <td > 7-8pm </td>
        <td > Take some notes</td>
        <td > free </td>
        <td > Implementation </td>
        <td > free </td>
        <td > free </td>
    </tr>
    </TABLE>

</BODY >
</HTML >
```

Here are instructions on how to organise and display data in a table:

1. Insert the <TABLE> tag and decide on the table's dimensions (if required)
2. Add a row using the <TR> tag
3. In the newly created row, insert a cell <TD> with the necessary dimensions and other attributes
4. Add the data to be displayed
5. Terminate the data cell </TD>
6. Repeat steps 3-5 as necessary
7. Terminate the row </TR>
8. Repeat steps 2-7 until all the necessary rows have been added
9. Terminate the table </TABLE>

To Do

Look up the basic table structure in your textbooks and on the Internet. Draw up a list of the tags for your own use and reference.

Check your list against this one:

HTML tag	Comments
<TABLE> </TABLE>	Table definition and end tag
<CAPTION> </CAPTION>	Caption definition and end tag
<TR> </TR>	Row definition and end tag
<TD> </TD>	Cell definition and end tag

4.1.4 Activity 2: HTML Colour Table

This Activity's objective is to write the HTML code to display the following table. Feel free to add more colours.

-

Some HTML Colors

Colour	Name	hexidecimal	RGB value
	Salmon	FA8072	250-128-114
	Gold	FFD700	255-215-0

See the end of the chapter for the solution.

To Do

Read up on 'Spanning Rows and Columns' and 'Table Appearance and Colours' in your textbooks. Add the new tags to your list of table related tags.

4.2 Using Tables in Page Design

4.2.1 Using Tables in Page Design

Tables are useful for laying out text and images on in Web page. Before continuing with instructions on how to do this, let us first consider why there is a need to manage layout.

It is important to realise that it is not a monitor's absolute size that is usually of interest, but rather its screen **resolution**. While a Web browser can manage to layout a document at any resolution, different resolutions do effect the layout and presentation of an HTML document. Resolution is measured in picture elements, called **pixels**. Typical monitor resolutions are 640x480, 800x600, 1024x870, 1280x1024 and 1600x1200.

Resolution and monitor size are independent of one another: a large monitor can have a low resolution, while a small monitor may have a high resolution. Resolution is determined by the hardware, the user, and the video card driver installed on the computer. A single monitor may have a choice of resolutions.

There is also the issue of a browser's 'live space'. Live space refers to the browser area the Web page is displayed in. This can vary from user to user, as the toolbars and status bars the user chooses to have displayed in the browser will reduce or increase the live space available to a Web page.

It is for all these reasons that the dichotomy between **fixed** and **flexible** Web page design has occurred.

By default, all Web pages are designed with flexibility in mind. Flexibility can be defined as a Web page's ability to resize and adapt to the available resolution, monitor and window sizes. Such an approach has both advantages and disadvantages.

Advantages:

- **Default Setting**: therefore no new tags are needed — the Web page fills entire space.
- **Philosophical**: flexibility is the philosophy of the Web i.e. it should be accessible by the greatest number of users.
- **Realistic**: resolutions, monitor and window sizes are always different. Keeping a Web page flexible allows it to be viewed on many available formats.

Disadvantages:

- **Uncomfortable**: reading text on large monitors is uncomfortable as the lines are too long.
- **Unpredictability**: the designer often cannot predict how a Web page will appear under varying resolutions and live space sizes.
- **Coherence**: on small monitors, everything may not appear correctly.

4.2.2 Flexible Design

While HTML is flexible by default, it should not be confused with thinking a flexible document is disorganised, poorly managed with an unstructured layout. A flexible HTML document can still be structured and organised by using, for instance, tables to create columns of text (as in newspapers), and provide layout design.

Flexible layout can be achieved by using percentage measurements for table dimensions. As an example, view the table below by changing the size of your browser's window (i.e. the live space). Observe that as the window size changes, so does the table size.

The table measurements used in this example are called 'relative' measurements, as the sizes are expressed in terms of a percentage of the screen space.

Books about Computing	
IBM PC Assembly Language and Programming	Abel, P
Object-Oriented Analysis and Design	Booch, G
Demonstration of a Flexible Table.	

4.2.3 Fixed Design

Fixed design expresses all dimensions in pixels: the dimensions remain fixed regardless of the size of the device it is viewed on. Such an approach has advantages and disadvantages:

Advantages:

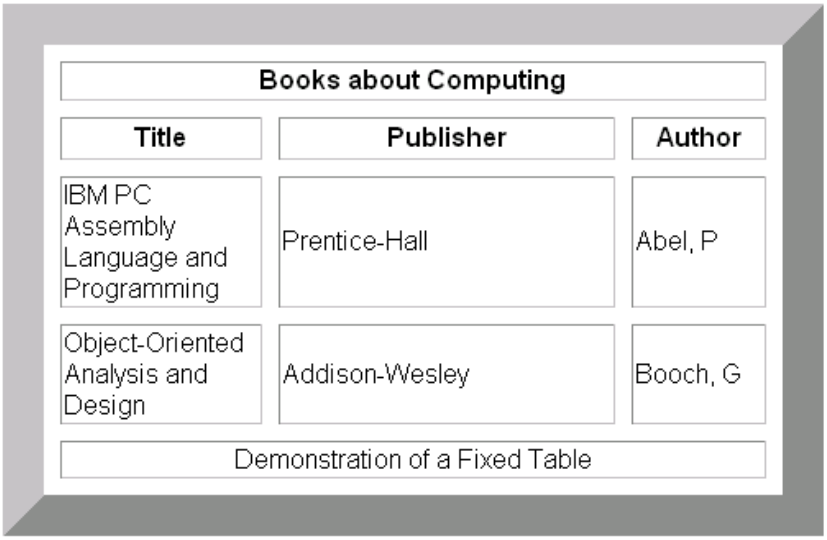
- **Consistency**: it is usually important for companies to maintain a consistent image.
- **Control**: fixed design imposes restrictions on line length and hence stops uncomfortably long lines from occurring on Web pages.

Disadvantages:

- **Incompatible**: the chosen fixed size may be too large for a user's available live space, causing the user to scroll in order to view the whole page; a fixed Web page may also appear too small, leaving unsightly blank spaces.
- **Totalitarian**: the Web does not want to run the risk of being under too much control, i.e. we do not want every Web page to be identical. Some issues in print design are not transferable to the Web, which has its own idiosyncrasies, giving it the advantage of being independent of the print media.

To develop a fixed Web page using tables, supply all measurements in pixels. The table below is a demonstration

of a fixed table — change the size of your browser window and see the effect it has.



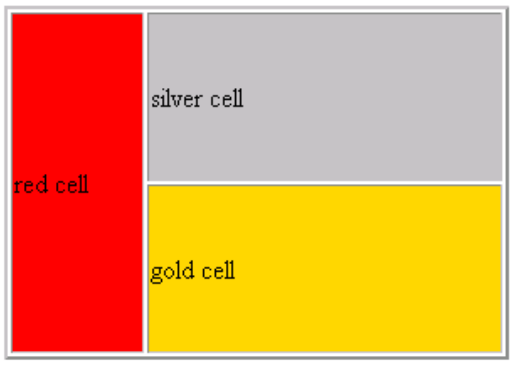
Books about Computing		
Title	**Publisher**	**Author**
IBM PC Assembly Language and Programming	Prentice-Hall	Abel, P
Object-Oriented Analysis and Design	Addison-Wesley	Booch, G
Demonstration of a Fixed Table		

To Do

Read up about standard table templates in your textbooks.

4.2.4 Activity 3: Using rowspan

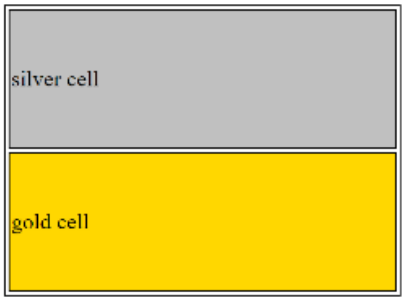
This Activity introduces you to the attribute **rowspan**. The objective of this Activity is to create the following table.



1. Let us start by creating the necessary code for displaying the silver and gold cells.

```
<TABLE style= "width: 30%" align = "center">
<tr >
      <td bgcolor = "silver" height =100> silver cell</td>
</tr>

<tr >
      <td bgcolor = "gold" height =100> gold cell</td>
</tr>
</TABLE>
```

2. Save this file as adv_tab1.html and view it in your browser. It should appear as so:



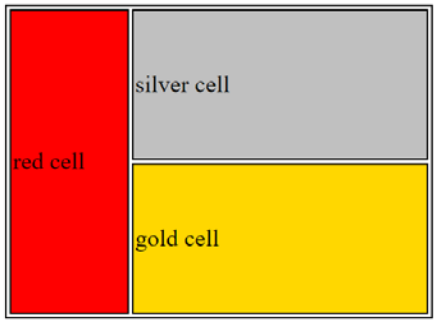
3. Now we insert a red cell spanning two rows. This is done with the **rowspan** attribute. The following syntax is used when designing tables that include **rowspan**.

   <td **rowspan=x**> where **x** is the number of rows to be spanned.

   Reopen adv_tab1.html and make the amendment shown in bold, below.

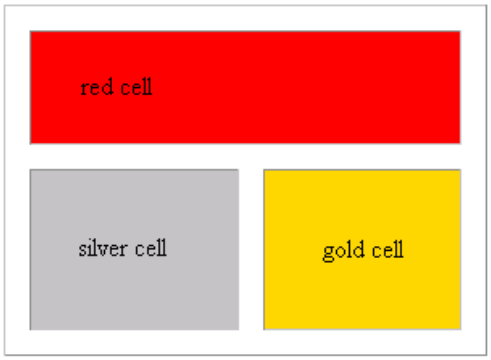
```
<TABLE style= "width: 30%" align = "center">
<tr >
      <td bgcolor = "red" rowspan = 2 width = 75> red cell</td>
      <td bgcolor = "silver" height =100> silver cell</td>
</tr>

<tr >
      <td bgcolor = "gold" height =100> gold cell</td>
</tr>
</TABLE>
```

4. Save this exercise as adv_tab2.html and view it in your browser. It should now look as below:



4.2.5 Activity 4: Using colspan, cellspacing and cellpadding

This Activity introduces you to the attributes **colspan**, **cellspacing** and **cellpadding**. The objective of this Activity is to create the following table.



1. Let us begin by creating the silver and gold cells.

```
<TABLE style= "width: 30%" align = "center">
<tr >
      <td bgcolor = "silver" height =100> silver cell</td>
      <td bgcolor = "gold" height =100> gold cell</td>
</tr>
</TABLE>
```

2. Save this file as adv_tab3.html and view it in your browser. It should appear as below:

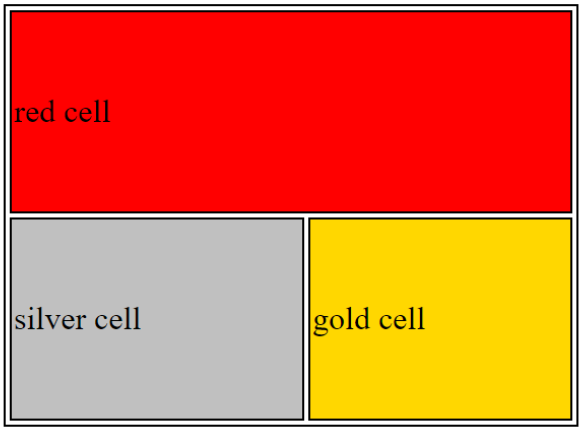
3. For the next step we insert a cell that spans the two columns, using the **colspan** attribute:

   <td **colspan=x**> where **x** is the number of columns to be spanned.

   Reopen adv_tab3.html and make the amendment as shown in bold, below:

```
<TABLE style= "width: 30%" align = "center">
<tr>
<td   colspan=2 height=100 bgcolor="red">red cell</td>
</tr>
<tr >
      <td bgcolor = "silver" height =100> silver cell</td>
      <td bgcolor = "gold" height =100> gold cell</td>
</tr>
</TABLE>
```

4. Save this exercise as adv_tab4.html and view it in your browser. It should appear as below.



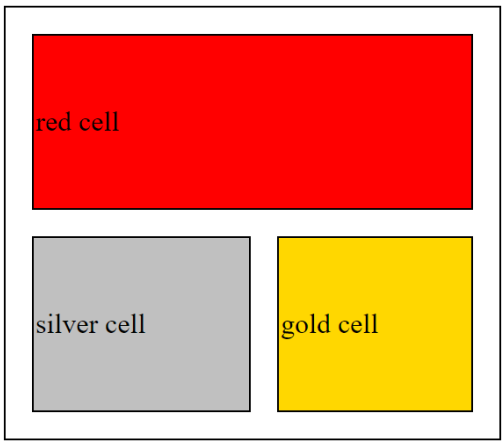
5. The space between the cells is known as the cellspacing. This is controlled with the table attribute **cellspacing**. The following syntax is used with **cellspacing**:

   <table **cellspacing=x**> where **x** is the amount of cellspacing required.

Reopen adv_tab4.html and make the following alterations to the code, as shown in bold.

```
<TABLE style= "width: 30%" align = "center" cellspacing = 15>
<tr>
<td   colspan=2 height=100 bgcolor="red">red cell</td>
</tr>
<tr >
      <td bgcolor = "silver" height =100> silver cell</td>
      <td bgcolor = "gold" height =100> gold cell</td>
</tr>
</TABLE>
```

6. Save this exercise as adv_tab5.html and view it in your browser. It should appear as below:



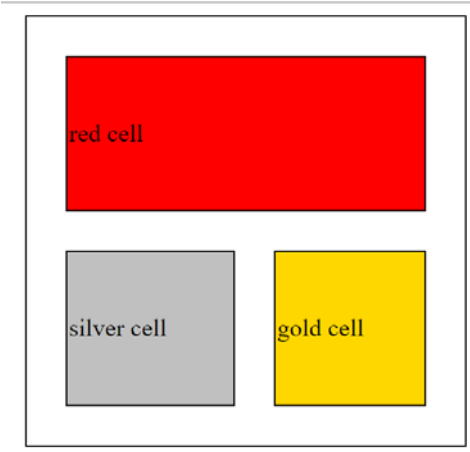
7. The space between the text 'red cell' and the cell's border is known as the cellpadding. This can be altered with the attribute **cellpadding**:

   <table **cellpadding = x**> where **x** is the thickness, measured in pixels, of the desired cellpadding.

   Reopen adv_tab5.html and make the following amendments to the code, shown in bold.

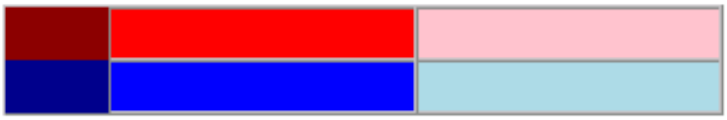
```
<TABLE style= "width: 30%" align = "center" cellspacing = 2>
<tr>
<td   colspan=2 height=100 bgcolor="red">red cell</td>
</tr>
<tr >
      <td bgcolor = "silver" height =100> silver cell</td>
      <td bgcolor = "gold" height =100> gold cell</td>
</tr>
</TABLE>
```

8. Save this exercise as adv_tab6.html and view it in your browser. It should appear as required.



4.2.6 Activity 5: More cellspacing and cellpadding

The objective of this Activity is to create the table shown below.



This Activity also introduces some of the anomalies that Web page developers deal with during Web page design. In particular, we focus on anomalies with the cellspacing attribute.

Three points should be noted for the above table:

- There are no visible borders, therefore **border=0** is needed.
- There is no cellspacing between cells, so **cellspacing=0**.
- There is no cellpadding, therefore **cellpadding=0**. Now follow these steps to complete the activity.

1. Begin a new file in a text editor and enter the following HTML code:

```
<TABLE style= "width: 30%" height = 30 align = "center" cellspacing = 0
cellpadding = 0>
      <tr>
      <td   height=15 bgcolor="darkred"></td>
      <td   bgcolor="red"></td>
      <td   bgcolor="pink"></td>
      </tr>
      <td   height=15 bgcolor="darkblue"></td>
      <td   bgcolor="blue"></td>
      <td   bgcolor="lightblue"></td>
      </TABLE>
```

2. Save your file as adv_tab7.html and view it in your browser. It should appear as required.

To Do

Read up about cell spacing in your textbooks. Try to discover common problems and mistakes that Web designers encounter when using tables. While doing this, continue to build on your list of table related tags.

4.2.7 Activity 6: Fixed and flexible Web page design

The objective of this Activity is to compare fixed and flexible Web page design.

1. Design a table with two columns, each containing text.

2. First create the table using a flexible page layout. Ensure that your table works as expected by resizing the browser window.

3. Now create the table using a fixed Web page design. Again, check that your answer is correct by resizing the browser window.

4.2.8 Activity 7: Time Table

Write the necessary HTML code for your own study timetable. This should look similar to the one shown below. (Hint: use the **colspan** and **rowspan** attributes).

Ian's Timetable

		Morning			lunch	afternoon				evening		
		9-10 am	10-11 am	11-12 am	1-2 pm	2-3 pm	3-4 pm	4-5 pm	5-6 pm	6-7 pm	7-8 pm	8-9 pm
Work	Monday	Development Meeting				Client Meeting			commute	Free		
	Tuesday	in Office				in Office						
Lecture	Wednesday	CSC205				preparation						
	Thursday	MAM200				Tutorials for MAM200			Free			
Research	Friday	Research				Research						

4.3 Review Questions

Use the following questions to assess your understanding of HTML tables. Compare your answers to those given below.

1. Initially, why were tables introduced?

2. Why do Web designers prefer to use tables instead of frames?

3. Make a list of possible instructions to develop a table in HTML.

4. Define the difference between fixed and flexible Web page design. State their advantages and disadvantages.

5. You can find the answers at the end of the chapter.

4.4 Discussions and Answers

4.4.1 Correct code for Activity 1 step 1

The code should appear as below, with the </TITLE> tag before the </HEAD> tag

```
<HTML>
<HEAD>
<TITLE>
  HTML Table Design
</TITLE>
</HEAD>
```

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

4.4.2 Solution to Activity 1 step 5

```
<HTML>
<HEAD>
<TITLE>
    HTML Table Design
</TITLE>

<style>
table, th, td {
    border: 1px solid black;
}
</style>

</HEAD>
<BODY>

    <TABLE style= "width: 80%">
    <tr>
        <TD width=25% height=70 bgcolor="red">  red cell </td>
        <TD width=75% bgcolor="lightblue">  light blue cell </td>
    </tr>
    </TABLE>

</BODY >
</HTML >
```

4.4.3 Solution to Activity 2: HTML Color Table

Here is the code used to create the table in Activity 2.

