

# WEB PAGE DESIGNING 9

## USING HTML

### Objectives

After completing this Chapter, the student will be able to:

- recognise HTML tags and their attributes,
- use HTML tags and attributes to develop web pages,
- use HTML tags and attributes to format text and images, design layouts, present tables and lists and create hyperlinks,
- use HTML forms to add interactivity to web pages, and
- design a web site and construct it using HTML.

*“Logic will get you from A to B. Imagination will take you everywhere.”*

*Albert Einstein*

### Introduction

As we have learnt earlier, the Internet is a very large network of computers spread across the world. The Internet allows for each of these computers to send and receive information from each other. One of the major applications of the Internet is the World Wide Web (WWW). The World Wide Web or the web in short, can be thought of as an interconnected set of documents, images, audio, video files or software files.

When you connect to the Internet and look around (we call it browsing), you are using a very interesting feature of the web, hyperlinks. Each time you click on a highlighted piece of text or image, you jump to another piece of text or image and this could be on the same page, another page on the same hard disk or on a page residing on another computer, half way across the world.

How this works and how we could make it happen is what we will learn in this chapter. Web pages use a language called the Hyper Text Markup Language (HTML). The browser

applications (Microsoft's Internet Explorer, Mozilla's Firefox, etc.) are designed to interpret HTML to display text, images and using hyperlinks, navigate the web.

## 9.1 INTRODUCTION TO HTML

Let us begin by looking at a web page (Figure 9.1) and analysing its features. This web page has the following major sections :



Figure 9.1 : Our web page

- **a header area** : In our example, it consists of a mast head, which signifies the subject of the web page. It may also contain a picture, a logo, etc., depending on the purpose of the page.
- **a menu bar** : This is a set of links, designed as buttons. Each link takes you to a separate section of the web page. In a way, this is like the chapter headings in a book.

- **body of the page** : This part of the page contains all the content that you wish to present. The content could be of a variety of types – text, images, tables, forms, etc. We have organised the page in two columns. While the left column contains text, formatted in different ways, the right column contains some features unique to a web page – links, images which link to an image gallery, a comment form and a poll. It is such features which make a page interactive.
- **a footer area** : This is at the bottom of the page and usually contains disclaimers, copyright information, etc. In our case, the footer text states, “All material on this site can be freely downloaded, distributed or reused”.

We will analyse the page and its sections further. We will also learn how to construct such web pages. In order to extend your knowledge, do look at a variety of web pages, each time noting down the layout and categorisation of content. Web page design has evolved into a very sophisticated activity, where artists and programmers work together to put together very elaborate features in order to make web pages interactive.

As we mentioned in the introduction, you need an application software called an Internet browser to view web pages. While all browsers are designed to interpret HTML and display web pages, they can be designed for different operating systems. You can have many browsers installed on your computer and use them simultaneously. Most browsers are free and can be downloaded from the web. Some popular browsers are listed below :

| <b>Browser</b>    | <b>Operating System</b>                   |
|-------------------|---|
| Mozilla Firefox   | Microsoft Windows, Linux or Apple's MacOS |
| Internet Explorer | Microsoft Windows                         |
| Google's Chrome   | Microsoft Windows                         |
| Opera             | Microsoft Windows                         |
| Safari            | Microsoft Windows and Apple's MacOS       |

During the course of this chapter, you will encounter a number of browser commands. It would be unwieldy to refer to the commands in each kind of browser. Therefore, we will use Mozilla's Firefox and refer to its commands. Every browser will have commands which will achieve similar purposes. They may however be placed in a different category, or called something else. So, when we mention a command, do check out how you will do a similar thing in Internet Explorer, Chrome or Opera.

Every HTML file (also referred to as a web page) is a text file. It does not contain images, sound, or such information. The HTML file has

references to such files. So when you see a photograph on the web page, the HTML file has called the image file and displayed it in the appropriate place on the web page. The fact that this is a text file, makes it platform independent – meaning that the same web page could be seen on a Windows, Unix, Linux or a MacOS computer. This is also the reason why the web is emerging as a universal medium of communication.

We mentioned that what traverses the web is the HTML code. It is this code that helps a browser construct the web page. But, how do you see the HTML code? Right-click on the page and from the options displayed, select view page source. You should now see the HTML code. This may look very threatening, but let us assure you that it is actually quite simple and you should be able to write it by the end of this chapter.

## 9.2 HTML TAGS

What does your HTML code contain? It contains the contents of the web page, information about how and where to display each of the content elements, and if you input information or click on a link, what actions to take. In other words, a HTML page contains the content, the programming and the design elements.

The basic element of the HTML code is a tag. Let us look at the code of our web page (Figure 9.2).

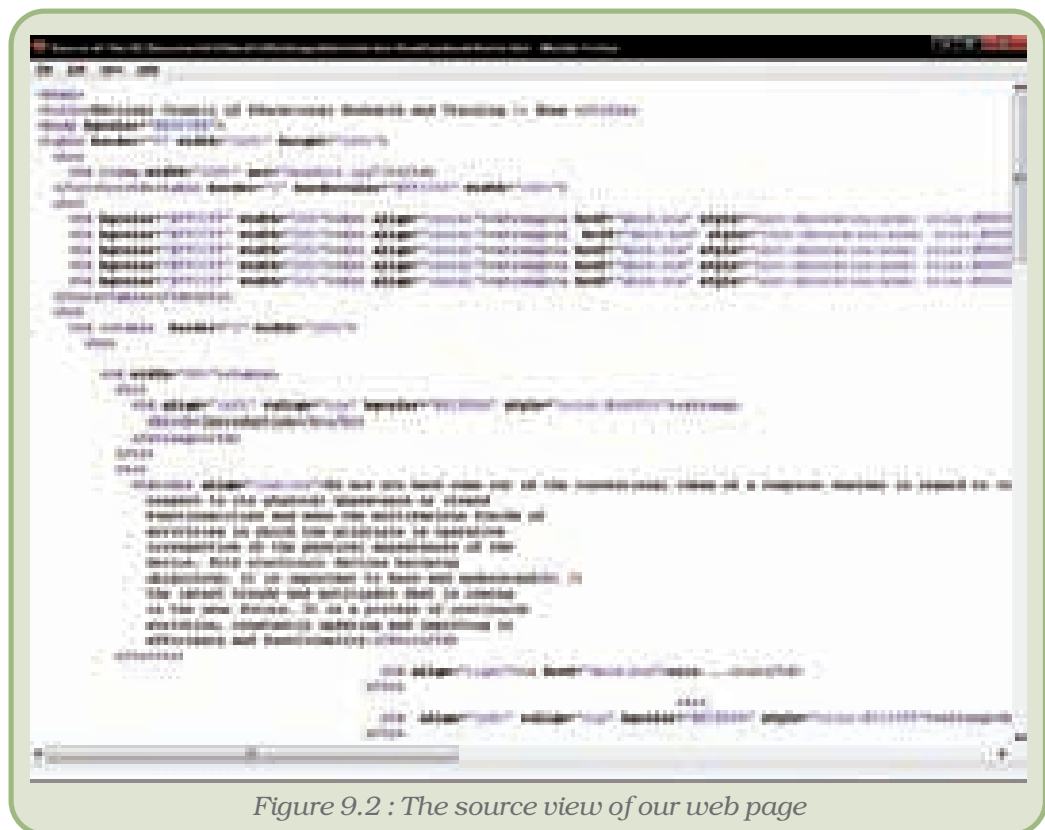


Figure 9.2 : The source view of our web page

Notice the code <HTML> at the top of the page. If we scroll down to the bottom of the page, we will see the code </HTML>. We refer to such codes as **tags**. Learning to make web pages is all about learning to use these tags to present content the way you want. The general format for a HTML tag is :

```
<tag_name> affected text </tag_name>
```

The opening tag consists of just the tag name and it instructs the web browser to start tagging (formatting) the text. The ending tag includes a slash preceding the tag name. This “/” slash tells a web browser to stop tagging the text. Let us look at a few examples:

If you want your text to be highlighted, say using bold type face, you use

```
<b> HTML tags </b>
```

and what you get on the web page is **HTML tags**. Notice that the tag is not case sensitive, that is, <B> HTML tags </B> would have yielded the same result. An interesting aspect of a HTML tag is that if the browser does not know what to do with a given tag, it will simply ignore it!

There are two types of tags :

- **Container tags** : The HTML tags which require a closing tag are called container tags. These tags are also called *paired* or *wrapper* tags. The <b> text </b> is one such pair. If you do not close such tags, all the following text would be treated as if they are part of this tag.
- **Non-container tags** : The HTML tags which do not require a closing tag are called non-container tags. These tags are also called *stand alone* or *empty tags*. A typical example is that of <br>. This tag inserts a line break.

### 9.2.1 ATTRIBUTES

Some tags are complete in themselves. In other words, they have only one option. For instance, when you want your text, say italicised, you use <i> text </i>. But if you wish to display text, say in some particular font, how will the tag look then?

```
<font face="Arial"> text</font>
```

Notice the tag in this example is <font> </font>. What about the remaining part, face= “Arial” We refer to this additional information as the tag’s attribute. So, in our example, the tag is font, its attribute is face and the value given to the attribute is Arial.

Attributes may have many different values you may like to use simultaneously. For instance, you may not only like a particular font,

but also a particular size of letter and in a particular colour. How do you do this? The tag would now be

```
<font face="Arial"; size=14px; color=#ff0000">text </font>
```

Note the use of quotes to indicate values and semi-colons to separate attributes. The general form of the tag is :

```
<tag_name attribute-name = "attribute-value">....</ tag_name>
```

As a general rule, most attributes – that include only letters, digits, hyphens, or periods – work fine without quotes. Attributes that have other characters, such as spaces, % signs, or # signs, however, *always* require quotes. Nevertheless, it is a good habit to enclose all attribute values in quotes.

The purpose of the attribute is to extend the scope of a tag. It provides additional information about the tag, to the browser.

How does the browser treat a tag, which requires an attribute, but no attribute is mentioned. For instance in our example if we had just used `<font>text</font>`, how would the browser show it. The best way to figure this out is to try it out yourself.

Each browser has a default setting, which is the value it takes when no other value is defined. In Windows Internet Explorer, the default font setting is “font:Times New Roman; font-size:12pt and color:#000000. So if you only have the `<font>text</font>` tag without any attributes, then the text would be in 12 point, Times New Roman font and its colour would be black. You can of course change this default setting to whatever you wish. Go to Tools->Internet Options and click on the font button under the general tab. There are many other defaults and you can set any or all of them to customise your browser to your requirement.

### 9.3 STRUCTURE OF A HTML DOCUMENT

In the subsequent sections, we will analyse the web page we have made (Figure 9.1) and in the process learn about various HTML tags. Let us look at the HTML coding of our web page. Right-click on the page and opt for view page source. (Figure 9.2)

A basic concept of programming is to present the code in a very structured format. Not only does it make it easy to read, but also helps detect and correct errors. In HTML, we do these in the following way.

The entire HTML document is enveloped between a pair of tags: `<HTML>` and `</HTML>`. In other words, when enclosed between these two tags, the browser treats the page as a web page.

The sections of the page are then defined. In our page we have a header, a body and a footer. The tags are `<head></head>` and `<body></body>`.

So the HTML page looks like this :

```
<html>
<head>
Instructions for the browser goes here;
</head>
<body>
Everything on the web page goes here;
</body>
</html>
```

All the contents of the web page displayed in the browser window are between the tags `<body></body>`. In our web page, this includes the following :

- the banner, Computers and Communication Technology against a coloured background;
- the menu bar with buttons to each chapter;
- all the text in the left column, again against a coloured background;
- a set of hyperlinks;
- two input boxes; and
- the footer

We will explore the tags associated with each of these in the subsequent sections.

## 9.4 STARTING WITH CODING

### What tools do we need?

To create a HTML document, we need two basic tools:

1. **A HTML editor**, to create and save the HTML documents. Notepad or Word Pad for Windows, vi for Unix, and Teach Text for Macintosh are good examples of text editors. There are more advanced and feature rich HTML editors, but a text editor will suffice for our purpose.

**Note :** Do not use a word processor like OpenOffice.org Writer or Microsoft Word. These applications generate a lot of invisible code for formatting, which is beyond the capability of a browser. But if you have to use it, save the document as plain ASCII text. In this format, all additional code generated by the word processor is removed.

2. **A Web browser**, to view and test the HTML documents. The most common and popular web browsers are Mozilla Firefox and Internet Explorer.

### Creating Our First HTML Document

We will now create the web page that we explored in figure 9.3. We will use Windows Notepad as the text editor. Type in the basic tags. Save the file in a convenient location with a suitable name, say **home**.



Notepad will give the file a default extension of **.txt**. But we wish to make a web page. Therefore name the file **home.htm**



Figure 9.3 : Text editor showing HTML codes

Let us now preview our web page. Open the file home.htm in a browser (Figure 9.4).

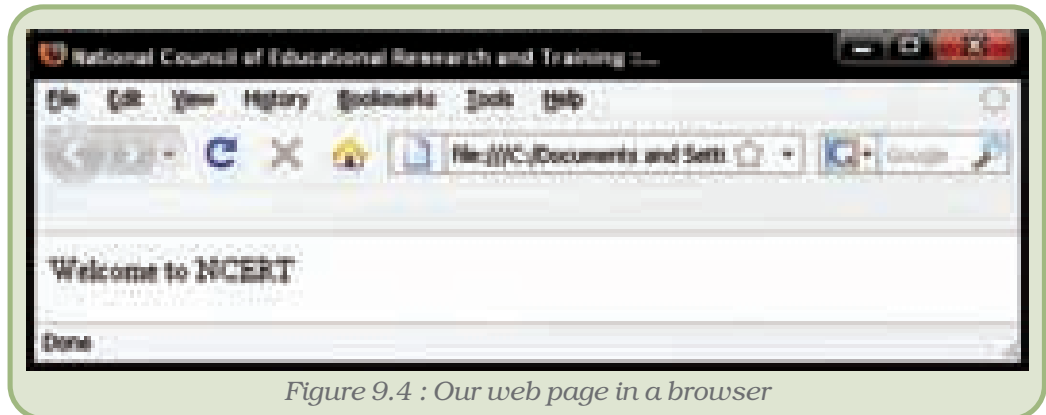


Figure 9.4 : Our web page in a browser

We can continue to edit the file home.htm, adding additional HTML code. Each time you wish to preview your work in the browser, refresh your browser (go to View ->Reload in the Mozilla Firefox browser or press Ctrl+R). The browser reloads the page showing you the latest version.

### Adding the Header Section

Open the file home.htm in Notepad. Insert the tag pair `<head></head>` after the `<html>` tag and before the `<body>` tag. The code that goes between the `<head></head>` tags are not displayed by the browser, with the exception of the title tag. This part of the html document contains all references to scripts, special media players, special fonts, particularly when the web page is made for some other language, the meta data, which is used by search engines to index this web page, and any styles that are used for the content, etc. In our example, we have no such special requirement. So the header section of our web page contains only the `<title>text for the title</title>` tags. Insert this tag and save the page. Preview the web page in your browser.

What we write between the tags `<title></title>` is displayed on the top of the browser window, above its menu bar. Notice the text "National Council of Educational Research and Training :: Home". This will also appear when you print the web page (Figure 9.5).





Figure 9.5 : The title of our web page

## 9.5 CONTENTS OF THE WEB PAGE – FILLING BETWEEN THE <BODY></BODY> TAG

### 9.5.1 BEAUTIFYING THE WEB PAGE – THE “BGCOLOR=” ATTRIBUTE

The body tag can also take a number of attributes. We will look at only one of them, namely “bgcolor=”. This attribute is used to apply a background colour to the web page. Let us see a few examples:

```
<body bgcolor="cyan"> text </body>
<body bgcolor="#00ff00"> text </body>
<body bgcolor="red" background="bg.jpg"> text </body>
```

From the first two examples, you will notice that you can either use the name of the colour or its hexadecimal code. You may look up the hex codes in a graphics application like Gimp or Photoshop, or look up on the web.

From the third example you will notice that we have defined not only a colour, but also an image for the background. The image will be overlaid on the coloured background. So if the image is big enough, you will only see the image as the background. If it is smaller than the screen, the background colour will be visible all around. You will learn about managing images in a later section.

In the previous section, we examined the <head></head> pair of the HTML tags, included a title for our web page. Let us continue our exploration of our web page. Our web page could be visualised as figure 9.6 consisting of the following:

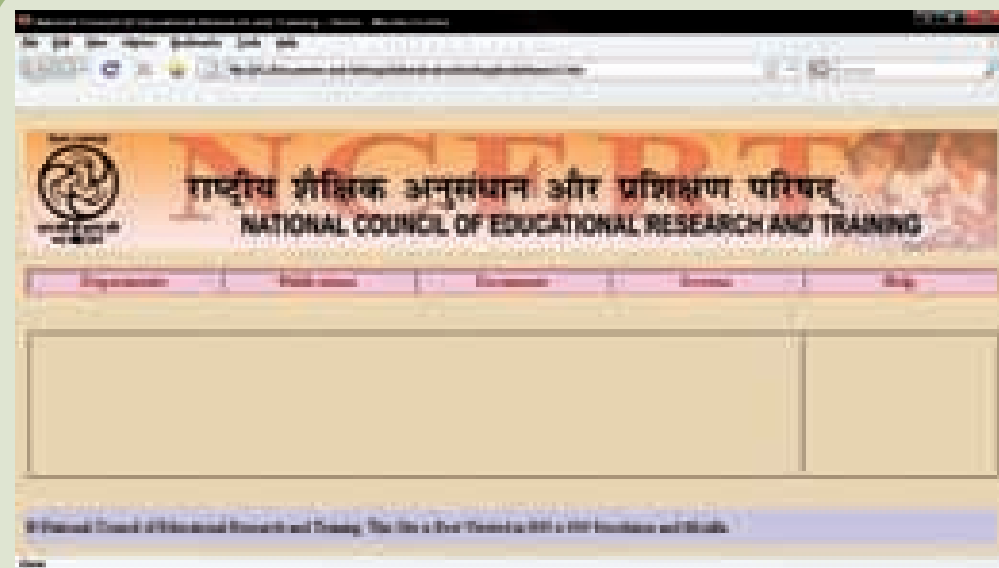


Figure 9.6 : A schematic layout of our web page

### 9.5.2 THE COMMENTS TAG: USE OF `<!-- -->`

When you write a long piece of coding, or if you are writing a piece of code which you want someone else to edit or use, it would be a good habit to mark out sections with labels. At the same time, you do not want these to be visible on the web page. One useful tag for this purpose is the comment tag `<!-- text -->`. For example,

`<!-- the following section will display the image gallery -->`

Anything in between the comment tags is ignored and is **NOT** displayed in a browser. Comments can be multi-line too, and only one set of start and end tags can be used to enclose all the lines.

### 9.5.3 THE BANNER AND THE USE OF THE `<h1></h1>` TAG

Notice that the banner has text in a large font size. There are many ways of achieving this. You could define a font size, you could use pre-defined sizes, or even use an image containing the text. The last option is preferred when a particularly unique style of text or text effect is to be used, something which cannot be obtained using normal HTML coding.

While it is possible to define the exact font size (we will look at this option a little later), HTML provides six levels of text for use as headings. They are numbered 1 to 6, with 1 being the largest and 6 being the smallest. But, how big is h1 and how small is h6? These are relative sizes. Now, if you have six sentences in sizes h1 to h6 as in figure 9.7 and go to view->zoom->zoom in or press Ctrl++, each of these sentences become bigger. Continue doing this and the text keeps growing in size. They maintain their relative sizes. Now, go to view ! zoom ! zoom out or press Ctrl—, the text becomes progressively smaller, still retaining their relative sizes.

The tags used in this case are `<h1>text</h1>`, `<h2>text</h2>`, etc. all the way to `<h6>text</h6>`. Look at these examples.



Figure 9.7 : Defining headings using the `<h1></h1>` tag

### 9.5.4 THE USE OF THE `<font></font>` TAG

The `<h1></h1>` tag of course will use the default font of the browser and display text in varying sizes. So, what do you do if you want the text to be displayed in different fonts?

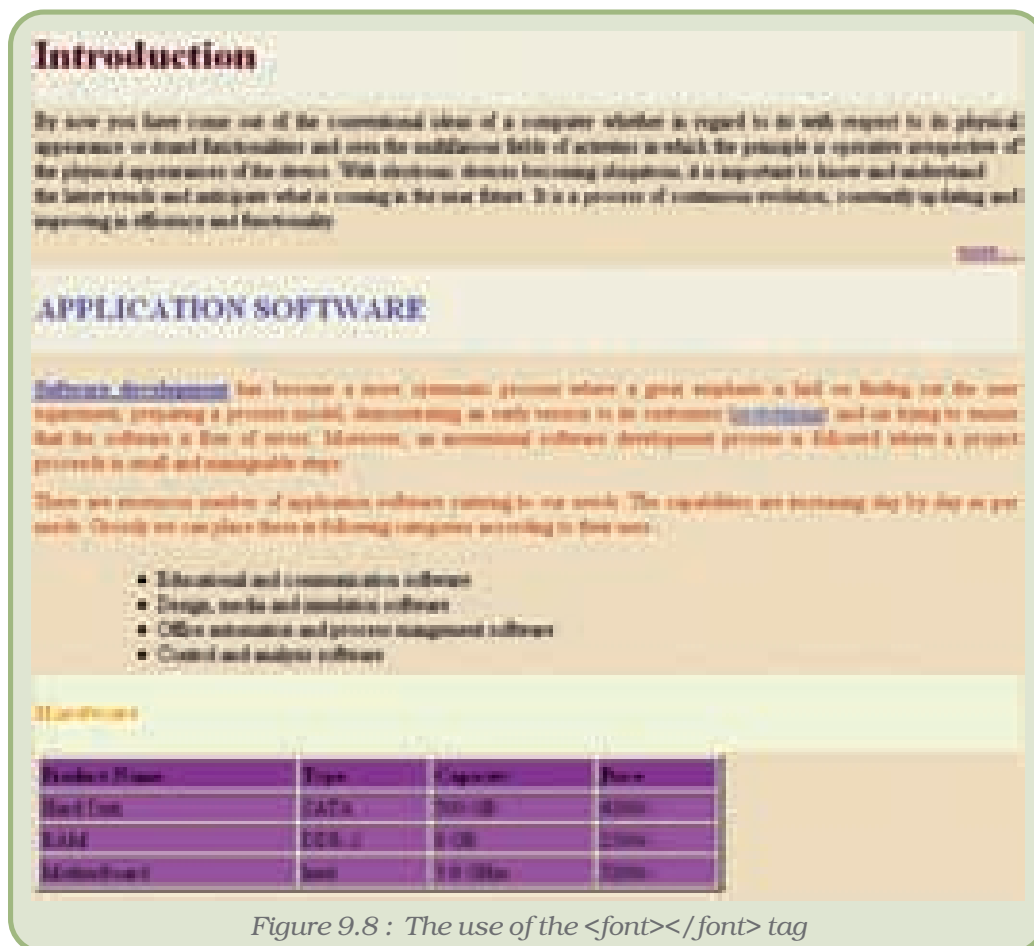
If you use a word processor like OpenOffice.org Writer or Microsoft Word, you are already familiar with the use of different font faces. These are different styles of lettering. To achieve this in HTML, we use the `<font>text</font>` tag. If you do not use any attributes, then the browser chooses its default font face. But if you use the attribute `font-face=` or simply `face=` and give it a value, the name of a font, then the text would be displayed using this font. Of course, the computer on which the web page is viewed should have that font installed.