```
<HTML>
<HEAD>
<TITLE>
    HTML Table Design
</TITLE>

<style>
table, th, td {
    border: 1px solid black;
    }
</style>

</HEAD>
<BODY>

    <TABLE style= "width: 80%" align = "center">
    <caption> Some HTML colors </caption>
    <tr >
        <th width = 150> Color</th>
        <th width = 150> Name</th>
        <th width = 150> Hexadecimal</th>
        <th width = 150> RGB Value</th>
    </tr>
    <tr >
        <td bgcolor = "fa8072"> </td>
        <td > Salmon</td>
        <td > fa8072 </td>
        <td > 250-128-114 </td>
    </tr>
    <tr >
        <td bgcolor = "ffd700"> </td>
        <td > Gold</td>
        <td > ffd700 </td>
        <td > 255-215-0 </td>
    </tr>
    </TABLE>

</BODY >
</HTML >
```

4.4.4 Discussion Topic

The purpose of this exercise is to give you the opportunity to discussion the subject of fixed and flexible page design with other students in the course.

Before going to the Discussion Forum, do the following.

1. Go to the O'Reilly's Web site [http://www.oreilly.com] and the UCT website [http://www.uct.ac.za] and
   - Decide if these pages use fixed or flexible;
   - Consider whether there are any disadvantages with this type of Web page design.
2. Find another example of a Web page that uses a fixed page design, and another that uses a flexible page design.

You are now ready to join the Discussion Forum. In the forum you should be able to:

- Discuss your thoughts on the design principles used to create the O'Reilly and UCT websites
- Share the Web page design examples that you have found and discuss the advantages and disadvantages of fixed and flexible Web page design.

Additional Resources

You may find these online resources useful in your study of tables.

- Table Tutorial [http://www.ncsa.uiuc.edu/General/Internet/WWW/HTMLPrimerPrintable.html#TA]
- HTML 4.0 Specification Tables Section [http://www.w3.org/TR/REC-html40/struct/tables.html]

4.4.5 Answers to Review Questions

1. Tables were initially developed as a tool to organise and display data in columns and rows, but they are now also use to support Web page design.
2. Designers prefer to use tables to frames because older versions of browsers do not support frames.
3. List of possible instructions to develop a table in HTML:
   a. Insert the table tag and decide on its dimensions.
   b. Add a row with the TR tag.
   c. Insert a cell in the newly created row, with dimensions and other characteristics.
   d. Add the data to be displayed.
   e. Close the data cell.
   f. Repeat steps three through five as necessary.
   g. Terminate the row.
   h. Return to the second step and repeat until all of necessary rows have been added.
   i. Terminate the table.
4. Fixed Web page design gives designers more control over the Web page, as they can define the exact dimensions of the layout of the page. Flexible Web page design uses relative measurements, and so the element sizes may change, depending on the window size. The advantages of a fixed page design: constant, consistent look to the page; controls line length. Disadvantages of a fixed page design: design may be too large and cause scrolling on the screen. Advantages of a flexible page design: Web page takes up whole page, meets the needs of all resolutions, monitor and window sizes. Disadvantages of a flexible page design: can create unreadable line lengths; possibly unpredictable design.

Chapter 5. HTML Forms

Table of Contents

Objectives

At the end of this chapter you will be able to:

- Create forms with basic elements such as text boxes and buttons:
- Create forms using HTML5 elements such as form validation and email address fields.

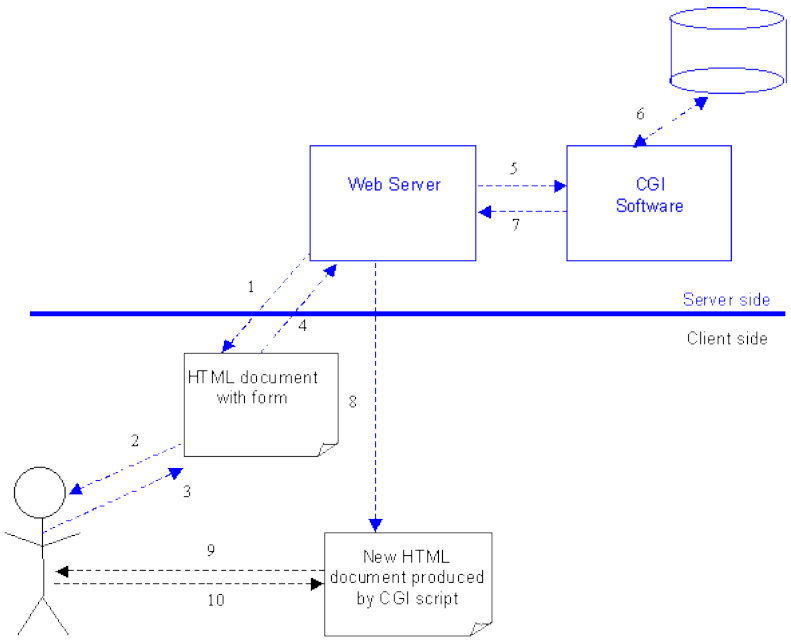
5.1 Introduction

Forms are best learnt using a hands on approach. To become proficient with HTML forms you need to create many, sorting out the problematic nuances as you go along. Therefore, the main content of the unit is a series of sections: the first is a short introduction to HTML forms; the second discusses each form element, and involves some textbook study. (You may find it more convenient to postpone activities until you have covered all the form elements).

This introduction covers the main form elements. It also explains the process that occurs when a form is submitted. The main elements of forms are: Text fields; Password fields; Text areas; Radio buttons; Check boxes; Menu buttons and scrolling lists; Submit and reset buttons; and file picker. HTML5 defines a number of new input types that can be used in forms. Examples are Email address fields; web address fields; numbers as spin boxes and sliders; date pickers; search boxes; color pickers; form validation; and required fields. We will look at some of these in this chapter.

5.1.1 Processing Forms

Although forms could simply be used to display information, HTML provides them in order to supply a way for the user to interact with a Web server. The most widely used method to process the data submitted through a form is to send it to server-side software typically written in a scripting language, although any programming language can be used. The figure below outlines the kind of processing that takes place.



1. The user retrieves a document containing a form from a Web server.
2. The user reads the Web page and interacts with the form it contains.
3. Submitting the form sends the form data to the server for processing.
4. The Web server passes the data to a CGI programme.
5. The CGI software may use database information or store data in a server-side database.

6. The CGI software may generate a new Web page for the server to return to the user.

7. The user reads the new Web document and may interact with it.

Typically, form data is sent to a server (or to an email address) as a sequence of pairs, each pair being made up of a name and an associated value. The method that this data uses to arrive at its destination depends on the data encoding. Normally the pairs will be sent as binary-encoded characters, making them straightforward to process by software, and easy to read by humans. For example, an on-line store selling used computer parts might use a form when ordering second-hand disk drives; the form would send to the server for processing information identifying the manufacturer, the model name, and maybe quote price thus:

```
manufacturer=syquest&model=ez135&price=45
```

This text represents a sequence of three name/value pairs. The names are **manufacturer**, **model** and **price**, and their associated values are *syquest*, *ez135* and *45*. There is nothing special about the names chosen or the way values are written, except that what is sent depends entirely on what the CGI software expects. If it expected **maker**, **item**, and **cost**, then the data from submitting the form would have to be:

```
maker=syquest&item=ez135&cost=45
```

Quite simply, whatever the processing software expects determines what the HTML form must provide. Often the same person or team develops both form and CGI software, so this is usually of little concern.

Because of the standard way in which the server-side software that process form data is supplied with data, such software is usually referred to as a Common Gateway Interface (CGI) script. Quite often CGI scripts on Unix servers are written in a language called Perl, but languages such as Python are becoming popular; when complex or fast processing is required, C, C++ or Java may used.

To avoid server side programming when developing forms, and to avoid depending on scripts that may require considerable study, we will mostly use a different method of processing form information: email. In fact, it is very useful to submit form data to an email address, particularly in situations when the data should be seen by a human before being processed by software.

Review Questions

Do Review Questions 1-2.

5.2 Creating Forms

This section explores the main elements found on HTML forms. This is done in a manner matching the way many people develop forms — a little bit at a time. The discussion of each form element involves both reading from your textbook as well as a practical activity. (Some of the activities will take considerable time.) You may prefer to postpone doing Activities 1-7 until you have reached the end of the section on submit and reset buttons.

5.2.1 Starting a Form

All forms start with the <FORM> tag and end with </FORM>. All other form objects go between these two tags.

The form tag has two main properties: **METHOD** and **ACTION**.

METHOD refers to **post** or **get**. The post attribute will send the information from the form as a text document. The **get** attribute is used mostly with search engines, and will not be discussed. We will generally set **METHOD="post"**.

ACTION usually specifies the location of the CGI script that will process the form data. We are not using CGI

scripts, and are instead setting this attribute to an imaginary email address (which causes the form data to be emailed to that address).

```
ACTION="mailto:put.your@email.address.here"
```

Putting these together gives us:

```
<FORM METHOD="post" ACTION="mailto:put.your@email.address.here"></FORM>
```

To Do

Read about Forms in your textbooks.

Activity 1: Starting a Form

This is the first step in creating a form. You may build on this activity in later ones by using the document you create here as a starting point (or you may prefer to save each different form with a name corresponding to the activity). First, create a new HTML document called form.htm.

Enter the normal <HEAD> and <BODY> tags. Then include the following <FORM> tag:

```
<FORM METHOD="post" ACTION="mailto:put.your@email.address.here">
```

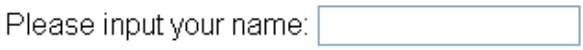
Remember to close the form with the </FORM> tag. Press the return key a few times to move it down the page, as you will be entering further code.

Finally, view your work in a Web browser, but be warned: at the moment there is little to show!

Read Discussion of Activity 1 at the end of the Unit.

5.2.2 Single-line Text Entry

A single-line text entry allows the user to input a small amount of text, like a name or an email address.



This is achieved with the following:

```
Please input your name: <INPUT TYPE="text" SIZE="40" MAXLENGTH="30"
NAME="personal-name">
```

The tag has the following elements:

<INPUT> is the tag used for most of the form objects.

TYPE="text" sets the object to a single-line text field.

Size="40" sets the field to show 40 characters.

MAXLENGTH="30" means that only a total of 30 characters can be typed in this field.

NAME="personal-name" sets the text field's name to be personal-name (this information is part of the form data sent on for further processing). The name is required to identify the value data which will be associated with it. For example, if the text box was for an email address or telephone number, we might set this attribute

with a more suggestive value, e.g. NAME="email" or NAME="tel-no". The easiest way to choose the name is simply to use the purpose of the field.

VALUE=... Another attribute, **VALUE**, can be used to place an initial text in the field. As might be expected, if this text is left unchanged it becomes the value associated with the NAME when the form is submitted. Putting an initial value in the field is usually not required, but can be used as a prompt. For example, the following HTML produces the figure that comes after it:

```
Name: <INPUT TYPE="text" NAME="name" SIZE="35"
VALUE="---please type here---">
```

Do not confuse the use of 'name' when we refer to the **NAME** attribute of an <INPUT> tag with the possibility of using **name** as the text field's actual name, or 'Name' being used in a text field that prompts the user for their own name (as in the example immediately above).

Exercise 1

After the following HTML form has been rendered by a browser, a user enters their age. The form is subsequently submitted for processing to a CGI script. Write down the name/value pair that is sent to the server-side software. Solution can be found at the end of the chapter.

To Do

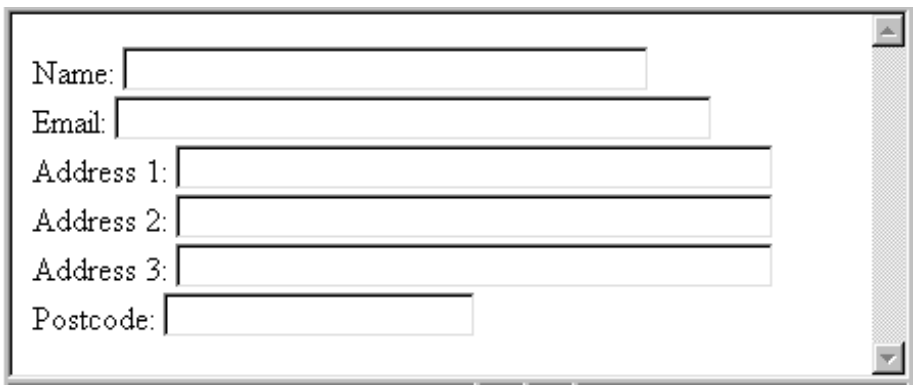
From your textbooks and Internet resources, read about the types of text that you can set for a text-entry, such as password and hidden types. Look at the Additional Content and Activities Section for some on-line resources.

Activity 2: Single-line Text Entry

In the HTML document you created before (form.htm), or a new one (as you prefer), create

1. A text field for your name that is 35 characters long.
2. A text field for your email that is 40 characters long.
3. Three text fields for your address, each 40 characters long, and an additional smaller field for your postal code.

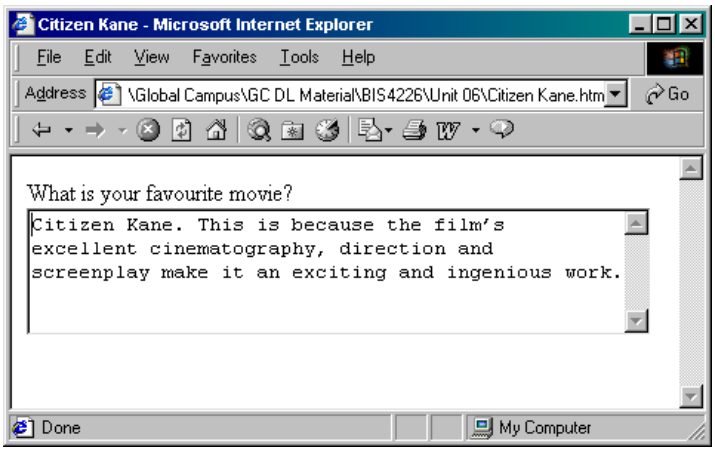
Test your code to ensure that it produces something as shown below. You may find it convenient to implement and test each part of the form separately.



Read Discussion of Activity 2 at the end of the Unit.

5.2.3 Multi-line Text Entry

Although it is possible to type as much text as needed into a single line text field, it does become more practical to enter large amounts of text into a multiple-line input field. HTML calls these fields' text areas, as in the example below:



This is achieved with using the **<TEXTAREA></TEXTAREA>** tags. They create a scrollable text area for larger amount of text:

```
<TEXTAREA NAME="movie-comments" COLS="50" ROWS="5"></TEXTAREA>
```

NAME="movie-comments" supplies the text field with the given label. Here, because the example allows the user to write about movies, we have named the form element "**movie-comments**".

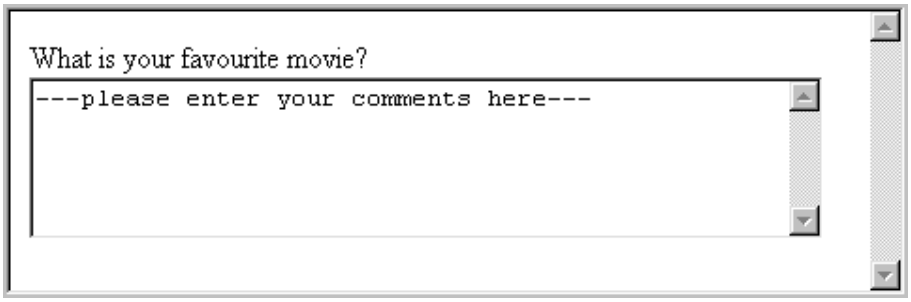
COLS="50" specifies the width, in characters, of the text area. Here 50 characters have been specified.

ROWS="5" specifies the height of the text area in rows. This examples specifies five columns. Note that the text

that should appear in the multi-line field is placed between the <TEXTAREA> and </TEXTAREA> tags. So, for example, users could be prompted to input their comments on a movie thus:

```
<TEXTAREA NAME="movie-comments" COLS="50" ROWS="5">
---please enter your comments here--- </TEXTAREA>
```

This would produce the following:

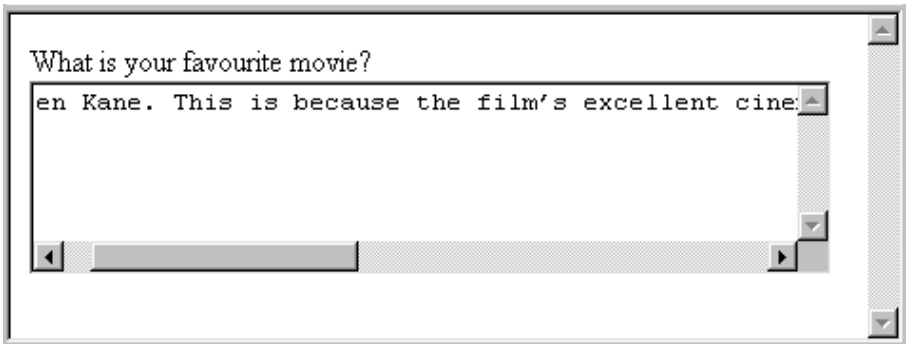


No horizontal scroll bars are present in the above text area examples (just as there rarely are any in a Web browser). This is because, by default, text-wrapping is on. Wrapping is a feature that causes words that cannot fit within a window pane to be moved to the following line or row. Web browsers wrap text (and most other elements) so that when a window is resized, text is redistributed over subsequent lines. A **WRAP** attribute is available, and the default value is **WRAP="ON"**. You can change wrapping to off, which will cause a horizontal scroll bar to appear at the bottom edge of the text area.

Exercise 2

Change the HTML for the movie opinion text area so that the text does not wrap and a horizontal scroll bar

is provided, as in the figure below:



Solution can be found at the end of the Unit.

To Do

Read about Text Area in your textbooks.

Activity 3: Multi-line Text Entry

In an HTML document do the following:

1. Replace the address text fields with one text area large enough to hold the whole address. Use the facilities of the <INPUT> and <TEXTAREA> tags to prompt the user by including placeholder information in the text fields and text area.

2. Now make the layout of the form more aesthetically pleasing by placing the form in a two-column table, with field labels (e.g. 'Name:', 'Email:') in the left hand column, and the widgets in the right-hand column.

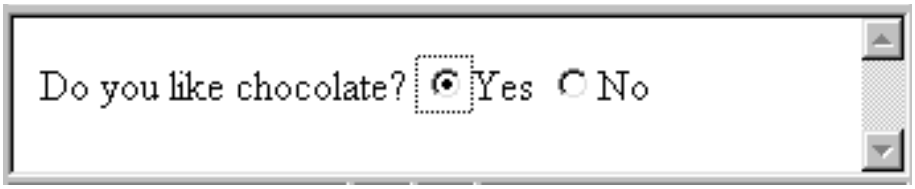
3. Think of three questions relating to the Internet that you might like to ask a user if you were a market researcher trying to gather user information. Create the additional text areas in your form to ask these questions.

Read Discussion of Activity 3 at the end of the chapter.

5.2.4 Radio Buttons

Radio buttons are often used on questionnaires to indicate a person's opinion, or their likes and dislikes. They can also be used for 'yes' or 'no' responses. Radio buttons should be used when only one answer from a set of possible answers may be chosen. This is best illustrated by example:

Example 1

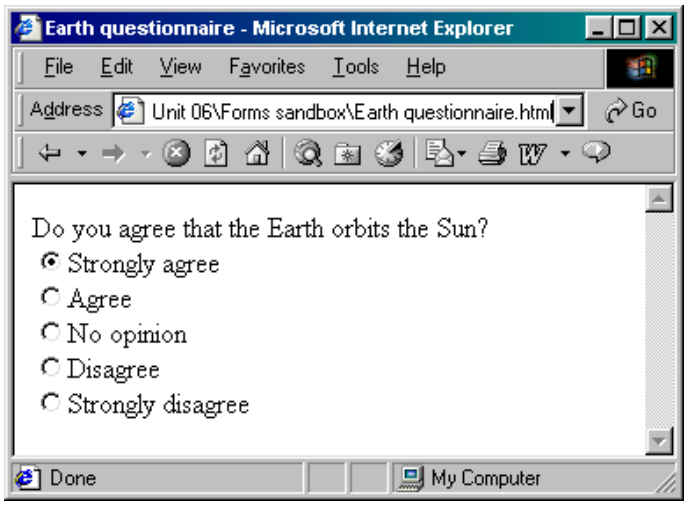


This is achieved with:

```
Do you like chocolate?
<input type="radio" name="chocolate" value="yes">Yes
<input type="radio" name="chocolate" value="no">No
```

Notice that the *same* name — chocolate — has been used for each element. This is necessary because when the form is submitted only the value of the currently selected radio button will be submitted with the given name. The software processing the data will *either* receive **chocolate=yes** or **chocolate=no**, which allows for straightforward processing.

Example 2



This is achieved with:

```
Do you agree that the Earth orbits the Sun?<br>
<INPUT TYPE="radio" NAME="orbits" VALUE="strongly_agree">Strongly agree<BR>
<INPUT TYPE="radio" NAME="orbits" VALUE="agree">Agree<BR>
<INPUT TYPE="radio" NAME="orbits" VALUE="no_opinion">No opinion<BR>
<INPUT TYPE="radio" NAME="orbits" VALUE="disagree">Disagree<BR>
<INPUT TYPE="radio" NAME="orbits" VALUE="strongly_disagree">Strongly
  disagree<BR>
```

The tag and its attributes are as follows.

<input> is the tag used for most of the form objects.

type="radio" sets the object to a radio button.

name="orbits" labels the entire set of radio buttons. This makes identifying the data easier. For example if the radio button was for the question 'Do you own an automobile?', you might set **name="automobile"**

VALUE=... With a set of radio buttons for one question, it is not enough to provide a name only. We need to give each radio button a value so that the form-processing software (CGI script or email) can determine which radio button has been selected. This is where the **value** attribute comes in. Each of the values above is set to the answer for that radio button. This information is sent with the data when the form is submitted. Hence, the form in the above figure would submit **orbits=strongly_agree**.

CHECKED This has not been used in the above example. If it were included, the button is set as if it had been clicked. This is useful is one choice should be made the default.

To Do

Read about Radio Buttons in your textbooks.

Activity 4: Radio Buttons

In an HTML document, add these numbered questions and create radio buttons for them.

1. Do you like playing computer games? Yes / No

2. How much did you enjoy the 'Star Wars' Trilogy? I enjoyed it / I did not enjoy it / I have not seen it

3. Do you have an email address? Yes / No

4. Chocolate is delicious? strongly agree / agree / neutral / disagree / strongly disagree

5. Then create four more questions of your own choice that use radio buttons.

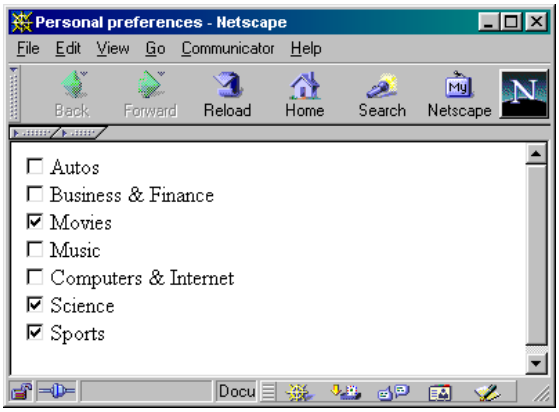
(While working on this activity, you might like to think how the form elements might interact with list structures.)

Read Discussion of Activity 4 at the end of the chapter.

5.2.5 Check Boxes

Checkboxes are one of the simplest objects that can be placed on a form. These input elements can only be selected and de-selected. Unlike radio buttons, more than one can be selected at a time.

For example, when signing up for a free e-mail account with GMail [http://www.gmail.com] or Hotmail [http://www.hotmail.com], a user may well have to fill in a series of forms. One of them is often an interests form, and looks something like this:




```
<INPUT TYPE="checkbox" NAME="autos" VALUE="yes">Autos<BR>
<INPUT TYPE="checkbox" NAME="business"
VALUE="yes">Business & Finance<BR>
<INPUT TYPE="checkbox" NAME="movies"
VALUE="yes">Movies<BR>
<INPUT TYPE="checkbox" NAME="music" VALUE="yes">Music<BR>
<INPUT TYPE="checkbox" NAME="computing"
VALUE="yes">Computers & Internet<BR>
<INPUT TYPE="checkbox" NAME="science"
VALUE="yes">Science<BR>
<INPUT TYPE="checkbox" NAME="sports"
VALUE="yes">Sports<BR>
```

<**INPUT**> is the tag used for most of the form

objects. **type="checkbox"** sets the object to a

checkbox. **name** is used to supply a name to the

checkbox.

VALUE="yes" if the item is checked, this is the value that will be associated with the name

when the form is submitted for processing. Hence, in the above example, **yes** will be associated with each of the name values **movies**, **science** and **sports**.

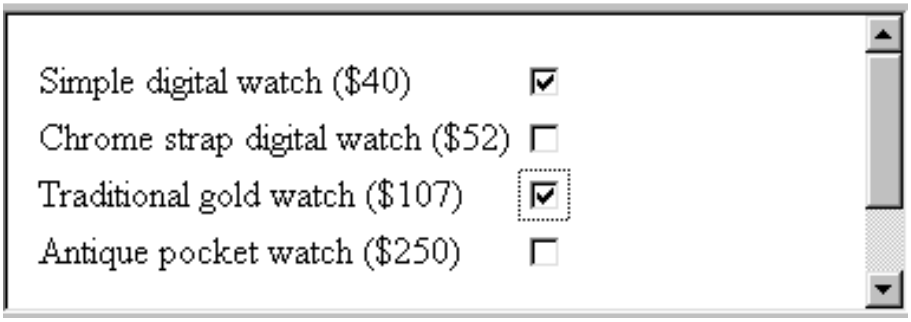
CHECKED This has not been used in the above examples. If it were included as an attribute of the tag, the check box would be set as if it had been clicked by the user. This is useful if one or more options are to be offered as defaults.

To Do

Read about Check Box in your textbooks.

Activity 5: Check Boxes

Imagine you are developing a website that sells various types of watch. You want to allow customers to select what they want to buy with a set of check boxes, as in the figure below.

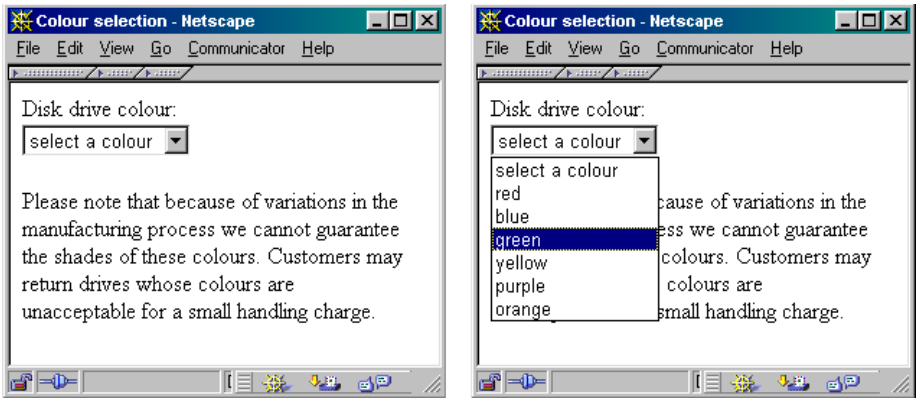


Write an HTML form to achieve this (note the tabular layout). Remember that a server-side script may be expecting the prices of the checked items, so these values will need to be transmitted for processing.

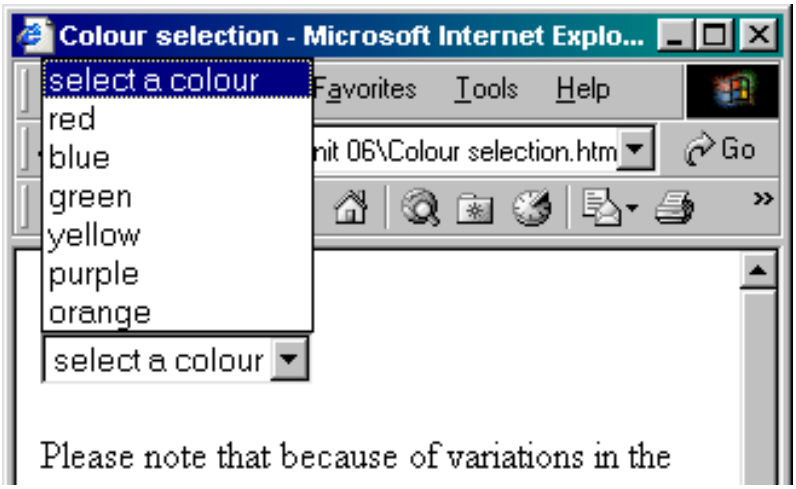
Read Discussion of Activity 5 at the end of the chapter.

5.2.6 Menu Buttons and Scrolling Lists

Menu buttons and scrolling lists are useful when you have multiple options to choose from. For example, the following shows a menu button (sometimes imprecisely called a pull-down or drop- down menu). Clicking on the 'select a colour' option on the button displays all the other options in the button's menu. Pulling down to the required option will set the value for this element (for subsequent transmission).



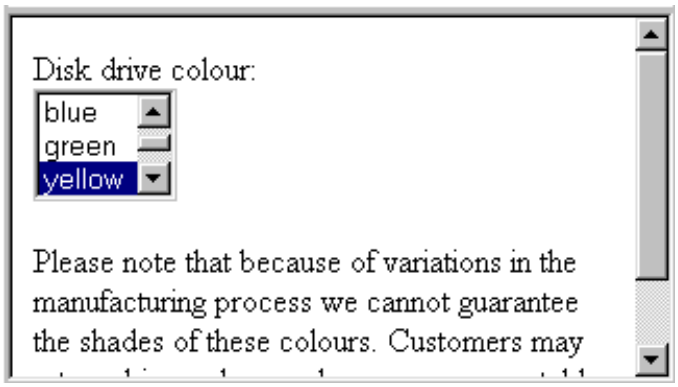
If the menu button is near the bottom of a window, the menu pops up:



Notice that, by convention, the first option of a menu button is usually not an option at all, but a prompt to suggest an action to the user. Hence, in the previous example, 'select a colour' is not an option but a prompt. However, if the form containing this menu button were submitted for processing, the value 'select a colour' would be associated with the named attribute.

Scrolling lists, otherwise known as selection lists, are similar to menu buttons, but they usually display more than one of the available options at a time. They rarely show all options, and the user is required to scroll in order to view them all. If all the options are displayed, no scroll bar is included with the list, and it may not be obvious to the user that they should select an option.

The above example could be redesigned using a scrolling list. The figures below show the three options, and all six options (excluding the prompt):



One clear benefit of these two input elements are that they do not take up as much space as, say, a list of radio buttons. Like radio buttons, they are also used to select one choice one from a set of mutually exclusive options.

The HTML for the menu button and scrolling list (the three-option version) are given below:

Disk drive colour: <SELECT NAME="colour"> <OPTION>select a colour</OPTION> <OPTION>red</OPTION> <OPTION>blue</OPTION> <OPTION>green</OPTION> <OPTION>yellow</OPTION> <OPTION>purple</OPTION> <OPTION>orange</OPTION> </SELECT> <P>Please note that because of variations in the manufacturing process we cannot guarantee the shades of these colours. Customers may return drives whose colours are unacceptable for a small handling charge.</P>	Disk drive colour: <SELECT NAME="colour" SIZE="3" MULTIPLE> <OPTION>red</OPTION> <OPTION>blue</OPTION> <OPTION>green</OPTION> <OPTION>yellow</OPTION> <OPTION>purple</OPTION> <OPTION>orange</OPTION> </SELECT> Please note that because of variations in the manufacturing process we cannot guarantee the shades of these colours. Customers may return drives whose colours are unacceptable for a small handling charge.</P>

<SELECT></SELECT> are used for both menu buttons and scrolling lists (the <INPUT> tag is not used).

name="colour" specifies the data name to be transmitted on form submission.

<OPTION></OPTION> are used to specify the option we want to appear in both types of menus.

size="3" This attribute sets the number of options that the scrolling list shows. In the above example, the number of options has been set to 3.

MULTIPLE This attribute guarantees that the **<SELECT>** tag results in a scrolling list rather than a menu button. It is not needed if **SIZE** is set to be greater than 1. However, if **MULTIPLE** is omitted and **SIZE=1**, or is omitted, the tag produces a menu button and not a scrolling list.

SELECTED This attribute can be included in an **<OPTION>** tag; it has not been used in the above examples. If it were included, the option is selected by default.

Exercise 3

A small Egyptian airline flies internally to Egypt while also offering flights to Kuwait, Lebanon, Bahrain, and Oman. Their new website is being developed to promote their business, especially the external services. The website should provide a list of destination options to users. Bearing in mind that most of the company's business is internal and that it wants to advertise its flights to other countries, select an appropriate selection mechanism and arrangement of options. Write the HTML.

A solution can be found at the end of the Unit.

To Do

Read about creating menus with the <SELECT> tag in your textbooks.

Activity 6: Menu Buttons and Scrolling Lists

In an HTML document:

1. Create menu buttons for the following:
   - Title: Mr / Mrs / Miss/ Ms / Dr
   - Age: under 20 (<19) / 20—29 / 30—39 / 40—49 / 50—59 / over 60 (60>)
   - Gender: Male / Female
   - Status: Employed / Unemployed / Student
2. Create scrolling menus to enter a date of birth using the following:
   - Day: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 151, 6, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31

- Month: January, February, March, April, May, June, July, August, September, October, November, December

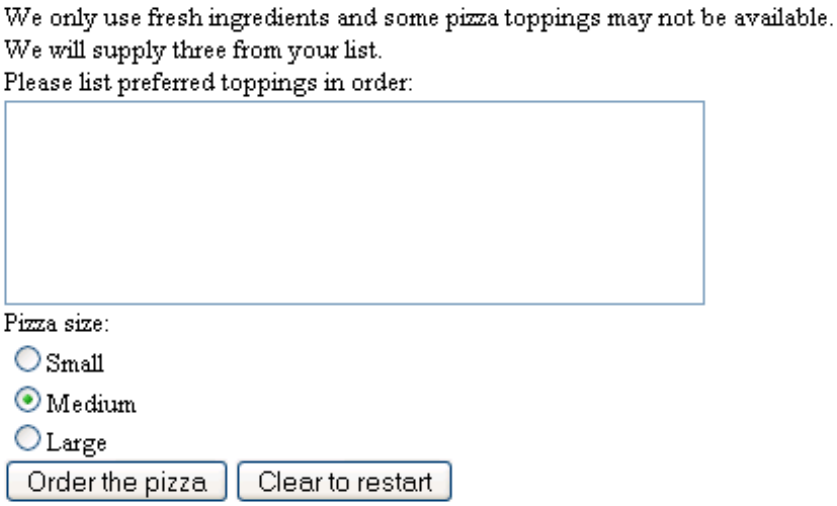
- Year: 1960, 1961, 1962, 1963, 1964, 1965, 1966, 1967, 1968, 1969, 1970, 1971, 1972, 1973, 1974, 1975, 1976, 1977, 1978, 1979, 1980, 1981, 1982, 1983, 1984, 1985, 1986, 1987, 1988, 1989, 1991, 1992, 1993, 1994, 1995, 1996

Read Discussion of Activity 6 at the end of the chapter.

5.2.7 Submit and Reset Buttons

Submit and reset buttons have simple uses: they allow the user to send the form data onwards for processing, or to clear the form fields of all entered information.

For example: a user has ordered a large pizza on-line. They have entered four possible topping selections. To send the data for processing, the user clicks on the button labelled 'Order the pizza'. If the user were to make an error and wish to restart, clicking the button labelled 'Clear to restart' will remove the text from the multi-line text area and (as it happens) reset the choice of pizza size to 'Medium'.



The new elements in this example are the buttons labelled 'Order the pizza' and 'Clear to restart', which are submit and reset buttons respectively. When a submit button is clicked, the Web browser looks at the <form> tag to see how the data should be processed. It will either be sent as an email, as shown below, or sent for processing by a server-side script. Here is HTML for the above example:

```
<FORM METHOD="post" ACTION="mailto:pizza-orders@medpizzas.com.eg">
<P>We only use fresh ingredients and some pizza toppings
may not be available. We will supply three from your list.</P> Please
list preferred toppings in order:<BR>
<TEXTAREA NAME="toppings" COLS="40" ROWS="5" WRAP=OFF></TEXTAREA><br>
Pizza size:<BR>
<INPUT TYPE="radio" NAME="pizza_size" VALUE="small">Small<BR>
<INPUT TYPE="radio" NAME="pizza_size" VALUE="medium" CHECKED>Medium<BR>
<INPUT TYPE="radio" NAME="pizza_size" VALUE="large">Large<BR>
<INPUT TYPE="submit" VALUE="Order the pizza">
<INPUT TYPE="reset" VALUE="Clear to restart">
</FORM>
```

Buttons can be customised to perform other functions, such as navigation, by using JavaScript (developers mostly use images and text links for navigation, however) See the Extension section.

A submit button may, in fact, be replaced with an image. For instance, the above example can be

modified as follows:

The <INPUT> tag for this submit button uses **image** as the value of the **TYPE** attribute and adds a **SRC** attribute to specify the image to be used, as in:

```
<INPUT TYPE="image" SRC="go-pizza-button.gif">
```

Clicking on an image used as if it were a submit button has exactly the same effect as a standard submit button, in that the form data is dispatched for processing.

Note that there is no corresponding facility to replace a reset button with an image.

To Do

Read about Submit and Reset Buttons in your textbooks.

Activity 7: Submit and Reset Buttons

Implement the check box example on personal interests and add submit and reset buttons. Test the reset button by checking several options and then clearing them by clicking Reset. To test the submit button, set the form (using the **ACTION** attribute) to send email to a real address.

Test how the submit button works when using the default encoding (see your textbook). You should receive a binary file attachment at the email address in the **ACTION** attribute. You can open this with a text editor (like Notepad).

Change the <FORM> tag to include the encoding **ENCTYPE="text/plain"** and resubmit the form. Examine the email to see how its contents have changed.

Read Discussion of Activity 7 at the end of the Unit.

5.3 HTML5 form elements

Create a new HTML file and practice the HTML5 elements in this section.

5.3.1 Email address field

The first of these new input types is for email addresses. You can allow the input field to access multiple inputs by using the multiple attribute as shown below. For <input type="email">: separate each email with a comma. Try typing an email address without the @ field and notice that the browser performs a simple validation.

```
<form>
  <input type="email" multiple>
  <input type="submit" value="Go">
</form>
```

5.3.2 Search

Search functions are perfomed on popular websites such as Google, e-commerce sites as well as personal blogs. It is probably the most common action performed on the Web every day. With HTML5 you can create a simple search action by using:

```
<form >
  <input type="search" name="search">
</form>
```

When you start typing in the search bar you will notice that you may get some suggestions that you can click on, just like you would on Google. Notice also the ‘x’ on the right which you can use to clear your search results, like you would on Safari browser.

5.3.3 Url

The url input type is for web addresses. You can use the multiple attribute to enter more than one URL. Like type="email", a browser will carry out simple validation on these fields and present an error message on form submission. This is likely to include looking for forward slashes, periods, and spaces, and possibly detecting a valid top-level domain (such as .com or .co.uk). Use the url input type like so:

```
<input type="url" name="url">
```

5.3.4 Autofocus

If you wish to have one of the field on you form automatically selected when a user loads the webpage, you would give that field the autofocus attribute. Assume that you want to mark the ‘url’ field from 5.33 with autofocus, you would use the markup:

```
<input type="url" name="url" autofocus>
```

5.3.5 Dates and Times

Usually, date pickers are constructed using JavaScript library. HTML5 enables this functionality possible within the browser. Use the markup:

```
<input id="dob" name="dob" type="date">
```

To only show the month and year, use the markup:

```
<input id="expiry" name="expiry" type="month" required>
```

To show the time, use the markup:

```
<input id="time" name="time" type="time">
```

5.3.6 Color

The color input type is pretty self-explanatory: it allows the user to select a color and returns the hex value for that color. It is anticipated that users will either be able to type the value or select from a color picker, which will either be native to the operating system or a browser’s own implementation. If you are using Chrome you will be able to pick a color from a color picker. To use the color tag, use the markup:

```
<input id="color" name="color" type="color">
```

5.3.7 Numbers as Spinboxes

Using HTML5 you can create an input field that accepts minimum and maximum numbers and then you can be able to scroll through the numbers and pick one. Use the following markup:

```
<input type="number" min="0" max="10" step="1" value="0">
```

- type="number" means that this is a number field.

- min="0" specifies the minimum acceptable value for this field.
- max="10" is the maximum acceptable value.
- step="1", combined with the min value, defines the acceptable numbers in the range: 0, 1, 2, and so on, up to the maxvalue.
- value="0" is the default value that displays before selecting another number.

5.3.8 Numbers as Sliders

By changing the input type from ‘number’ to ‘range’ you can change the spibnoxes to a slider. Use the markup:

```
<input type="range" min="0" max="10" step="1" value="0">
```

5.4 Using Forms

It is time to consider how organizations can make use of forms on their websites. This primarily involves investigating and discussing how commercial enterprises on the Web currently make use of forms.

Activity 8: How websites use Forms

For this Activity you will need to take a look at a number of websites:

- Amazon Books [http://www.amazon.com]
- Loot Free Ads [http://www.loot.com]
- Google [http://www.google.com]
- Ebay on-line auctions [http://www.ebay.com]
- Any of the computer manufacturers (e.g. Dell [http://www.dell.com])

Make a note of the following:

- The ways in which websites make use of forms.
- The different types of data that the website are trying to gather. Are there any distinct differences?

To finish this section, you will develop two extensive Web pages that make use of forms.

Activity 9: Job Application Form

Create an on-line job application form. The application form is for a computer company called SpeedyPC who are advertising for computer programmers. Your form’s action should post the application to your email address.

Your application form must have the following elements:

- Position applied for (autofocus), name, nationality, date of birth (selected from an auto picker), address (in a text area), telephone number and email (required).
- Educational history and qualifications.
- Work experience/employment/training in terms of employer history and number of years of experience selected from a slider. Set maximum years of experience to 10 years.
- Personal statement.
- Two referees including names, occupation, relationship, address, telephone.

Read discussion at the end of the chapter.

5.5 Additional Content and Activities

5.5.1 Supplementary information on HTML forms

Spend 10-20 minutes reviewing the tutorials at one or both of these sites:

- Bare Bones Guide to HTML [http://werbach.com/barebones/barebone.html]
- Yale C/AIM Web Style Guide [http://info.med.yale.edu/caim/manual/contents.html]

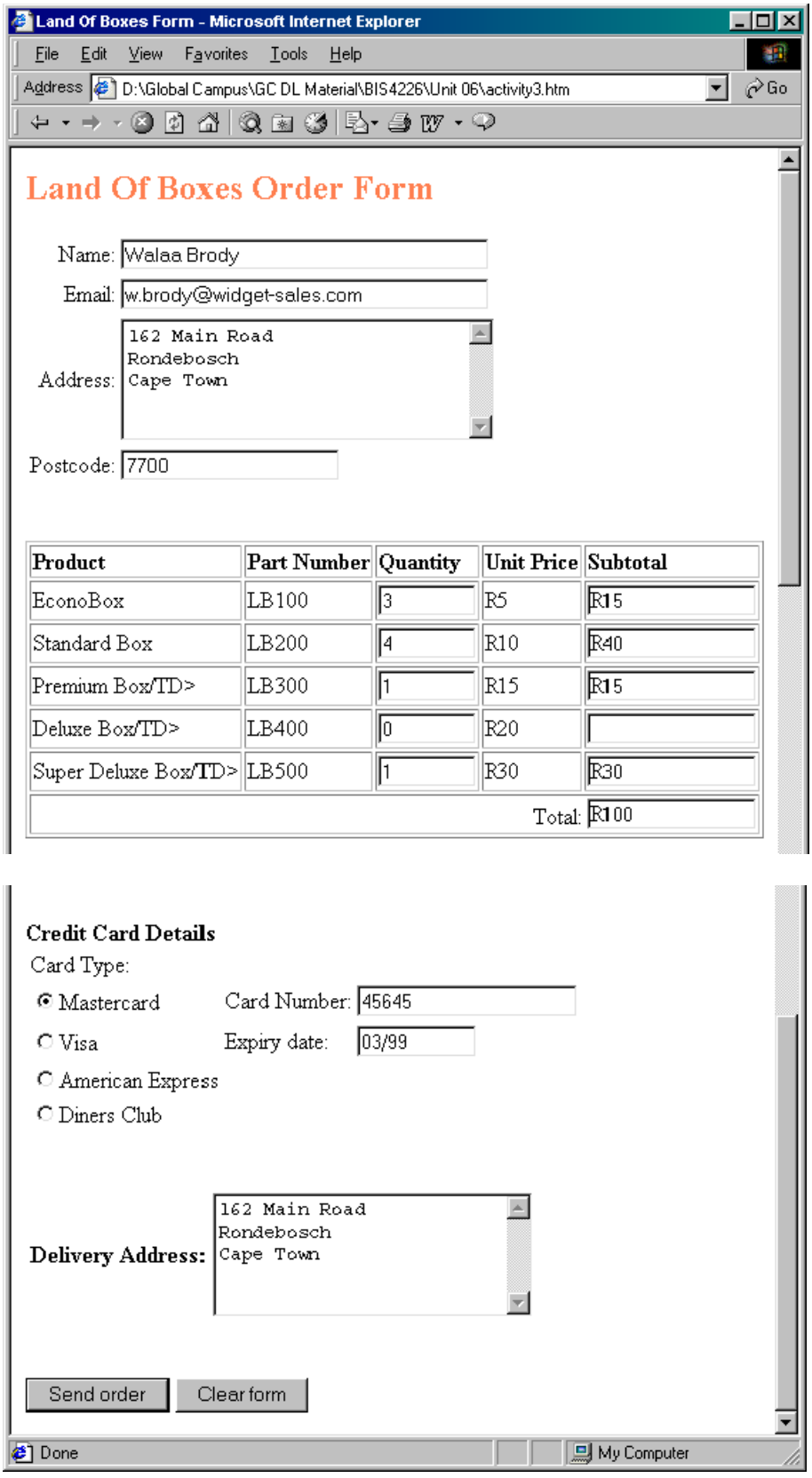
5.5.2 Additional Activity — Tabular Layout of a Complex Form

Create a form for processing orders for boxes that looks like those in the figures below. The order form is for a company called 'Land of Boxes'. Their product line consists of only five items:

1. The EconoBox: Cost R5
2. The Standard Box: Cost R10
3. The Premium Box: Cost R15
4. The Deluxe Box: Cost R20
5. The Super Deluxe Box:. Cost R30

The order form should include the following elements:

1. Contact details of purchaser, including name, address, telephone number and email.
2. Product details including price, product name, product number and product quantity.
3. Method of payment including credit card type, card number and expiry date.
4. Billing address and delivery address if they are different.
5. The final layout should look as shown in the next two figures. You will need to use tables or CSS styles to achieve this layout.



Read Discussion of Additional Activity at the end of the chapter.

5.6 Review Questions

1. Write down the main purpose of HTML forms. Answer at the end of the chapter.
2. What is CGI software?

Answer at the end of the chapter.

3. Is the size of the text a user can enter in a text field limited by the value of the **SIZE** attribute in the <INPUT> tag?

   Answer at the end of the chapter.

4. What attribute must be included in a <TEXTAREA> tag to ensure a horizontal scroll bar is included in a multi-line text field?

   Answer at the end of the chapter.

5. What is wrong with the following HTML? The code should allow the user to specify their seating arrangement, food, and preferred newspaper, for a flight with a particular airline.