The HTML code reads:

```
<font face=Times New Roman>text to be displayed</font>
```

You may also like to use other attributes like `size=` or `color=` and put in values of your choice. With `size`, you may choose +1, +2, etc. -1, -2, etc. or use absolute values like 24px, where px stands for pixels or even use percentages, e.g., `size=30%`. With `colour`, first of all remember to use the American spelling “color” and then use either the name or the hexadecimal (or hex code) of the colour.

You will notice each paragraph of text is in a different font face, font size and colour. The HTML code for the page can be seen in figure 9.8.



### 9.5.5 TAGS FOR HIGHLIGHTING TEXT

While there are a large number of font faces, in a variety of languages, following different styles of calligraphy, many of them are not suitable for reading large chunks of text. Also, use of too many different font faces on the same page can be strenuous for the reader. One would therefore prefer other ways of highlighting.

We already encountered one such technique when we used the `<head>` tag. You can use text of different sizes to highlight. While this may be suitable for titles and headings, how do we highlight words or phrases within running text. Usually, we use bold, italics or underline. Let us look at the tags which can do this.

|   |  |
|---|--|
| This is how we convert text into<br><code>&lt;b&gt;bold&lt;/b&gt;;</code> | Display the results here<br><b>bold</b>      |
| This is how we<br><code>&lt;i&gt;italicize&lt;/i&gt;text; and</code>      | Display the results here<br><i>italicize</i> |
| This is how we<br><code>&lt;u&gt;underline&lt;/u&gt; text.</code>         | Display the results here<br><u>underline</u> |

You can also achieve bold face text using `<strong>text</strong>` and italicized text using `<em>text</em>`. Do remember to use the closing tag. If you don't, all the subsequent text will be highlighted. Try this out on your web page.

### 9.5.6 MAKING PARAGRAPHS AND SEPARATING CHUNKS OF TEXT

Running text is usually broken into paragraphs, and with the advent of word processors, text is also justified (aligned) in different ways. We may also have to force a line break to avoid an orphaned or a widowed line (a single line of text on the bottom or top of a page). Let us explore the tags used for this purpose.

#### The `<p>text</p>` tag :

Look at the text in our web page `home.htm` again. The first paragraph is titled Introduction. Look at the structure of the paragraph (see Figure 9.9). After the title you have a blank line, followed by a block of text. The next paragraph doesn't begin immediately. You have one or more blank lines in between.

Now examine the tags which will help us achieve this. The tag pair `<p></p>` is used to define a paragraph. The initial or opening tag `<p>` will define a blank line before the tag. So the paragraph enclosed by these tags will separate it from the previous paragraph. But, the end or closing tag `</p>` will not define a line after the tag. So the next piece of text will have to begin with `<p>`.

When used on its own, this tag will define blank lines. This is useful when you wish to forcibly separate blocks of text. See the example below.



The screenshot shows a web page with two paragraphs of text. The first paragraph is followed by a blank line, and then the second paragraph begins. The text is displayed in a serif font, and the background is a light beige color.

Figure 9.9 : This piece of text will have a blank line in between

### The **<br>** tag :

You will notice that the `<p></p>` tag pair provides a large blank between the paragraphs. When you use the tag to provide blank lines, it may result in the page looking ugly. You will prefer to provide smaller gaps. Also, you may like to forcibly close a line and want to continue on the next line. When you type in text, depending on the size of your window, you may notice the text shifting to the next line. But if you resize the window, the text rearranges. This means that if the window were very big, all the text would come on to the same line. Without a line break defined, the text will not be forced on to the next line.

The tag to use in these contexts is `<br>`, which stands for a line break. This tag moves the following text into the next line. It is a standalone tag without a closing tag.

**Note :** The line break tags `<br>` can be used to give as many blank lines as desired but the `<p>` tag can't be used more than once. In other words, specifying `<p><p><p>` won't give us 3 empty lines, it will just give us 1 empty line. But specifying `<br><br><br>` gives us 3 empty lines.

### The **<hr>** tag :

If you wish to separate two sections of your web page using a line, you can do it in two ways. One, you use an image. Two, you use the `<hr>` tag. This again is a standalone tag and stands for horizontal rule. You may use various attributes with this tag, as shown in the examples below.

```
<hr width=60>, <hr width=70%>, <hr size=5>
```

While width indicates the length of the line, size indicates the thickness.

## 9.5.7 ALIGNING TEXT

Most times text is aligned with the left margin of the page. In particular cases, you may also like to right align it. If it is a heading, you may also like to center it on the page. Let us look at the paragraphs in our web page. The first paragraph is left aligned. Notice that while each line starts from the same vertical position on the left, they stop at different places on the right. The `align="left"` attribute can be used with a variety of tags. For instance, with the heading tag, or the paragraph tag or even within tables.

For example `<p align="left">This text is left aligned. </p>`

Similarly, you can use the `align="right"`, `align="center"` and `align="justify"` attributes to align the text to the right, center or to justify – align the text to both the left and right margins. This attribute can also be combined with various other attributes like font, colour or size.

### 9.5.8 PRESENTING LISTS OF CONTENT

There are two commonly used types of lists – a bulleted list or an unordered list, and a numbered list or an ordered list.

#### An example of an Unordered List...

|  |   |
|--|---|
| <ul style="list-style-type: none"> <li>• bicycle</li> <li>• bullock cart</li> <li>• car</li> <li>• bus</li> <li>• train</li> <li>• ship</li> </ul> | <pre>&lt;ul&gt; &lt;li&gt;bicycle &lt;li&gt;bullock cart &lt;li&gt;car &lt;li&gt;bus &lt;li&gt;train &lt;li&gt;ship &lt;/ul&gt;</pre> |
|--|---|

The tag pair `<ul>` and `</ul>` stands for the unordered list, while `<li>` stands for list item. You may use a `</li>` to close the tag, but this is optional. Notice the bullet before each item. This is generated by the `<li>` tag. There is no limit to the number of list items you may have in a single list.

The ordered list is very much similar in structure to the unordered list, except that the opening tag for the list is `<ol>` instead of `<ul>`, and the closing tag is `</ol>` instead of `</ul>`. List items within the list still use the same tags.

#### An example of an Ordered List...

|  |   |
|--|---|
| <ol style="list-style-type: none"> <li>1 Argentina</li> <li>2 Bhutan</li> <li>3 Canada</li> <li>4 Denmark</li> <li>5 Ethiopia</li> <li>6 France</li> </ol> | <pre>&lt;ol&gt; &lt;li&gt;Argentina &lt;li&gt;Bhutan &lt;li&gt;Canada &lt;li&gt;Denmark &lt;li&gt;Ethiopia &lt;li&gt;France &lt;/ol&gt;</pre> |
|--|---|

The Ordered list offers the following three attributes :

The **TYPE** attribute of the `<OL>` tag can be used to change the type of numbering displayed in front of the list items (1/A/a/I/i ). For example :

`<OL TYPE=1>` makes numbers (1, 2, 3) (this is the default)

`<OL TYPE=A>` makes uppercase letters (A, B, C)

`<OL TYPE=a>` makes lowercase letters (a, b, c)

`<OL TYPE=I>` makes uppercase Roman numerals (I, II, III)

`<OL TYPE=i>` makes lowercase Roman numerals (i, ii, iii)

The **START** attribute of the `<OL>` tag is used to start a numbered list with a value other than “1” (or A, a, I, or i). For example :

`<OL TYPE=1 START=5>` will start the numbering from 5

`<OL TYPE=A START=5>` will start the numbering from E

The **VALUE** attribute of the `<LI>` tag is used to change specific numbers within a list. For example :

`<LI TYPE=1 VALUE=5>` will display the number 5.

### 9.5.9 TABLES – THE USE OF `<TABLE>` `</TABLE>` TAG

Tables are a powerful technique of organising content on a web page. There are two significant ways in which the concept of tables are used; one where the table is visible on the web page and two where it is invisible, but restricts content into particular regions of the web page. We will consider examples of both these ways.

Let us examine the structure of a typical table. Look at figure 9.10 below.



Figure 9.10 : Structure of a typical table

The table is organised in to rows and columns. Some rows and columns can be merged to form larger spaces within the table. They may also be split in to multiple rows or columns. The table may have a



header row, indicating what each column contains. Alternately it may have captions in the first column, indicating what each row contains. Each of these rows, columns, individual cells or captions may be highlighted in different ways, using different font faces, sizes of letters, emphasising text, aligning it differently, or filling the background with a colour or image. So there can be an enormous variety of tables.

The rows and columns of tables are demarcated by lines. In the HTML case, we refer to them as borders. Their colours or their line thickness could be modified. The text inside the cells are separated from the borders. The cells themselves can be separated from each other. In effect, when we learn to use the tag pair `<table>` `</table>` we are learning a whole new way of organising content.

This large variety of presentation techniques can be achieved using a very small number of tags, tag attributes and values.

The `<table>` `</table>` pair is the defining pair. All tags, attributes or values, text or images form the content of the table. A simple three row, two column table. The first row contains captions for each column. How does the HTML code look like? See figure 9.11.

```
<table border="1">
<caption align="center" >This is our table's caption</caption>
<tr><td>first cell in the first row</td>
    <td>second cell in the first row</td></tr>

<tr><td>first cell in the second row</td>
    <td>second cell in the second row</td></tr>

<tr><td>first cell in the third row</td>
    <td>second cell in the third row</td></tr>
</table>
```

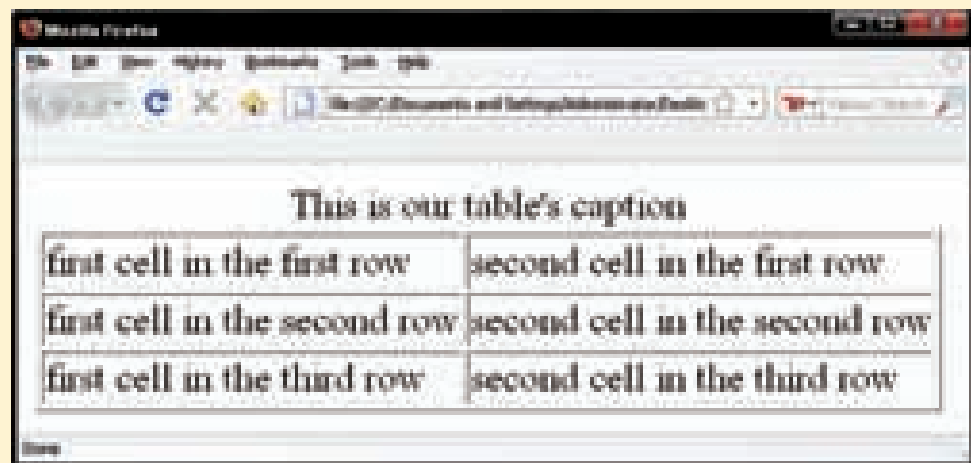


Figure 9.11 : A table with 3 rows and 2 columns

The basic table creation uses the following 5 tags:

|   |  |
|---|--|
| <b>&lt;table&gt; &lt;/table&gt;</b>     | This pair defines the beginning and end of a table   |
| <b>&lt;tr&gt;&lt;/tr&gt;</b>            | This pair (<tr> stands for table row) defines a row of the table. So each row of your table requires one pair.   |
| <b>&lt;td&gt;&lt;/td&gt;</b>            | This pair (<td> stands for table data) defines a cell and number of such pairs you use will define the number of cells your row is divided into.                             |
| <b>&lt;th&gt;&lt;/th&gt;</b>            | Used in place of the <td></td> tag, this pair (<th> stands for table row Heading) formats the text in a cell as bold face and centre it to give the effect of a row heading. |
| <b>&lt;caption&gt; &lt;/caption&gt;</b> | This tag pair is used to provide a heading to the table.   |

These basic table tags can be expanded to produce a variety of tables. Other HTML tags for highlighting and formatting text can be used along with their attributes and values. Let us examine some of these attributes, their possible values and the effect it has on the table.

### Attributes of the <Table> tag

#### Specifying the table border and border colour

The **BORDER** attribute of the <TABLE> tag is used to give the table a border. The BORDER attribute specifies the thickness of the table border as a number. The default setting is BORDER=0, which means, if you do not use the border attribute, you will end up with a table with no border.

**Example :** <table border=2>This table has a border of thickness 2</table>

There are two ways of colouring a table. You could colour the border or you could fill the cells with colour. You use the **BORDERCOLOR=** attribute to specifies a colour for the table border. You use the **BGCOLOR=** attribute to specify a colour to fill.

**Example :** <table bordercolor="red">This table will have a red border</table>

<table bgcolor="green">The cells of the table will have a green background</table>

Like we saw earlier, the value of these attributes could also be the hexcode of the colour. So you could use #0000ff in place of blue. This is very useful when you want to use a particular shade of a colour and that does not have a name.

Instead of a colour, you could also use an image as a background in the table. The **BACKGROUND** attribute specifies the image to fill the background of the table or of individual cells. While specifying these attributes, remember :

- in the <table> tag, it affects the entire table (cells as well as the margins).
- in the <tr> tag, it affects the entire row (cells only). It overrides the colour specified in the <table> tag.
- The **BACKGROUND** attribute does not work if specified in the <tr> tag.
- in the <td> tag, it affects the specified cell. It overrides the colour/image specified in the <table> or the <tr> tag.

### Adjusting the sizes of the table and its cells

When no sizes are specified the table is only as big as it needs to be according to its contents. The attributes **WIDTH** and **HEIGHT** can be used to control the width and the height of the table respectively. These attributes can be specified either as a percentage (of the browser window) or as a specific size measured in pixels.

There are a few peculiarities related to these attributes, when you use them with the <tr> and <td> tags that you should bear in mind.

- Using the HEIGHT attribute in a particular <TD> or <TR> tag overrides the HEIGHT specifications of the <TABLE> tag.
- Using the HEIGHT/WIDTH attribute in a particular <TD> or <TH> tag causes a height/width change in all the <TD> tags of that column. In case two cells of the same column contain HEIGHT/WIDTH attributes, then normally the browser uses the higher width specified.

### Customising Margins

The attribute **CELLPADDING** specifies (in pixels) how much empty “space” exists between items in the cells and the walls of the cells, a higher value for this attribute making the cells larger, “padding” the cell.

The attribute **CELLSPACING** specifies (in pixels) the distance between two adjacent cells. This is particularly useful when you use the table tag to separate blocks of text. For instance you could use this attribute to create a two column text.

You will notice that if you do not specify a cell padding, the table border will stick to the edge of the contents of the table. Some examples are given below in figure 9.12.



Figure 9.12 : The use of cell padding and cell spacing attributes

## Customising Cell Alignments

Cell alignment refers to the horizontal or vertical alignment of text in a cell. By default the cell contents are left aligned (horizontally) and centered (vertically). To change this, we use the `align =` attribute in the `<td>` tag. Look at these examples:

For horizontal alignment,

`<TD ALIGN=left>` aligns text to the left

`<TD ALIGN=right>` aligns text to the right

`<TD ALIGN=center>` aligns text to center

For vertical alignment

`<TD VALIGN=top>` aligns text to the top

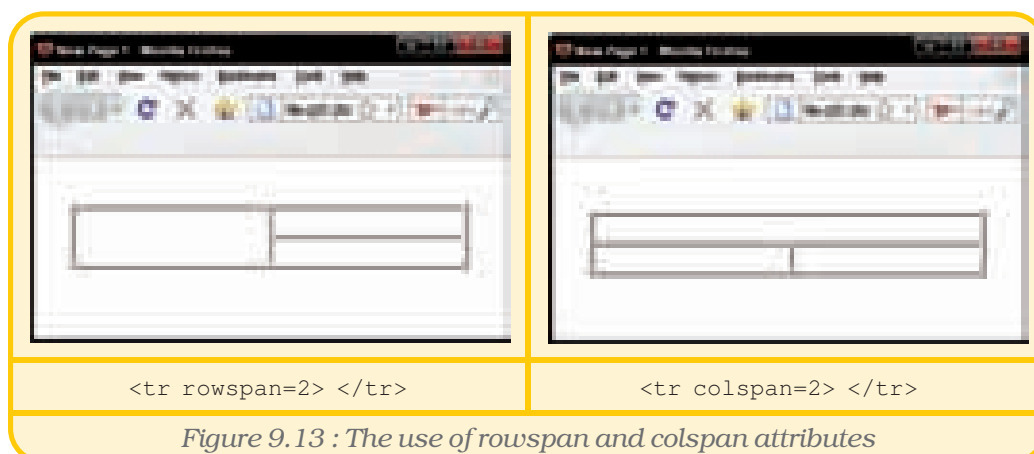
`<TD VALIGN=bottom>` aligns text to the bottom

`<TD VALIGN=middle>` aligns text to the middle

You can also combine a horizontal `align =` attribute with a vertical `align =` attribute.

## Merging Rows and Columns

Merging of two or more adjacent cells or two or more rows can be achieved using the **ROWSPAN/COLSPAN** attribute. This helps us stretch a cell across the entire row, or over multiple rows or columns. Look at the examples below (Figure 9.13):



## Nesting tables

Try placing two tables side by side on a web page. You will notice that the `<table>` tag automatically sends the second table to the next line. To get around this problem, we simply nest the two tables inside a third table, a sort of border around the two tables, thus keeping them on the same line adjacent to each other.

You create a table with one row and two columns and put in a table into each of the two cells. See the code and the result in figure 9.14 below.

```
<html>
<table border="1">
<tr>
<td>
<table >
<tr>Table1</tr>
</table>
</td>
<td>
<table ><tr>Table2</tr></table>
</td>
</tr>
</table>
</html>
```

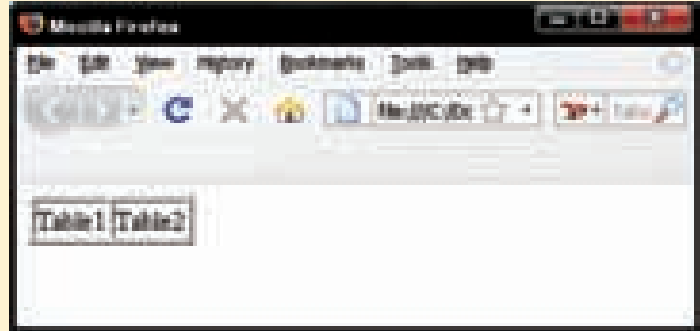


Figure 9.14 : Code of table within a table

Nesting tables can also be used to create tables where each table row has different cellpadding and cellspacing. Try this out.

Let us now examine the use of tables in our web page. Look at figure 9.15. There are two distinct ways in which tables are used here. The overall web page itself has a table like look. This table contains two rows. While the first row has only one cell or column, the second, containing the menu options has five cells. The third which constitutes the text of the web page has two cells or columns.

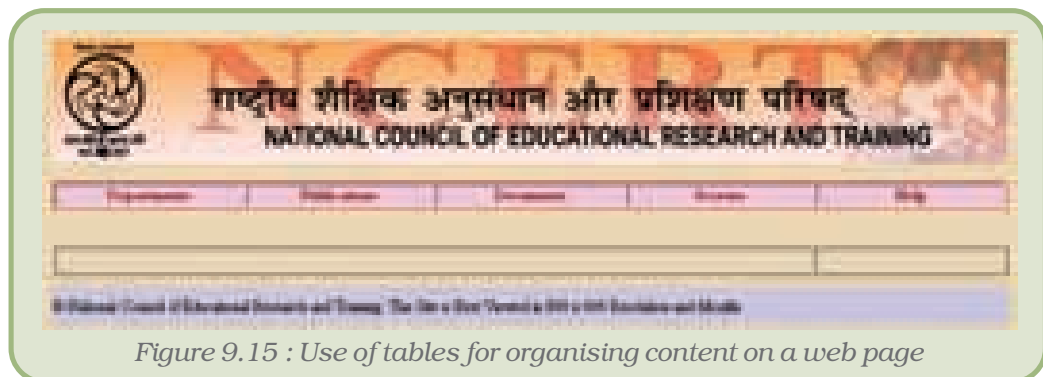


Figure 9.15 : Use of tables for organising content on a web page

You also have three tables visible on the page :

- The first table listing the contents has six rows or a single cell each. This table is nested within the first column of the third row of the main table.
- The second listing the specifications and prices of hardware, is nested in the second column. While the header row indicates the types of information, the first column lists the type of hardware.
- The third table in the third column of the third row of the main table holds thumbnails of an image gallery. Each image is a hyperlink to a larger version of the image and will open on another web page. We will discuss the tags for this table in the section 9.5.10.

If you observe the source code of this page, you should be able to identify the tags for these tables. Write down the tags for these tables. Take care to ensure that each tag has its corresponding closing tag. This is particularly necessary while nesting tables, rows or cells. Also notice the use of various attributes and their values. Vary them and observe the effect.

### 9.5.10 PHOTOGRAPHS AND OTHER MEDIA ON THE WEB PAGE

Most of the websites you encounter would have one or more images. It could be the use of a logo of the organisation, some design elements, a background, an image gallery or even a photograph used for beautifying the web page.

Images can be drawings, photographs, graphs, or even decorated text saved as images. Keeping in view the need for small file sizes, HTML supports jpg, gif, and png formats. But as we learnt earlier, HTML produces a text file. Hence images are not placed inside the file. A reference to the file is made and the image file accompanies the text file.