```
<INPUT TYPE="radio" NAME="flight_prefs" VALUE="veggie">Vegetarian
   Food<BR>
<INPUT TYPE="radio" NAME="flight_prefs"
   VALUE="euro_food">European Food<BR>
<INPUT TYPE="radio" NAME="flight_prefs"
   VALUE="midEast_food">Middle Eastern Food<BR>
<INPUT TYPE="radio" NAME="flight_prefs" VALUE="farEast_food">Far
   Eastern Food<BR><BR>
<INPUT TYPE="checkbox" NAME="flight_prefs" VALUE="times">London
   Times<BR>
<INPUT TYPE="checkbox" NAME="flight_prefs" VALUE="post">Washinton
   Post<BR>
<INPUT TYPE="checkbox" NAME="flight_prefs" VALUE="figaro">Le
   Figaro<BR><BR>
<INPUT TYPE="radio" NAME="flight_prefs" VALUE="aisle">Aisle
   Seat <BR>
<INPUT TYPE="radio" NAME="flight_prefs" VALUE="window">Window
   Seat<BR>
<INPUT TYPE="radio" NAME="flight_prefs" VALUE="middle">Middle
   Seat<BR><BR>
```

Answer at the end of the chapter.

6. How do you set the initial state of a check

   box? Answer at the end of the chapter.

7. How do menu buttons and scrolling lists differ in how they save space on a Web

page? Answer at the end of the chapter.

8. Create a scrolling list that shows four items from the following list of personal computer processor types: Motorola 68000, Intel 8088, Intel Pentium MMX, Intel Pentium II, Intel Pentium III, Intel Celeron, PowerPC G3, PowerPC G4, AMD Athlon.

Answer at the end of the chapter

5.7 Discussions and Answers

5.7.1 Discussion of Activity 1

Your document will probably look something like this:

```
<HEAD>
<TITLE>Form example</TITLE>
</HEAD>
<BODY>
<FORM METHOD="post" ACTION="mailto:put.your@email.address.here">
</FORM>
</BODY>
```

You will notice that nothing shows when this form is rendered because it has none of the interactive elements that make forms useful.

5.7.2 Discussion of Activity 2

To produce the text fields one by one:

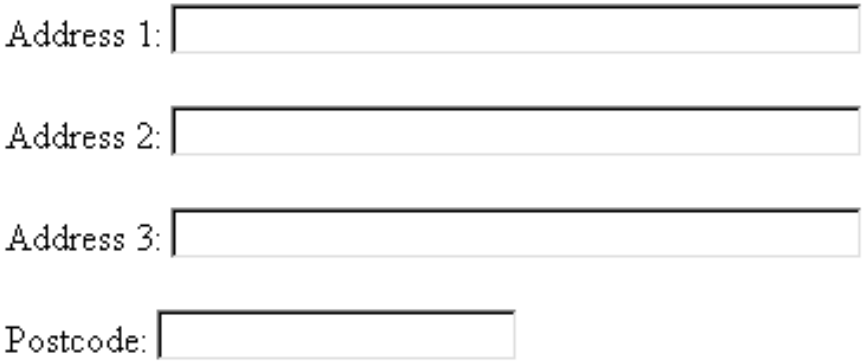
1. The name text field (below) and its HTML.


```
Name: <INPUT TYPE="text" NAME="name" SIZE="35"><BR>
```

2. The email text field (below) and its HTML.


```
Email: <INPUT TYPE="text" NAME="email" SIZE="40"><BR>
```

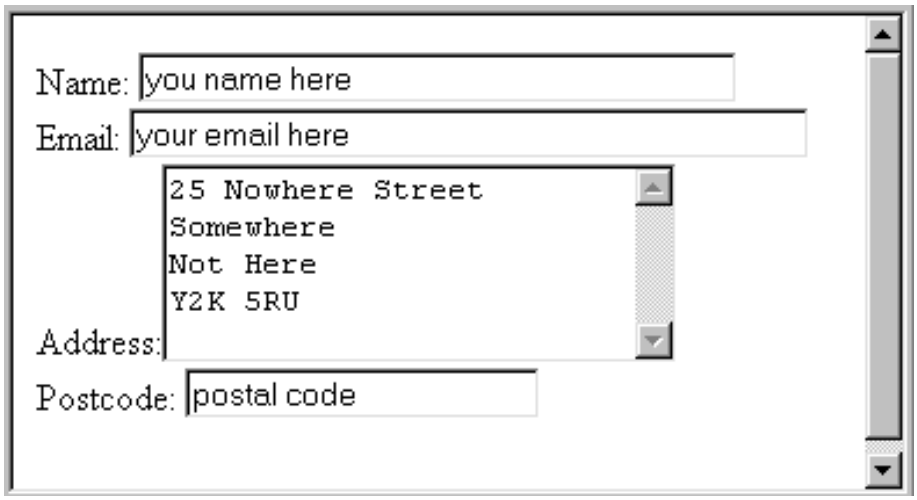
3. The three address fields and the post code field (below), with the HTML.




```
Address 1: <INPUT TYPE="text" NAME="address1" SIZE="40"><BR> Address
2:  <INPUT  TYPE="text"  NAME="address2"  SIZE="40"><BR>  Address  3:
<INPUT  TYPE="text"  NAME="address3"  SIZE="40"><BR>  Postcode:  <INPUT
TYPE="text" NAME="postcode" SIZE="20"><BR>
```

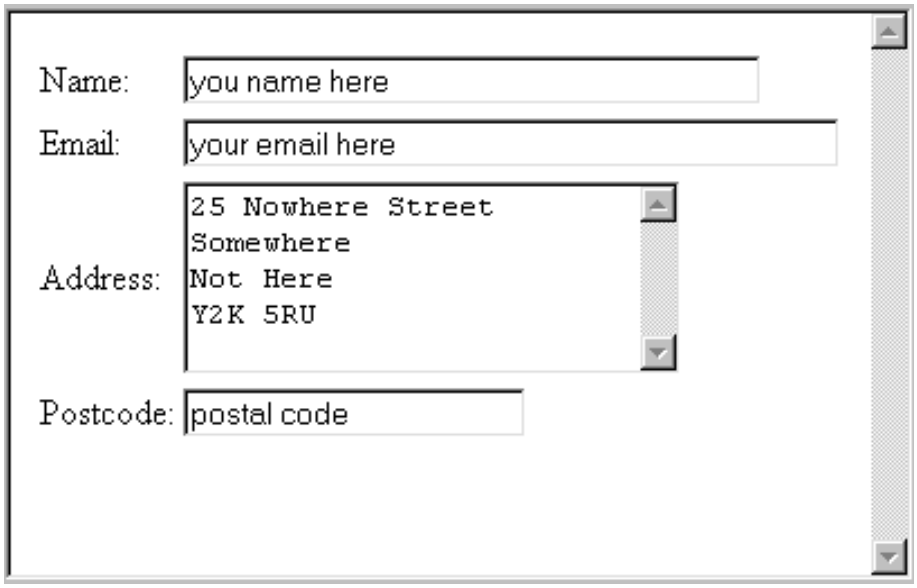
5.7.3 Discussion of Activity 3

1. The multi-line address text area and its HTML are as follows.




```
Name: <INPUT TYPE="text" NAME="name" SIZE="35" VALUE="you name here"><BR>
Email: <INPUT TYPE="text" NAME="email" SIZE="40" VALUE="your email
  here"><BR>
Address:<TEXTAREA NAME="textfield3" COLS="25" ROWS="5">
25 Nowhere Street Somewhere
Not Here Y2K
5RU
</TEXTAREA><BR>
Postcode: <INPUT TYPE="text" NAME="postcode" SIZE="20" VALUE="postal
  code"><BR><BR>
```

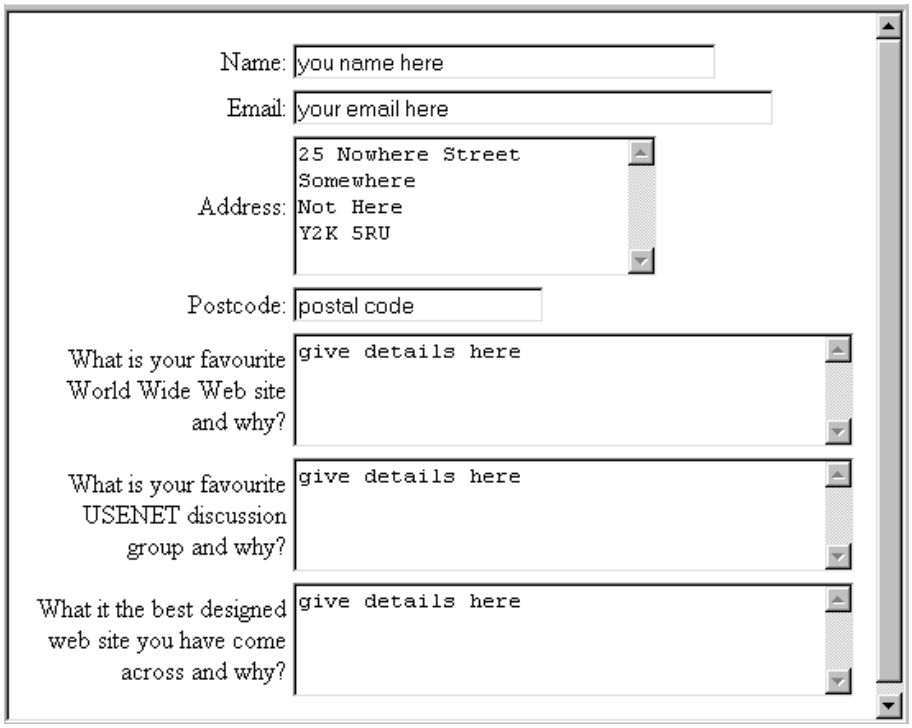
2. The nicely laid out version and the table-based HTML follow:




```
</FORM>
<TABLE>
<TR>

<TD>Name:</TD>
<TD><INPUT TYPE="text" NAME="name" SIZE="35" VALUE="you name here"></TD>
</TR>
<TR>
<TD>Email:</TD>
<TD><INPUT TYPE="text" NAME="email" SIZE="40" VALUE="your email
here"></TD>
</TR>
<TR>
<TD>Address:</TD>
<TD><TEXTAREA NAME="textfield3" COLS="25" ROWS="5">
25 Nowhere Street
Somewhere
Not Here
Y2K 5RU
</TEXTAREA></TD>
</TR>
<TR>
<TD>Postcode:</TD>
<TD><INPUT TYPE="text" NAME="postcode" SIZE="20" VALUE="postal
code"></TD>
</TR>
</TABLE>
</FORM>
```

Note that the cells have been indented to aid readability. Note that the initial value of the text area (between <TEXTAREA> and </TEXTAREA> tags) should not be indented unless the text, as rendered by the browser, should itself appear indented.

3. Adding three more text areas to the table structure is more complicated than adding them to a simple (non-table) form: adding a lot of text before the text areas separates them the labels ('Name:', 'Email:', etc.). Aligning the labels to the right brings them closer, as in the figure below. The extra HTML follows the figure:



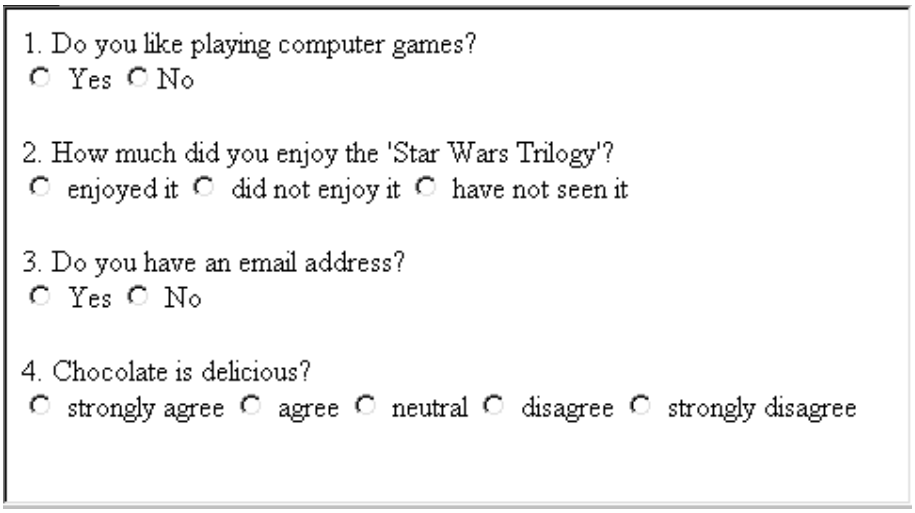

```
<TR>

   <TDALIGN="right">What is your favourite World Wide Web site
      and why?</TD>
   <TD><TEXTAREA name="textfield4" cols="40" rows="4">give details
      here</TEXTAREA></TD>

</TR>
<TR>

   <TDALIGN="right">What is your favourite USENET discussion group
      and why?</TD>
   <TD><TEXTAREA name="textarea" cols="40" rows="4">give details
      here</TEXTAREA></TD>

</TR>
<TR>

   <TDALIGN="right">What it the best designed website you have
      come across and why?</TD>
   <TD><TEXTAREA name="textarea2" cols="40" rows="4">give details
      here</TEXTAREA></TD>

</TR>
```

5.7.4 Discussion of Activity 4

The following HTML produces the required form (shown after the HTML).

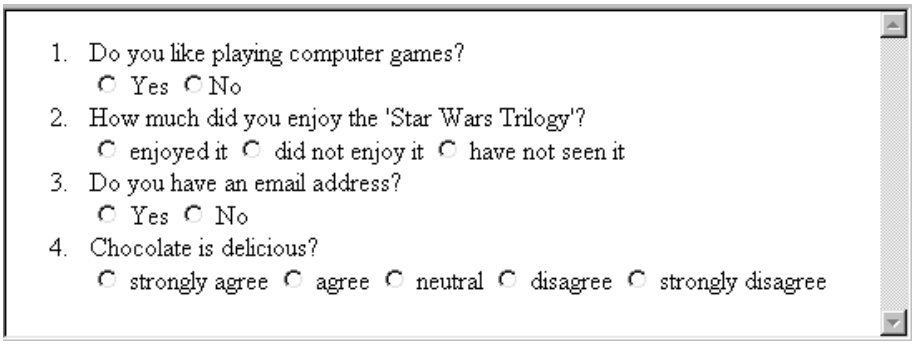
```
<FORM METHOD="post" ACTION=mailto:name@xyz.ac.uk>
<p>1. Do you like playing computer games? <br>
<input type="radio" name="games" value="yes"> Yes
<input type="radio" name="games" value="no">No</p>
<p>2. How much did you enjoy the 'Star Wars Trilogy'? <br>
<input type="radio" name="starwars" value="eit"> enjoyed it
```

```
<input type="radio" name="starwars" value="dneit"> did not enjoy it
<input type="radio" name="starwars" value="hnsit"> have not seen
it</p>
<p>3. Do you have an email address? <br>
<input type="radio" name="email" value="yes"> Yes
<input type="radio" name="eamil" value="yes"> No</p>
<p>4. Chocolate is delicious? <br>
<input type="radio" name="radiobutton" value="sagree"> strongly
agree
<input type="radio" name="radiobutton" value="agr"> agree
<input type="radio" name="radiobutton" value="neutral"> neutral
<input type="radio" name="radiobutton" value="disagree"> disagree
<input type="radio" name="radiobutton" value="sdisagree"> strongly
disagree</p
</FORM>
```



Note that a different style of layout can be achieved by combining the form elements with the <OL> and <LI> tags. Here is a fragment of HTML and its associated layout.

```
<OL>
<li> Do you like playing computer games? <br>
<input type="radio" name="games" value="yes"> Yes
<input type="radio" name="games" value="no">No</li>
...
</OL>
```



Here is HTML code for extra questions. Note how the values have been abbreviated. The values are never seen by the user, but are for processing by software, typically a server-side script, and so do not need to be understood by the end users.

```
<p>1. Which of these foods do you prefer? <br>
```

```
<input type="radio" name="food" value="italian">Italian
<input type="radio" name="food" value="chinese">Chinese
<input type="radio" name="food" value="indian">Indian </p>
<p>2. Which car do you prefer? <br>
<input type="radio" name="car" value="ferrari">Ferrari
<input type="radio" name="car" value="astin">Aston Martin
<input type="radio" name="car" value="lotus">Lotus
<input type="radio" name="car" value="jaguar">Jaguar</p>
<p>3. Do you like music? <br>
<input type="radio" name="music" value="yes">Yes
<input type="radio" name="music" value="no">No</p>
<p>4. I enjoy reading?<br>


<input type="radio" name="read" value="sa">strongly agree
<input type="radio" name="read" value="a">agree
<input type="radio" name="read" value="n">neutral
<input type="radio" name="read" value="d">disagree
<input type="radio" name="read" value="sd">strongly disagree</p>
```

5.7.5 Discussion of Activity 5

Here is the HTML. Note that summing the totals of the selected items requires server-side software. (This is one of the reasons that JavaScript is used to allow such calculations before a form is submitted; see Unit 16.)

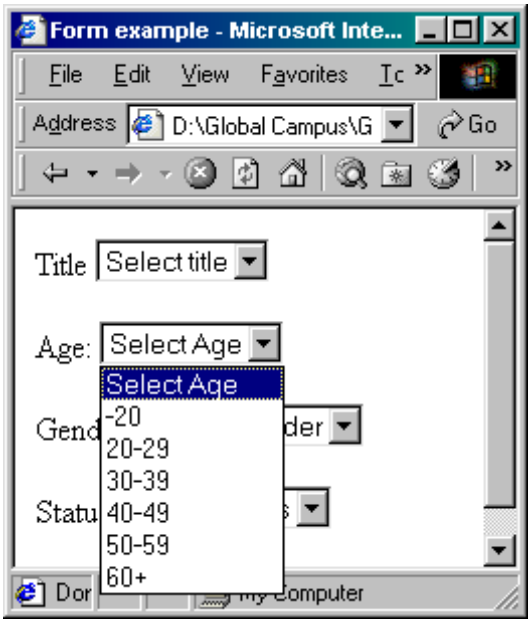
```
<TABLE>
<TR>
<TD>Simple digital watch ($40)></TD>
<TD><INPUT TYPE="checkbox" VALUE="40"></TD>
</TR>
<TR>
<TD>Chrome strap digital watch ($52)</TD>
<TD><INPUT TYPE="checkbox" VALUE="52"></TD>
</TR>
<TR>
<TD>Traditional gold watch ($107)</TD>
<TD><INPUT TYPE="checkbox" VALUE="107"></TD>
</TR>
<TR>
<TD>Antique pocket watch ($250)</TD>
<TD><INPUT TYPE="checkbox" VALUE="250"></TD>
</TR>
</TABLE>
```

5.7.6 Discussion of Activity 6

1. The HTML and the output for Title, Age, Gender, employment Status.

```
<FORM METHOD="post" ACTION=mailto:name@xyz.ac.uk>
Title
<SELECT NAME="title">
<OPTION>Select title</OPTION>
<OPTION>Mr</OPTION>
<OPTION>Mrs</OPTION>
<OPTION>Miss</OPTION>
<OPTION>Ms</OPTION>
<OPTION>Dr</OPTION>
```

```
</SELECT><BR><BR>
Age:
<SELECT NAME="age">
<OPTION>Select Age</OPTION>
<OPTION>-19</OPTION>
<OPTION>20-29</OPTION>
<OPTION>30-39</OPTION>
<OPTION>40-49</OPTION>
<OPTION>50-59</OPTION>
<OPTION>60+</OPTION>


</SELECT><BR><BR>
Gender:
<SELECT NAME="gender">
<OPTION>Select Gender</OPTION>
<OPTION>Male</OPTION>
<OPTION>Female</OPTION>
</SELECT><BR><BR>
Status:
<SELECT NAME="status">
<OPTION>Select Status</OPTION>
<OPTION>Employed</OPTION>
<OPTION>Unemployed</OPTION>
<OPTION>Student</OPTION>
</SELECT><BR><BR>
</FORM>
```

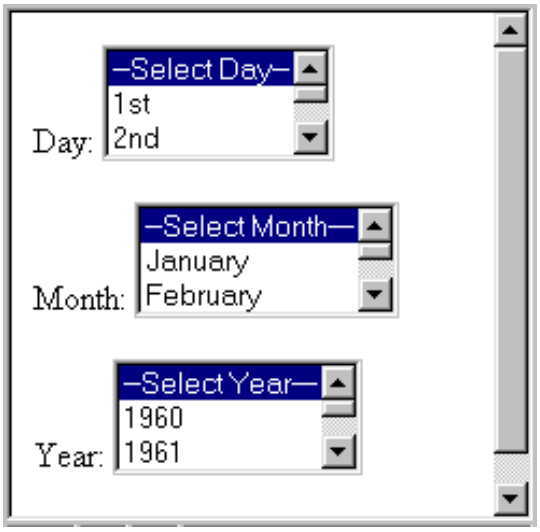


2. The HTML and the output for Title, Age, Gender, employment Status

```
<FORM METHOD="post" ACTION=mailto:name@xyz.ac.uk>
Day:

<SELECT name="occupation" size="3">
<OPTION SELECTED>--Select Day--</OPTION>
<OPTION>1st</OPTION>
<OPTION>2nd</OPTION>
<OPTION>3rd</OPTION>
<OPTION>4th</OPTION>
<OPTION>5th</OPTION>
<OPTION>6th</OPTION>
```

```
<OPTION>7th</OPTION>
<OPTION>8th</OPTION>
<OPTION>9th</OPTION>
<OPTION>10th</OPTION>
<OPTION>11th</OPTION>
<OPTION>12th</OPTION>
<OPTION>13th</OPTION>
<OPTION>14th</OPTION>
<OPTION>15th</OPTION>
<OPTION>16th</OPTION>
<OPTION>17th</OPTION>
<OPTION>18th</OPTION>
<OPTION>19th</OPTION>
<OPTION>20th</OPTION>
<OPTION>21st</OPTION>
<OPTION>22nd</OPTION>
<OPTION>23rd</OPTION>
<OPTION>24th</OPTION>
<OPTION>25th</OPTION>
<OPTION>26th</OPTION>
<OPTION>27th</OPTION>
<OPTION>28th</OPTION>
<OPTION>29th</OPTION>
<OPTION>30th</OPTION>
<OPTION>31st</OPTION>

</SELECT><BR><BR>

Month:

<SELECT name="select" size="3">

<OPTION SELECTED>--Select Month---</OPTION>
<OPTION>January</OPTION>
<OPTION>February</OPTION>
<OPTION>March</OPTION>
<OPTION>April</OPTION>
<OPTION>May</OPTION>
<OPTION>June</OPTION>
<OPTION>July</OPTION>
<OPTION>August</OPTION>
<OPTION>September</OPTION>
<OPTION>October</OPTION>
<OPTION>November</OPTION>
<OPTION>December</OPTION>

</SELECT><BR><BR>

Year:

<SELECT name="select2" size="3">

<OPTION SELECTED>--Select Year---</OPTION>
<OPTION>1960</OPTION>
<OPTION>1961</OPTION>
<OPTION>1962</OPTION>
<OPTION>1963</OPTION>
<OPTION>1964</OPTION>
<OPTION>1965</OPTION>
<OPTION>1966</OPTION>
<OPTION>1967</OPTION>
<OPTION>1968</OPTION>
<OPTION>1969</OPTION>
```

```
<OPTION>1970</OPTION>
<OPTION>1971</OPTION>
<OPTION>1972</OPTION>
<OPTION>1973</OPTION>
<OPTION>1974</OPTION>
<OPTION>1975</OPTION>
<OPTION>1976</OPTION>
<OPTION>1977</OPTION>
<OPTION>1978</OPTION>
<OPTION>1979</OPTION>
<OPTION>1980</OPTION>
<OPTION>1982</OPTION>
<OPTION>1983</OPTION>
<OPTION>1984</OPTION>
<OPTION>1985</OPTION>
<OPTION>1986</OPTION>
<OPTION>1987</OPTION>
<OPTION>1988</OPTION>
<OPTION>1989</OPTION>
<OPTION>1990</OPTION>
<OPTION>1991</OPTION>
<OPTION>1992</OPTION>
<OPTION>1993</OPTION>
<OPTION>1994</OPTION>
<OPTION>1995</OPTION>
<OPTION>1996</OPTION>

</SELECT><BR><BR>

</FORM>
```



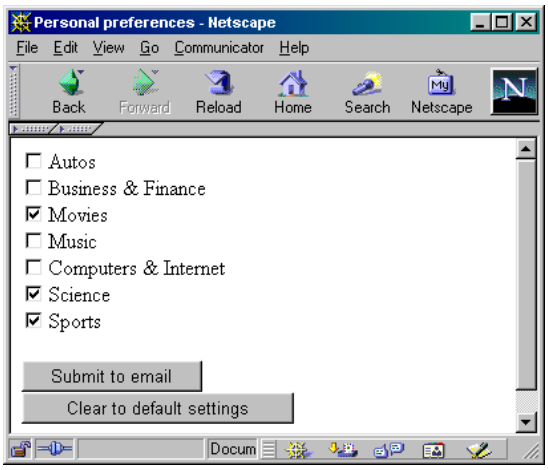
5.7.7 Discussion of Activity 7

The HTML code is as follows:

```
<FORM METHOD="post" ACTION="mailto:your-email@your.isp">
<INPUT TYPE="checkbox" NAME="autos" VALUE="yes">Autos<BR>
<INPUT TYPE="checkbox" NAME="b" VALUE="yes">Business & Finance<BR>
<INPUT TYPE="checkbox" NAME="movies" VALUE="yes">Movies<BR>
<INPUT TYPE="checkbox" NAME="music" VALUE="yes">Music<BR>
<INPUT TYPE="checkbox" NAME="comps" VALUE="yes">Computers & Internet
<BR>
<INPUT TYPE="checkbox" NAME="science" VALUE="yes">Science<BR>
```

```
<INPUT TYPE="checkbox" NAME="sports" VALUE="yes">Sports <BR><BR>
<INPUT TYPE="submit" VALUE="Submit to email"><BR>
<INPUT TYPE="reset" VALUE="Clear to default settings">
</FORM>
</BODY>
```

For example, say the boxes labelled 'Movies', 'Science' and 'Sports', were checked, as in the figure below.



With default encoding you would receive an attached file in your email containing:

```
movies=yes&science=yes&sports=yes
```

This can be verified with a text editor.

With **ENCTYPE="text/plain"** you would receive the following in your email:

```
movies=yes
science=yes
sports=yes
```

5.7.8 Discussion of Activity 9

Sample HTML and output are given below. Give the form a more presentable layout using either tables or CSS.

```
<!DOCTYPE html>
<html>
<head>
<title>
Job application form
</title>
</head>

<BODY>
    <div class="formLayout">
   <H1>SpeedyPC Job Application Form</H1>
   <HR><BR>
   <FORM METHOD="post" ACTION=mailto:chaombogho@gmail.com>

   <B>Position Applied for:</B> <INPUT type="text" name="position"
size="30" autofocus>

   Job reference code: <INPUT type="text" name="ref" size="15"
bgcolor = "yellow"><BR>

   <HR><BR>
```

```
   First Name:
   <INPUT type="text" name="firstname" size="30"
maxlength="30"><BR> Family Name:
   <INPUT type="text" name="familyname" size="30"><BR>

   Nationality:
   <INPUT type="text" name="nationality" size="30"><BR>

   Date of Birth:
   <INPUT id="dob" name="dob" type = "date"><BR>

   Address:<BR>
   <TEXTAREA name="address" cols="30" rows="5"></TEXTAREA>
   <BR>

   Telephone:
   <INPUT type="text" name="phone" size="30" maxlength="30">
   Email:
   <INPUT type="text" name="email" size="30" maxlength="30"
required /><BR>

   <HR><BR>

   <B>Educational History</B><BR>

   School/College, Course Studied, Qualifications Received, Grades:
   <TEXTAREA name="eduhistory" rows="6" cols="70"></TEXTAREA>

   <B>Employment History</B><BR>

   Employer, Date, Position/responsibilities:
   <TEXTAREA name="emphistory" rows="6" cols="70"></TEXTAREA><BR>


   <B>Personal Statement:</B><BR>
   <TEXTAREA name="statement" rows="6"
cols="70"></TEXTAREA><BR><BR>
   <HR><BR>

   <B>YEARS OF WORK EXPERIENCE:</B><BR>
   <input type="range" min="0" max="10" step="1" value="0">
   <HR><BR>

   <B>First Referee Details:</B><BR><BR> First name:
   <INPUT type="text" name="firstname1" size="30"> Family name:
   <INPUT type="text" name="surname1" size="30"><BR> Title:
   <INPUT type="text" name="title" size="8"> Occupation:
   <INPUT type="text" name="occ1" size="30"><BR> Relationship:
   <INPUT type="text" name="rel1" size="20"><BR> Address:<BR>
   <TEXTAREA name="address1" rows="4" cols="30"></TEXTAREA><BR>
Telephone:
   <INPUT type="text" name="phone1" size="30"> Email:
   <INPUT type="text" name="email1" size="30"><BR><BR>

   <B>Second Referee Details:</B><BR><BR> First name:
   <INPUT type="text" name="firstname1" size="30"> Family name:
   <INPUT type="text" name="surname1" size="30"><BR> Title:
   <INPUT type="text" name="title" size="8"> Occupation:
   <INPUT type="text" name="occ1" size="30"><BR> Relationship:
   <INPUT type="text" name="rel1" size="20"><BR> Address:<BR>
   <TEXTAREA name="address1" rows="4" cols="30"></TEXTAREA><BR>
Telephone:
   <INPUT type="text" name="phone1" size="30"> Email:
   <INPUT type="text" name="email1" size="30"><BR><BR>
   <HR><BR>
   <INPUT type="submit" name="Submit" value="Submit">
```

```
            <INPUT type="reset" name="reset" value="Reset">

            </FORM>
            </div>
          </BODY>

</html>
```

5.7.9 Discussion of Additional Activity

Your HTML should look something like this:

```
          For example, say you were to check the boxes labelled 'Movies',
          'Science' and 'Sports', as in the figure below.
          <HEAD>
          <TITLE>Land Of Boxes Form</TITLE>
          </HEAD>
          <H1><FONT SIZE="5" COLOR="coral">Land Of Boxes Order
          Form</FONT></H1>
          <BODY>
          <FORM METHOD="post" ACTION=mailto:name@xyz.co.za>
          <TABLE>
          <TR>
          <TD ALIGN="right">Name:</TD>
          <TD><INPUT TYPE="text" NAME="name" SIZE="35"></TD>
          </TR>
          <TR>
          <TD ALIGN="right">Email:</TD>
          <TD><INPUT TYPE="text" NAME="email" SIZE="35" VALUE=""></TD>
          </TR>
          <TR>
          <TD ALIGN="right">Address:</TD>
          <TD><TEXTAREA NAME="address" COLS="30" ROWS="5"></TEXTAREA></TD>
          </TR>
          <TR>
          <TD ALIGN="right">Postcode:</TD>
          <TD><INPUT TYPE="text" NAME="postcode" SIZE="20"></TD>
          </TR>
          </TABLE><BR><BR>
          <TABLE BORDER=1>
          <TR>
          <TD><B>Product</B></TD>
          <TD><B>Part Number</B></TD>
          <TD><B>Quantity</B></TD>
          <TD><B>Unit Price</B></TD>
          <TD><B>Subtotal</B></TD>
          </TR>
          <TR>
          <TD>EconoBox</TD><TD>LB100</TD>
          <TD><INPUT type="text" name="Qlb100" size="8"></TD><TD>R5</TD>
          <TD><INPUT type="text" name="Sublb100" size="15" value="R"></TD>
          </TR>
          <TR>
          <TD>Standard Box</TD><TD>LB200</TD>
          <TD><INPUT type="text" name="Qlb200" size="8"></TD><TD>R10</TD>
          <TD><INPUT type="text" name="Sublb200" size="15" value="R"></TD>
          </TR>
          <TR>
          <TD>Premium Box/TD><TD>LB300</TD>
```

```
<TD><INPUT type="text" name="Qlb300" size="8"></TD><TD>R15</TD>
<TD><INPUT type="text" name="Sublb300" size="15" value="R"></TD>
</TR>
<TR>
<TD>Deluxe Box/TD><TD>LB400</TD>
<TD><INPUT type="text" name="Qlb400" size="8"></TD><TD>R20</TD>
<TD><INPUT type="text" name="Sublb400" size="15" value="R"></TD>
</TR>
<TR>
<TD>Super Deluxe Box/TD><TD>LB500</TD>
<TD><INPUT type="text" name="Qlb500" size="8"></TD><TD>R30</TD>
<TD><INPUT type="text" name="Sublb500" size="15" value="R"></TD>
</TR>
<TR>
<TD COLSPAN=5 ALIGN="right">
Total: <INPUT type="text" name="total" size="15" value="R"></TD>
</TR>
</TABLE><BR><BR>
<B>Credit Card Details</B><BR>
<TABLE>
<TR>
<TD>Card Type:</TD>
</TR>
<TR>
<TD><INPUT type="radio" name="cardmake"
value="master">Mastercard</TD>
<TD>Card Number:</TD>
<TD><INPUT type="text" name="cardnumber" size="20"></TD>
</TR>

<TR>
<TD><INPUT type="radio" name="cardmake" value="visa">Visa</TD>
<TD>Expiry date:</TD>
<TD><INPUT type="text" name="expiry" size="10" maxlength="5"
value="mm/yy"></TD
</TR>

<TR>
<TD><INPUT type="radio" name="cardmake" value="amex">American
Express</TD>
</TR>

<TR>


<TD><INPUT   type="radio"   name="cardmake"   value="diners">Diners
Club</TD>
</TR>
</TABLE><BR><BR>

<TABLE>
<TR>
<TD ALIGN="right"><B>Delivery Address:</B></TD>
<TD><TEXTAREA NAME="delivery" COLS="25" ROWS="5"></TEXTAREA></TD>
</TR>
</TABLE><BR><BR>
<INPUT type="submit" name="submit" value="Send order">
<INPUT type="reset" name="reset" value="Clear form">
</FORM>
</BODY>
```

5.7.10 Answer to Review Question 1

HTML forms provide a way for the user to interact with a server.

5.7.11 Answer to Review Question 2

CGI software is software that adheres to the Common Gateway Interface protocol. A CGI programme is written for a special purpose and performs server-side processing of data submitted from a form.

5.7.12 Answer to Review Question 3

No, the **SIZE** attribute in an <INPUT> tag merely sets the size of the field that will be rendered by a browser. The

size of text is limited by the **MAXLENGTH** attribute.

5.7.13 Answer to Review Question 4

To ensure a horizontal scroll bar is included in a multi-line text field, you must add the <TEXTAREA> tag the attribute **WRAP=OFF**

5.7.14 Answer to Review Question 5

The problem here is that all the buttons have the name flight_prefs, which means that only one name/ value pair will be submitted where three are needed. Further, the first and last sets of radio buttons have unintentionally been made mutually exclusive, again because of the common name

5.7.15 Answer to Review Question 6

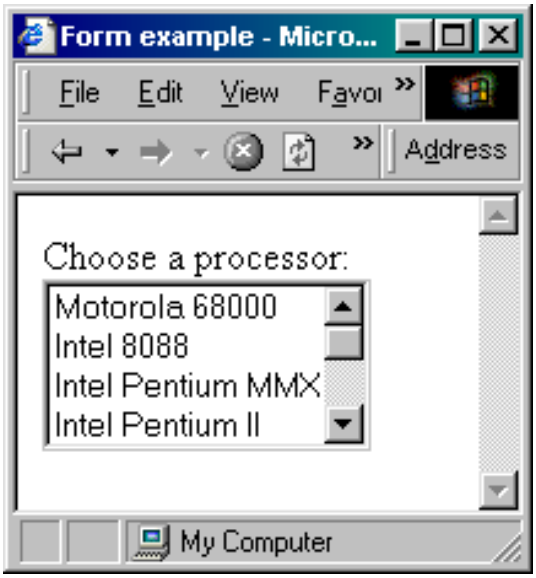
To initially set a check box, include the **CHECKED** attribute in the <INPUT> tag.

5.7.16 Answer to Review Question 7

Menu buttons require only one line when rendered. The list of options appears over (usually) whatever is below it on the page. Scrolling lists occupy as many lines as specified by the **SIZE** attribute of the <SELECT> tag; since all the options are rarely shown, fewer lines than options are usually used.

5.7.17 Answer to Review Question 8

Output and HTML are given below:




```
Choose a processor:<BR>
<SELECT name="processor" SIZE=4>
<OPTION>Motorola 68000</OPTION>
<OPTION>Intel 8088</OPTION>
<OPTION>Intel Pentium MMX</OPTION>
<OPTION>Intel Pentium II</OPTION>
<OPTION>Intel Pentium III</OPTION>
<OPTION>Intel Celeron</OPTION>
<OPTION>PowerPC G3</OPTION>
<OPTION>PowerPC G4</OPTION>
<OPTION>AMD Athlon</OPTION></SELECT>
```

5.7.18 Answer to Exercise 1

The following name/value pair will be sent to the CGI software. An ampersand (&) would precede or follow it if other pairs were sent.

```
age=39
```

Two layout tips are worth remembering:

1. Use tables to layout the form in an organised way (see Unit 4).
2. You can use the
 tag (no closing tag needed) to move text or form objects to the next line. You can use the <P></P> tags to space your work in paragraphs.

5.7.19 Answer to Exercise 2

Your HTML code should look something like this:

```
What is your favourite movie?<BR>
<TEXTAREA NAME="movie-comments" COLS="50" ROWS="5"
WRAP=OFF></TEXTAREA>
```

5.7.20 Answer to Exercise 3

There may be conflicting requirements here. It is not easy to reflect the fact that the usual destination for customers is within Egypt while promoting the others. For example, a scrolling list with Egypt at the top would appear to satisfy this requirement, but the other countries would not be visible. Breaking alphabetical ordering might distract some users:


```
Destination Country:<BR>
<SELECT NAME="colour" SIZE="3" MULTIPLE>
<OPTION>Egypt</OPTION>
<OPTION>Bahrain</OPTION>
<OPTION>Kuwait</OPTION>
<OPTION>Lebanon</OPTION>
<OPTION>Oman</OPTION>
</SELECT>
```

Alternatively, you could use alphabetical ordering to force the user to scroll through at least the options that precede the most likely destination, Egypt. But there would only be one option before Egypt, so it could be moved to last. To help users find Egypt, it can be pre-selected:


```
Destination Country:<BR>
<SELECT NAME="colour" SIZE="3" MULTIPLE>
<OPTION>Bahrain</OPTION>
<OPTION>Kuwait</OPTION>
<OPTION>Lebanon</OPTION>
<OPTION>Oman</OPTION>
<OPTION SELECTED>Egypt</OPTION>
</SELECT>
```

The final alternative is to use a menu button: it can preserve alphabetic ordering and shows all the other options:


```
Destination Country:<BR>
<SELECT NAME="colour">
<OPTION>Bahrain</OPTION>
<OPTION>Egypt</OPTION>
<OPTION>Kuwait</OPTION>
<OPTION>Lebanon</OPTION>
<OPTION>Oman</OPTION>
</SELECT>
```

Note that, when using a menu button, you should not pre-select Egypt if you want customers for internal flights to see the other options. Pre-selecting does exactly that: it would leave only Egypt visible on the button and most customers would not click on the button to reveal the other options in the menu.

Chapter 6. HTML5 iFrames

Table of Contents

Objectives

At the end of this chapter you will be able to:

- Understand the use of iframes;
- Create frames using some of the iframe elements.

6.1 Introduction to iframes

The basic frames are deprecated in HTML5 and are out of use. The `frameset` tag and its helper tags `frame/noframes` are removed from the modern HTML5 standard. The basic frames had this markup:

```
<FRAMESET COLS="20%,80%">
<FRAME src=left.html>
<FRAME src=right.html>
</FRAMESET>
```

This sets up the frameset to consist of two frames in two columns. The first frame takes 20% of the browser window and the second 80%. A file called left.html will appear in one frame and a file called right.html in the other. In order to view the framed page in your browser you will need to create these two pages.

HTML5 incorporates the inline frame element, **iframe,** which is a HTML page embedded into the current page.

6.2 iframes Elements

6.2.1 Web page with one or multiple frames

Activity 1: Create a web page with a single Frame

This Activity sets up a webpage with a single frame.

1. Open a text editor, such as Notepad, and enter the <HEAD> and <BODY> portions of an HTML page.

2. . Use the HTML code below to lay out the page with one frame that embeds the UCT website:

```
<iframe src="http://www.uct.ac.za/"></iframe>
```

Save this page as frame1.html and load it in your web browser.

Activity 2: Create a web page with a multiple frames

This Activity sets up a webpage with two frames.

1. Open a text editor, such as Notepad, and enter the <HTML> and <BODY> portions of an HTML page.

```
<iframe src="right.html"></iframe>
<iframe src="left.html"></iframe>
```

Save this page as frame2.html.

2. Begin another document and enter the following code. Enter the following code:

```
<BODY>This is the left frame</BODY>
```

Save this file as left.html in the same folder as file2.html.

3. Begin another document and enter the following code. Save this as right.html. Note that the `bgcolor` tag was deprecated in HTML5 and now we use the CSS style.

```
<head>
<style>
BODY
{
      background-color:blue;
}
</style>
</head>

<BODY>
This is the right frame with a blue background
</BODY>
```

4. Now, load frame2.html in your Web browser. You should see that both frames, with the two files loaded in them as appropriate.

6.2.2 Set Width and Height

Use the height and width attributes to specify the size.

The attribute values are specified in pixels by default, but they can also be in percent (like "80%"), for example:

```
<iframe src="right.html" width = 200 height = 100></iframe>
```

6.2.3 Remove the Border

By default, an iframe has a black border around it.

To remove the border, add the style attribute and use the CSS border property:

```
<iframe src="right.html" style = "border:none"></iframe>
```

6.2.4 Use iframe as a Target for a Link

An iframe can be used as the target frame for a link.

Activity 3: Target iframe

1. In this activity you will create a link that, when clicked, will open the UCT website in a frame.

2. Open frame2.html and make the following changes.

```
<iframe src="right.html" name = "right"></iframe>
<p><a href="http://www.uct.ac.za/" target="right">Go to UCT</a></p>

<p>Click on the link above to open UCT website in the right frame</p>
```

3. Save the file and load in in your browser. Click on 'Got to UCT' and the UCT website should load within the right frame.

6.3 Advantages and disadvantages of iframes

The major disadvantages of using iframes are:

- Frames can make the production of a website complicated, although current software addresses this problem.
- It is easy to create badly constructed websites using frames. The most common mistake is to include a link that creates duplicate Web pages displayed within a frame.
- Search engines that reference a Web page only give the address of that specific document. This means that search engines might link directly to a page that was intended to be displayed within a frameset.
- Users have become so familiar with normal navigation using tables, the back button, and so on, that navigating through a site that uses frames can be a problem.
- The use of too many frames can put a high workload on the server. A request for, say, ten files, each 1 Kilobyte in size, requires a greater workload than a request for a single 10 Kilobyte file.

The advantages of HTML5 iframes include:

- The main advantage of frames is that it allows the user to view multiple documents within a single Web page.
- It is possible to load pages from different servers in a single frameset.

To Do

Research iframes and their uses. Discuss the following topics with other students on the forum:

- The merits of designing a Web page with and without iframes.
- The alternatives to using iframes.

Try to cite some examples of good and poor website design using frames of your own.

Chapter 8. Style Sheets

Table of Contents

Objectives

At the end of this chapter you will be able to:

- Explain the advantages and disadvantage of using stylesheets;
- Use CSS to create web pages.

8.1 Introduction to Style Sheets

There is no format to follow for teaching the aesthetics of style - most people, though, can recognise something that follows a classical design. But some things can be said about the style of a website. For instance, when Web pages belong to the same website, each page should have a consistent look in order to provide familiarity for the user.

Style sheets (sometimes referred to as templates) are used in desktop publishing to provide consistency when formatting text. The format applied by the stylesheet could be to indent every first line of a paragraph by 2cm, insert a page break at the end of every chapter, and so on. Naturally, due to multimedia, Web pages not only have to consider text formatting, but also visual and sound presentation, and various multimedia formats in general. Before we continue, let us briefly discuss the advantages and disadvantages of using style sheets.

8.1.1 Advantages of Style Sheets

1. **Multiple Styles** - A single document can be presented in multiple styles by using multiple style sheets.

2. **Re-styling** - The use of style sheets (which are separate to the HTML files) allows the quick re- styling of any document, without modifying the original HTML.

3. **Document maintenance** - The ability to re-style many documents allows us to easily make changes to the appearance of many Web pages without separately editing each one.

4. **Consistency** - Style sheets guarantee consistency throughout website.

5. **Optimal file size** - The smaller the files the faster the download. Using style sheets can help minimize file sizes, since, for example, every < *font* > tag, is defined in one place in a style sheet, rather than in multiple places in the HTML file.

6. **Style and structure** - When first developed, HTML was only concerned with document markup and not with the document's formatting. This eventually changed, with more and more functionality being added to HTML to allow for formatting. With the introduction of style sheets, the HTML document is again concerned only with structural document markup — all formatting is now placed in the style sheet.

Exercise 1

Visit the UCT Computer Science Department website [http://www.cs.uct.ac.za] and suggest how the above advantages of style sheets can be used. You can find some discussion on this at the end of this chapter.

8.1.2 Disadvantages of Style Sheets

1. **Browser dependency** - Style sheets format things slightly differently on different browsers. Unfortunately, browsers have different support for HTML and style sheets. Newer browsers have largely converged on HTML support so that HTML documents look the same across different browsers. The state of style sheet support is somewhat worse, largely because style sheets are newer than HTML.

2. **Old Browsers** - Some very old browsers (such Netscape Navigator 2) do not support style sheets. All in all, style sheets have only minor disadvantages, and should be used when developing websites.

To Do

Read up on style sheets on your text books. Can you see any advantages or disadvantages that have not been presented here? Also use the Internet to find out more about the usage of style sheets. A good starting place is the W3C's section on style sheets [http://www.w3.org/Style/].

8.2 CSS

The style sheet standard supported by modern browsers is called cascading style sheets, or CSS. CSS files contain a set of rules for the formatting of HTML documents. An example is given below:

```
<html>
<head>
<title>UCT MSc IT Example 1 on style sheets</title>
<style>
    BODY {

    font-family : "times new roman;
    margin-left : 20%;
    margin-right: 20%;
    text-align : justify;
    background : ivory;
    color : black;
    }