We use the `<img>` tag to insert an image on the web page. This is always used with the attribute SRC, which stands for the source of the file (what is the image file and where is it located?). Typical examples are :

```
<IMG SRC="image.jpg">
<IMG SRC="\images\image2.gif">
```

In the first example, the image referred to lies in the same folder as the html file. In the second example, the image is in a sub-folder called images, which is in the same folder as the html file. This also brings about an interesting option. The image file can be anywhere, as long as we can define its exact location. The image can even be on a different web site, residing on a server half way across the world. The tag would look like :

```
<IMG SRC="www.mywebsite.com/photographs/photo1.jpg">
```

Of course this assumes that the website and the image is available on the remote website, when you are viewing your web page.

#### Attributes of the `<IMG>` tag

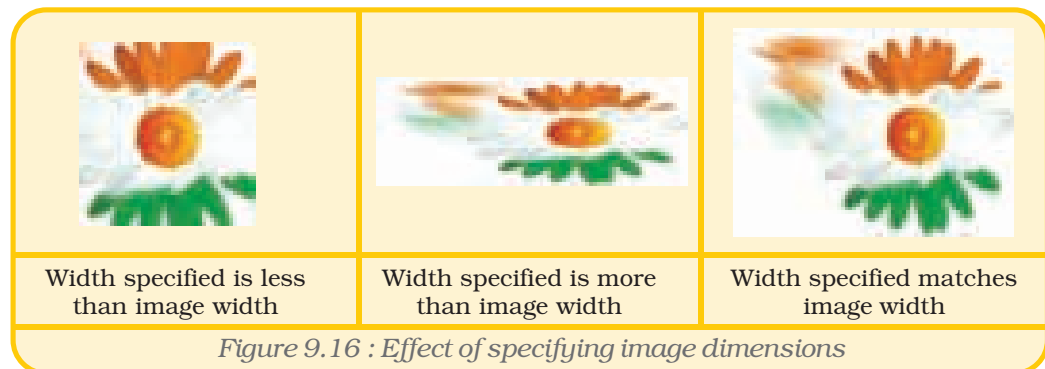
The `<IMG>` tag can also take many attributes, which allow formatting, highlighting and specifying the size of the image.

#### Specifying Image Dimensions

Remember that images can have a large file size. Specifying an image's dimensions (height and width) can control the loading time of images. As the browser loads the page, it notes the height and width attributes and leaves that much space for the image. Next, it lays out the remaining

text, and then it goes back and fills in the image. If these attributes are not specified, the browser has to load enough of the image to get the dimensions before it can lay out the rest of the text, thereby slowing the display of other page elements. To specify image height and width, add the **HEIGHT** and **WIDTH** (in pixels) attributes in the <IMG> tag.

What happens if the height or width values are smaller or greater than the image? Simple, the image gets cropped (cut) or stretched (Figure 9.16). So, height or width should exactly match the actual image.

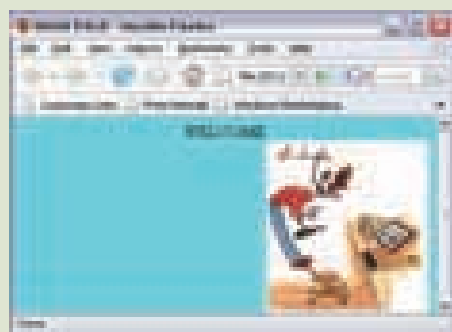


### Aligning Images

By default the image is aligned to the left of the web page and the text is aligned at the bottom most line of the image. To change this default setting, HTML offers several image alignment options:

- Three options vertically align the image with respect to a line of text: align=top, align=bottom and align=middle
- Two options align the image to the left or the right of the window (with corresponding text wrap)

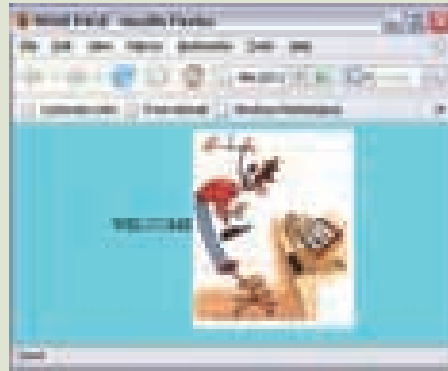
Look at figure 9.17 below.



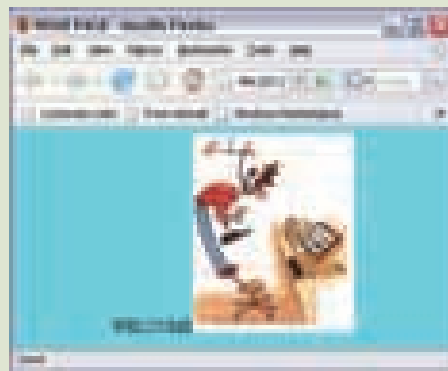
The ALIGN=TOP aligns the top of the image with the top of the surrounding text as displayed in the adjacent figure.



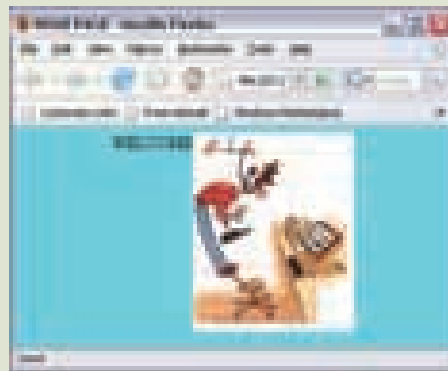
ALIGN = MIDDLE places the middle of the image at the baseline of surrounding text.



Similarly, ALIGN = BOTTOM places the bottom of the image on the same line as the text. This is the default setting.



ALIGN = LEFT places the image on the left of the browser window.



ALIGN = RIGHT places the image on the right of the browser window.

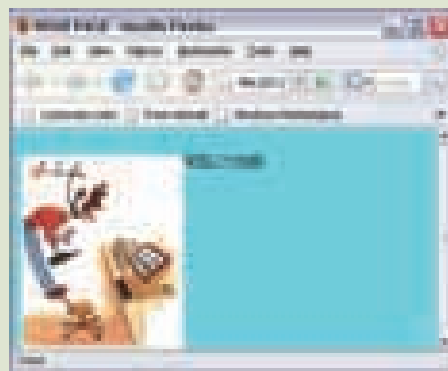


Figure 9.17 : Use of align = attribute for the image tag

## Image Borders

The **BORDER** attribute of the <IMG> tag displays a border. You can also control its thickness by specifying a value. The default value is 0, indicating no border. If you do not specify a border colour using the bordercolor= attribute, then the colour specified in the <body> tag will be used. If you have not done so, then the default colour, black will be used.

## Displaying Alternative Text

The web is used by all people and a few of them could be blind. How do blind people use the web? Screen reader software read the page for them. But what happens when you encounter an image?

To facilitate this, the <IMG> tag uses the **ALT** attribute. It is used to specify the **Alternative** text to be displayed. This attribute can also be useful if :

- The user uses a text-only browser.
- The user turns off images so that a file loads faster.
- The browser does not display images correctly.
- Images are not displayed because links aren't working properly.
- Text is to be displayed while images load.

Now let us look at our web page and look at how the image gallery was created. Look at the source code for this table in figure 9.18. Nothing extraordinary. The table has five rows with five cells each, all of them equally spaced.

```
<table border="1" width="735">
  <tr>
    <td width="245" height="245"></td>
    <td width="245" height="245"></td>
    <td width="245" height="245"></td>
    <td width="245" height="245"></td>
  </tr>
  <tr>
    <td width="245" height="245"></td>
    <td width="245" height="245"></td>
    <td width="245" height="245"></td>
    <td width="245" height="245"></td>
  </tr>
</table>
```



Figure 9.18 : Source code for the image gallery

As expected, each `<tr></tr>` pair has five `<td></td>` pairs within in, representing the five cells in each row. Each cell contains an image and the image itself functions as a link to another image. Therefore you should expect a code of the form `<a href="page.htm"><img src=img01.jpg></a>`, where the `<a></a>` pair refers to the hyperlink and the `<img>` tag refers to the thumbnail. Observe the source code and confirm it.

### 9.5.11 MAKING THE WEB PAGE INTERACTIVE – HYPERLINKS

What makes the Internet one of mankind's greatest resources is its ability to link resources to one another and on the click of a mouse, you can jump to a page residing on a machine half way across the world. We call these **hyperlinks**. In this section, we will explore the tags that makes this happen. This would be one of the simplest, but still one of the most significant tag pair `<A>text</A>`.

The text between the tags is converted to a link. This link could be a link to another paragraph on the same page, another page in the same folder, another page in some other folder or even a page in some other machine, elsewhere. But on its own this tag would be useless, as it still doesn't say, which page it links to.

The attribute that enables this addressing is `HREF=`. So, the hyperlink tag looks like :

```
<A HREF="http://mywebsite.com/mywebpage.htm">link to my site
</A>
```

What does this say? The `HREF=` attribute has a value which refers to a page "mywebpage.htm" on the site "mywebsite.com". As long as this page actually exists on this site and the server hosting this site is connected to the web, you will be able to call this page – a click on the link would display this page.

The World Wide Web uses an addressing scheme known as **URL** to indicate the location of files on the web. URL stands for **Uniform Resource Locator** (sometimes also called "Universal Resource Locator"), which is the address of a document on the Internet. You will notice the URL for your web page displayed in the address bar of your Web browser.

Hover your mouse cursor over a hyperlink. Notice that the cursor changes shape into a pointed finger icon. At the same time, you will see the address to which this hyperlink points in the status bar at the bottom-left corner of the page.

### Types of URLs

URLs are of two types, absolute URL and relative URL.

An absolute URL contains the complete address of a file on the Internet. An absolute URL therefore refers to the hostname, folder

name and filename, and also the protocol (set of rules) it will use to connect to the web page or file.

For example: <http://www.ncert.nic.in/html/schoolcurriculum.htm>

Notice, we keep referring to the file on the Internet. The website not only hosts web pages, but also images, video, audio, documents, zipped files or folders, and practically any type of digital file. Browsers mostly interact with web pages, which are files of the html type. All other files are downloaded on to your machine (or executed if it is a program).

A relative URL, on the other hand, is a shorter form, and points to files on the same server or in the same folder. A relative URL contains only the folder name and filename or just the filename if the web page containing the link and the resource called are in the same folder.

For example: </html/school.htm>

This URL refers to a file [school.htm](#) residing in a folder [html](#). The folder resides on the same site as the web page containing the link. We could have written the complete address as

<http://www.mywebsite.com/html/school.htm> but the shorter form is sufficient.

## Types of hyperlinking

As we mentioned earlier, hyperlinks can be to a destination within the same page, or to some other page. We could also use hyperlinks to launch scripts or applications and perform other functions, for example, we could launch a slide show, play a movie or even send an e-mail. Let us examine the attributes for these actions.

## External linking

This is perhaps the most common usage of hyper links. Page1 containing the link hyperlinks to Page 2. And as we mentioned earlier, page 2 can be on the same site or on a remote site.

The format is `<A HREF="http://mywebsite.com/myfile.htm">link to my file</A>`. If the destination file, in this case [myfile.htm](#), happens to be a web page, then the browser handles it on its own. The new web page is displayed.

If the destination file is let us say, a video, then the browser calls upon a media player to help display the contents. We call such add on software as plugins. Java scripts are also used in some cases. Browsers can display text and images, but for all other media they would require external support. So, you could have plugins to play video, audio, animations, or scripts to play slide shows or even an executable (.exe) program.

Let us consider an example of playing a video clip, using the Windows Media Player. You will have to embed the following code in to the `<body></body>` tag at the top of the web page.

```
<embed type="application/x-mplayer2" pluginspage="http://
www.microsoft.com/Windows/MediaPlayer/" name="mediaplayer1"
autostart="false" width="320" height="240" src="myvideo.wmv" />
```

Be sure to have a video file named `myvideo.wmv` in the same directory as the html file, or simply change the `src` attribute to point to the path of the video file.

Let us review the attributes of the `<embed>` tag :

- `type=""` defines the type of embedded object you'd like to use. This tag is mandatory for Windows Media Player to be embedded in the web page;
- `pluginspage=""` this attribute is needed only if the user computer does not have a Windows Media Player plugin; as you cannot be sure, include it;
- `name=""` this attribute assigns a unique name to this embedded object, which will allow you to place and access multiple embedded objects (see the section on named anchors below);
- `autostart=""` defines whether or not the video should start playing automatically when the page is loaded;
- `width=""` and `height=""` defines the size in pixels of the Windows Media Player embedded window. If you want the visitors to see the video at its original resolution, omit the width and height attributes.
- `src` refers to the source, it defines the path to the file you want to play.

## Internal linking

As we mentioned earlier, the link could be to another part of the same page. Typically, you would need this when you have a contents page listing different sections of a document and the sections themselves, all in the same web page.

You would require to mark particular sections of a page using a reference marker. This is done using the `NAME` attribute. Let us look at this example :

| Code   | Output               |
|--|----------------------|
| <code>&lt;html&gt;</code>  |                      |
| <code>&lt;A NAME="CONTENTS"&gt;Contents &lt;/A&gt;</code>        | Contents             |
| <code>&lt;A HREF="#chap1"&gt;Chapter 1&lt;/A&gt;</code>          | Chapter 1            |
| <code>&lt;A HREF="#chap1"&gt;Chapter 2&lt;/A&gt;Chapter 2</code> | Chapter 2            |
| <code>&lt;p&gt;Introduction:</code>                              | Introduction:        |
| Text of Introduction   | Text of Introduction |

|  |                   |
|--|-------------------|
| <code>&lt;A NAME ="chap1"&gt;Chapter 1&lt;/A&gt;</code>            | Chapter 1         |
| Text of chapter 1  | Text of Chapter 1 |
| <code>&lt;A HREF="# CONTENTS"&gt;Back to contents&lt;/A&gt;</code> | Back to contents  |
| <code>&lt;A NAME ="chap2"&gt;Chapter 2&lt;/A&gt;</code>            | Chapter 2         |
| Text of chapter 2  | Text of Chapter 2 |
| <code>&lt;A HREF="# CONTENTS"&gt;Back to contents&lt;/A&gt;</code> | Back to contents  |
| <code>&lt;/html&gt;</code>   |                   |

From the output above, you will notice that the tag pair and the attribute `<A NAME=" "> </A>` and the tag pair and attribute `<A HREF=" "> </A>` are converted to links. Let us examine what these links refer to.

A click on the links Chapter 1 or Chapter 2 leads to the beginning of the respective chapters. The link at the bottom of each chapter Back to contents will lead back to the contents list. The tag pair `<A Name=" "></A>` is called a named anchor and is actually not visible on the web page. This tag functions as the destination for the link. So, when you click on the link Chapter 1, you are led to this anchor. Similarly, when you click on the link Back to contents, you are led to the anchor for the contents list.

The other tag pair, `<A HREF=" "></A>` is similar to any other link, but the destination reference is preceded by a hash (#). This hash prompts the browser to look for a named anchor on the same page.

This technique of navigation within the page is very useful when you wish to keep a document together and not split it into many different web pages. Printing, or downloading this page will yield the entire document in one piece. While it may look silly on short web pages, it is certainly very useful if you have a very long web page and have to frequently go back and forth or seek particular sections.

You can also use this technique when you wish to refer to a particular section on a different web page. In this case, the hyperlink will refer to a different web page and a named anchor on that web page. See the example below :

```
<a href="index.html#section2">
```

The link refers to an anchor named section2 on a web page named index.html.

### Using images as links :

Large images have big file sizes and take time to load. To overcome this, we represent the image with a smaller image on the web page. We refer to this as the thumbnail. So we would like the user to click on this image to see the larger image. Thumbnails can be used to represent

not just images, but any resource. For instance, you may like a photograph of the cover page of a book to represent the web page containing the book.

The tags used in this case is a combination of the `<A HREF=""></A>` and the `<IMG SRC=>` tags. Look at this example:

```
<A HREF="mywebpage.htm"><IMG SRC="myphoto.jpg"></A>
```

You would have used the tag `<IMG SRC="myphoto.jpg">` to display the picture. Enclosing this tag with the `<A HREF=></A>` tag, converts this picture itself into a link. So a click on this link would lead you to a new page, `mywebpage.htm`.

Notice, that it does not say anything about what the new page contains. It simply links to a different page. In fact, this facet of HTML allows it to hand over control to the next page, leaving that page to handle resources relevant to that page.

## E-mail links

If you wish someone to comment about your web page or the content on it, or you wish to invite someone to contact you, an easy way of doing so would be to put in your contact details (address or phone numbers). An easier way would be to provide an e-mail link so that an e-mail can be sent to you from right there. The tag to use is still the `<A HREF=></A>`. See this example:

```
<A HREF="mailto: dceta.ncert@nic.in">Contact Us</A>
```

The "mailto:" part of the attribute launches your default mail client (see figure 9.19 below). Of course, you should have configured your mail client to send mails. If you usually use a browser based mail programme like Hotmail, or Yahoo, or Gmail, you may not have configured it. Typically, in a Windows environment, Outlook Express is the default e-mail client, unless you have configured some other mail client. So, this tag will be useful and serve the purpose only on those machines, where an e-mail client is configured. Nevertheless, it is a very useful and time saving tag attribute.

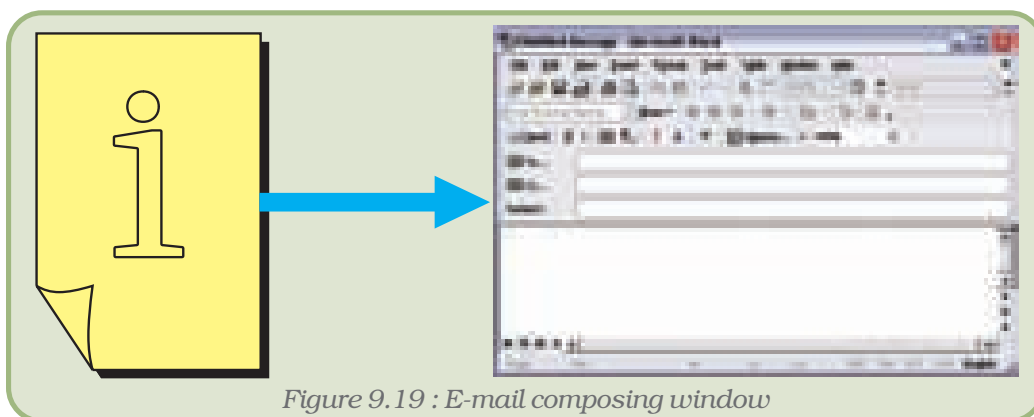


Figure 9.19 : E-mail composing window



Specifying an e-mail address within the anchor tags instructs the browser to automatically insert the mentioned address in the “to” text box of the e-mail composing window.

### 9.5.12 MAKING THE WEB PAGE INTERACTIVE – INPUT FORMS

In the previous section, we explored web interactivity using the <A></A> tag pair. Hyperlinks are only one form of interactivity that the web provides. It does allow us to jump around from one page to another, but all this is predefined. What if you wish to seek an answer to a question!

This would require a way of sending your question to the server and some way in which the server processes your question and respond to it. HTML provides for such forms of interactivity too. There is a range of types of inputs that can be submitted. Not only does it allow a web site to respond to user queries but also seek information from the users.

The tag pair used is <form></form>. Let us explore its use. Primarily this tag must permit the input of text. It must then define a delivery mechanism and the destination for this message. This is achieved by the attributes ACTION and METHOD respectively.

The ACTION attribute specifies where the information will be sent for processing. Information processing destinations can be CGI programs, JavaScript functions, or an e-mail address. **CGI** or the Common Gateway Interface (CGI) is a standard protocol for interfacing external software with an information server, commonly a web server. Look at these examples :

| Action                              | Code  |
|-------------------------------------|---|
| Sending request to a cgi program    | ACTION="/cgi-bin/something.cgi" (relative url) or ACTION="http://mywebsite.com/cgi-bin/something.cgi"(absolute url) |
| Sending request to a javascript     | ACTION="javascript:SomeFunction()"  |
| Sending request to an email address | ACTION="mailto:me@mywebsite.org"  |

There are two methods of sending information in a form to the server, the get and the post method. We use the METHOD attribute of the <FORM> tag to specify this. Of course, the server and the associated software will predetermine what method you will use and so you may not have a choice. But it is useful to know the difference between the two methods.

With the “get” method, the query or information along with relevant codes is appended to the URL specified by the action attribute and this

new URL is sent to the processing agent. The “get” method should be used when the form is idempotent (i.e., causes no side-effects). Many database searches have no visible side-effects and make ideal applications for the “get” method. Further the get method restricts the input to ASCII characters.

So, if you searched for solar eclipse 2009 on mywebsite.com, the form submitted would read `http://www.mywebsite.com/search?q=solar+eclipse+2009`.

With the “post” method, the query or information is included in the body of the form and sent to the processing agent. If the service associated with the processing of a form causes side effects (for example, if the form modifies a database or subscription to a service), the “post” method should be used.

## Form Elements

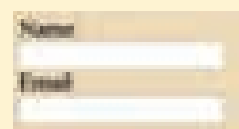
Having defined the action and the method, we define the format of the information to be input and sent to the server. These can be in the form of text entry fields, check boxes, radio selections, password entry fields, form buttons, file upload fields, or image buttons. All these are defined as attributes of the **<INPUT> tag**.

The **<INPUT>** tag can have several other attributes depending on the value of the “TYPE” attribute. The type attribute specifies the type of input we want from the user. It can be used to create the following type of interface elements:

1. Text field
2. Password field
3. Radio button
4. Check box
5. Submit button
6. Reset button
7. Image button
8. Hidden field

The **TYPE=“TEXT”** attribute is a common form input field. It creates a place wherein one line of information can be typed.

```
<LABEL for="firstname">Name: </LABEL><br>
<INPUT type="text" id="firstname" size="20"><BR>
<LABEL for="email">Email: </LABEL><br>
<input name="text" type="text" id="email" size="20">
```



The **TYPE=“PASSWORD”** attribute works the same as the type=“TEXT” attribute except that the characters typed into the input area are displayed as asterisks (or some special character). Password text is scrambled during transmission and then unscrambled when the form data is received at the server end.