    P  {
    text-indent : 2cm;
    }
</style>
</head>
```

A style sheet is a collection of rules that describe the format of the HTML tags. In the above example there are six

rules describing the format of the BODY tag, and one rule for the P tag. There are two ways:

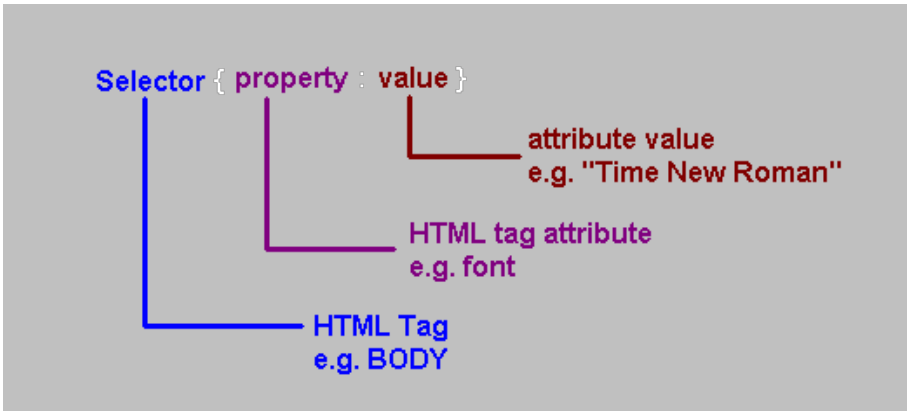
to write style sheets: the technically easier rule-based approach, and an approach that procedurally constructs a style sheet — such an approach is outside the scope of this unit, but feel welcome (if you any spare time) to visit various sites on this topics such as this one[http://csgrs6k1.uwaterloo.ca/~dmg/dsssl/tutorial/tutorial.html] or search on Google [http://www.google.com] for more

Below is a description of the rules used in the above example.

Body - *bgcolor* set to "ivory", left and right margins indented relatively by 20%, font set to "Times New Roman" and text colour set to black.

P to indent the first line by and absolute value of two centimeters.

There are three parts to a style sheet rule, shown in the figure below:



8.3 Important Note About Rules

Notice in the given example how each rule is separated by a semi-colon (;). If your code is not working, ensure that the semi-colons are present.

There are three ways to apply the above CSS rules to an HTML file:

1. add them in-line to the HTML file itself, attached directly to the relevant HTML tag.
2. embed the rules into the HTML file
3. link the CSS file to the HTML file

The next section will look at each of these methods.

Exercise 2

Add a style sheet property to the given example to change the text color to dark red. You can find the solution at the end of the chapter.

8.4 In-line Styles

In-Line styles are added to individual tags and are usually avoided. Like the FONT tag they clog up HTML documents, making them larger and increasing their download times.

An example of an in-line style is given below:

```
<P style="text-indent: 2cm; color:darkred;">
This paragraph has been formatted using the in-line style command.

</P>
```

```
<P>
This paragraph has not been formatted using the in-line style
command.
>/P>
```

8.5 Embedded Style sheets

This method avoids duplication within a single HTML document. However, it still has its drawbacks: every Web page on your site needs this embedded style sheet inserted; consequently any updates to the style sheet have to be made to every HTML document that has the style sheet embedded in it. We have already used embedded style sheets as they are the simplest to implement, here is another example:

```
<html>
<head>
<title> University of Cape Town, Example on embedded style
sheets</title>
<sty
le>
BODY
{
font-family : "times new
roman; margin-left : 20%;
margin-right:
20%; text-align
: justify;
background :
ivory; color :
darkred;
}
P {
text-indent : 2cm;
}
h1,h2,h3{
color:red
; margin-
left:2cm
}
</style>
</head>

.
.
.

</html>
```

Exercise 3

The above code has a deliberate mistake in it. Can you find it? You can find the corrected version at the end of the chapter.

Notice how rules for the h1, h2, and h3 tags can be simultaneously defined. Embedded style sheets should always be placed within the HEAD — preferably before the TITLE (i.e. not like in the above example).

8.6 Imported Style Sheet

This gives you all the advantages of style sheets: by changing a single value in one file the format

change is propagated to all of the HTML documents linked to the style sheet. The style sheet is written just as an embedded style sheet is, but, instead of inserting it in an HTML file, it is saved as a separate file (usually with a .css extension). Each HTML document then imports the CSS file. There are two ways to import a file:

Linking it

```
<              link              rel="stylesheet"              href="
..//pathname//stylesheet_filename.css" type="tex
```

Importing it

```
<style>
@import (http://pathname/stylesheet.css);
</style>
```

The second option will only work for HTML documents on a Web server. If this is not the case, use the first option.

Exercise 4

From the in-line style listing below, create an embedded style sheet and a linked style sheet.

```
< h1 style="color:red;margin-left:2cm" >h1
heading</h1>
< h2 style="color:red;margin-left:4cm" >h2 sub-
heading</h2>
< p style="text-indent:2cm;color:darkred;margin-
left:6cm">Paragraph 1</p>
< p style="text-indent:2cm;color:darkred;margin-
left:6cm">Paragraph 2</p>
< h2 style="color:red;margin-left:4cm" >h2 sub-
heading</h2>
< p style="text-indent:2cm;color:darkred;margin-
left:6cm">Paragraph 1</p>
```

You can find a solution to this exercise at the end of this chapter.

8.7 Classes

Sometimes you may wish to give different formats to different paragraphs in the same HTML document. This can be accomplished by using classes.

A classic example is the formatting of publications in journals, where the leading paragraph is an abstract and has the following format:

Title: 16pt, Arial, bold, centered

Authors: 14pt, Arial, bold, centered

abstract : 12 pt Times New Roman, italic, justified left, left margin 4cm, right margin 6cm first line indent 1cm

Section Headings: Arial 14pt, bold, first line indent 1 cm

Sub-Section Headings:Arial 12pt, bold, first line indent 1 cm

body text: Times New Roman, 12 pt, first line indent 1 cm, left margin 3cm, right margin 3cm

There are also many different formats for various other paragraphs. Each different paragraph type can be given its own class name, such as "abstract". To do this, we use the P tag as shown in the code below. Open a new file and type in the following code and save it as csexercise.css

```
p.abstract{
margin-left:4cm;
margin-right:4cm;
font-style:italic;
font-family:"times New roman";
font-size : 12pt;
text-indent : 1cm;
text-align : left;
}
.body{
margin-left:3cm;
margin-right:3cm;
font-family:"times New roman";
font-size: 12pt;
text-indent: 1cm;
}
```

Open a blank document and import the above style sheet. The abstract class is used as shown in the code below:

```
<P CLASS="abstract">
```

Exercise 5

As an exercise, add code to the stylesheet (csexercise.css) for the title, author, section heading and subsection heading. You can find the solution at the end of this chapter.

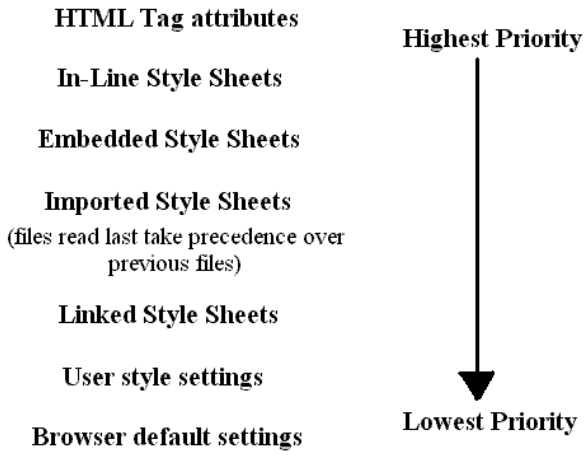
To Do

Visit Webmonkey's article "Inject Some Style (Sheets) into your HTML" [http://webmonkey.wired.com/webmonkey/html/96/34/index2a.htm]. Also visit the sites referred to at the end of the article.

8.8 Cascading Style Sheets

The cascading style sheet standard supplies very powerful tools to control Web page formatting. For instance, consider a university with many departments — each with their own individual design criteria — that is producing a website. It is possible to create a hierarchy of style sheets that allows each department's website to maintain formatting consistency with all the other university sites, while allowing each department to deviate from the format where needed.

The hierarchical (cascading) structure of style sheets can be used to do this. The figure below illustrates the style hierarchy design by W3C.



To Do

Visit John Allsop's complete CSS Guide [http://www.westciv.com/style_master/academy/ css_tutorial/index.html] for more information on style sheets.

Activity 1

You are designing a Web page to display pop song lyrics (naturally the copyright has been permitted). The information needs to be displayed on the page in the style presented below. Use only the h1, h2, body and p HTML tags; note that two or more classes are needed for some of these tags.

- **Artist:** 14pt, Arial, Bold, Moccasin
- **Song Title:** 14pt, Arial, Normal, Khaki
- **Album:** 12pt, Arial, Normal, Khaki
- **Year:**12pt, Arial, bold, ivory
- **Background:** colour=Chocolate
- **Verse:** 14pt, Trebuchet MS, Bold, gold, margin-left 2cm, margin-right 2cm
- **Chorus:** 14pt, Trebuchet MS, Bold & Italic, gold, margin-left 4cm, margin-right 4cm

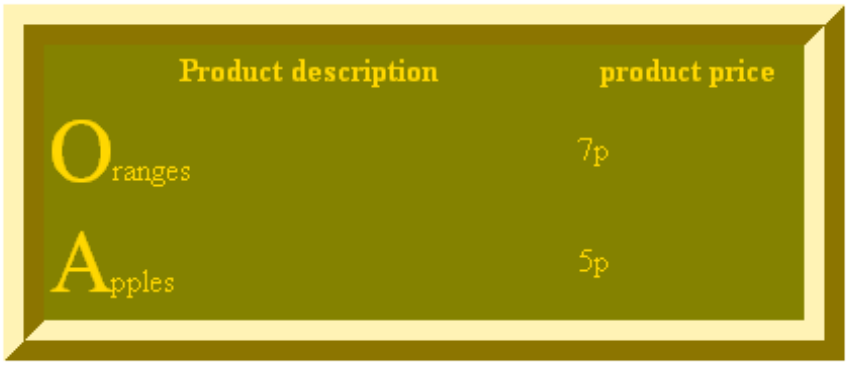
A solution can be found at the end of the chapter.

Activity 2

Use the style sheet from Activity 1 to create several Web pages that *import* the style sheet.

Activity 3: Drop-cap lettering & Tables

A company wants you to design a similar website to that of Activity 1. This company wants the Web page to display their products prices in the format listed below:



The first letter example uses the span tag and is detailed below. Create a class fl with

the following attributes:

```
span.fl{font-size:32pt;font-family:"brush script mt";}
```

Place the required letter between span tags using the above class, as shown below:

```
<span class="fl">A</span>pples
```

8.9 Review Question

8.9.1 Review Question 1: Reflection on Style Sheets

Read the following case study and give reasons for asking certain question in relation to style sheets.

A school wishes to design a website. The function of this website is to let its students know when assignments are due, the dates of exams, timetables, subjects taught, syllabuses and the teachers who teach them.

The school wishes to have quality controls over the Web design, since the teachers themselves will be responsible for updating the information on the Web.

1. The school has approached you to advise them on how this may be possible.

2. The school wants you to explain the process you wish to use, and why it is better than letting every teacher write their own style.

3. Write a list of ten or more HTML tags and classes that may be used for the website. Consider the attributes that may have to be used, such as background colour, departmental style, tables (for timetable, exam, curriculum and annual), headers and so on.

4. After choosing the HTML tags, consider the CSS code required to style them.

Look at the end of the chapter for a discussion of this review question.

8.10 Discussions and Answers

8.10.1 Discussion of Exercise 1

Style sheets can be used to make the formatting of the site more consistent. This consistency enables the

user to recognise that they are on the Computer Science site no matter what page they are on.

Style sheets also enables the site to be more easily maintained. For instance, if it was decided that the indentation was too large, the indentation on each of the site's pages can be changed by modifying just one document: the site's style sheet.

8.10.2 Answer to Exercise 2

```
P {
 text-
 indent :
 2cm; color
        :
 darkred;
}
```

8.10.3 Answer to Exercise 3

```
BODY {
font-family : "times new
roman"; margin-left :
20%;
margin-right:
20%; text-
align :
justify;
background :
ivory; color
: darkred;
}
P {
text-indent : 2cm;
}
h1,h2,h3
{
color:re
d;
margin-
left:2cm
;
}
```

8.10.4 Solutions to Exercise 4

embedded:

```
<h1 style="color:red;margin-left:2cm">h1 heading</h1>
<h2 style="color:red;margin-left:4cm">h2 sub-heading</h2>
<p style="text-indent:2cm;color:darkred;margin-left:6cm">Paragraph
1</p>
<p style="text-indent:2cm;color:darkred;margin-left:6cm">Paragraph
2</p>
<h2 style="color:red;margin-left:4cm">h2 sub-heading</h2>
<p style="text-indent:2cm;color:darkred;margin-left:6cm">Paragraph 1      </p>
```

Linked - html file

```
<html>
<head>
<link rel="stylesheet" href="ex4.css" type="text/css">
</head>
<body>
<h1>h1 heading</h1>
<h2>h2 sub-heading</h2>
<p>div.abstract </p>
<p>Paragraph 1</p>
<p>Paragraph 2</p>
<h2>h2 sub-heading</h2>
<p>Paragraph 1</p>
<body>
<html>
```

Linked - ex4.css

```
p.abstract{
margin-left : 4cm;
margin-right : 4cm;
font-style : italic;
font-family : "times New roman";
font-size : 12pt;
text-indent : 1cm;
text-align  : left;
}
.body{
margin-left : 3cm;
margin-right : 3cm;
font-family : "times New roman";
font-size : 12pt;
text-indent : 1cm;
text-align : left;
}
```

8.10.5 Solutions to Exercise 5

New csexercise.css

```
h1{
font-family : arial;
font-size : 16pt;
font-style : bold;
Text-align : center;
}

h2
{
font-family : arial;
font-size : 14pt;
font-style : bold;
Text-align : center;
}
```

```
h3.sectionHeading
{
font-family : times new roman;
font-size : 14pt;
font-style : bold;
Text-align : left;
text-indent : 1cm;
margin-left : 3cm;
margin-right : 3cm;
}

h3.sectionSubHeading
{
font-family : times new roman;
font-size : 12pt;
font-style : bold;
Text-align : left;
text-indent : 1cm;
margin-left : 3cm;
margin-right : 3cm;
}

div.abstract{
margin-left : 4cm;
margin-right : 4cm;
font-style : italic;
font-family : "times New roman";
font-size : 12pt;
text-indent : 1cm;
text-align  : left;
}
.body{
margin-left : 3cm;
margin-right : 3cm;
font-family : "times New roman";
font-size : 12pt;
text-indent : 1cm;
text-align : left;
}
```

8.10.6 Solutions to Activity 1

```
body{background-color:chocolate;
font-family:"arial";
font-weight:bold;}

h1.artist{color:moccasin;font-size:14pt;}
h1.songtitle{color:khaki;font-size:14pt;font-
weight:normal;} h1.album{color:khaki;font-size:12pt;font-
weight:normal;} h1.year{color:ivory;font-size:12pt}

p.verse{font-family:"trebuchet ms";
color:gold;
margin-left:2cm;
margin-right:2cm;
font-weigth:bold;

}
p.chorus{font-family:"trebuchet ms";
```

```
font-weight:bold;
font-style:italic;
color:gold;
margin-left:2cm;
margin-right:2cm;
}
```

8.10.7 Discussion of Review Question 1

The following answers are a guide only — you may have listed answers not included here. If this is the case, start a discussion forum "Why Use style sheets?" and compare answers with other colleagues.

- You advise the school that cascading style sheets should be used. Your reasoning could include?
  - Cascading style sheets reflects the school's structure. In other words, the school and its various departments can each tailor the style sheet (using a hierarchy of style sheets) to suite their own image, thereby providing some control and consistency over the overall layout without being too restrictive on each department's Web page design.
  - If the school were to change its image in the future, style sheets provide a quick and easy way to do so.
  - If a design change is required during the development of the website, then style sheets provide a quick and easy method to do so.
- You mention that if every teacher were to write their own style, the school's image would be lost. Further, updating these separate designs to meet some future design standard is an arduous task.
- HTML tags:
  - p, for text presentations.
  - table.exam, for exam timetables.
  - table.time, for student timetables.
  - table.year, for yearly timetables.
  - body, the style for the main page, which will be inherited by other pages.
  - body.english, the style for the english department.
  - body.maths, the style for the mathematics department.
  - h1, headers.
  - h2, other headers.