Radio buttons are used when the user is expected to select one – and just one – option from a set of alternatives. The **TYPE="RADIO"** specifies a set of controls that are linked so that only one radio button among each set is selected at a time; if the user clicks one radio button, the others in the set are automatically de-selected. All options for the radio button should have the same name. The value sent by the form is the value of the radio button that was last selected. Adding the option **CHECKED** to one of the buttons in a set will make that button pre-selected when the page loads, a kind of default value. Look at the example below:

```
<LABEL for="firstname">Name:
</LABEL><br>
<INPUT type="text" id="firstname" size="20"><BR> <LABEL
for="email">Email:
</LABEL><br>
<input name="text" type="text" id="email" size="20">
Gender<BR>
<INPUT type="radio" name="sex" value="Male">
Male <INPUT type="radio" name="sex" value="Female"> Female
```

Check boxes are used to allow the user to select multiple options from a set of alternatives. A check box is thus not affected by other buttons, so it is possible to have more than one in a group checked at a time. They can be created using the **TYPE="CHECKBOX"** attribute. Notice that unlike a radio button, every check box has a unique name. Clicking it will place a check mark ("X" or "✓") and clicking it again will remove the mark. The value sent in the web form is the value of the check box if it was selected; otherwise the value will be empty. You may add the option **CHECKED** to a particular check box to define it as the default value.

```
Language<br><input type="checkbox" name="option1"
value="hindi">Hindi<input type="checkbox"
name="option2" value="english" checked>English
```

The **TYPE="SUBMIT"** creates a submit button on the form and the **TYPE="RESET"** creates a reset button. The Submit button tells the web browser to gather up all the selections, values, and entered text in the form elements and dispatch it off to the destination defined in the **ACTION** part of the **<FORM>** tag. The Reset button restores the form to its default state.

```
<INPUT type="submit" value="Send"><INPUT type="reset">
```

The submit or the reset button has a typical shape and size, which may not be in harmony with the rest of the look and feel of the web page. Hence you may prefer to use an image instead using the **TYPE="image"** attribute. When a user clicks an image button the

form is sent to the address specified in the action attribute of the <form> tag.

```
<p align="center">
  <INPUT TYPE="image" SRC="flower.jpg"
  NAME="imageclick"
  ALIGN=top HEIGHT=50 WIDTH=50>
  This image works like the submit button
</p>
```



This image works like the submit button

Alternately, the **TYPE= "BUTTON"** creates a click-able button with text the user specifies in the "value" attribute. When the button is clicked, it fires the **On Click** event. This event can be processed by, any scripting language such as VBScript or JavaScript to do some specific task. Of course if the event is not programmed nothing will happen on clicking the button. The **Submit** and **Reset** buttons, for example are special types of buttons that have been pre-programmed to do specific tasks.

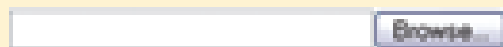
```
<INPUT TYPE="Button" VALUE="I am a simple button">
```



The **File** field allows users to upload files. The user can submit any information like picture, a spreadsheet, a scanned document or a word-processor document after verifying that the server on which the form will be processed supports file upload. The browser displays a "BROWSE" button next to a text box that lets the user select a field from their computer's storage to use as the value of the file input element. Typically, you would have encountered this option when attaching files to your email messages.

The output produced will be:

```
<INPUT TYPE ="FILE" NAME ="name"
SIZE ="30" MAXLENGTH ="200">
```



A hidden field is similar to text field with one major difference. The hidden field does not show on the page. Therefore the user cannot type anything in the hidden field. The **TYPE="hidden"** attribute is used to specify information for the receiving program or function that is not visible to the user and that the user may not change. This information is usually required by the program or function in order to process other information.

```
<INPUT TYPE="hidden" NAME="Language" VALUE="English">
```

The hidden field does not show, but still, when the form is submitted the hidden field is sent with it. In the above example the hidden field would tell the program that handles the form, that the users preferred language is English. Bilingual or multilingual sites, for example, can then switch to the correct language.

### The <SELECT> tag

The **<SELECT>** tag is used to construct drop-down list boxes (also called drop-down menus) and scrolling list boxes (also called scrolling menus). Drop-down menus can serve the same purpose as radio buttons (one selection only) or check boxes (multiple selections allowed), depending on the options specified. The advantage of a drop-down menu, compared to radio buttons or check boxes, is that it takes up less space. But that is also a disadvantage, because people can't see all options in the menu right away. With the **SIZE** attribute, the menu can be customised to show more than just one option at a time. Such lists are typically encountered when the user is to select a city or a country.

The **<OPTION>...</OPTION>** tag is used to specify each item displayed in the list. The value of the option last selected is returned when the form data is transmitted. Adding the **SELECTED** attribute in the **<OPTION>** tag indicates which element is displayed initially when the page loads. If this is not provided, the first item is selected by default.

```
<select name="abc"><option value="Math">Math
</option><option value="Science">Science
</option></select>
```



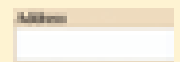
The above listing will display a drop down list since the size attribute is not specified:

### The <TEXTAREA> tag

Text areas are text fields that can span several lines. They are used for accepting multi-line input from the form user. Unlike most other form fields, text areas are not defined with an **<INPUT>** tag. Instead we enter a **<TEXTAREA>** tag where we want the text area to start and a closing **</TEXTAREA>** tag where we want the area to end. Everything written between these tags will be presented in the text area box. The scrollbars will be activated automatically, if the text input exceeds the visual number of columns or rows specified.

The above listing will display a text area with 2 rows and 20 columns as displayed in the following output:

```
Address<br><textarea name="as" rows="2"
cols="20"></textarea>
```



## 9.6 EXTENDING INTERACTIVITY – TRENDS IN WEB TECHNOLOGIES

We learnt about two kinds of interactivity in this chapter, hyperlinks and input forms. Web technologies have developed rapidly and today support a range of other features, unheard of a few years ago.

Web sites today support interactive forums, rich multimedia applications, and even radio stations. Web sites support commercial transactions like banking, bookings and reservations, and shopping. All such web features require large amounts of information to be stored and retrieved. They should also support different information for different users simultaneously.

The Indian Railways reservation system is a good example. Different people would request tickets for different places, by different trains, by different classes, on different dates. Simultaneously the system should make available information to each of these users, storing their names, ages, etc. and charge them accordingly. This charging is also done on the web by interlinking with banks. It is impossible to create different web pages for each possibility. What is required is a technology by which, based on the user's needs a new page is created by the server and sent to the user's browser. We refer to these as dynamic web pages.

### **9.6.1 DYNAMIC WEBSITES**

Web sites containing dynamic web pages are supported by databases. The web page itself contains empty tables which are populated based on the users request. Let us consider another common example, a search engine, say Google. You put in a search word or phrase and click on the search button. Within seconds, the web page is reconstructed with lists of web sites containing information you sought.

We refer to the search word you input as the keyword. Let us say, your keywords were "next solar eclipse in India". What you actually sought to know was "When will the next solar eclipse occur in India?". So, when you clicked on the search button, you actually submitted a request to the server to answer this question. We refer to your request as a query. You queried the server. The server looked up the databases for information matching your query and reported back with information relevant to your query.

Not only search engines, but a variety of different web applications use dynamic sites. Online libraries, dictionaries, encyclopedias, and book stores require dynamic features. E-mail providers, social networking sites, online courses, shopping malls, and tourist bureaus also require dynamic features. In fact, any site which seeks to provide constantly changing information, and different information for different users require the support of dynamic technologies. Programming technologies like active server pages (asp), java server pages (jsp) and hypertext preprocessor (php) are used to construct dynamic web sites. Such sites require a database to hold the information and a server to query, retrieve and serve the results of the query to the user.

### **9.6.2 SEPARATING CONTENT, STYLE AND CODE**

When you look at the html coding of our web page, you would have noticed that the coding, the design elements – like the tables, fonts



and colours, and the content of the web page are all put together. Not only did this make web pages slow to load, but also made it difficult to change. It required the graphic designer, who was mostly concerned with the look and feel of the web page, the subject expert who was interested in the content of the web page and the programmer, who handled the html code to work closely together. Each time a new page had to be made, all the three had to get together again.

Developing dynamic sites with thousands of different pages would have been very difficult to make and manage using such methods. An interesting technology has allowed us to separate the design, the content and the programming elements.

This technology, known as cascading style sheets allows the placing of all the design elements – colours, images, positioning, fonts, sizes and layouts to be placed in a separate file, the style sheet file. The content is placed in a text file or as records of a database. The html code along with other advanced code for supporting dynamic features is placed in the web page. The style sheet and the web page together function as a template, a container with predefined style elements. When the user calls for a page, the content, the style and the code are combined dynamically to generate a web page.

The use of cascading style sheets and databases now allows you to concentrate on content development. Once developed, the style sheets, the database and the web page code will remain fixed, unless you wish to modify it. New content gets populated in to the database, making it very easy for web site content to be renewed frequently.

## Summary

- One of the major applications of the Internet is the World Wide Web (www).
- The web can be thought of as an interconnected set of documents, images, audio, video or software files.
- The web uses the Hyper Text Markup Language to request and serve web pages and programs. Browsers are designed to read and interpret HTML.
- HTML consists of tags; the general format for a HTML tag is `<tag_name>affected text or image</tag_name>`
- Tags have different attributes and attributes take different values. Together, they extend the scope and functionality of HTML tags.
- You will need a HTML editor to write your code and a browser to test your web page; browsers differ in the way they interpret specific tags; it is therefore advisable to test your web page on different popular browsers.
- The basic structure of a web page is

```
<html>
  <head> </head>
  <body> </body>
</html>
```



- The code that goes into the `<head></head>` tag pair is not displayed by the browser; the title tag is an exception; it is displayed in the title bar of the browser window.
- All references to scripts, fonts, meta data are made in the head section.
- All elements of the web page visible in the browser window go into the `<body></body>` tag.
- Tags associated with size, colour, font are used to highlight text elements on a web page; You can also highlight text using bold, italic or underline.
- Text can be organised into paragraphs and paragraphs or lines of text can be separated from each other with the use of `<p></p>` and `<br>` tags.
- Lists can be prepared and presented using the ordered or unordered list tags; list items can be numbered in different ways using the type attribute.
- The `<table></table>` tags along with its associated tags and attributes is used in two ways; to separate different blocks of content and to present content in the form of tables.
- Tables consist of rows and cells; they can be structured and beautified using backgrounds, images, controlling their sizes and contents; tables can be nested within other tables.
- Images, audio and video files can be embedded in web pages. While browsers display images on their own, they require the support of media plugins to play back audio and video files.
- Hyperlinks are the most powerful feature of the web. Hyperlinks allow us to link two resources. The resources can be on the same or a different page, on the same storage or on a computer located half way across the world.
- Hyperlinks are achieved using the `<a href="source of resource">link</a>` tags.
- Forms can be used to submit information to, place a request on and retrieve information from the server. This extends the interactivity of web pages.
- Linking web pages to databases have enabled content on request, whereby servers respond to specific queries of users and dynamically generate web pages containing responses to the queries.

## EXERCISES

### SHORT ANSWER TYPE QUESTIONS

1. Identify and name the following:
  - (a) One physical tag and an equivalent logical tag used for making the text bold.
  - (b) One physical tag and an equivalent logical tag used for making the text appear in italics.
  - (c) A Windows accessory used as a text editor for creating HTML documents.
  - (d) Two possible extensions of an HTML file.
  - (e) A tag used for inserting section breaks.
  - (f) A tag used for displaying headings in a web page.
  - (g) Different tags necessary for creating tables.
  - (h) Tags used for creating framesets.
  - (i) Mandatory attribute of the `<A>` tag.
  - (j) An attribute of the `<A>` tag used to specify name of the frame in which the linked file is to open.
  - (k) Two attributes of the `<FRAME>` tag used to control the amount of white space between the frame borders and the contents of the frame.

2. Define the following terms:
  - (a) WWW
  - (b) Hypertext
  - (c) Markup
  - (d) Definition List
3. Differentiate between:
  - (a) Container and Non-container tags
  - (b) Text editors and WYSIWYG editors
  - (c) Logical and Physical tags
  - (d) <BR> and <P> tags
  - (e) Ordered List and Unordered List
  - (f) Internal and External Linking
  - (g) <A> as an anchor and <A> as a link
  - (h) Cellpadding and Cellspacing attribute of the Table tag
  - (i) Align and Valign attribute of <TD> tag
  - (j) <TD> and <TH> tag
  - (k) <TD> and <TR> tag
  - (l) <FRAMESET> and <FRAME> tags
  - (m) FRAMEBORDER and BORDER attribute of the <FRAMESET> tag
4. Correct the errors in the following HTML codes:
  - (a)
 

```
<HTML>
<T> Term Test </T>
<BODY>
<FONT SIZE=6>My Shopping List</FONT>
<BREAK>
<OL Number="A" START=D>
<LI>Shoes
<LI>Socks
</OL>
</BODY>
</HTML>
```
  - (b)
 

```
<HTML>
<TITLE><HEAD>Using Link</TITLE></HEAD>
<BODY>
<A Name="#top">Links</A>
<P>This page is an example of using links
using the anchor tag. </P>
<A Href="top">Goto top</A>
</BODY>
</HTML>
```
5. Identify the tag and attribute(s) used:
  - (a) To mark up text to retain its formatting along with blank spaces, carriage returns etc.
  - (b) For creating a list which uses alphabets as the numbering style and starts with the alphabet D.
  - (c) To insert an image as a background of a web page.
  - (d) To insert a line along the width of the web page to break up long sections of text that is 10 pixels thick.
  - (e) For linking within a web page.
  - (f) Create named frames for linking.
  - (g) Creating a link to a specific point named MARKS in a web page named Result.html stored in the same directory as your web page.
  - (h) For displaying "Welcome" as a tool tip text for an image with a 7 pixels thick border.

- (i) For creating an e-mail link to send a message to [ncert@gmail.com](mailto:ncert@gmail.com).
- (j) To display the cell contents at the bottom of the cell of a table.
- (k) To give a colour to the background of the entire table.
- (l) To specify the text in a cell of a table to appear in bold letters.
- (m) To place the Caption of the table at the bottom of it.

## LONG ANSWER TYPE QUESTIONS

1. Explain the role of HTML in creating web page.
2. Justify the statement: "HTML web pages are always saved as **text** files only".
3. Explain the structure of a HTML document with an example.
4. What is a comment? How do you add comments in a HTML document? Explain with the help of an example.
5. What are the usage of the <frameset> tag?
6. Explain the usage of the TARGET attribute of the <A> tag.
7. What is the significance of the SRC attribute of the <FRAME> tag?
8. Give the output of the following code:

a) <HTML>

```
<FRAMESET COLS=20%, *>
<FRAME NAME="F1" SRC="a.htm">
<FRAMESET ROWS=50%, *>
<FRAME NAME="F2" SRC=" b.htm">
<FRAME NAME="F3" SRC=" c.htm">
</FRAMESET></FRAMESET>
</HTML>
```

b) <HTML>

```
<BODY>
<TABLE CELLSPACING="4" CELLPADDING="6" ALIGN="CENTER" BORDER="3">
<CAPTION ALIGN="TOP"><CENTER><B>MARK LIST</B></CENTER></CAPTION>
<TR VALIGN="MIDDLE">
<TH>NAME</TH>
<TH>MARKS</TH>
<TH>PERCENTAGE</TH>
</TR>
<TR VALIGN="MIDDLE">
<TD>MEGHA</TD>
<TD>48/50</TD>
<TD VALIGN="BOTTOM">96%</TD>
</TR>
<TR ALIGN="MIDDLE">
<TD>TARUN</TD>
<TD>15/50</TD>
<TD>28%</TD>
</TR>
<TR ALIGN="CENTER">
<TD>VARUN</TD>
<TD ALIGN="RIGHT" VALIGN="TOP" BGCOLOR="YELLOW">42/50</TD>
<TD>84%</TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

## Activities

1. Write HTML code to display the names of any 5 subjects on different lines (use <BR> tags). Change the <BR> tags to <P> tags and notice the change. Next try to display the names with a number in front of them. Finally, change the numbers to bullets.
2. Write HTML code to illustrate the usage of nesting of tags. Change the nesting order and see if it makes any difference in the output.
3. Write the HTML code for the following web page:

### List of Subjects

1. SCIENCE
  - (a) BIOLOGY
    - (i) BOTANY
    - (ii) ZOOLOGY
  - (b) CHEMISTRY
    - (i) ORGANIC
    - (ii) INORGANIC
    - (iii) PHYSICAL
  - (c) PHYSICS
2. SOCIAL STUDIES
  - (a) GEOGRAPHY
  - (b) CIVICS
  - (c) HISTORY
  - (d) ECONOMICS

4. Play around with the horizontal lines by typing the following code in a notepad file and viewing the output. Try out other variations by changing the size, align and color options.

```
<HTML>
<TITLE> Home Page </TITLE>
<BODY BACKGROUND="bg.gif">
<!-- A website created bout famous Cartoon Characters-->
What is a Cartoon?<BR>
<HR WIDTH = 50% ALIGN=LEFT SIZE=5 NOSHADE >
<HR SIZE = 8 COLOR = "blue">
<HR WIDTH = 50% ALIGN=RIGHT SIZE=5 COLOR="RED">
<P>A cartoon is any of several forms of illustrations
with varied meanings.
</BODY>
</HTML>
```

5. Write the HTML code to design the following web page:

### States of Matter

**There are three states of matter:**

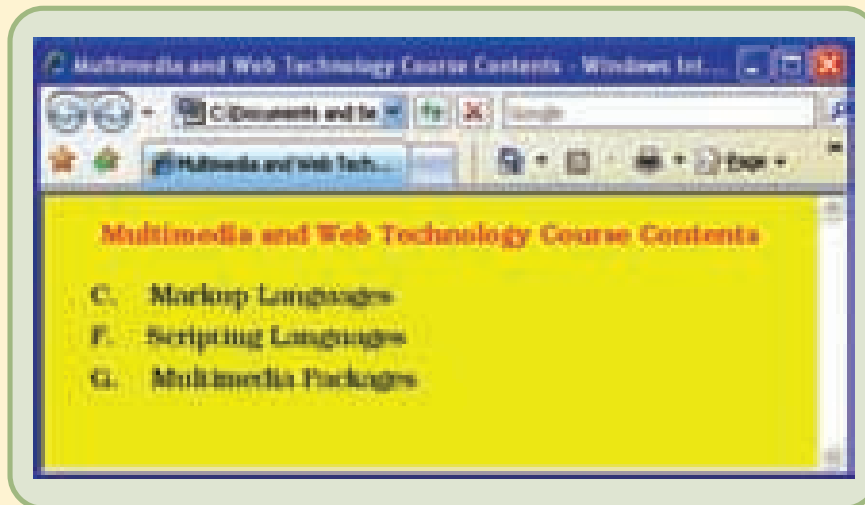
Solids – Example: Pencil

Liquids – Example: Water

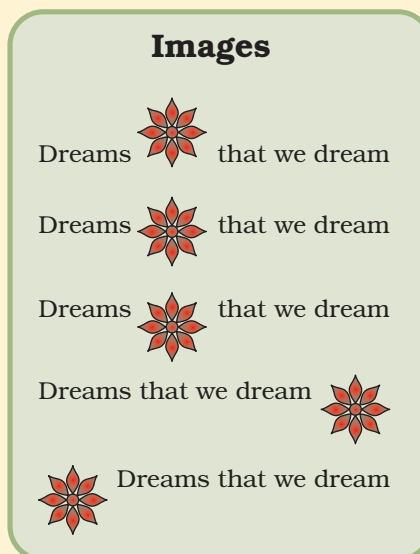
Gases – Example: Air

Gases are lighter than solids and liquids.

6. Write the HTML code to design a website (at least two pages) about your favourite Pastime with the following properties:
  - (i) Each page should have appropriate titles.
  - (ii) The headings should scroll on the page.
  - (iii) Insert an appropriate image as a background of the web pages.
  - (iv) The heading of the page should be in RED colour and style "Arial". It should be underlined and right aligned.
  - (v) The first page should have the information in paragraphs each of which is in different colour and alignment.
  - (vi) The second page should contain a list of the famous tourist spots in that area.
7. Write the HTML code to produce the output shown below with the specified numbering.



8. Write the HTML code to generate the output shown below (assume that the name of the file is flower.jpg).

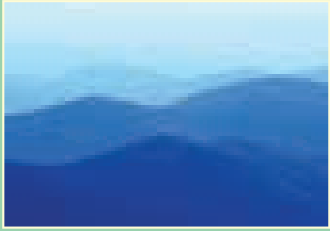
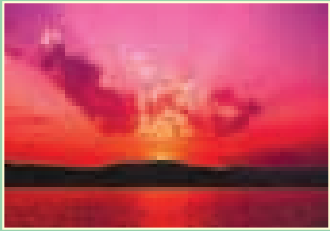



9. Your best friend has created a web page with the url: [www.tipod.com/personal/cooldude.html](http://www.tipod.com/personal/cooldude.html). You now want to link to this web page, when you click on the photograph of your friend which you have placed on your web page. Assume that the name of file containing your friend's photograph is "c:\mypictures\myfriend.jpg".

10. Write the HTML code to generate the following output:

| Average |         |               |
|---------|---------|---------------|
| Class   | Section | Average Marks |
| XII     | A       | 94.5%         |
| XI      | B       | 91.8%         |

11. Write the HTML code to generate the following output (assume the names of the images as hill.jpg, sea.jpg and water.jpg respectively).

| S No | Place         | Image  |
|------|---------------|--|
| 1.   | Hill Stations |  |
| 2.   | Sea Shores    |  |
| 3.   | Back- waters  |  |

12. Your school's web address is [www.littlebuds.com](http://www.littlebuds.com). You want to create a link to this website from your web page. Write the HTML code to do this?

13. Write the HTML code to generate the following page:

### ***My School Library***

My school has a big library with lots of books.

Click on any of the categories to see the list of books available :

1. Fiction
  - Mystery
  - Comedy
  - Drama
2. Reference Books
  - Science
  - Maths
  - Social Science
3. Languages
  - Hindi
  - English
  - Sanskrit

**Note:** Pages linked to (Create the links, assuming that the files already exist):

- Fiction as “fict.html”
- Reference Books as “ref.html”
- Language Books as “lang.html”

14. Write the HTML code to create the following framesets:

(a)

|            |             |
|------------|-------------|
|            | Second.html |
| First.html | Third.html  |
|            | Fourth.html |

(b)

|        |        |        |
|--------|--------|--------|
| A.html |        |        |
| B.html | C.html |        |
|        | D.html | E.html |

(c)

|        |        |        |
|--------|--------|--------|
| A.html | D.html | C.html |
| B.html |        | E.html |